PENGINDEKSAN DATA SPASIAL MENGGUNAKAN STRUKTUR DATA R*-TREE.

SKRIPSI

Diajukan untuk memenuhi persyaratan
meperoleh gelar Sarjana Komputer

Disusun Oleh :

NANANG AKHMAD CHOIRUL ANAM
NIM. 0710963048

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
PROGRAM STUDI TEKNIK INFORMATIKA
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2012

LEMBAR PERSETUJUAN

PENGINDEKSAN DATA SPASIAL MENGGUNAKAN STRUKTUR DATA R*-TREE.

SKRIPSI

Diajukan untuk memenuhi persyaratan
meperoleh gelar Sarjana Komputer

Disusun Oleh :

NANANG AKHMAD CHOIRUL ANAM
NIM. 0710963048

Telah diperiksa dan disetujui oleh :

Dosen Pembimbing I	Dosen Pembimbing II
Candra Dewi, S.Kom, M.Sc.	**Drs. Marji, MT.**
NIP. 197711142003122001	**NIP. 196708011992031001**

LEMBAR PENGESAHAN

PENGINDEKSAN DATA SPASIAL MENGGUNAKAN STRUKTUR DATA R*-TREE.

SKRIPSI

KONSENTRASI REKAYASA PERANGKAT LUNAK

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :

NANANG AKHMAD CHOIRUL ANAM
NIM. 0710963048

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 24 Oktober 2012

Penguji I	Penguji II
Dian Eka Ratnawati, S.Si., M.Kom.	**Dewi Yanti Liliana, S.Kom., M.Kom.**
NIP. 197306192002122001	**NIP. 198111162005012004**

Penguji III

Ahmad Afif Supianto, S.Si., M.Kom.

Mengetahui,
Ketua Program Studi Teknik Informatika

Drs. Marji, MT.
NIP. 196708011992031001

PERNYATAAN
ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata didalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku. (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 12 Juni 2012

Mahasiswa,

Nanang Akhmad Choirul Anam
NIM. 0710963048

ABSTRAK

Nanang Akhmad Choirul Anam. 2012. Pengindeksan Data Spasial Menggunakan Struktur Data R*-Tree. Skripsi Program Studi Teknik Informatika, Program Teknologi Informasi Dan Ilmu Komputer, Universitas Brawijaya.

Dosen Pembimbing : Candra Dewi, S.Kom, M.Sc. dan Drs. Marji, MT.

Sistem informasi geografis adalah salah satu penggunaan teknologi informasi untuk mengolah peta dalam bentuk digital, sehingga memudahkan peta tersebut dimanipulasi dan diolah datanya, salah satu penyimpanan data digital dalam bentuk data spasial adalah menggunakan struktur data R*-tree.

Strukturdata R*-tree dapat digunakan untuk pencarian sebuah area tertentu yang di cakupi oleh Minimum Bounding Rectangle (MBR), kriteria MBR dalam struktur data R*-tree (i) meminimalkan daerah yang di cakup oleh masing-masing MBR (ii) meminimalkan tumpang tindih antara MBR (iii) meminimalkan margin MBR (iv) memaksimalkan penggunaan penyimpanan.

Hasil dari penelitian ini adalah (i) data spasial dapat diimplementasikan dengan menggunakan strukturdata R*-tree (ii) pengindeksan dan query pada struktur data R*-tree dapat digunakan pada data spasial bertipe polygon dan polyline, serta waktu yang dibutuhkan dalam proses pengindeksan maupun query pada data spasial di pengaruhi oleh jumlah data dan proses reintegrasi pada struktur data R*-tree yang membutuhkan optimasi dan minimasi.

Kata kunci : SIG, R*-Tree, Spatial Data, Minimum Bounding Rectangle (MBR), Runtime Query, Indeksing.

ABSTRACT

Nanang Akhmad Choirul Anam. 2012. ***Indexing Spatial Data Using R*-Tree Data Structures. Final Exam Informatic Engineering, Program Of Information Technology And Computer Science, Brawijaya University.***
Advisor : **Candra Dewi, S.Kom, M.Sc. dan Drs. Marji, MT.**

Geographic information system is one of the use of information technology to process map in digital form, making it easier for the maps to manipulated and processed data, one of the storage of digital data in the spatial data form is used R*-tree data structures.

R*-tree data structures can be used to searching a specific area on the Minimum Bounding Rectangle (MBR), MBR criteria in R*-tree data structures (i) minimize the area in cover by each MBR (ii) minimize the overlap between the MBR (iii) minimize the margin of MBR (iv) maximize use of the storage.

The results of this research were (i) the spatial data can be implemented using the R*-tree data structures(ii) indexing and query R*-tree data structures can be used in the spatial data polygon and polyline type, and the time required in the process of indexing and queries on spatial data is influenced by the amount of data and the reintegration of the data structures R*-tree that require optimization and minimization.

Keywords : SIG, R*-Tree, Data Spatial, Minimum Bounding Rectangle (MBR), Query Runtime, indexing.

KATA PENGANTAR

Puji syukur kehadirat Allah SWT dan sholawat serta salam selalu pada beliau Rosulullah SAW, hanya dengan rahmat, ridlo dan syafaat serta karunia yang telah diberikan kepada penulis, sehingga penulis dapat menyelesaikan skripsi dengan judul "Pengindeksan Data Spasial Menggunakan Struktur Data R*-Tree".

Skripsi ini merupakan salah satu syarat untuk memenuhi persyaratan akademis dalam menyelesaikan studi di program Sarjana Ilmu Komputer Universitas Brawijaya. Pada kesempatan ini penulis mengucapkan banyak terima kasih atas segala bantuan dan dedikasi moral maupun material dan semoga Allah SWT membalas-Nya dengan kenikmatan yang lebih dan selalu bertambah kepada beliau yang telah banyak membantu dalam rangka penyusunan skripsi ini.

1. Candra Dewi, S.Kom, M.Sc dan Drs. Marji, MT selaku dosen pembimbing yang telah membimbing dengan bijaksana dan sabar dalam penyusunan skripsi ini.
2. Segenap Bapak dan Ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada penulis selama menempuh pendidikan di Program Studi Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Brawijaya.
3. Segenap keluarga besar Program Studi Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam dan Program Teknologi Informasi Dan Ilmu Komputer Universitas Brawijaya, yang tidak bisa penulis sebut satu persatu yang telah banyak membantu demi terselesainya penyusunan skripsi ini.

4. Ayah, Ibu, dan saudara-saudara tercinta, serta keluarga besar yang tersayang, terima kasih atas dukungan dan doanya.
5. Yudi Ariesta, Supardi, Marissa Hamnon dan teman-teman Prodi Ilmu Komputer serta teman-teman lain yang selalu memberikan dukungan dan doanya

Penulis menyadari bahwa masih banyak kekurangan dalam penulisan laporan ini yang disebabkan oleh keterbatasan kemampuan dan pengalaman. Oleh karena itu, penulis sangat menghargai saran dan kritik dari pembaca demi pengembangan dan menambah hasanah keilmuan bagi penulis.

Penulis berharap semoga laporan ini dapat memberikan manfaat kepada pembaca dan bisa diambil manfaatnya untuk pengembangan selanjutnya.

Malang,

Penulis

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

DAFTAR SOURCECODE

Halaman

BAB I
PENDAHULUAN

1.1 Latar Belakang

Sistem Informasi Geografis (SIG) atau Geographic Information System (GIS) Menurut Puntodewo, dkk. (2003) adalah "suatu komponen yang terdiri dari perangkat keras, perangkat lunak, data geografis dan sumberdaya manusia yang bekerja bersama secara efektif untuk menangkap, menyimpan, memperbaiki, memperbaharui, mengelola, memanipulasi, mengintegrasikan, menganalisa, dan menampilkan data dalam suatu informasi berbasis geografis". Dilihat dari definisinya, SIG merupakan suatu sistem informasi yang didesain untuk bekerja dengan data yang direferensikan oleh spasial atau koordinat geografi yang terkomputerisasi. Dengan kata lain, SIG adalah sistem informasi berbasis komputer yang digunakan untuk mengolah dan menyimpan data atau informasi geografis (Hermawan, 2009).

Penyimpanan data spasial tidak seperti halnya data-data biasa yang di simpan dalam *database* melainkan membutuhkan proses pengindeksan yang khusus karena berhubungan dengan data ruang dua dimensi dan juga menggunakan struktur data tertentu(Yeung, dkk. 2007).

Kebutuhan penyimpanan data yang sangat besar dengan menyimpan informasi keruangan yang kompleks maka dibutuhkan struktur data yang mampu menangani pengaksesan data spasial, secara umum di dalam struktur data pengaksesan metode spasial yang ada dan telah banyak di gunakan untuk melakukan pengindeksan banyak memerlukan waktu dan harga yang sangat mahal

untuk bisa memberikan informasi yang cepat dalam meload bentuk rupa bumi ke dalam sebuah grafis informasi(Popescu,2003).

Pengindeksan spasial adalah mekanisme untuk memfasilitasi akses ke database spasial melalui koordinat yang disimpan dalam ruang dua dimensi, ada berbagai pilihan untuk pengindeksan, seperti R-tree, B-tree dan quadtree, masing-masing memiliki kelebihan dan kelemahan tergantung pada kebutuhan format data dan aplikasi masing-masing (Yeung, dkk, 2007).

R*-tree adalah salah satu struktur data yang bisa menangani pengindeksan dan pengaksesan dari data spasial, karena menurut Beckmann, dkk.(1990), R*-tree merupakan pengembangan dari versi sebelumnya yaitu B-tree dan R-tree.

R*-tree merupakan hirarki struktur data dari sekumpulan obyek persegi panjang dengan menyimpan data informasi geometri spasial ke dalam batasan kotak-kotak persegi yang disebut dengan Minimum Bounding Rectangle(MBR) yang disusun dalam node *tree* (Popescu,2003).

Minimum Bounding Rectangles (MBR) adalah struktur yang sangat efisien yang secara luas diterapkan dalam GIS, pada dasarnya MBR mendefinisikan persegi pembatas minimal yang sisi-sisinya sejajar dengan sumbu x dan y pada sistem koordinat yang membungkus satu set satu atau lebih pada obyek geografis, hal ini di definisikan oleh dua koordinat di kiri bawah (x minimal, y minimal) dan kanan atas (x maksimal, y maksimal) (Longley,dkk.2005).

Perkembangan struktur data R*-tree memperbaiki dari versi sebelumnya dimana versi sebelumnya R-tree hanya melakukan optimasi pada area lokal MBR sedangkan R*-tree mengembangkan dengan banyak optimasi yang di antaranya

memperkecil area mati, memperkecil tumpang tindih antara MBR, meminimalisasi margin beberapa MBR dan memaksimalkan ruang penyimpanan (Manolopoulos, dkk. 2006).

Proses optimasi dan minimasi pada struktur data R*-tree di lakukan pada saat reintegrasi dengan memasukkannya *node* kembali mencapai seperti penyeimbangan tree yang secara signifikan meningkatkan kinerja selama pemrosesan *query* (Manolopoulos, dkk. 2006).

Himpunan prosentrase *entry* yang akan dimasukkan kembali adalah dimana jarak pusat dari pusat *node* adalah (p) di antara 30% terbesar, heuristik ini adalah untuk mendeteksi dan membuang *entry* terjauh yang berdasarkan dari hasil terbaik eksperimen N. Beckmann, dkk (1990). (Manolopoulos, dkk.2006).

Berdasarkan kebutuhan pengaksesan data spasial terkini, optimasi dan pencarian cepat didalam penggunaan data spasial pada GIS begitu sangat di butuhkan, dengan kebutuhan ini penulis ingin mencoba menggunakan struktur data R*-tree ke sebuah sistem untuk pengindeksan data spasial,

Berdasarkan kriteria optimasi dan minimasi yang ada dalam struktur data R*-tree menurut Manolopoulos, dkk,(2006) kriteria ini dapat meningkatkan kinerja selama pemrosesan *query* dan struktur MBR yang digunakan oleh struktur data R*-tree, dan juga menurut Karen & Kemp (2008) Obyek-obyek MBR yang tersusun di dalamnya tersimpan dalam ruang dengan sistem berbasis kedekatan yang memungkinkan pengaksesan data spasial sangat cepat.

1.2 Rumusan Masalah

Permasalahan yang akan dijadikan obyek penelitian pada skripsi ini adalah

1. Bagaimana mengimplementasikan struktur data R*-tree untuk pengindeksan data spasial.
2. Bagaimana waktu yang dihasilkan dari pengindeksan data spasial serta *query* terhadap prosentase *reinsert* untuk *reintegrasi* pada struktur data R*-tree.

1.3 Batasan Masalah

Batasan masalah dalam penulisan skripsi ini adalah :

1. Pengindeksan data spasial hanya sampai pada pembentukan node-node struktur data R*-tree.
2. Pada skripsi ini tidak dilakukan perbandingan struktur data R*-tree dengan strukturdata lain.
3. Kriteria runtime yang di gunakan dalam analisa adalah waktu yang dihasilkan dari pengindeksan dan *runtime query* setelah dilkukan optimasi dan minimasi dalam pengindeksan.
4. Metode *query* spasial yang digunakan hanya pada *range* pencarian wilayah feature geom yang di cakupi MBR.
5. Tipe data spasial geometri yang di gunakan dalam pengindeksan hanya geometri-geometri hasil konversi dari data vektor polygon dan polyline.

1.4 Tujuan

Tujuan dari penulisan skripsi ini adalah

1. Menghasilkan perangkat lunak pengindeksan data spasial dengan menggunakan struktur data R*-tree.

2. Mengetahui efesiensi waktu yang dihasilkan dari pengindeksan data spasial serta *query* terhadap prosentase reintgrasi pada struktur data R*-tree.

1.5 Manfaat

Manfaat yang diperoleh dari penulisan skripsi ini adalah memberikan gambaran efisiensi kinerja struktur data R*-tree yang dapat digunakan untuk pengindeksan serta *query* pada data spasial yang digunakan dalam perangkat sistem informasi geografi.

1.6 Metodologi Penelitian

Metodologi yang digunakan dalam penulisan skripsi ini adalah sebagai berikut:

1. Studi Literatur

   Mempelajari metode yang digunakan untuk pengindeksan data spasial menggunakan struktur data R*-tree

2. Pendefinisian Dan Analisis Masalah

   Mendefinisikan dan menganalisis masalah untuk memperoleh solusi yang tepat.

3. Perancangan Dan Implementasi Sistem

   Merancang perangkat lunak yang akan dibuat, kemudian mengimplementasikan rancangan tersebut.

4. Uji Coba Dan Analisis Hasil Implementasi

   Menguji coba perangkat lunak yang telah dibuat kemudian dianalisa hasilnya, apakah sudah sesuai dengan tujuan yang telah dirumuskan sebelumnya,

kemudian dievaluasi dan dilakukan perbaikan hingga sesuai dengan tujuan yang ingin dicapai.

1.7 Sistematika Penulisan

Skripsi ini disusun berdasarkan sistematika penulisan sebagai berikut:

1. BAB I PENDAHULUAN

   Berisi latar belakang masalah, rumusan masalah, batasan masalah, tujuan penulisan, manfaat penulisan, metodologi penelitian, sistematika penulisan.

2. BAB II TINJAUAN PUSTAKA

   Berisi teori-teori dasar yang menunjang pengindeksan data spasial seperti pengertian MBR, tipe data spasial sampai pada pengindeksan data spasial dengan menggunakan struktur data R*-tree

3. BAB III METODOLOGI DAN PERANCANGAN

   Berisi metode yang digunakan dalam penelitian dan langkah – langkah yang harus dilakukan. Beberapa hal yang dibahas adalah: subyek penelitian, perangkat lunak dan perangkat keras, gambaran sistem dan rancangan penelitian.

4. BAB IV HASIL DAN PEMBAHASAN

   Bab ini berisi penjelasan implementasi dan rancangan yang telah diuraikan pada Bab III dan hasil pengujian yang dilakukan. Implementasi yang dijelaskan berupa implementasi program. Selain itu bab ini juga menjelaskan penerapan aplikasi, analisis dari hasil uji coba berupa tingkat kecepatan waktu yang diperlukan didapatkan dari pengindeksan dan query, analisis perbandingan

hasil dari waktu yang dibutuhkan selama pengindeksan dan juga waktu hasil query dari program struktur data R*-tree dengan data record yang bervariasi

5. BAB V KESIMPULAN DAN SARAN

   Bab ini berisi kesimpulan yang diperoleh dari hasil pembahasan dan saran yang diharapkan bermanfaat untuk pengembangan penelitian yang lebih lanjut.

BAB II
TINJAUAN PUSTAKA

2.1 Sistem Informasi Geografi

2.1.1 Pendahuluan

Sistem Informasi Geografi (SIG) atau yang juga dikenal dengan Sistem Informasi keruangan adalah sistem informasi berbasis komputer yang digunakan untuk mengumpulkan, menyimpan, menggabungkan, mengatur, mentransformasi, memanipulasi dan menganalisis data yang *bergeoreferensi*. (Hermawan, 2009).

SIG akan menghasilkan informasi yang lebih berkualitas dan informasi yang disajikan tersebut dapat dengan mudah dipahami oleh pengguna, sesuai dengan perkembangan ilmu pengetahuan dan teknologi terkini pembuatan peta pun bisa dilakukan tidak dengan cara konvensional(sederhana), melainkan sudah dikembangkan dengan menggunakan komputer sehingga menjadi lebih mudah dan cepat (Hermawan, 2009).

GIS adalah sebuah teknologi yang telah terbukti dan operasi dasar GIS saat ini menyediakan landasan yang aman dan didirikannya GIS ini untuk dapat mengukur, memetakkan, dan menganalisis dari dunia nyata. (Longley, dkk, 2005)

SIG merupakan sistem yang dapat mendukung pengambilan keputusan dan mampu mengintegrasikan deskripsi-deskripsi lokasi dengan karakteristik fenomena gejala geografis yang ditemukan di lokasi tersebut(Hermawan,2009).

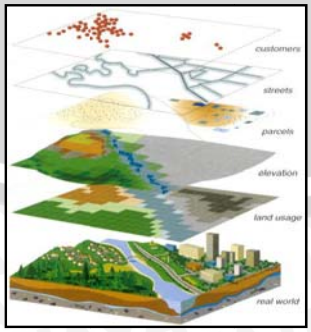
Setiap sistem GIS hendaknya harus mampu memberikan informasi tentang fenomena geospasial yang pada prinsipnya tugas atau fungsi ini adalah untuk: 1) menangkap masukan atau input, 2) penataan atau memanajemen, 3) manipulasi, 4)

analisis, dan 5) presentasi (Raper & Maguire, 1992 dalam Rahman & Pilouk 2008).

SIG menyajikan dunia nyata *(real world)* pada layar komputer seperti lembaran peta, semua informasi unsur geografis disimpan ke basis data dengan penyimpanan dalam bentuk tabel, data ini dapat dipanggil dan dicari berdasarkan segala atribut yang dimilikinya *(storage and retrieval)*, selain itu SIG juga menyediakan data untuk berbagai kegunaan diberbagai bidang keilmuan (Hermawan,2009).

Melihat definisi diatas Teknologi *SIG* tidak hanya sebatas sistem komputer melainkan melibatkan beberapa subsistem untuk menggambar dan menyimpan area geografis kesebuah tampilan peta, tetapi juga mampu menyimpan data yang dapat digunakan untuk menggambar atau menampilkan suatu informasi dengan sistem berbasis computer

SiG juga digunakan untuk membantu manusia dalam memahami dunia nyata dengan melakukan proses-proses manipulasi dan presentasi data yang direalisasikan dengan lokasi-lokasi geografis dari permukaan bumi, sebagai contohnya pada **Gambar 2.1** dimana layer SIG yang mempresentasikan informasi dari unsur pelanggan dengan unsur jalan dari dunia nyata.



Gambar 2.1 Representasi SIG terhadap dunia nyata.

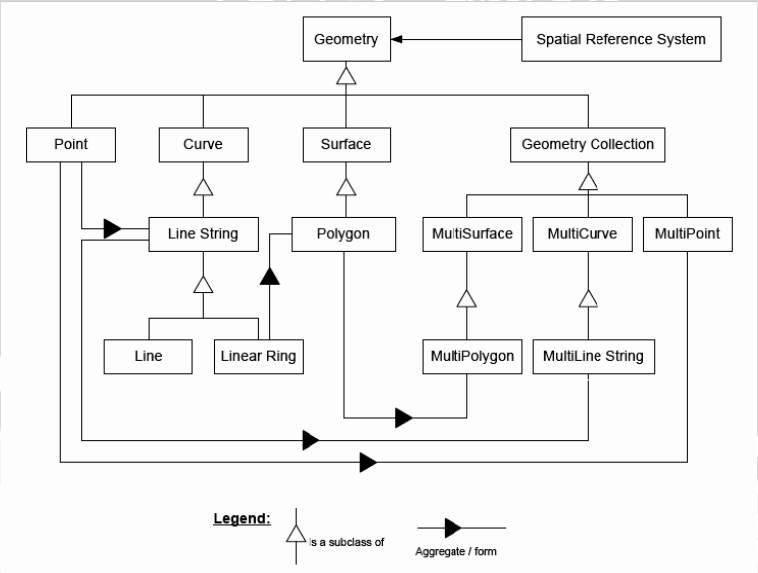
2.2 Data GeoSpasial

2.2.1 Geometry Spasial

Bidang geometri umumnya dipahami sebagai sebuah cabang matematika yang berhubungan dengan titik, garis, sudut, permukaan dan ruang dua-dimensi seperti ditunjukan pada **Gambar 2.2.**

Dalam konteks pemrosesan data spasial, geometri bermakna baru ketika Open Geospatial Consortium (OGC) telah diresmikan penggunaannya dalam publikasi fungsi spesifikasi OpenGIS Simple SQL (OGC, 1999), dalam tulisannya, OGC mengusulkan hirarki tipe data spasial, yang disebut model obyek geometri, yang memungkinkan fungsi-fungsi spasial di representasikan dalam database

Dalam model obyek geometri, kata "geometri" digunakan untuk mewakili fungsi spasial sebagai "obyek" untuk memiliki setidaknya satu atribut dari tipe geometri dalam database(Yeung, dkk, 2007).



Gambar 2.2 The OGC geometry object model(Yeung.2007).

Obyek hirarki model sebagai kelas akar dari geometri adalah bangunan *non-instantiable* (yaitu tidak berisi kejadian atau contoh karakteristik dunia nyata).

Kelas dasar geometri ini memiliki empat subkategori, yaitu item(point), koleksi geometri, kurva dan permukaan (*surface*), sedangkan subkategori konstruksi geometri *instantiable* (yaitu, kejadian yang mengandung contoh fitur dunia nyata) dan berisi satu set proses akhir yang disebut metode digunakan untuk menguji sifat-sifat geometri.

Masing-masing geometri mendefinisikan hubungan spasialnya dan mendukung penggunaannya dalam analisis spasial (Yeung.2007).

2.2.2 Data spasial

Data spasial mempunyai dua bagian penting yang membedakan dari data lain, yaitu informasi lokasi dan informasi atribut (Puntodewo, dkk,2003)

Data spasial adalah data yang dapat ditampilkan, dimanipulasi dan dianalisis dengan cara atribut spasial yang menunjukkan lokasi yang sama dengan permukaan bumi, atribut ini biasanya disediakan dalam bentuk koordinat spasial yaitu pasangan posisi dan bentuk dari fitur spasial yang memungkinkan untuk dapat diukur dan disajikan secara grafik Yeung (2007), data spasial memiliki dua sifat penting :

- Informasi lokasi atau informasi spasial.

  Contoh yang umum adalah informasi lintang dan bujur yang termasuk diantaranya informasi datum dan proyeksi, contoh lain dari informasi spasial adalah untuk mengidentifikasikan lokasi misalnya adalah kode pos. Menurut Yeung (2007), spasial geografis berarti data yang terdaftar pada sistem

koordinat ini mencakupi beberapa area permukaan tanah sehingga data dari sumber yang berbeda dapat saling terintegrasi dan direferensikan spasial yang penyimpanannya dalam bentuk koordinat x,y (vektor) atau dalam bentuk image (raster) yang memiliki nilai tertentu.

- Informasi deskriptif (atribut) atau informasi non spasial.

  Suatu area bisa memiliki beberapa atribut atau properti yang memiliki informasi saling berkaitan contohnya jenis vegetasi, populasi dan pendapatan pertahun. Menurut Yeung (2007), atribut terwakili dalam berbagai data spasial geografis ketika mendaftar pada skala relatif kecil di dalam atau di permukaan bumi harus umum dan disimbolkan untuk mewakili daerah yang besar.

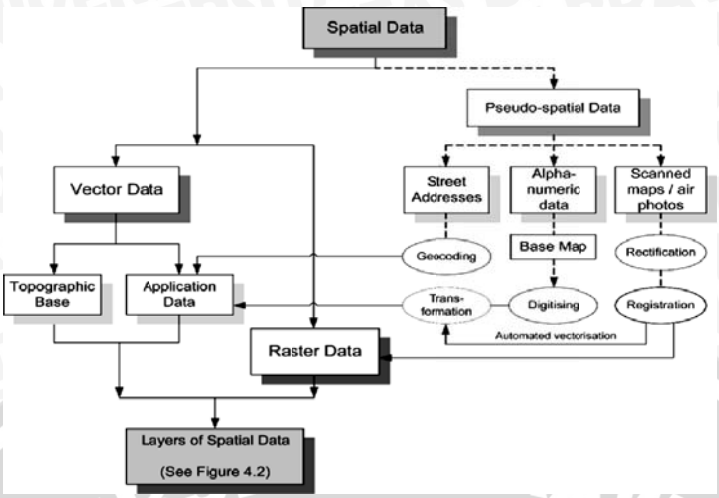
2.2.3 Tipe Data Spasial

Data spasial dikumpulkan dan disimpan dalam dua bentuk tipe data dasar yang disebut vektor dan raster (Yeung, dkk, 2007).

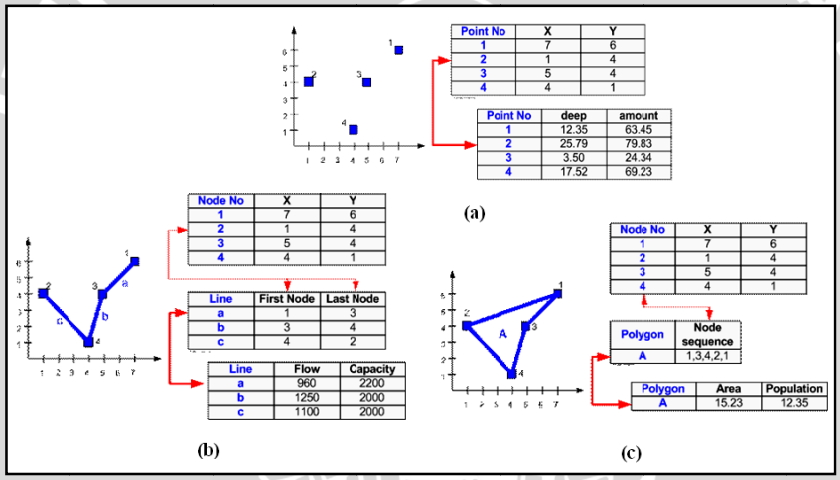
2.2.3.1 Tipe Data Vektor

Tipe data 12ector dalam database spasial yang di tunjukkan pada **Gambar 2.3** data disimpan sebagai bagian dari topografi dasar yang bereferensikan spasial dengan fungsi untuk menyediakan kerangka kerja pada saat pengumpulan data dan analisis.

Fitur data dalam suatu set atau lebih yang biasanya ditemukan pada sebuah peta topografi konvensional seperti jalan, sungai, daerah perkotaan dan karakteristik dari vegetasi alami mencakupi beberapa poin-poin dasar seperti topografi dan 12ector12 12ector geodesi (Yeung,2007).



Gambar 2.3 Tipe data spasial(Yeung, dkk, 2007).



Gambar 2.4 Struktur Data Vektor, a) Point, b) Line, c) Polygon.

Bentuk data 13ector yang di ilustrasikan pada **Gambar 2.4,** bumi yang direpresentasikan sebagai suatu 13ector dari garis(arc/line), polygon dan point, polygon adalah daerah yang dibatasi oleh garis yang berawal dan berakhir pada titik yang sama sedangkan titik atau *point* yaitu *node* yang mempunyai label dengan titik perpotongan antara dua buah garis (Puntodewo, dkk, 2003).

Keuntungan utama dari format data 13ector menurut Puntodewo, dkk, (2003) adalah ketepatan dalam merepresentasikan fitur titik, batasan dan garis

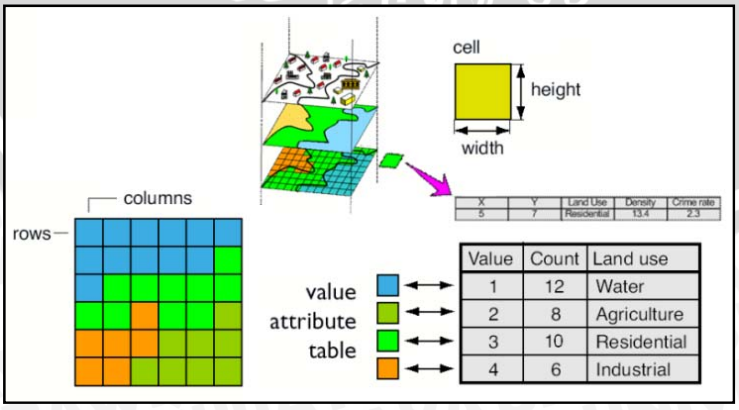
lurus, hal ini sangat berguna untuk analisa yang membutuhkan ketepatan posisi, misalnya batas-batas kadaster pada basisdata.

2.2.3.2 Tipe Data Raster

Data raster atau disebut juga dengan sel grid seperti di ilustrasikan pada **Gambar 2.5** adalah data yang dihasilkan dari sistem penginderaan jauh, obyek geografis pada data raster direpresentasikan sebagai struktur sel grid yang disebut dengan pixel (*picture element*) dan resolusi (definisi visual) tergantung pada ukuran pixel-nya.

Ukuran pixel pada resolusi menggambarkan ukuran sebenarnya di permukaan bumi yang diwakili oleh setiap pixel pada citra, semakin kecil ukuran permukaan bumi yang di representasikan oleh satu sel maka semakin tinggi resolusinya.

Data raster sangat baik untuk merepresentasikan batas-batas yang berubah secara gradual, seperti jenis tanah, kelembaban tanah, vegetasi dan suhu tanah. Keterbatasan utama dari data raster adalah besarnya ukuran file, sehingga semakin tinggi resolusi *grid*-nya semakin besar pula ukuran filenya. (Puntodewo, dkk, 2003).



Gambar 2.5 Struktur Data Raster

2.3 Pengindeksan Dan Pengaksesan Data Spasial

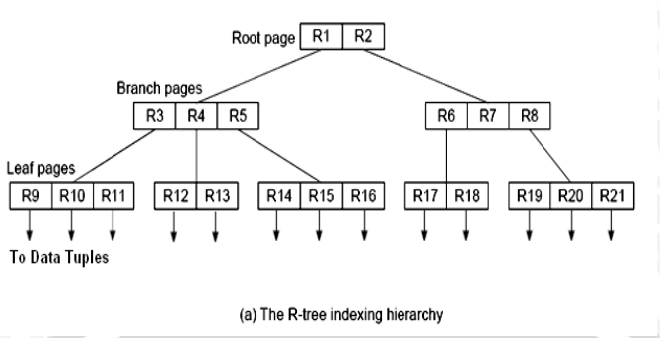
Pengindeksan spasial yaitu bertujuan untuk mempercepat akses data ke pengguna dari database, pengindeksan spasial jauh lebih rumit daripada pengindeksan berbasis teks dalam tabel karena berhubungan dengan ruang dua-dimensi (Yeung, 2007).

Index adalah koleksi *entry* data dengan nilai kunci pencarian *k*, dimana setiap *entry* yang dilambangkan *k** mengandung informasi yang cukup untuk bisa mendapatkan satu atau lebih data record dengan nilai kunci kembalian *k, index* diciptakan satu atau lebih dalam kolom table yang memiliki dasar baik untuk *rapid random look up* dan akses yang efisien untuk record berurutan, ruang hardisk penyimpanan *index* biasaynya lebih kecil dari pada yang dibutuhkan tabel(Handayani,2001).

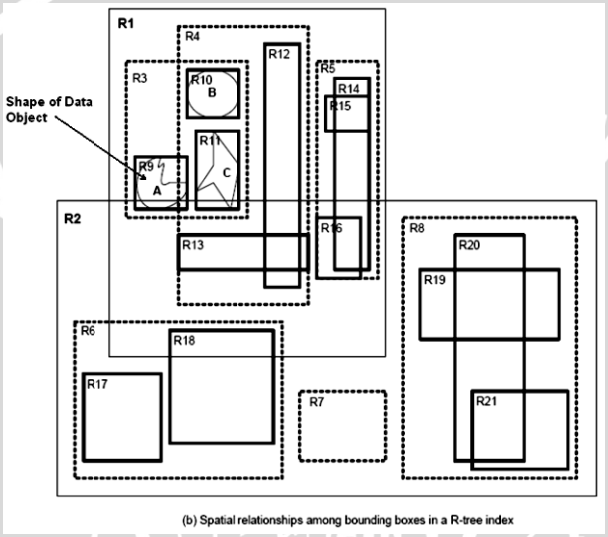
Pengindeksan Spasial adalah mekanisme untuk memfasilitasi akses ke database spasial melalui koordinat yang disimpan dalam ruang dua dimensi, ada berbagai pilihan untuk pengindeksan, seperti R-tree, B-tree dan quadtree, masing-masing memiliki kelebihan dan kelemahan tergantung pada kebutuhan format data dan aplikasi masing-masing (Yeung, dkk, 2007).

Konsep dasar pengindeksan dari pendekatan spasial adalah dimana proses pengindeksan dalam penggunaan akses spasial secara bertahap dengan mempersempit daerah pencarian hingga ditemukannya obyek database yang diperlukan.

Salah satu metode yang lebih umum diadopsi pengindeksan spasial yang banyak digunakan untuk 15roper database spasial adalah R-tree dimana R adalah singkatan dari “region” seperti diungkapkan oleh Brinkhoff et al, (1994). Struktur *index* R-tree seperti ditunjukkan pada **Gambar 2.6**.(Yeung, 2007).



(a) The R-tree indexing hierarchy



(b) Spatial relationships among bounding boxes in a R-tree index

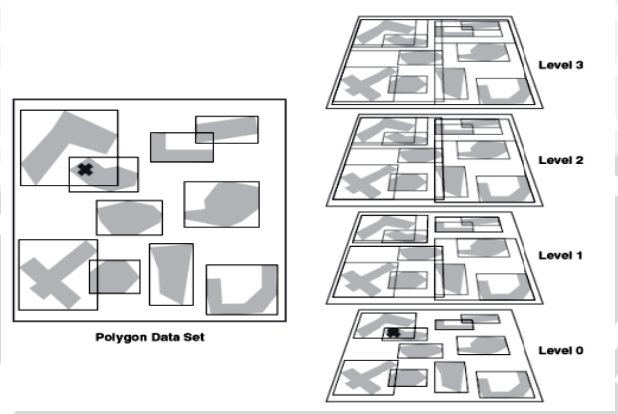
Gambar 2.6 The R-tree *Index*(Yeung, 2007).

R-tree termasuk *tree* bertingkat yang menyimpan satu set empat persegi panjang di setiap *node*, seperti B+-tree(Gutman,1984)

R-tree dirancang untuk *index* dataset 16roper sebagai segmen garis lurus terstruktur yang menghubungkan pasangan titik data spasial yang mengandung polygon. Tahap pertama pembentukan R-tree adalah membatasi masing-masing polygon kedalam batasan minimum persegi panjang(MBR) (Karen & Kemp 2008).

Fitur R-tree adalah mekanisme yang efisien untuk penyisipan dan penghapusan suatu data, R-tree juga memungkinkan memiliki jumlah maksimum

MBR yang berbeda dalam suatu kelompok, seperti di tunjukan pada **Gambar 2.7** Polygon dalam MBR (Karen & Kemp 2008).



Gambar 2.7 Polygon dalam MBR (Karen & Kemp 2008).

Ketika dilakukan *query* pada 17roper database untuk menemukan suatu wilayah dari parameter pencarian, 17roper akan menganalisis koordinat yang tersimpan didalam MBR dimulai dari *root* untuk menentukan MBR yang terdapat ditingkat subdivisi dilanjutkan ketingkat cabang, kemudian 17roper menscan persegi panjang MBR untuk mengidentifikasi obyek di tingkat daun yang mengandung MBR termasuk dalam batasan-batasan koordinat pencarian(Yeung 2007).

R-tree bisa dengan cepat untuk menemukan suatu wilayah polygon pada MBR yang berisi titik X yang di ilustrasikan pada **Gambar 2.7** tanpa harus mencari masing-masing MBR. Dengan langkah "menyaring" yaitu menghapus data polygon yang tidak mengandung titik dalam 17roperty, namun langkah penyaringan ini membutuhkan perhitungan lebih lanjut yang diperlukan untuk "meningkatkan" respon *index* yang memungkinkan titik berada dalam MBR dari polygon lain(Karen & Kemp 2008).

MBR yang lebih besar menyimpan kumpulan sekelompok MBR yang pada urutannya dapat di kelompokkan secara rekursif dan berturut-turut ke tingkat lebih tinggi, hasilnya adalah sebuah hirarki persegi bersarang yang memungkinkan untuk saling tumpang tindih, seperti digambarkan data set polygon pada sisi kanan pada **Gambar 2.7** (Karen & Kemp 2008).

Misalkan M adalah jumlah maksimum *entry* yang dimuat dalam *node* dan m menentukan jumlah minimum *entry* dalam sebuah *node* dengan $(2 \leq m \leq M/2)$(gutman,1984).

Sebuah R-tree memenuhi property sebagai berikut:

- Setiap *node leaf* berisi *entry index record* antara m dan M kecuali *root*
- Untuk setiap *index record* di dalam *node leaf* adalah (*I,tuple-identifier*), dimana *I* persegi tekecil yang berisi n-dimensi obyek spasial yang di representasikan dengan *tuple*
- Setiap *node non-leaf* memiliki anak antara m dan M kecuali *root*
- Untuk setiap *entry* di dalam *node nonleaf* adalah (*I,child-pointer)*, dimana *I* persegi terkecil dari persegi spasial yang berisi persegi *node* anak
- *Root* setidaknya memiliki dua anak kecuali itu adalah *node leaf*
- Semua *node leaf* berada pada *level* yang sama.

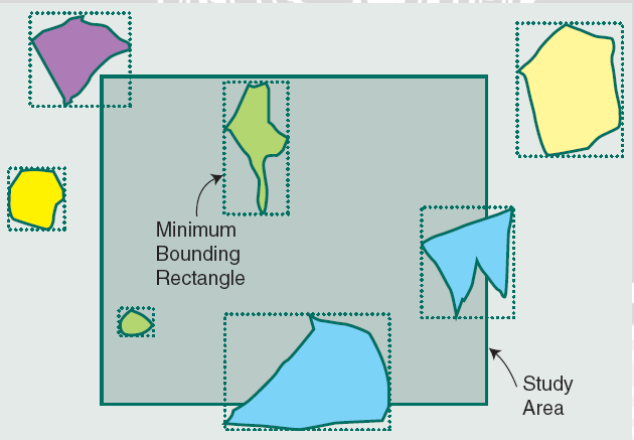
2.3.1 Minimum Bounding Rectangle (MBR).

Minimum Bounding Rectangle(MBR) atau disebut juga Minimum Bounding Box (MBB) adalah persegi panjang minimal berisi geometri ortogonal yang mendefinisikan fitur geografis atau koleksi geometri dalam dataset geografis, hal ini juga disebut "sampul" dari fitur (Karen & Kemp 2008).

Minimum Bounding Rectangles (MBR) adalah struktur yang sangat efisien yang secara luas diterapkan dalam GIS, pada dasarnya MBR mendefinisikan persegi pembatas minimal yang sisi-sisinya sejajar dengan sumbu x dan y pada sistem koordinat yang mencakup satu set satu atau lebih pada obyek geografis, hal ini di definisikan oleh dua koordinat di kiri bawah (x minimal, y minimal) dan kanan atas (x maksimal, y maksimal) seperti yang di tunjukkan pada **Gambar. 2.8** (Longley,dkk.2005).

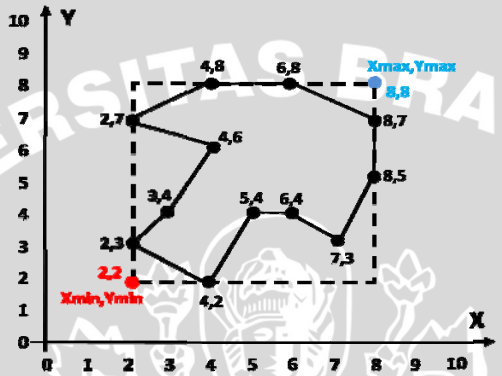
Sumbu pada MBR yang sejajar dengan sumbu x dan y adalah pendekatan paling sederhana yang hanya membutuhkan penyimpanan dua koordinat yang juga digunakan untuk memperkirakan cakupan fungsi atau data sebagai pembatas dari batas-batas obyek spasial(Karen & Kemp 2008)

Pada umumnya obyek spasial polygon didalam GIS memiliki batas-batas yang kompleks hal ini bisa menjadi tugas yang sangat mahal, solusi pendekatannya adalah dengan menggeneralisasi satu set data geometri dari obyek spasial kedalam dua koordinat persegi, hal ini juga digunakan untuk pencarian cepat.



Gambar 2.8 Study Area MBR polygon (Longley, dkk.2005).

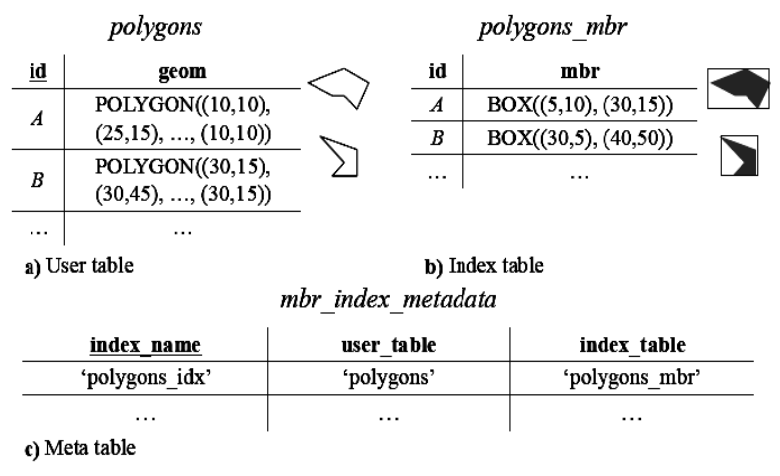
Pada **Gambar 2.8,** study area pertama menyaring semua obyek yang bukan termasuk kriteria pencarian dengan membandingkan MBR-nya dilanjutkan dari garis geometri obyek *polygon* yang tersisa, karena perbandingan koordinat yang dibutuhkan sangat sedikit maka pencarianpun akan sangat cepat.(Longley,dkk.2005).



Gambar 2.9 Susunan *tuple* planar komplek MBR Polygon

Ilustrasi pada **Gambar 2.9** mempresentasikan obyek spasial polygon yang memiliki *tuple* Polygon{2.3, 4.2, 5.4, 6.4, 7.3, 8.5, 8.7, 6.8, 4.8, 2.7, 4.6, 3.4, 2.3}, hasil yang didapatkan dengan mengurutkan *tuple* terendah dan tertinggi dari x dan y dihasilkan nilai Xmin dari *tuple* adalah 2 dan Ymin adalah 2 sedangkan nilai Xmax dari *tuple* adalah 8, dan Ymax dari *tuple* adalah 8, maka didapatkan MBR(2.2, 8.8).

Untuk contoh pengindeksan polygon ditunjukkan pada **Gambar 2.10** yang menggambarkan konsep metode akses relasional sederhana dari dua dimensi yang menunjukkan daftar meliputi persegi panjang minimal (MBR)**,** pada **Gambar 2.10a** user_table menyimpan obyek-relasional tabel *polygon* yang termasuk di dalamnya adalah atribut untuk tipe data *polygon* (GEOM) dan identifier obyek id atau OID.



Gambar 2.10 MBR *Index* (Manolopoulos,dkk.2005)

Untuk mempercepat pengaksesan spasial yaitu daftar idx polygon_mbr pada **Gambar 2.10b** ditetapkan pada idx user_table pada **Gambar 2.10a**, sehinggga tabel *index* dibuat dan diisi dengan menempatkan persegi panjang minimum (MBR) dari masing-masing batas *polygon* dari id kunci external.

Dengan demikian tabel *index* menyimpan informasi yang diperoleh secara murni dari user_table. Obyek skema memiliki semua *index* relasional terutama *index* nama tabel dan parameter *index* lainnya yang disimpan dalam sebuah tabel meta global yang ditunjukkan pada **Gambar 2.10c**(Manolopoulos,dkk.2005).

2.4 Pengindeksan R*-tree.

R*-tree adalah varian dari R-tree yang menawarkan beberapa perbaikan algoritma penyisipan, pada dasarnya pengembangan ini ditujukan untuk mengoptimalkan beberapa parameter(Rigaux, dkk. 2002).

R-tree hanya meminimalkan berdasarkan *21ocal21i* 21ocal daerah masing-masing MBR, disisi lain, R*-tree melebihi kriteria tersebut dengan memeriksa beberapa 21ocal21ia yang lain yang dapat meningkatkan kinerja selama

pemrosesan *query* (Manolopoulos, dkk. 2006). Kriteria yang dipertimbangkan oleh R*-tree tersebut sebagai berikut:

Meminimalkan daerah yang di cakup oleh masing-masing MBR. Kriteria ini bertujuan untuk meminimalkan ruang mati (wilayah yang di cakup oleh MBR) untuk mengurangi banyaknya jalur yang ditempuh selama pemrosesan *query*. Ini adalah kriteria tunggal yang juga diperiksa oleh R-tree.

Meminimalkan tumpang tindih antara MBR. Karena semakin besar tumpang tindih, maka semakin besar pula jumlah jalur yang dicari selama pemrosesan *query*, kriteria ini memiliki tujuan yang sama seperti sebelumnya.

Meminimalkan margin MBR (perimeter). Kriteria ini bertujuan membentuk persegi panjang lebih kuadratis, untuk meningkatkan kinerja *query* yang memiliki bentuk kuadratik besar, selain itu karena obyek-obyek yang kuadratik lebih mudah dikemas maka kumpulan MBR yang terkait pada level lebih atas diharapkan lebih kecil yaitu meminimisasi area.

Memaksimalkan penggunaan penyimpanan. *Node* lebih cenderung diakses selama pemrosesan *query* ketika pemakaian masih rendah, hal ini berlaku terutama selama *query* yang lebih besar, dimana sebagian besar *entry* memenuhi 22ocal22ia tersebut. Terlebih lagi dengan mengurangi penggunaan *node* disebabkan peningkatan *tree* yang tinggi, hal ini akan mempercepat pengaksesan *node*

Teknik untuk menemukan kemungkinan kombinasi terbaik dari kriteria tersebut adalah dengan mengatasi masalah pemilihan yang baik pada proses penyisipan, dimana versi sebelumnya R-tree yang hanya memperhitungkan

parameter *account* 23ocal, R*-tree memeriksa area parameter margin dan tumpang tindih dalam kombinasi yang berbeda

Untuk menentukan salah satu biaya tumpang tindih yang mendekati minimum dengan mengurutkan persegi panjang di N.

Untuk menggabungkan data baru persegi panjang yang membutuhkan pembesaran area. Misalkan A, *entry* p pertama dari *entry* di A, pertimbangkan semua *entry* dalam N, pilih *entry* persegi panjang yang lengkap setidaknya dibutuhkan hubungan pembesaran dalam tumpang tindihnya(Beckmann, dkk.1990).

Untuk setiap distribusi nilai terbaik ditentukan oleh tiga kombinasi yang berbeda yang digunakan untuk ke tiga nilai hasil distribusi terbaik tersebut adalah sebagai berikut:

(i) Area-value

Area[bb(group pertama)] + Area[bb(group ke dua)] (2.1)

(ii) Margin-value

Margin[bb(group pertama)] + Margin[bb(group kedua)]. (2.2)

(iii) Overlap-value

Area[bb(group pertama) ∩ bb(group kedua)]. (2.3)

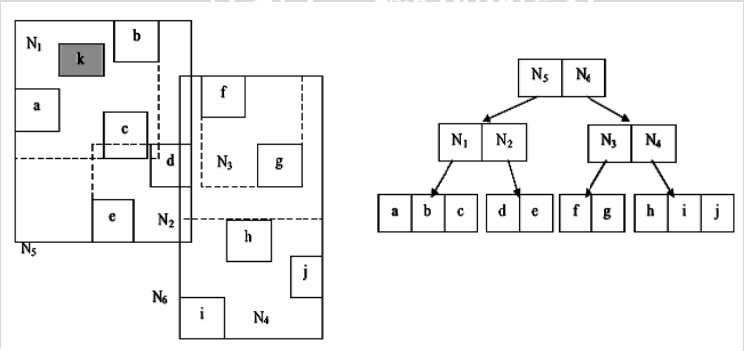
Dimana (bb) adalah Bounding Box atau kata lain adalah Minimum Bounding Rectangle (MBR), metode pengolahan tersebut untuk menentukan minimum pada sumbu, jumlah minimum dari nilai-nilai terbaik pada sumbu dan minimum global, dari nilai yang diperoleh dapat diterapkan untuk menentukan sumbu divisi atau distribusi akhir pada saat sumbu split dipilih(Beckmann, dkk.1990).

2.4.1 Metode Insert

R*-tree yang di gambarkan pada **Gambar 2.11**. pada saat menyisipkan data persegi panjang I, ChooseSubtree dimulai dari tingkat akar, kemudian memilih *entry* MBR yang daerahnya memerlukan pembesaran. Dalam contoh ini diasumsikan *Node* yang dibutuhkan adalah N5 (yang pembesarannya tidak diperlukan sama sekali).

Karena kriteria yang diperiksa adalah meminimalkan daerah untuk *node* daun, **ChooseSubtree** memilih kriteria yang meminimasi tumpang tindih, karena berdasarkan hasil eksperimen dari N. Beckmann, dkk (1990) menunjukkan bahwa kriteria ini yang terbaik daripada yang lain.

Oleh karena itu diasumsikan subtree berakar di N5, kemudian **ChooseSubtree** memilih dimana *entry* pembesaran MBR di antara *entry* saudara di setiap *node* saling tumpang tindih terkecil, yaitu *node* N1(Manolopoulos, dkk.2006).



Gambar 2.11 Contoh R*-tree (Manolopoulos, dkk. 2006).

ChooseSubtree untuk memilih sebuah *node* daun yang tidak bisa menampung *entry* baru semisal daun sudah memiliki jumlah *entry* maksimum, R*-tree ini tidak langsung meresor untuk membelah *node*, sebaliknya dengan

menemukan sebagian kecil dari *entry* dari *node* yang meluap kemudian mereinserts ke *entry* yang meluap tersebut.

Himpunan *entry* yang akan dimasukkan kembali adalah dimana jarak pusat dari pusat *node* adalah (p) di antara 30% terbesar, heuristik ini adalah untuk mendeteksi dan membuang *entry* terjauh yang berdasarkan dari hasil terbaik eksperimen N. Beckmann, dkk (1990). (Manolopoulos, dkk.2006).

Algoritma InsertData

ID1 Panggil **Algoritma Insert** yang dimulai di tingkat daun sebagai parameternya untuk memasukkan data baru persegi panjang.

Algoritma Insert

I1 Memanggil **Algoritma ChooseSubtree**. Dengan level sebagai parameter untuk menemukan *node* yang sesuai dengan N dan menempatkan *entry* baru E

I2 If N memiliki kurang dari *entry* M,

Tampung E dalam N

If N memiliki *entry* M.

[Identifikasi reinsertion atau split]

Panggil **Algoritma OverflowTreatment** dengan level dari N sebagai parameter.

I3 If OverflowTreatment dipanggil dan split dilakukan, bila overflow merambat ke level lebih tinggi.

Lakukan OverflowTreatment pada level tersebut..

If OverflowTreatment menyebabkan perpecahan pada root.

Buat root baru

I4 Sesuaikan semua persegi panjang pembatas minimal yang meliputi di jalur penyisipan begitu juga pembatas persegi panjang yang disertai persegi panjang pembatas semua anak

Algoritma ChooseSubtree

CS1 Tetapkan N menjadi akar

CS2 If N adalah daun,

Kembalikan N

Else

If *childpointers* di titik N pada daun

[menentukan biaya minimum saling tumpang tindih]

memilih *Entry* persegi panjang dalam N yang membutuhkan setidaknya pembesaran daerah tumpang tindih untuk memasukkan data baru persegi panjang pilih diantara *entry* persegi panjang yang kebutuhan pembesaran daerahnya paling besar

Then

Entry dengan luas daerah persegi panjang terkecil

If *childpointers* di N tidak menunjuk ke daun

[menentukan biaya area minimum]

pilih *entry* dalam N yang kebutuhan daerah pembesaran persegi panjang setidaknya untuk memasukkan data baru persegi panjang,

putuskan hubungan dengan cara memilih wilayah terkecil dari *entry* persegi panjang

End

CS3 Tetapkan N untuk menjadi *childnode* yang ditunjuk oleh *childpointer* dari *entry* yang dipilih dan ulangi dari **CS2**

Algoritma OverflowTreatment

OT1 If level bukanlah level akar dan ini adalah panggilan pertama kali dari OverflowTreatment pada level yang diberikan selama satu penyisipan data persegi panjang

Then

Panggil Algoritma Reinsert

Else

Panggil Algoritma Split

End

Algoritma Reinsert

RI1 Untuk semua M+1 *entry* dari simpul N, menghitung jaraknya diantara pusat persegi panjang dengan pusat pembatas persegi panjang dari N

RI2 Urutkan *entry* dalam urutan menurun dari jaraknya yang dihitung di *RI*1

RI3 Hapus *entry* p pertama dari N dan sesuaikan pembatas persegi panjang setelah penghapusan *entry* p dari N

RI4 Didalam mengurutkan yang telah didefinisikan dalam RI2, dimulai dengan jarak maksimum atau jarak minimum, dengan memanggil **Algoritma Insert** untuk memasukkan *entry* kembali

2.4.2 Metode Split

Metode yang digunakan R*-tree untuk menemukan pembelahan (split) terbaik sepanjang sumbu masing-masing yaitu *entry* persegi panjang pertama diurutkan dengan nilai terendah, kemudian diurutkan dengan nilai tertinggi.

Untuk setiap distribusi terendah dan tertinggi menentukan M-2m+2 dengan M+1 *entry* dalam dua kelompok(G1) dan (G2), dimana distribusi ke-k dengan (k=1, (M-2m+2)) yang digambarkan sebagai kelompok pertama(G1) berisi (m-1) pertama +k input, kelompok kedua (G2) berisi *entry* yang tersisa (Beckmann,dkk.1990).

R*-tree menemukan partisi di dua *node* dengan memilih sumbu tumpang tindih yang minimal, proses untuk mencari sumbu terbaik ini dengan cara mengurutkan nilai atas (max) dan nilai bawah (min) di setiap persegi, hal ini menghasilkan 2 (2 (M – 2m + 2)) perhitungan.

Pada algoritma R*treeChooseAxis sumbu terbaik ditunjukkan oleh S dengan jumlah dari nilai batas-batas minimal yang diberikan oleh *axis* terbaik, apabila dalam pemilihan ditribusi tumpang tindih minimal menghasilkan nilai yang sama maka pilihlah salah satu diantaranya dengan ruang mati yang minimum. (Rigaux, dkk, 2002).

R*treeChooseAxis (*E:*** set of *entry*)**

begin

Mengatur variabel min-perimeter ke nilai yang mungkin maksimum

for each *axis a* **do**

mengurutkan *E* yang berkaitan dengan *a*

S = 0

// Pertimbangkan kemungkinan semua distribusi

for *k* = 1 **to** (*M*− 2*m*+ 2) **do**

*G*1 = *E*[1,*m*+*k* – 1]; *G*2 = *E*[*m*+*k*,*M*+1]

// M+k-1 *entry* pertama disimpan di G1, sisanya di G2.

Mg = perimeter (mbb(G1)) + perimeter (mbb(G2)) // nilai Perimeter

S = *S* + *Mg* // Hitung jumlah dari perimeter untuk sumbu saat ini

end for

// Jika S adalah nilai minimal terakhir, maka menjadi sumbu terbaik

if (*S* < *min-perimeter*) **then**

axis-terbaik = a

min-perimeter = S

end if

end for

return *axis*-terbaik

end

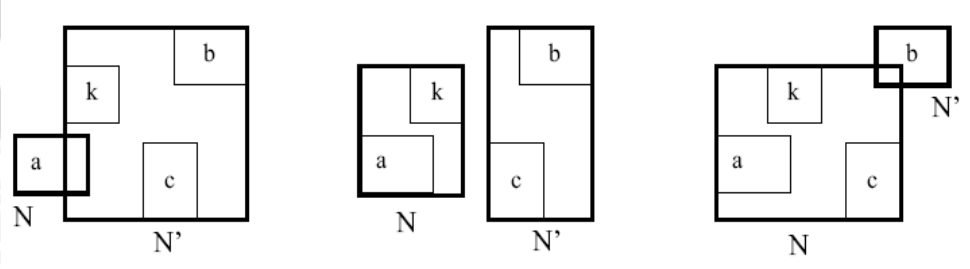
Pada ilustrasi selama proses reintegrasi yang di tunjukkan pada **Gambar 2.11** dimana N1 diasumsikan mengalami perluapan dan memilih *entry* b yang ada di N1.

Proses reintegrasi tersebut dengan memasukkannya *node* kembali mencapai seperti penyeimbangan tree yang secara signifikan meningkatkan kinerja selama pemrosesan *query*, namun reintegrasi adalah operasi yang mahal,

oleh karena itu hanya sekali di setiap level tree reintegrasi dilakukan(Manolopoulos, dkk. 2006).

Ketika overflow tidak dapat menangani reintegrasi, pembelahan *node* dilakukan dengan memanggil algoritma split, pada algoritma split ini terdiri dari dua langkah(Manolopoulos, dkk. 2006)..

1. Memutuskan sumbu perpecahan antara semua dimensi, sumbu perpecahan adalah salah satu dari keseluruhan perimeter terkecil dengan memilah semua *entry* pada koordinat batas kirinya, kemudian setiap divisi dari daftar diurutkan yang memastikan bahwa setiap *node* minimal 40% penuh berdasarkan eksperimen hasil terbaik dari Beckmann, dkk (1990). Pada **Gambar 2.12** pembagian *node* di setiap divisi diasumsikan dengan divisi (1-3) yang mengalokasikan *entry* pertama ke N dan tiga *entry* lainnya ke N^1, kemudian algoritma menghitung perimeter N dan N^1 pada divisi tersebut, begitu juga untuk divisi (2-2) dan (3-1).
2. Mengulangi proses ini berkenaan dengan batas kanan MBR.



Gambar 2.12 Divisi sumbu split (a)1-3 divisi (b) 2-2 divisi (c)3-1 divisi(Manolopoulos, dkk, 2006).

Pada **Gambar 2.12** adalah suatu contoh dari pemeriksaan divisi pada saat sumbu split dipilih dengan menentukan berbagai 30riteria optimasi dan minimasi

untuk menyisipkan sumbu x, algoritma split memilih dengan mempertimbangkan batas-batasnya lebih rendah atau tinggi pada setiap sumbu dimensi.

Hasil akhir dari pemilihan divisi adalah salah satu dari divisi yang memiliki minimal saling tumpang tindih antara MBR dari *node* yang dihasilkan yaitu divisi 2-2 dengan nilai hasil akhir nol yang tidak menimbulkan saling tumpang tindih antara N dan N', dengan demikian menjadi solusi akhir dari divisi (Manolopoulos, dkk. 2006).

Algoritma Split

S1 Memanggil **Algoritma ChooseSplitAxis** untuk menentukan dan membagi sumbu tegak lurus

S2 Memanggil **Algoritma ChooseSplitIndex** untuk menentukan distribusi sepanjang sumbu terbaik dalam dua kelompok

S3 Bagikan *entry* ke dalam dua kelompok

Algoritma ChooseSplitAxis

CSA1 Untuk setiap sumbu

Urutkan *entry* dengan lebih rendah dan tinggi maka berdasarkan nilai tertinggi dan terendah persegi panjang tentukan semua distribusi

Hitunglah S dari jumlah semua nilai margin disetiap distribusi yang berbeda

Selesai

CSA2 Pilih sumbu dengan S minimal sebagai sumbu perpecahan

Algoritma ChooseSplitIndex

CSI1 Sepanjang memilih sumbu untuk membelah, distribusi dipilih dengan nilai tumpang tindih minimal, putuskan hubungan dengan memilih nilai daerah distribusi yang minimum

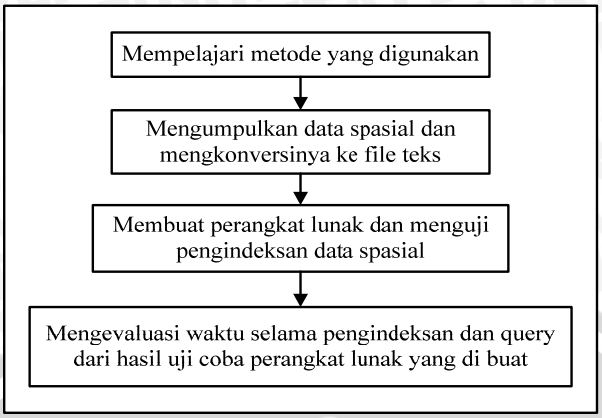
BAB III
METODOLOGI DAN PERANCANGAN

Pada bab metodologi dan perancangan ini akan dibahas penggunaan metode yang digunakan dalam pembuatan perangkat lunak untuk pengindeksan data spasial

Tahap-tahap pembuatannya sebagai berikut.

1. Melakukan studi literarur mengenai struktur data R*-tree yang akan digunakan dalam pembuatan perangkat lunak pengindeksan data spasial dari jurnal-jurnal yang sesuai, yang telah disinggung pada bab 2.
2. Mengumpulkan data spasial dan mengkonversinya ke geometri menggunakan ARCGist 9.0 dan disimpan dalam file Teks.
3. Membuat perangkat lunak sesuai dengan analisa dan perancangan yang dilakukan.
4. Menguji coba perangkat lunak dengan melakukan pengindeksan dan *query* pada data spasial.
5. Mengevaluasi hasil yang di peroleh dari uji coba tersebut dengan membandingkan waktu selama pengindeksan dan juga waktu yang dibutuhkan selama *query* dari jumlah *record* yang bervariasi.

Langkah – langkah pembuatan perangkat lunak dapat di gambarkan seperti Gambar 3.1.



Gambar 3.1 Diagram Alir Pembuatan Perangkat Lunak

3.1 Deskripsi Umum Sistem

Pada pengindeksan data spasial ini akan di coba di bandingkan waktu selama proses pengindeksan dan juga waktu yang di butuhkan selama proses *query* menggunakan struktur data R*-tree.

Uji coba perangkat lunak untuk pengindeksan maupun *query* dengan tipe data Vektor dan di lakukan terhadap masing-masing tipe data(line dan polygon), dengan masing-masing range jumlah data yang bervariasi.

Hal nantinya yang akan di perbandingkan menyangkut waktu yang di butuhkan selama proses pengindeksan begitu juga waktu yang di butuhkan selama proses *query* dengan masing-masing prosentase reinsert untuk reintegrasi yang ada pada struktur data R*-tree.

3.2 Pengujian Data

Pada penilitian ini, data yang digunakan dalam pengujian adalah data bertipe vector dengan jumlah *record* yang bervariasi pada masing-masing tipe data yaitu line dan polygon, data yang digunakan berasal dari data yang berformat shp yang di konversi ke file teks geometri dengan menggunakan software batuan ARCGist 9.0 agar bisa di baca dan di indekskan ke struktur data R*-tree

Selama proses uji coba, file shp saling berintegrasi dengan 2 file lain yaitu file berformat dbf sebagai data atribut dan file berformat shx sebagai file *index* untuk mengintegrasikan file shp dan dbf.

Sebagai view dari data shp bertipe line ditunjukan pada **Gambar 3.2** dan untuk tipe polygon di tunjukan pada **Gambar 3.3.**

Gambar 3.2 View data shp dengan tipe vector-line

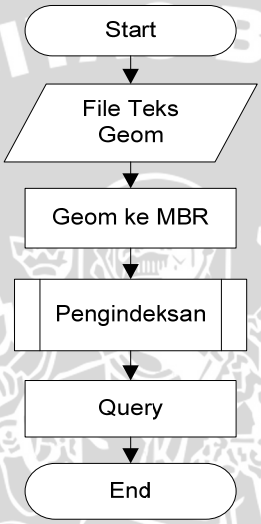
Gambar 3.3 View data shp dengan tipe vector-polygon

3.3 Perancangan Proses

Proses di dalam perancangan pengindeksan data spasial menggunakan struktur data R*-tree ini mula-mula file teks yang menyimpan geometri dari sebuah feature *polygon* atau *line* di tentukan MBR untuk menjadi area MBR yang

mencangkupi dari masing-masing tuple data geometri spasial dari file teks spasial, dengan susunan MBR{Xl,Xu,Yl,Yu}.

Proses selanjutnya mengindekskan MBR dari data spasial ke struktur data R*-tree kemudian di lakukan *query*, perancangan proses pengindeksan menggunakan struktur data R*-tree seperti di tunjukkan pada **Gambar 3.4** berikut.



Gambar 3.4 Gambaran umum system

3.3.1 Pengindeksan Metode R*-tree

Algoritma R*-tree untuk memasukkan *Entry* baru dengan algoritma **InsertData** dimulai di tingkat daun untuk memasukan data persegi panjang baru menggunakan Algoritma **InsertData.**

Algoritma **InsertData** terlebih dahulu memutuskan cabang mana yang akan di pilih pada setiap tingkatan *tree*, dengan memanggil Algoritma ChosseSubtree.

Algoritma ChooseSubtree untuk menempatkan data baru dan memilih *node* di setiap tingkatan *tree* terlebih dahulu menghitung berkenaan dengan batas-

batasnya yang memerlukan area pembesaran dan *overlap* untuk menyisipkan *node*.

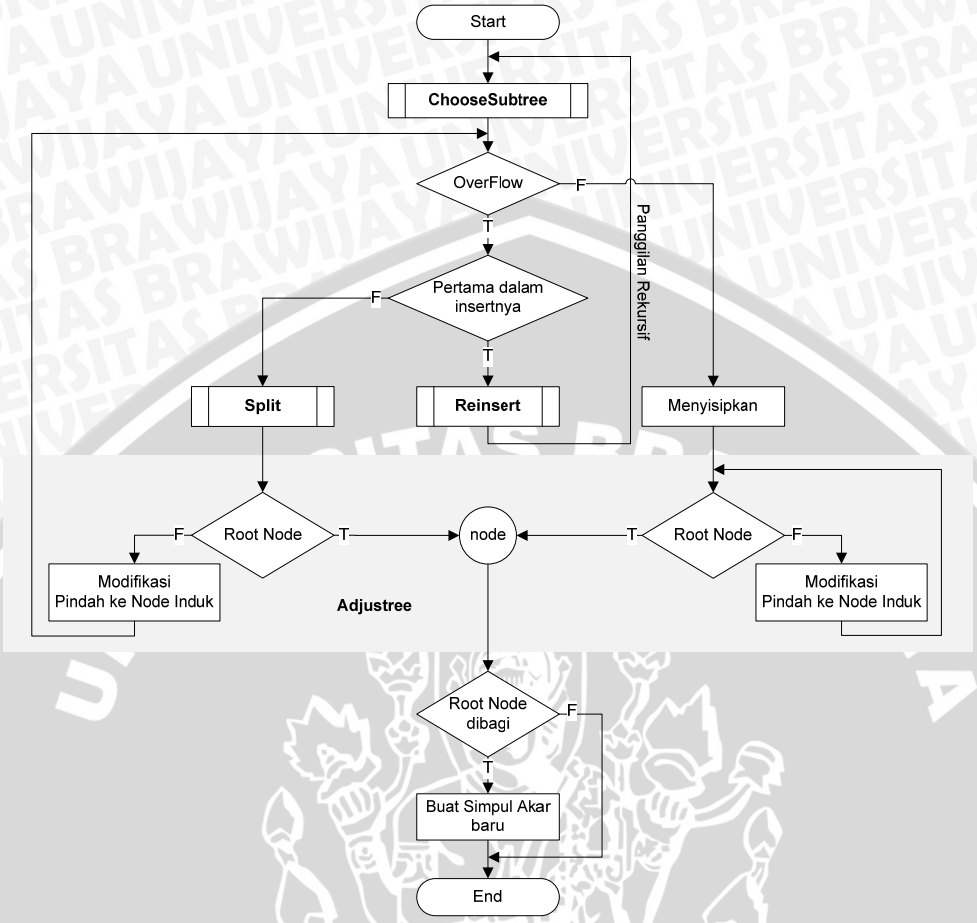
proses dilanjutkan ke algoritma **OverflowTreatment** untuk memastikan bahwa kapasitas dalam *node* yang akan di tempati data baru pada anak atau daun telah penuh atau tidak, bila kapasitas telah penuh dan proses ini bukan pertama kali selama pemasukan berkenaan dengan *Entry* tersebut maka pembelahan *node* dilakukan dengan algoritma ***split***, apabila proses ini adalah proses pertama kali berkenaan dengan pemasukan *Entry* tersebut maka memasukkan kembali *node* tersebut dengan memanggil Algoritma *Reinsert*.

Algoritma ***Split*** sebelum meresor untuk membelah terlebih dahulu memproses perhitungan untuk beberapa kriteria minimasi dan optimasi dengan menggunakan algoritma **ChooseSplitAxis** untuk menentukan dan memilih sumbu terbaik yang hasilnya dilanjutkan dengan algoritma **Chose*SplitIndex*** untuk optimasi dan meminimasi ruang mati serta tumpang tindih minimal

Algoritma ***Reinsert*** sebelum me*reinsert node* kembali terlebih dahulu menghitung jarak pusat setiap *Entry* dengan jarak pusat pembatas dari kumpulan *Entry* tersebut.

berdasarkan *Entry* jarak pusat terjauh dari jarak pusat pembatasnya dimasukkan kembali(*reinsert*) berulang secara rekursif dengan memanggil Algoritma *Insert* untuk memasukkan data,

proses *reinsert* dengan jarak pusat ini digunakan untuk membuang *Entry node* terjauh dari *node* pembatasnya, gambaran umum struktur data R*-tree ditunjukan pada **Gambar 3.5** Flowchart Gambaran Umum Struktur Data R*-tree.



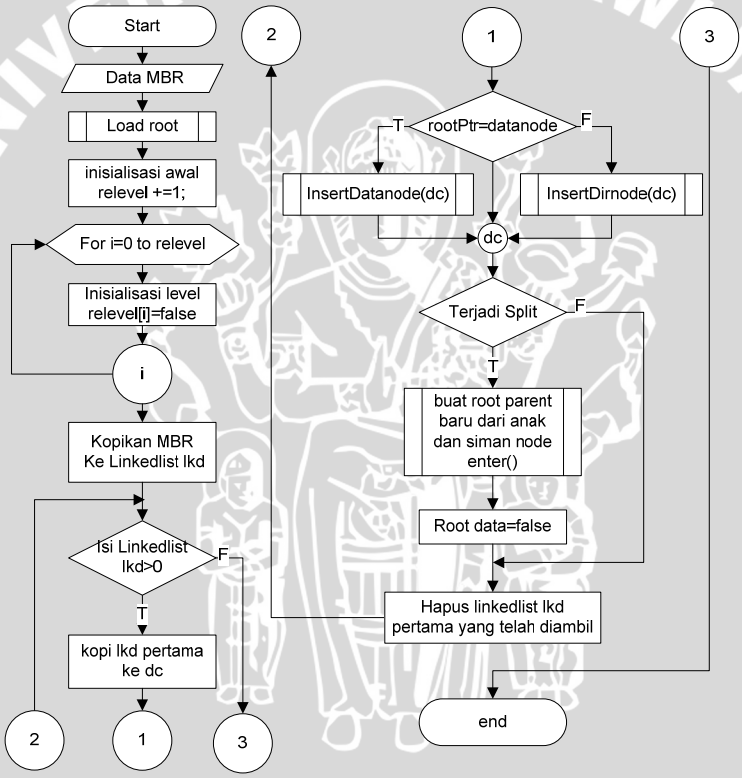
Gambar 3.5 Flowchart Gambaran Umum Struktur Data R*-tree

3.3.2 Prosedur *Insert*

Data masukan berupa MBR yang membatasi tuple dari geometri obyek spasial line maupun polygon, dimasukkan ke dalam R*-tree, pertama kali proses *insert* melakukan pengecekan dalam *node tree* jika *tree* dalam keadaan kosong termasuk proses pertama kali maka root_isdata di inisialisasi true kemudian memanggil load root.

Root *pointer* akan menunjuk ke data *node* yaitu *node* yang berada di tingkat daun, begitu sebaliknya apakah data *insert* dimasukkan ke direktori *node* yaitu *pointer* yang menunjuk untuk *node parent* (*node nonleaf*).

Proses selanjutnya data dikopikan ke dalam linkedlist untuk dimasukkan ke data *node*, setelah proses *insert* mengidentifikasi apakah pemasukan menyebabkan pembelahan jika terjadi pembelahan apakah hasil pembelahan menyebabkan pembagian root atau tidak, jika iya lakukan pembagian root untuk anaknya dan buat root baru dari anak tersebut dan menyimpannya ke dalam *node* dengan *parent* baru yang terbentuk dari hasil pembelahan, untuk lebih jelasnya bisa di lihat di **Gambar 3.6** berikut :

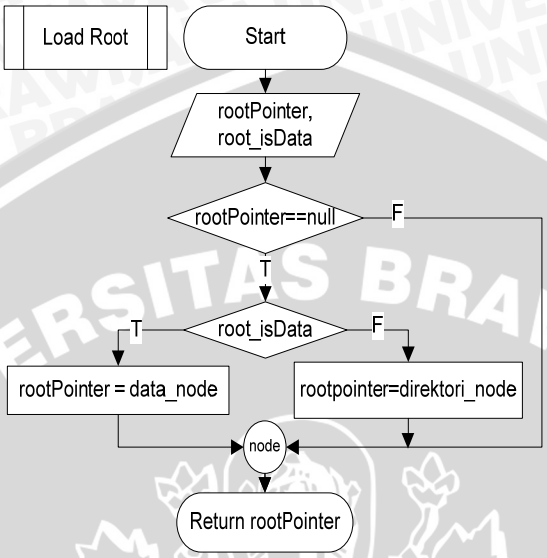


Gambar 3.6 Prosedur *insert*

3.3.3 Prosedur Load Root

Prosedur Loadroot pada **Gambar 3.7** digunakan untuk menunjukkan posisi *pointer*, pada saat pertama kali pengecekan apakah root*Pointer* null yang berarti belum ada masukan jika iya apakah root is data (data *nodeleaf*) maka

pointer menunjuk ke data*node* (data *nodeleaf*) jika bukan menunjuk ke direktori *node* (data *node nonleaf*)

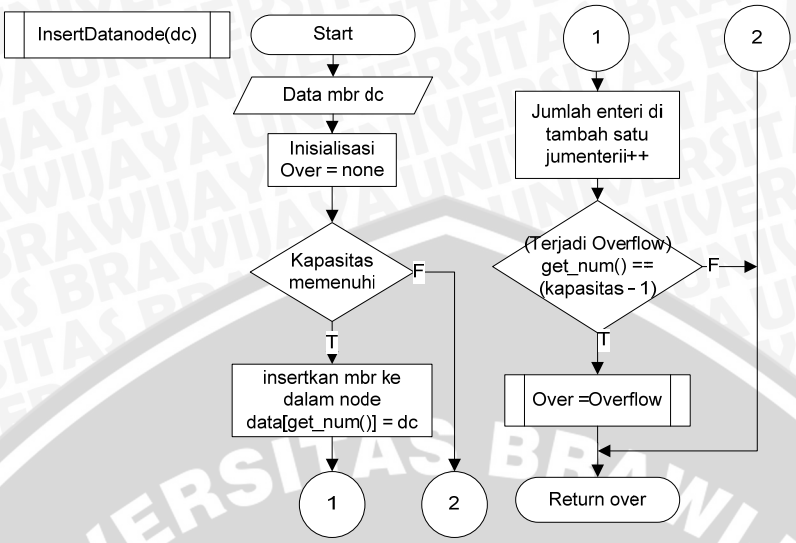


Gambar 3.7 Prosedur load *pointer*

3.3.4 Prosedur *Insert* Data *Node*

Proses *insert* data*node* pada **Gambar 3.8** digunakan untuk memasukkan *node leaf*, proses pertama kali mengecek apakah kapasitas *Entry* masih memenuhi bila masih data MBR di sisipkan pada *node* yang sesuai posisi jumlah terakhir *Entry* dari pemasukan sebelumnya (jum*Entry*), kemudian jumlah *Entry* ditambah satu untuk *insert* data selanjutnya.

Pengecekan selanjutnya apakah pemasukan menyebabkan overflow dengan jumlah *Entry* melebihi kapasitas (getnum() == kapasitas-1) bila iya panggil overflow-treatment bila tidak maka mengembalikan none untuk proses pemasukan data selanjutnya.



Gambar 3.8 Prosedur *insert node* data

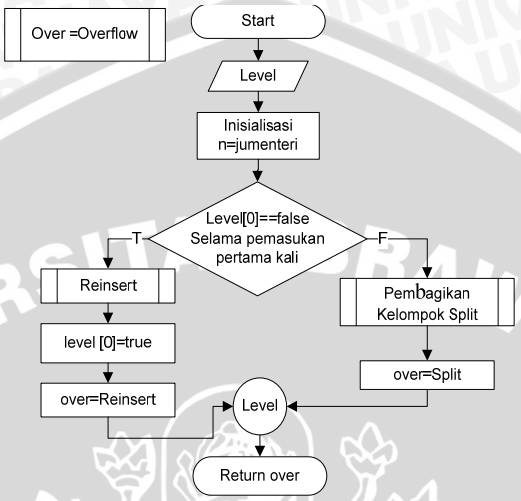
3.3.5 Prosedur Overflow Treatement

Prosedur OverflowTreatment **Gambar 3.9** digunakan untuk mengidentifikasi apakah *Entry node* memerlukan pembelahan yang disebabkan kapasitas penuh ataukah memasukkan kembali untuk reintegrasi *tree* karena selama proses pemasukan berkenaan dengan *entry* tersebut adalah pertama kali dalam *tree*.

Pengecekan overflowtreatement dilakukan pada level di root apakah root pada level daun false yang berarti level masukan berkenaan dengan *entry* ini adalah pertama kali saat memasukkan data di *tree* maka proses akan memanggil prosedur *reinsert* untuk memasukkan data kembali dan mengganti level nol dengan true setelah proses *reinsert*, karena reintegrasi diperkenankan sekali dalam level *tree* berkenaan dengan *entry* ini.

Pengecekan overflowtreatement yang disebabkan kapasitas penuh, overflowtreatement mengidentifikasi apakah level pada *leaf true*, yang berarti level daun selama proses pemasukan berkenan dengan *entry* tersebut bukanlah

pertama kali maka dilakukan *split* yang mula-mula mengelompokkan data menjadi dua untuk di distribusikan dan melakukan pembelahan dengan memanggil prosedur pengelompokan *split*.



Gambar 3.9 Prosedur Overflow Treatment

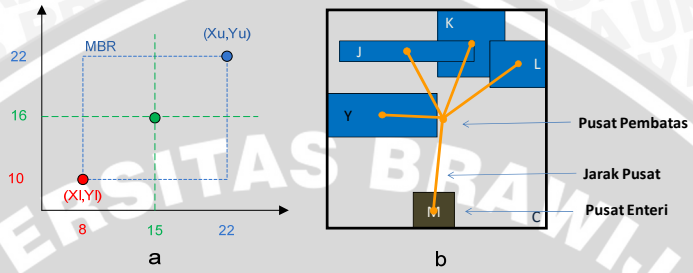
3.3.6 Prosedur *Reinsert*

Prosedur *reinsert* **Gambar 3.11** digunakan untuk memasukkan kembali data *node* karena berkenaan dengan *entry* ini adalah pertama kali saat pemasukkan data di *tree* dari prosedur overflowtreatment

Data MBR sebelum dimasukkan kembali di urutkan berdasarkan jarak pusat tiap *Entry* dengan pembatas pusat MBR tersebut

Hasil jarak pusat sumbu x,y setiap *Entry* MBR di simpan kedalam variabel *center* berdasarkan id MBR, nilai pusat terhadap x dari pembatas *Entry* maupun *Entry* itu sendiri didapatkan dengan Xl+Xu/2 sedangkan terhadap y didapatkan dengan Yl+Yu/2 seperti di gambarkan pada Gambar **3.10a** ilustrasi *center* x = (22+10/2) dan *center* y= (22+8/2) :

Proses selanjutnya mengalokasikan MBR untuk *sorting* yang di isi kan berdasarkan perhitungan jarak pusat tersebut dan memanggil quicksort dengan kriteria *sorting center*. Ilustrasi jarak pusat setiap *Entry* di ilustrasikan **Gambar 3.10b** Jarak pusat *Entry* terhadap pusat pembatasnya

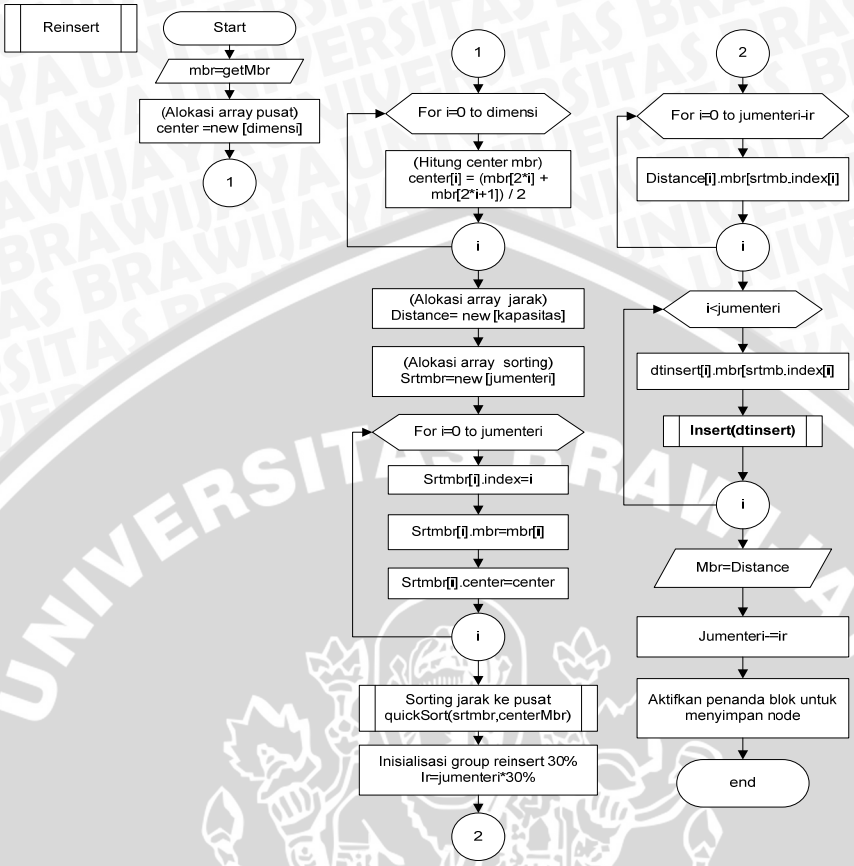


Gambar 3.10 (a) Ilustrasi pusat MBR , (b) Jarak pusat *Entry* terhadap pusat pembatas

Untuk memasukkan kembali *node* di isi 30% penuh yaitu (jumlah *Entry* * 30%) hal ini untuk membuang *Entry* terjauh dari suatu *node* MBR pembatas, hasil 30% dari jumlah *Entry sorting* jarak pusat tertinggi disimpan dalam variabel ir, gunakan perulangan sebanyak ir untuk menentukan MBR dengan jarak terjauh yang akan dimasukkan kembali, dari ilustrasi **Gambar 3.10b** jarak terjauh adalah M.

MBR sebanyak ir tersebut satu-persatu dimasukkan kembali dengan berulang memanggil prosedur *insert* yang dimasukkan dalam linkedlist lkd

Hasil akhir dari prosedur *reinsert* yaitu *node* anak sekarang berisi sisa dari MBR yang telah dimasukkan 30%, dengan demikian jumlah *Entry* dalam *node* tersebut sekarang dikurangi dengan 30% dari jumlah *Entry* yang telah dimasukkan kembali.



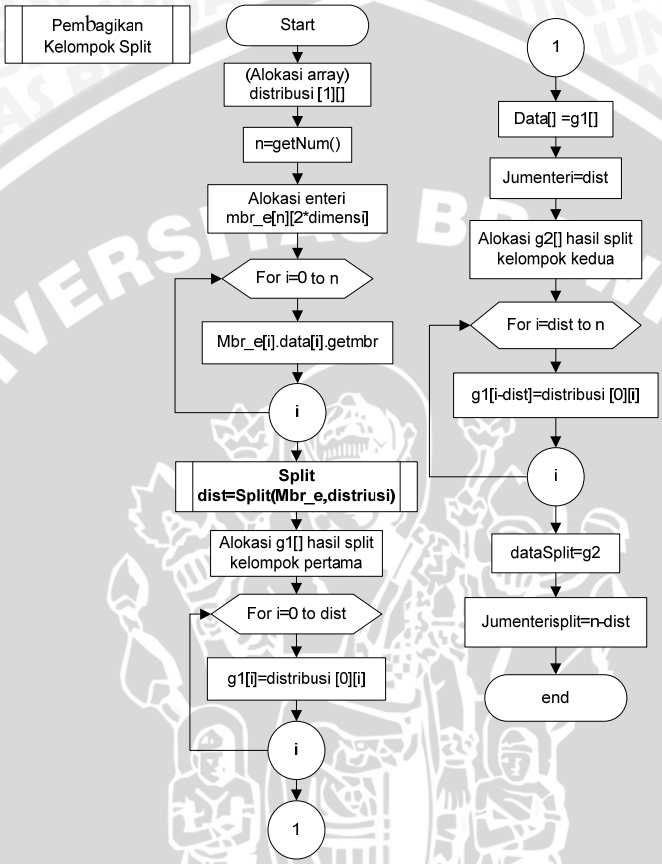
Gambar 3.11 Prosedur *reinsert*

3.3.7 Prosedur Pembagian Kelompok *Split*

Prosedur pembagian kelompok *split* **Gambar 3.12** digunakan untuk mendistribusikan pengelompokan pembelahan dengan mengalokasikan array untuk hasil pembelahan yang di simpan dalam distribusi[1][] yang berarti data array distribusi sebanyak 2 kolom untuk group pertama dan group ke dua, kemudian mengisikan MBR ke mbr_e sebanyak jumlah *Entry* dari MBR dalam *node* tersebut untuk di lakukan pembelahan dengan memanggil prosedur *split*().

Hasil pembagian kelompok dari prosedur *split* disimpan dalam array distribusi[0] untuk kelompok pertama dan distriusi[1] untuk kelompok ke dua yang sesuai dengan id MBR hasil dari optimasi dan minimasi yang didistribusikan

oleh pembelahan terbaik dari prosedur *split*, jumlah *Entry* sekarang di kurangi oleh jumlah *Entry* hasil distribusi *split*(dist), proses selanjutnya di lakukan penyisipan kedalam *node* dari hasil distribusi.



Gambar 3.12 Pembagian kelompok *split*

3.3.8 Prosedur *Split*

Prosedur *split* yang di gambarkan pada **Gambar 3.13** digunakan untuk mendistribusikan pembelahan terbaik dari hasil optimasi dan minimasi yang di sebabkan kapasitas *entry* penuh dari prosedur overflowtreatment

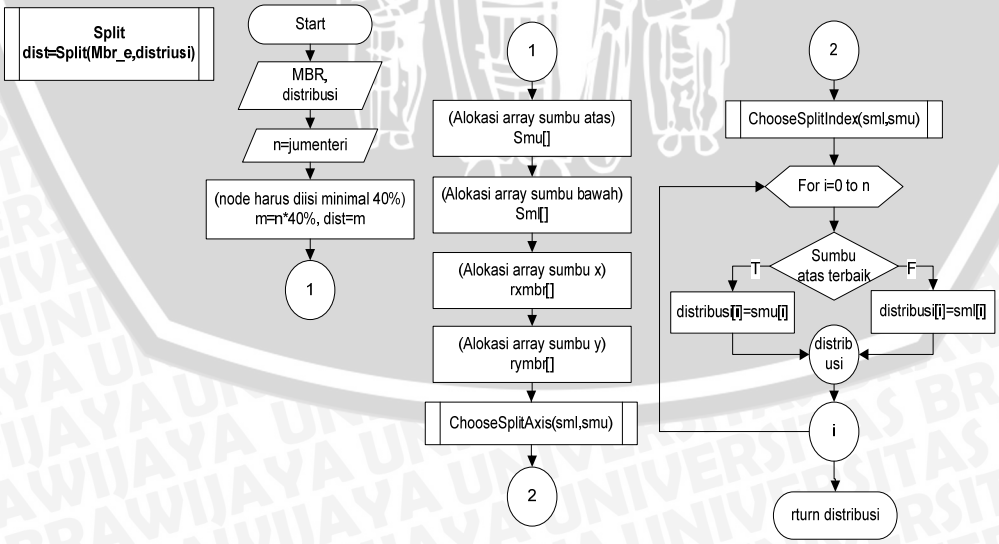
Sebelum dilakukan pembelahan isikan m minimal 40% dari jumlah *entry*, hal ini berdasarkan eksperimen hasil terbaik dari Beckman, dkk (1990), kemudian mengalokasikan array untuk menyimpan hasil *sorting* batas atas Xmax MBR

dengan alokasi array smu[] dan batas bawah Xmin MBR dengan sml[] dan sumbu dari x,y MBR untuk minimal tumpang tindih dan area mati.

Prosedur *Split* selanjutnya memanggil **ChooseSplitAxis** untuk menentukan sumbu pembelahan terbaik berdasarkan *margin* dari persamaan 2.2 sebelum dilakukan optimasi dan minimasi ruang mati serta tumpang tindih pada persamaan 2.1 dan 2.3 dengan prosedur ***ChooseSplit*Index**

Hasil pemilihan sumbu terbaik dari prosedur **ChooseSplitAxis** dihitung ruang mati dan tumpang tindih minimal oleh ***ChooseSplit*Index,** hasil distribusi kelompok pembelahan terbaik dari proses ***ChooseSplit*Index** disimpan di sml(batas atas) dan smu(batas bawah),

Hasil Distribusi terbaik berkenaan dengan batas atas (smu) maka distribusi di isi oleh smu dengan lu=true dan sebaliknya jika distribusi terbaik adalah batas bawah maka lu=false, perbandingan untuk pengecekan hasil terbaik dari batas atas ataukah batas bawah hal ini dilakukan dengan looping sebanyak jumlah *Entry* dari MBR yang sebelumnya telah di distribusikan.



Gambar 3.13 Prosedur *Split*

3.3.9 Prosedur ChooseSplitAxis

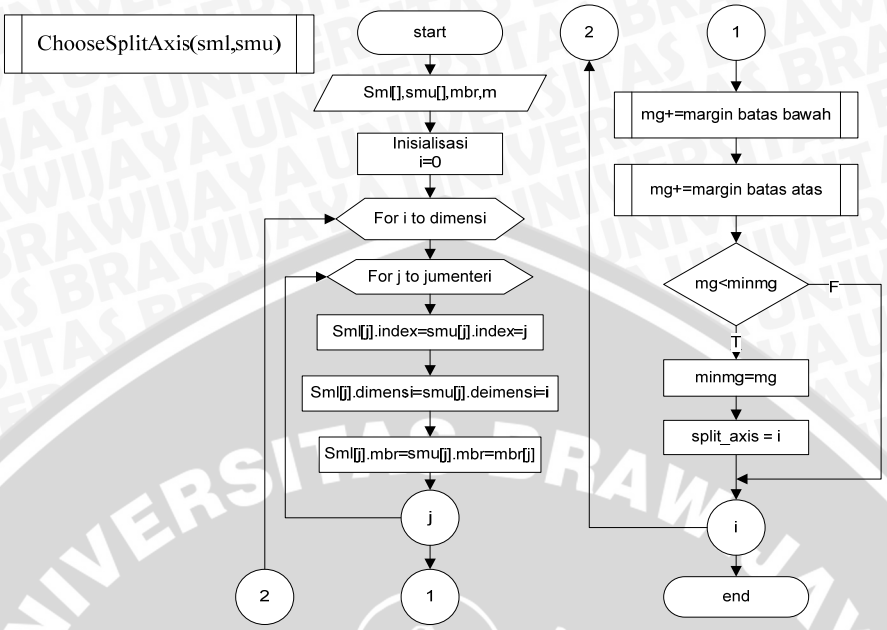
Prosedur ChooseSplitAxis pada **Gambar 3.14** digunakan untuk menentukan sumbu pembelahan terbaik berdasarkan persamaan 2.2 sebelum dilakukan minimasi ruang mati dan tumpang tindih dengan persamaan 2.1 dan 2.3 oleh prosedur ChooseSplitIndext.

Sumbu pembelahan terbaik ChooseSplitAxis dihitung dari sumbu x dan y dari setiap *margin Entry* yang dilakukan dengan meminimasi *margin* berdasarkan batas bawah MBR dan batas atas MBR yaitu Xl,Xu dan Yl,Yu dari susunan MBR{Xl,Xu,Yl,Yu} jadi MBR[0] adalah Xl dan MBR[1] adalah Xu dan MBR[2] adalah Yl dan MBR[3]adalah Yu.

Looping pencarian dalam menentukan *margin* yaitu dilakukan sebanyak dimensi, yang di sini yaitu dua dimensi diwakili oleh Xl,Yl dan Xu,Yu.

Pengisian sml (*sort* MBR *Lower*) dan smu (*sort* MBR *Upper*) di isikan sebanyak jumlah *Entry* MBR yang telah di distribusikan dilanjutkan dengan mengisi penambahan mg yaitu nilai *margin* dari hasil batas atas dan batas bawah di setiap perulangan dari masing-masing distribusi kelompok disetiap *margin* batas atas dan bawah dari setiap dimensi.

Akhir dari perhitungan setiap perulangan dimensi mg selalu di bandingkan dengan minmg, dimana nilai awal minmg adalah maxreal yang apabila pertamakali di bandingkan nilainya selalu terbesar dari mg, karena diharapkan pada awal perulangan mg selalu minimal dari minmg kemudian minmg di isi oleh mg dan dibandingkan kembali dari hasil mg dengan demikian minmg akan selalu berisi nilai *margin* sebelumya yang akan menjadi pembanding dari nilai hasil *margin* baru mg, hingga di ulang sebanyak dimensi.



Gambar 3.14 Prosedur *Choose Split Axis*

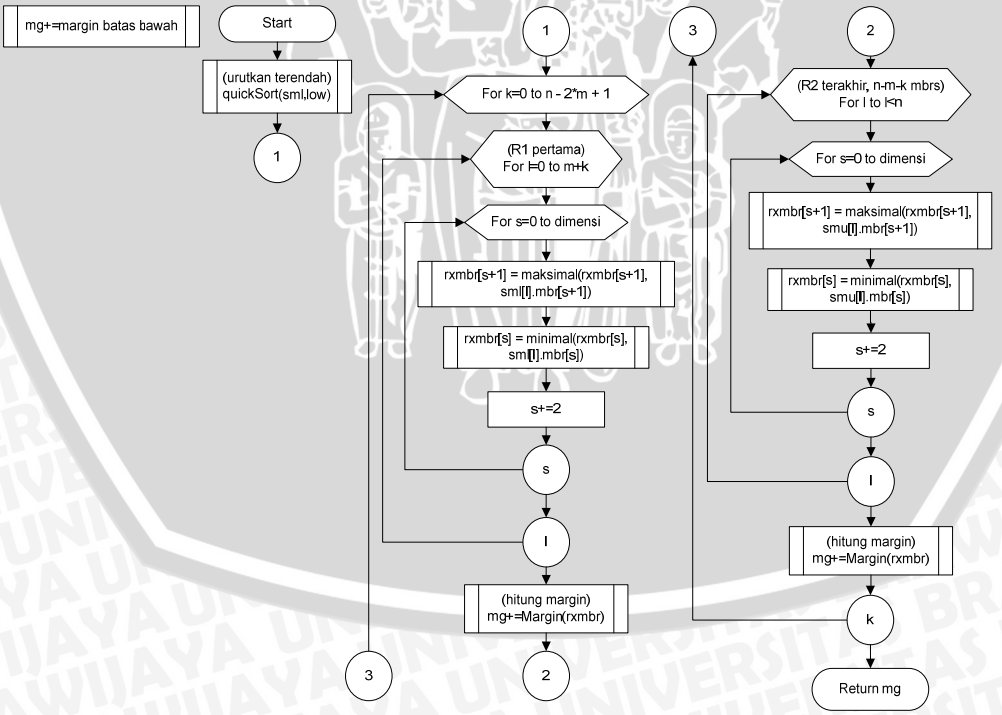
3.3.10 Prosedur *Margin* Batas Bawah

Prosedur *margin* batas bawah pada **Gambar 3.15** digunakan untuk menentukan sumbu pembelahan terbaik berdasarkan sumbu batas bawah MBR, Awal mula untuk menetentukan sumbu pembelahan terbaik dari batas bawah, sml diurutkan dengan nilai terendah dengan quickSort yaitu xl dan yl pad MBR dan begitu sebaliknya dengan smu yaitu xu, yu MBR, kemudian dengan penambahan k+1 hingga n-2*m+1 untuk m+k kelompok pertama yaitu R1 dan kelompok ke dua R2 yaitu n-m-k terakhir, dimana n adalah jumlah *Entry* dan m adalah 40% dari n, hal ini berdasarkan eksperimen hasil terbaik dari Beckmann, dkk (1990).

Pada awal pertama kali menetukan pembatas di setiap kelompok dengan menginisialisasi (rxmbr[s] = MAXREAL) dan (rxmbr[s+1] = MINREAL), rxmbr[s] adalah Xl dan Yl sedangkan rxmbr[s+1] adalah Xu dan Yu, dengan inisialisasi tersebut dimana nilai awal rxmbr[s] adalah maxreal yang bila di bandingkan awal adalah selalu terbesar dari sml[s], karena diharapkan pada awal

perulangan sml[s] selalu bernilai minimal dari rxmbr[s], yang kemudian rxmbr[s] di isi oleh sumbu tiap perulangan dari MBR sml[s] dan dibandingkan kembali dari hasil sml[s] selanjutnya hingga di ulang sebanyak perulangan dimensi dengan penambahan k+1 dari kelompok pertama R1 begitu juga sebaliknya untuk rxmbr[s+1] yang dibandingkan dengan sml[s+1] untuk mencari nilai terbesar Xu dan Yu, setiap akhir dari perulangan kelompok pertama dan kelompok kedua dihitung nilai *margin* berdasarkan persamaan 2.2 dengan menambahkan mg dari perhitungan *margin* akhir dari rxmbr

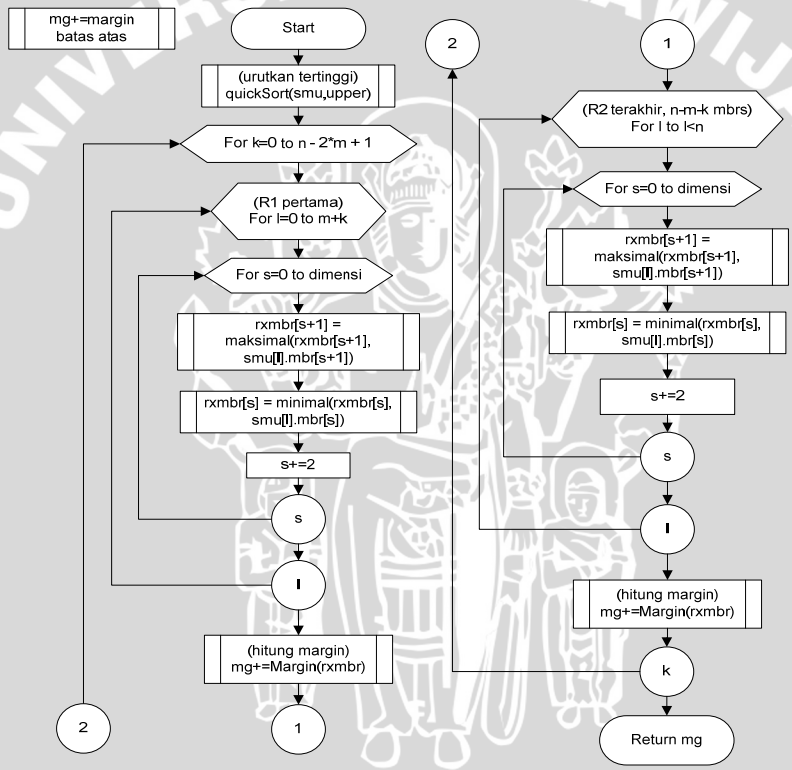
Perhitungan *margin* selanjutnya yaitu menambahkan hasil *margin* dari setiap pembatas kelompok R1 dan R2 di akhir pengelompokan masing-masing di mana *margin* selalu ditambahkan di setiap perulang dari distribusi(k) hingga n-2*m+1 distribusi.



Gambar 3.15 Prosedur *Margin* Batas Bawah

3.3.11 Prosedur *Margin* Batas Atas

Prosedur *margin* batas atas pada **Gambar 3.16** digunakan untuk menentukan sumbu pembelahan terbaik batas atas yaitu menghitung *margin* batas atas yang tidak jauh berbeda dengan cara perhitungan dari *margin* batas bawah pada prosedur sebelumnya, hanya saja array smu yang berisi MBR diurutkan dengan pengurutan tertinggi dari Xu dan Yu dengan quickSort dan proses selanjutnya adalah sama.



Gambar 3.16 Prosedur *Margin* Batas Atas

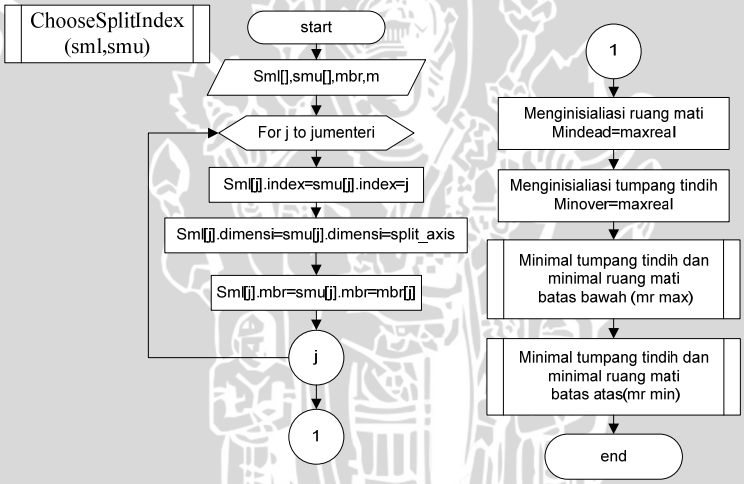
3.3.12 Prosedur Choose Split Index

Prosedur ChooseSplitIndex pada **Gambar 3.17** digunakan untuk menghitung optimasi dari tumpang tindih minimal dan ruang mati berdasarkan persamaan 2.1 dan 2.3 hasil dari pemilihan sumbu pembelahan terbaik prosedur **ChooseSplitAxis**, untuk menemukan optimasi terbaik ditentukan oleh batas atas

dan batas bawah yang diwakili oleh MBR Xl,Yl dan Xu,Yu dengan asumsi *node* 40% penuh berdasarkan eksperimen hasil terbaik dari Beckmann, dkk (1990).

Langkah selanjutnya mengisikan MBR kedalam sml dan smu berdasarkan jumlah *Entry* yang di distribusikan kemudian dimensi dari sml dan smu di isi sesuai dengan hasil *split_axis* terbaik dari minimal *margin* sebelumnya yang telah dihitung di **ChooseSplitAxis.**

Prosedur ChooseSplitIndex selanjutnya memanggil prosedur ruang mati dan tumpang tindih batas bawah dan batas atas untuk menghitung optimasi dan minimasi ruang mati serta tumpang tindih.



Gambar 3.17 Prosedur ChooseSplitIndex

3.3.13 Prosedur Ruang Mati Dan Tumpang Tindih Batas Bawah

Prosedur ruang mati dan tumpang tindih batas bawah pada Gambar **3.18** digunakan untuk menghitung optimasi dan minimasi area mati dan tumpang tindih berdasarkan sumbu batas bawah MBR dengan persamaan 2.1 dan 2.3

Prosedur ruang mati dan tumpang tindih batas bawah untuk menentukan area mati dan tumpang tindih minimal pertama kali sml diurutkan dengan urutan index sumbu terbaik dari **ChooseSplitAxis**, kemudian membagi kelompok

distribusi(k) menjadi dua kelompok, dimana nilai awal k=0 hingga (n-2*m+1) dengan penambahan k+1 untuk kelompok pertama R1 yaitu m+k dan kelompok ke dua R2 yaitu n-m-k terakhir, dimana n adalah jumlah *Entry* dan m adalah 40% dari n, hal ini berdasarkan eksperimen hasil terbaik dari Beckmann, dkk (1990).

Pertamakali sebelum menghitung ruang mati(dead) dan minimal tumpang tindih(over) terlebih dahulu menentukan pembatas kelompok pertama R1 dengan menginisialisasi *over* dan *dead* dengan nol kemudian (rxmbr[s] = MAXREAL) dan(rxmbr[s+1] = MINREAL), dimana rxmbr[s] adalah Xl dan Yl sedangkan rxmbr[s+1] adalah Xu dan Yu, dengan inisialisasi awal rxmbr tersebut nilai rxmbr akan selalu diisi dari hasil perbandingan minimal rxmbr dan sml untuk xl dan yl begitu juga sebaliknya untuk mendapatkan nilai maksimal Xu dan Yu dari perbandingan rxmbr dan sml yang diulang sebanyak dimensi dari tiap-tiap enteri begitu juga untuk kelompok ke dua R2.

Proses selanjutnya menghitung ruang mati minimal yaitu luas area pembatas *entry* dikurangi oleh luas area tiap-tiap *entry* dengan cara memberikan nilai minus pada *dead* untuk penambahan setiap luas area entery sml selama perulangan kelompok *entry* masing-masing dan menambahkannya dengan luas *entry* pembatas (rxmbr) di akhir setiap perulangan dari kelompok masing-masing

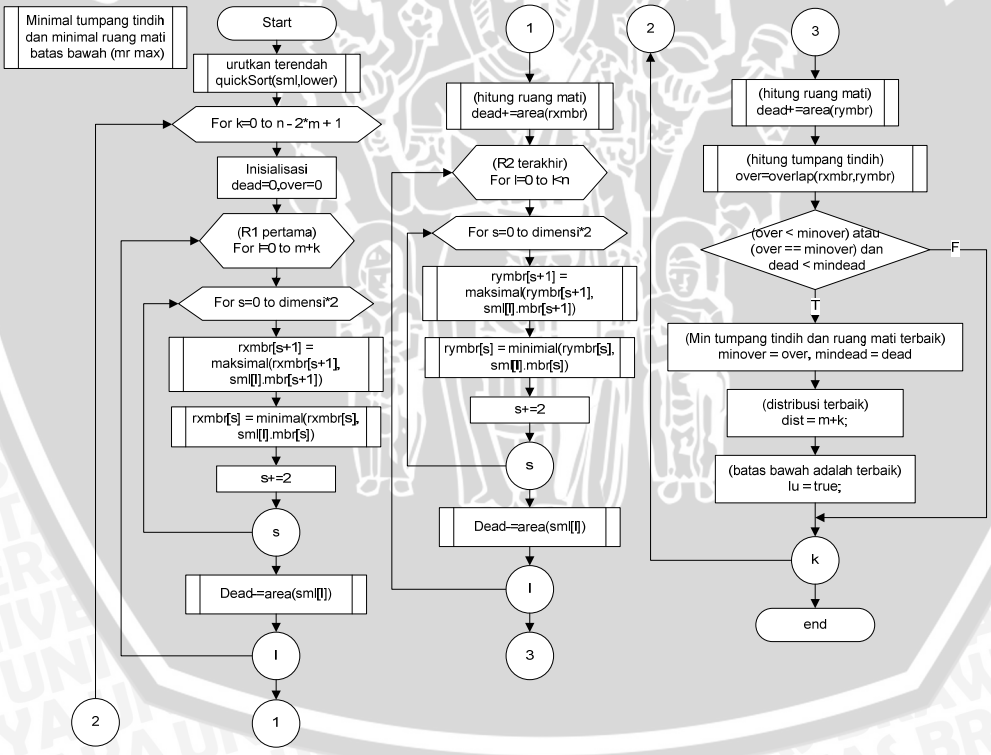
Proses selanjutnya menghitung tumpang tindih minimal dari tiap pembatas MBR kelompok R1 pertama rxmbr dengan pembatas kelompok ke dua R2 terakhir rymbr (*overlap*(rxmbr,rymbr)) di setiap perulangan distribusi(k)

Proses terakhir menghitung optimasi dan minimasi berkenaan dengan sumbu terbaik batas bawah dari tumpang tindih dan ruang mati kelompok R1 dan R2 di setiap distribusi(k), dimana minover dan mindead diisi nilai awalnya dengan

maxreal yaitu dengan membandingkan dead kurang dari mindead dan over kurang dari minover selama perulangan distribusi(k) hingga n-2*m+1 perulangan.

Berkenaan dengan sumbu batas bawah yang di distribusikan(k) adalah pembelahan terbaik selama perulangan n-2*m+1 dengan k+1 penambahan, maka jumlah *entry* terbaik(dist) diisi nilainya dengan *m+k* pada saat prosess tersebut dan tetapkan batas bawah adalah distribusi terbaik (lu=true).

Apabila selama perbandingan dalam perulangan dead sama dengan mindead maka dipilih mindead yang minimal dengan setatemen perbandingan (over < minover) atau (over == minover) dan dead < mindead hal ini berdasarkan eksperimen hasil terbaik dari Beckmann, dkk (1990).

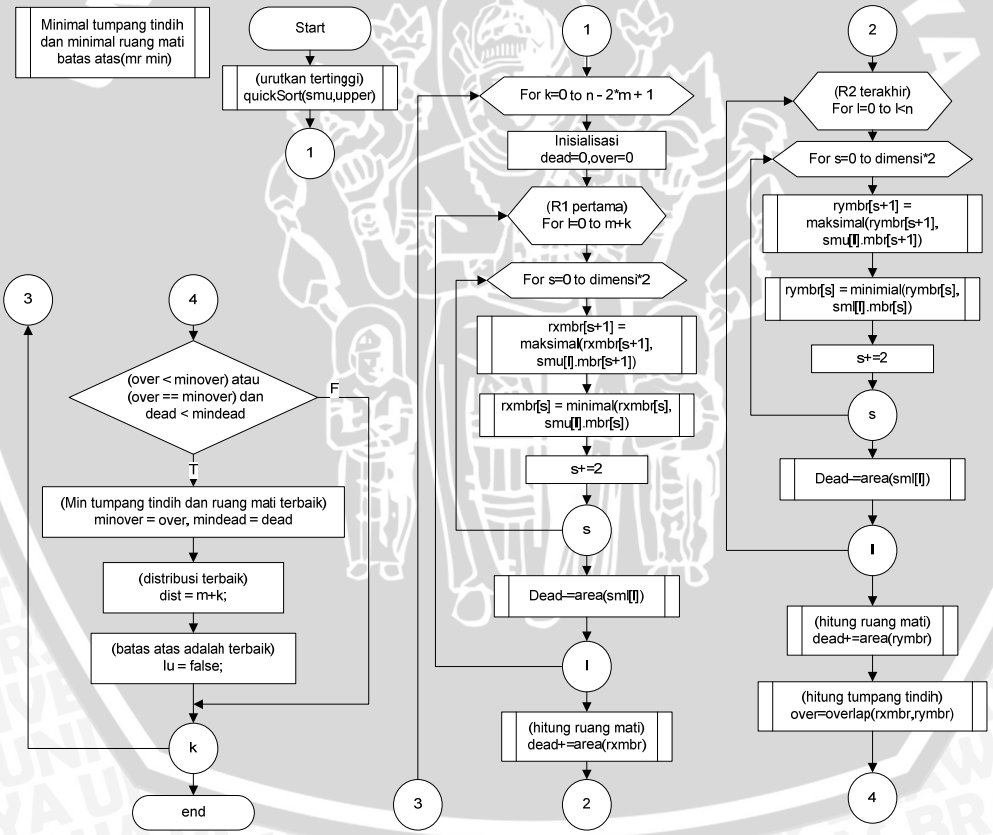


Gambar 3.18 Prosedur Ruang Mati Dan Tumpang Tindih Batas Bawah

3.3.14 Prosedur Ruang Mati Dan Tumpang Tindih Batas Atas

Prosedur ruang mati dan tumpang tindih batas atas pada **Gambar 3.19** digunakan untuk menghitung optimasi dan minimasi area mati dan tumpang tindih berdasarkan sumbu batas atas MBR dengan persamaan 2.1 dan 2.3.

Prosedur ruang mati dan tumpang tindih batas atas tidak jauh berbeda dengan prosedur ruang mati dan tumpang tindih batas bawah, hanya saja untuk smu di urutkan berdasarkan sumbu minimal batas bawah dari **ChooseSplitIndex** menggunakan quickSort. Sebelum menghitung optimasi dan minimasi area mati serta tumpang tindih minimal di lakukan.



Gambar 3.19 Prosedur Ruang Mati Dan Tumpang Tindih Batas Atas

3.3.15 Prosedur QuickSort

Prosedur QuickSort pada **Gambar 3.20** digunakan untuk mengurutkan sekumpulan *Entry* MBR dengan berdasarkan kriteria pengurutan yang diantaranya upper xu,yu, lower xl,yl dan *center* dari nilai pusat MBR x(xu+xl)/2, y(yu+yl)/2

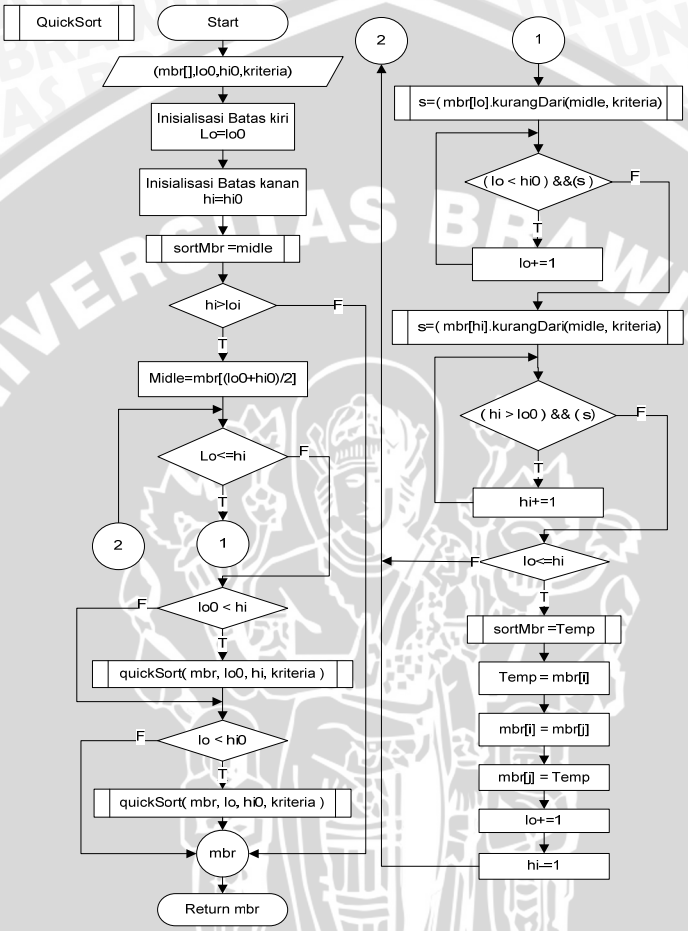
Quicksort pertama kali membagi sekumpulan *entry* menjadi dua pada ruas kanan dan ruas kiri dengan posisi id sebagai kunci masing-masing. Quicksort dimulai dari id yang terkecil untuk ruas kanan dan id yang tertinggi untuk ruas kiri. Jika id pada ruas kanan lebih besar dengan id ruas kiri maka bandingkan berdasarkan kriteria dan id MBR, apakah sumbu pada id ruas kanan lebih besar dengan sumbu id pada ruas kiri.

Prosedur sortMBR yang menghitung berdasarkan kriteria *sorting* dan ruas kiri kurang sama dengan ruas kanan pada saat tersebut maka tambahkan ruas kiri dengan satu begitu juga untuk perbandingan ruas kanan, dengan ruas kanan di tambah satu jika ruas kanan lebih besar dari ruas kiri pada saat tersebut

Proses selanjutnya apakah ruas kiri kurang sama dengan ruas kanan maka pindah posisi sumbu MBR berdasarkan posisi id dari ruas kanan ke ruas kiri, kemudian posisi ruas kiri di tambah satu dan posisi ruas kanan di kurang satu

Proses berulang selama posisi ruas kiri kurang sama dengan posisi ruas kanan selain dari itu maka apakah posisi ruas kiri pada saat tersebut kurang dari posisi ruas kanan bila iya panggilan rekursif QuickSort dilakukan untuk mengurutkan berdasarkan kriteria dengan posisi ruas kiri pada saat tersebut dengan ruas kanan, begitu sebaliknya jika ruas kiri kurang dari posisi ruas kanan pada saat tersebut rekursif untuk *sorting* di lakukan berdasarkan kriteria pada posisi ruas kiri dengan posisi ruas kanan pada saat tersebut.

Sorting akan berakhir apabila data sudah terurut ber dasarkan kriteria dengan posisi yang telah dilalui untuk di urutkan dengan pengecekan ruas kanan lebih besar dari ruas kiri dengan posisi ruas kanan tertinggi dan ruas kiri terendah.



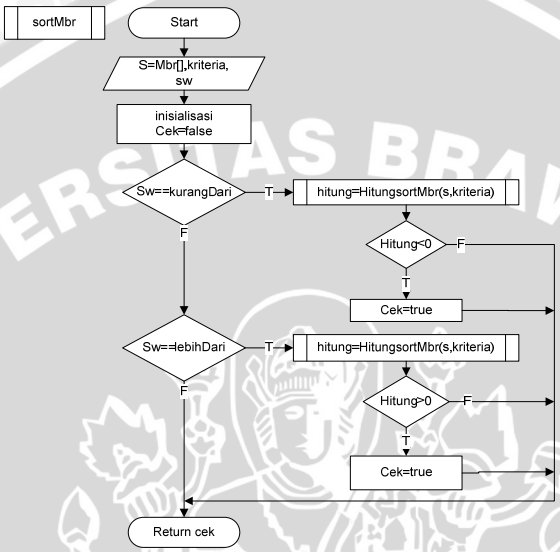
Gambar 3.20 Prosedur Quick Sort

3.3.16 Prosedur Sort MBR

Prosedur sortMBR pada **Gambar 3.21** digunakan untuk perbandingan *sorting* quickSort dengan menghitung batas-batas sumbu MBR berdasarkan kriteria hasil perhitungan dari prosedur Hitung sort MBR

Proses pengecekan yang di hasilkan dari perhitungan prosedur Hitung sort MBR dilakukan dengan membandingkan suatu sumbu lebih atau kurang dari

sumbu yang lain berdasarkan kriteria *sorting* yang di lakukan oleh quickSort dari hasil perhitungan Hitung sortMBR dengan pengembalian true atau false dari hasil pengecekan, hasil pengembalian sort MBR selanjutnya digunakan oleh prosedur quicksort untuk *sorting* MBR.



Gambar 3.21 Prosedur Sort MBR

3.3.17 Prosedur Hitung Sort MBR

Prosedur hitung sortMBR pada **Gambar 3.23** digunakan untuk menghitung batas-batas sumbu MBR berdasarkan kriteria dan hasil perhitungan dalam perbandingan Sort MBR.

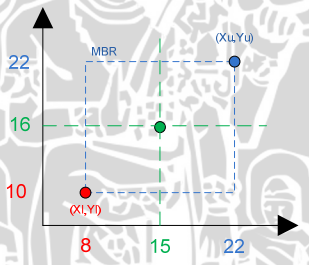
Kriteria dalam perbandingan diantaranya batas terendah(LowerMBR), batas tertinggi(UperMBR),dan pusat MBR yang digunakan untuk *sorting* jarak pusat ke pusat MBR (*center*MBR).

Dimensi adalah *iterasi* 2 kali dari dimensi itu sendiri yang di *iterasi*-kan dalam prosedurQuicksort dengan nilai awal nol dengan penambahan dua hinga sebanyak dimensi.

Prosedur sortMBR jika pengecekan kriteria adalah lowerMBR maka yang dihitung adalah berkenaan dengan batas bawah (lower bound) dari MBR untuk dibandingkan, yaitu xl dan yl, pada *iterasi* pertama MBRxl – Sxl, dan *iterasi* kedua MBRyl-Syl.

Prosedur sortMBR jika pengecekan kriteria adalah uperMBR maka yang dihitung adalah batas atas (upper bound) dari MBR untuk dibandingkan yaitu xu dan yu, pada *iterasi* pertama MBRxu – Sxu, dan *iterasi* kedua MBRyu-Syu.

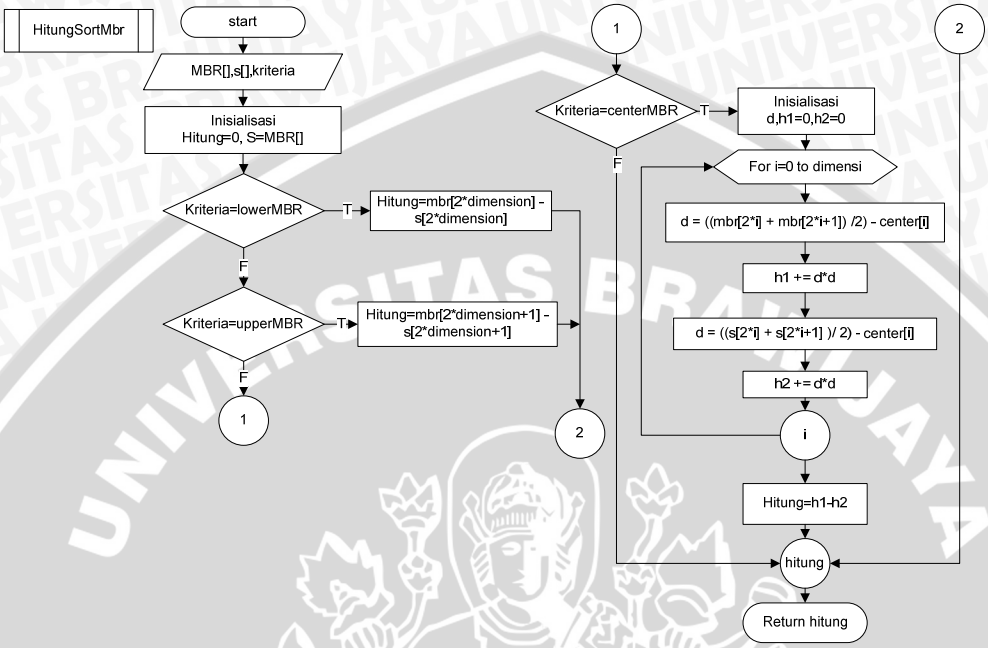
Prosedur sortMBR jika pengecekan kriteria adalah *center*MBR maka membandingkan jarak pusat ke pusat yang menghitung nilai pusat sumbu dari MBR untuk dibandingkan yaitu xu+xl/2 dan yu+yl/2, seperti di gambarkan pada **Gambar 3.22** ilustrasi pusat dimana pusat x = (22+10/2) dan pusat y= (22+8/2) :



Gambar 3.22 Ilustrasi Pusat MBR

Pada *iterasi* pertama ((MBRxu + MBRxl)/2)-*CENTER*xl, hasil perhitungan ini di kuadrartkan dan disimpan dalam h1 dengan penambahan dari h1 sebelumnya di tiap perulangan begitu juga untuk nilai pembanding S yang mewakili MBR pembanding yang hasil perhitungannya di simpan dalam h2, kemudian untuk *iterasi* kedua sama halnya dengan *iterasi* pertama hanya saja sumbu yang dihitung adalah sumbu y.

Nilai akhir dari perhitungan untuk perbandingan jarak dari pusat ke pusat dikembalikan dengan h1 dikurangi oleh h2.



Gambar 3.23 Perhitungan Sort MBR

3.3.18 Prosedur *Overlap*

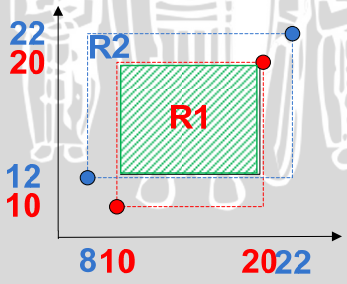
Prosedur *Overlap* pada Gambar 3.25 digunakan untuk menentukan kemungkinan terdapat tumpang tindih diantara MBR dari R1 dan R2 yang digunakan untuk menentukan pemilihan sumbu *split* dan ChooseSubtree dengan meminimasi tumpang tindih.

Prosedur *overlap* berdasarkan persamaan 2.3 pada awal perhitungan dilakukan *iterasi* sebanyak dimensi, dimana setiap *iterasi* digunakan untuk membandingkan antara sumbu MBR apakah *inside* dari batas bawah ataukah *inside* dari batas atas dan menghitung luas dari hasil irisan tumpang tindih tersebut dengan perkalian sisi-sisinya (Ub-Lb) dari setiap *iterasi*, yaitu hasil *iterasi* sebelumnya dikalikan dengan hasil *iterasi* sesudahnya, perhitungan *iterasi* adalah

sum*= Ub-Lb, dimana susunan ub(*Upper bound)* dari MBR R1 maupun R2 adalah $R1x_u$, $R1_{Yu}$, $R2x_u$, dan $R2_{Yu}$ sedangkan lb(*lower bound)* dari MBR R1 maupun R2 adalah $R1x_l$, $R1_{Yl}$, $R2x_l$, $R2_{Yl}$.

Pada *iterasi* pertama menentukan terlebih dahulu sumbu X yang dilanjutkan *iterasi* kedua sumbu Y, setiap *iterasi* membadingkan apakah batas atas R1 berada dalam R2, jika iya apakah batas bawah R1 juga dalam R2, maka tentukan sisi-sisinya dari batas atas R2ub di kurangi batas bawah R1lb. Jika bukan apakah R1 berisi R2 (R1 *containt* R2), jika iya tentukan sisi-sisinnya dari R2ub dikurangi R2lb. Jika pengecekan sejauh ini bukan *inside* atau *contain* apakah hanya batas atas R1ub berada dalam R2ub, dan batas bawah R1lb dalam R2lb, jika benar tentukan sisi-sisinnya dari batas atas R1ub di kurangi batas bawah R1lb jika bukan tentukan sisi-sisinnya dari batas atas R1ub dikurangi batas bawah R2lb

Dikatakan MBR R1 ***overlap*** MBR R2 apabila salah satu sisi MBR dari R1 berada dalam MBR R2 seperti di tunjukkan pada **Gambar 3.24** berikut dimana salah satu sisi R1 berada dalam R2 :



Gambar 3.24 MBR1 *overlap* dengan MBR2

Inside terhadap sumbu X :

$$(R1_{Xl} = 10) \geq (R2_{Xl} = 8) \&\& (R1_{Xu} = 20) \leq (R2_{Xu} = 22)$$

Pengecekan benar maka Sisi X adalah (R1ub-R1lb) =20-10

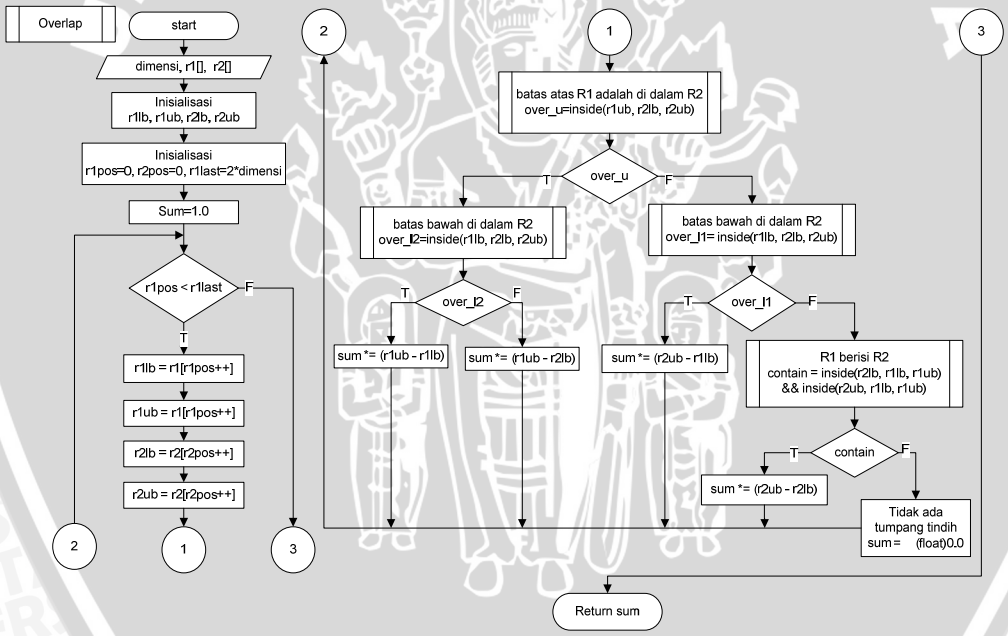
Inside terhadap sumbu Y :

$$(R1_{Yl} = 10) \geq (R2_{Yl} = 12) \&\& (R1_{Yu} = 20) \leq (R2_{Yu} = 22)$$

Pengecekan salah maka Sisi Y (R1ub-R2lb)=20-12

Luas irisan dari R1 dengan R2 yaitu (sisi X * sisi Y) adalah 80cm^2

Apabila kondisi pengecekan tidak ada *inside* atau *contain* antara R1 dan R2 maka sum menyimpan sisi suatu MBR dikembalikan dengan nol, dan sebaliknya apabila terjadi *inside* atau *contain* maka sum dikembalikan sesuai dengan perhitungan luas dari sisi-sisi irisan antara *lower bound* maupun *upper bound* yang *inside* maupun *contain.*



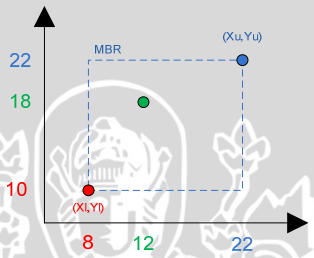
Gambar 3.25 Prosedur *overlap*

3.3.19 Prosedur *Inside*

Prosedur *Inside* pada **Gambar 3.27** digunakan untuk pengecekan dari suatu *point* apakah berada dalam batas atas atau batas bawah MBR dengan membandingkan nilai xl, xu, yl atau yu dimana perbandingan untuk memanggil

prosedur ini di asumsikan bahwa *point* x harus dibandingkan dengan sumbu x dari MBR atau *point* y dengan sumbu y pada MBR.

Point(x,y) dikatan *Inside* apabila salah satu *point* berada dalam persegi dengan membandingkan *point* lebih besar sama dengan *lower bound* atau batas bawah dari persegi MBR dan *point* kurang sama dengan *upper bound* atau batas atas persegi MBR seperti di tunjukkan pada ilustrasi **Gambar 3.26** berikut dimana *point* berada dalam persegi pada sumbu x dan y :



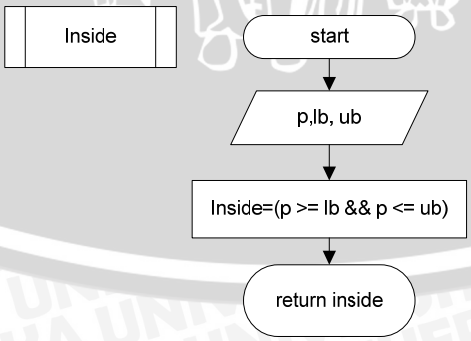
Gambar 3.26 *Point* di dalam ruang MBR

Pada sumbu x :

$$(p = 12) \geq (lb_{Xl} = 8) \&\& (p = 12) \leq (ub_{Xu} = 22)$$

Pada sumbu y :

$$(P_Y = 18) \geq (lb_{Yl} = 10) \&\& (P_Y = 18) \leq (ub_{Yu} = 22)$$

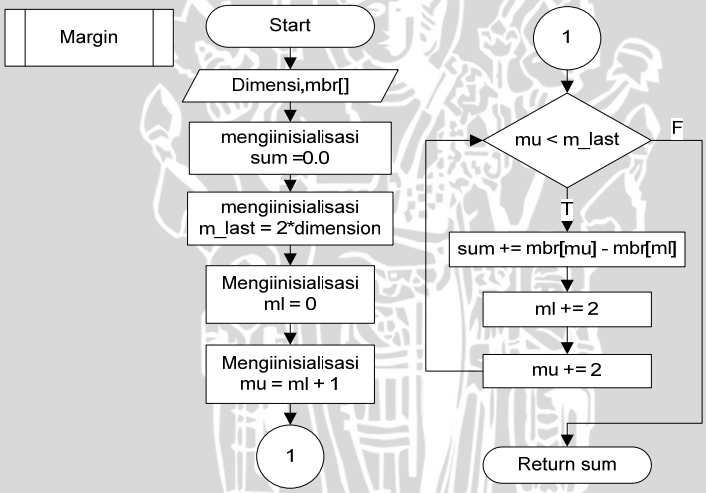


Gambar 3.27 Prosedur *inside*

3.3.20 Prosedur *Margin*

Prosedur *margin* pada **Gambar 3.**28 adalah digunakan untuk menentukan sumbu pembelahan pada persamaan 2.1 oleh prosedur ChooseSplitAxis dengan menghitung besar sumbu dari garis tepi sisi x dan y MBR, sumbu *margin* pada MBR diwakili oleh{Xl,Xu,Yl,Yu}, maka *margin* = Xu-Xl +Yu-Yl.

Hasil *margin iterasi* pertama dari xu-xl di tambahkan sum dimana sum awal adalah 0 maka sum nilainya adalah *margin* pertama, kemudian saat *iterasi* ke dua dimana mu=1+2 dan ml =0 +2 didapatkan sumbu dari MBR adalah yu-yl, hasil dari *iterasi* kedua di tambah kan dengan hasil *margin* pertama yang terdapat pada sum, sum dikembalikan dengan hasil perhitungan total *margin* dari MBR.



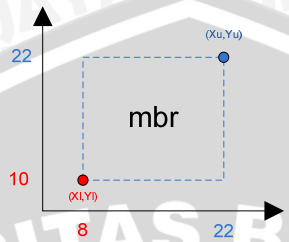
Gambar 3.28 Prosedur *margin*

3.3.21 Prosedur Area

Prosedur area adalah digunakan untuk menghitung luas dari salah satu MBR pada persamaan 2.1 yang di tunjukkan pada **Gambar 3.30** , luas area MBR didapatkan dengan mengalikan sisi-sisi MBR.

Perulangan dimensi pada prosedur ini digunakan untuk mendapatkan sisi-sisi dari MBR, dengan susunan MBR {Xl,Xu,Yl,Yu}, maka didapatkan sisi

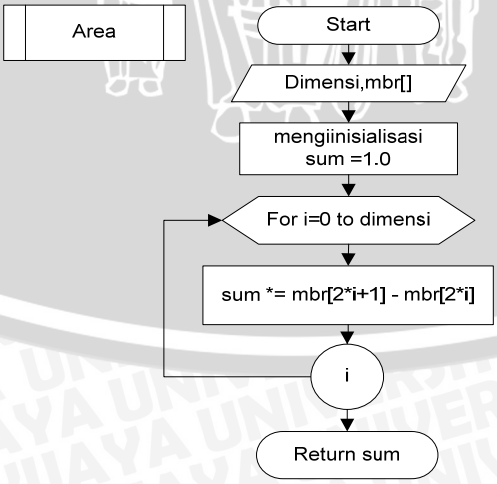
panjang dari suatu MBR pada *iterasi* pertama adalah MBR_{Yu} - MBR_{Yl}, sedangkan untuk mendapatkan sisi lebar pada *iterasi* ke dua adalah MBR_{Xu} - MBR_{Xl} seperti ilustrasi pada **gambar 3.29** berikut untuk menentukan sisi-sisi MBR :



Gambar 3.29 Area MBR

P=MBR_{Yu} - MBR_{Yl}, maka P=12 dan $L = MBR_{Xu}$ - MBR_{Xl} maka L=14, Dengan demikian luas MBR adalah PxL = 168 cm^2

Hasil perhitungan panjang dari MBR dikalikan dengan variable sum, dimana nilai awal sum adalah satu sehingga nilai sum adalah nilai panjang itu sendiri, pada *iterasi* kedua di dapatkan lebar dari MBR, kemudian hasilnya di kalikan oleh sum yang berisi nilai panjang MBR sebelumnya, dengan demikian luas MBR di simpan dalam sum dan dikembalikan.



Gambar 3.30 Prosedur area

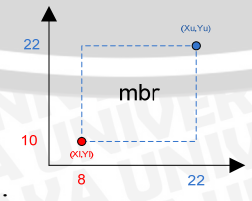
3.3.22 Prosedur Enlarge

Prosedur Enlarge pada **Gambar 3.32** digunakan untuk memperluas pembatas MBR (*bounces* MBR) yang memerlukan pembesaran untuk menempatkan data baru MBR, Perulangan untuk pengecekan adalah 2*dimensi, dimana dua dimensi menyimpan 4 sumbu xl,xu,yl dan yu

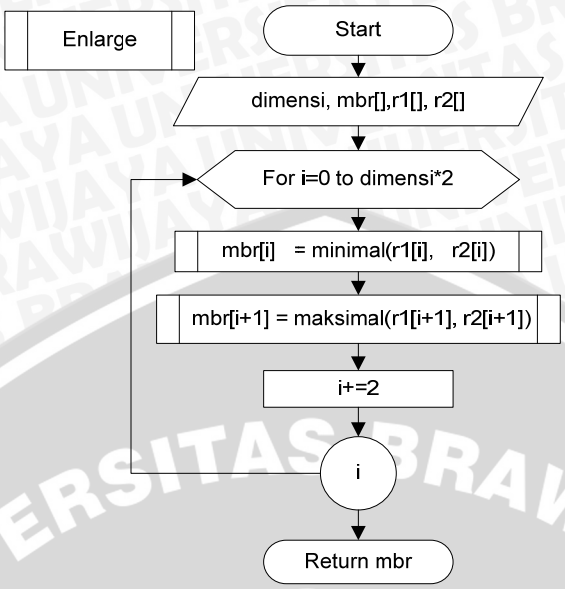
MBR yang dikembalikan adalah hasil perluasan dari setiap sumbu xl,xu, dan yl yu, pada *iterasi* pertama MBR xl di isi nilai minimal xl dari perbandingan sumbu r1xl dengan r2xl, dan MBR xu di isi dari perbandingan nilai maksimal dari r1xu dengan r2 xu, kemudian *iterasi* ke dua MBR yl di isi dari nilai minimal yl dari perbandingan sumbu r1yl dengan r2yl, dan MBR yu di isi dari perbandingan nilai maksimal dari r1yu dengan r2 yu.

R1 adalah MBR yang nantinya di isikan dalam MBR *bounces* sedangkan R2 adalah *Entry bounces* yang memerlukan pembesaran untuk menempatkan data baru MBR R1.

Kategori perbandingan dibagi menjadi dua yaitu nilai minimal untuk membandingkan xl dan yl dan nilai maksimal untuk menentukan xu dan yu, untuk mencari area yang terluas maka semakin kecil xl dan yl dan semakin besar nilai xu dan yu area yang di dapatkan juga akan semakin luas, karena di dalam suatu MBR di wakili oleh xl yl pada sudut bawah MBR dan xu yu pada sudut atas MBR seperti ilustrasi **Gambar 3.31** berikut



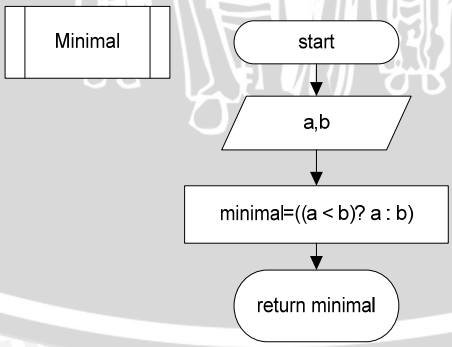
Gambar 3.31 Ilustrasi perluasan sumbu MBR



Gambar 3.32 Prosedur Enlarge

3.3.23 Prosedur Minimal

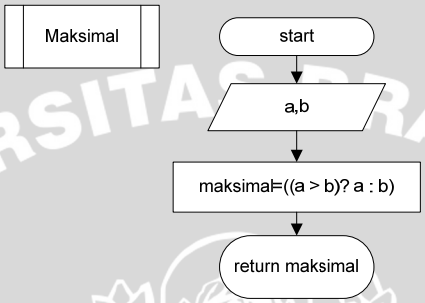
Prosedur minimal pada **Gambar 3.33** digunakan untuk menentukan sumbu minimal dengan sumbu lain untuk melakukan pengecekan dari sumbu apakah nilai sumbu lebih kecil dari sumbu lain dengan membandingkan kedua variable a dan b jika a kurang dari b maka a adalah minimal dari b dan juga sebaliknya.



Gambar 3.33 Prosedur Minimal

3.3.24 Prosedur Maksimal

Prosedur maksimal **Gambar 3.34** digunakan untuk menentukan sumbu maksimal dengan sumbu lain untuk melakukan pengecekan dari sumbu apakah nilai sumbu lebih besar dari sumbu lain dengan membandingkan kedua variable a dan b jika a lebih besar dari b maka a adalah maksimal dari b dan juga sebaliknya.



Gambar 3.34 Prosedur Maksimal

3.3.25 Prosedur *Insert* Dirnode

Prosedur *insert* Dirnode pada **Gambar 3.35** adalah digunakan untuk menyisipkan ke *node* internal atau *node* nonleaf hasil dari prosedur ChooseSubtree

Pada saat *insert* Dirnode ini dipanggil, MBR yang dimasukkan ke Dirnode di lakukan pengecekan di **ChooseSubtree** untuk beberapa optimasi dan minimasi

ChooseSubtree untuk memilih sebuah *node* daun yang tidak bisa menampung *Entry* baru semisal daun sudah memiliki jumlah *Entry* maksimum, R*-tree ini tidak langsung meresor untuk membelah *node*, sebaliknya dengan menemukan sebagian kecil dari *Entry* dari *node* yang meluap lihat pada prosedur **ChooseSubtree** dan hasilnya di kembalikan dari **ChooseSubtree** untuk di masukkan berdasarkan *pointer* dari *node* dengan memanggil getSon() berdasarkan

pointer Entry dari hasil ChooseSubtree. kemudian me*reinsert*s ke *Entry* yang meluap tersebut

Dari hasil memasukkan kembali memungkinkan terjadi *split* atau *reinsert* data, jika terjadi *split* atau *reinsert*, maka berdasarkan MBR yang dimasukkan dilakukan proses reintregasi yaitu menyeimbangkan struktur *tree*, dengan memasukkan data pembatas anak dan menciptakan *root* baru utuk pembatas anak tersebut, yaitu *parent* dengan MBR bagi pembatas anak-anaknya (MBR *bounces*) yang di simpan dalam obyek MBR.

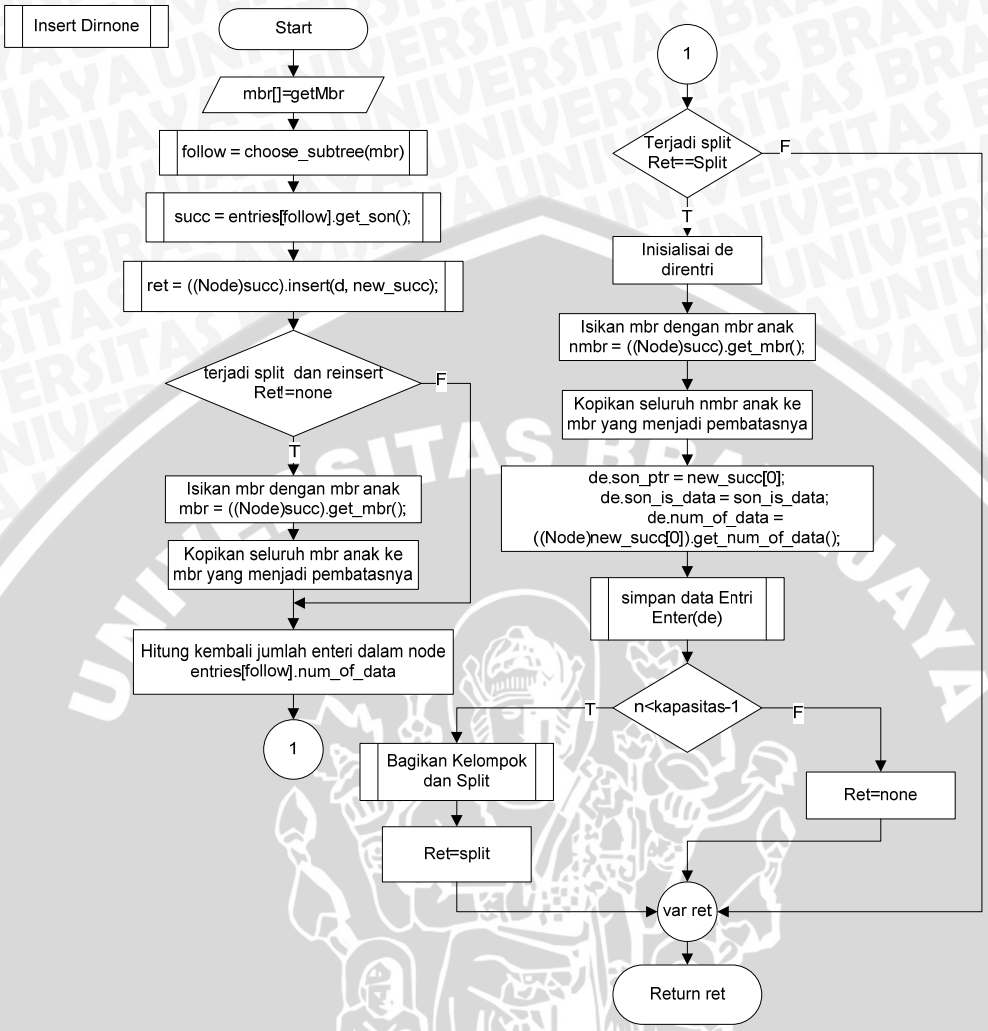
Kondisi selanjutnya bila terjadi *split*, *Entry* dengan obyek de atau yang di inisilisasi dari prosedur Entries di isi dengan informasi *node* MBR yang di inisialisasi dan menyimpan data anak sebelumnya dengan son isdata adalah true dan jumlah data anak di isi sesuai jumlah anak-anak yang di batasinya tersebut.

Data anak ini sebagai *parent* dari anak-anaknya yang proses sebelumnya di simpan dalam MBR, kemudian menyimpan MBR *parent* kedalam *node* dengan memanggil prosedur enter.

Setelah proses pemasukan *node* pada prosedur enter jumlah *Entry* bertambah satu, kemudian dilakukan pengecekan apakah kapasitas penyimpanan dalam *node* melebihi kapasitas.

Kapasitas dalam *node Entry* adalah $2 \leq m \leq M/2$, bila pemasukan *Entry* melebihi kapasitas lakukan *split* dengan memanggil prosedur pembagian kelompok *split* untuk dilakukan *split* dan kembalikan penanda *split* karena terjadi *split* bila tidak maka kemblikan penanda *none* karena tidak terjadi *split*.

Pemberian penanda *split* atau none ini nantinya digunakan oleh prosedur insertData untuk pengecekan akibat pemasukan, lihat pada prosedur insertData.



Gambar 3.35 Prosedur *Insert* Dirnode

3.3.26 Prosedur *Choose* Subtree

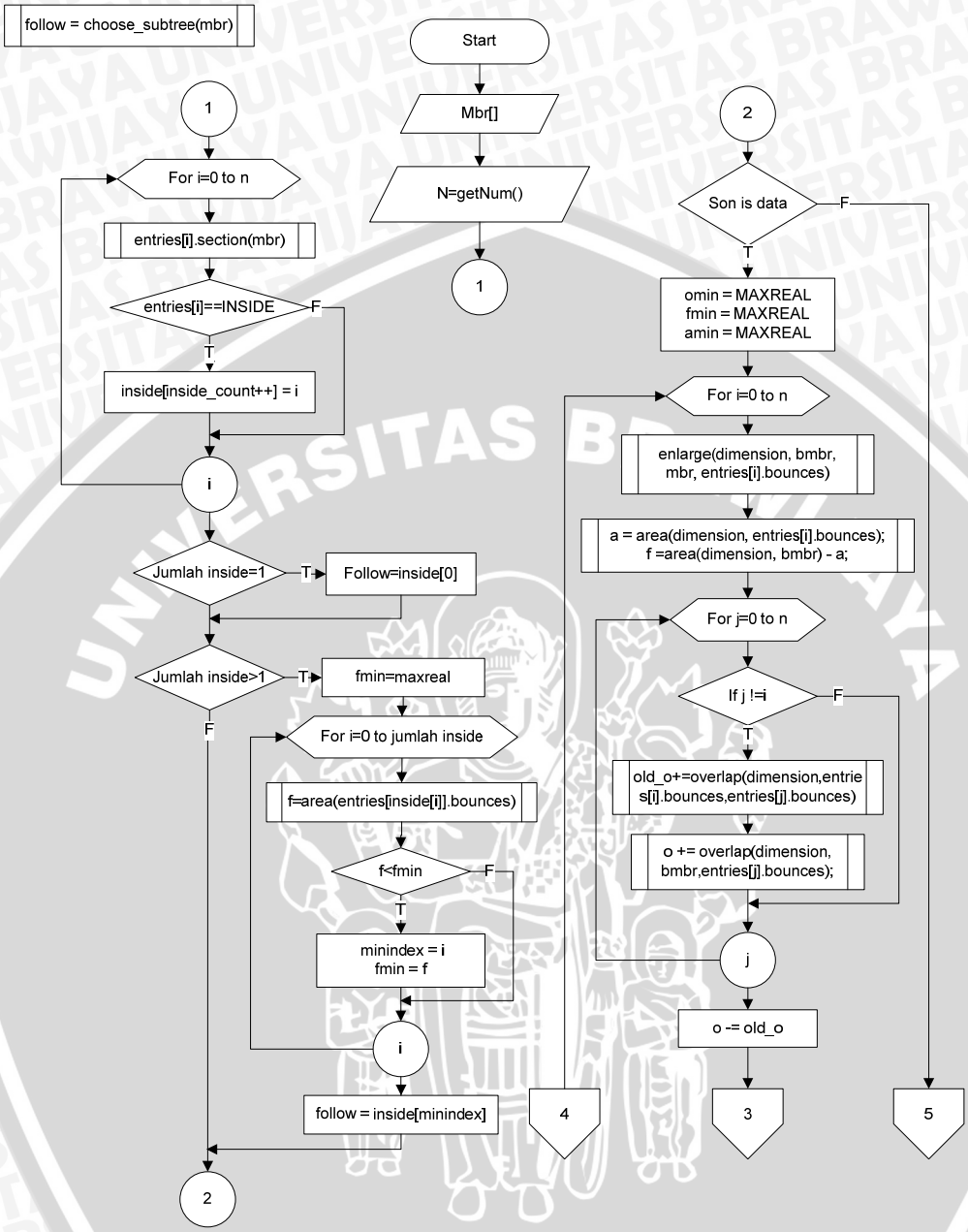
Prosedur ChooseSubtree pada **Gambar 3.36** dan **3.37** digunakan untuk menentukan *node* dari level *tree* yang akan di sisipi *node Entry* baru, sebelum penyisipan tersebut ChooseSubtree mempertimbangkan kriteria minimasi perluasan area dan meminimasi tumpang tindih, karena hasil eksperimen dari Beckmann, dkk (1990), menunjukkan bahwa kriteria ini adalah yang terbaik daripada yang lain.

ChooseSubtree pertamakali untuk menyisipkan *Entry* baru mengidentifikasi MBR *node* di level *tree* terhadap *Entry* baru apakah *inside* atau *overlap* dengan prosedur *section*.

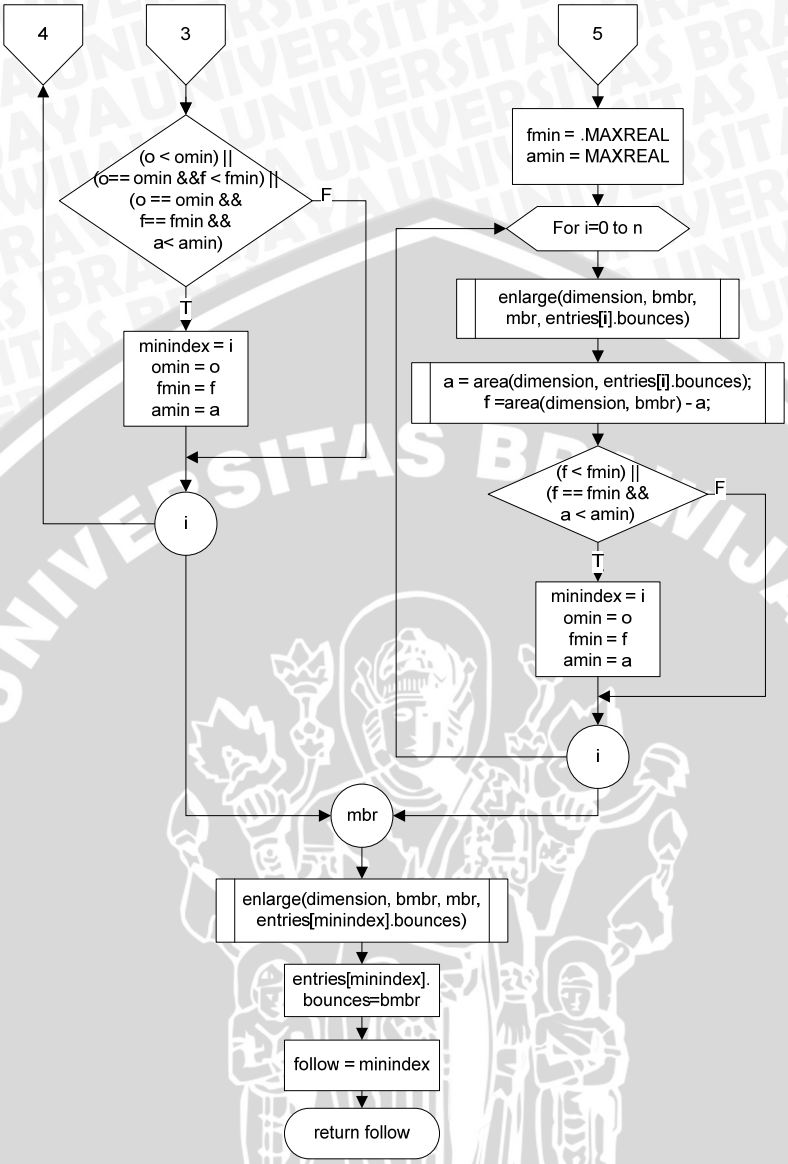
Proses ChooseSubtree dari hasil pengecekan terhadap *node* dalam *tree*, bila *Entry* baru berada di salah satu *node* tersebut maka *Entry* baru di masukkan ke dalam *node* tersebut (Follow = *inside*[0]), jika *Entry* baru berada di dalam ruang lebih dari satu *node* anak maka mengambil ruang yang pembesarannya paling kecil dan memasukkan *Entry* baru ke dalam *node* tersebut yang areanya paling kecil (Follow=*inside*[min*index*]).

Bila mana *Entry* baru tidak berada di dalam salah satu *node* dan *Entry* masukan *childpointers* di titik N pada daun maka hitung optimasi pembesaran area dan tumpang tindih dari setiap *node* terhadap *Entry* baru, bila mana *Entry* baru adalah *node* noneleaf *childpointers* di N tidak menunjuk ke daun maka hitung pembesaran area yang minimal, susunan pembesaran *node* MBR terhadap *entry* baru adalah (Xl_{min}, Xu_{maks}, Yl_{min}, Yu_{maks}).

ChooseSubtree terakhir memilih salah satu dari MBR *node* yang akan di tempati *Entry* baru dimana *Entry* pembesaran MBR di antara *Entry* saudara di setiap *node* noneleaf untuk memasukkan *Entry* baru dengan area pembesaran terkecil serta tumpang tindih minimal (Follow=min*index*).



Gambar 3.36 Prosedur ChooseSubtree

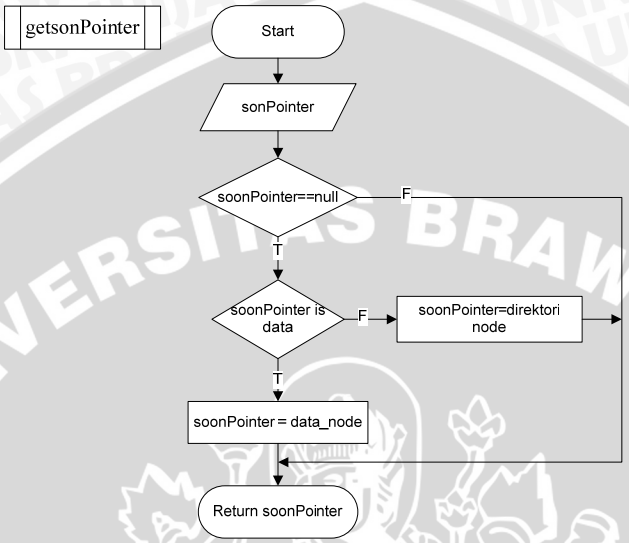


Gambar 3.37 Lanjutan prosedur ChooseSubtree

3.3.27 Prosedur GetSoonPointer

Prosedur GetSoonPointer pada **Gambar 3.38** digunakan untuk mendapatkan *pointer* anak dari *parent* MBR yang mencangkupi MBR *node* anak, pada saat prosedur getson*Pointer* dipanggil mula-mula mengecek apakah *pointer* anak masih kosong bila iya dilakukan pengecekan apakah *node pointer* mengarah ke data *node*, dalam hal ini menunjuk ke *node* leaf bila iya maka soon *pointer*

mengarah ke data*node* bila tidak maka mengarah ke direktori *node*, direktori *node* adalah node *nonleaf* (*node* internal) yang berisi *parent* untuk mencangkupi dan membatasi MBR dari node anak.



Gambar 3.38 Prosedur untuk mendapatkan *pointer* anak

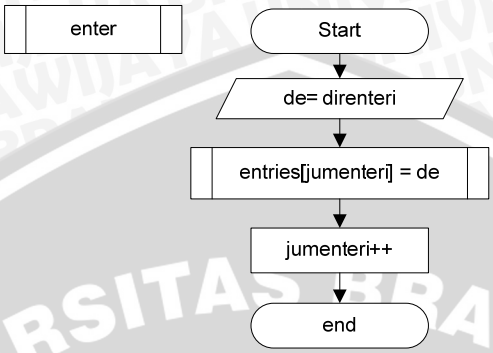
3.3.28 Prosedur Enter

Prosedur enter yang di gambarkan pada **Gambar 3.39** akan selalu dipanggil apabila sebuah data diperlukan untuk di masukkan dalam *node* dan atau data diperlukan untuk perubahan dalam *node* yang disebabkan *split* atau *reinsert* dari sebuah *node* MBR.

Obyek de menyimpan informasi *node* yang akan dimasukkan dalam *tree* dengan prosedur Entries, dimana posisi *Entry* yang akan dimasukkan dalam *tree* adalah jumlah *Entry node* yang terakhir dimasukkan dalam *tree* yang sesuai pada saat tersebut.

Setelah penyimpanan tersebut jumlah *Entry* ditambah satu, yang memungkin terjadi overflow untuk *split* atau *reinsert* pada proses kemudian saat

pemasukan data baru dimana jumlah *Entry* dipanggil dengan getnum() yang mengembalikan jumlah *Entry* pada saat tersebut.

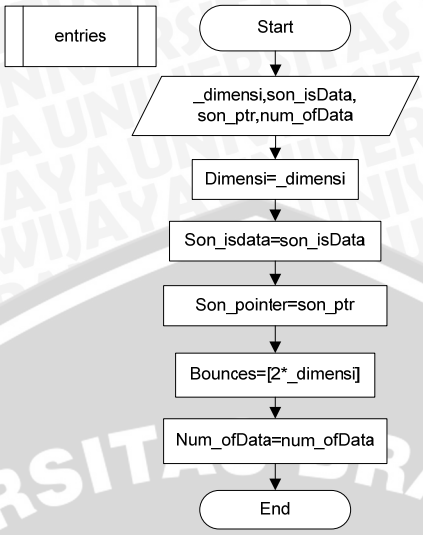


Gambar 3.39 Prosedur Enter

3.3.29 Prosedur Entries

Prosedur Entries pada **Gambar 3.40** digunakan untuk menyimpan informasi dari suatu *node* saat data disimpan dalam *node*, dimana informasi *Entry* yang diperlukan dalam penyimpanan *node* diantaranya yaitu 2 dimensi yang diwakili oleh Xl,Yl dan Xu,Yu, kemudian son_isdata bertipe bolean jika *node* tersebut adalah data anak dari sebuah *parent*, selanjutnya son_*pointer* yaitu *pointer* anak yang menunjuk ke sebuah *node* yang dimiliki dari *node* tersebut

Kemudian *bounces* yaitu di alokasikan 2*dimensi dimana tiap dimensi menyimpan koordinat x da y dari pembatas *Entry*, karena berdimensi 2 maka alokasi penyimpanan di kalikan 2 untuk menyimpan data yang bertipe float persegi pembatas dari *node* tersebut dengan berturut-turut {Xl,Xu,Yl,Yu}.



Gambar 3.40 Prosedur Entries

3.3.30 Prosedur Section

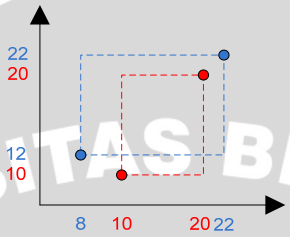
Prosedur section pada **Gambar 3.43** digunakan untuk mengetahui apakah *Entry* MBR yang akan dimasukkan di dalam sebuah *node tree* berada dalam(*inside*) *node* MBR yang terdapat dalam *node tree* tersebut, prosedur ini digunakan oleh ChooseSubtree untuk mengecek *entry* baru sebelum dimasukkan dalam *tree* dengan susunan MBR$\{X_l,\ X_u, Y_l, Y_u\}$.

Pada saat MBR di lakukan pengecekan dengan MBR lain disini adalah *bounces* yaitu pembatas dari suatu persegi yang meliputi data dari persegi anak.

Variable *inside* dan *overlap* pertamakali dikondisikan true kemudian dilakukan perhitungan antara MBR dengan *bounces* dan mengasumsikan salah satu sisi dari kedua persegi tidak saling bertemu.

Dikatan *overlap* jika MBR Xl kurang dari *bounces* Yl atau sebaliknya MBR Yl lebih dari *bounces* Xl berarti salah satu dari sisi persegi saling tumpang tindih dan juga MBR Xu kurang dari *bounces* Yu atau sebaliknya MBR Yu lebih besar dari *bounces* Xu yang berarti salah satu dari sisi persegi saling tumpang

tindih selain dari kondisi tersebut atau dengan membalikkan kondisi di masing-masing perbandingan maka salah satu sisi tidak akan saling bertemu dan nilai dari *overlap* adalah *false* seperti ditunjukkan pada **Gambar 3.41** berikut untuk nilai *overlap* adalah *true* (saling tumpang tindih) :



Gambar 3.41 Ilustrasi MBR tumpang tindih dengan MBR Lain

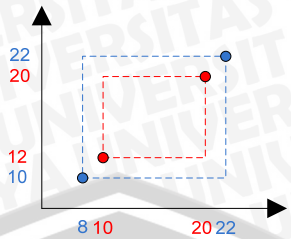
Pada *iterasi* pertama pengecekan terhadap batas bawah :

$$(MBR_{Xl} = 10) < (BOUNCES_{Yl} = 12) \parallel (MBR_{Yl} = 10) > (BOUNCES_{Xl} = 8)$$

Pada *iterasi* ke dua pengecekan terhadap batas atas :

$$(MBR_{Xu} = 20) < (BOUNCES_{Yu} = 22) \parallel (MBR_{Yu} = 20) > (BOUNCES_{Xu} = 22)$$

Dikatakan ***Inside*** apabila salah satu persegi berada dalam persegi yang lain dengan membandingkan MBR_{Xl} lebih besar dari $BOUNCES_{Xl}$ atau MBR_{Xu} kurang dari $BOUNCES_{Xu}$ dan juga untuk *iterasi* ke dua pada MBR_{Yl} lebih besar dari $BOUNCES_{Yl}$ atau MBR_{Yu} kurang dari $BOUNCES_{Yu}$ dengan membalikkan kondisi tersebut maka nilai *inside* adalah false karena dari persegi tidak berada dalam persegi yang lain, seperti di tunjukkan pada **Gambar 3.42** berikut dimana persegi berada dalam persegi lain :



Gambar 3.42 Ilustrasi MBR *inside* MBR Lain

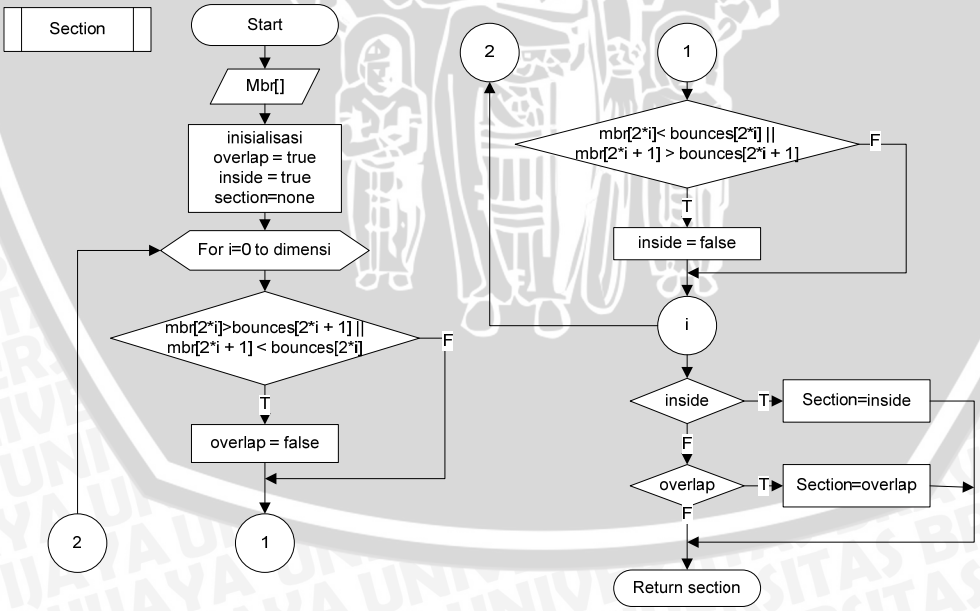
Pada *iterasi* pertama pengecekan terhadap batas bawah :

$(MBR_{Xl} = 10) > (BOUNCES_{Xl} = 8) \parallel (MBR_{Xu} = 20) < (BOUNCES_{Xu} = 22)$

Pada *iterasi* ke dua pengecekan terhadap batas atas :

$(MBR_{Yl} = 12) > (BOUNCES_{Yl} = 10) \parallel (MBR_{Yu} = 20) < (BOUNCES_{Yu} = 22)$

Hasil dari proses perhitungan dilanjutkan untuk pengecekan kondisi *inside* ataukah *overlap*, hasil pengecekan nilainya di kembalikan berdasarkan kondisi true jika *inside* atau *overlap*e dan sebaliknya



Gambar 3.43 Prosedur Section

3.4 Perhitungan Manual

3.4.1 Pengindeksan R*-tree

Sebagai contoh perhitungan manual pengindeksan data spasial digunakan prosentase *reinsert* sebesar 30% untuk *reintegrasi* yang memerlukan optimasi dan minimasi pada struktur data R*-tree.

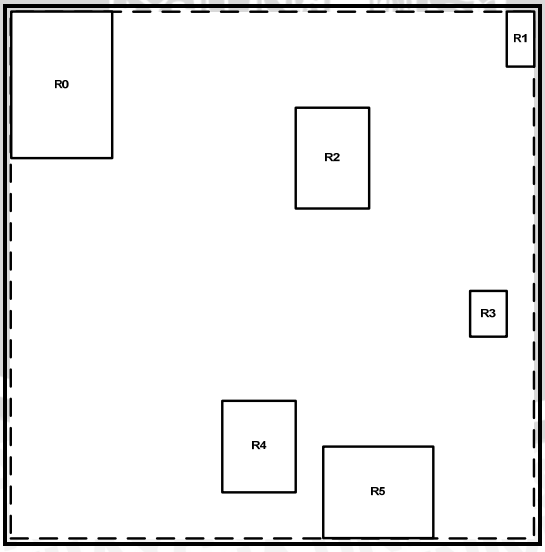
Jumlah enteri MBR yang dimasukkan sebanyak 6 enteri ke dalam struktur data R*-tree dengan id *Entry* nol hingga lima yang di ilustrasikan pada **Gambar 3.44**, MBR-MBR ini mewakili dari suatu pembatas koordinat geometri dari polygon,

Jumlah maksimum dan minimum *entry* yang dimuat pada struktur R*-tree adalah m=2 dan M =3, kumpulan *Entry* dari R0 hingga R5 dengan susunan sumbu berturut-turut {xl,xu,yl,yu} sebagai berikut :

R0={0.0,11.0,84.0,100.0}, R1={54.0,57.0,94.0,100.0},

R2={31.0,39.0,78.5,89.5}, R3={50.0,54.0,64.5,69.0},

R4={23.0,31.0,47.5,57.5}, R5={34.0,46.0,42.5,52.5}.



Gambar 3.44 Ilustrasi MBR polygon

Langkah-langkah sebagai contoh perhitungan untuk pengindeksan R*-tree sebagai berikut :

1. Bilamana *node* memerlukan ChooseSubtree.
   a) Identifikasi seluruh *node* MBR pembatas untuk memasukkan *Entry* R.
   b) Lakukan pengecekan kemungkinan tumpang tindih atau *inside* pada masing-masing *node* pembatas (Bmbr) yang akan di sisipi *Entry* R , pengecekan dilakukan sebagai berikut :

      Pada sumbu x *Entry* R terhadap sumbu x *node* anak :

      $(R_{Xl}) < (\text{Bmbr}_{Yl}) \parallel (R_{Yl}) > (\text{Bmbr}_{Xl})$

      Pada sumbu y *Entry* R terhadap sumbu y *node* anak :

      $(R_{Xu}) < (\text{Bmbr}_{Yu}) \parallel (R_{Yu}) > (\text{Bmbr}_{Xu})$

   c) Hasil pengecekan jika pada *node* anak terhadap pemasukan *Entry* R berada didalam ruang salah satu *node* anak maka *Entry* R dimasukkan ke dalam *node* anak tersebut,
   d) Hasil pengecekan jika *Entry* berada lebih dari satu *node* anak maka mengambil ruang yang pembesarannya paling kecil dan masukkan *Entry* R ke dalam *node* anak yang areanya paling kecil tersebut,
   e) Hasil pengecekan jika *Entry* R tidak berada di dalam *node* anak dan *entry* R adalah *node* yang akan di sisipkan ke *node* leaf tentukan pembesaran area dari *node* anak (NBmbr) terhadap masukan *Entry* R(Bmbr), susunan pembesaran adalah (Xl_{min}, Xu_{maks}, Yl_{min}, Yu_{maks}), lakukan minimasi dan optimasi ruang mati serta tumpang tindih dengan langkah 1g dan 1h
   f) Hasil pengecekan jika *Entry* R tidak berada di dalam *node* anak dan *entry* R adalah *node* yang akan di sisipkan ke *node nonleaf* tentukan pembesaran

area dari *node* anak (NBmbr) terhadap masukan *Entry* R(Bmbr), susunan pembesaran adalah (Xl_{min}, Xu_{maks}, Yl_{min}, Yu_{maks}), lakukan minimasi dan optimasi ruang mati dengan langkah 1g

g) Menghitung selisih ruang kosong dari pembesasaran baru yaitu ruang pembesaran baru dari *node* anak terhadap masukan *Entry* R(NBmbr) dikurangi oleh ruang dari *node* anak, perhitungan ruang untuk setiap *Entry node* anak mapun pembesaran ruang baru R4 dengan *node* anak adalah (Xu-Xl)*(Yu-Yl)

h) Menghitung luas irisan tumpang tindih MBR dari seluruh kemungkinan *Entry* pembesaran *node* anak baru (N_i) terhadap *node* anak ($Entry_i$) di kurangi oleh perhitungan dari tumpang tindih ($Entry_i$) *node* anak yang satu dengan ($Entry_i$) *node* anak yang lain.

i) Bandingkan hasilnya dari perhitungan langkah 1e jika *entry* R adalah masukan ke *node* leaf, jika masukan ke *node noneleaf* langkah 1f, dan sisipkan *entry* R ke *node* yang hasilnya paling optimal.

2. Bilamana *node* memerlukan untuk *reinsert*.

Kapasitas *Entry* penuh yang menyebabkan overflow, dan ini adalah pertama kali selama proses pemasukan R maka hitung optimasi berdasarkan jarak nilai sumbu pusat dengan *Entry node* pembatas dan *reinsert* berdasarkan jarak sumbu pusat yang maksimal.

a) Pertama kali identifikasi seluruh *entry* dalam *node* yang akan dilakukan *reinsert*

b) Untuk menghitung jarak maksimal pusat *Entry* adalah dengan menentukan pusat utama pembatas dari seluruh *Entry* (CBmbr) dari langkah 2.a, dengan

susunan sumbu pembatas *Entry* adalah {Xl_{min},Xu_{maks},Yl_{min},Yu_{maks}), dengan susunan tersebut tentukan sumbu pusat pembatas x=Xu_{maks}+Xl_{min}/2 dan sumbu pembatas pusat y=Yu_{maks}+Yl_{min}/2,

c) Langkah selanjutnya menghitung pusat dari setiap calon *Entry* (Cmbr) dengan sumbu pusat x=Xu+Xl/2 dan sumbu pusat y=Yu+Yl/2,

d) Menghitung jarak pusat ke pusat secara keseluruhan dengan tiap sumbu dari pusat tiap *Entry* (Cmbr) dengan pusat pembatas *Entry* (CBmbr) adalah Cx^2 = Cmbr x – CBmbr x dan Cy^2 = Cmbr y – CBmbr y, yang terakhir adalah hasil jarak pusat dari setiap *Entry* (DC) adalah DC= Cx^2 + Cy^2.

e) Proses terakhir adalah mengurutkan calon *Entry* MBR berdasarkan jarak setiap pusat (DC) dari setiap *Entry node* MBR, untuk menentukan *Entry* yang akan dimasukkan kembali(*reinsert*) dengan 30% dari seluruh jumlah *Entry*.

f) Berdasarkan hasil perhitungan langkah 2e dilakukan pemasukan kembali 30% dari jumlah *Entry*, dengan demikian terdapat 2 kelompok, kelompok pertama adalah 70% pembulatan keatas dari jumlah *Entry* yang tersisa dalam *node* dan yang ke dua adalah 30% pembulatan kebawah dari jumlah data *Entry* yang dimasukan kembali (*reinsert*).

3. Bilamana *node* memerlukan *split*.

Pada proses *split* terdapat dua proses, yaitu menentukan sumbu minimal *index* pembelahan dengan algoritma **ChooseSplitAxis** yang hasil minimalnya dihitung optimasi dan minimasi tumpang tindih serta ruang mati dengan algoritma **ChooseSplitIndex,**

1. Identifikasi seluruh *entry* dalam *node* yang akan dilakukan *split.*

2. Algoritma ChooseSplitAxis

a) Algoritma ChooseSplitAxis pada proses pertama kali kelompok *index* pertama diurutkan berdasarkan nilai sumbu tegak lurus batas bawah xl (SML), dan untuk kelompok kedua pengurutan berdasarkan nilai sumbu tegak lurus batas atas xu (SMU).

b) Berdasarkan kedua data *sorting* langkah 3.2.a, *Entry* hasil *sorting* di kelompokan untuk menentukan optimasi dan minimasi sumbu *margin* dengan distribusi(k) adalah M - 2*m + 1, dengan penambahan k+1 yang dimulai dengan k =0, dimana M adalah jumlah seluruh *Entry node* dan m adalah 40% dari M, untuk kelompok *Entry* pertama(G1) adalah m+k dan kelompok *Entry* kedua(G2) M-m-k.

c) Hasil distribusi kelompok dari langkah 3.2.b dibuat pembatatas masing-masing, dengan susunan pembatas BG1 dan BG2 adalah (GXl_{min}, GXu_{maks}, GYl_{min}, GYu_{maks}).

d) Kemudian dari hasil tiap-tiap pembatas kelompok dari langkah 3.2.c dihitung pembesaran dari *margin* sumbu SML dan SMU untuk menjadi *index* pembelahan terbaik, yaitu *margin* pembatas kelompok pertama (MG1) adalah (BG1Xu – BG1Xl + BG1Yu – BG1Yl) hasilnya di tambahkan dengan pembatas kelompok kedua (MG2) adalah (BG2Xu – BG2Xl + BG2Yu – BG2Yl) berdasarkan persamaan 2.2.

e) Ulangi langkah 3.2 b,c dan d berkenaan dengan *index* ke dua, dengan seluruh *entry* dalam *node* diurutkan berdasarkan nilai sumbu tegak lurus batas bawah yl (SML), dan untuk kelompok kedua pengurutan berdasarkan nilai sumbu tegak lurus batas atas yu (SMU)

f) Bandingkan dari hasil *index* pertama dan kedua yang hasil minimalnya dari perbandingan tersebut dihitung optimasi dan minimasi tumpang tindih serta ruang mati dengan algoritma ChooseSplitIndex

3. Algotitma ChooseSplitIndex

a) Untuk menghitung area dan tumpang tindih minimal terlebih dahulu menghitung area dari tiap *Entry* berdasarkan distribusi ke (k) di setiap *Entry* di dalam kelompok dari G1 dan G2 *index* pembelahan pertama, dengan perhitungan tiap area *Entry* adalah (Xu-Xl)*(Yu-Yl).

b) Perhitungan area selanjutnya adalah hasil distribusi pembatas kelompok SML dan SMU *index* pertama dilakukan perhitungan dengan hasil penjumlahan dari area pembatas kelompok pertama(BG1) dengan area pembatas dari kelompok kedua(BG2), dimana jumlah tiap-tiap area (BArea) yaitu area pembatas kelompok pertama (BG1) adalah (BG1Xu – BG1Xl) * (BG1Yu – BG1Yl) hasilnya di tambah dengan area pembatas kelompok ke dua (BG2) adalah (BG2Xu – BG2Xl) * (BG2Yu – BG2Yl) dari setiap distribusi (k). dengan penambahan k+=1.

c) Perulangan distribusi ke (k) SML dan SMU dihitung area minimal (MArea) dengan selisih dari Jumlah pembatas area (BArea) di kurangi dengan hasil penjumlah dari *Entry* area kelompok G1 + jumlah *Entry* area kelompok G2 (Garea) dari setiap distribusi ke (k) berdasarkan persamaan 2.1.

d) Proses terakhir menghitung luas irisan tumpang tindih berdasarkan persamaan 2.3 disetiap distribusi(k) kelompok sml dan smu dari *Entry* pembatas MBR kelompok BG1 dengan BG2 adalah hasil kali dari sisi

x(Ub-Lb) dengan sisi Y(Ub-Lb) dengan menggunakan tabel 3.1 dan 3.2 untuk pengecekan tumpang tindih.

Tabel 3.1 Tumpang Tindih terhadap Sumbu X

Tumpang Tindih Sumbu X	Ub-Lb	Keterangan
R1xu >= R2xl && R1_xu <= R2Xu	R1xu - R2xl	Batas atas R1 Didalam R2
(R1xu>=R2xl, && R1_xu <= R2xu) && R1xl>=R2xl && R1xl<=R2xu	R1xu - R1xl	Batas atas dan bawah R1 Didalam R2
R1xl>=R2xl, && R1xl<=R2xu	R2xu - R1xl	Batas bawah R1 Didalam R2
(R2xl>=R1xl&&R2xl<=R1xu) && (R2xu>=R1xl,&&(R2xu<= R1xu)	R2xu - R2xl	R1 berisi R2

Tabel 3.2 Tumpang Tindih terhadap Sumbu Y

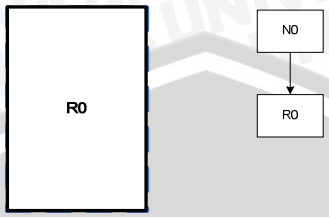
Tumpang Tindih Sumbu Y	Ub-Lb	Keterangan
R1yu >=R2yl && R1yu<=R2yu	R1yu - R2yl	Batas atas R1 Didalam R2
(R1yu>=R2yl && R1yu <= R2yu) && R1yl>=R2yl && R1yl<=R2yu	R1yu - R1yl	Batas atas dan bawah R1 Didalam R2
R1yl>=R2yl && R1yl<=R2yu	R2yu - R1yl	Batas bawah R1 Didalam R2
(R2yl>=R1yl && R2yl<=R1yu) && (R2yu>=R1yl && R2yu<= R1yu)	R2yu - R2yl	R1 berisi R2

e) Langkah terakhir untuk mendistribusikan pengelompokan pembelahan terbaik adalah berdasarkan hasil minimal dari perhitungan optimasi dan minimasi ruang mati langkah 3.3c dan tumpang tindih dari langkah 3.3d.

Masukkan data R0 ={0.0,11.0,84.0,100.0} kedalam R*-tree, ilustrasi pemasukan *node* R0 di tunjukkan pada **Gambar 3.45** berikut:

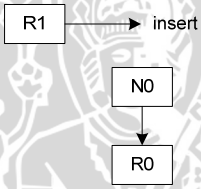
Gambar 3.45 Struktur R*-tree pemasukan R0

Karena data masih kosong masukkan R0 ke *tree*, buat *node Entry* 0, struktur R*tree setelah pemasukan R0 di tujukan pada **Gambar 3.46** berikut:



Gambar 3.46 Struktur R*-tree pemasukan *Entry* R0

Masukkan data R1 ={54.0,57.0,94.0,100.0} kedalam R*-tree, ilustrasi pemasukan *node* R1 di tunjukkan pada **Gambar 3.47** berikut:

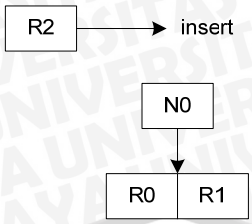


Gambar 3.47 Struktur R*-tree pemasukan *node Entry* R1

Karena kapasitas masih memenuhi , masukkan R1 ke sebuah *node*, buat *node Entry* 1, struktur R*tree setelah pemasukan R1 di tunjukan pada **Gambar 3.48** berikut:

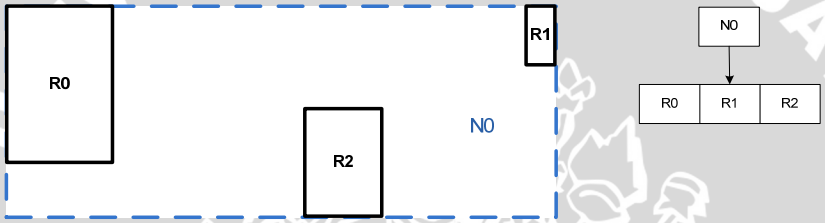
Gambar 3.48 Struktur R*-tree setelah pemasukan *node* R1

Masukkan data R2={31.0,39.0,78.5,89.5} kedalam R*-tree, ilustrasi pemasukan *node* R2 di tunjukkan pada **Gambar 3.49** berikut:



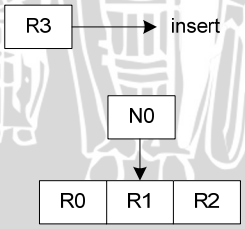
Gambar 3.49 Struktur R*-tree pemasukan *node Entry* R2

Karena kapasitas masih memenuhi , masukkan R2 ke sebuah *node*, buat *node Entry* 2 ilustrasi pemasukan *node* R2 di tunjukkan pada **Gambar 3.50** berikut:



Gambar 3.50 Struktur R*-tree setelah pemasukan *node* R2

Masukkan data *Entry* R3= {7.0, 21.0, 26.5, 42.5} kedalam R*-*tree*, ilustrasi pemasukan *node* R3 di tunjukkan pada **Gambar 3.51** berikut:



Gambar 3.51 Struktur R*-tree pemasukan *node Entry* R3

Karena data kapasitas *Entry* penuh dimana (M=3) sedangakan setelah pemasukan R3, jumlah *Entry* M=4 yang menyebabkan overflow, dan ini adalah pertama kali selama proses pemasukan R3 maka dengan langkah 2.a, keseluruhan *Entry node* di tunjukkan pada tabel 3.3 *Entry node* MBR

Tabel 3.3 *Entry Node* MBR

Entry	xl	xu	yl	Yu
0	0,0	11,0	84,0	100,0
1	54,0	57,0	94,0	100,0
2	31,0	39,0	78,5	89,5
3	50,0	54,0	64,5	69,0

Menetukan sumbu pusat pembatas berdasarkan langkah 2.b (Bbmbr) adalah {0.0, 57.0, 64,5, 100.0} dengan sumbu pusat pembatas x= 28.5 dan y=82.5

Proses selanjutnya menghitung pusat dari setiap calon *Entry* dengan langkah 2e di tunjukkan pada tabel 3.4 sumbu pusat setiap calon *Entry*

Tabel 3.4 Sumbu pusat setiap calon *Entry*

Entry	MBR(xl,xu,yl,yu)	Pusat x,y(Cmbr)	
		xu+xl /2	yu+yl/2
0	0.0,11.0,84.0,100.0	5,5	92
1	54.0,57.0,94.0,100.0	55,5	97
2	31.0,39.0,78.5,89.5	35	84
3	50.0,54.0,64.5,69.0	52	66,75

Menghitung jarak pusat ke pusat secara keseluruhan hasil dari langkah 2d, Seperti di tunjukkan pada tabel 3.5 jarak pusat ke pusat *Entry*

Tabel 3.5 Jarak pusat ke pust *Entry*

Cmbr - CBmbr		Cx^2	Cy^2	DC
Cx	Cy			
-23	9,75	529	95,0625	624,0625
27	14,75	729	217,5625	946,5625
6,5	1,75	42,25	3,0625	45,3125
23,5	-15,5	552,25	240,25	792,5

Proses terakhir adalah mengurutkan calon *Entry* MBR berdasarkan jarak pusat dengan langkah 2.e, hasil pengurutan jarak pusat di tunjukkan oleh tabel 3.6

Tabel 3.6 *Sorting Entry* berdasarkan jarak pusat (DC)

Entry	MBR	Jarak Pusat (DC)
2	31.0,39.0,78.5,89.5	45,3125
0	0.0,11.0,84.0,100.0	624,0625
3	50.0,54.0,64.5,69.0	792,5

Entry	MBR	Jarak Pusat (DC)
1	54.0,57.0,94.0,100.0	946,5625

Berdasarkan hasil *sorting* pusat MBR dari langkah 2f seperti di tunjukkan pada tabel 3.7 dan tabel 3.8

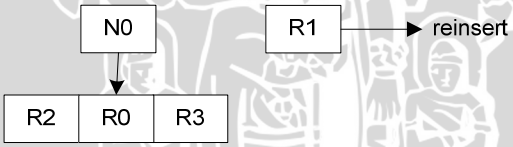
Tabel 3.7 Data calon pemasukan 70%

Entry	MBR
2	31.0,39.0,78.5,89.5
0	0.0,11.0,84.0,100.0
3	50.0,54.0,64.5,69.0

Tabel 3.8 Data pemasukan 30%

Entry	MBR
1	54.0,57.0,94.0,100.0

Masukkan kembali data *Entry* {2,0,3) dalam *node* dan *reinsert* data *Entry* R1, ilustrasi *reinsert Entry node* R1 seperti di tunjukkan pada **Gambar 3.52** berikut.



Gambar 3.52 Struktur R*-tree pemasukan *Entry* R1 kembali dengan 30% dari jumlah *Entry*

Karena jumlah *Entry* melebihi kapasitas dimana M=3, dan ini bukan proses pertamakali selama pemasukan R3, lakukan *Split* untuk mebagi *node* menjadi dua kelompok, berdasarkan langkah 3.1, jumlah keseluruhan *Entry* dalam *node* ditunjukkan pada tabel 3.9 *Entry split* pemasukan R1

Tabel 3.9 *Entry Split* pemasukan R1

Entry	MBR
2	31.0,39.0,78.5,89.5
0	0.0,11.0,84.0,100.0

Entry	MBR
3	50.0,54.0,64.5,69.0
1	54.0,57.0,94.0,100.0

Proses selanjutnya menentukan *index* pembelahan *node* terbaik dengan Algoritma ChooseSplitAxis ,berdasarkan langkah 3.2a pengurutan *Entry* ditunjukkan pada tabel 3.10 dan tabel 3.11.

Tabel 3.10 Pengurutan batas bawah Xl

Entry	MBR	Xl
0	0.0, 11.0, 84.0, 100.0	0.0
2	31.0, 39.0, 78.5, 89.5	31.0
3	50.0, 54.0, 64.5, 69.0	50.0
1	54.0, 57.0, 94.0, 100.0	54.0

Tabel 3.11 Pengurutan batas atas Xu

Entry	MBR	Xu
0	0.0, 11.0, 84.0, 100.0	11.0
2	31.0, 39.0, 78.5, 89.5	39.0
3	50.0, 54.0, 64.5, 69.0	54.0
1	54.0, 57.0, 94.0, 100.0	57.0

Berdasarkan langkah 3.2.b dimana m adalah 40% dari M, dengan demikian terbentuk 3 distribusi(k) dari 4 *Entry*, distribusi kelompok setiap *Entry* ditunjukkan pada tabel 3.12

Tabel 3.12 Distribusi kelompok setiap *Entry*

Sumbu	(k)	G1 (m+k)		G2 (M-m-k)	
		Entry	MBR	*Entry*	MBR
SML	0	0	0.0, 11.0, 84.0, 100.0	2	31.0, 39.0, 78.5, 89.5
				3	50.0, 54.0, 64.5, 69.0
				1	54.0, 57.0, 94.0, 100.0
	1	0	0.0, 11.0, 84.0, 100.0	3	50.0, 54.0, 64.5, 69.0
		2	31.0, 39.0, 78.5, 89.5	1	54.0, 57.0, 94.0, 100.0
	2	0	0.0, 11.0, 84.0, 100.0	1	54.0, 57.0, 94.0, 100.0
		2	31.0, 39.0, 78.5, 89.5		
		3	50.0, 54.0, 64.5, 69.0		

Sumbu	(k)	G1 (m+k)		G2 (M-m-k)	
		Entry	MBR	*Entry*	MBR
SMU	0	0	0.0, 11.0, 84.0, 100.0	2	31.0, 39.0, 78.5, 89.5
				3	50.0, 54.0, 64.5, 69.0
				1	54.0, 57.0, 94.0, 100.0
	1	0	0.0, 11.0, 84.0, 100.0	3	50.0, 54.0, 64.5, 69.0
		2	31.0, 39.0, 78.5, 89.5	1	54.0, 57.0, 94.0, 100.0
	2	0	0.0, 11.0, 84.0, 100.0	1	54.0, 57.0, 94.0, 100.0
		2	31.0, 39.0, 78.5, 89.5		
		3	50.0, 54.0, 64.5, 69.0		

Hasil distribusi kelompok berdasarkan langkah 3.2.c ditunjukkan pada tabel 3.13

Tabel 3.13 Distribusi perluasan setiap kelompok *Entry*

Sumbu	(k)	BG1	BG2
SML	0	0.0, 11.0, 84.0, 100.0	31.0, 57.0, 64.5, 100.0
	1	0.0, 39.0, 78.5, 100.0	50.0, 57.0, 64.5, 100.0
	2	0.0, 54.0, 64.5, 100.0	54.0, 57.0, 94.0, 100.0
SMU	0	0.0, 11.0, 84.0, 100.0	31.0, 57.0, 64.5, 100.0
	1	0.0, 39.0, 78.5, 100.0	50.0, 57.0, 64.5, 100.0
	2	0.0, 54.0, 64.5, 100.0	54.0, 57.0, 94.0, 100.0

Kemudian dari hasil tiap-tiap pembatas kelompok dihitung pembesaran dari *margin*, berdasarkan langkah 3.2d, hasil pembesaran dari *margin* ditunjukkan pada tabel 3.14

Tabel 3.14 Optimasi *margin* setiap kelompok *Entry*

Sumbu	Distribusi (k)	*Margin*		
		MG1	MG2	Jumlah
SML	0	27	61,5	88,5
	1	60,5	42,5	103
	2	89,5	9	98,5
SMU	0	27	61,5	88,5
	1	60,5	42,5	103
	2	89,5	9	98,5
Total *Margin* Kelompok *index* Pertama				**580**

Hasil perhitugan akhir dari optimasi *margin index* pertama adalah 580, kemudian berdasarkan langkah 2.3. e di tunjukkan pada tabel 3.15 dan 3.16.

Tabel 3.15 Pengurutan batas bawah Yl

Entry	MBR	Yl
3	50.0, 54.0, 64.5, 69.0	64,5
2	31.0, 39.0, 78.5, 89.5	78,5
0	0.0, 11.0, 84.0, 100.0	84
1	54.0, 57.0, 94.0, 100.0	94

Tabel 3.16 Pengurutan batas atas Yu

Entry	MBR	Yu
3	50.0, 54.0, 64.5, 69.0	69
2	31.0, 39.0, 78.5, 89.5	89,5
0	0.0, 11.0, 84.0, 100.0	100
1	54.0, 57.0, 94.0, 100.0	100

Berdasarkan kedua data *sorting* terendah dari sumbu batas bawah dengan langkah 3.2b dimana m adalah 40% dari M dan M=4, terbentuk 3 distribusi(k), ditunjukkan pada tabel 3.17

Tabel 3.17 Distribusi kelompok setiap *Entry*.

Sumbu	(k)	G1 (m+k)		G2 (M-m-k)	
		Entry	MBR	*Entry*	MBR
SML	0	3	50.0, 54.0, 64.5, 69.0	2	31.0, 39.0, 78.5, 89.5
				0	0.0, 11.0, 84.0, 100.0
				1	54.0, 57.0, 94.0, 100.0
	1	3	50.0, 54.0, 64.5, 69.0	0	0.0, 11.0, 84.0, 100.0
		2	31.0, 39.0, 78.5, 89.5	1	54.0, 57.0, 94.0, 100.0
	2	3	50.0, 54.0, 64.5, 69.0	1	54.0, 57.0, 94.0, 100.0
		2	31.0, 39.0, 78.5, 89.5		
		0	0.0, 11.0, 84.0, 100.0		
SMU	0	3	50.0, 54.0, 64.5, 69.0	2	31.0, 39.0, 78.5, 89.5
				0	0.0, 11.0, 84.0, 100.0
				1	54.0, 57.0, 94.0, 100.0
	1	3	50.0, 54.0, 64.5, 69.0	0	0.0, 11.0, 84.0, 100.0
		2	31.0, 39.0, 78.5, 89.5	1	54.0, 57.0, 94.0, 100.0
	2	3	50.0, 54.0, 64.5, 69.0	1	54.0, 57.0, 94.0, 100.0

Sumbu	(k)	G1 (m+k)		G2 (M-m-k)	
		Entry	MBR	*Entry*	MBR
SMU	2	2	31.0, 39.0, 78.5, 89.5		
		0	0.0, 11.0, 84.0, 100.0		

Hasil distribusi kelompok berdasarkan langkah 3.2.c ditunjukkan pada tabel 3.18

Tabel 3.18 Distribusi perluasan setiap kelompok *Entry*

Sumbu	(k)	BG1	BG2
SML	0	50.0, 54.0, 64.5, 69.0	0.0, 57.0, 78.5, 100.0
	1	31.0, 54.0, 64.5, 89.5	0.0, 57.0, 84.0, 100.0
	2	0.0, 54.0, 64.5, 100.0	54.0, 57.0, 94.0, 100.0
SMU	0	50.0, 54.0, 64.5, 69.0	0.0, 57.0, 78.5, 100.0
	1	31.0, 54.0, 64.5, 89.5	0.0, 57.0, 84.0, 100.0
	2	0.0, 54.0, 64.5, 100.0	54.0, 57.0, 94.0, 100.0

Kemudian dari hasil tiap-tiap pembatas kelompok dengan langkah 3.2d hasil pembesaran dari *margin* ditunjukkan pada tabel 3.19

Tabel 3.19 Distribusi *margin* perluasan setiap kelompok *Entry*

Sumbu	(k)	*Margin*		
		MG1	MG2	Jumlah
SML	0	8,5	78,5	87
	1	48	73	121
	2	89,5	9	98,5
SMU	0	8,5	78,5	87
	1	48	73	121
	2	89,5	9	98,5
Total *Margin* Kelompok *index* Kedua				631

Hasil perhitugan akhir dari *margin* kelompok *index* Kedua adalah 631, bila dibandingkan hasil dari *index* pertama adalah lebih besar dari *margin index* kelompok pertama yaitu 580, dengan demikian Kelompok *index* pertama adalah *margin* yang paling minimal dengan *index* pembelahan terbaik dan terpilih untuk dihitung Area mati dan Tumpang tindih minimal pada Algoritma ChooseSplitIndex berdasarkan langkah 3.3.

Berdasarkan langah 3.3.a, perhitungan area *Entry* di setiap kelompok distribusi seperti di tunjukkan pada tabel 3.20

Tabel 3.20 Distribusi area *Entry*.

Sumbu	(k)	G1 (m+k)		G2 (M-m-k)	
		Entry	**Area**	***Entry***	**Area**
SML	0	0	176	2	88
				3	18
				1	18
	1	0	176	3	18
		2	88	1	18
	2	0	176	1	18
		2	88		
		3	18		
SMU	0	0	176	2	88
				3	18
				1	18
	1	0	176	3	18
		2	88	1	18
	2	0	176	1	18
		2	88		
		3	18		

Perhitunga selanjutnya berdasarkan langkah 3.3.b, area pembatas kelompok *index* pertama ditunjukkan pada tabel 3.21.

Tabel 3.21 Distribusi optimasi area kelompok *Entry*

Sumbu	(k)	Area		
		BG1	**BG2**	**Barea**
SML	0	176	923	1099
	1	838,5	248,5	1087
	2	1917	18	1935
SMU	0	176	923	1099
	1	838,5	248,5	1087
	2	1917	18	1935

Perhitungan area minimal berdasarkan langkah 3.3.c distribusi minimasi ruang kelompok *Entry* di tunjukkan pada tabel 3.22

Tabel 3.22 Distribusi minimasi ruang kelompok *Entry*

Sumbu	(k)	g1,g2	*Entry*	Area		
				Barea	Garea	Marea
SML	0	1,3	(0),(2,3,1)	1099	300	799
	1	2,2	(0,2),(3,1)	1087	300	787
	2	3,1	(0,2,3),(1)	1935	282	1653
SMU	0	1,3	(0),(2,3,1)	1099	300	799
	1	2,2	(0,2),(3,1)	1087	300	787
	2	3,1	(0,2,3),(1)	1935	300	1635

Proses terakhir menghitung luas irisan tumpang tindih minimal di setiap kelompok dengan langkah 3.3d, ditunjukkan pada tabel 3.23.

Tabel 3.23 Distribusi Tumpang tindih setiap kelompok *Entry*

Sumbu	(k)	Keterangan		Ub-Lb		Moverlap
		x	y	x	y	
SML	0	tidak	R1ub dan lb di R2	0	16	0
	1	tidak	R1ub dan lb di R2	0	21,5	0
	2	R1ub di R2	R1ub di R2	0	6	0
SMU	0	tidak	R1ub dan lb di R2	0	16	0
	1	tidak	R1ub dan lb di R2	0	21,5	0
	2	R1ub di R2	R1ub di R2	0	6	0

Dari hasil perhitungan akhir tumpang tindih (Moverlap) dan ruang mati (Marea) berdasarkan langkah 3.3e dan perbandingan ruang mati dan tumpang tindih ditunjukkan pada tabel 3.24

Tabel 3.24 Perbandingan ruang mati dan tumpang tindih

Sumbu	(k)	g1,g2	*Entry*	Marea	Moverlap
SML	0	1,3	(0),(2,3,1)	799	0
	1	2,2	(0,2),(3,1)	787	0
	2	3,1	(0,2,3),(1)	1653	0
SMU	0	1,3	(0),(2,3,1)	799	0
	1	2,2	(0,2),(3,1)	787	0
	2	3,1	(0,2,3),(1)	1635	0

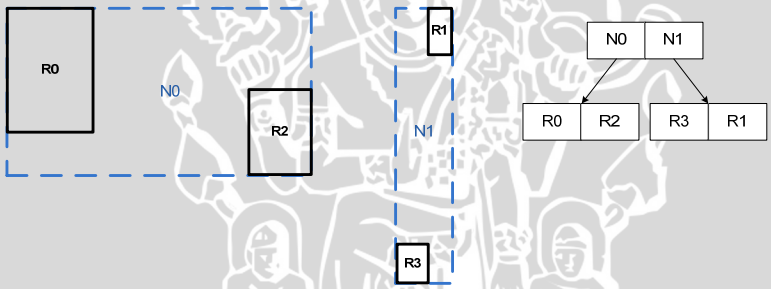
Dengan demikian berdasarkan hasil perhitungan akhir optimasi dan minimasi ruang mati serta tumpang tindih terbaik untuk didistribusikan

pengelompokan pembelahan adalah *Entry* 0 dan 2 pada kelompok distribusi pertama , kemudian *Entry* 3 dan 1 untuk distribusi kelompok ke dua pada sumbu sml dengan distribusi(k) adalah 1, hasil distribusi dan optimasi pembelah terbaik di tunjukkan pada tabel 3.25

Tabel 3.25 Distribusi dan optimasi pembelah terbaik

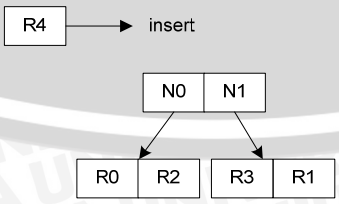
Kel	Pembatas	*Entry*	MBR
1	0.0, 39.0, 78.5, 100.0	0	0.0, 11.0, 84.0, 100.0
		2	31.0, 39.0, 78.5, 89.5
2	50.0, 57.0, 64.5, 100.0	3	50.0, 54.0, 64.5, 69.0
		1	54.0, 57.0, 94.0, 100.0

Struktur *tree* setelah di lakukan pembelahan berdasarkan distribusi dan optimasi pembelah terbaik di tunjukkan oleh **Gambar 3.53** berikut:.



Gambar 3.53 Struktur R*-tree pemasukan *node* R3 setelah distribusi dan optimasi pembelah terbaik

Masukkan data *Entry* R4= {23.0,31.0,47.5,57.5}kedalam R*-tree, ilustrasi pemasukan *node* R4 di tunjukkan pada **Gambar 3.54** berikut:



Gambar 3.54 Struktur R*-tree pemasukan *Entry* 4

Identifikasi root pada *nodeleaf* untuk memasukkan *Entry* R4, karena root pada *nodeleaf* tidak kosong panggil Algoritma ChooseSubtree untuk menentukan tumpang tindih dan pembesaran area berdasarkan langkah 1 yang akan ditempati R4, seluruh *Entry node* pembatas anak berdasarkan langkah 1.a ditunjukan pada tabel 3.26

Tabel 3.26 *Entry* MBR pembatas *node* anak

Entry	**MBR**
0	0.0 ,39.0 ,78.5 ,100.0
1	50.0 ,57.0 ,64.5 ,100.0

Proses pada ChooseSubtree pertama kali dilakukan pengecekan tumpang tindih berdasarkan langkah 1b, hasil pengecekan ditunjukkan pada tabel 3.27

Tabel 3.27 Pengecekan *node* anak

R4		**Bmbr**		***overlap / inside***
Entry	**MBR**	***Entry***	**MBR**	
4	23.0,31.0,47.5,57.5	0	0.0 ,39.0 ,78.5 ,100.0	tidak
		1	50.0 ,57.0 ,64.5 ,100.0	tidak

Karena R4 tidak *inside* atau *overlap* terhadap *node* anak dan R4 dimasukan ke *nodeleaf* berdasarkan langkah 1.e pembesaran ruang baru di tunjukkan pada tabel 3.28

Tabel 3.28 Pembatas ruang baru(NBmbr) R4 terhadap *node* anak

R4		**Bmbr**		**NBmbr**
Entry	**MBR**	***Entry***	**MBR**	
4	23.0,31.0,47.5,57.5	0	0.0 ,39.0 ,78.5 ,100.0	0.0,39.0,47.5,100.0
		1	50.0 ,57.0 ,64.5 ,100.0	23.0,57.0,47.5,100.0

Proses selanjutnya menghitung selisih ruang kosong, berdasarkan langkah 1g ditunjukkan pada tabel 3.29

Tabel 3.29 Selisih ruang kosong

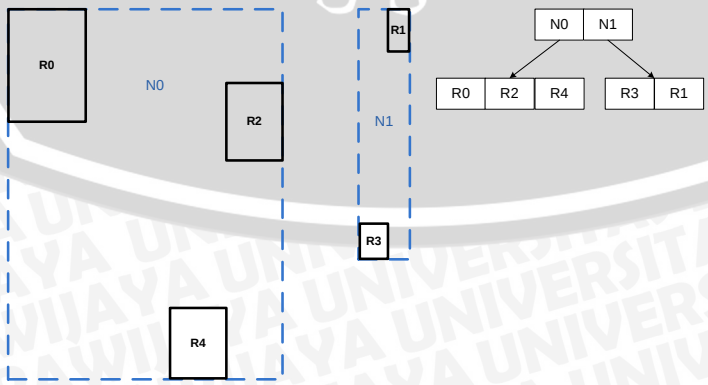
Entry	NBmbr	Bmbr	Selisih
0	2047,5	838,5	1209
1	1785	248,5	1536,5

Proses selanjutnya adalah optimasi dan minimasi tumpang tindih berdasarkan langkah 1h, di hasilkan perhitungan pada tabel 3.30

Tabel 3.30 Perbandingan tumpang tindih antara *Entry* MBR

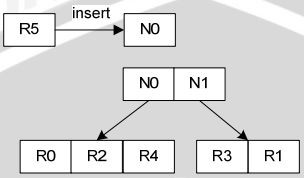
Entry	*Entry* Bmbr,Nmbr	keterangan		Ub-Lb			Selisih
		x	y	x	y	*overlap*	
0	(N0.1)	tidak	R1 ub di R2	0	35,5	0	0
	(0.1)	tidak	tidak	0	0	0	
1	(N0.1)	R1 lb di R2	R1ub di R2	16	21,5	344	344
	(1.0)	tidak	R1 ub di R2	0	21,5	0	

Dari hasil perhitungan tabel 3.30 perbandingan tumpang tindih antara *Entry* MBR dan ruang kosong *Entry* pembesaran *node* anak terhadap *Entry* masukan R4 yang optimal berdasarkan langkah 1i di tunjukkan pada *Entry* pembesaran *node* anak baru ke nol. Masukkan R4 ke *bounces* 0 dengan pembesaran area *bounces* 0.0, 39.0, 47.5, 100.0, karena kapasitas *Entry* dalam *node* anak tersebut masih muat yaitu m<=3 maka masukkan *Entry* R4 ke *node* anak tersebut, proses pemasukan setelah ChooseSubtree di tunjukkan pada **Gambar 3.55** berikut



Gambar 3.55 Struktur R*-tree pemasukan R4 setelah ChooseSubtree

Pemsukan MBR selanjutnya memasukkan data *Entry* R5= {34.0,46.0,42.5,52.5} kedalam R*-tree yang ditunjukkan pada **Gambar 3.56** berikut.



Gambar 3.56 Struktur R*-tree pemasukan *Entry* R5

Identifikasi root pada *nodeleaf* untuk memasukkan *Entry* R4, karena root pada *nodeleaf* tidak kosong panggil Algoritma ChooseSubtree untuk menentukan tumpang tindih dan pembesaran area berdasarkan langkah 1 yang akan ditempati R5, seluruh *Entry node* pembatas anak berdasarkan langkah 1.a ditunjukan pada tabel 3.31

Tabel 3.31 *Entry* mbr pembatas *node* anak

Entry	***Bounces***
0	0.0 ,39.0 ,47.5 ,100.0
1	50.0 ,57.0 ,64.5 ,100.0

Proses pengecekan tumpang tindih pada ChooseSubtree berdasarkan langkah 1b di tunjukkan pada tabel 3.32

Tabel 3.32 Pengecekan *node* anak

R5		**Bmbr**		***overlap / inside***
Entry	**MBR**	***Entry***	**MBR**	
5	34.0,46.0,42.5,52.5	0	0.0 ,39.0 ,47.5 ,100.0	tidak
		1	50.0 ,57.0 ,64.5 ,100.0	tidak

Karena R5 tidak *inside* atau *overlap* terhadap node anak dan R5 dimasukan ke *nodeleaf* berdasarkan langkah 1.e, dan pembesaran ruang baru di tunjukkan pada tabel 3.33

Tabel 3.33 Pembatas ruang baru(NBmbr) R5 terhadap *node* anak

R5		Bmbr		NBmbr
Entry	**MBR**	***Entry***	**MBR**	
5	34.0,46.0,42.5,52.5	0	0.0 ,39.0 ,47.5 ,100.0	0.0 46.0 42.5 100.0
		1	50.0 ,57.0 ,64.5 ,100.0	34.0 57.0 42.5 100.0

Proses selanjutnya menghitung selisih ruang kosong, berdasarkan langkah 1g ditunjukkan pada tabel 3.34

Tabel 3.34 Selisih ruang kosong

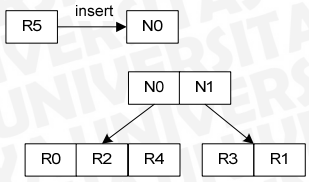
Entry	**NBmbr**	**Bmbr**	**Selisih**
0	2645	2047,5	597,5
1	1322,5	248,5	1074

Proses selanjutnya adalah optimasi dan minimasi tumpang tindih berdasarkan langkah 1h, di hasilkan perhitungan pada tabel 3.35

Tabel 3.35 Perbandingan tumpang tindih antara *Entry* MBR

Entry	***Entry*** **Bmbr/Nmbr**	**Keterangan**		**Ub-Lb**			**Selisih**
		x	**y**	**x**	**y**	**overl**	
0	(N0.1)	tidak	R1 ub di R2	0	35,5	0	0
	(0.1)	tidak	R1 ub di R2	0	35,5	0	
1	(N0.1)	R1 lb di R2	R1 ub di R2	5	52,5	262,5	262,5
	(1.0)	tidak	R1ub dan lb di R2	0	35,5	0	

Dari hasil perhitungan tabel 3.35 perbandingan tumpang tindih antara *Entry* MBR dan ruang kosong *Entry* pembesaran *node* anak terhadap *Entry* masukan R5 yang optimal, berdasarkan langkah 1i, di tunjukkan pada *Entry* pembesaran *node* anak baru nol, masukkan R5 ke *node* anak 0 dengan pembesaran area MBR *node* anak baru adalah 0.0,46.0,42.5,100.0, proses pemasukan setelah ChooseSubtree di tunjukkan pada **Gambar 3.57** berikut:



Gambar 3.57 Struktur R*-tree *reinsert* R5 ke *node* anak nol setelah ChooseSubtree

Karena pemasukan *Entry* R5 ke dalam *node* anak tersebut menyebabkan overflow dimana m>3, karena ini adalah pertama kali selama proses pemasukan R5 maka lakukan *reinsert*, berdasarkan langkah 2a, seluruh *Entry* didalam *node* di tunjukkan pada tabel 3.36

Tabel 3.36 *Entry node* MBR

Entry	**MBR(xl,xu,yl,yu)**
0	0.0,11.0,84.0,100.0
2	31.0,39.0,78.5,89.5
4	23.0,31.0,47.5,57.5
5	34.0,46.0,42.5,52.5

Menentukan pusat utama pembatas dari seluruh *Entry* (CBmbr) dengan langkah 2.b adalah {0.0, 46.0, 42,5, 100.0} dengan sumbu pusat pembatas x= 23.0 dan y=71.25

Proses selanjutnya menghitung pusat dari setiap calon *Entry* (Cmbr) berdasarkan langkah 2.c, seperti di tunjukkan pada tabel 3.37

Tabel 3.37 Sumbu pusat setiap calon *Entry*

Entry	**MBR(xl,xu,yl,yu)**	**Pusat x,y(Cmbr)**	
		xu+xl /2	**yu+yl/2**
0	0.0,11.0,84.0,100.0	5,5	92
2	31.0,39.0,78.5,89.5	35	84
4	23.0,31.0,47.5,57.5	27	52,5
5	34.0,46.0,42.5,52.5	40	47,5

Menghitung jarak pusat ke pusat secara keseluruhan berdasarkan langkah 2.d di tunjukkan pada tabel 3.38

Tabel 3.38 Jarak pusat ke pust *Entry*

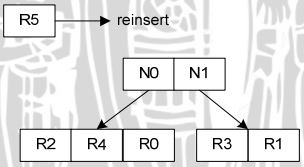
Cmbr - CBmbr		Cx^2	Cy^2	DC
Cx	Cy			
-17,5	20,75	306,25	430,5625	736,8125
12	12,75	144	162,5625	306,5625
4	-18,75	16	351,5625	367,5625
17	-23,75	289	564,0625	853,0625

Proses terakhir adalah mengurutkan calon *Entry* MBR berdasarkan langkah 2.e, di tunjukkan oleh tabel 3.39

Tabel 3.39 *Sorting Entry* berdasarkan jarak pusat (DC)

Entry	MBR	Jarak Pusat (DC)
2	31.0,39.0,78.5,89.5	306,5625
4	23.0,31.0,47.5,57.5	367,5625
0	0.0,11.0,84.0,100.0	736,8125
5	34.0,46.0,42.5,52.5	853,0625

Berdasarkan hasil *sorting* pusat mbr dengan langkah 2.f dimana jumlah *Entry* 4 maka 70% dari 4 adalah 3 *Entry* dan 30% adalah satu *Entry* dengan demikian *Entry* R5={34.0,46.0,42.5,52.5} pada **Gambar 3.58** terpilih untuk di *reinsert* kembali



Gambar 3.58 Struktur R*-tree pemasukan kembali R5 setelah minimal jarak pusat

Identifikasi root pada *nodeleaf* untuk me*reinsert* R5, karena root pada *node* leaf tidak kosong panggil Algoritma ChooseSubtree berdasarkan langkah 1a keseluruhan *Entry* ditunjukan pada tabel 3.40

Tabel 3.40 *Entry* MBR pembatas *node* anak

Entry	*Bounces*
0	0.0 ,39.0 ,47.5 ,100.0

Entry	*Bounces*
1	50.0 ,57.0 ,64.5 ,100.0

Proses ChooseSubtree pengecekan R5 terhadap *node* anak pada langkah 1b di tunjukkan pada tabel 3.41

Tabel 3.41 Pengecekan *node* anak

R5		BmBr		*Overlap / Inside*
Entry	MBR	*Entry*	MBR	
5	34.0,46.0,42.5,52.5	0	0.0 ,39.0 ,47.5 ,100.0	tidak
		1	50.0 ,57.0 ,64.5 ,100.0	tidak

Karena R5 tidak berada di dalam *node* anak dan *Entry* R5 dimasukan ke *nodeleaf* berdasarkan langkah 1.e pembesaran ruang baru di tunjukkan pada tabel 3.42

Tabel 3.42 Pembatas ruang baru(NBmbr) R5 terhadap *node* anak

R5		Bmbr		NBmbr
Entry	MBR	*Entry*	MBR	
5	34.0,46.0,42.5,52.5	0	0.0 ,39.0 ,47.5 ,100.0	0.0 46.0 42.5 100.0
		1	50.0 ,57.0 ,64.5 ,100.0	34.0 57.0 42.5 100.0

Proses selanjutnya menghitung selisih ruang kosong berdasarkan langkah 1.g ditunjukkan pada tabel 3.43

Tabel 3.43 Selisih ruang kosong

Entry	NBmbr	Bmbr	Selisih
0	2645	2047,5	597,5
1	1322,5	248,5	1074

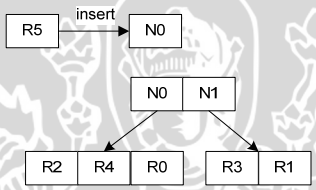
Proses selanjutnya adalah optimasi dan minimasi tumpang tindih dengan langkah 1.h di hasilkan perhitungan pada tabel 3.44

Tabel 3.44 Perbandingan tumpang tindih antara *Entry* MBR

Entry	Bmbr/ Nmbr	keterangan		Ub-Lb			Selisih
		x	y	x	y	overlape	
0	(N0.1)	tidak	R1 ub di R2	0	35,5	0	0
	(0.1)	tidak	R1 ub di R2	0	35,5	0	

Entry	Bmbr/ Nmbr	keterangan		Ub-Lb			Selisih
		x	y	x	y	overlape	
1	(N0.1)	R1 lb di R2	R1 ub di R2	5	52,5	262,5	262,5
	(1.0)	tidak	R1ub dan lb di R2	0	35,5	0	

Dari hasil perhitungan tabel perbandingan tumpang tindih antara *Entry* MBR dan ruang kosong *Entry* pembesaran *node* anak terhadap *Entry* masukan R5 yang optimal, berdasarkan langkah 1.i adalah di tunjukkan pada *Entry* pembesaran *node* anak baru ke nol, dengan demikian masukkan R5 ke *bounces* 0 dengan pembesaran area *bounces* {0.0,46.0,42.5,100.0} proses pemasukan setelah ChooseSubtree di tunjukkan pada **Gambar 3.59** berikut:



Gambar 3.59 Struktur R*-tree *reinsert Entry* 5 ke *node* anak nol setelah minimal jarak pusat di lakukan

Karena *bounces* ke 0 kapasitas *Entry nodeleaf* sudah penuh, setelah pemasukan *Entry* R5 menyebabkan Overflow, Karena selama proses memasukkan R5 bukan proses pertama kali selama pemasukan di tingkat *tree*, berdasaran langkah 3 lakukan *split* untuk menempatkan R5.

Proses *split* pada langkah 3.1 keseluruhan *Entry* yang akan didistribusikan ditunjukkan pada tabel 3.45

Tabel 3.45 *Entry Split* pemasukan R5

Entry	MBR
2	31.0,39.0,78.5,89.5
4	23.0,31.0,47.5,57.5
0	0.0,11.0,84.0,100.0
5	34.0,46.0,42.5,52.5

Untuk menentukan sumbu *splitAxis* optimasi margin Terbaik, berdasarkan langkah 3.2a pengurutan *entry* sumbu tegak lurus batas bawah xl ditunjukkan pada tabel 3.46 dan pengurutan sumbu tegak lurus batas atas xu di tunjukkan pada tabel 3.47.

Tabel 3.46 Pengurutan batas bawah Xl

Entry	**MBR**	**Xl**
0	0.0,11.0,84.0,100.0	0
Entry	**MBR**	**Xl**
4	23.0,31.0,47.5,57.5	23
2	31.0,39.0,78.5,89.5	31
5	34.0,46.0,42.5,52.5	34

Tabel 3.47 Pengurutan batas atas Xu

Entry	**MBR**	**Xu**
0	0.0,11.0,84.0,100.0	11
4	23.0,31.0,47.5,57.5	31
2	31.0,39.0,78.5,89.5	39
5	34.0,46.0,42.5,52.5	46

Berdasarkan kedua data *sorting* terendah dari sumbu batas atas dengan langkah 3.2b dimana m adalah 40% dari M dan M=4, terbentuk 3 distribusi(k), ditunjukkan pada tabel 3.48

Tabel 3.48 Distribusi kelompok setiap *Entry*

Sumbu	**(k)**	**G1 (m+k)**		**G2(M-m-k)**	
		Entry	**MBR**	***Entry***	**MBR**
SML	0	0	0.0, 11.0, 84.0, 100.0	4	23.0, 31.0, 47.5, 57.5
				2	31.0, 39.0, 78.5, 89.5
				5	34.0, 46.0, 42.5, 52.5
	1	0	0.0,11.0,84.0,100.0	2	31.0,39.0,78.5,89.5
		4	23.0,31.0,47.5,57.5	5	34.0,46.0,42.5,52.5
	2	0	0.0,11.0,84.0,100.0	5	34.0,46.0,42.5,52.5
		4	23.0,31.0,47.5,57.5		
		2	31.0,39.0,78.5,89.5		
SMU	0	0	0.0, 11.0, 84.0, 100.0	4	23.0, 31.0, 47.5, 57.5

Sumbu	(k)	G1 (m+k)		G2(M-m-k)	
		Entry	MBR	*Entry*	MBR
SMU	0			2	31.0, 39.0, 78.5, 89.5
				5	34.0, 46.0, 42.5, 52.5
	1	0	0.0,11.0,84.0,100.0	2	31.0,39.0,78.5,89.5
		4	23.0,31.0,47.5,57.5	5	34.0,46.0,42.5,52.5
	2	0	0.0,11.0,84.0,100.0	5	34.0,46.0,42.5,52.5
		4	23.0,31.0,47.5,57.5		
		2	31.0,39.0,78.5,89.5		

Hasil distribusi pembatas kelompok berdasarkan langkah 3.2.c ditunjukkan pada tabel 3.49

Tabel 3.49 Distribusi perluasan setiap kelompok *Entry*

Sumbu	(k)	BG1	BG2
SML	0	0.0, 11.0, 84.0, 100.0	23.0, 46.0, 42.5, 89.5
	1	0.0, 31.0, 47.5, 100.0	31.0, 46.0, 42.5, 89.5
	2	0.0, 39.0, 47.5, 100.0	34.0, 46.0, 42.5, 52.5
SMU	0	0.0, 11.0, 84.0, 100.0	23.0, 46.0, 42.5, 89.5
	1	0.0, 31.0, 47.5, 100.0	31.0, 46.0, 42.5, 89.5
	2	0.0, 39.0, 47.5, 100.0	34.0, 46.0, 42.5, 52.5

Kemudian dari hasil tiap-tiap pembatas kelompok dihitung pembesaran dari *margin*, berdasarkan langkah 3.2d, hasil pembesaran dari *margin* ditunjukkan pada tabel 3.50

Tabel 3.50 Optimasi *margin* setiap kelompok *Entry*

Sumbu	(k)	MG1	MG2	Jumlah
SML	0	27	70	97
	1	83,5	62	145,5
	2	91,5	22	113,5
SMU	0	27	70	97
	1	83,5	62	145,5
	2	91,5	22	113,5
Total *margin index* pertama				712

Hasil perhitungan akhir dari optimasi *margin index* pertama adalah 712, kemudian berdasarkan langkah 2.3. e di tunjukkan pada tabel 3.51 dan 3.52

Tabel 3.51 Pengurutan batas bawah Yl

Entry	MBR	yl
5	34.0,46.0,42.5,52.5	42,5
4	23.0,31.0,47.5,57.5	47,5
2	31.0,39.0,78.5,89.5	78,5
0	0.0,11.0,84.0,100.0	84

Tabel 3.52 Pengurutan batas atas Yu

Entry	MBR	yu
5	34.0,46.0,42.5,52.5	52,5
4	23.0,31.0,47.5,57.5	57,5
2	31.0,39.0,78.5,89.5	89,5
0	0.0,11.0,84.0,100.0	100

Berdasarkan kedua data *sorting* dari sumbu batas atas dengan langkah 3.2b dimana m adalah 40% dari M dan M=4, terbentuk 3 distribusi(k), ditunjukkan pada tabel 3.53

Tabel 3.53 Distribusi kelompok setiap *Entry*.

Sumbu	(k)	G1 (m+k)		G2 (M-m-k)	
		Entry	MBR	*Entry*	MBR
SML	0	5	34.0,46.0,42.5,52.5	4	23.0,31.0,47.5,57.5
				2	31.0,39.0,78.5,89.5
				0	0.0,11.0,84.0,100.0
	1	5	34.0,46.0,42.5,52.5	2	31.0,39.0,78.5,89.5
		4	23.0,31.0,47.5,57.5	0	0.0,11.0,84.0,100.0
	2	5	34.0,46.0,42.5,52.5	0	0.0,11.0,84.0,100.0
		4	23.0,31.0,47.5,57.5		
		2	31.0,39.0,78.5,89.5		
SMU	0	5	34.0,46.0,42.5,52.5	4	23.0,31.0,47.5,57.5
				2	31.0,39.0,78.5,89.5
				0	0.0,11.0,84.0,100.0
	1	5	34.0,46.0,42.5,52.5	2	31.0,39.0,78.5,89.5
		4	23.0,31.0,47.5,57.5	0	0.0,11.0,84.0,100.0
	2	5	34.0,46.0,42.5,52.5	0	0.0,11.0,84.0,100.0
		4	23.0,31.0,47.5,57.5		
		2	31.0,39.0,78.5,89.5		

Hasil distribusi perluasan kelompok berdasarkan langkah 3.2.c ditunjukkan pada tabel 3.54

Tabel 3.54 Distribusi perluasan setiap kelompok *Entry*

Sumbu	(k)	BG1	BG2
SML	0	34.0,46.0,42.5,52.5	0.0, 39.0, 47.5, 100.0
	1	23.0, 46.0, 42.5, 57.5	0.0, 39.0, 78.5, 100.0
	2	23.0, 46.0, 42.5, 89.5	0.0,11.0,84.0,100.0
SMU	0	34.0,46.0,42.5,52.5	0.0, 39.0, 47.5, 100.0
	1	23.0, 46.0, 42.5, 57.5	0.0, 39.0, 78.5, 100.0
	2	23.0, 46.0, 42.5, 89.5	0.0,11.0,84.0,100.0

Kemudian dari hasil tiap-tiap pembatas kelompok dengan langkah 3.2d hasil pembesaran dari *margin* ditunjukkan pada tabel 3.55

Tabel 3.55 Distribusi *margin* perluasan setiap kelompok *Entry*

Sumbu	(k)	MG1	MG2	Jumlah
SML	0	22	91,5	113,5
	1	38	60,5	98,5
	2	70	27	97
SMU	0	22	91,5	113,5
	1	38	60,5	98,5
	2	70	27	97
Total *margin index* ke dua				618

Hasil perhitugan akhir dari *margin* kelompok *index* Kedua adalah 712, bila dibandingkan hasil dari *index* pertama adalah lebih besar dari *margin index* kelompok pertama yaitu 618, dengan demikian Kelompok *index* pertama adalah *margin* yang paling minimal dengan *index* pembelahan terbaik dan terpilih untuk dihitung area mati dan tumpang tindih minimal pada Algoritma ChooseSplitIndex berdasarkan langkah 3.3.

, Perhitungan area *Entry* di setiap kelompok distribusi berdasarkan langah 3.3.a seperti di tunjukkan pada tabel 3.56

Tabel 3.56 Distribusi area *Entry*.

Sumbu	(k)	G1 (m+k)		G2 (M-m-k)	
		Entry	**Area**	***Entry***	**Area**
SML	0	5	120	4	80
				2	88
				0	176
	1	5	120	2	88
		4	80	0	176
	2	5	120	0	176
		4	80		
		2	88		
SMU	0	5	120	4	80
				2	88
				0	176
	1	5	120	2	88
		4	80	0	176
	2	5	120	0	176
		4	80		
		2	88		

Perhitungan area selanjutnya berdasarkan langkah 3.3.b, area pembatas kelompok *index* pertama ditunjukkan pada tabel 3.57.

Tabel 3.57 Distribusi optimasi area kelompok *Entry*

Sumbu	(k)	Area		
		BG1	**BG2**	**Barea**
SML	0	120	2047,5	2167,5
	1	345	838	1183
	2	1081	176	1257
SMU	0	120	2047,5	2167,5
	1	345	838	1183
	2	1081	176	1257

Perhitungan minimal Area berdasarkan langkah 3.3.c di tunjukkan pada tabel 3.58.

Tabel 3.58 Distribusi minimasi ruang kelompok *Entry*

Sumbu	(k)	g1,g2	*Entry*	Area		
				Barea	Garea	Marea
SML	0	1,3	(5),(4,2,0)	2167,5	464	1703,5
	1	2,2	(5,4),(2,0)	1183	464	719
	2	3,1	(5,4,2),(0)	1257	464	793
SMU	0	1,3	(5),(4,2,0)	2167,5	464	1703,5
	1	2,2	(5,4),(2,0)	1183	464	719
	2	3,1	(5,4,2),(0)	1257	464	793

Proses terakhir menghitung tumpang tindih minimal di setiap kelompok dengan langkah 3.3d, ditunjukkan pada tabel 3.59

Tabel 3.59 Distribusi Tumpang tindih setiap kelompok *Entry*

Sumbu	(k)	keterangan		Ub-Lb		Moverlap
		x	y	x	y	
SML	0	R1lb di R2	R1ub di R2	5	5	25
	1	R1lb di R2	tidak	16	0	0
	2	tidak	R1ub di R2	0	5,5	0
SMU	0	R1lb di R2	R1ub di R2	5	5	25
	1	R1lb di R2	tidak	16	0	0
	2	tidak	R1ub di R2	0	5,5	0

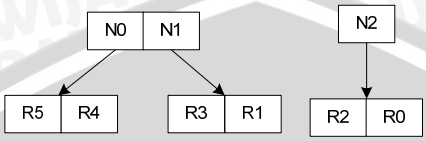
Dari hasil perhitungan akhir tumpang tindih (Moverlap) dan ruang mati (Marea) berdasarkan langkah 3.3e ditunjukkan pada tabel 3.60

Tabel 3.60 Perbandingan ruang mati dan tumpang tindih *index* pertama

Sumbu	(K)	G1,G2	*Entry*	Marea	Moverlap
SML	1	2,2	(5,4),(2,0)	719	0
SMU	1	2,2	(5,4),(2,0)	719	0
SML	2	3,1	(5,4,2),(0)	793	0
SMU	2	3,1	(5,4,2),(0)	793	0
SML	0	1,3	(5),(4,2,0)	1703,5	25
SMU	0	1,3	(5),(4,2,0)	1703,5	25

Dengan demikian berdasarkan hasil perhitungan akhir optimasi dan minimasi ruang mati serta tumpang tindih terbaik untuk didistribusikan pengelompokan pembelahan adalah *Entry* 5 dan 4 pada kelompok distribusi

pertama , kemudian *Entry* 2 dan 0 untuk distribusi kelompok ke dua pada sumbu sml dengan distribusi(k) adalah 1, hasil distribusi dan optimasi pembelah terbaik di tunjukkan pada **Gambar 3.60** berikut:



Gambar 3.60 Struktur R*-tree pemasukan *Entry* R5 setelah distribusi optimasi dan minimasi pembelahan terbaik

Karena terjadi *split* dari *nodeleaf* setelah pemasukan ulang dan *bounces* baru dari *nodeleaf* terbentuk setelah pembelahan, maka buat root baru dari *bounces* tersebut, berkenaan dengan m=2 dan *bounces* sekaranag adalah 3 menyebabkan overflow dan panggil algoritma *split* berdasarkan langkah ke- 3 untuk pembelahan, jumlah keseluruhan *Entry* dalam *node* berdasarkan langkah 3.1 ditunjukkan pada tabel 3.61

Tabel 3.61 *Entry node* anak

Entry	**MBR *Bounces***
0	23.0, 46.0, 42.5, 57.5
1	50.0, 57.0, 64.5, 100.0
2	0.0, 39.0, 78.5, 100.0

Proses selanjutnya menentukan *index* pembelahan *node* terbaik dengan Algoritma ChooseSplitAxis, berdasarkan langkah 3.2a ditunjukkan pada tabel 3.62 dan tabel 3.63

Tabel 3.62 Pengurutan batas bawah Xl

Entry	**MBR *Bounces***	**Xl**
2	0.0, 39.0, 78.5, 100.0	0
0	23.0, 46.0, 42.5, 57.5	23
1	50.0, 57.0, 64.5, 100.0	50

Tabel 3.63 Pengurutan batas atas Xu

Entry	MBR *Bounces*	Xu
2	0.0, 39.0, 78.5, 100.0	39
0	23.0, 46.0, 42.5, 57.5	46
1	50.0, 57.0, 64.5, 100.0	57

Berdasarkan kedua data *sorting* terendah dari sumbu batas bawah SML dan atas SMU, langkah 3.2.b dimana m adalah 40% dari M, dengan demikian terbentuk 2 distribusi(k) dari 3 *Entry*, distribusi kelompok setiap *Entry* ditunjukkan pada tabel 3.64

Tabel 3.64 Distribusi kelompok setiap *Entry*

Sumbu	(k)	G1 (m+k)		G2 (M-m-k)	
		Entry	MBR	*Entry*	MBR
SML	0	2	0.0, 39.0, 78.5, 100.0	0	23.0, 46.0, 42.5, 57.5
				1	50.0, 57.0, 64.5, 100.0
	1	2	0.0, 39.0, 78.5, 100.0	1	50.0, 57.0, 64.5, 100.0
		0	23.0, 46.0, 42.5, 57.5		
SMU	0	2	0.0, 39.0, 78.5, 100.0	0	23.0, 46.0, 42.5, 57.5
				1	50.0, 57.0, 64.5, 100.0
	1	2	0.0, 39.0, 78.5, 100.0	1	50.0, 57.0, 64.5, 100.0
		0	23.0, 46.0, 42.5, 57.5		

Hasil distribusi kelompok berdasarkan langkah 3.2.c ditunjukkan pada tabel 3.65

Tabel 3.65 Distribusi perluasan setiap kelompok *Entry*

Sumbu	(k)	BG1	BG2
SML	0	0.0, 39.0, 78.5, 100.0	23.0, 57.0, 42.5, 100.0
	1	0.0, 46.0, 42.5, 100.0	50.0, 57.0, 64.5, 100.0
SMU	0	0.0, 39.0, 78.5, 100.0	23.0, 57.0, 42.5, 100.0
	1	0.0, 46.0, 42.5, 100.0	50.0, 57.0, 64.5, 100.0

Kemudian dari hasil tiap-tiap pembatas kelompok dihitung pembesaran dari *margin*, berdasarkan langkah 3.2d, hasil pembesaran dari *margin* ditunjukkan pada tabel 3.66

Tabel 3.66 Optimasi *margin* setiap kelompok *Entry*

Sumbu	(k)	MG1	MG2	Jumlah
SML	0	60,5	91,5	152
	1	103,5	42,5	146
SMU	0	60,5	91,5	152
	1	103,5	42,5	146
Total *margin index* pertama				596

Hasil perhitugan akhir dari optimasi *margin index* pertama adalah 596, kemudian berdasarkan langkah 2.3. e di tunjukkan pada tabel 3.67 dan 3.68

Tabel 3.67 Pengurutan batas bawah Yl

Entry	MBR *Bounces*	Yl
0	23.0, 46.0, 42.5, 57.5	42,5
1	50.0, 57.0, 64.5, 100.0	64,5
2	0.0, 39.0, 78.5, 100.0	78,5

Tabel 3.68 Pengurutan batas atas Yu

Entry	MBR *Bounces*	Yu
0	23.0, 46.0, 42.5, 57.5	57,5
2	0.0, 39.0, 78.5, 100.0	100
1	50.0, 57.0, 64.5, 100.0	100

Berdasarkan kedua data *sorting* terendah dari sumbu batas atas dengan langkah 3.2b dimana m adalah 40% dari M dengan M=3, terbentuk 2 distribusi(k), distribusi kelompok setiap *Entry* ditunjukkan pada tabel 3.69.

Tabel 3.69 Distribusi kelompok setiap *Entry*

Sumbu	(k)	G1 (m+k)		G2 (M-m-k)	
		Entry	MBR	*Entry*	MBR
SML	0	0	23.0, 46.0, 42.5, 57.5	1	50.0, 57.0, 64.5, 100.0
				2	0.0, 39.0, 78.5, 100.0
	1	0	23.0, 46.0, 42.5, 57.5	2	0.0, 39.0, 78.5, 100.0
		1	50.0, 57.0, 64.5, 100.0		
SMU	0	0	23.0, 46.0, 42.5, 57.5	2	0.0, 39.0, 78.5, 100.0
				1	50.0, 57.0, 64.5, 100.0
	1	0	23.0, 46.0, 42.5, 57.5	1	50.0, 57.0, 64.5, 100.0
		2	0.0, 39.0, 78.5, 100.0		

Hasil distribusi pembatas kelompok berdasarkan langkah 3.2.c ditunjukkan pada tabel 3.70

Tabel 3.70 Distribusi perluasan setiap kelompok *Entry*

Sumbu	(k)	BG1	BG2
SML	0	23.0, 46.0, 42.5, 57.5	0.0, 57.0, 64.5, 100.0
	1	23.0, 57.0, 42.5, 100.0	0.0, 39.0, 78.5, 100.0
SMU	0	23.0, 46.0, 42.5, 57.5	0.0, 57.0, 64.5, 100.0
	1	0.0, 46.0, 42.5, 100.0	50.0, 57.0, 64.5, 100.1

Kemudian dari hasil tiap-tiap pembatas kelompok dengan langkah 3.2d hasil pembesaran dari *margin* ditunjukkan tabel 3.71

Tabel 3.71 Distribusi *margin* perluasan setiap kelompok *Entry*

Sumbu	(k)	MG1	MG2	Jumlah
SML	0	38	92,5	130,5
	1	91,5	60,5	152
SMU	0	38	92,5	130,5
	1	103,5	42,5	146
Total *margin index* ke dua				559

Hasil perhitugan akhir dari *margin* kelompok *index* Kedua adalah 559, bila dibandingkan hasil dari *index* pertama adalah lebih kecil dari *margin index* kelompok pertama yaitu 596, dengan demikian Kelompok *index* kedua adalah *margin* yang paling minimal dengan *index* pembelahan terbaik dan terpilih untuk dihitung area mati dan tumpang tindih minimal pada Algoritma ChooseSplitIndex berdasarkan langkah 3.3.

Berdasarkan langah 3.3.a, perhitungan area *Entry* di setiap kelompok distribusi seperti di tunjukkan pada tabel 3.72.

Tabel 3.72 Distribusi area *Entry*.

Sumbu	(k)	G1 (m+k)		G2 (M-m-k)	
		Entry	**Area**	***Entry***	**Area**
SML	0	0	345	1	248,5
				2	838,5
SML	1	0	345	2	838,5
		1	248,5		

Perhitungan area selanjutnya berdasarkan langkah 3.3.b, area pembatas kelompok *index* ke dua ditunjukkan pada tabel 3.73

Tabel 3.73 Distribusi optimasi *margin* kelompok *Entry*

Sumbu	(k)	Area		
		BG1	**BG2**	**Barea**
SML	0	345	2023,5	2368,5
	1	1955	838,5	2793,5
SMU	0	345	2023,5	2368,5
	1	2645	248,5	2893,5

Perhitungan minimal Area berdasarkan langkah 3.3.c di tunjukkan pada 3.74

Tabel 3.74 Distribusi dan minimasi ruang kelompok *Entry*

Sumbu	(k)	g1,g2	***Entry***	Area		
				BArea	**Garea**	**Marea**
SML	0	1,2	(0),(1,2)	1432	2368,5	936,5
	1	2,1	(0,1),(2)	1432	2793,5	1361,5
SMU	0	1,2	(0),(1,2)	1432	2368,5	936,5
	1	2,1	(0,1),(2)	1432	2893,5	1461,5

Proses terakhir menghitung tumpang tindih minimal di setiap kelompok dengan langkah 3.3d, ditunjukkan pada tabel 3.75.

Tabel 3.75 Distribusi Tumpang tindih setiap kelompok *Entry*

Sumbu	(k)	Keterangan		Ub-Lb		Moverlap
		x	**y**	**x**	**y**	
SML	0	R1ub dan lb di R2	tidak	23	0	0
	1	R1 lb di R2	R1ub di R2	16	21,5	344
SMU	0	R1ub dan lb di R2	tidak	23	0	0

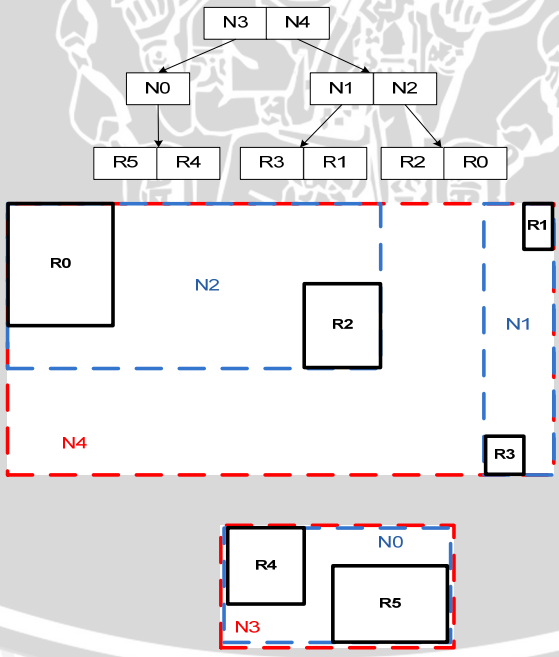
Sumbu	(k)	Keterangan		Ub-Lb		Moverlap
		x	y	x	y	
SMU	1	tidak	R1ub di R2	0	35,5	0

Dari hasil perhitungan akhir tumpang tindih (Moverlap) dan ruang mati (Marea) *index* ke dua, berdasarkan langkah 3.3e ditunjukkan pada tabel 3.76

Tabel 3.76 Perbandingan ruang mati dan tumpang tindih *index* kedua

Sumbu	(k)	g1,g2	*Entry*	Marea	Mover
SML	0	1,2	(0),(1,2)	936,5	0
SML	1	2,1	(0,1),(2)	936,5	0
SMU	0	1,2	(0),(1,2)	1361,5	0
SMU	1	2,1	(0,1),(2)	1461,5	344

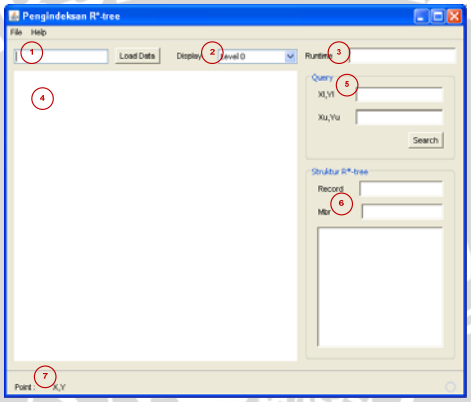
Ilustrasi susunan *node* terakhir sebagai contoh perhitungan untuk pengindeksan R*-tree berdasarkan tumpang tindih dan area mati minimal dari tabel 3.76 ditunjukan pada **Gambar 3.61** berikut.



Gambar 3.61 Akhir struktur R*-tree setelah distibusi optimasi dan minimasi pembelahan *node* anak terbaik

3.5 Perancangan User Interface

Perancangan antar muka pengindeksan data spasial menggunakan struktur data R*-tree terdiri dari tiga fungsi, yaitu meload data uji dan menampilkan data dalam bentuk map, melakukan *query* untuk uji runtime *query* dan mencatat masing-masing waktu dari hasil uji maupun *index*ing serta menampilkan struktur *index*ing yang terbentuk dalam strukturdata R*-tree, gambar antarmuka pengindeksan ditunjukkan pada **Gambar 3.62** berikut :



Gambar 3.62 Perancangan *User Interface* Pengindeksan R*-tree

Keterangan gambar:

1. Textfield yang disertai button digunakan untuk meload data teks koordinat geometri spasial uji coba
2. ComboBox yang digunakan untuk menampilkan hasil indeksing spasial dalam bentuk peta dan MBR dengan kriteria level yang terdapat pada struktur R*-tree, dan juga digunakan untuk menampilkan hasil *query*.
3. Textfield runtime yang menginformasikan waktu hasil uji coba *Indexing* maupun *Query* .
4. Canvas digunakan untuk menampilkan peta dan MBR hasil uji coba *Indexing* maupun *Query*.

5. Textfield yang digunakan untuk memasukan parameter Xl dan Yl serta button search untuk uji coba *query*.
6. Textfield *record* dan MBR yang menginformasikan jumlah *record* dan MBR yang digunakan selama uji *indexing* atau *quey,* dan TexArea yang menginformasikan struktur MBR dalam R*-tree
7. textField *point* x,y digunakan untuk mengetahui koordinat pada canvas saat mouse berada di canvas.

3.6 Perancangan Proses Perbandingan

Untuk melakukan uji coba perbandingan yaitu di lakukan ujicoba berdasarkan presentase yang di gunakan untuk reintegrasi pada strukturdata R*-tree yang akan di lakukan query setelah dilakukan pengindeksan.

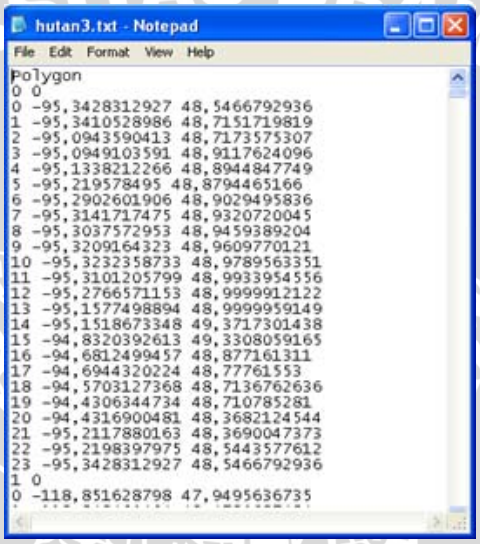
Besar waktu yang di butuhkan selama pengindeksan yang dilakukan terhadap masing-masing prosentase reinsert atau pemasukan kembali untuk perbaikan struktur data (reintegrasi) dengan masing-masing tipe data polygon dan polyline, dengan masing-masing presentase reinsert sebesar 30%,50% dan 70%.

Berdasarkan prosentase reinsert yang bervariasi dari pengindeksan akan dilakukan query dengan membandingkan setiap hasil runtime selama proses query.

Hasil analisa runtime dari query menunjukkan waktu yang relative kecil atau rata-rata waktu hampir sama pada hasil setiap presentase reinsert, maka besar prosentase reinsert yang digunakan pada metode pengindeksan R*-tree tersebut pada data spasial mempunyai kemampuan query yang lebih baik.

3.6.1 Bentuk File Teks Geom.

Masing - masing file teks geom berisi satu fiture spasial dengan tipe yang sama, sebagai contoh pada **Gambar 3.63** adalah fiture geometri hutan yang bertipe polygon, pada baris pertama menunjukkan tipe data geom, dan baris kedua menunjukkan id dari geom, kemudian id *record* dengan jumlah 23 *record* pada tiap *record* berisi geometri latitude dan longitude dengan *point* masing-masing geometri.



Gambar 3.63 Bentuk geometri polygon

3.6.2 Perancangan Uji coba

Simulasi yang akan di lakukan selama uji coba pengindeksan maupun *query* akan dilakukan sesuai aturan berikut :

- Pada data yang ditentukan akan dilakukan pengindeksan dengan metode indeksing R*-tree yang dilanjutkan dengan *query*.
- Pada masing-masing pengindeksan digunakan prosentase *reinsert* 30%, 50% dan 70% pemasukan kembali untuk perbaikan struktur data.

- *Query* yang digunakan dari masing-masing pengindeksan dengan *reinsert* yang bervariasi adalah range *query*, dimana satu obyek berada dalam radius jarak tertentu dengan obyek lain, *query* ini di pilih karena paling banyak di lakukan oleh operator SIG.
- Pengindeksan maupun *Query* akan dilakukan terhadap data yang bertipe Polygon dan data bertipe polyline
- Dari masing-masing perlakuan tersebut akan dicatat waktu yang dibutuhkan selama pengindeksan begitu juga untuk waktu yang di butuhkan selama proses *query* dengan masing-masing record yang dihasilkan.
- Dari pengukuran tersebut akan di analisa hasil dari pengindeksan dan *query* dengan R*-tree untuk kemudian diketahui faktor penyebabnya dari masing-masing uji coba.
- Masing-masing bentuk tabel runtime pengindeksan pada data bertipe polygon dan polyline terlihat seperti berikut

Tabel 3.77Runtime pengindeksan

Reinsert	Runtime
30%	
50%	
70%	

- Bentuk tabel koordinat uji *query* dari setiap query dengan reinsert 30%, 50% dan 70% terlihat seperti tabel 3.78, dimana X1 X2 dan Y1, Y2 mewakili Xl Xu dan Yl Yu yang ada pada masing-masing sumbu MBR.

Tabel 3.78 Koordinat uji coba query

Uji Ke	X1	X2	Y1	Y2
1				
2				
3				
…				

- Bentuk tabel runtime hasil *query* dari setiap pengindeksan dengan berdasarkan jumlah record dari koordinat pencarian dengan reinsert 30%, 50% dan 70% terlihat seperti berikut:

Tabel 3.79 Runtime *query*

Jumlah *Record*	Runtime reinsert		
	30%	50%	70%
200			
300			
400			
…			

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

Pada bab ini akan dijelaskan seluruh proses yang sudah dirancang pada bab sebelumnya, tampilan dan bagian–bagian *source code* yang dibuat, serta analisa terhadap data yang dihasilkan sistem.

4.1 Lingkungan Implementasi

Implementasi merupakan proses transformasi representasi rancangan ke dalam bahasa pemrograman yang dapat dimengerti oleh komputer. Pada bab ini, lingkungan implementasi yang akan dijelaskan meliputi lingkungan implementasi perangkat keras dan perangkat lunak.

4.1.1 Lingkungan perangkat keras

Perangkat keras yang digunakan dalam pengujian pengindeksan data spasial menggunakan struktur data R*-tree ini adalah sebuah Notebook dengan spesifikasi sebagai berikut :

1. Monitor : 13,1”
2. CPU : Intel Core 2 Duo T5500, 2.40 GHz
3. Hard Disk : 120 GB
4. Memori : 2 GB
5. Sound Card : IDT
6. Perangkat Masukan : Keyboard, Mouse

Perangkat keras ini akan difungsikan sebagai tempat perangkat lunak untuk penelitian dijalankan.

4.1.2 Lingkungan Perangkat lunak

Perangkat lunak yang digunakan dalam pengindeksan data spasial menggunakan struktur data R*-tree ini adalah :

1. Sistem operasi *Windows XP* sebagai tempat aplikasi dijalankan.
2. *JAVA NETBean JDK 1.6* sebagai *programming software development* dalam pembuatan program pengindeksan data spasial menggunakan struktur data R*-tree.

4.2 Implementasi Program

Berdasarkan analisa dan perancangan proses yang telah dipaparkan pada Bab III, maka pada bab ini akan dijelaskan proses-proses implementasinya.

4.2.1 Implementasi load data Spasial

Tahap awal yang dilakukan dari pengindeksan adalah *load data Spasial yang berextensi Shp* yang diperoleh dari *ESRI Data*. Prosedur yang digunakan ditunjukkan pada *sourcecode* 4.1.

```
try{
   switch(rshp.getType())
   {
   case Shapefile.SHAPETYPE_POLYGON :
   t.shape_type=Constants.POLYGON;
   break;
   }
   Tuples tp;
   Iterator itrShapeObjects = rshp
   .getShapeObjects().iterator();
   while(itrShapeObjects.hasNext())
```

```
      {
        tp=new Tuples();
        ShapeObject obj =
        (ShapeObject)itrShapeObjects.next();
        if(obj.getType() == ShapeObject.POLYGON)
        {
          if(obj.getType() !=
           rshp.getType())
          {
          // bukan polygon
          }
          tp.shape_type=Constants.POLYGON;
          tp=translatePolygonObject(obj);
          d=tp.tp.getMBRTuple();
          lbox.add(d);
          rt.insert(d);
        }else{
        }
      }
    }catch(Exception e){
      e.printStackTrace();
    }
    dur.Stop();
    if(lbox.size()>0==false){
      System.out.println("Data tidak dapat
      di baca MBR = "+lbox.size());
      System.exit(0);
    }
f = new RStarFrame(this);
```

```
f.recordindex=totalrecord;
f.boxQuery=lbox;
f.timeIndex=dur.getDurasi()+" / detik";
f.jTextField1.setText(f.timeIndex);
f.jLabel1.setText("Runtime Indexing");
f.jTextField3.setText(f.recordindex+" Record");
f.jLabel6.setText("Jumlah Record Indexing");

f.setSize(600, 600);
f.pack();
f.show();
}
```

Sourcecode 4.1 Prosedur *Load data Shp*

4.2.2 Implementasi Proses MBR

Langkah selanjutnya adalah mengimplementasikan proses pembentukan MBR, pada prosedur ini akan diperoleh sekumpulan coordinat x dan y yang kemudian di ambil x,y min dan x,y max. Implementasi Proses MBR dapat dilihat pada *sourcecode* 4.2.

```
private Tuples translatePolygonObject(ShapeObject
obj)throws IOException
{
   Iterator itrPoints = obj.getPoints().iterator();
   Tuples tp = new Tuples();
   tp.tp =new tuple(obj.getPointCount());
   tp.MBR[0]=obj.getBoundingBox().getXMin();
   tp.MBR[1]=obj.getBoundingBox().getXMax();
   tp.MBR[2]=obj.getBoundingBox().getYMin();
   tp.MBR[3]=obj.getBoundingBox().getYMin();
```

```
   int i=0;
   while(itrPoints.hasNext())
   {
      Point pt = (Point)itrPoints.next();
      tp.tp.tupleX[i]=(pt.getX())*100;
      tp.tp.tupleY[i]=(pt.getY())*100;
      totalrecord++;
      i++;
   }
   Record rec = obj.getRecord();
   Iterator itrFields = rec.getFields().iterator();
   while(itrFields.hasNext())
   {
      RecordField recField = (RecordField)
      itrFields.next();
   }
   return tp;
}
```

Sourcecode 4.2 Prosedur MBR

4.2.3 Implementasi Proses Pengindeksan Insert R*-tree

Langkah selanjutnya adalah mengimplementasikan proses Pengindeksan dari data MBR kedalam struktur data. Pada prosedur ini akan di masukkan data entry MBR ke data node atau directory node yang memungkinkan untuk dimasukkan kembali (reinsert) yang di tampung pada linkedlist re_data_cands. Implementasi dari Insert dapat dilihat pada *sourcecode* 4.3

```
void insert(Data d){
   int i, j;
   RTNode sn[] = new RTNode[1];
```

```
RTDirNode nroot_ptr;
int nroot;
DirEntry de;
int split_root = Constants.NONE;
Data d_cand, dc;
float nMBR[];
load_root();
re_level = new boolean[root_ptr.level+1];

for (i = 0; i <= root_ptr.level; i++){
  re_level[i] = false;
}

dc = (Data)d.clone();
re_data_cands.add(dc);
j = -1;

while (re_data_cands.size() > 0)
{
   d_cand = (Data) re_data_cands.getFirst
   if (d_cand != null)
   {
      dc = (Data) d_cand.clone();
      re_data_cands.remove(d_cand);
      split_root = ((Node)root_ptr).insert(dc, sn);
   }
   else{
      Constants.error("RTree::insert:
       kesalahan pada reinsert", true);
   }
```

```
if (split_root == Constants.SPLIT){
  //inisialisasi root baru
  nroot_ptr = new RTDirNode(this);
  nroot_ptr.son_is_data = root_is_data;
  nroot_ptr.level = (short)(root_ptr.level + 1);
  nroot = nroot_ptr.block;
  de = new DirEntry(dimension,
    root_is_data, this);
  nMBR = ((Node)root_ptr).get_MBR();
  System.arraycopy(nMBR, 0, de.bounces,
    0, 2*dimension);
  de.son = root_ptr.block;
  de.son_ptr = root_ptr;
  de.son_is_data = root_is_data;
  de.num_of_data= ((Node)root_ptr)
    .get_num_of_data();
  nroot_ptr.enter(de);
  de = new DirEntry(dimension,
    root_is_data, this);
  nMBR = ((Node)sn[0]).get_MBR();
  System.arraycopy(nMBR, 0, de.bounces,
    0, 2*dimension);
  de.son = sn[0].block;
  de.son_ptr = sn[0];
  de.son_is_data = root_is_data;
  de.num_of_data = ((Node)sn[0])
    .get_num_of_data();
  nroot_ptr.enter(de);
  root = nroot;
  root_ptr = nroot_ptr;
```

```
        root_is_data = false;
      }
      j++;
    }
    num_of_data++;
}//end insert
```

Sourcecode 4.3 Proses insert R*tree

4.2.4 Implementasi Proses Insert Data Node

Langkah selanjutnya adalah mengimplementasikan proses insert ke data node.

Pada prosedur ini akan diperoleh sekumpulan MBR yang memungkinkan untuk mereinsert atau dilakukan split, pengecekan bila alokasi belum penuh hal ini saat pertamakali insert data dan else jika penuh, penuh karena terjadi reinsert dari rtdatanode untuk dilempar ke linkedlist rtree sebelumnya untuk menambah data agar berulang di while yang mengakibatkan rtree menginsert kembali di rtdatanode, karena node sudah penuh maka dilempar ke else yang mengakibatkan pemngaggilan split di rtdatanode. Implementasi dari proses inser data node dapat dilihat pada *sourcecode* 4.4.

```
public int insert(Data d, RTNode sn[])
{
   int i, last_cand;
   float MBR[], center[];
   SortMBR sm[]; //used for REINSERT
   Data nd, new_data[];

   if (get_num() == capacity)
   Constants.error("RTDataNode.insert:
```

```
    maximum capacity violation", true);

// insert data into the node
data[get_num()] = d;

num_entries ++;
dirty = true;

if (get_num() == (capacity - 1))
// overflow
{

  if (my_tree.re_level[0] == false)
  {
    //reinsert 30%  dari data
    //karena bukan yang pertama insert
    //hitung pusat kepusat
    MBR = get_MBR();
    center = new float[dimension];

    for (i = 0; i < dimension; i++){
      center[i] = (MBR[2*i] +
      MBR[2*i+1]) / (float)2.0;
    }
    new_data = new Data[capacity];
    sm = new SortMBR[num_entries];
    for (i = 0; i < num_entries; i++)
    {
      sm[i] = new SortMBR();
      sm[i].index = i;
```

```
        sm[i].dimension = dimension;
        sm[i].MBR = data[i].get_MBR();
        sm[i].center = center;
        sm[i].tuple = data[i].tuple;

      }
      // sorting pusat
      Constants.quickSort(sm, 0, sm.length - 1,
        Constants.SORT_CENTER_MBR);
      last_cand = (int) ((float)num_entries * 0.30);

      // kopikan 70% kandidat ke array baru
      for (i = 0; i < num_entries - last_cand; i++){
        new_data[i] = data[sm[i].index];
      }
      //ambil 30% yang terakhir,
      //kandidat untuk proses reinsertion list
      //ke Rtree.insert
      for ( ; i < num_entries; i++)
      {
        nd = new Data(dimension);
        nd = data[sm[i].index];
        my_tree.re_data_cands.add(nd);
      }

      data = new_data;
      my_tree.re_level[0] = true;
      num_entries -= last_cand;
      dirty = true;
      //kembali ke rtree dengan nilai 1
```

```
        //dan re_level 0 =true
        return Constants.REINSERT;
      }else{
        //lakukan split karena bukan
        //yang pertama insert
        sn[0] = new RTDataNode(my_tree);
        sn[0].level = level;
        split((RTDataNode)sn[0]);
      }
      //sebelum di kembalikan ke rtree
      //pengecekan split untuk true
      return Constants.SPLIT;
    }else{
      return Constants.NONE;
    }
}
```

Sourcecode 4.4 Proses insert DataNode

4.2.5 Implementasi Split DataNode

Langkah selanjutnya adalah mengimplementasikan proses split yang di gunakan rtdatanode dalam menentukan distribusi terbaik. sebelum di lakukan split rtdatanode menentukan optimasi dan minimasi pada split. Implementasi pada proses split ditunjukkan pada *sourcecode* 4.5

```
public void split(RTDataNode splitnode)
{
   int i, distribution[][], dist, n;
   float MBR_array[][];
   Data new_data1[], new_data2[];
   // distribusi split
```

```
    n = get_num();
    distribution = new int[1][];
    MBR_array = new float[n][2*dimension];

    for (i = 0; i < n; i++)
    MBR_array[i] = data[i].get_MBR();

    //distribusi untuk melakukan split
    dist = super.split(MBR_array, distribution);

    //untuk hasil split
    new_data1 = new Data[capacity];
    new_data2 = new Data[capacity];

    for (i = 0; i < dist; i++)
    new_data1[i] = data[distribution[0][i]];

    for (i = dist; i < n; i++)
    new_data2[i-dist] = data[distribution[0][i]];

    //isikan data ke split untuk proses split
    data = new_data1;
    splitnode.data = new_data2;

    //jumlah data hasil split
    num_entries = dist;
    splitnode.num_entries = n - dist;
}
```

Sourcecode 4.5 Prosedur SplitDataNode

4.2.6 Implementasi Insert DirNode

Prosedur Insert DirNode digunakan untuk memasukkan MBR pada node parent yang memer lukan chose subtee untuk MBR paren mana yang perlu pemasukan, apabila parent dari anak node pada node saudara memerlukan split karena maksimal daa parent pada data saudara maka memrlukan split yang menghitung optimasi dan minimasi untuk distribusi split terbaik pada struktur daa R*-tree. Implementasi dari Proses Insert DirNode ditunjukkan pada *sourcecode* 4.6

```
public int insert(Data d, RTNode sn[])
{
   int follow;
   RTNode succ = null;
   RTNode new_succ[] = new RTNode[1];
   DirEntry de;
   int ret;
   float MBR[],nMBR[];

   //pemilihan subtree
   //choose_subtree
   MBR = d.get_MBR();
   follow = choose_subtree(MBR);
   // mengambil anak dari subtree
   succ = entries[follow].get_son();
   // reinsert ke data anak
   ret = ((Node)succ).insert(d, new_succ);

   //jika terjadi split atau reinsert
   //ikuti data anak dan lakukan
```

```
//perubahan pada nilai pembatas masing-masing
if (ret != Constants.NONE)
{
   MBR = ((Node)succ).get_MBR();
   System.arraycopy(MBR, 0,
   entries[follow].bounces, 0, 2*dimension);
}

// succeeder di dalam tree
entries[follow].num_of_data =
   ((Node)succ).get_num_of_data();
//setelah terjadi split
if (ret == Constants.SPLIT)
{
   // cek isi tiap i anak
   if (get_num() == capacity)
   Constants.error("insert: kapasitas penuh", true);

   // buat entry untuk succeeder baru
   de = new DirEntry(dimension, son_is_data, my_tree);
   nMBR = ((Node)new_succ[0]).get_MBR();

   System.arraycopy(nMBR, 0, de.bounces,
      0, 2*dimension);
   de.son = new_succ[0].block;
   de.son_ptr = new_succ[0];
   de.son_is_data = son_is_data;
   de.num_of_data = ((Node)new_succ[0])
      .get_num_of_data();
```

```
      // insertkan ke direntry
      enter(de);

      //bila kapsitas penuh pada direntry
      //lakukan split direntry, terjadi oferflow
      //pada direktori node
      if (get_num() == (capacity - 1))
      {
         // inisialisasi untuk spit node saudara
         sn[0] = new RTDirNode(my_tree);
         ((RTDirNode)sn[0]).son_is_data =
               ((RTDirNode)this).son_is_data;
         sn[0].level = level;
         //lakukan split
         split((RTDirNode)sn[0]);
         ret = Constants.SPLIT;
      }else
      ret = Constants.NONE;
   }
   dirty = true;
   return ret;
}
```

Sourcecode 4.6 Proses insert DirEntry node

4.2.7 Implementasi Chose Subtree

Prosedur Chose Subtree digunakan untuk menentukan MBR mana yang perlu di ikuti dan dilakukan pemasukkan dengan menentukan area mana yang memerlukan pembesaran dan tumpang tindih, bila data yang di temukan pada node telah penuh maka node yang berisi MBR tersebut aka dilakukan split. Implementasi dari Proses Chose Subtree ditunjukkan pada *sourcecode* 4.7

```
public int choose_subtree(float MBR[])
{
   int i, j, n, follow, minindex=0,
   inside[], inside_count, over[];
   float bMBR[] = new float[2*dimension];
   float old_o, o, omin, a, amin, f, fmin;
   n = get_num();
   inside_count = 0;
   inside = new int[n];
   // tentukan inside[]
   for (i = 0; i < n; i++)
   {
      switch (entries[i].section(MBR))
      {
         case Constants.INSIDE:
            // MBR berada dalam entries[i] MBR
            inside[inside_count++] = i;
         break;
      }
   }

   if (inside_count == 1){
      //1. hanya satu MBR berada di dalamnya
      follow = inside[0];
   }else if (inside_count > 1){
      //2. terdapat banyak MBR di dlamnya
      //maka memilih untuk insert dengan
      //pembesaran area yang kecil
      fmin = Constants.MAXREAL;
      for (i = 0; i < inside_count; i++)
```

```
    {
       f = Constants.area(dimension,
          entries[inside[i]].bounces);
       if (f < fmin)
       {
          minindex = i;
          fmin = f;
       }
    }
    follow = inside[minindex];
 }else{
    //3. tidak ada direktori MBR di dalamya
    //maka memilih salah satu dengan tumpang
    //tindih terkecil lalu memilih pembesaran
    //area yang kecil untuk reinsert
    if (son_is_data)
    {
       omin = Constants.MAXREAL;
       fmin = Constants.MAXREAL;
       amin = Constants.MAXREAL;
       for (i = 0; i < n; i++)
       {
          //hitung dari tiap MBR ke
          //MBR dan masukan ke i
          Constants.enlarge(dimension, bMBR,
                MBR, entries[i].bounces);

          // hitung area dan pembesaran area
          a = Constants.area(dimension,
                entries[i].bounces);
```

```
            f = Constants.area(dimension, bMBR) - a;

            //hitung tumpang tindih sebelum
            //menghitung pembesaran area
            old_o = o = (float)0.0;
            for (j = 0; j < n; j++)
            {
               if (j != i)
               {
                  old_o += Constants.overlap(dimension,
                  entries[i].bounces,
                  entries[j].bounces);
                  o += Constants.overlap(dimension,
                  bMBR,entries[j].bounces);
               }
            }
            o -= old_o;

            // menentukan yang optimum
            if ((o < omin) ||(o == omin && f < fmin) ||
            (o == omin && f == fmin && a < amin))
            {
               minindex = i;
               omin = o;
               fmin = f;
               amin = a;
            }
         }
      }else{
        //bukan data anak
```

```
        fmin = Constants.MAXREAL;
        amin = Constants.MAXREAL;
        for (i = 0; i < n; i++)
        {
          //hitung dari tiap MBR ke
          //MBR dan masukan ke i
          Constants.enlarge(dimension, bMBR, MBR,
             entries[i].bounces);

          //hitung area dan pembesaran area
          a = Constants.area(dimension,
               entries[i].bounces);
          f = Constants.area(dimension, bMBR) - a;

          // menentukan yang paling optimum
          if ((f < fmin) || (f == fmin && a < amin))
          {
             minindex = i;
             fmin = f;
             amin = a;
          }
        }
      }
      // mengikuti data MBR dari hasil optimasi
      Constants.enlarge(dimension, bMBR, MBR,
          entries[minindex].bounces);
      System.arraycopy(bMBR, 0, entries[minindex]
          .bounces, 0, 2*dimension);
      follow = minindex;
      dirty = true;
```

```
    }
    return follow;
}
```

Sourcecode 4.7 Prosedur ChooseSubtree

4.2.8 Implementasi Split DirEntry

Prosedur Split DirEntry digunakan untuk reintegrasi yang memerlukan optimasi dan minimasi pada struktur parent yang terjadi perluapan karena node prent hasil dari choose subtree telah penuh. Implementasi dari Proses Peluang Split DirEntry ditunjukkan pada *sourcecode* 4.8

```
public void split(RTDirNode brother)
{
    int i, dist, n;
    int distribution[][];
    float MBR_array[][];
    DirEntry new_entries1[], new_entries2[];

    n = get_num();
    distribution = new int[1][];

    //array MBR untuk kumpulan MBR dari seluruh entry
    MBR_array = new float[n][dimension*2];
    for (i = 0; i < n; i++)
    MBR_array[i] = entries[i].bounces;

    //memanggil super class untuk hasil distribusi split
    dist = super.split(MBR_array, distribution);
    //inisialisasi entry baru untuk hasil split
    new_entries1 = new DirEntry[capacity];
```

```
   new_entries2 = new DirEntry[capacity];
   //isikan entry baru dengan
   //MBR dari hasil split
   for (i = 0; i < dist; i++)
   {
      new_entries1[i] = entries[distribution[0][i]];
   }

   for (i = dist; i < n; i++)
   {
     new_entries2[i-dist] = entries[distribution[0][i]];
   }
   // merubah entri saudara dengan hasil split
   entries = new_entries1;
   brother.entries = new_entries2;

   // sisa jumlah entri dari hasil split
   num_entries = dist;
   brother.num_entries = n - dist;
}
```

Sourcecode 4.8 Prosedur Split DirEntry

4.2.9 Implementasi Enter Direntry

Prosedur Enter Direntry digunakan untuk pengecekan apakah kapasitas node pada tree telah penuh bila tidak berdasarkan nomer entry data di masukkan dan pengecekan di kembalikan dengan nilai entri selanjutnya. Implementasi dari Proses Enter Direntry ditunjukkan pada *sourcecode* 4.9

```
public void enter(DirEntry de)
{
```

```
   if (get_num() > (capacity-1))
      Constants.error("entry data penuh", true);
   entries[num_entries] = de;
   num_entries++;
}
```

Sourcecode 4.9 Prosedur Enter

4.2.10 Implementasi getMBR

Pada prosedur getMBR dipanggil prosedur ini mengembalikan nilai MBR yang menjadi pembatas atau mencakupi dari sekumpulan MBR. Implementasi dari Prosedur getMBR ditunjukkan pada *sourcecode* 4.10

```
public float[] get_MBR()
{
   int i, j, n;
   float MBR[];


   MBR = new float[2*dimension];
   for (i = 0; i < 2*dimension; i ++ )
   MBR[i] = entries[0].bounces[i];


   n = get_num();
   for (j = 1; j < n; j++)
   {
      for (i = 0; i < 2*dimension; i += 2)
      {
        MBR[i]   = Constants.min(MBR[i],
         entries[j].bounces[i]);
        MBR[i+1] = Constants.max(MBR[i+1],
         entries[j].bounces[i+1]);
      }
```

```
    }
    return MBR;
}
```

Sourcecode 4.10 Prosedur getMBR

4.2.11 Implementasi SPLIT

Pada proses split yang ada pada data node maupun directory node yang memerlukan split setelah di tentukan MBR yang akan di split maka prosedur split yang ditunjukkan pada *sourcecode* 4.11 dipanggil untuk menetukan distribusi terbaik dengan menghitung optimasi dan minimasi yang akan dilakukan split.

```
public int split(float MBR[][], int distribution[][])
{
    boolean lu = false;
    int i, j, k, l, s, n, m1;
    int dist = 0;
    int split_axis = 0;
    SortMBR sml[], smu[];
    float minmarg;
    float marg, minover, mindead, dead,
       over, rxMBR[], ryMBR[];
       // jumlah node untuk split
    n = get_num();

    // isikan node setidaknya 40%
    m1 = (int) ((float)n * 0.40);
    dist = m1;

    // urutkan sml dengan axis terendah
    // urutkan smu dengan axis tertinggi
```

```
    sml = new SortMBR[n];
    smu = new SortMBR[n];
    rxMBR = new float[2*dimension];
    ryMBR = new float[2*dimension];

    // memilih split axis untuk minimasi margin
    minmarg = Constants.MAXREAL;
    for (i = 0; i < dimension; i++)
    // untuk setiap axis
    {
       ...
       marg = (float)0.0;
       // untuk seluruh kemungkinan di R1,R2 pada sml
       for (k = 0; k < n - 2*m1 + 1; k++)
       {
          ...
          // R2 = yang terakhir adalah n-m1-k MBRs
          for ( ; l < n; l++)
          {
    for (i = 0; i < n; i++)
    {
       if (lu)
          distribution[0][i] = sml[i].index;
       else
          distribution[0][i] = smu[i].index;
    }
    // mengembalikan index berdasarkan split terbaik
    return dist;
}
```

Sourcecode 4.11 Prosedur Split

4.2.12 Implementasi Prosedur enlarge

Prosedur enlarge digunakan untuk menghitung pembesaran area pada MBR. Implementasi dari Prosedur enlarge ditunjukkan pada *sourcecode* 4.12

```
public static void enlarge(int dimension, float MBR[],
float r1[], float r2[])
{
   int i;
   for (i = 0; i < 2*dimension; i += 2)
   {
      MBR[i]   = min(r1[i],   r2[i]);
      MBR[i+1] = max(r1[i+1], r2[i+1]);
   }
}
```

Sourcecode 4.12 Proses Enlarge

4.2.13 Implementasi Prosdur Overlape

Prosedur Overlape digunakan untuk menghitung luas irisan dari tumpang tindih yang digunakan untuk menetukan inside atau contain pada MBR. Implementasi dari Proses Overlape ditunjukkan pada *sourcecode* 4.13

```
public static float overlap(int dimension, float r1[],
float r2[])
{
   float sum;
   int r1pos, r2pos, r1last;
   float r1_lb, r1_ub, r2_lb, r2_ub;
   sum = (float)1.0;
   r1pos = 0; r2pos = 0;
   r1last = 2 * dimension;
   while (r1pos < r1last)
   {
```

```
    r1_lb = r1[r1pos++];
    r1_ub = r1[r1pos++];
    r2_lb = r2[r2pos++];
    r2_ub = r2[r2pos++];
    if (inside(r1_ub, r2_lb, r2_ub))
    {
     if (inside(r1_lb, r2_lb, r2_ub))
      sum *= (r1_ub - r1_lb);
     else
      sum *= (r1_ub - r2_lb);
    }else{
     if (inside(r1_lb, r2_lb, r2_ub)){
      sum *= (r2_ub - r1_lb);
     }else{
      if (inside(r2_lb, r1_lb, r1_ub) &&
         inside(r2_ub, r1_lb, r1_ub))
        sum *= (r2_ub - r2_lb);
      else
        sum =(float)0.0;
     }
    }
   }
   return sum;
}
```

Sourcecode 4.13 Prosedur *Overlape*

4.2.14 Implementasi inside

Prosedur inside pada *sourcecode* 4.14 digunakan oleh prosedur overlape untuk menentukan inside. Berdasarkan xl1,xl2 engan y pada susunan MBR yang di hitung luas irisannya.

```
public static boolean inside(float p, float lb, float
ub)
{
   return (p >= lb && p <= ub);
}
```

Sourcecode 4.14 Prosedur Inside

4.2.15 Implementasi Margin

Prosedur margin digunakan untuk menghitung margin yang ada pada MBR. Implementasi dari prosedur margin ditunjukkan pada *sourcecode* 4.15

```
public static float margin(int dimension, float MBR[])
{
   int i;
   int ml, mu, m_last;
   float sum;
   sum = (float)0.0;
   m_last = 2*dimension;
   ml = 0;
   mu = ml + 1;

   while (mu < m_last)
   {
      sum += MBR[mu] - MBR[ml];
      ml += 2;
      mu += 2;
   }
   return sum;
}
```

Sourcecode 4.15 Prosedur Margin

4.2.16 Implementasi Area

Prosedur area digunakan untuk menghitung luas area dari MBR. Implementasi dari Proses Area ditunjukkan pada *sourcecode* 4.16

```
public static float area(int dimension, float MBR[])
{
   int i;
   float sum;
   sum = (float)1.0;

   for (i = 0; i < dimension; i++){
      float a=sum;
      sum *= MBR[2*i+1] - MBR[2*i];
   }
   return sum;
}
```

Sourcecode 4.16 Prosedur Area

4.2.17 Implementasi Prosedur Range Query

Prosedur Range Query digunakan untuk menetukan node mana yang menyimpan MBR terjadi tumpang tindih atau inside dengan susunan coordinat xlyl xu yu MBR pencarian, prosedur ini di gunakan untuk criteria pencarian area dengan range query. Implementasi dari range query ditunjukkan pada *sourcecode* 4.17

```
public void rangeQuery(float MBR[], LinkedList res)
{
   int i, n;
   int s;
   RTNode succ;
```

```
    my_tree.page_access++;
    n = get_num();

    for (i = 0; i < n; i++)
    {
       s = entries[i].section(MBR);
       if (s == Constants.INSIDE ||
          s == Constants.OVERLAP)
       {
          succ = entries[i].get_son();
          ((Node)succ).rangeQuery(MBR,res);
       }
    }
}
```

Sourcecode 4.17 Prosedur Range Query

4.2.18 Implementasi Prosedur Random Input

Prosedur Random input digunakan untuk mengambil nilai random index pada sebuah MBR yang akan di lakukan pencarian berdasarkan index MBR yang terpilih. Implementasi dari Prosedur Random input ditunjukkan pada *sourcecode* 4.18

```
private void randomInput(){
 if(t.f.boxQuery.size()>0==false){
  t.f.boxQuery=t.lbox;
 }

 Random randomNumbers = new Random();
 if(t.f.boxQuery.size()>0){
  int face = randomNumbers.nextInt(t.f.boxQuery.size());
```

```
    Data d = ((Data)t.f.boxQuery.get(face));
    jTextField1.setText(String.valueOf(d.data[0]));
    jTextField2.setText(String.valueOf(d.data[1]));
    jTextField3.setText(String.valueOf(d.data[2]));
    jTextField4.setText(String.valueOf(d.data[3]));
  }
}
```

Sourcecode 4.18 Prosedur Random Input

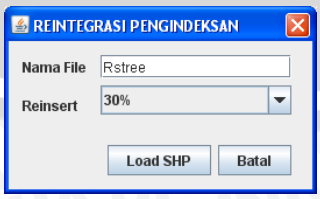
4.3 Implementasi Antarmuka

4.3.1 Tampilan Utama

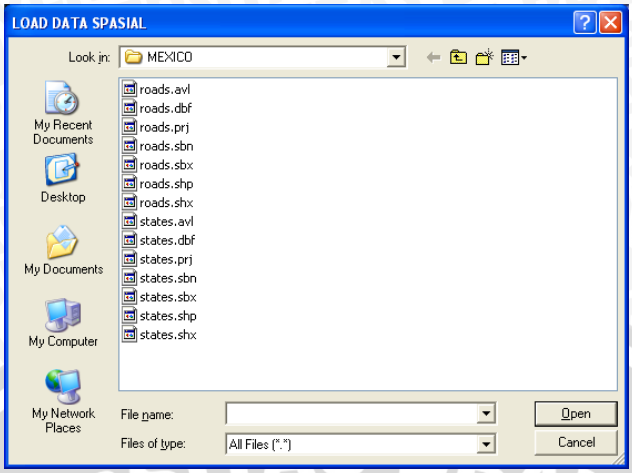
Tampilan load file SHP yang menyimpan geometri koordinat peta yang akan dilakukan pengindeksan pada struktur data R*-tree dengan prosentase reinsert untuk reintegrasi struktur data R*-tree seperti di tunjukkan pada **Gambar 4.2 dan** untuk mengatur besar prosentase reinsert di tunjukkan pada **Gambar 41**

Hasil load file SHP ditampilkan dalam bentuk peta yang di gambarkan pada halaman utama **Gambar 4.3**.

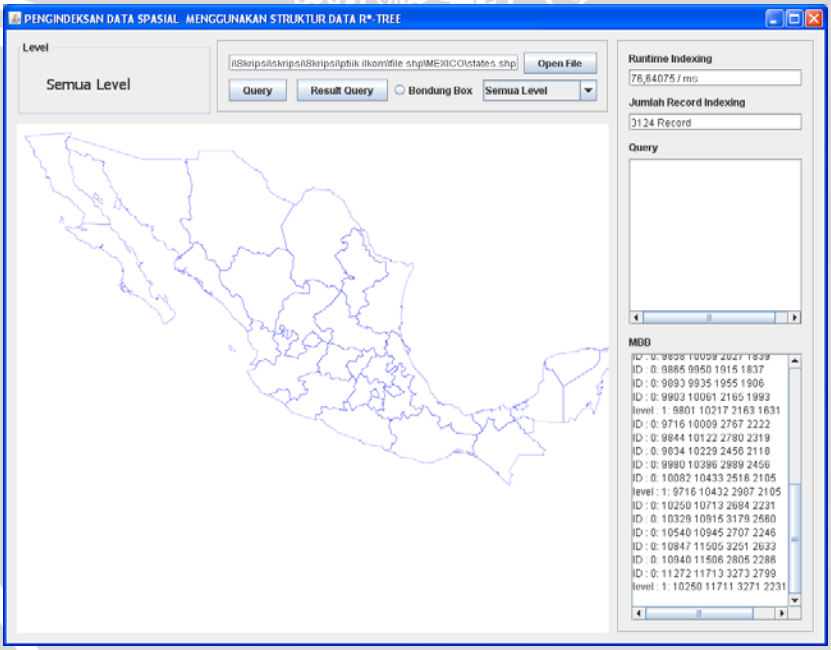
Halaman utama pada **Gambar 4.3** terdapat beberapa pilihan dan informasi seperti hasil pengindeksan dan query serta runtime dam MBR yang di hasilkan, hasil dari pengindeksan data *vector* bertipe polygon di gambarkan pada **Gambar 4.3** dan gambar hasil pengindeksan data *vector* bertipe polyline seperti di tunjukkan pada **Gambar 4.4**



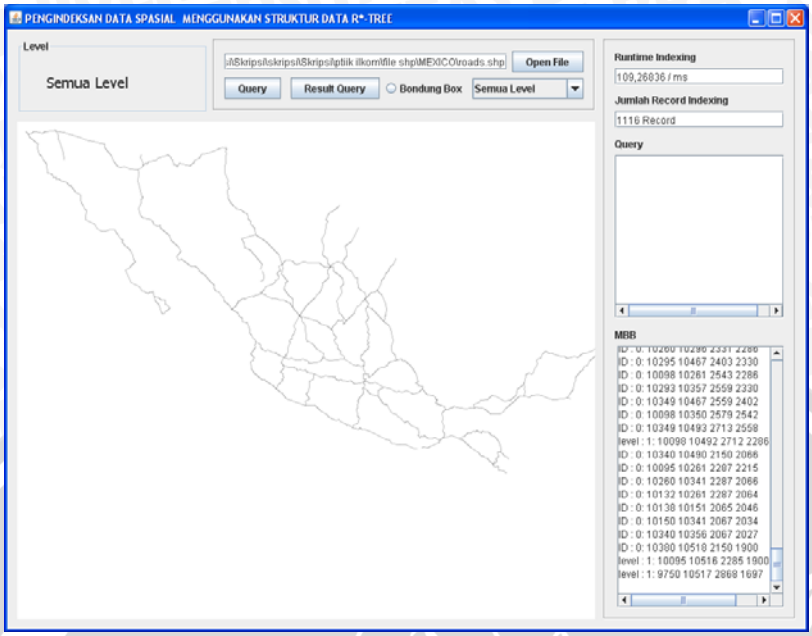
Gambar 4.1 Prosentase reintegrasi pengindeksan



Gambar 4.2 Load file SHP



Gambar 4.3 Peta tipe data polygon



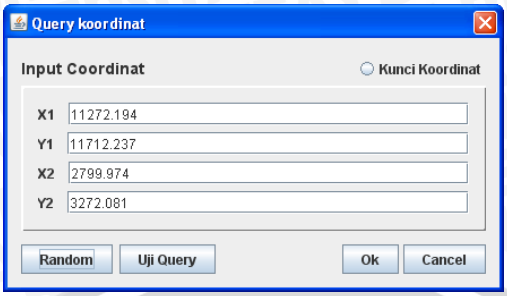
Gambar 4.4 Peta tipe data polyline

4.3.2 Dialog Range Query

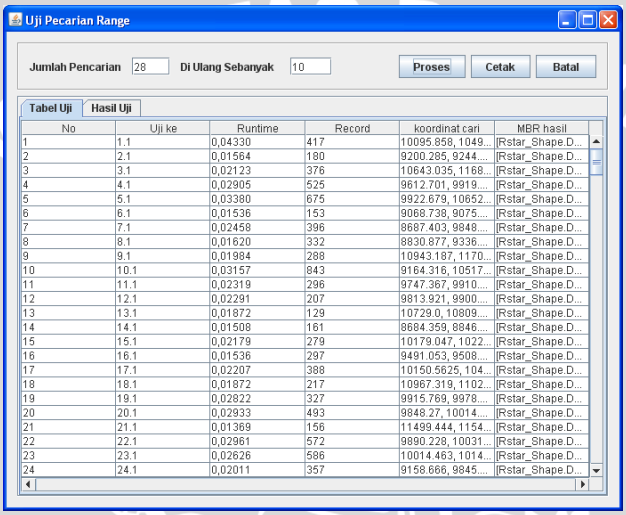
Dialog Query yang ditunjukkan pada **gambar 4.5** merupakan bagian program yang dijalankan untuk memasukkan koordinat pencarian di dalam struktur data R*-tree.

Pada bagian ini terdapat random untuk uji pencarian dengan koordinat masukan yang dirandom berdasarkan MBR yang terbentuk dalam struktur data R*-tree,

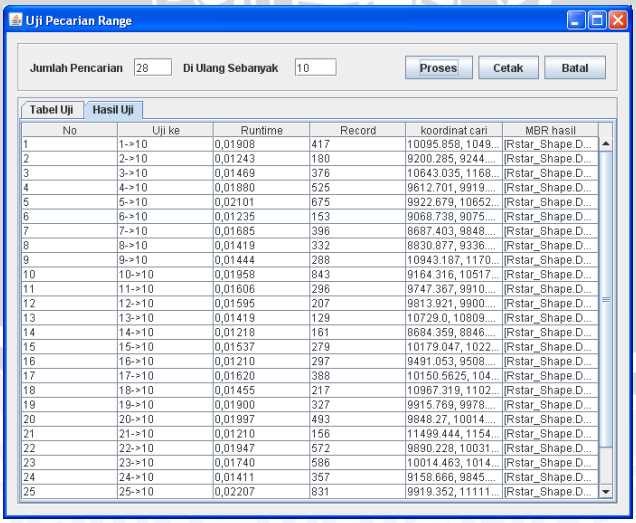
Pada dialog query terdapat uji query dimana button ini di gunakan untuk melakukan banyak uji query seperti ditunjukan pada **Gambar 4.6** dengan 28 perulangan dan random query yang setiap query di ulang sebanyak 10 kali, hasil rata-rata dari tiap perulangan ditunjukkan pada **Gambar 4.7**



Gambar 4.5 Dialog query



Gambar 4.6 Dialog perulangan query



Gambar 4.7 Dialog rata-rata hasil perulangan query

4.4 Implementasi Uji Coba

Pada subbab ini akan dilakukan pembahasan uji coba mengenai pengujian serta ringkasan hasil dari pengindeksan dan query pada struktur data R*-tree untuk tipe data polygon dan tipe data polyline yang telah dijelaskan pada bab 3 sebelumnya.

4.4.1 Uji Coba Pengindeksan Dan Query Tipe Data Polyline

File uji coba yang digunakan adalah file shp yang di peroleh dari data ESRI yang menyimpan koordinat peta jalan yang ada pada negara Mexico dengan nama file roads.shp dengan ukuran 23,2 kb.

Pada file roads.shp terdapat 1116 Record, untuk pembacaanya file roads.shp memerlukan file index dan file tabel yaitu roads.shx dengan ukuran file 1,16 kb dan file roads.dbf dengan ukuran 11,2 kb.

Pada uji coba pengindeksan dan query masing-masing di lakukan dengan prosentase reinsert 30%,50% dan 70% pada jumlah MBR yang ada pada struktur data R*-tree.

Besar prosentase 30% tersebut untuk dilakukan reinsert dan sisanya 70% untuk dilakukan split yang memerlukan optimasi dan minimasi MBR pada proses reintegrasi yang ada pada struktur data R*-tree, begitu juga untuk prosentase reinsert sebesar 50% dan 70%.

Masing-masing uji coba pengindeksan diulang sebanyak 10 kali kemudian di hitung rata-rata hasil *runtime*-nya. Hasil dari masing-masing pengindeksan yang berdasarkan prosentase *reinsert* dilakukan *query* dengan *range* pencarian koordinat yang dicakupi oleh MBR seperti ditunjukkan pada tabel 4.2.

Hasil dari ujicoba pengindeksan pada data bertipe polyline yang masing-masing di ulang sebanyak 10 kali dan dihitung rata-rata waktunya ditunjukkan pada tabel 4.1 berikut:

Tabel 4.1 Runtime pengindeksan tipe data polyline

REINSERT	RUNTIME(ms)
30%	111,442685
50%	111,194886
70%	110,939545

Tabel 4.2 Koordinat uji coba query tipe data polyline

Uji ke	X1	X2	Y1	Y2
1	10.057.846	10.076.135	2.115.589	2.164.631
2	8.743.292	8.846.775	18.526.744	20.245.477
3	8.684.359	8.743.292	20.245.477	2.115.689
4	11.544.101	11.548.423	32.616.191	32.663.682
5	9.909.986	9.982.539	2.373.166	25.182.058
6	9.297.396	9.336.692	17.995.392	18.031.494
7	9357.56	9.491.053	17.935.885	18.092.994
8	115.301.875	11.544.101	32.152.256	32.616.191
9	9.164.316	9.244.663	14.571.367	15.127.095
10	8.846.775	9.073.566	18.450.398	18.700.626
11	9.800.937	9848.27	22.552.048	24.215.222
12	10.355.304	10.380.115	19.008.768	19.380.632
13	89.563.545	9.073.927	19.344.089	2.096.062
14	10.057.846	10.076.135	2.115.589	2.164.631
15	8.743.292	8.846.775	18.526.744	20.245.477
16	8.684.359	8.743.292	20.245.477	2.115.689
17	10.492.945	10.510.816	27.127.773	27.616.223
18	9.073.927	9.339.549	18.000.159	19.344.089
19	9.813.921	9.900.114	22.552.048	23.316.829
20	10.380.115	10.517.512	19.008.768	21.493.213
21	9.817.978	9.845.763	18.597.588	1.908.523
22	9.940.689	9.978.825	16.973.055	1.857.529
23	9.897.017	9912.97	23.316.829	2.373.166
24	10228.01	10.380.115	18.144.607	19.008.768
25	101.505.625	10.340.112	20.341.769	20.661.514
26	10.179.047	10.345.088	19.799.131	20.275.204
27	10.179.047	10.229.085	18.144.607	19.799.131

Uji ke	X1	X2	Y1	Y2
28	10.446.971	10.564.235	2.402.434	27.127.773
29	9.978.825	10228.01	16.973.055	18.151.449
30	10.132.846	10.260.679	2.064.568	22.860.923
31	10.095.858	10.260.679	22.150.076	22.860.923
32	10.031.549	10.098.526	25.423.787	25.695.952

Setiap pengujian pencarian di ulang sebanyak 10 kali dan dihitung rata-rata waktu yang dihasilkan dengan satuan waktu mili detik (ms), kemudian berdasarkan jumlah record yang dihasilkan ditentukan antara 10,20,30,.. 80 dari setiap record hasil query dan dihitung rata-rata runtimenya.

Hasil dari uji coba pengindeksan yang telah dilakukan pada reinsert 30% yang diulang 10 kali didapat rata–rata runtime 111,442685 ms dengan total record 1116 Record dan hasil query dari pengindeksan berdasarkan prosentase reinsert 30% ditunjukan pada tabel 4.3 berikut:

Tabel 4.3 Runtime query tipe data polyline (Reinsert 30%)

NO	RUNTIME(ms)	RECORD
1	0,003406	10
2	0,003182	20
3	0,003429	30
4	0,003854	40
5	0,00419	50
6	0,004378	60
7	0,004022	70
8	0,004661	80

Hasil dari uji coba selanjutnya adalah pengindeksan dengan menggunakan prosentase reinsert 50% yang diulang 10 kali didapat rata–rata runtime 111,194886 ms dengan total record 1116 Record dengan hasil query ditunjukan pada tabel 4.4 berikut:

Tabel 4.4 Runtime query tipe data polyline (Reinsert 50%)

NO	RUNTIME(ms)	RECORD
1	0,003378	10
2	0,003434	20
3	0,003434	30
4	0,003915	40
5	0,004148	50
6	0,004322	60
7	0,004414	70
8	0,004521	80

Hasil dari uji coba yang terakhir yaitu pengindeksan dengan menggunakan prosentase reinsert 70% yang diulang 10 kali didapat rata–rata runtime 110,939545 ms dengan total record 1116 record dan hasil rata-rata query di tunjukan pada tabel 4.5 berikut:

Tabel 4.5 Runtime query tipe data polyline (Reinsert 70%)

NO	RUNTIME(ms)	RECORD
1	0,002902	10
2	0,003322	20
3	0,003588	30
4	0,003789	40
5	0,004246	50
6	0,004518	60
7	0,00454	70
8	0,004321	80

4.4.2 Uji Coba Pengindeksan Dan Query Tipe Data Polygon

File uji coba yang digunakan adalah file shp yang di peroleh dari data ESRI yang menyimpan koordinat peta Mexico dengan nama file states.shp dengan ukuran 51 kb, pada file states.shp terdapat 3124 Record.

Untuk pembacaanya file states.shp memerlukan file index dan file tabel yaitu states.shx dengan ukuran file 1kb dan file states.dbf dengan ukuran 4kb.

Pada uji coba pengindeksan dan query masing-masing di lakukan dengan prosentase reinsert 30%,50% dan 70% pada jumlah MBR yang ada pada struktur data R*-tree.

Prosentase 30% yang dilakukan untuk reinsert, hal ini sisanya untuk split yang digunakan untuk optimasi dan minimasi pada MBR, begitu juga untuk prosentase reinsert 50% dan 70%, masing-masing uji coba pengindeksan diulang sebanyak 10 kali kemudian di hitung rata-rata hasil runtimenya.

Hasil dari ujicoba pengindeksan pada data bertipe polygon yang masing-masing di ulang sebanyak 10 kali dan dihitung rata-rata waktunya ditunjukkan pada tabel 4.6 berikut:

Tabel 4.6 Runtime pengindeksan tipe data polygon

REINSERT	RUNTIME(ms)
30%	86,19436
50%	86,13609
70%	86,07108

Hasil dari masing-masing pengindeksan yang berdasarkan prosentase reinsert dilakukan query dengan range pencarian koordinat yang di cakupi MBR seperti ditunjukkan pada tabel 4.7 berikut:

Tabel 4.7 Koordinat uji coba query tipe data polygon

Uji Ke	X1	X2	Y1	Y2
1	9.893.918	9.934.056	1.906.139	1.954.083
2	8673.5	8.943.044	17.818.885	21.623.333
3	8.753.915	9.043.641	1.965.194	21.606.667
4	10347.96	10459.46	18.688.069	1.951.833
5	9.865.641	9.949.653	18.370.791	19.142.441
6	8.941.833	9.247.829	17.819.163	20.845.298
7	9.761.334	9.871.278	1.909.555	1.971.389
8	11.272.194	11.712.237	2.799.974	3.272.081
9	10005.7	10.374.549	17.921.895	20.403.049
10	10184.62	10285.17	2.165.555	22.458.892

Uji Ke	X1	X2	Y1	Y2
11	9.903.461	10.060.061	1.993.804	21.643.892
12	9.037.889	9.412.307	14.550.547	1.799.139
13	9.098.242	9.413.782	17.241.108	18.651.676
14	10394.49	10.664.751	20.693.254	23.067.458
15	9844.39	10.121.859	2.319.444	2.779.417
16	10.329.015	10914.64	25.606.091	31.786.804
17	10.940.417	11.505.973	22.863.887	28.049.443
18	9.802.945	9982.91	19.585.399	2.139.889
19	9.716.862	10008.92	22.221.309	27.665.857
20	9.386.806	9.855.469	1.564.361	18.681.111
21	9.801.334	10.218.086	16.319.343	18.860.189
22	9.972.473	10210.88	1.990.778	2.186.139
23	9.672.639	9.905.049	17.892.219	2080.5
24	10847.13	11.504.149	26.332.776	32.504.448
25	9.858.139	10.058.641	18.390.829	2027.0
26	9.980.781	10395.47	2456.0	29.880.115
27	10250.5	10.712.471	2.231.921	2.683.861
28	10082.39	10432.96	21.054.438	2.515.555
29	10152.49	10.567.696	1895.5	22.764.722
30	10540.21	10.944.334	22.468.362	2.706.139
31	9358.94	9.864.223	1.715.028	2.247.139
32	9.834.558	10.228.061	2118.5	2456.0

Setiap pengujian pencarian di ulang sebanyak 10 kali dan dihitung rata-rata waktu yang dihasilkan dengan satuan waktu mili detik (ms), kemudian berdasarkan jumlah record yang dihasilkan ditentukan antara 200,300,400,.. 1000 dari setiap record hasil query dan dihitung rata-rata runtimenya

Hasil dari uji coba pengindeksan yang telah dilakukan pada prosentase reinsert 30% yang diulang 10 kali didapat rata–rata runtime 86,19436 ms dengan total record 3124 Record dan hasil query dari pengindeksan berdasarkan prosentase reinsert 30% ditunjukan pada tabel 4.8 berikut:

Tabel 4.8 Runtime query tipe data polygon (Reinsert 30%)

NO	RUNTIME(ms)	RECORD
1	0,01366	200
2	0,013465	300
3	0,01296	400

NO	RUNTIME(ms)	RECORD
4	0,015033	500
5	0,014815	600
6	0,015887	700
7	0,015962	800
8	0,01811	900
9	0,018123	1000

Hasil dari uji coba selanjutnya adalah pengindeksan dengan menggunakan prosentase reinsert 50% yang diulang 10 kali didapat rata–rata runtime 86,13609 ms dengan total record 3124 record dengan hasil query ditunjukan pada tabel 4.9 berikut:

Tabel 4.9 Runtime query tipe data polygon (Reinsert 50%)

NO	RUNTIME(ms)	RECORD
1	0,013457	200
2	0,01369	300
3	0,01215	400
4	0,013743	500
5	0,015287	600
6	0,015805	700
7	0,015482	800
8	0,01829	900
9	0,017733	1000

Hasil dari uji coba yang terakhir yaitu pengindeksan dengan menggunkan prosentase reinsert 70% yang diulang 10 kali didapat rata –rata runtime 86,07108 ms dengan total record 3124 Record dan hasil query di tunjukan pada tabel 4.10 berikut:

Tabel 4.10 Runtime query tipedata polygon (Reinsert 70%)

NO	RUNTIME(MS)	RECORD
1	0,01354	200
2	0,01362	300
3	0,0143	400
4	0,013747	500
5	0,015217	600
6	0,016382	700
7	0,015772	800

NO	RUNTIME(MS)	RECORD
8	0,018363	900
9	0,017767	1000

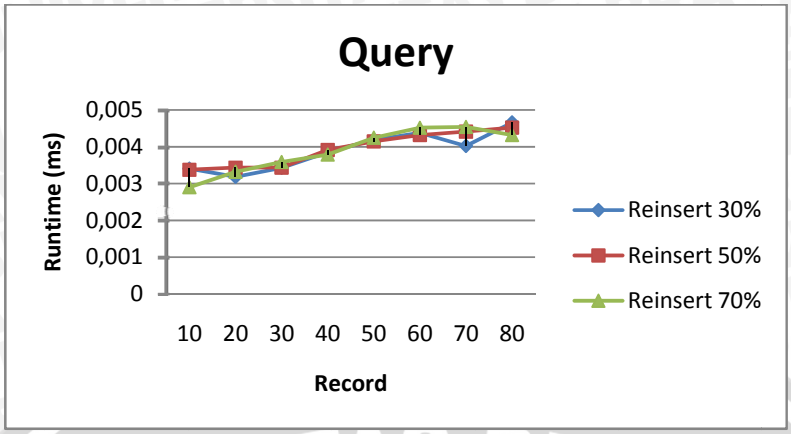
4.5 Analisa Hasil

a. Hasil Uji Coba Tipe Data Polyline

Dari ketiga hasil uji coba yang telah dilakukan pada pencarian dengan pengindeksan yang menggunakan prosentase reinsert 30%.50%,70% untuk selanjutnya dilakukan perbandingan berdasarkan waktu dan jumlah record yang di peroleh dari hasil query yang ditunjukkan pada tabel 4.11 dan di presentasikan pada sebuah grafik yang di tunjukkan pada **Gambar 4.8**

Tabel 4.11 Uji perbandingan query tipe data polyline

No	Record	Runtime query		
		Reinsert 30%	Reinsert 50%	Reinsert 70%
1	10	0,003406	0,003378	0,002902
2	20	0,003182	0,003434	0,003322
3	30	0,003429	0,003434	0,003588
4	40	0,003854	0,003915	0,003789
5	50	0,00419	0,004148	0,004246
6	60	0,004378	0,004322	0,004518
7	70	0,004022	0,004414	0,00454
8	80	0,004661	0,004521	0,004321
Runtime indeks		111,442685	111,194886	110,939545



Gambar 4.8 Grafik perbandingan ujicoba query tipe data polyline

Dari tabel 4.6 dan **Gambar 4.8** dapat dianalisa sebagai berikut:

1. Rata-rata waktu dalam query antara prosentase reinsert 30%, 50% dan 70% memiiki rata-rata waku query yang hampir sama, namun waktu pengindeksan terkecil yaitu pengindeksan dengan prosentase reinsert 70%.
2. Waktu pengindeksan terbesar yaitu waktu yang diperlukan pada reinsert dengan prosentase sebesar 30%.
3. Waktu pngindeksan pada prosentase reinsert 50% lebih kecil dari waktu yang diperlukan oleh prosentase reinsert 30% dan lebih besar dari waktu yang diperlukan oleh prosentase reinsert 70%.
4. Setiap bertambahnya jumlah record begitu pula bertambahnya waktu yang dibutuhkan selama pencarian dan pengindeksan.

Dari keempat analisa tersebut dapat ditarik kesimpulan, yaitu kecepatan dalam pencarian suatu area pada data spasial meggunakan struktur data R*-tree akan evektif dengan pengindeksan yang menggunakan prosentase reinsert 70%, karena memiliki waktu pengindeksan yang terkecil diantara waktu yang

diperlukan oleh reinsert 30% dan 50% dari masing-masing ketiga prosentase reinsert berdasarkan hasil rata-rata waktu yang hampir sama,

Waktu pengindeksan pada prosentase reinsert 30% lebih besar dari masing-masing ketiga prosentase reinsert, hal ini di sebabkan karena 30% yang dilakukan reinsert pada struktur data R*-tree untuk dimasukkan kembali dan sisanya 70% untuk dilakukan split.

Selama proses Split untuk reintegrasi yang digunakan oleh struktur data R*-tree akan memakan waktu yang diperlukan untuk optimasi dan minimasi

Berdasarkan waktu yang diperlukan oleh optimasi dan minimasi bilamana bertambahnya suatu jumlah data area maka akan bertambah pula waktu yang di butuhkan dalam proses pengindeksan begitu pula jumlah record dan jumlah susunan MBR didalam sruktur tree,

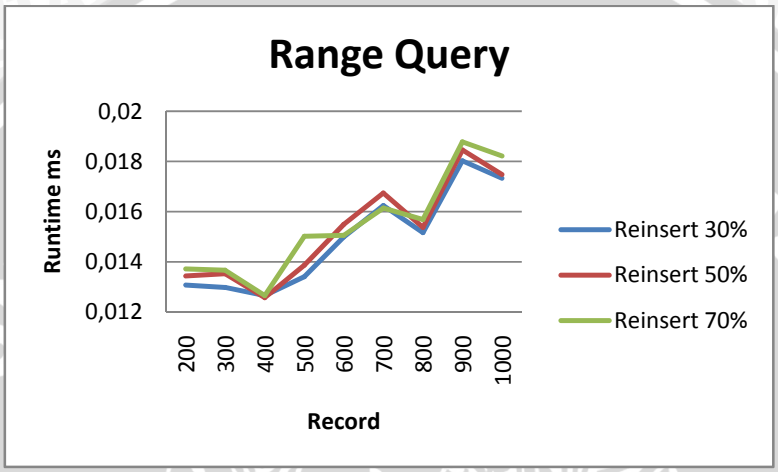
b. Hasil Uji Coba Type Data Polygon

Dari ketiga hasil uji coba yang telah dilakukan pada pencarian dengan pengindeksan yang menggunakan prosentase reinsert 30%.50%,70% untuk selanjutnya dilakukan perbandingan berdasarkan waktu dan jumlah record yang di peroleh dari hasil query yang ditunjukkan pada tabel 4.12 dan di presentasikan pada sebuah grafik yang di tunjukkan pada **Gambar 4.9**

Tabel 4.12 Uji perbandingan query tipe data polygon

No	Record	Runtime query		
		Reinsert 30%	Reinsert 50%	Reinsert 70%
1	200	0,013073	0,013437	0,013717
2	300	0,012977	0,013522	0,013661
3	400	0,012654	0,012571	0,012655
4	500	0,013401	0,013866	0,015021
5	600	0,014988	0,015495	0,015057

No	Record	Runtime query		
		Reinsert 30%	Reinsert 50%	Reinsert 70%
6	700	0,016245	0,016738	0,016138
7	800	0,015163	0,015371	0,015683
8	900	0,018027	0,018456	0,018782
9	1000	0,017329	0,017478	0,018214
Runtime indeks		86,19436	86,13609	86,07108



Gambar 4.9 Grafik perbandingan ujicoba query tipe data polygon

Dari tabel 4.12 dan **Gambar 4.9** dapat dianalisa sebagai berikut:

1. Waktu terkecil dalam query yaitu pengindeksan dengan prosentase reinsert 30%, namun memiliki waktu indeks terbesar.
2. Rata-rata waktu terbesar query yaitu pengindeksan dengan prosentase reinsert 70%. namun memiliki waktu indeks yang lebih kecil dari prosentase reinsert 30%.
3. Pada prosentase reinsert 50% waktu indeks lebih kecil dari prosentase reinsert 30% dan lebih besar dari prosentase reinsert 70% namun memiliki rata-rata waktu query yang lebih tinggi dari prosentase reinsert 30% dan lebih kecil dari rata-rata waktu yang di butuhkan oleh prosentase reinsert 70%.

4. Setiap bertambahnya jumlah record begitu pula bertambahnya waktu yang dibutuhkan selama pencarian.

Dari keempat analisa tersebut dapat ditarik kesimpulan, yaitu kecepatan dalam pencarian suatu area pada data spasial meggunakan strukturdata R*-tree untuk data bertipe polygon akan lebih evektif dengan pengindeksan yang menggunakan prosentase reinsert yang kecil, namun membutukan rata-rata waktu pengindeksan yang cukup lama.

Pada prosentase reinsert 30% hal ini disebabkan karena 30% dimasukkan kembali untuk reinsert dan sisanya 70% dilakukan split untuk optimasi dan minimasi yang digunakan struktur data R*-tree dalam membangun susunan tree.

Bertambahnya suatu jumlah data area yang diindekskan pada struktur data R*-tree maka bertambah pula jumlah MBR begitu pula struktur tree yang terbentuk di dalamnya, maka akan bertambah pula waktu yang di butuhkan dalam pencarian maupun dalam pengindeksan.

Dengan demikian prosentase reinsert untuk proses reintegrasi yang memerlukan optimasi dan minimasi pada MBR di dalam struktur data R*-tree sangat mempengaruhi besar waktu yang diperlukan pada suatu pencarian dan pengindeksan.

BAB V
KESIMPULAN DAN SARAN

Bagian ini berisi kesimpulan dari seluruh pengerjaan skripsi yang telah dilakukan dan juga saran untuk pengembangan lebih lanjut.

5.1 Kesimpulan

Dari hasil uji coba yang telah dilakukan, maka dapat diambil kesimpulan antara lain :

1. Implementasi struktur data R*-tree untuk pengindeksan data spasial dapat di implementasikan dengan membentuk *tuple* dari geometri menjadi MBR yang dimasukkan dengan algoritma *InsertData*, *Insert* dan *ChooseSubtre.* Bila hasil pemasukan *node* MBR penuh, maka dilakukan *reintegrasi* dengan algoritma *OverflowTreatment*, *Reinsert*, *Split*, *ChooseSplitAxis* dan *ChooseSplitlndex.* Hasil akhir pemasukan MBR pada struktur *tree* adalah terbentuknya hirarki obyek MBR yang tersusun di dalamnya dengan berbasis kedekatan yang mencakupi area geometri.
2. Waktu yang dihasilkan dari rata-rata *runtime query* terkecil pada data bertipe *polygon* dengan data vektor negara Mexico diperoleh dari prosentase *reinsert* 30% dengan *runtime* indeks terbesar dari ketiga prosentase *reinsert* sebesar 86,19436 ms, untuk *runtime* indeks terkecil 86,07108 ms berada pada prosentase *reinsert* 70% dengan *runtime query* terbesar, sedangkan untuk data bertipe *polyline* untuk data vektor jalan memiliki rata-rata waktu *query* yang hampir sama dari masing-masing ke tiga prosentase *reinsert*, sedangkan *runtime* indeks terkecil diperoleh dari prosentase *reinsert* sebesar 30% dengan

runtime sebesar 62,88151 ms, dengan demikian kecepatan *runtime query* suatu area pada data spasial meggunakan struktur data R*-tree akan efisien dengan pengindeksan yang menggunakan prosentase *reinsert* yang semakin kecil, karena semakin besar prosentase *reinsert* pada proses *reintegrasi* struktur data R*-tree yang memerlukan optimasi dan minimasi mempengaruhi efisiensi besarnya waktu pengindeksan dan *query*.

5.2 Saran

Untuk pengembangan lebih lanjut perangkat lunak maka ada beberapa saran yang dapat diberikan yaitu melakukan uji coba *query* dengan mengkombinasikan algoritma *K-Nearest Neighbor Query* untuk pencarian pada data spasial menggunakan struktur data R*-tree.

DAFTAR PUSTAKA

Beckmann, N., Kriegel, H.,P., Schneider, R., & Seeger, B. 1990. The R∗-tree: an Efficient and Robust Method for Points and Rectangles, Proceedings ACM SIGMOD Conference on Management of Data, Atlantic City, pp.322-331 NJ.

Guttman, A. 1984. R-Trees: A Dynamic Index Structure for Spatial Searching, University Of California,Berkeley.

Handayani, D.,U.,N. 2001, Sistem Berkas, Yogyakarta : J&J Learning

Hermawan, I. 2009. GEOGRAFI Sebuah Pengantar, Private Publishing, Bandung.

Karen, K., & Kemp. 2008. Encyclopedia Of Geographic information Science, Thousand Oaks, California : SAGE Publications, Inc

Longley, P., A., Goodchild M., F., Maguire, D.,J., Rhind, D., W. 2005. Geographical Information Systems and Science 2nd Edition, England: John Wiley & Sons Ltd

Manolopoulos, Y., Nanopoulos, A., Papadopoulos, A., N., & Theodoridis, Y. 2006. R-Trees: Theory and Applications, London: Springer-Verlag

Popescu, A., R. 2003. A Study of R-tree Based Spatial Access Methods, UNIVERSITY OF HELSINKI, Department of Computer Science, Helsinki.

Puntodewo, A., Dewi, S., & Tarigan, J. 2003. Sistem informasi geografis untuk pengelolaan sumberdaya alam, Bogor: Center for International Forestry Research (CIFOR).

Rahman, A., & Pilouk, M. 2008. Spatial Data Modelling for 3D GIS, Berlin : Springer-Verlag. Hal:15-16.

Rigaux, P., Scholl, M., Voisard, A. 2002. *Spatial Databases: With Application to GIS,* San Francisco : Elsevier Science

Yeung, A., K.,W., Hall, G., B. 2007. Spatial Database Systems Design, Implementation and Project Management 3300 AA Dordrecht, The Netherlands: Springer

Lampiran-Lampiran

- **Hasil Uji Coba Tipe Data Polyline**
- **Uji Coba Pengindeksan**

Tabel 4.1 Runtime Pengindeksan Reinsert 30%, 50% dan 70% tipe data Polyline

Reinsert	Runtime uji ke (ms)										Rata-rata Runtime
	1	2	3	4	5	6	7	8	9	10	
30%	110,25871	108,18889	112,2436	112,1157	109,8081	112,2137	112,0187	113,91	111,5382	112,1313	111,4427
50%	112,74645	109,97571	110,93393	110,5836	111,429	110,485	111,2745	112,9551	111,3681	110,1975	111,1949
70%	110,70374	110,72245	112,61906	110,0704	110,6646	109,8601	110,568	110,7135	110,8982	112,5755	110,9395

- **Uji Coba Query**

Tabel 4.2 Koordinat uji coba query tipe data polyline

No	MBR Range Query			
	X1	X2	Y1	Y2
1	10.057.846	10.076.135	2.115.589	2.164.631
2	8.743.292	8.846.775	18.526.744	20.245.477
3	8.684.359	8.743.292	20.245.477	2.115.689
4	11.544.101	11.548.423	32.616.191	32.663.682
5	9.909.986	9.982.539	2.373.166	25.182.058
6	9.297.396	9.336.692	17.995.392	18.031.494
7	9357.56	9.491.053	17.935.885	18.092.994
8	115.301.875	11.544.101	32.152.256	32.616.191
9	9.164.316	9.244.663	14.571.367	15.127.095
10	8.846.775	9.073.566	18.450.398	18.700.626
11	9.800.937	9848.27	22.552.048	24.215.222
12	10.355.304	10.380.115	19.008.768	19.380.632
13	89.563.545	9.073.927	19.344.089	2.096.062
14	10.057.846	10.076.135	2.115.589	2.164.631
15	8.743.292	8.846.775	18.526.744	20.245.477
16	8.684.359	8.743.292	20.245.477	2.115.689

No	MBR Range Query			
	X1	X2	Y1	Y2
17	10.492.945	10.510.816	27.127.773	27.616.223
18	9.073.927	9.339.549	18.000.159	19.344.089
19	9.813.921	9.900.114	22.552.048	23.316.829
20	10.380.115	10.517.512	19.008.768	21.493.213
21	9.817.978	9.845.763	18.597.588	1.908.523
22	9.940.689	9.978.825	16.973.055	1.857.529
23	9.897.017	9912.97	23.316.829	2.373.166
24	10228.01	10.380.115	18.144.607	19.008.768
25	101.505.625	10.340.112	20.341.769	20.661.514
26	10.179.047	10.345.088	19.799.131	20.275.204
27	10.179.047	10.229.085	18.144.607	19.799.131
28	10.446.971	10.564.235	2.402.434	27.127.773
29	9.978.825	10228.01	16.973.055	18.151.449
30	10.132.846	10.260.679	2.064.568	22.860.923
31	10.095.858	10.260.679	22.150.076	22.860.923
32	10.031.549	10.098.526	25.423.787	25.695.952

Tabel 4.3 Runtime query tipe data polyline (Reinsert 30%)

Query	Runtime Uji Ke (ms)										Rata-rata Runtime	Record
	1	2	3	4	5	6	7	8	9	10		
1	0,00335	0,00363	0,00335	0,00335	0,00363	0,00335	0,00335	0,00335	0,00335	0,00335	0,003406	0,00335
2	0,00307	0,00307	0,00307	0,00335	0,00307	0,00307	0,00307	0,00307	0,00335	0,00307	0,003126	0,00307
3	0,00307	0,00335	0,00307	0,00335	0,00335	0,00335	0,00335	0,00307	0,00335	0,00307	0,003238	0,00307
4	0,00335	0,00307	0,00335	0,00307	0,00307	0,00307	0,00307	0,00307	0,00307	0,00335	0,003154	0,00335
5	0,00335	0,00307	0,00307	0,00335	0,00307	0,00307	0,00307	0,00307	0,00307	0,00307	0,003126	0,00335
6	0,00363	0,00335	0,00335	0,00335	0,00335	0,00307	0,00307	0,00307	0,00335	0,00335	0,003294	0,00363
7	0,00391	0,00363	0,00335	0,00363	0,00335	0,00363	0,00335	0,00363	0,00363	0,00307	0,003518	0,00391
8	0,00363	0,00335	0,00363	0,00363	0,00363	0,00363	0,00335	0,00363	0,00335	0,00363	0,003546	0,00363
9	0,00391	0,00391	0,00391	0,00391	0,00391	0,00391	0,00391	0,00419	0,00391	0,00391	0,003938	0,00391
10	0,00335	0,00335	0,00335	0,00335	0,00363	0,00335	0,00335	0,00335	0,00307	0,00335	0,00335	0,00335
11	0,00475	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	0,00349	0,00475
12	0,00419	0,00391	0,00447	0,00391	0,00391	0,00391	0,00391	0,00391	0,00391	0,00419	0,004022	0,00419
13	0,00475	0,00447	0,00419	0,00419	0,00447	0,00447	0,00419	0,00447	0,00419	0,00419	0,004358	0,00475
14	0,00391	0,00391	0,00391	0,00391	0,00391	0,00391	0,00391	0,00391	0,00391	0,00391	0,00391	0,00391
15	0,00419	0,00391	0,00391	0,00391	0,00419	0,00391	0,00391	0,00391	0,00391	0,00419	0,003994	0,00419
16	0,00419	0,00419	0,00391	0,00391	0,00419	0,00419	0,00419	0,00391	0,00391	0,00391	0,00405	0,00419
17	0,00419	0,00391	0,00391	0,00391	0,00447	0,00391	0,00391	0,00391	0,00726	0,00391	0,004329	0,00419
18	0,00475	0,00447	0,00447	0,00447	0,00447	0,00447	0,00447	0,00447	0,00447	0,00447	0,004498	0,00475
19	0,00475	0,00475	0,00475	0,00475	0,00475	0,00447	0,00447	0,00475	0,00475	0,00475	0,004694	0,00475
20	0,00447	0,00447	0,00447	0,00447	0,00447	0,00475	0,00447	0,00447	0,00447	0,00447	0,004498	0,00447
21	0,00447	0,00447	0,00447	0,00447	0,00419	0,00447	0,00419	0,00447	0,00419	0,00475	0,004414	0,00447
22	0,00447	0,00475	0,00447	0,00419	0,00475	0,00447	0,00447	0,00419	0,00447	0,00419	0,004442	0,00447
23	0,00419	0,00419	0,00419	0,00391	0,00391	0,00447	0,00419	0,00391	0,00391	0,00419	0,004106	0,00419
24	0,00391	0,00419	0,00419	0,00391	0,00391	0,00391	0,00391	0,00391	0,00391	0,00419	0,003994	0,00391
25	0,00419	0,00419	0,00391	0,00419	0,00419	0,00419	0,00391	0,00419	0,00419	0,00391	0,004106	0,00419
26	0,00419	0,00391	0,00391	0,00391	0,00391	0,00391	0,00391	0,00391	0,00391	0,00391	0,003938	0,00419
27	0,00419	0,00391	0,00447	0,00391	0,00447	0,00419	0,00419	0,00419	0,00419	0,00391	0,004162	0,00419
28	0,00475	0,00475	0,00447	0,00447	0,00447	0,00419	0,00447	0,00447	0,00419	0,00447	0,00447	0,00475
29	0,00447	0,00475	0,00475	0,00475	0,00475	0,00475	0,00475	0,00447	0,00475	0,00475	0,004694	0,00447
30	0,00475	0,00475	0,00475	0,00475	0,00447	0,00475	0,00475	0,00447	0,00447	0,00475	0,004666	0,00475
31	0,00475	0,00475	0,00503	0,00475	0,00475	0,00475	0,00503	0,00475	0,00475	0,00503	0,004834	0,00475
32	0,00531	0,00531	0,00531	0,00503	0,00531	0,00503	0,00503	0,00503	0,00503	0,00503	0,005142	0,00531

Tabel 4.4 Rata-rata runtime query tipe data polyline (Reinsert 30%)

No	Runtime Uji Ke (ms)										Rata-rata Runtime	Record
	1	2	3	4	5	6	7	8	9	10		
1	0,00335	0,00363	0,00335	0,00335	0,00363	0,00335	0,00335	0,00335	0,00335	0,00335	0,003406	10
2	0,00307	0,00321	0,00307	0,00335	0,00321	0,00321	0,00321	0,00307	0,00335	0,00307	0,003182	20
3	0,00363	0,003397	0,003443	0,00349	0,003397	0,003397	0,003303	0,003443	0,003397	0,003397	0,003429	30
4	0,00419	0,003817	0,003863	0,00377	0,00391	0,003817	0,00377	0,003817	0,003723	0,003863	0,003854	40

5	0,00419	0,00405	0,00391	0,00391	0,00433	0,00405	0,00405	0,00391	0,005585	0,00391	0,00419	50
6	0,00443	0,00447	0,00443	0,00431	0,00435	0,00443	0,00431	0,00431	0,00431	0,00443	0,004378	60
7	0,00419	0,00405	0,00391	0,00405	0,00405	0,00405	0,00391	0,00405	0,00405	0,00391	0,004022	70
8	0,004703	0,004703	0,004797	0,00461	0,004703	0,00461	0,004703	0,004563	0,004563	0,004657	0,004661	80

Tabel 4.5 Runtime query tipe data polyline (Reinsert 50%)

Query	Runtime Uji Ke (ms)										Rata-rata Runtime	Record
	1	2	3	4	5	6	7	8	9	10		
1	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	0,00363	0,00335	0,003378	9
2	0,00363	0,00335	0,00307	0,00335	0,00335	0,00307	0,00335	0,00335	0,00335	0,00363	0,00335	18
3	0,00363	0,00363	0,00363	0,00363	0,00335	0,00335	0,00363	0,00335	0,00363	0,00335	0,003518	18
4	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	0,00307	0,00307	0,00307	0,00307	0,003238	27
5	0,00335	0,00307	0,00335	0,00279	0,00307	0,00335	0,00335	0,00335	0,00307	0,00335	0,00321	28
6	0,00335	0,00335	0,00307	0,00335	0,00335	0,00335	0,00307	0,00335	0,00307	0,00363	0,003294	31
7	0,00363	0,00335	0,00363	0,00335	0,00363	0,00363	0,00363	0,00335	0,00363	0,00335	0,003518	31
8	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	31
9	0,00391	0,00419	0,00391	0,00419	0,00391	0,00391	0,00391	0,00419	0,00391	0,00391	0,003994	34
10	0,00363	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	0,00363	0,00335	0,00335	0,003406	36
11	0,00391	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	0,00307	0,00335	0,00335	0,003378	36
12	0,00419	0,00391	0,00391	0,00419	0,00391	0,00391	0,00419	0,00419	0,00419	0,00391	0,00405	37
13	0,00419	0,00419	0,00419	0,00419	0,00391	0,00391	0,00419	0,00419	0,00419	0,00419	0,004134	41
14	0,00475	0,00475	0,00447	0,00447	0,00475	0,00447	0,00447	0,00447	0,00475	0,00475	0,00461	42
15	0,00419	0,00391	0,00391	0,00391	0,00391	0,00391	0,00391	0,00363	0,00391	0,00391	0,00391	42
16	0,00419	0,00447	0,00419	0,00419	0,00419	0,00419	0,00419	0,00447	0,00419	0,00503	0,00433	47
17	0,00419	0,00363	0,00391	0,00391	0,00391	0,00391	0,00391	0,00419	0,00391	0,00419	0,003966	53
18	0,00419	0,00419	0,00419	0,00419	0,00419	0,00391	0,00419	0,00391	0,00419	0,00419	0,004134	58
19	0,00475	0,00475	0,00447	0,00447	0,00475	0,00447	0,00447	0,00447	0,00447	0,00447	0,004554	58
20	0,00447	0,00447	0,00447	0,00447	0,00419	0,00391	0,00419	0,00419	0,00419	0,00447	0,004302	59
21	0,00419	0,00391	0,00391	0,00391	0,00391	0,00391	0,00391	0,00419	0,00391	0,00391	0,003966	61
22	0,00447	0,00447	0,00447	0,00447	0,00419	0,00447	0,00447	0,00447	0,00419	0,00419	0,004386	61
23	0,00475	0,00475	0,00475	0,00475	0,00447	0,00447	0,00475	0,00475	0,00447	0,00475	0,004666	62
24	0,00419	0,00419	0,00447	0,00419	0,00419	0,00447	0,00419	0,00419	0,00419	0,00419	0,004246	64
25	0,00475	0,00447	0,00475	0,00447	0,00475	0,00475	0,00475	0,00447	0,00447	0,00475	0,004638	66
26	0,00782	0,00391	0,00391	0,00363	0,00363	0,00391	0,00391	0,00391	0,00363	0,00363	0,004189	73
27	0,00419	0,00419	0,00447	0,00419	0,00419	0,00419	0,00419	0,00447	0,00447	0,00419	0,004274	75
28	0,00419	0,00419	0,00419	0,00419	0,00447	0,00419	0,00363	0,00419	0,00419	0,00419	0,004162	75
29	0,00475	0,00475	0,00475	0,00447	0,00475	0,00447	0,00419	0,00447	0,00475	0,00447	0,004582	76
30	0,00615	0,00447	0,00447	0,00447	0,00419	0,00447	0,00447	0,00447	0,00391	0,00419	0,004526	77
31	0,00391	0,00419	0,00419	0,00419	0,00419	0,00419	0,00419	0,00419	0,00419	0,00419	0,004162	80
32	0,00531	0,00531	0,00531	0,00559	0,00531	0,00531	0,00531	0,00559	0,00559	0,00559	0,005422	83

Tabel 4.6 Rata-rata runtime query tipe data polyline (Reinsert 50%)

No	Runtime Uji Ke (ms)										Rata-rata Runtime	Record
	1	2	3	4	5	6	7	8	9	10		
1	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	0,00363	0,00335	0,003378	10
2	0,00363	0,00349	0,00335	0,00349	0,00335	0,00321	0,00349	0,00335	0,00349	0,00349	0,003434	20
3	0,00349	0,003443	0,003443	0,003397	0,003443	0,00349	0,003397	0,003443	0,00335	0,003443	0,003434	30
4	0,004143	0,00391	0,003863	0,00391	0,003863	0,003817	0,00391	0,003863	0,003957	0,00391	0,003915	40
5	0,00419	0,00405	0,00405	0,00405	0,00405	0,00405	0,00405	0,00433	0,00405	0,00461	0,004148	50
6	0,00443	0,00439	0,00439	0,00435	0,00427	0,00423	0,00431	0,00431	0,00423	0,00431	0,004322	60
7	0,006285	0,00419	0,00433	0,00405	0,00419	0,00433	0,00433	0,00419	0,00405	0,00419	0,004414	70
8	0,00475	0,004517	0,004563	0,004517	0,004517	0,00447	0,00433	0,004563	0,004517	0,00447	0,004521	80

Tabel 4.7 Runtime query tipe data polyline (Reinsert 70%)

Query	Runtime Uji Ke (ms)										Rata-rata Runtime	Record
	1	2	3	4	5	6	7	8	9	10		
1	0,00307	0,00279	0,00279	0,00307	0,00279	0,00279	0,00279	0,00307	0,00307	0,00279	0,002902	9
2	0,00335	0,00335	0,00335	0,00307	0,00335	0,00307	0,00307	0,00335	0,00335	0,00335	0,003266	18
3	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	0,00307	0,00391	0,00335	0,003378	18
4	0,00307	0,00335	0,00307	0,00307	0,00335	0,00307	0,00307	0,00335	0,00335	0,00335	0,00321	27
5	0,00307	0,00307	0,00307	0,00307	0,00335	0,00307	0,00307	0,00307	0,00335	0,00307	0,003126	28
6	0,00335	0,00307	0,00307	0,00363	0,00335	0,00307	0,00335	0,00335	0,00307	0,00335	0,003266	31
7	0,00363	0,00363	0,00363	0,00391	0,00363	0,00363	0,00363	0,00363	0,00363	0,00363	0,003658	31
8	0,00419	0,00419	0,00419	0,00419	0,00391	0,00419	0,00391	0,00419	0,00419	0,00391	0,004106	31
9	0,00475	0,00419	0,00391	0,00391	0,00419	0,00419	0,00419	0,00391	0,00419	0,00419	0,004162	34
10	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	36
11	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	0,00335	36
12	0,00447	0,00391	0,00391	0,00391	0,00419	0,00391	0,00391	0,00391	0,00391	0,00391	0,003994	37
13	0,00391	0,00363	0,00363	0,00391	0,00391	0,00363	0,00363	0,00391	0,00391	0,00391	0,003798	41
14	0,00447	0,00447	0,00447	0,00419	0,00391	0,00447	0,00419	0,00447	0,00419	0,00419	0,004302	42
15	0,00419	0,00391	0,00391	0,00391	0,00391	0,00391	0,00363	0,00391	0,00419	0,00391	0,003938	42
16	0,00391	0,00363	0,00391	0,00363	0,00391	0,00363	0,00363	0,00391	0,00391	0,00391	0,003798	47
17	0,00475	0,00447	0,00447	0,00503	0,00447	0,00475	0,00475	0,00475	0,00475	0,00475	0,004694	53
18	0,00475	0,00475	0,00475	0,00475	0,00475	0,00475	0,00475	0,00475	0,00503	0,00503	0,004806	58
19	0,00503	0,00503	0,00503	0,00503	0,00503	0,00475	0,00475	0,00475	0,00503	0,00503	0,004946	58
20	0,00419	0,00447	0,00447	0,00475	0,00447	0,00447	0,00447	0,00475	0,00475	0,00475	0,004554	59
21	0,00447	0,00447	0,00419	0,00419	0,00419	0,00419	0,00419	0,00419	0,00447	0,00447	0,004302	61
22	0,00447	0,00447	0,00447	0,00447	0,00419	0,00419	0,00419	0,00419	0,00475	0,00419	0,004358	61
23	0,00447	0,00447	0,00447	0,00475	0,00447	0,00447	0,00447	0,00447	0,00447	0,00419	0,00447	62
24	0,00419	0,00419	0,00419	0,00391	0,00419	0,00419	0,00419	0,00419	0,00419	0,00447	0,00419	64

25	0,00531	0,00503	0,00475	0,00475	0,00475	0,00531	0,00503	0,00503	0,00503	0,00503	0,005002	66
26	0,00391	0,00419	0,00419	0,00391	0,00419	0,00391	0,00419	0,00419	0,00391	0,00419	0,004078	73
27	0,00475	0,00447	0,00419	0,00475	0,00447	0,00447	0,00447	0,00447	0,00447	0,00447	0,004498	75
28	0,00447	0,00419	0,00419	0,00419	0,00447	0,00447	0,00419	0,00447	0,00447	0,00447	0,004358	75
29	0,00419	0,00419	0,00419	0,00391	0,00419	0,00419	0,00419	0,00419	0,00391	0,00419	0,004134	76
30	0,00447	0,00447	0,00447	0,00475	0,00447	0,00419	0,00447	0,00447	0,00475	0,00447	0,004498	77
31	0,00391	0,00391	0,00419	0,00419	0,00419	0,00391	0,00391	0,00391	0,00391	0,00391	0,003994	80
32	0,00447	0,00447	0,00447	0,00419	0,00447	0,00447	0,00447	0,00447	0,00475	0,00419	0,004442	83

Tabel 4.8 Rata-rata runtime query tipe data polyline (Reinsert 70%)

No	Runtime Uji Ke (ms)										Rata-rata Runtime	Record
	1	2	3	4	5	6	7	8	9	10		
1	0,00307	0,00279	0,00279	0,00307	0,00279	0,00279	0,00279	0,00307	0,00307	0,00279	0,002902	10
2	0,00335	0,00335	0,00335	0,00321	0,00335	0,00321	0,00321	0,00321	0,00363	0,00335	0,003322	20
3	0,003677	0,003583	0,00349	0,00363	0,00363	0,003537	0,003537	0,00358	0,00363	0,003583	0,003588	30
4	0,003957	0,00377	0,00377	0,00377	0,00377	0,00377	0,003677	0,00382	0,00382	0,00377	0,003789	40
5	0,00433	0,00405	0,00419	0,00433	0,00419	0,00419	0,00419	0,00433	0,00433	0,00433	0,004246	50
6	0,00451	0,00455	0,00451	0,00455	0,00447	0,00443	0,00443	0,00447	0,00467	0,00459	0,004518	60
7	0,00461	0,00461	0,00447	0,00433	0,00447	0,00461	0,00461	0,00461	0,00447	0,00461	0,00454	70
8	0,004377	0,004283	0,004283	0,00433	0,004377	0,004283	0,004283	0,00433	0,00438	0,004283	0,004321	80

- **Hasil Uji Coba Tipe Data polygon**
- **Uji Coba Pengindeksan**

Tabel 4.9 Runtime Pengindeksan Reinsert 30%, 50% dan 70% tipe data Polygon

Reinsert	Runtime uji ke (ms)										Rata-rata Runtime
	1	2	3	4	5	6	7	8	9	10	
30%	85,93327	86,25845	86,67051	86,52022	85,79582	86,73393	85,61619	85,45779	86,08887	86,86858	86,19436
50%	85,88913	86,67303	85,75978	85,70922	85,57568	85,08707	86,7032	86,86635	86,25063	86,84679	86,13609
70%	86,70404	86,10396	85,93858	86,1548	85,73185	85,4522	86,72638	85,67653	86,22129	86,00115	86,07108

- **Ujicoba Query**

Tabel 4.10 Koordinat uji coba query tipe data polygon

No	MBR Range Query			
	X1	X2	Y1	Y2
1	9.893.918	9.934.056	1.906.139	1.954.083
2	8.753.915	9.043.641	1.965.194	21.606.667
3	8673.5	8.943.044	17.818.885	21.623.333
4	10347.96	10459.46	18.688.069	1.951.833
5	9.865.641	9.949.653	18.370.791	19.142.441
6	8.941.833	9.247.829	17.819.163	20.845.298
7	9.761.334	9.871.278	1.909.555	1.971.389
8	11.272.194	11.712.237	2.799.974	3.272.081
9	10005.7	10.374.549	17.921.895	20.403.049
10	10184.62	10285.17	2.165.555	22.458.892
11	9.903.461	10.060.061	1.993.804	21.643.892
12	9.037.889	9.412.307	14.550.547	1.799.139
13	9.098.242	9.413.782	17.241.108	18.651.676
14	10394.49	10.664.751	20.693.254	23.067.458
15	9844.39	10.121.859	2.319.444	2.779.417
16	10.329.015	10914.64	25.606.091	31.786.804

No	MBR Range Query			
	X1	X2	Y1	Y2
17	10.940.417	11.505.973	22.863.887	28.049.443
18	9.802.945	9982.91	19.585.399	2.139.889
19	9.716.862	10008.92	22.221.309	27.665.857
20	9.386.806	9.855.469	1.564.361	18.681.111
21	9.801.334	10.218.086	16.319.343	18.860.189
22	9.972.473	10210.88	1.990.778	2.186.139
23	9.672.639	9.905.049	17.892.219	2080.5
24	10847.13	11.504.149	26.332.776	32.504.448
25	9.858.139	10.058.641	18.390.829	2027.0
26	9.980.781	10395.47	2456.0	29.880.115
27	10250.5	10.712.471	2.231.921	2.683.861
28	10082.39	10432.96	21.054.438	2.515.555
29	10152.49	10.567.696	1895.5	22.764.722
30	10540.21	10.944.334	22.468.362	2.706.139
31	9358.94	9.864.223	1.715.028	2.247.139
32	9.834.558	10.228.061	2118.5	2456.0

Tabel 4.11 Runtime query tipe data polygon (Reinsert 30%)

Query	Runtime Uji Ke (ms)										Rata-rata Runtime	Record
	1	2	3	4	5	6	7	8	9	10		
1	0,02151	0,02179	0,01285	0,01257	0,01425	0,01425	0,01453	0,01425	0,01425	0,01481	0,0155	212
2	0,0162	0,019	0,0109	0,0109	0,01229	0,01229	0,01257	0,01285	0,01257	0,00726	0,01263	222
3	0,01592	0,01844	0,01173	0,01145	0,01229	0,01229	0,01201	0,01229	0,01201	0,00782	0,01268	222
4	0,01453	0,01704	0,01062	0,01034	0,01229	0,01201	0,01201	0,01229	0,01229	0,00726	0,01207	285
5	0,05168	0,02263	0,01313	0,01285	0,01285	0,01453	0,01453	0,01453	0,01536	0,01425	0,01863	327
6	0,01676	0,0162	0,01117	0,0109	0,01257	0,01257	0,01257	0,01257	0,01257	0,01257	0,01305	404
7	0,02207	0,01844	0,01592	0,01257	0,01425	0,01453	0,01481	0,01425	0,01425	0,01034	0,01514	498
8	0,01369	0,01397	0,01006	0,01034	0,01173	0,01145	0,01173	0,01173	0,01173	0,01173	0,01182	513
9	0,02067	0,0176	0,01257	0,01313	0,01425	0,01425	0,01453	0,01453	0,01397	0,00894	0,01444	545
10	0,02402	0,02905	0,01369	0,01453	0,01564	0,01509	0,01592	0,0162	0,01564	0,01006	0,01699	568
11	0,03129	0,03101	0,01592	0,01592	0,01732	0,01732	0,01788	0,01704	0,01788	0,02207	0,02037	568
12	0,01676	0,01676	0,0109	0,0109	0,01229	0,01257	0,01201	0,01201	0,01257	0,01285	0,01296	579
13	0,01648	0,02738	0,01313	0,0109	0,01341	0,01257	0,01397	0,01257	0,01285	0,01285	0,01461	579
14	0,0243	0,02402	0,01341	0,01341	0,01536	0,01509	0,01509	0,01536	0,0162	0,01704	0,01693	609
15	0,01425	0,01453	0,01034	0,01062	0,01229	0,01201	0,01173	0,01173	0,01201	0,01229	0,01218	630
16	0,019	0,01704	0,01173	0,01313	0,01341	0,01341	0,01369	0,01341	0,01369	0,00838	0,01369	656
17	0,01453	0,01425	0,01006	0,01117	0,01229	0,01173	0,01173	0,01173	0,01173	0,01173	0,0121	667
18	0,03101	0,03129	0,01592	0,01844	0,01704	0,01816	0,01816	0,0176	0,01816	0,01788	0,02037	682
19	0,02626	0,02626	0,01425	0,01397	0,01536	0,01537	0,0162	0,01536	0,0162	0,01592	0,01752	692
20	0,02151	0,02179	0,01257	0,01257	0,01397	0,01425	0,01453	0,01425	0,01453	0,01509	0,0155	700
21	0,02766	0,02179	0,01453	0,01564	0,01648	0,01648	0,01648	0,01928	0,01648	0,01145	0,01763	709

22	0,0257	0,02542	0,01425	0,01397	0,01481	0,0162	0,0162	0,0162	0,0162	0,01564	0,01746	764
23	0,02235	0,019	0,01341	0,01257	0,01509	0,01453	0,01509	0,01453	0,01481	0,01006	0,01514	770
24	0,01425	0,01453	0,01006	0,01034	0,01201	0,01173	0,01173	0,01145	0,01173	0,01201	0,01198	779
25	0,03213	0,02794	0,01481	0,01508	0,01676	0,01676	0,01732	0,01704	0,01676	0,01732	0,01919	814
26	0,019	0,0162	0,01341	0,01285	0,01397	0,01397	0,01397	0,01369	0,01397	0,00866	0,01397	839
27	0,02514	0,02067	0,01341	0,01425	0,01564	0,01592	0,01536	0,01592	0,01564	0,01034	0,01623	883
28	0,02933	0,02933	0,01536	0,01536	0,01676	0,01704	0,01676	0,0176	0,01732	0,01257	0,01875	895
29	0,02961	0,02933	0,01536	0,01536	0,01732	0,0176	0,01704	0,01732	0,01732	0,0176	0,01939	923
30	0,02011	0,02011	0,01257	0,01257	0,01425	0,01369	0,01509	0,01369	0,01425	0,01397	0,01503	969
31	0,02766	0,0271	0,01453	0,01508	0,0162	0,01704	0,01676	0,01564	0,01648	0,01732	0,01838	1114
32	0,03296	0,04135	0,01648	0,01648	0,01844	0,01844	0,01816	0,01816	0,01872	0,01341	0,02126	1191

Tabel 4.12 Rata-rata runtime query tipe data polygon (Reinsert 30%)

No	Runtime Uji Ke (ms)										Rata-rata Runtime	Record
	1	2	3	4	5	6	7	8	9	10		
1	0,0178767	0,0197433	0,0118267	0,01164	0,0129433	0,012943	0,0130367	0,01313	0,0129433	0,00996333	0,013605	200
2	0,033105	0,019835	0,011875	0,011595	0,01257	0,01327	0,01327	0,01341	0,013825	0,010755	0,015351	300
3	0,01676	0,0162	0,01117	0,0109	0,01257	0,01257	0,01257	0,01257	0,01257	0,01257	0,013045	400
4	0,01788	0,016205	0,01299	0,011455	0,01299	0,01299	0,01327	0,01299	0,01299	0,011035	0,01348	500
5	0,02111	0,0229071	0,0128514	0,0127729	0,0143657	0,014129	0,0144471	0,0142057	0,0144457	0,01372857	0,015496	600
6	0,0233283	0,02207	0,0131767	0,0141533	0,0147583	0,0149	0,0151317	0,0152717	0,0151317	0,01340833	0,016133	700
7	0,022686	0,020618	0,013188	0,012962	0,014528	0,014638	0,014862	0,014582	0,014694	0,012738	0,01555	800
8	0,0280267	0,0264433	0,01471	0,01499	0,0165733	0,016853	0,0163867	0,0169467	0,01676	0,01350333	0,018119	900
9	0,02691	0,02952	0,0145267	0,01471	0,0162967	0,01639	0,01667	0,01583	0,0164833	0,0149	0,018224	1000

Tabel 4.13 Runtime query tipe data polygon (Reinsert 50%)

Query	Runtime Uji Ke (ms)										Rata-rata Runtime	Record
	1	2	3	4	5	6	7	8	9	10		
1	0,02235	0,02235	0,01285	0,01285	0,01425	0,01397	0,01453	0,01481	0,01425	0,01453	0,01567	212
2	0,01592	0,01425	0,0109	0,01173	0,01257	0,01201	0,01201	0,01229	0,01201	0,00698	0,01207	222
3	0,01704	0,03101	0,0109	0,0109	0,01229	0,01229	0,01285	0,01285	0,01229	0,01285	0,01453	222
4	0,0447	0,01508	0,0109	0,01034	0,01229	0,01201	0,01229	0,01201	0,01201	0,01201	0,01536	285
5	0,02347	0,02319	0,01481	0,01313	0,01425	0,01509	0,01425	0,01508	0,01453	0,01509	0,01629	327
6	0,01704	0,01956	0,01118	0,0109	0,01285	0,01285	0,01285	0,01285	0,01257	0,0081	0,01307	404
7	0,02291	0,02319	0,01257	0,01341	0,01425	0,01397	0,01481	0,01453	0,01481	0,01453	0,0159	498
8	0,01453	0,01453	0,01034	0,01006	0,01173	0,01173	0,01201	0,01173	0,01173	0,01229	0,01207	513
9	0,02151	0,01341	0,01397	0,01425	0,01453	0,01425	0,01481	0,01481	0,01425	0,00922	0,0145	545
10	0,02542	0,03492	0,01425	0,01397	0,01592	0,01564	0,01564	0,01536	0,01592	0,01928	0,01863	568
11	0,03296	0,02682	0,01676	0,01648	0,01816	0,0176	0,01816	0,01816	0,0176	0,01257	0,01953	568
12	0,01732	0,01676	0,0109	0,01173	0,01201	0,01201	0,01257	0,01257	0,01229	0,01257	0,01307	579
13	0,02067	0,01732	0,01117	0,0109	0,01229	0,01229	0,01285	0,01257	0,01257	0,01229	0,01349	579
14	0,02514	0,02011	0,03995	0,01425	0,01564	0,01592	0,01564	0,01564	0,01592	0,0109	0,01891	609
15	0,01425	0,0162	0,01062	0,01145	0,01201	0,01229	0,01257	0,01201	0,01257	0,00698	0,0121	630
16	0,01928	0,019	0,01201	0,01201	0,01453	0,01397	0,01341	0,01453	0,01341	0,01341	0,01456	656
17	0,01453	0,01285	0,01006	0,01118	0,01173	0,01201	0,01201	0,01201	0,01173	0,0067	0,01148	667
18	0,03324	0,03296	0,01648	0,01648	0,01816	0,01816	0,0176	0,01816	0,01956	0,019	0,02098	682
19	0,02682	0,02738	0,01453	0,01509	0,01592	0,01564	0,01592	0,01648	0,01648	0,0176	0,01819	692
20	0,02235	0,01816	0,01285	0,01285	0,01453	0,01453	0,01425	0,01481	0,01397	0,0095	0,01478	700
21	0,02933	0,03408	0,01481	0,01508	0,0176	0,01676	0,01676	0,01732	0,01732	0,01229	0,01914	709
22	0,02654	0,02095	0,01648	0,01564	0,01592	0,0162	0,01648	0,0162	0,0162	0,01173	0,01724	764
23	0,02402	0,02766	0,01341	0,01341	0,01509	0,01481	0,01453	0,01481	0,01508	0,01509	0,01679	770
24	0,01537	0,01509	0,01034	0,01062	0,01201	0,01201	0,01229	0,01229	0,01201	0,01201	0,0124	779
25	0,02961	0,0285	0,01536	0,01536	0,01704	0,01676	0,01704	0,01648	0,0162	0,0176	0,019	814
26	0,02011	0,02039	0,01201	0,01257	0,01397	0,01397	0,01397	0,01397	0,01397	0,01425	0,01492	839
27	0,0257	0,02682	0,01453	0,01369	0,0162	0,01592	0,01592	0,0162	0,0162	0,01173	0,01729	883
28	0,03073	0,0243	0,01592	0,01564	0,0176	0,01788	0,0176	0,02123	0,01788	0,01201	0,01908	895
29	0,03101	0,0366	0,01592	0,0162	0,0176	0,01844	0,01788	0,01788	0,01788	0,01844	0,02078	923
30	0,02095	0,02151	0,01257	0,01229	0,01425	0,01453	0,01397	0,01453	0,01425	0,01425	0,01531	969
31	0,03157	0,02738	0,01508	0,01453	0,01676	0,01676	0,01676	0,01704	0,01704	0,01676	0,01897	1114
32	0,03492	0,03436	0,01676	0,0176	0,01816	0,01844	0,01844	0,019	0,01844	0,01844	0,02146	1191

Tabel 4.14 Rata-rata runtime query tipe data polygon (Reinsert 50%)

No	Runtime Uji Ke (ms)										Rata-rata Runtime	Record
	1	2	3	4	5	6	7	8	9	10		
1	0,0184367	0,0225367	0,01155	0,0118267	0,013037	0,012757	0,01313	0,013317	0,01285	0,011453	0,014089	200
2	0,034085	0,019135	0,012855	0,011735	0,01327	0,01355	0,01327	0,013545	0,01327	0,01355	0,015827	300
3	0,01704	0,01956	0,01118	0,0109	0,01285	0,01285	0,01285	0,01285	0,01257	0,0081	0,013075	400
4	0,01872	0,01886	0,011455	0,011735	0,01299	0,01285	0,01341	0,01313	0,01327	0,01341	0,013983	500
5	0,0224671	0,0207914	0,0168029	0,01329	0,014366	0,014286	0,014606	0,014446	0,014446	0,011973	0,015747	600
6	0,0242583	0,0240717	0,0134567	0,0137817	0,015412	0,015178	0,014992	0,015552	0,015412	0,013083	0,01652	700
7	0,02313	0,022518	0,01352	0,01352	0,014806	0,01475	0,014862	0,01475	0,014692	0,014136	0,016068	800
8	0,0291467	0,02924	0,0154567	0,0151767	0,017133	0,017413	0,017133	0,018437	0,01732	0,01406	0,019052	900
9	0,0291467	0,02775	0,0148033	0,0148067	0,01639	0,016577	0,01639	0,016857	0,016577	0,016483	0,018578	1000

Tabel 4.15 Runtime query tipe data polygon (Reinsert 70%)

Query	Runtime Uji Ke (ms)										Rata-rata Runtime	Record
	1	2	3	4	5	6	7	8	9	10		
1	0,02207	0,01788	0,01453	0,01313	0,01481	0,01453	0,01453	0,01481	0,01453	0,01006	0,01509	212
2	0,0162	0,01425	0,0109	0,01062	0,01229	0,01257	0,01229	0,01285	0,01257	0,00754	0,01221	222
3	0,01592	0,01397	0,0109	0,01257	0,01257	0,01229	0,01201	0,01229	0,01257	0,00726	0,01224	222
4	0,01481	0,01704	0,01062	0,01062	0,01257	0,01201	0,01229	0,01229	0,01285	0,01229	0,01274	285

5	0,02235	0,01872	0,01285	0,01397	0,01481	0,01453	0,01481	0,01844	0,01481	0,0095	0,01548	327
6	0,01704	0,01648	0,01118	0,0109	0,01257	0,01257	0,01285	0,01229	0,01257	0,01285	0,01313	404
7	0,02263	0,02347	0,01313	0,01313	0,01453	0,01453	0,01453	0,01481	0,01481	0,01536	0,01609	498
8	0,01425	0,01229	0,01006	0,01118	0,01173	0,01173	0,01173	0,01201	0,01173	0,00726	0,0114	513
9	0,0514	0,02095	0,01313	0,01285	0,01509	0,01453	0,01425	0,01425	0,01481	0,01453	0,01858	545
10	0,02486	0,03911	0,01397	0,01397	0,01592	0,01592	0,01564	0,01592	0,01564	0,0162	0,01872	568
11	0,03185	0,03855	0,01648	0,01676	0,0176	0,01872	0,01872	0,01788	0,01816	0,01397	0,02087	568
12	0,01676	0,01453	0,01062	0,01118	0,01257	0,01257	0,01229	0,01257	0,01257	0,00754	0,01232	579
13	0,01704	0,01648	0,01118	0,01117	0,01257	0,01257	0,01257	0,01229	0,01229	0,01285	0,0131	579
14	0,02458	0,02011	0,01397	0,01536	0,01508	0,01592	0,01564	0,01537	0,01536	0,0109	0,01623	609
15	0,01453	0,01453	0,0109	0,01173	0,01229	0,01201	0,01257	0,01229	0,01229	0,01257	0,01257	630
16	0,01928	0,01928	0,01201	0,01201	0,01369	0,01397	0,01341	0,01369	0,01369	0,01397	0,0145	656
17	0,01453	0,01648	0,01006	0,01062	0,01173	0,01173	0,01229	0,01201	0,01201	0,00754	0,0119	667
18	0,03185	0,04163	0,0162	0,0162	0,01788	0,01844	0,01844	0,01816	0,01816	0,01844	0,02154	682
19	0,02598	0,0257	0,01425	0,01397	0,01648	0,01564	0,01592	0,0162	0,0162	0,01592	0,01763	692
20	0,02207	0,02179	0,01257	0,01285	0,01453	0,01397	0,01425	0,01509	0,01453	0,01425	0,01559	700
21	0,02766	0,02822	0,01509	0,01648	0,04274	0,01648	0,01648	0,01676	0,0162	0,01118	0,02073	709
22	0,02961	0,02542	0,01648	0,01425	0,0162	0,0162	0,01592	0,0162	0,0162	0,01676	0,01833	764
23	0,02375	0,03017	0,01285	0,01369	0,01508	0,01536	0,01509	0,01536	0,01508	0,01536	0,01718	770
24	0,01453	0,01285	0,0109	0,01173	0,01229	0,01257	0,01201	0,01173	0,01201	0,0067	0,01173	779
25	0,02877	0,02822	0,01481	0,01508	0,01704	0,01648	0,0176	0,01704	0,01704	0,01704	0,01891	814
26	0,01956	0,01676	0,01257	0,01425	0,01397	0,01425	0,01425	0,01397	0,01425	0,00866	0,01425	839
27	0,02514	0,0257	0,01397	0,01453	0,0162	0,01592	0,01592	0,01536	0,01648	0,01956	0,01788	883
28	0,03157	0,02989	0,01592	0,01648	0,01788	0,01788	0,01732	0,0176	0,0176	0,01816	0,02003	895
29	0,02989	0,02989	0,01564	0,01676	0,01732	0,01788	0,0176	0,01704	0,01788	0,01732	0,01972	923
30	0,02095	0,02347	0,01201	0,01257	0,01453	0,01453	0,01508	0,01453	0,01453	0,00922	0,01514	969
31	0,02794	0,02207	0,01704	0,01592	0,01648	0,01676	0,01648	0,01704	0,01676	0,01229	0,01788	1114
32	0,03352	0,03268	0,01704	0,01704	0,019	0,01816	0,01872	0,01816	0,019	0,019	0,02123	1191

Tabel 4.16 Rata-rata runtime query tipe data polygon (Reinsert 70%)

No	Runtime uji ke (ms)										Rata-rata Runtime	Record
	1	2	3	4	5	6	7	8	9	10		
1	0,0180633	0,0153667	0,01211	0,01210667	0,013223	0,01313	0,012943	0,013317	0,013223	0,008287	0,013177	200
2	0,01858	0,01788	0,011735	0,012295	0,01369	0,01327	0,01355	0,015365	0,01383	0,010895	0,01411	300
3	0,01704	0,01648	0,01118	0,0109	0,01257	0,01257	0,01285	0,01229	0,01257	0,01285	0,01313	400
4	0,01844	0,01788	0,011595	0,012155	0,01313	0,01313	0,01313	0,01341	0,01327	0,01131	0,013745	500
5	0,02586	0,0234657	0,01289286	0,01328857	0,014446	0,014606	0,014526	0,014367	0,014446	0,012651	0,016055	600
6	0,0235617	0,0255167	0,01336333	0,01368833	0,019508	0,015038	0,015132	0,015318	0,015132	0,01355	0,016981	700
7	0,023244	0,022684	0,013522	0,0138	0,014916	0,014972	0,014974	0,01486	0,014916	0,012904	0,016079	800
8	0,0288667	0,0284933	0,01517667	0,01592333	0,017133	0,017227	0,016947	0,016667	0,01732	0,018347	0,01921	900
9	0,02747	0,0260733	0,01536333	0,01517667	0,01667	0,016483	0,01676	0,016577	0,016763	0,013503	0,018084	1000

- Data uji Coba Polygon

polygone
0
0 -113.13971710205078 29.017776489257812
1 -113.24057006835938 29.06777572631836
2 -113.45084381103516 29.286663055419922
3 -113.51194763183594 29.303054809570312
4 -113.60028076171875 29.439163208007812
5 -113.57389831542969 29.50583267211914
6 -113.58623046875 29.58361053466797
7 -113.40528869628906 29.482498168945312
8 -113.3650131225586 29.403331756591797
9 -113.3819580078125 29.31999969482422
10 -113.18417358398438 29.294166564941406
11 -113.189453125 29.141109466552734
12 -113.12445068359375 29.058887481689453
13 -113.13971710205078 29.017776489257812
14 -115.17945861816406 28.02471923828125
15 -115.30388641357422 28.098888397216797
16 -115.35529327392578 28.090274810791016
17 -115.2497329711914 28.22833251953125
18 -115.28056335449219 28.31583023071289
19 -115.24083709716797 28.37055206298828
20 -115.17888641357422 28.308887481689453
21 -115.1461181640625 28.17888641357422
22 -115.17945861816406 28.02471923828125
23 -115.01789855957031 31.94748878479004
24 -115.0352783203125 31.95722198486328
25 -115.01445007324219 31.90777587890625
26 -114.95305633544922 31.896385192871094
27 -114.8739013671875 31.805553436279297
28 -114.82389831542969 31.79666519165039
29 -114.77944946289062 31.64360809326172
30 -114.85111999511719 31.526386260986328
31 -114.88111877441406 31.154720306396484
32 -114.81806945800781 31.05999755859375
33 -114.82972717285156 30.996109008789062
34 -114.70529174804688 30.924999237060547
35 -114.69389343261719 30.651107788085938
36 -114.62445068359375 30.487777709960938
37 -114.65972900390625 30.1986083984375
38 -114.5452880859375 30.001110076904297
39 -114.4133529663086 29.919166564941406
40 -114.376953125 29.798053741455078
41 -114.30445861816406 29.75916290283203
42 -114.26334381103516 29.78472137451172
43 -114.20612335205078 29.758609771728516
44 -114.05612182617188 29.59583282470703
45 -113.72834777832031 29.357219696044922
46 -113.65084838867188 29.26166534423828
47 -113.65471649169922 29.20861053466797
48 -113.54833984375 29.110275268554688
49 -113.54640197753906 28.95638656616211
50 -113.50473022460938 28.891387939453125
51 -113.45305633544922 28.892498016357422
52 -113.463623046875 28.939163208007812
53 -113.41251373291016 28.964996337890625
54 -113.34834289550781 28.908607482910156
55 -113.34333801269531 28.79582977294922
56 -113.2316665649414 28.830276489257812
57 -113.19416809082031 28.814441680908203
58 -113.11195373535156 28.479721069335938
59 -113.01806640625 28.437496185302734
60 -112.86279296875 28.433330535888672
61 -112.87251281738281 28.275554656982422
62 -112.78778076171875 28.19305419921875
63 -112.77863311767578 28.029720306396484
64 -112.7219467163086 28.002220153808594
65 -112.72239685058594 27.999740600585938
66 -114.14082336425781 28.000539779663086
67 -114.12861633300781 28.023887634277344
68 -114.11195373535156 28.178333282470703
69 -114.18279266357422 28.26166534423828
70 -114.09750366210938 28.398887634277344
71 -114.06360626220703 28.5272216796875
72 -114.14306640625 28.594444274902344
73 -114.16195678710938 28.67166519165039
74 -114.2641830444336 28.683609008789062
75 -114.26167297363281 28.713607788085938
76 -114.39250183105469 28.829998016357422
77 -114.40528869628906 28.88582992553711
78 -114.49083709716797 28.938610076904297
79 -114.54112243652344 28.92916488647461
80 -114.55751037597656 28.975276947021484
81 -114.61361694335938 29.021942138671875
82 -114.64806365966797 29.11444091796875
83 -114.71000671386719 29.134998321533203
84 -114.74445343017578 29.199443817138672
85 -114.95028686523438 29.377498626708984
86 -115.1875 29.427757263183594
87 -115.23251342773438 29.489166259765625
88 -115.46945190429688 29.626388549804688
89 -115.526123046875 29.62833023071289
90 -115.5727767944336 29.696388244628906
91 -115.69389343261719 29.768329620361328
92 -115.6944580078125 29.866943359375
93 -115.72917175292969 29.930274963378906
94 -115.8083267211914 29.95444107055664
95 -115.78307342529297 30.107498168945312
96 -115.82640075683594 30.33194351196289
97 -115.86805725097656 30.384441375732422
98 -115.96806335449219 30.398052215576172
99 -115.92916870117188 30.445552825927734
100 -115.98056030273438 30.496665954589844
101 -115.95777893066406 30.444721221923828
102 -116.01334381103516 30.438888549804688
103 -115.99138641357422 30.37305450439453
104 -116.03639221191406 30.44277572631836
105 -116.05416870117188 30.797775268554688
106 -116.20694732666016 30.89221954345703
107 -116.25750732421875 30.957775115966797
108 -116.3266830444336 30.973609924316406
109 -116.30168151855469 31.0897216796875
110 -116.3369369506836 31.213886260986328
111 -116.49445343017578 31.424999237060547
112 -116.59333801269531 31.46999740600586
113 -116.6775131225586 31.555274963378906
114 -116.63751220703125 31.586666107177734
115 -116.6352767944336 31.658885955810547
116 -116.7216796875 31.74805450439453
117 -116.62583923339844 31.738609313964844
118 -116.60305786132812 31.840274810791016
119 -116.74417114257812 31.916942596435547
120 -116.7841796875 31.98388671875
121 -116.84889221191406 31.996109008789062
122 -116.90972900390625 32.22833251953125
123 -117.02667236328125 32.29999542236328
124 -117.12222290039062 32.45582580566406
125 -117.12237548828125 32.53533172607422
126 -116.10612487792969 32.619407653808594
127 -114.72126007080078 32.72080993652344
128 -114.80860137939453 32.61599349975586
129 -114.81930541992188 32.50444793701172
130 -114.93669891357422 32.47304916381836
131 -114.96420288085938 32.36861038208008
132 -115.04149627685547 32.254669189453125
133 -114.9989013671875 32.1361083984375
134 -115.01789855957031 31.94748878479004
1
0 -111.20612335205078 25.802776336669922
1 -111.23029327392578 25.834163665771484
2 -111.19139099121094 26.03888702392578
3 -111.086669921875 26.074443817138672
4 -111.09945678710938 26.004444122314453
5 -111.06916809082031 25.971385955810547
6 -111.14222717285156 26.00360870361328
7 -111.20612335205078 25.802776336669922
8 -112.13362121582031 25.281108856201172
9 -112.20250701904297 24.84499740600586
10 -112.13278198242188 24.7147216796875
11 -112.14222717285156 24.647777557373047
12 -112.07084655761719 24.59471893310547
13 -112.05223083496094 24.518054962158203
14 -112.17916870117188 24.662498474121094
15 -112.1533432006836 24.708885192871094
16 -112.18055725097656 24.784164428710938
17 -112.30416870117188 24.81055450439453
18 -112.23667907714844 24.914997100830078
19 -112.13362121582031 25.281108856201172
20 -110.69609069824219 25.08884048461914
21 -110.57890319824219 25.033885955810547
22 -110.5322265625 24.884719848632812
23 -110.64389038085938 24.931110382080078
24 -110.70889282226562 25.042499542236328
25 -110.69609069824219 25.08884048461914
26 -111.70806884765625 24.328330993652344
27 -112.01667785644531 24.53249740600586
28 -111.8369369506836 24.54111099243164
29 -111.82613372802734 24.492496490478516
30 -111.69473266601562 24.391666412353516
31 -111.70806884765625 24.328330993652344
32 -109.7883529663086 24.131942749023438
33 -109.87139892578125 24.187496185302734
34 -109.91555786132812 24.36888885498047
35 -109.7883529663086 24.131942749023438
36 -112.72239685058594 27.999740600585938
37 -112.75250244140625 27.834720611572266
38 -112.70584106445312 27.806941986083984
39 -112.67333984375 27.721385955810547
40 -112.62583923339844 27.71277618408203
41 -112.5727767944336 27.63055419921875
42 -112.50389099121094 27.627220153808594
43 -112.34445190429688 27.539997100830078
44 -112.29279327392578 27.341663360595703
45 -112.20333862304688 27.238609313964844
46 -112.23001098632812 27.232776641845703
47 -112.2216796875 27.197498321533203
48 -111.95529174804688 27.101943969726562
49 -111.947509765625 27.076942443847656
50 -112.00418090820312 27.053333282470703
51 -112.03083801269531 27.001110076904297
52 -111.89806365966797 26.83944320678711
53 -111.917236328125 26.738887786865234
54 -111.76194763183594 26.563888549804688
55 -111.73029327392578 26.552776336669922
56 -111.68528747558594 26.602218627929688
57 -111.80612182617188 26.70722198486328
58 -111.86862182617188 26.87249755859375
59 -111.8477783203125 26.901107788085938
60 -111.56083679199219 26.723609924316406
61 -111.55695343017578 26.564998626708984
62 -111.44223022460938 26.51361083984375
63 -111.47944641113281 26.417774200439453
64 -111.40139770507812 26.34555435180664
65 -111.39556884765625 26.236942291259766
66 -111.32112121582031 26.107776641845703
67 -111.36195373535156 25.958332061767578
68 -111.32501220703125 25.84471893310547
69 -111.29332733154297 25.836387634277344
70 -111.29972839355469 25.780277252197266
71 -111.22805786132812 25.715831756591797
72 -111.165283203125 25.577499389648438
73 -111.01834106445312 25.525554656982422
74 -111.01944732666016 25.419166564941406
75 -110.94612121582031 25.308887481689453
76 -110.91111755371094 25.173053741455078
77 -110.85501098632812 25.088333129882812
78 -110.74667358398438 25.01972198486328
79 -110.69084167480469 24.908885955810547
80 -110.66832733154297 24.79694366455078
81 -110.72666931152344 24.674163818359375
82 -110.73417663574219 24.578330993652344
83 -110.68890380859375 24.38055419921875
84 -110.61279296875 24.284442901611328
85 -110.5058364868164 24.221664428710938
86 -110.30416870117188 24.188888549804688
87 -110.34001159667969 24.15999984741211
88 -110.3989028930664 24.182220458984375
89 -110.35417175292969 24.115833282470703
90 -110.26944732666016 24.189163208007812
91 -110.30000305175781 24.334442138671875
92 -110.213623046875 24.351943969726562
93 -110.13945007324219 24.24944305419922
94 -110.00334167480469 24.164165496826172
95 -109.95889282226562 24.043888092041016
96 -109.9202880859375 24.022499084472656
97 -109.8194580078125 24.053054809570312
98 -109.79444885253906 24.021385192871094
99 -109.82362365722656 23.916385650634766
100 -109.6977767944336 23.797775268554688
101 -109.6864013671875 23.65999984741211
102 -109.47834777832031 23.57555389404297
103 -109.40416717529297 23.454166412353516
104 -109.43501281738281 23.232776641845703
105 -109.48834228515625 23.15555191040039
106 -109.66555786132812 23.05388641357422
107 -109.70500183105469 22.98944091796875
108 -109.81305694580078 22.917774200439453
109 -109.95195007324219 22.863887786865234
110 -110.02500915527344 22.902496337890625
111 -110.08056640625 22.986663818359375
112 -110.17195129394531 23.328330993652344
113 -110.24861145019531 23.41527557373047
114 -110.31668090820312 23.56749725341797
115 -110.63445281982422 23.731666564941406
116 -111.04167175292969 24.112220764160156
117 -111.47084045410156 24.334442138671875
118 -111.37779235839844 24.31027603149414
119 -111.602783203125 24.459720611572266
120 -111.6550064086914 24.57971954345703
121 -111.68528747558594 24.593887329101562
122 -111.70612335205078 24.54694366455078
123 -111.79361724853516 24.562496185302734
124 -111.76555633544922 24.522777557373047
125 -111.8083267211914 24.51361083984375
126 -111.82695007324219 24.642498016357422
127 -111.93112182617188 24.746665954589844
128 -112.001953125 24.88638687133789
129 -111.97389221191406 24.75749969482422
130 -112.0341796875 24.76082992553711
131 -112.04139709472656 24.85305404663086
132 -112.05196380615234 24.76972198486328
133 -112.09445190429688 24.73583221435547
134 -112.07084655761719 24.76860809326172
135 -112.12583923339844 24.8780517578125
136 -112.07861328125 24.95638656616211
137 -112.09612274169922 25.025833129882812
138 -112.1489028930664 24.901107788085938
139 -112.17945861816406 24.8941650390625
140 -112.12418365478516 25.05388641357422
141 -112.12806701660156 25.167499542236328
142 -112.06806945800781 25.272220611572266
143 -112.07084655761719 25.68527603149414
144 -112.07890319824219 25.717777252197266
145 -112.08834838867188 25.69832992553711
146 -112.08528137207031 25.56833267211914
147 -112.11334228515625 25.524166107177734
148 -112.11250305175781 25.773609161376953
149 -112.22834777832031 26.014720916748047
150 -112.30917358398438 26.093608856201172
151 -112.34194946289062 26.082496643066406
152 -112.37834167480469 26.25499725341797
153 -112.43000793457031 26.291385650634766
154 -112.48638916015625 26.26888656616211
155 -112.54167175292969 26.29582977294922
156 -112.53695678710938 26.32583236694336
157 -112.67083740234375 26.329166412353516
158 -112.78111267089844 26.412220001220703
159 -112.77084350585938 26.436386108398438
160 -113.10334777832031 26.644996643066406
161 -113.07972717285156 26.689441680908203
162 -113.11666870117188 26.672222137451172
163 -113.22917175292969 26.711387634277344
164 -113.2316665649414 26.78055191040039
165 -113.12806701660156 26.88055419921875
166 -113.12834167480469 26.958885192871094
167 -113.17861938476562 26.97027587890625
168 -113.18501281738281 26.874996185302734
169 -113.24138641357422 26.81749725341797
170 -113.20361328125 26.822219848632812
171 -113.24334716796875 26.79610824584961
172 -113.25028991699219 26.742774963378906
173 -113.44584655761719 26.82166290283203
174 -113.401123046875 26.82416534423828
175 -113.43860626220703 26.84471893310547
176 -113.53279113769531 26.745830535888672
177 -113.59750366210938 26.736942291259766
178 -113.73029327392578 26.836666107177734
179 -113.83556365966797 26.974163055419922
180 -113.90834045410156 27.000553131103516
181 -114.00140380859375 26.982776641845703
182 -114.08778381347656 27.096664428710938
183 -114.17001342773438 27.147499084472656
184 -114.243896484375 27.16555404663086
185 -114.28111267089844 27.14388656616211
186 -114.41056823730469 27.18471908569336
187 -114.4336166381836 27.23194122314453
188 -114.47862243652344 27.24222183227539
189 -114.51363372802734 27.41360855102539
190 -114.60833740234375 27.487499237060547
191 -114.73667907714844 27.533885955810547
192 -114.79945373535156 27.621944427490234
193 -114.86279296875 27.64611053466797
194 -114.84390258789062 27.65805435180664
195 -114.87222290039062 27.694442749023438
196 -114.90834045410156 27.670555114746094
197 -114.95084381103516 27.72083282470703
198 -115.00666809082031 27.72222137451172
199 -115.05972290039062 27.83110809326172
200 -115.04112243652344 27.862777709960938
201 -114.99861145019531 27.8316650390625
202 -114.85582733154297 27.836387634277344
203 -114.61361694335938 27.76721954345703
204 -114.50140380859375 27.76944351196289
205 -114.34722900390625 27.878887176513672
206 -114.33222961425781 27.781387329101562
207 -114.27890014648438 27.732498168945312
208 -114.16584777832031 27.69305419921875
209 -114.04833984375 27.7147216796875
210 -114.00222778320312 27.686664581298828
211 -113.97138977050781 27.71999740600586
212 -114.03611755371094 27.770553588867188
213 -114.16111755371094 27.716941833496094
214 -114.2258529663086 27.76944351196289
215 -114.23722839355469 27.832775115966797
216 -114.31056213378906 27.865554809570312
217 -114.28500366210938 27.946388244628906
218 -114.21826934814453 28.00020980834961
219 -114.1572265625 28.049442291259766
220 -114.1593017578125 28.00054931640625
221 -114.16056823730469 27.96277618408203
222 -114.14082336425781 28.000539779663086
223 -112.72239685058594 27.999740600585938
2
0 -106.62107849121094 21.565311431884766
1 -106.64750671386719 21.693885803222656
2 -106.58556365966797 21.715831756591797
3 -106.5300064086914 21.69083023071289
4 -106.48529052734375 21.612777709960938
5 -106.51500701904297 21.51305389404297
6 -106.62107849121094 21.565311431884766
7 -106.399169921875 21.419719696044922
8 -106.51139831542969 21.450275421142578
9 -106.4716796875 21.510555267333984
10 -106.34638977050781 21.5030517578125
11 -106.32721710205078 21.468608856201172
12 -106.399169921875 21.419719696044922
13 -105.27159118652344 20.693254470825195
14 -105.3227767944336 20.767776489257812
15 -105.417236328125 20.753887176513672
16 -105.54444885253906 20.78499984741211
17 -105.4675064086914 20.819721221923828
18 -105.31861877441406 21.017498016357422
19 -105.24138641357422 21.064720153808594
20 -105.21305847167969 21.22777557373047
21 -105.23861694335938 21.347774505615234
22 -105.18084716796875 21.450275421142578
23 -105.22112274169922 21.520275115966797
24 -105.27971649169922 21.521663665771484
25 -105.4364013671875 21.607776641845703
26 -105.49751281738281 21.77527618408203
27 -105.65416717529297 21.987777709960938
28 -105.64028930664062 22.286663055419922
29 -105.7145767211914 22.468360900878906
30 -105.6947021484375 22.47311019897461
31 -105.64430236816406 22.48617935180664
32 -105.45449829101562 22.54888916015625
33 -105.47940063476562 22.6813907623291
34 -105.57530212402344 22.753049850463867
35 -105.54170227050781 22.83694076538086
36 -105.44419860839844 22.90389060974121
37 -105.48780059814453 22.97110939025879
38 -105.46279907226562 23.040830612182617
39 -105.402099609375 23.067459106445312
40 -105.3125 23.034719467163086
41 -105.1719970703125 23.03972053527832
42 -104.91470336914062 22.924720764160156
43 -104.87999725341797 22.783050537109375
44 -104.99810028076172 22.678890228271484
45 -104.99810028076172 22.548330307006836
46 -104.75749969482422 22.67667007446289
47 -104.66059875488281 22.624439239501953
48 -104.61250305175781 22.47249984741211
49 -104.49140167236328 22.41027069091797
50 -104.34500122070312 22.45138931274414
51 -104.3114013671875 22.319210052490234
52 -104.32959747314453 22.26453971862793
53 -104.40280151367188 22.07638931274414
54 -104.09359741210938 21.785829544067383
55 -104.15280151367188 21.59804916381836
56 -104.20719909667969 21.54722023010254
57 -103.94490051269531 21.374969482421875
58 -103.9614028930664 21.28778076171875
59 -104.0425033569336 21.211389541625977
60 -104.22769927978516 21.177730560302734
61 -104.20999908447266 20.978050231933594
62 -104.2750015258789 20.860830307006836
63 -104.28559875488281 20.708049774169922
64 -104.46720123291016 20.82971954345703
65 -104.53500366210938 20.916109085083008
66 -104.625 20.92361068725586
67 -104.7219009399414 21.012779235839844
68 -104.7699966430664 21.020549774169922
69 -104.94889831542969 20.92555046081543
70 -105.08329772949219 20.92527961730957
71 -105.27159118652344 20.693254470825195
3
0 -101.52490234375 21.856639862060547
1 -101.58830261230469 21.77277946472168
2 -101.5436019897461 21.656940460205078
3 -101.63200378417969 21.5333309173584
4 -101.57640075683594 21.32666015625
5 -101.6583023071289 21.24304962158203
6 -101.83999633789062 21.150550842285156
7 -101.84829711914062 21.10222053527832
8 -102.07469940185547 20.81389045715332
9 -102.09190368652344 20.773609161376953
10 -101.97830200195312 20.591110229492188
11 -102.08920288085938 20.464439392089844
12 -102.10880279541016 20.389080047607422
13 -102.21170043945312 20.343050003051758
14 -102.443603515625 20.338050842285156
15 -102.62220001220703 20.22916030883789
16 -102.77239990234375 20.196889877319336
17 -102.88809967041016 20.163909912109375
18 -103.04889678955078 20.092220306396484
19 -103.09639739990234 20.023889541625977
20 -103.0864028930664 19.98917007446289
21 -103.03890228271484 19.98138999938965
22 -103.02200317382812 19.89971923828125
23 -102.92900085449219 19.952449798583984
24 -102.91510009765625 19.931379318237305
25 -102.80760192871094 19.89958953857422
26 -102.78970336914062 19.89974021911621
27 -102.74150085449219 19.882400512695312
28 -102.72699737548828 19.8186092376709
29 -102.83000183105469 19.757049560546875
30 -102.82379913330078 19.706680297851562
31 -102.76309967041016 19.592500686645508
32 -102.74810028076172 19.473609924316406
33 -102.60919952392578 19.491390228271484
34 -102.5730972290039 19.406940460205078
35 -102.67449951171875 19.22443962097168
36 -102.76809692382812 19.2549991607666
37 -102.96890258789062 19.17527961730957
38 -102.97560119628906 19.096660614013672
39 -103.09500122070312 19.036109924316406
40 -103.13279724121094 18.954999923706055
41 -103.2853012084961 19.066669464111328
42 -103.34829711914062 18.97443962097168
43 -103.47959899902344 18.967220306396484
44 -103.52469635009766 19.07278060913086
45 -103.49199676513672 19.325279235839844
46 -103.64420318603516 19.480270385742188
47 -103.82279968261719 19.392499923706055
48 -104.06780242919922 19.518329620361328
49 -104.14669799804688 19.463890075683594
50 -104.12920379638672 19.3830509185791
51 -104.25559997558594 19.31916046142578
52 -104.38690185546875 19.27111053466797
53 -104.42949676513672 19.284719467163086
54 -104.47309875488281 19.230270385742188
55 -104.54000091552734 19.253889083862305
56 -104.59459686279297 19.14349937438965
57 -104.66278076171875 19.168052673339844
58 -104.73500061035156 19.229999542236328
59 -104.80972290039062 19.22083282470703
60 -104.79695129394531 19.28860855102539
61 -104.8852767944336 19.280277252197266
62 -104.99250793457031 19.34499740600586
63 -105.06973266601562 19.44832992553711
64 -105.10250854492188 19.564998626708984

65 -105.2699966430664 19.67972183227539
66 -105.5186233520578 20.026107788085938
67 -105.56195068359375 20.219165802001953
68 -105.67472839355469 20.371665954589844
69 -105.67696380615234 20.424163818359375
70 -105.56028747558594 20.489997863769531
71 -105.35195922851562 20.513053894042969
72 -105.24417114257812 20.57416534423828
73 -105.23806762695312 20.644443511962891
74 -105.27159118652344 20.693254470825195
75 -105.08329772949219 20.925279617309570
76 -104.94889831542969 20.925550460815430
77 -104.76999664306641 21.020549774169922
78 -104.72190093994141 21.012779235839844
79 -104.625 20.923610687255859
80 -104.53500366210938 20.916109085083008
81 -104.46720123291016 20.829719543457031
82 -104.28559875488281 20.708049774169922
83 -104.27500152587891 20.860830307006836
84 -104.20999908447266 20.978050231933594
85 -104.22769927978516 21.177730560302734
86 -104.04250335693359 21.211389541625977
87 -103.96140289306641 21.287780761718750
88 -103.94490051269531 21.374969482421875
89 -104.20719909667969 21.547220230102539
90 -104.15280151367188 21.598049163818359
91 -104.09359741210938 21.785829544067383
92 -104.40280151367188 22.076389312744141
93 -104.32959474334473 22.264539718627930
94 -104.14420318603516 22.342220306396484
95 -103.95030212402344 22.368049621582031
96 -103.92169952392578 22.510829925537109
97 -104.02940368652344 22.581939697265625
98 -103.99420166015625 22.658609390258789
99 -104.00700378417969 22.764720916748047
100 -103.80200195312500 22.723049163818359
101 -103.77059936523438 22.636669158935547
102 -103.87079620361328 22.577220916748047
103 -103.83360290527344 22.489440917968750
104 -103.88390350341797 22.461109161376953
105 -103.86859893798828 22.183889389038086
106 -103.74140167236328 22.576389312744141
107 -103.65889739990234 22.573329925537109
108 -103.61499786376953 22.524719238281250
109 -103.70089721679688 22.145990371704102
110 -103.68250274658203 22.112779617309570
111 -103.63829803466797 22.081670761108398
112 -103.52230072021484 22.117290496826172
113 -103.37169647216797 22.327499389648438
114 -103.40920257568359 22.435550689697266
115 -103.37249755859375 22.505830764770508
116 -103.17949676513672 22.368610382080078
117 -103.20140075683594 22.307779312133789
118 -103.05560302734375 22.286109924316406
119 -103.12779998779297 22.147779464721680
120 -103.09110260009766 22.090269088745117
121 -103.17079925537109 21.975280761718750
122 -103.29309844970703 21.982500076293945
123 -103.39420318603516 21.933330535888672
124 -103.44699859619141 21.848049163818359
125 -103.54779815673828 21.785549163818359
126 -103.50948859619141 21.731939315795898
127 -103.51419830322266 21.593050003051758
128 -103.65029907226562 21.461389541625977
129 -103.73359680175781 21.516389846801758
130 -103.70279693603516 21.386940002441406
131 -103.75309753417969 21.251659393310547
132 -103.76560211181641 21.223890304565430
133 -103.73699951171875 21.203329086303711
134 -103.64610290527344 21.241939544677734
135 -103.60169982910156 21.188049316406250
136 -103.54280090332031 21.198059082031250
137 -103.05580139160156 21.054439544677734
138 -103.08560180664062 21.1875
139 -103.0344009399414 21.306659698486328
140 -102.96219635009766 21.284719467163086
141 -102.90670013427734 21.328609466552734
142 -102.83360290527344 21.320550918579102
143 -102.68720245361328 21.381940841674805
144 -102.63919830322266 21.546939849853516
145 -102.76969909667969 21.617769241333008
146 -102.74140167236328 21.724170684814453
147 -102.64499664306641 21.763889312744141
148 -102.49310302734375 21.687219619750977
149 -102.24030303955078 21.655550003051758
150 -102.08329772949219 21.768610000061035
151 -102.04609680175781 21.851669311523438
152 -101.96530151367188 21.883050918579102
153 -101.84619903564453 22.011760711669922
154 -101.80030059814453 22.015270233154297
155 -101.52490234375 21.856639862060547
4
0 -101.84619903564453 22.011760711669922
1 -101.96530151367188 21.883050918579102
2 -102.04609680175781 21.851669311523438
3 -102.08329772949219 21.768610000061035
4 -102.24030303955078 21.655550003051758
5 -102.49310302734375 21.687219619750977
6 -102.64499664306641 21.763889312744141
7 -102.74140167236328 21.724170684814453
8 -102.85169982910156 21.823329925537109
9 -102.84470367431641 21.930269241333008
10 -102.70690155029297 22.083330154418945
11 -102.63500213623047 22.278329849243164
12 -102.45059967041016 22.337219238281250
13 -102.32579803466797 22.458890914916992
14 -102.28720092773438 22.456390308059375
15 -102.27359771728516 22.355829238916016
16 -102.21920013427734 22.372219085693359
17 -102.15579986572266 22.324169158935547
18 -102.15440368652344 22.285270690917969
19 -102.02420043945312 22.251939773559570
20 -102.05639648437500 22.137779235839844
21 -101.93640136718750 22.114440917968750
22 -101.84619903564453 22.011760711669922
5
0 -100.2802963256836 20.204509735107422
1 -100.34860229492188 20.057220458984375
2 -100.48139953613281 19.907779693603516
3 -100.67919921875 19.985279083251953
4 -100.84140014648438 19.927499771118164
5 -100.89839935302734 19.941110610961914
6 -100.91529846191406 20.036659240722656
7 -100.98449707031250 20.059719085693359
8 -101.15440368652344 20.086389541625977
9 -101.27439880371094 20.023889541625977
10 -101.36109924316406 20.035270690917969
11 -101.40920257568359 20.079999923706055
12 -101.39939880371094 20.178609848022461
13 -101.46060180664062 20.333679199218750
14 -101.60690307617188 20.318050384521484
15 -101.67389678955078 20.190830231201172
16 -101.82080078125 20.211940765380859
17 -101.88890075683594 20.191110610961914
18 -101.92030334472656 20.211109161376953
19 -101.95059967041016 20.364440917968750
20 -101.98059844970703 20.367290496826172
21 -101.99420166015625 20.326940536499023
22 -101.99559783935547 20.403049468994141
23 -102.10880279541016 20.389080047607422
24 -102.08920288085938 20.464439392089844
25 -101.97830200195312 20.591110229492188
26 -102.09190368652344 20.773609161376953
27 -102.07469940185547 20.813890457153320
28 -101.84829711914062 21.102220535278320
29 -101.83999633789062 21.150550842285156
30 -101.65830230712891 21.243049621582031
31 -101.57640075683594 21.326660156250000
32 -101.63200378417969 21.533330917358398
33 -101.54360198974609 21.656894046020508
34 -101.58830261230469 21.772779464721680
35 -101.52490234375 21.856639862060547
36 -101.42669677734375 21.834999084472656
37 -101.32279968261719 21.861389160156250
38 -101.20390319824219 21.767499923706055
39 -100.96859741210938 21.745000839233398
40 -100.75170135498047 21.568889617919922
41 -100.60810089111328 21.506389617919922
42 -100.54969787597656 21.515550613403320
43 -100.43250274658203 21.650550842285156
44 -100.29699707031250 21.648889541625977
45 -100.19319915771484 21.586259841918945
46 -99.79054260253906 21.419179916381836
47 -99.77751159667969 21.303609848022461
48 -99.72473144531250 21.238889694213867
49 -99.82333374023438 21.174160003662109
50 -99.96112060546875 21.201389312744141
51 -100.00949859619141 21.180269241333008
52 -100.02940368652344 21.090000152587891
53 -100.08059692382812 21.048330307006836
54 -100.11000061035156 20.902219772338867
55 -100.36810302734375 20.925830841064453
56 -100.40529632568359 20.946659088134766
57 -100.46610260009766 20.924999237060547
58 -100.60060119628906 20.691110610961914
59 -100.49330139160156 20.609170913696289
60 -100.44920349121094 20.373609542846680
61 -100.38809967041016 20.331110000061035
62 -100.40440368652344 20.290830612182617
63 -100.28029632568359 20.204509735107422
6
0 -100.122802734375 19.938039779663086
1 -100.18250274658203 20.082220077514648
2 -100.28029632568359 20.204509735107422
3 -100.40440368652344 20.290830612182617
4 -100.38809967041016 20.331110000061035
5 -100.44920349121094 20.373609542846680
6 -100.49330139160156 20.609170913696289
7 -100.60060119628906 20.691110610961914
8 -100.46610260009766 20.924999237060547
9 -100.40529632568359 20.946659088134766
10 -100.36810302734375 20.925830841064453
11 -100.11000061035156 20.902219772338867
12 -100.08059692382812 20.048330307006836
13 -100.02940368652344 20.090000152587891
14 -100.00949859619141 21.180269241333008
15 -99.96112060546875 21.201389312744141
16 -99.82333374023438 21.174160003662109
17 -99.72473144531250 21.238889694213867
18 -99.77751159667969 21.303609848022461
19 -99.79054260253906 21.419179916381836
20 -99.74305725097656 21.521659851074219
21 -99.69056701660156 21.554719924926758
22 -99.57972717285156 21.423889160156250
23 -99.41195678710938 21.461109161376953
24 -99.36695861816406 21.556659698486328
25 -99.29667663574219 21.563890457153320
26 -99.25527954101562 21.625829696655273
27 -99.20001220703125 21.643890380859375
28 -99.08778381347656 21.287220001220703
29 -99.04325866699219 21.268320083618164
30 -99.03460693359375 21.157100677490234
31 -99.06529235839844 21.181659698486328
32 -99.21861267089844 21.112499237060547
33 -99.29437255859375 21.148700714111328
34 -99.31749725341797 21.101390838623047
35 -99.37390136718750 21.098049163818359
36 -99.34500122070312 21.045280456542969
37 -99.39028930664062 20.915279388427734
38 -99.49445343017578 20.815549850463867
39 -99.51918029785156 20.719440460205078
40 -99.48990631103516 20.661130905151367
41 -99.81945800781250 20.512500762939453
42 -99.82910156250000 20.270000457763672
43 -99.95028686523438 20.242769241333008
44 -99.96222686767578 20.129999160766602
45 -100.12280273437500 19.938039779663086
7
0 -98.65940856933594 19.585399627685547
1 -98.65556335449219 19.595550537109375
2 -98.58139038085938 19.738889694213867
3 -98.69500732421875 19.838890075683594
4 -98.87262725830078 19.849720001220703
5 -98.91445922851562 19.803329467773438
6 -98.95806884765625 19.807500839233398
7 -98.97666931152344 19.867219924926758
8 -98.94277954101562 19.992500305175781
9 -99.03083801269531 20.041940689086914
10 -99.20361328125000 19.976940155029297
11 -99.27778625488281 19.820270538330078
12 -99.38195800781250 19.776109695434570
13 -99.42833709716797 19.883050918579102
14 -99.51944732666016 19.951940536499023
15 -99.48557281494141 20.081939697265625
16 -99.55917358398438 20.145000457763672
17 -99.66335296630859 20.138610898437500
18 -99.82910156250000 20.270000457763672
19 -99.81945800781250 20.512500762939453
20 -99.48990631103516 20.661130905151367
21 -99.51918029785156 20.719440460205078
22 -99.49445343017578 20.815549850463867
23 -99.39028930664062 20.915279388427734
24 -99.34500122070312 21.045280456542969
25 -99.37390136718750 21.098049163818359
26 -99.31749725341797 21.101390838623047
27 -99.29437255859375 21.148700714111328
28 -99.21861267089844 21.112499237060547
29 -99.06529235839844 21.181659698486328
30 -99.03460693359375 21.157100677490234
31 -99.04325866699219 21.268320083618164
32 -98.94306945800781 21.293609619140625
33 -98.90556335449219 21.216110229492188
34 -98.81111145019531 21.184999465942383
35 -98.62001037597656 21.215000152587891
36 -98.60582733154297 21.334440231323242
37 -98.51500701904297 21.398889541625977
38 -98.47722625732422 21.352220535278320
39 -98.48722839355469 21.241939544677734
40 -98.41056823730469 21.154159545898438
41 -98.33750915527344 21.152219772338867
42 -98.29888916015625 21.233890533447266
43 -98.26306152343750 21.213050842285156
44 -98.27290344238281 21.183610916137695
45 -98.28816223144531 21.129778815673828
46 -98.21250915527344 21.157220804541016
47 -98.13056945800781 21.074720382690430
48 -98.15306091308594 21.019439697265625
49 -98.17639160156250 21.072509615850830
50 -98.22000122070312 20.961940765380859
51 -98.23056030273438 20.830829620361328
52 -98.36695861816406 20.858610153198242
53 -98.42111206054688 20.790279388427734
54 -98.51083374023438 20.756389617919922
55 -98.49890136718750 20.712499618530273
56 -98.42445373535156 20.718610763549805
57 -98.56611633300781 20.501659393310547
58 -98.49474334716797 20.376390457153320
59 -98.45278930664062 20.359170913696289
60 -98.40223693847656 20.441110610961914
61 -98.33500671386719 20.434999465942383
62 -98.09529113769531 20.661659240722656
63 -98.03083801269531 20.642499923706055
64 -98.02944946289062 20.606939315795898
65 -98.04279327392578 20.506940841674805
66 -98.09818276822266 20.432170867919922
67 -98.16251373291016 20.324720382690430
68 -98.23806762695312 20.313890457153320
69 -98.24582672111914 20.276939392089844
70 -98.24474334716797 20.217220036396484
71 -98.13417053222656 20.198890686035156
72 -98.09584045410156 20.104999542236328
73 -98.25805664062500 19.846109390258789
74 -98.14306640625000 19.672779083251953
75 -98.25750732421875 19.713890075683594
76 -98.31500244140625 19.640550611340332
77 -98.34390258789062 19.589439392089844
78 -98.49223327636719 19.644720077514648
79 -98.65940856933594 19.585399627685547
8
0 -103.47959899902344 18.967220306396484
1 -103.34829711914062 18.974439620971680
2 -103.28530120849609 19.066669464111328
3 -103.13279724121094 18.954999923706055
4 -103.09500122070312 19.036109924316406
5 -102.97560119628906 19.096660614013672
6 -102.96890257889062 19.175279617309570
7 -102.76809692382812 19.254999160766602
8 -102.67449951171875 19.224439620971680
9 -102.57309722900039 19.406940460205078
10 -102.60919952392578 19.491390228271484
11 -102.74810028076172 19.473609924316406
12 -102.76309967041016 19.592500686644508
13 -102.82379913330078 19.706680297851562
14 -102.83000183105469 19.757049560546875
15 -102.72699737548828 19.818609237670900
16 -102.74150085449219 19.882400512695312
17 -102.78970336914062 19.899740219116211
18 -102.80760192871094 19.899589553857422
19 -102.91510009765625 19.931379318237305
20 -102.92900085449219 19.952449798583984
21 -103.02200317382812 19.899719238281250
22 -103.03890228271484 19.981389999389648
23 -103.08640289306664 19.989170074462891
24 -103.09639739990234 20.023889541625977
25 -103.04889678955078 20.092220306396484
26 -102.88809967041016 20.163909912109375
27 -102.77239990234375 20.196889877319336
28 -102.62220001220703 20.229160308837891
29 -102.44360351562500 20.338050842285156
30 -102.21170043945312 20.343050003051758
31 -102.10880279541016 20.389080047607422
32 -101.99559783935547 20.403049468994141
33 -101.99420166015625 20.326940536499023
34 -101.98059844970703 20.367290496826172
35 -101.95059967041016 20.364440917968750
36 -101.92030334472656 20.211109161376953
37 -101.88890075683594 20.191110610961914
38 -101.82080078125 20.211940765380859
39 -101.67389678955078 20.190830231201172
40 -101.60690307617188 20.318050384521484
41 -101.46060180664062 20.333679199218750
42 -101.39939880371094 20.178609848022461
43 -101.40920257568359 20.079999923706055
44 -101.36109924316406 20.035270690917969
45 -101.27439880371094 20.023889541625977
46 -101.15440368652344 20.086389541625977
47 -100.98449707031250 20.059719085693359
48 -100.91529846191406 20.036659240722656
49 -100.89839935302734 19.941110610961914
50 -100.84140014648438 19.927499771118164
51 -100.67919921875 19.985279083251953
52 -100.48139953613281 19.907779693603516
53 -100.34860229492188 20.057220458984375
54 -100.28029632568359 20.204509735107422
55 -100.18250274658203 20.082220077514648
56 -100.12280273437500 19.938039779663086
57 -100.05699920654297 19.877220153808594
58 -100.14420318603516 19.827220916748047
59 -100.18699645996094 19.641389846801758
60 -100.13890075683594 19.415830612182617
61 -100.29720306396484 19.334720611572266
62 -100.28420257568359 19.262779235839844
63 -100.53389739990234 19.983329772949219
64 -100.52940368652344 19.940549850463867
65 -100.58640289306664 19.859720230102539
66 -100.68260192871094 19.786069869995117
67 -100.72810363769531 19.860189437866211
68 -100.76969906967969 19.791109085083008
69 -100.72059631347656 19.525550842285156
70 -100.59359741210938 19.402240753173828
71 -100.62449645996094 19.353330612182617
72 -100.68560028076172 19.387500762939453
73 -100.79309844970703 19.472219467163086
74 -100.91529846191406 19.477500915527344
75 -100.90940009399414 19.450000762939453
76 -100.94670104980469 19.441940307617188
77 -101.01110076904297 19.517219543457031
78 -101.08750152587891 19.501110076904297
79 -101.29560089111328 19.533609390258789
80 -101.45161437988281 19.479204177856445
81 -101.50981903076172 19.485439300537109
82 -101.57425689697266 19.524932861328125
83 -101.61998748779297 19.608079910278320
84 -101.84447479248047 19.595607775756836
85 -101.87773895263672 19.537405140838086
86 -101.86318206787109 19.290046691894531
87 -101.90029907226562 19.261390686035156
88 -101.98750305175781 19.202220916748047
89 -102.14610290527344 19.174160003662109
90 -102.18085479736328 19.921894073486328
91 -102.18890380859375 19.922496795654297
92 -102.48750305175781 19.023330688476562
93 -102.74501037597656 19.065830231201172
94 -103.02944946289062 18.189998626708984
95 -103.45001220703125 18.313610076904297
96 -103.57917785644531 18.500831604003906
97 -103.69862365722656 18.577499389648438
98 -103.68695068359375 18.621387481689453
99 -103.74548339843750 18.688068389892578
100 -103.68309783935547 18.775829315185547
101 -103.63110351562500 18.791940689086914
102 -103.61080169677734 18.889999389648438
103 -103.57749938964844 18.881660461425781
104 -103.47959899902344 18.967220306396484
9
0 -98.62798309326172 19.475759506225586
1 -98.66612243652344 19.405830383300781
2 -98.63694763183594 19.165279388427734
3 -98.66223144531250 18.996759414672852
4 -98.75389099121094 18.968889236450195
5 -98.96385192871094 19.089029312133789
6 -98.93917846679688 19.137779235839844
7 -98.96750640869141 19.306110382080078
8 -99.02834320068359 19.372219085693359
9 -99.03167724609375 19.454439163208008
10 -99.08528137207031 19.475830078125000
11 -99.11306762695312 19.540830612182617
12 -99.22445678710938 19.405830383300781
13 -99.34056091308594 19.357780456542969
14 -99.28486633300781 19.142440795898438
15 -99.32444763183594 19.090549468994141
16 -99.30416870117188 19.972499847412109
17 -99.42973327636719 18.882219314575195
18 -99.49653625488281 18.666709899902344
19 -99.65055847167969 18.764999389648438
20 -99.79583740234375 18.633890151977539
21 -99.88863372802734 18.657779693603516
22 -100.09329986572266 18.607219696044922
23 -100.12249755859375 18.516109466552734
24 -100.25949859619141 18.397499084472656
25 -100.30560302734375 18.390829086303711
26 -100.38610076904297 18.528329849243164
27 -100.45500183105469 18.809999465942383
28 -100.52780151367188 18.844449996948242
29 -100.58640289306664 18.859720230102554
30 -100.52940368652344 18.940549850463867
31 -100.53389739990234 18.983329772949219
32 -100.28420257568359 19.262779235839844
33 -100.29720306396484 19.334720611572266
34 -100.13890075683594 19.415830612182617
35 -100.18699645996094 19.641389846801758
36 -100.14420318603516 19.827220916748047
37 -100.05699920654297 19.877220153808594
38 -100.12280273437500 19.938039779663086
39 -99.96222686767578 20.129999160766602
40 -99.95028686523438 20.242769241333008
41 -99.82910156250000 20.270000457763672
42 -99.66335296630859 20.138610898437500
43 -99.55917358398438 20.145000457763672
44 -99.48557281494141 20.081939697265625
45 -99.51944732666016 19.951940536499023
46 -99.42833709716797 19.883050918579102
47 -99.38195800781250 19.776109695434570
48 -99.27778625488281 19.820270538330078
49 -99.20361328125000 19.976940155029297
50 -99.03083801269531 20.041940689086914
51 -98.94277954101562 19.992500305175781
52 -98.97666931152344 19.867219924926758
53 -98.95806884765625 19.807500839233398
54 -98.91445922851562 19.803329467773438
55 -98.87262725830078 19.849720001220703
56 -98.69500732421875 19.838890075683594
57 -98.58139038085938 19.738889694213867
58 -98.65556335449219 19.595550537109375
59 -98.65940856933594 19.585399627685547
60 -98.71278381347656 19.577499389648438
61 -98.62798309326172 19.475759506225586
10
0 -99.28486633300781 19.142440795898438
1 -99.34056091308594 19.357780456542969
2 -99.22445678710938 19.405830383300781
3 -99.11306762695312 19.540830612182617
4 -99.08528137207031 19.475830078125
5 -99.03083801269531 20.041940689086914
6 -99.02834320068359 19.372219085693359
7 -98.94277954101562 19.992500305175781
8 -98.97666931152344 19.867219924926758
9 -98.96385192871094 19.089029312133789
10 -99.03140258789062 19.061389923095703
11 -99.13362121582031 19.116109848022461
12 -99.28486633300781 19.142440795898438
11
0 -103.74548339843750 18.688068389892578
1 -104.00446319580078 18.896385192871094
2 -104.31500244140625 19.008052825927734
3 -104.32778930664062 19.095275878906250
4 -104.44833374023438 19.090831756591797
5 -104.59459686279297 19.143499374389648
6 -104.54000091552734 19.253889083862305
7 -104.47309875488281 19.230270385742188
8 -104.42949676513672 19.284719467163086
9 -104.38690185546875 19.271110534667969
10 -104.25559997558594 19.319160461425781
11 -104.12920379638672 19.383050918579102
12 -104.14669799804688 19.463890075683594
13 -104.06780242919922 19.518329620361328
14 -103.82279968261719 19.392499923706055
15 -103.64420318603516 19.480270385742188

16 -103.49199676513672 19.325279235839844
17 -103.52469635009766 19.07278060913086
18 -103.47959899902344 18.967220306396484
19 -103.57749938964844 18.88166046142578
20 -103.61080169677734 18.889999389648438
21 -103.631103515625 18.791940689086914
22 -103.68309783935547 18.775829315185547
23 -103.7454833984375 18.688068389892578
12
0 -98.6622314453125 18.99675941467285
1 -98.65640258789062 18.905000686645508
2 -98.74474334716797 18.798330307006836
3 -98.665283203125 18.6924991607666
4 -98.74992370605469 18.71902084350586
5 -98.67111206054688 18.438610076904297
6 -98.69528198242188 18.4183292388916
7 -98.81916809082031 18.4950008392334
8 -98.92250061035156 18.415000915527344
9 -99.05049133300781 18.370790481567383
10 -99.1494369506836 18.533889770507812
11 -99.227783203125 18.526660919189453
12 -99.25639343261719 18.459720611572266
13 -99.31168365478516 18.463329315185547
14 -99.49653625488281 18.666709899902344
15 -99.42973327636719 18.882219314575195
16 -99.30416870117188 18.97249984741211
17 -99.32444763183594 19.09054946899414
18 -99.28486633300781 19.142440795898438
19 -99.13362121582031 19.11610984802246
20 -99.03140258789062 19.061389923095703
21 -98.96385192871094 19.08902931213379
22 -98.75389099121094 18.968889236450195
23 -98.6622314453125 18.99675941467285
13
0 -90.37337493896484 20.84529685974121
1 -90.33889770507812 20.941665649414062
2 -90.3866958618164 20.863229751586914
3 -90.4364013671875 20.781665802001953
4 -90.40779113769531 20.863229751586914
5 -90.38583374023438 20.925830841064453
6 -90.33584594726562 21.02527618408203
7 -90.10639953613281 21.160274505615234
8 -90.0 21.189472198486328
9 -89.7711181640625 21.284442901611328
10 -88.84861755371094 21.411663055419922
11 -88.70806884765625 21.447776794433594
12 -88.60166931152344 21.533885955810547
13 -88.45140075683594 21.568885803222656
14 -88.2711181640625 21.553607940673828
15 -88.08639526367188 21.584999084472656
16 -88.24334716796875 21.567218780517578
17 -88.15695190429688 21.606666564941406
18 -87.99417114257812 21.60277557373047
19 -87.70722961425781 21.536663055419922
20 -87.86528015136719 21.551387786865234
21 -87.75445556640625 21.50555419921875
22 -87.61666870117188 21.498332977294922
23 -87.689453125 21.520275115966797
24 -87.65528869628906 21.528610229492188
25 -87.53915405273438 21.5023136138916
26 -87.54055786132812 21.02471923828125
27 -87.75389099121094 20.662500381469727
28 -89.41832733154297 19.651939392089844
29 -90.0283432006836 20.49444007873535
30 -90.06500244140625 20.443050384521484
31 -90.22695922851562 20.48971939086914
32 -90.20668029785156 20.55777931213379
33 -90.37806701660156 20.553890228271484
34 -90.37337493896484 20.84529685974121
14
0 -91.83445739746094 18.63805389404297
1 -91.84194946289062 18.659442901611328
2 -91.6461181640625 18.75360870361328
3 -91.5533447265625 18.788330078125
4 -91.52389526367188 18.770553588867188
5 -91.52444458007812 18.748607635498047
6 -91.62251281738281 18.736663818359375
7 -91.69139099121094 18.65749740600586
8 -91.72334289550781 18.659442901611328
9 -91.70306396484375 18.695552825927734
10 -91.83445739746094 18.63805389404297
11 -92.47828674316406 18.651676177978516
12 -91.98139953613281 18.72833251953125
13 -91.85890197753906 18.61111068725586
14 -91.87834167480469 18.58110809326172
15 -91.94056701660156 18.591663360595703
16 -91.94639587402344 18.6280517578125
17 -92.0050048828125 18.61471939086914
18 -91.99166870117188 18.590553283691406
19 -92.03445434570312 18.59471893310547
20 -92.05056762695312 18.544719696044922
21 -91.95668029785156 18.543331146240234
22 -91.88612365722656 18.500831604003906
23 -91.97084045410156 18.583885192871094
24 -91.90306091308594 18.575275421142578
25 -91.87055969238281 18.528888702392578
26 -91.88972473144531 18.516666412353516
27 -91.8255615234375 18.495830535888672
28 -91.85722351074219 18.430553436279297
29 -91.8033447265625 18.37999725341797
30 -91.81417846679688 18.44277572631836
31 -91.7711181640625 18.441387176513672
32 -91.80195617675781 18.48444366455078
33 -91.47500610351562 18.439441680908203
34 -91.48194885253906 18.492496490478516
35 -91.53834533691406 18.461387634277344
36 -91.4908447265625 18.518054962158203
37 -91.33416748046875 18.565277099609375
38 -91.3033447265625 18.61888885498047
39 -91.18972778320312 18.6441650390625
40 -91.29750061035156 18.62833023071289
41 -91.26362609863281 18.740833282470703
42 -91.41445922851562 18.811107635498047
43 -91.23722839355469 18.95722198486328
44 -91.37528991699219 18.88249969482422
45 -91.4202880859375 18.81999969482422
46 -91.51112365722656 18.808609008789062
47 -91.43000793457031 18.897220611572266
48 -91.17445373535156 19.002777099609375
49 -90.99751281738281 19.118610382080078
50 -90.75666809082031 19.31583023071289
51 -90.68112182617188 19.762218475341797
52 -90.52334594726562 19.881385803222656
53 -90.45472717285156 19.975276947021484
54 -90.50056457519531 20.085277557373047
55 -90.46501159667969 20.398052215576172
56 -90.49111938476562 20.520832061767578
57 -90.4586181640625 20.72777557373047
58 -90.38362121582031 20.816665649414062
59 -90.37337493896484 20.84529685974121
60 -90.37806701660156 20.553890228271484
61 -90.20668029785156 20.55777931213379
62 -90.22695922851562 20.48971939086914
63 -90.06500244140625 20.443050384521484
64 -90.0283432006836 20.49444007873535
65 -89.41832733154297 19.651939392089844
66 -89.4304428100586 17.819162368774414
67 -90.982421875 17.82065200805664
68 -90.98306274414062 17.9677791595459
69 -91.18860626220703 17.976110458374023
70 -91.32112121582031 18.063329696655273
71 -91.45390319824219 18.09943962097168
72 -91.60917663574219 18.096660614013672
73 -91.62640380859375 17.950830459594727
74 -91.85529327392578 17.95138931274414
75 -91.97944641113281 18.017780303955078
76 -92.15779113769531 18.1572208404541
77 -92.15306091308594 18.511940002441406
78 -92.42167663574219 18.513050079345703
79 -92.47828674316406 18.651676177978516
15
0 -96.7506332397461 18.430830001831055
1 -96.72638702392578 18.385000228881836
2 -96.78751373291016 18.284719467163086
3 -96.88667297363281 18.240829467773438
4 -96.96278381347656 18.150550842285156
5 -97.08000183105469 18.138330459594727
6 -97.20668029785156 18.179439544677734
7 -97.28140258789062 18.16027069091797
8 -97.36974334716797 18.102779388427734
9 -97.44944763183594 17.977779388427734
10 -97.64140319824219 18.172779083251953
11 -97.6138916015625 18.2933292388916
12 -97.64806365966797 18.34054946899414
13 -97.71917724609375 18.30944061279297
14 -97.79640197753906 18.172779083251953
15 -97.7388916015625 17.992769241333008
16 -97.84445190429688 17.924999237060547
17 -97.92262268066406 17.998790740966797
18 -97.94277954101562 18.032779693603516
19 -98.1591796875 18.024999618530273
20 -98.2469482421875 17.909719467163086
21 -98.30972290039062 17.923049926757812
22 -98.3477783203125 17.89221954345703
23 -98.44862365722656 17.994159698486328
24 -98.61695861816406 17.973329544067383
25 -98.7630615234375 18.011390686035156
26 -98.83168029785156 18.13443946838379
27 -98.90416717529297 18.12527084350586
28 -98.92723083496094 18.202220916748047
29 -99.03140258789062 18.23777961730957
30 -99.05049133300781 18.370790481567383
31 -98.92250061035156 18.415000915527344
32 -98.81916809082031 18.4950008392334
33 -98.69528198242188 18.4183292388916
34 -98.67111206054688 18.438610076904297
35 -98.74992370605469 18.71902084350586
36 -98.665283203125 18.6924991607666
37 -98.74474334716797 18.798330307006836
38 -98.65640258789062 18.905000686645508
39 -98.6622314453125 18.99675941467285
40 -98.63694763183594 19.165279388427734
41 -98.66612243652344 19.40583038330078
42 -98.62798309326172 19.475759506225586
43 -98.46833801269531 19.421939849853516
44 -98.46056365966797 19.366939544677734
45 -98.19862365722656 19.095550537109375
46 -98.0824966430664 19.120830535888672
47 -97.99362182617188 19.202770233154297
48 -97.90194702148438 19.156389236450195
49 -97.84390258789062 19.204439163208008
50 -97.83416748046875 19.28166961669922
51 -97.65640258789062 19.286109924316406
52 -97.61334228515625 19.35638999938965
53 -97.68417358398438 19.374439239501953
54 -97.77500915527344 19.456390380859375
55 -97.84722900390625 19.435550689697266
56 -97.88278198242188 19.509719848632812
57 -97.84557342529297 19.54138946533203
58 -97.96417236328125 19.62639045715332
59 -98.01112365722656 19.61610984802246
60 -98.00110626220703 19.677780151367188
61 -98.14306640625 19.672779083251953
62 -98.258056640625 19.84610939025879
63 -98.09584045410156 20.104999542236328
64 -98.13417053222656 20.198890686035156
65 -98.24474334716797 20.217220306396484
66 -98.2458267211914 20.276939392089844
67 -98.23806762695312 20.31389045715332
68 -98.16251373291016 20.32472038269043
69 -98.09818267822266 20.432170867919922
70 -97.96305847167969 20.51972007751465
71 -97.94862365722656 20.66666030883789
72 -97.883056640625 20.706390380859375
73 -97.8739013671875 20.80500030517578
74 -97.73390197753906 20.79305076599121
75 -97.74223327636719 20.650550842285156
76 -97.57917785644531 20.58860969543457
77 -97.57064819335938 20.490299224853516
78 -97.62889099121094 20.417770385742188
79 -97.6933364868164 20.4697208404541
80 -97.75887298583984 20.439899444580078
81 -97.75279235839844 20.255279541015625
82 -97.69249725341797 20.176109313964844
83 -97.61473083496094 20.1683292388916
84 -97.564453125 20.106670379638672
85 -97.51500701904297 20.121110916137695
86 -97.47057342529297 20.23971939086914
87 -97.4122314453125 20.26222038269043
88 -97.3814468383789 20.263729095458984
89 -97.14639282226562 20.147220611572266
90 -97.13722229003906 20.11750030517578
91 -97.3094482421875 19.895549774169922
92 -97.2852783203125 19.75
93 -97.30889892578125 19.683889389038086
94 -97.35444641113281 19.619720458984375
95 -97.44000244140625 19.585830688476562
96 -97.35334777832031 19.538330078125
97 -97.33445739746094 19.401390075683594
98 -97.24639892578125 19.37360954284668
99 -97.18556213378906 19.306659698486328
100 -97.05612182617188 19.30777931213379
101 -97.00167846679688 19.267499923706055
102 -97.07972717285156 19.18305015563965
103 -97.17001342773438 19.1936092376709
104 -97.26471710205078 19.15999984741211
105 -97.25083923339844 19.026390075683594
106 -97.24778747558594 18.88722038269043
107 -97.34529113769531 18.769439697265625
108 -97.27278137207031 18.63249969482422
109 -97.14418029785156 18.643329620361328
110 -97.03861999511719 18.476669311523438
111 -96.80751037597656 18.552780151367188
112 -96.7506332397461 18.430830001831055
16
0 -88.29949951171875 18.482929229736328
1 -88.48394775390625 18.477611541748047
2 -88.60002136230469 18.23583221435547
3 -88.68112182617188 18.18555450439453
4 -88.71058654785156 18.061195373535156
5 -88.83757019042969 17.93695068359375
6 -88.84097290039062 17.877559661865234
7 -89.03584289550781 18.00583267211914
8 -89.14305114746094 17.956066131591797
9 -89.14195251464844 17.818885803222656
10 -89.4304428100586 17.819162368774414
11 -89.41832733154297 19.651939392089844
12 -87.75389099121094 20.662500381469727
13 -87.54055786132812 21.02471923828125
14 -87.53915405273438 21.5023136138916
15 -87.501953125 21.49388885498047
16 -87.48638916015625 21.463886260986328
17 -87.2408447265625 21.43694305419922
18 -87.14111328125 21.487777709960938
19 -87.12918090820312 21.55472183227539
20 -87.17001342773438 21.567218780517578
21 -87.25666809082031 21.527496337890625
22 -87.3416748046875 21.55221939086914
23 -87.3961181640625 21.504165649414062
24 -87.41445922851562 21.52694320678711
25 -87.36029052734375 21.57971954345703
26 -87.26889038085938 21.561664581298828
27 -87.11250305175781 21.623332977294922
28 -87.00334167480469 21.578330993652344
29 -86.90806579589844 21.429443359375
30 -86.82833862304688 21.430274963378906
31 -86.81333923339844 21.183330535888672
32 -86.7388916015625 21.151386260986328
33 -86.78306579589844 21.03249740600586
34 -86.82528686523438 21.012496948242188
35 -86.87834167480469 20.838333129882812
36 -87.0675048828125 20.61471939086914
37 -87.22639465332031 20.503887176513672
38 -87.43000793457031 20.2147216796875
39 -87.47195434570312 20.092777252197266
40 -87.43333435058594 19.895553588867188
41 -87.43453216552734 19.868398666381836
42 -87.44287109375 19.847431182861328
43 -87.47222900390625 19.776386260986328
44 -87.48001098632812 19.8316650390625
45 -87.44733428955078 19.8536319732666
46 -87.44139099121094 19.909164428710938
47 -87.48445129394531 19.94305419921875
48 -87.45973205566406 19.876110076904297
49 -87.52056884765625 19.80221939086914
50 -87.58500671386719 19.79666519165039
51 -87.66445922851562 19.627220153808594
52 -87.6572265625 19.678333282470703
53 -87.7388916015625 19.674720764160156
54 -87.73028564453125 19.592777252197266
55 -87.66084289550781 19.56194305419922
56 -87.67001342773438 19.50527572631836
57 -87.56695556640625 19.56027603149414
58 -87.44612121582031 19.542499542236328
59 -87.41844177246094 19.580665588378906
60 -87.43251037597656 19.600276947021484
61 -87.43667602539062 19.57166290283203
62 -87.52723693847656 19.58138656616211
63 -87.4444580078125 19.57944107055664
64 -87.4375 19.63416290283203
65 -87.41166687011719 19.578887939453125
66 -87.46556091308594 19.447219848632812
67 -87.53167724609375 19.40027618408203
68 -87.570556640625 19.395553588867188
69 -87.54251098632812 19.434444427490234
70 -87.62640380859375 19.39999771118164
71 -87.6763916015625 19.31805419921875
72 -87.6875 19.24777603149414
73 -87.64028930664062 19.210830688476562
74 -87.55223083496094 19.316665649414062
75 -87.5 19.326385498046875
76 -87.51139831542969 19.28221893310547
77 -87.46112060546875 19.313888549804688
78 -87.53973388671875 19.215274810791016
79 -87.65223693847656 18.764999389648438
80 -87.729736328125 18.667774200439453
81 -87.75917053222656 18.412498474121094
82 -87.82806396484375 18.310832977294922
83 -87.84750366210938 18.19083023071289
84 -87.85389709472656 18.235553741455078
85 -87.89111328125 18.241107940673828
86 -87.85751342773438 18.320274353027344
87 -87.88555908203125 18.286388397216797
88 -87.92472839355469 18.438053131103516
89 -88.08084106445312 18.51721954345703
90 -88.00111389160156 18.68138885498047
91 -88.03750610351562 18.869441986083984
92 -88.12251281738281 18.72083282470703
93 -88.13027954101562 18.78221893310547
94 -88.25306701660156 18.68471908569336
95 -88.18861389160156 18.73332977294922
96 -88.1944580078125 18.670555114746094
97 -88.15084838867188 18.686664581298828
98 -88.29949951171875 18.482929229736328
99 -86.99334716796875 20.25555419921875
100 -87.02000427246094 20.391666412353516
101 -86.93890380859375 20.539443969726562
102 -86.90028381347656 20.564441680908203
103 -86.82890319824219 20.541664123535156
104 -86.73500061035156 20.590553283691406
105 -86.88555908203125 20.353885650634766
106 -86.99334716796875 20.25555419921875
17
0 -98.14306640625 19.672779083251953
1 -98.00110626220703 19.677780151367188
2 -98.01112365722656 19.61610984802246
3 -97.96417236328125 19.62639045715332
4 -97.84557342529297 19.54138946533203
5 -97.88278198242188 19.509719848632812
6 -97.84722900390625 19.435550689697266
7 -97.77500915527344 19.456390380859375
8 -97.68417358398438 19.374439239501953
9 -97.61334228515625 19.35638999938965
10 -97.65640258789062 19.286109924316406
11 -97.83416748046875 19.28166961669922
12 -97.84390258789062 19.204439163208008
13 -97.90194702148438 19.156389236450195
14 -97.99362182617188 19.202770233154297
15 -98.0824966430664 19.120830535888672
16 -98.19862365722656 19.095550537109375
17 -98.46056365966797 19.366939544677734
18 -98.46833801269531 19.421939849853516
19 -98.62798309326172 19.475759506225586
20 -98.71278381347656 19.577499389648438
21 -98.65940856933594 19.585399627685547
22 -98.49223327636719 19.64472007751465
23 -98.34390258789062 19.589439392089844
24 -98.31500244140625 19.64055061340332
25 -98.25750732421875 19.713890075683594
26 -98.14306640625 19.672779083251953
18
0 -99.05049133300781 18.370790481567383
1 -99.03140258789062 18.23777961730957
2 -98.92723083496094 18.202220916748047
3 -98.90416717529297 18.12527084350586
4 -98.83168029785156 18.13443946838379
5 -98.7630615234375 18.011390686035156
6 -98.61695861816406 17.973329544067383
7 -98.44862365722656 17.994159698486328
8 -98.3477783203125 17.89221954345703
9 -98.32112121582031 17.865829467773438
10 -98.37973022460938 17.688610076904297
11 -98.37973022460938 17.532499313354492
12 -98.30333709716797 17.411659240722656
13 -98.29084777832031 17.248329162597656
14 -98.07501220703125 17.112499237060547
15 -98.01334381103516 17.041109085083008
16 -98.05667114257812 16.884159088134766
17 -98.08168029785156 16.760000228881836
18 -98.16779327392578 16.700830459594727
19 -98.24037170410156 16.702669143676758
20 -98.20612335205078 16.645280838012695
21 -98.32972717285156 16.545000076293945
22 -98.32805633544922 16.405000686645508
23 -98.46861267089844 16.383329391479492
24 -98.5546875 16.31934356689453
25 -98.7822265625 16.553054809570312
26 -98.86473083496094 16.52444076538086
27 -99.03695678710938 16.596942901611328
28 -99.69223022460938 16.708332061767578
29 -99.8497314453125 16.787498474121094
30 -99.83778381347656 16.814441680908203
31 -99.87806701660156 16.870277404785156
32 -99.90000915527344 16.82555389404297
33 -99.93890380859375 16.882221221923828
34 -100.07833862304688 16.941665649414062
35 -100.1864013671875 16.955554962158203
36 -101.0486068725586 17.26694107055664
37 -101.10444641113281 17.35888671875
38 -101.41779327392578 17.51860809326172
39 -101.49778747558594 17.621944427490234
40 -101.55612182617188 17.617774963378906
41 -101.633056640625 17.666942596435547
42 -101.78778076171875 17.876110076904297
43 -101.95001220703125 17.977497100830078
44 -102.04723358154297 17.988887786865234
45 -102.14444732666016 17.919166564941406
46 -102.18085479736328 17.921894073486328
47 -102.14610290527344 18.17416000366211
48 -101.98750305175781 18.202220916748047
49 -101.90029907226562 18.261390686035156
50 -101.8631820678711 18.29004669189453
51 -101.87773895263672 18.537405014038086
52 -101.84447479248047 18.59560775756836
53 -101.61998748779297 18.60807991027832
54 -101.57425689697266 18.524932861328125
55 -101.50981903076172 18.48543930053711
56 -101.45161437988281 18.479204177856445
57 -101.29560089111328 18.53360939025879
58 -101.0875015258789 18.501110076904297
59 -101.01110076904297 18.51721954345703
60 -100.94670104980469 18.441940307617188
61 -100.9094009399414 18.450000762939453
62 -100.91529846191406 18.477500915527344
63 -100.79309844970703 18.472219467163086
64 -100.68560028076172 18.387500762939453
65 -100.62449645996094 18.353330612182617
66 -100.59359741210938 18.402240753173828
67 -100.72059631347656 18.525550842285156
68 -100.76969909667969 18.791109085083008
69 -100.72810363769531 18.86018943786621
70 -100.68260192871094 18.786069869995117
71 -100.5864028930664 18.85972023010254
72 -100.52780151367188 18.844449996948242
73 -100.45500183105469 18.809999465942383
74 -100.38610076904297 18.528329849243164
75 -100.30560302734375 18.39082908630371
76 -100.2594985961914 18.397499084472656
77 -100.12249755859375 18.516109466552734
78 -100.09329986572266 18.607219696044922
79 -99.88863372802734 18.657779693603516
80 -99.79583740234375 18.63389015197754
81 -99.65055847167969 18.764999389648438
82 -99.49653625488281 18.666709899902344
83 -99.31168365478516 18.463329315185547
84 -99.25639343261719 18.459720611572266
85 -99.227783203125 18.526660919189453
86 -99.1494369506836 18.533889770507812
87 -99.05049133300781 18.370790481567383
19
0 -96.7506332397461 18.430830001831055
1 -96.63555908203125 18.522220611572266
2 -96.67417907714844 18.681110382080078
3 -96.40583801269531 18.541109085083008
4 -96.35694885253906 18.38861083984375
5 -96.25473022460938 18.29166030883789
6 -96.23668670654297 18.21392059326172
7 -96.20529174804688 18.180269241333008
8 -96.16139221191406 18.185550689697266
9 -96.15028381347656 18.14167022705078
10 -96.09465026855469 18.164079666137695
11 -95.86222839355469 18.119159698486328
12 -95.80445861816406 18.051389694213867
13 -95.79945373535156 17.941110610961914
14 -95.91555786132812 17.779159545898438
15 -95.790283203125 17.52471923828125
16 -95.72501373291016 17.501659393310547
17 -95.56001281738281 17.533050537109375

18 -95.43917846679688 17.6330509185791
19 -95.36418151855469 17.641389846801758
20 -95.21028137207031 17.73332977294922
21 -95.20558166503906 17.648189544677734
22 -95.251708984375 17.594770431518555
23 -95.06916809082031 17.346660614013672
24 -95.00110626220703 17.335830688476562
25 -94.96721649169922 17.22249984741211
26 -94.32805633544922 17.172779083251953
27 -93.87342834472656 17.150279998779297
28 -93.86805725097656 17.01222038269043
29 -93.90528869628906 17.013050079345703
30 -93.9091796875 16.881940841674805
31 -94.04139709472656 16.80055046081543
32 -94.03584289550781 16.65304946899414
33 -94.12306213378906 16.510000228881836
34 -94.03639221191406 16.2833309173584
35 -94.08343505859375 16.150949478149414
36 -94.13667297363281 16.226943969726562
37 -94.20529174804688 16.196941375732422
38 -94.2952880859375 16.21999740600586
39 -94.36805725097656 16.29444122314453
40 -94.42445373535156 16.278888702392578
41 -94.41612243652344 16.20055389404297
42 -94.2711181640625 16.13416290283203
43 -94.3416748046875 16.176387786865234
44 -94.22195434570312 16.162220001220703
45 -94.18167114257812 16.11888885498047
46 -94.09017181396484 16.095170974731445
47 -94.06806945800781 16.08944320678711
48 -93.96250915527344 15.996386528015137
49 -94.06277465820312 16.03664779663086
50 -94.39556884765625 16.170276641845703
51 -94.52890014648438 16.18722152709961
52 -94.7247314453125 16.19666290283203
53 -94.6158447265625 16.258052825927734
54 -94.57806396484375 16.31833267211914
55 -94.667236328125 16.361942291259766
56 -94.78973388671875 16.257774353027344
57 -94.80862426757812 16.286388397216797
58 -94.77305603027344 16.33194351196289
59 -94.86195373535156 16.42749786376953
60 -95.06723022460938 16.27471923828125
61 -94.87083435058594 16.251941680908203
62 -94.83500671386719 16.283607482910156
63 -94.83168029785156 16.256107330322266
64 -94.93167114257812 16.240833282470703
65 -94.77833557128906 16.224720001220703
66 -94.7569580078125 16.194164276123047
67 -95.135009765625 16.202499389648438
68 -95.14500427246094 16.164718627929688
69 -95.22029113769531 16.14971923828125
70 -95.35917663574219 16.056110382080078
71 -95.36639404296875 16.01305389404297
72 -95.4202880859375 15.97805404663086
73 -95.9444580078125 15.818609237670898
74 -96.18194580078125 15.691665649414062
75 -96.43611145019531 15.688610076904297
76 -96.47611999511719 15.643610000610352
77 -96.83944702148438 15.727499008178711
78 -97.19667053222656 15.913331985473633
79 -97.78500366210938 15.968610763549805
80 -97.87112426757812 16.021385192871094
81 -97.87001037597656 16.06194305419922
82 -98.167236328125 16.196941375732422
83 -98.064453125 16.183887481689453
84 -98.09806823730469 16.21416473388672
85 -98.39889526367188 16.26138687133789
86 -98.5546875 16.31934356689453
87 -98.46861267089844 16.383329391479492
88 -98.32805633544922 16.405000686645508
89 -98.32972717285156 16.545000076293945
90 -98.20612335205078 16.645280838012695
91 -98.24037170410156 16.702669143676758
92 -98.16779327392578 16.700830459594727
93 -98.08168029785156 16.760000228881836
94 -98.05667114257812 16.884159088134766
95 -98.01334381103516 17.041109085083008
96 -98.07501220703125 17.112499237060547
97 -98.29084777832031 17.248329162597656
98 -98.30333709716797 17.411659240722656
99 -98.37973022460938 17.532499313354492
100 -98.37973022460938 17.688610076904297
101 -98.32112121582031 17.865829467773438
102 -98.3477783203125 17.89221954345703
103 -98.30972290039062 17.923049926757812
104 -98.2469482421875 17.909719467163086
105 -98.1591796875 18.024999618530273
106 -97.94277954101562 18.032779693603516
107 -97.92262268066406 17.998790740966797
108 -97.84445190429688 17.924999237060547
109 -97.7388916015625 17.992769241333008
110 -97.79640197753906 18.172779083251953
111 -97.71917724609375 18.30944061279297
112 -97.64806365966797 18.34054946899414
113 -97.6138916015625 18.2933292388916
114 -97.64140319824219 18.172779083251953
115 -97.44944763183594 17.977779388427734
116 -97.36974334716797 18.102779388427734
117 -97.28140258789062 18.16027069091797
118 -97.20668029785156 18.179439544677734
119 -97.08000183105469 18.138330459594727
120 -96.96278381347656 18.150550842285156
121 -96.88667297363281 18.240829467773438
122 -96.78751373291016 18.284719467163086
123 -96.72638702392578 18.385000228881836
124 -96.7506332397461 18.430830001831055
20
0 -93.58940124511719 17.375459671020508
1 -93.66751098632812 17.452770233154297
2 -93.65390014648438 17.522499084472656
3 -93.68936157226562 17.561370849609375
4 -93.74195861816406 17.683610916137695
5 -93.8577880859375 17.722219467163086
6 -93.86473083496094 17.749719619750977
7 -93.92388916015625 17.745830535888672
8 -93.96806335449219 17.831390380859375
9 -94.07528686523438 17.880550384521484
10 -94.0513916015625 17.99165916442871
11 -94.09278869628906 18.0686092376709
12 -94.09445190429688 18.15583038330078
13 -94.13782501220703 18.209142684936523
14 -93.86750793457031 18.303333282470703
15 -93.88667297363281 18.254165649414062
16 -93.79389953613281 18.261943817138672
17 -93.73918151855469 18.33361053466797
18 -93.58000183105469 18.352218627929688
19 -93.57028198242188 18.40777587890625
20 -93.84361267089844 18.311664581298828
21 -93.57806396484375 18.423053741455078
22 -93.15333557128906 18.439163208007812
23 -93.17222595214844 18.37277603149414
24 -93.11778259277344 18.38805389404297
25 -93.126953125 18.338886260986328
26 -93.08778381347656 18.403610229492188
27 -93.13890075683594 18.432220458984375
28 -92.92611694335938 18.445831298828125
29 -92.76722717285156 18.539997100830078
30 -92.70361328125 18.583885192871094
31 -92.66889953613281 18.42916488647461
32 -92.66056823730469 18.551666259765625
33 -92.68722534179688 18.619441986083984
34 -92.47828674316406 18.651676177978516
35 -92.42167663574219 18.513050079345703
36 -92.15306091308594 18.511940002441406
37 -92.15779113769531 18.1572208404541
38 -91.97944641113281 18.017780303955078
39 -91.85529327392578 17.95138931274414
40 -91.62640380859375 17.950830459594727
41 -91.60917663574219 18.096660614013672
42 -91.45390319824219 18.09943962097168
43 -91.32112121582031 18.063329696655273
44 -91.18860626220703 17.976110458374023
45 -90.98306274414062 17.9677791595459
46 -90.982421875 17.82065200805664
47 -90.98390197753906 17.256107330322266
48 -91.4375 17.241107940673828
49 -91.3880615234375 17.32666015625
50 -91.4263916015625 17.386940002441406
51 -91.4961166381836 17.404159545898438
52 -91.50758361816406 17.469900131225586
53 -91.665283203125 17.505550384521484
54 -91.66416931152344 17.64472007751465
55 -91.69862365722656 17.714160919189453
56 -91.79361724853516 17.726669311523438
57 -91.77287292480469 17.77368927001953
58 -91.78666687011719 17.85555076599121
59 -91.82333374023438 17.88916015625
60 -91.91251373291016 17.88722038269043
61 -91.945556640625 17.854719161987305
62 -91.9505615234375 17.895549774169922
63 -91.98989868164062 17.912220001220703
64 -92.07778930664062 17.832780838012695
65 -92.06861877441406 17.788610458374023
66 -92.15363311767578 17.789440155029297
67 -92.36583709716797 17.71693992614746
68 -92.38667297363281 17.667770385742188
69 -92.76167297363281 17.36166000366211
70 -92.83222961425781 17.40443992614746
71 -92.90556335449219 17.5302791595459
72 -92.9861068725586 17.544719696044922
73 -93.0130615234375 17.730270385742188
74 -92.99474334716797 17.91888999938965
75 -93.14222717285156 17.9436092376709
76 -93.2641830444336 17.991390228271484
77 -93.3094482421875 17.959720611572266
78 -93.39195251464844 17.60778045654297
79 -93.52667236328125 17.508609771728516
80 -93.58940124511719 17.375459671020508
21
0 -91.4375 17.241107940673828
1 -91.352783203125 17.176387786865234
2 -91.27723693847656 17.178333282470703
3 -91.1844482421875 17.035552978515625
4 -91.11834716796875 17.010276794433594
5 -91.06278991699219 16.902774810791016
6 -90.98583984375 16.868053436279297
7 -90.98333740234375 16.903053283691406
8 -90.95529174804688 16.899166107177734
9 -90.96612548828125 16.871944427490234
10 -90.92111206054688 16.82971954345703
11 -90.8033447265625 16.804996490478516
12 -90.71427917480469 16.726459503173828
13 -90.65806579589844 16.6441650390625
14 -90.665283203125 16.583053588867188
15 -90.63166809082031 16.58138656616211
16 -90.64500427246094 16.5191650390625
17 -90.61111450195312 16.510555267333984
18 -90.63417053222656 16.48388671875
19 -90.54640197753906 16.484996795654297
20 -90.48167419433594 16.458332061767578
21 -90.4808349609375 16.426109313964844
22 -90.39584350585938 16.41555404663086
23 -90.37889099121094 16.365276336669922
24 -90.4202880859375 16.359722137451172
25 -90.39111328125 16.340831756591797
26 -90.45945739746094 16.252777099609375
27 -90.4364013671875 16.237220764160156
28 -90.45973205566406 16.19083023071289
29 -90.42750549316406 16.167499542236328
30 -90.46084594726562 16.106388092041016
31 -90.43167114257812 16.102218627929688
32 -90.45806884765625 16.074718475341797
33 -91.72917175292969 16.074996948242188
34 -92.21139526367188 15.26222038269043
35 -92.06584167480469 15.077775955200195
36 -92.14973449707031 14.994443893432617
37 -92.14167785644531 14.897499084472656
38 -92.18583679199219 14.843610763549805
39 -92.15444946289062 14.675832748413086
40 -92.18722534179688 14.588333129882812
41 -92.24678039550781 14.550546646118164
42 -92.79833984375 15.105554580688477
43 -92.84445190429688 15.170555114746094
44 -92.77528381347656 15.15194320678711
45 -92.743896484375 15.086666107177734
46 -92.76918029785156 15.171110153198242
47 -92.84556579589844 15.208887100219727
48 -92.85139465332031 15.181665420532227
49 -92.97611999511719 15.259164810180664
50 -93.19416809082031 15.481388092041016
51 -93.54640197753906 15.759721755981445
52 -93.92889404296875 15.994720458984375
53 -93.92945861816406 16.016109466552734
54 -93.8819580078125 15.99860954284668
55 -93.8558349609375 16.021663665771484
56 -93.89418029785156 16.08777618408203
57 -94.07362365722656 16.136943817138672
58 -94.08343505859375 16.150949478149414
59 -94.03639221191406 16.2833309173584
60 -94.12306213378906 16.510000228881836
61 -94.03584289550781 16.65304946899414
62 -94.04139709472656 16.80055046081543
63 -93.9091796875 16.881940841674805
64 -93.90528869628906 17.013050079345703
65 -93.86805725097656 17.01222038269043
66 -93.87342834472656 17.150279998779297
67 -93.626953125 17.3075008392334
68 -93.58940124511719 17.375459671020508
69 -93.52667236328125 17.508609771728516
70 -93.39195251464844 17.60778045654297
71 -93.3094482421875 17.959720611572266
72 -93.2641830444336 17.991390228271484
73 -93.14222717285156 17.9436092376709
74 -92.99474334716797 17.91888999938965
75 -93.0130615234375 17.730270385742188
76 -92.9861068725586 17.544719696044922
77 -92.90556335449219 17.5302791595459
78 -92.83222961425781 17.40443992614746
79 -92.76167297363281 17.36166000366211
80 -92.38667297363281 17.667770385742188
81 -92.36583709716797 17.71693992614746
82 -92.15363311767578 17.789440155029297
83 -92.06861877441406 17.788610458374023
84 -92.07778930664062 17.832780838012695
85 -91.98989868164062 17.912220001220703
86 -91.9505615234375 17.895549774169922
87 -91.945556640625 17.854719161987305
88 -91.91251373291016 17.88722038269043
89 -91.82333374023438 17.88916015625
90 -91.78666687011719 17.85555076599121
91 -91.77287292480469 17.77368927001953
92 -91.79361724853516 17.726669311523438
93 -91.69862365722656 17.714160919189453
94 -91.66416931152344 17.64472007751465
95 -91.665283203125 17.505550384521484
96 -91.50758361816406 17.469900131225586
97 -91.4961166381836 17.404159545898438
98 -91.4263916015625 17.386940002441406
99 -91.3880615234375 17.32666015625
100 -91.4375 17.241107940673828
22
0 -112.29640197753906 28.756385803222656
1 -112.56417846679688 28.87833023071289
2 -112.48638916015625 28.961387634277344
3 -112.4969482421875 29.06583023071289
4 -112.45694732666016 29.186107635498047
5 -112.26862335205078 29.2530517578125
6 -112.2641830444336 29.146942138671875
7 -112.20084381103516 28.981666564941406
8 -112.26445007324219 28.811664581298828
9 -112.24806213378906 28.781665802001953
10 -112.29640197753906 28.756385803222656
11 -109.04500579833984 31.343347549438477
12 -108.82730865478516 31.34345054626465
13 -108.89140319824219 31.19222068786621
14 -108.83609771728516 31.156940460205078
15 -108.79810333251953 31.204999923706055
16 -108.73500061035156 30.63249969482422
17 -108.67780303955078 30.575830459594727
18 -108.55809783935547 29.992769241333008
19 -108.61360168457031 29.40110969543457
20 -108.70780181884766 29.400829315185547
21 -108.6239013671875 28.77111053466797
22 -108.68949890136719 28.69639015197754
23 -108.56559753417969 28.289169311523438
24 -108.65249633789062 28.21221923828125
25 -108.88079833984375 28.298330307006836
26 -108.97699737548828 28.30554962158203
27 -109.02200317382812 28.275829315185547
28 -109.05580139160156 28.29916000366211
29 -109.14640045166016 28.175830841064453
30 -108.91390228271484 27.78499984741211
31 -108.80970001220703 27.711669921875
32 -108.7750015258789 27.59943962097168
33 -108.6552963256836 27.520000457763672
34 -108.6675033569336 27.404159545898438
35 -108.63639831542969 27.31916046142578
36 -108.66529846191406 27.151939392089844
37 -108.60420227050781 27.03805923461914
38 -108.47129821777344 26.96133041381836
39 -108.47699737548828 26.865829467773438
40 -108.48639678955078 26.831939697265625
41 -109.14354705810547 26.338422775268555
42 -109.15834045410156 26.376388549804688
43 -109.24833679199219 26.332775115966797
44 -109.24111938476562 26.448055267333984
45 -109.27694702148438 26.535831451416016
46 -109.47334289550781 26.68694305419922
47 -109.4344482421875 26.703887939453125
48 -109.50556945800781 26.729442596435547
49 -109.51750183105469 26.764999389648438
50 -109.56668090820312 26.73360824584961
51 -109.50890350341797 26.682220458984375
52 -109.62750244140625 26.702774047851562
53 -109.69862365722656 26.675277709960938
54 -109.80667114257812 26.736385345458984
55 -109.94695281982422 26.986385345458984
56 -109.88223266601562 26.941665649414062
57 -109.96250915527344 27.105274200439453
58 -110.06333923339844 27.09583282470703
59 -110.31472778320312 27.155277252197266
60 -110.41472625732422 27.264163970947266
61 -110.5097427368164 27.299163818359375
62 -110.44640350341797 27.311664581298828
63 -110.49028015136719 27.38416290283203
64 -110.55388641357422 27.367496490478516
65 -110.57721710205078 27.533885955810547
66 -110.63583374023438 27.656108856201172
67 -110.56916809082031 27.67888641357422
68 -110.60166931152344 27.678607940673828
69 -110.59140014648438 27.719165802001953
70 -110.54444885253906 27.738052368164062
71 -110.60111999511719 27.74777603149414
72 -110.60945129394531 27.822776794433594
73 -110.51611328125 27.840831756591797
74 -110.5097427368164 27.86638641357422
75 -110.60362243652344 27.887218475341797
76 -110.61000061035156 27.859996795654297
77 -110.77555847167969 27.917499542236328
78 -110.8477783203125 27.905277252197266
79 -110.81001281738281 27.924720764160156
80 -110.85028076171875 27.986385345458984
81 -110.89167785644531 27.895832061767578
82 -110.85917663574219 27.89444351196289
83 -110.87918090820312 27.835552215576172
84 -110.99501037597656 27.967498779296875
85 -111.10111999511719 27.93555450439453
86 -111.2388916015625 28.05638885498047
87 -111.45668029785156 28.327499389648438
88 -111.43583679199219 28.379165649414062
89 -111.695556640625 28.4647216796875
90 -111.76334381103516 28.588333129882812
91 -111.9472427368164 28.76166534423828
92 -111.90444946289062 28.783885955810547
93 -111.86361694335938 28.74721908569336
94 -111.85529327392578 28.799720764160156
95 -111.96890258789062 28.833053588867188
96 -112.11029052734375 28.963607788085938
97 -112.16472625732422 28.97249984741211
98 -112.16612243652344 29.134719848632812
99 -112.22306823730469 29.18499755859375
100 -112.21056365966797 29.30194091796875
101 -112.28861999511719 29.335277557373047
102 -112.3364028930664 29.32332992553711
103 -112.33944702148438 29.292774200439453
104 -112.39167785644531 29.32971954345703
105 -112.4102783203125 29.38027572631836
106 -112.37722778320312 29.500553131103516
107 -112.57778930664062 29.713333129882812
108 -112.66361999511719 29.90027618408203
109 -112.74305725097656 29.916942596435547
110 -112.75611877441406 30.20861053466797
111 -112.8183364868164 30.27694320678711
112 -112.86083984375 30.278610229492188
113 -112.84278869628906 30.342777252197266
114 -112.87167358398438 30.431663513183594
115 -113.08139038085938 30.69888687133789
116 -113.11723327636719 30.814441680908203
117 -113.08056640625 30.949443817138672
118 -113.12029266357422 31.068607330322266
119 -113.06806945800781 31.0
120 -113.042236328125 31.173885345458984
121 -113.0966796875 31.232219696044922
122 -113.14222717285156 31.230552673339844
123 -113.10611724853516 31.202774047851562
124 -113.13639831542969 31.199718475341797
125 -113.2388916015625 31.28888702392578
126 -113.27166748046875 31.26694107055664
127 -113.21778869628906 31.24388885498047
128 -113.25306701660156 31.241107940673828
129 -113.63583374023438 31.349441528320312
130 -113.61167907714844 31.35277557373047
131 -113.63834381103516 31.496944427490234
132 -113.88555908203125 31.60916519165039
133 -113.9433364868164 31.60138702392578
134 -113.96000671386719 31.660274505615234
135 -113.98194885253906 31.572498321533203
136 -113.945556640625 31.56805419921875
137 -113.99057006835938 31.517776489257812
138 -114.04667663574219 31.492774963378906
139 -114.16972351074219 31.504444122314453
140 -114.58389282226562 31.76082992553711
141 -114.69695281982422 31.767776489257812
142 -115.01789855957031 31.94748878479004
143 -114.9989013671875 32.1361083984375
144 -115.04149627685547 32.254669189453125
145 -114.96420288085938 32.36861038208008
146 -114.93669891357422 32.47304916381836
147 -114.81930541992188 32.50444793701172
148 -114.82096862792969 32.48711395263672
149 -113.32837677001953 32.04356002807617
150 -111.36886596679688 31.431438446044922
151 -111.07131958007812 31.335535049438477
152 -110.45194244384766 31.33755874633789
153 -109.04500579833984 31.343347549438477
23
0 -108.47129821777344 26.96133041381836
1 -108.60420227050781 27.03805923461914
2 -108.66529846191406 27.151939392089844
3 -108.63639831542969 27.31916046142578
4 -108.6675033569336 27.404159545898438
5 -108.6552963256836 27.520000457763672
6 -108.7750015258789 27.59943962097168
7 -108.80970001220703 27.711669921875
8 -108.91390228271484 27.78499984741211
9 -109.14640045166016 28.175830841064453
10 -109.05580139160156 28.29916000366211
11 -109.02200317382812 28.275829315185547
12 -108.97699737548828 28.30554962158203
13 -108.88079833984375 28.298330307006836
14 -108.65249633789062 28.21221923828125
15 -108.56559753417969 28.289169311523438
16 -108.68949890136719 28.69639015197754
17 -108.6239013671875 28.77111053466797
18 -108.70780181884766 29.400829315185547
19 -108.61360168457031 29.40110969543457
20 -108.55809783935547 29.992769241333008
21 -108.67780303955078 30.575830459594727
22 -108.73500061035156 30.63249969482422
23 -108.79810333251953 31.204999923706055
24 -108.83609771728516 31.156940460205078
25 -108.89140319824219 31.19222068786621
26 -108.82730865478516 31.34345054626465
27 -108.2100601196289 31.34374237060547
28 -108.2026596069336 31.78680419921875
29 -107.28299713134766 31.78498077392578
30 -106.53897094726562 31.786197662353516
31 -106.38304138183594 31.733762741088867
32 -106.21275329589844 31.47812843322754
33 -105.99835968017578 31.393817901611328
34 -105.76972961425781 31.170780181884766
35 -105.60321807861328 31.086427688598633
36 -105.55438232421875 30.9982852935791
37 -105.40906524658203 30.902509689331055
38 -105.39031219482422 30.85308074951172
39 -105.31378173828125 30.81650733947754
40 -105.28759765625 30.83194923400879
41 -105.25818634033203 30.797653198242188
42 -105.21434020996094 30.81208610534668
43 -105.06056213378906 30.687870025634766
44 -104.99754333496094 30.68433380126953
45 -104.98693084716797 30.641324996948242
46 -104.89067840576172 30.570556640625
47 -104.85299682617188 30.392263412475586
48 -104.80647277832031 30.376447677612305
49 -104.81395721435547 30.3504695892334
50 -104.7026138305664 30.238489151000977
51 -104.67475891113281 30.148963928222656
52 -104.69649505615234 30.057302474975586
53 -104.67436981201172 29.909282684326172
54 -104.57756042480469 29.80793571472168
55 -104.53524780273438 29.679466247558594
56 -104.37759399414062 29.55061149597168
57 -104.2047348022461 29.484041213989258
58 -104.16438293457031 29.400714874267578
59 -104.0456314086914 29.3281192779541
60 -103.78699493408203 29.26725959777832

61 -103.76776123046875 29.281240463256836
62 -103.78215789794922 29.229795455932617
63 -103.73985290527344 29.230348587036133
64 -103.7203140258789 29.190631866455078
65 -103.52623748779297 29.146646499633789
66 -103.47407531738281 29.072134017944336
67 -103.37545013427734 29.032108306884766
68 -103.33551788330078 29.050338745117188
69 -103.29014587402344 28.997732162475586
70 -103.95469665527344 27.87055015563965
71 -103.63089752197266 26.66107940673828
72 -103.8442001342774 26.728889465332031
73 -104.18830108642578 26.756389617919922
74 -104.55110168457031 26.35055923461914
75 -104.60749816894531 26.35555076599121
76 -104.72560119628906 26.450550079345703
77 -104.79669952392578 26.433330535888672
78 -104.84390258789062 26.492769241333008
79 -105.01000213623047 26.459440231323242
80 -105.12190246582031 26.521390914916992
81 -105.13829803466797 26.54138946533203
82 -105.32610321044922 26.458890914916992
83 -105.58529663085938 26.587779998779297
84 -105.63690185546875 26.66278076171875
85 -105.75389862060547 26.655000686645508
86 -106.02749633789062 26.838609695434570
87 -106.09190368652344 26.735000610351562
88 -106.12699890136719 26.769439697265625
89 -106.15329742431641 26.752220153808594
90 -106.17220306396484 26.59139060974121
91 -106.23950195312500 26.415000915527344
92 -106.34470367431641 26.3688907623291
93 -106.44999694824219 26.376390457153320
94 -106.36750030517578 26.147499084472656
95 -106.40309906005859 26.079999923706055
96 -106.52079772949219 26.021110534667969
97 -106.53359985351562 25.789169311523438
98 -106.74030303955078 25.622499465942383
99 -107.08409881591797 25.606090545654297
100 -107.15170288085938 25.775550842285156
101 -107.29969787597656 25.943330764770508
102 -107.36640167236328 26.115280151367188
103 -107.7844009399414 26.200279235839844
104 -107.84670257568359 26.639999389648438
105 -108.00389862060547 26.819719314575195
106 -108.03559875488281 26.947500228881836
107 -108.22059631347656 26.97277069091797
108 -108.24890136718750 27.040830612182617
109 -108.30529785156250 27.061389923095703
110 -108.40499877929688 27.030830338300781
111 -108.47129821777344 26.961330413818360
24
0 -103.29014587402344 28.997732162475586
1 -103.28034973144531 28.986373901367188
2 -103.26658630371094 29.007453918457031
3 -103.1534652709961 28.978681564331055
4 -102.98809814453125 29.190862655563965
5 -102.86617279052734 29.229036331176758
6 -102.9083251953125 29.269203186035156
7 -102.88301086425781 29.353370666503906
8 -102.82220458984375 29.411844253540044
9 -102.80472564697266 29.530145645141602
10 -102.67636108398438 29.744224548339844
11 -102.63761138916016 29.732337951660156
12 -102.57649993896484 29.778247833251953
13 -102.55194854736328 29.749500274658203
14 -102.50309753417969 29.785455570373535
15 -102.38479614257812 29.767946243286133
16 -102.36756134033203 29.845289230334668
17 -102.32433319091797 29.880115509033203
18 -102.06399536132812 29.784570693969727
19 -101.97332000732422 29.818773269653320
20 -101.92422485351562 29.788501739501953
21 -101.81909942626953 29.814125061035156
22 -101.80520629882812 29.779998779296875
23 -101.75909423828125 29.787166595458984
24 -101.63967132568359 29.756959915161133
25 -101.58148956298828 29.765150070190443
26 -101.54395294189453 29.810119628906250
27 -101.53834533691406 29.763017654418945
28 -101.47046661376953 29.788690567016602
29 -101.44842529296875 29.760585784912109
30 -101.40127563476562 29.769905090332031
31 -101.41609954833984 29.745433807373047
32 -101.36840057373047 29.657161712646484
33 -101.30586242675781 29.652431488037109
34 -101.30892944335938 29.580909729003906
35 -101.25458526611328 29.628749847412109
36 -101.26142883300781 29.526473999023438
37 -101.06735992431641 29.473552703857422
38 -101.00905601913086 29.373254776000977
39 -101.79698944091797 29.242502212524414
40 -101.76860893261719 29.166570663452148
41 -101.66876983642578 29.080072402954102
42 -101.64722442626953 28.922349929980957
43 -101.58979034423828 28.894222259521484
44 -101.49790954589844 28.660987854003906
45 -101.40317535400391 28.589733123779297
46 -101.41953277587891 28.544191360473633
47 -101.34580230712891 28.500810623168945
48 -101.37677001953125 28.478651046752930
49 -101.35157012939453 28.394182205200195
50 -101.29289245605469 28.320360183715820
51 -101.29792022705078 28.280353546142578
52 -101.22346965820312 28.241456985473633
53 -101.21407318115234 28.201934814453125
54 -101.09692382812500 28.154281616210938
55 -99.99330902099609 28.003459930419922
56 -99.94185638427734 27.986881256103516
57 -99.87473297111914 27.797685623168945
58 -99.81572723388672 27.780107498168945
59 -99.80781555175781 27.770837783813477
60 -99.97306823730469 27.635280606933594
61 -101.18250274658203 27.794170379638672
62 -101.31140136718750 27.709999084472656
63 -101.42829895019531 27.400829315185547
64 -101.58499908447266 27.395000457763672
65 -101.82279968261719 27.235279083251953
66 -101.79530334472656 27.026390075683594
67 -101.75920104980469 27.046670913696289
68 -101.69999694824219 27.010280609130859
69 -101.65859985351562 27.071109771728516
70 -101.54969787597656 27.030550003051758
71 -101.53279876708984 26.867219924926758
72 -101.56559753417969 26.771940231323242
73 -101.61579895019531 26.750829696655273
74 -100.69470214843750 26.628330230371289
75 -100.79419708251953 26.707780838012695
76 -101.21859741210938 26.370550155563965
77 -101.03579711914062 26.149440765380859
78 -100.94940185546875 26.111389160156250
79 -100.91339874267578 26.056940078735352
80 -100.91590856933594 26.528329849243164
81 -100.82496643066406 25.918050765991211
82 -100.81970214843750 25.743890762329102
83 -100.70999908447266 25.611660003662109
84 -100.64219665527344 25.608329772949219
85 -100.63449859619141 25.553329467773438
86 -100.57199859619141 25.528329849243164
87 -100.57779693603516 25.498609542846680
88 -100.67449951171875 25.536390030456543
89 -100.69170379638672 25.489999771118164
90 -100.58170318603516 25.444160461425781
91 -100.44170379638672 25.327220916748047
92 -100.30280303955078 25.325000762939453
93 -100.18969726562500 25.275829315185547
94 -100.25920104980469 25.254999160766602
95 -100.19090270996094 25.190929412841797
96 -100.22920227050781 25.213630676269531
97 -100.37470245361328 25.157499313354492
98 -100.43720245361328 25.211940765380859
99 -100.54360198974609 25.228330612182617
100 -100.70860290527344 25.198610305786133
101 -100.77310180664062 25.155550003051758
102 -100.82530212402344 25.038890838623047
103 -100.69830322265625 24.931110382080078
104 -100.78720092773438 24.892780303955078
105 -100.82389831542969 24.559999465942383
106 -100.87220001220703 24.601390838623047
107 -100.99639892578125 24.589719772338867
108 -101.24220275878906 24.810279846191406
109 -101.32109832763672 24.778610229492188
110 -101.36029815673828 24.821109771728516
111 -101.44529724121094 24.761390686035156
112 -101.57969665527344 24.754440307617188
113 -101.61000061035156 24.788330078125
114 -101.58560180664062 24.858060836791992
115 -101.74639892578125 24.905830383330078
116 -101.83750152587891 25.026939392089844
117 -102.25720214843750 25.155550003051758
118 -102.66580200195312 25.118049621582031
119 -102.66690063476562 25.075830459594727
120 -102.82920074462891 24.861660003662109
121 -102.81140136718750 24.696659088134766
122 -102.95169830322266 24.798610682756348
123 -103.16030120849609 24.849720001220703
124 -103.23889923095703 24.904439926147461
125 -103.25920104980469 25.058889389038086
126 -103.39859771728516 25.150550842285156
127 -103.50469970703125 25.275960922241211
128 -103.42890167236328 25.333610534667969
129 -103.41280364990234 25.385280609130859
130 -103.48449707031250 25.465269088745117
131 -103.48470306396484 25.541940689086914
132 -103.32579803466797 25.743330001831055
133 -103.33609771728516 25.075550079345703
134 -103.27890014648438 26.284439086914062
135 -103.32250213623047 26.383890151977539
136 -103.63089752197266 26.661079460673828
137 -103.95469665527344 27.870550515563965
138 -103.29014587402344 28.997732162475586
25
0 -105.402099609375 23.067459106445312
1 -105.46279907226562 23.040830612182617
2 -105.48780059814453 22.971109390258789
3 -105.44419860839844 22.903890609074121
4 -105.54170227050781 22.836940765380859
5 -105.57530212402344 22.753049850463867
6 -105.47940063476562 22.681390762329102
7 -105.45449829101562 22.548889160156250
8 -105.64430236816406 22.486179351806641
9 -105.69470214843750 22.473110198974609
10 -105.7145767211914 22.468360900878906
11 -105.72057342529297 22.524440765380859
12 -105.75639343261719 22.532497406005859
13 -105.81696319580078 22.660274505615234
14 -106.00083923339844 22.816108703613281
15 -105.98390197753906 22.850555419921875
16 -106.02890014648438 22.827220916748047
17 -106.21833801269531 23.047496795654297
18 -106.37861633300781 23.184444427490234
19 -106.42472839355469 23.180553436279297
20 -106.52250671386719 23.401107788085938
21 -106.80223083496094 23.647499084472656
22 -106.91916656494141 23.868888854980469
23 -107.37473297111914 24.203609466552734
24 -107.39639282226562 24.248607635498047
25 -107.79583740234375 24.493888854980469
26 -107.77971649169922 24.515552520751953
27 -107.49741363525391 24.340015411376953
28 -107.55196380615234 24.380275726318359
29 -107.49501037597656 24.356941223144531
30 -107.47528076171875 24.392776489257812
31 -107.52500915527344 24.520275111596797
32 -107.58528137207031 24.521942138671875
33 -107.59557342529297 24.5
34 -107.55333709716797 24.498332977294922
35 -107.63583374023438 24.451663970947266
36 -107.67001342773438 24.495552062988281
37 -107.74166870117188 24.498607635498047
38 -107.80832672111914 24.587219238281250
39 -107.93472290039062 24.635829925537109
40 -107.81140136718750 24.525276184082031
41 -107.99223327636719 24.645275115966797
42 -108.06001281738281 24.778053283691406
43 -107.99195861816406 24.75
44 -107.97445678710938 24.768608093261719
45 -108.01055908203125 24.835277557373047
46 -107.98946380615234 24.961940765380859
47 -108.05029296875 24.998886108398438
48 -108.04055786132812 24.831108093261719
49 -108.10138702392578 24.819442749023438
50 -108.22862243652344 25.027496337890625
51 -108.32613372802734 25.098609924316406
52 -108.27863311767578 25.102497100830078
53 -108.17916870117188 24.980831146240234
54 -108.12750244140625 24.972499847412109
55 -108.13223266601562 25.017776489257812
56 -108.1633529663086 25.028053283691406
57 -108.14279174804688 25.056941986083984
58 -108.10057067871094 25.014163970947266
59 -107.99973297111914 25.004444122314453
60 -108.05805969238281 25.086940765380859
61 -108.12583923339844 25.123886108398438
62 -108.1672363281250 25.108055114746094
63 -108.21917724609375 25.169998168945312
64 -108.35250854492188 25.167221069335938
65 -108.35582733154297 25.203330993652344
66 -108.3144531250 25.186664581298828
67 -108.31696319580078 25.241386413574219
68 -108.36306762695312 25.263053894042969
69 -108.39862060546875 25.142776489257812
70 -108.39362335205078 25.205829620361328
71 -108.43695068359375 25.263053894042969
72 -108.72805786132812 25.355274200439453
73 -108.59068298339844 25.344528198242188
74 -108.65278625488281 25.394443511196289
75 -108.77027893066406 25.379444122314453
76 -108.72666931152344 25.401943206787109
77 -108.74722290039062 25.442497253417969
78 -108.77555847167969 25.431941986083984
79 -108.76779174804688 25.542221069335938
80 -108.89890289306641 25.561386108398438
81 -108.87779235839844 25.507221221923828
82 -108.92001342773438 25.456108093261719
83 -108.94555664062500 25.498886108398438
84 -109.01222229003906 25.495655206298828
85 -109.02944946289062 25.459999084472656
86 -109.10890197753906 25.526107788085938
87 -109.05695343017578 25.577499389648438
88 -108.99667358398438 25.569442749023438
89 -108.97944641113281 25.536941528320312
90 -108.97138977050781 25.589443206787109
91 -108.87834167480469 25.670276641845703
92 -108.82805633544922 25.798332214355469
93 -108.90084838867188 25.694999694824219
94 -109.06777954101562 25.587776182373047
95 -109.13694763183594 25.578052520751953
96 -109.17223358154297 25.647777557373047
97 -109.25778198242188 25.679721832275391
98 -109.15695190429688 25.555274963378906
99 -109.24973297119141 25.630275726318359
100 -109.40611267089844 25.641109466552734
101 -109.30029296875 25.658885955810547
102 -109.28751373291016 25.708885192871094
103 -109.37361145019531 25.763610838623047
104 -109.39972686767578 25.678607940673828
105 -109.40834045410156 25.759441375732422
106 -109.44333648681641 25.790275573730469
107 -109.41674804687500 25.860286712646484
108 -109.43028259277344 26.014442443847656
109 -109.25639343261719 26.306663513183594
110 -109.28527832031250 25.763610838623047
111 -109.21473693847656 26.339164733886719
112 -109.16528320312500 25.325275421142578
113 -109.15278625488281 26.276943206787109
114 -109.17529296875 26.264720916748047
115 -109.10111999511719 26.209163665771484
116 -109.08194732666016 26.281665820019531
117 -109.13139343261719 26.307220458984375
118 -109.14354705810547 26.338422775268555
119 -108.48639678955078 26.831939697265625
120 -108.47699737548828 26.865829467773438
121 -108.47129821777344 26.961330413818360
122 -108.40499877929688 27.030830338300781
123 -108.30529785156250 27.061389923095703
124 -108.24890136718750 27.040830612182617
125 -108.22059631347656 26.972770690917969
126 -108.03559875488281 26.947500228881836
127 -108.00389862060547 26.819719314575195
128 -107.84670257568359 26.639999389648438
129 -107.78440093994141 26.200279235839844
130 -107.36640167236328 26.115280151367188
131 -107.29969787597656 25.943330764770508
132 -107.15170288085938 25.775550842285156
133 -107.08409881591797 25.606090545654297
134 -107.12470245361328 25.944393157959
135 -107.10749816894531 25.148609161376953
136 -106.94889831542969 24.841939926147461
137 -106.88500213623047 24.770280838012695
138 -106.82109832763672 24.763330459594727
139 -106.63860321044922 24.572500228881836
140 -106.51920318603516 24.301940917968750
141 -106.39610290527344 24.284719467163086
142 -106.24919891357422 24.390270233154297
143 -106.00219726562500 24.211940765380859
144 -105.95970153808594 24.098890304565430
145 -105.90809613347656 24.055269241333008
146 -105.88580322265625 23.759719848632812
147 -105.76329803466797 23.566659698486328
148 -105.71640014648438 23.469720804541016
149 -105.68419647216797 23.287220001220703
150 -105.63390350341797 23.253440856933594
151 -105.52890014648438 23.144439697265625
152 -105.41670227050781 23.147220611572266
153 -105.40209960937500 23.067459106445312
26
0 -104.31140013671875 22.319210052490234
1 -104.34500122070312 22.451389312744141
2 -104.49140167236328 22.410270769091797
3 -104.61250305175781 22.472499847412109
4 -104.66059875488281 22.624439239501953
5 -104.75749969482422 22.676670074462891
6 -104.99810028076172 22.548330307006836
7 -104.99810028076172 22.678890228271484
8 -104.87999725341797 22.783050537109375
9 -104.91470336914062 22.924720764160156
10 -105.17199707031250 23.039720535278320
11 -105.3125 23.034719467163086
12 -105.40209960937500 23.067459106445312
13 -105.41670227050781 23.147220611572266
14 -105.52890014648438 23.144439697265625
15 -105.63390350341797 23.253440856933594
16 -105.68419647216797 23.287220001220703
17 -105.71640014648438 23.469720804541016
18 -105.76329803466797 23.566659698486328
19 -105.88580322265625 23.759719848632812
20 -105.90809613347656 24.055269241333008
21 -105.95970153808594 24.098890304565430
22 -106.00219726562500 24.211940765380859
23 -106.24919891357422 24.390270233154297
24 -106.39610290527344 24.284719467163086
25 -106.51920318603516 24.301940917968750
26 -106.63860321044922 24.572500228881836
27 -106.82109832763672 24.763330459594727
28 -106.88500213623047 24.770280838012695
29 -106.94889831542969 24.841939926147461
30 -107.10749816894531 25.148609161376953
31 -107.12470245361328 25.294439315795959
32 -107.08409881591797 25.606090545654297
33 -106.74030303955078 25.622499465942383
34 -106.53359985351562 25.789169311523438
35 -106.52079772949219 26.021110534667969
36 -106.40309906005859 26.079999923706055
37 -106.36750030517578 26.147499084472656
38 -106.44999694824219 26.376390457153320
39 -106.34470367431641 26.368890762329102
40 -106.23950195312500 26.415000915527344
41 -106.17220306396484 26.752220153808594
42 -106.15329742431641 26.752220153808594
43 -106.12699890136719 26.769439697265625
44 -106.09190368652344 26.735000610351562
45 -106.02749633789062 26.838609695434570
46 -105.75389862060547 26.655000686645508
47 -105.63690185546875 26.662780761718750
48 -105.58529663085938 26.587779998779297
49 -105.32610321044922 26.458890914916992
50 -105.13829803466797 26.541389465332031
51 -105.12190246582031 26.521390914916992
52 -105.01000213623047 26.459440231323242
53 -104.84390258789062 26.492769241333008
54 -104.79669952392578 26.433330535888672
55 -104.72560119628906 26.450550079345703
56 -104.60749816894531 26.355592346191406
57 -104.55110168457031 26.355592346191406
58 -104.18830108642578 26.756389617919922
59 -103.84420013427734 26.728889465332031
60 -103.63089752197266 26.661079460673828
61 -103.32250213623047 26.383890151977554
62 -103.27890014648438 26.284439086914062
63 -103.33609771728516 26.075550079345703
64 -103.32579803466797 25.743330001831055
65 -103.48470306396484 25.541940689086914
66 -103.48449707031250 25.465269088745117
67 -103.41280364990234 25.385280609130859
68 -103.42890167236328 25.333610534667969
69 -103.41909942626953 25.274960994720459
70 -103.39859771728516 25.150550842285156
71 -103.25920104980469 25.058889389038086
72 -103.23889923095703 24.904439926147461
73 -103.16030120849609 24.849720001220703
74 -102.95169830322266 24.798610682756348
75 -102.81140136718750 24.696659088134766
76 -102.82920074462891 24.861660003662109
77 -102.66690063476562 25.075830459594727
78 -102.50499725341797 24.828609466552734
79 -102.51360321044922 24.452220916748047
80 -102.73529815673828 24.458890914916992
81 -102.76719665527344 24.433889390038086
82 -103.26750183105469 24.476110458374023
83 -103.61250305175781 24.275550842285156
84 -103.60060119628906 24.182500839233398
85 -103.85088974267578 24.073059082031250
86 -103.87560272216797 23.861389160156250
87 -103.85859680175781 23.736940383911133
88 -103.80829620361328 23.647207064160156
89 -103.91970062255586 23.623329162597656
90 -103.93669891357422 23.573059082031250
91 -104.07810211181641 23.447500228881836
92 -104.09609985351562 23.195829391479492
93 -104.16970062255586 23.142780303955078
94 -104.20110321044922 23.062770843505859
95 -104.25859832763672 22.921939849853516
96 -104.31140136718750 22.319210052490234
27
0 -101.52490234375 21.856639862060547
1 -101.80030059814453 22.015270233154297
2 -101.84619903564453 22.011760711669922
3 -101.93640136718750 22.114440917968750
4 -102.05639648437500 22.137779235839844
5 -102.02420043945312 22.251397705957031
6 -102.15440368652344 22.285270690917969
7 -102.15579986572266 22.324169158935547
8 -102.21920013427734 22.372219085693359
9 -102.27359771728516 22.355829238891602
10 -102.28720092773438 22.456390380859375
11 -102.32579803466797 22.458890914916992
12 -102.40559967041016 22.337219238281250
13 -102.63500123623047 22.278329849243164
14 -102.70690155029297 22.083330154418945
15 -102.84470367431641 21.930269241333008
16 -102.85169982910156 21.823329925537109
17 -102.74140167236328 21.724170684814453
18 -102.76969909667969 21.617769241333008
19 -102.63919830322266 21.546939849853516
20 -102.68720245361328 21.381940841674805
21 -102.83360290527344 21.320550195190430
22 -102.90670013427734 21.328609466552734
23 -102.96219635009766 21.284719467163086
24 -103.03440093994141 21.306659698486328
25 -103.08560180664062 21.1875
26 -103.05580139160156 21.054439544677734
27 -103.54280009332031 21.198059082031250
28 -103.60169982910156 21.188049316406250
29 -103.64610290527344 21.241939544677734
30 -103.73699951171875 21.203329086303711
31 -103.76560211181641 21.223890304565430
32 -103.75309753417969 21.251659393310547
33 -103.70279693603516 21.386940002441406
34 -103.73359680175781 21.516389846801758
35 -103.65029907226562 21.461389541625977
36 -103.51419830322266 21.593050003051758
37 -103.50949859619141 21.731939315795898
38 -103.54779815673828 21.785549163818359
39 -103.44699859619141 21.848049163818359
40 -103.39420318603516 21.933330535888672
41 -103.29309844970703 21.982500076293945
42 -103.17079925537109 21.975280767181775
43 -103.09110260009766 22.090269088745117
44 -103.12779998779297 22.147779464721680
45 -103.05560302734375 22.286109924316406
46 -103.20140075683594 22.307779312133789
47 -103.17949676513672 22.368610382080078
48 -103.37249755859375 22.505830764770508
49 -103.40920257568359 22.435550689697266
50 -103.37169647216797 22.327499389648438
51 -103.52230072021484 22.117290496826172
52 -103.63828033466797 22.081670761184692
53 -103.68250274658203 22.112779617309570
54 -103.70089721679688 22.145990037170410
55 -103.61499786376953 22.524719238281250
56 -103.65888739990234 22.573329925537109
57 -103.74140167236328 22.576389312744141
58 -103.86859893798828 22.183889389038086
59 -103.88390353041797 22.461109161376953
60 -103.83360290527344 22.489440917968750
61 -103.87079620361328 22.577220916748047
62 -103.77059936523438 22.636669158935547

63 -103.802001953125 22.72304916381836
64 -104.00700378417969 22.764720916748047
65 -103.99420166015625 22.65860939025879
66 -104.02940368652344 22.581939697265625
67 -103.92169952392578 22.51082992553711
68 -103.95030212402344 22.36804962158203
69 -104.14420318603516 22.342220306396484
70 -104.32959747314453 22.26453971862793
71 -104.31140136718750 22.319210052490234
72 -104.25859832763672 22.421939849853516
73 -104.20110321044922 23.062770843505859
74 -104.16970062255859 23.142780303955078
75 -104.09609985351562 23.195829391479492
76 -104.07810211181641 23.447500228881836
77 -103.93669891357422 23.57305908203125
78 -103.91970062255859 23.623329162597656
79 -103.80829620361328 23.674720764160156
80 -103.85859680175781 23.736940383911133
81 -103.87560272216797 23.86138916015625
82 -103.85089874267578 24.07305908203125
83 -103.60060119628906 24.1825008392334
84 -103.61250305175781 24.275550842285156
85 -103.26750183105469 24.476110458374023
86 -102.76719665527344 24.433889389038086
87 -102.73529815673828 24.458890914916992
88 -102.51360321044922 24.452220916748047
89 -102.50499725341797 24.828609466552734
90 -102.66690063476562 25.075830459594727
91 -102.66580200195312 25.11804962158203
92 -102.25720214843750 25.155550003051758
93 -101.83750152587891 25.026939392089844
94 -101.74639892578125 24.90583038623047
95 -101.58560180664062 24.858060836791992
96 -101.61000061035156 24.788330078125
97 -101.57969665527344 24.754440307617188
98 -101.44529724121094 24.761390686035156
99 -101.36029815673828 24.821109771728516
100 -101.32109832763672 24.778610229492188
101 -101.24220275878906 24.810279846191406
102 -100.99639892578125 24.589719772338867
103 -100.87220001220703 24.601390838623047
104 -100.82389831542969 24.559999465942383
105 -100.98190307617188 24.398609161376953
106 -101.17310333251953 24.113330841064453
107 -101.40219879150339 23.898050308227754
108 -101.68530273437500 23.693889617919922
109 -101.73470306396484 23.610000610351562
110 -101.87030029296875 23.548049926757812
111 -102.05750274658203 23.373609542846680
112 -102.19280242919922 23.388610839843750
113 -102.19360315625000 23.333610534667969
114 -102.28060150146484 23.217500686645508
115 -102.19450378417969 23.113050460815430
116 -102.24500274658203 23.002220153808594
117 -102.14279937744141 22.810279846191406
118 -102.08139801025390 22.755830764770508
119 -101.93440246582031 22.620550153563965
120 -101.87079620361328 22.493129730224609
121 -101.70829772949219 22.460830688476562
122 -101.57080078125000 22.598049163818359
123 -101.48110198974609 22.618610382080078
124 -101.375 22.594169616699219
125 -101.31140136718750 22.535270690917969
126 -101.29889678955078 22.454439163208008
127 -101.36060333251953 22.395549774169922
128 -101.32859802246094 22.079170227050781
129 -101.52490234375 21.856639862060547
28
0 -100.05719757080078 23.24110984802246
1 -100.08920288085938 23.120830535888672
2 -100.02580261230469 23.127220153808594
3 -99.90807342529297 23.002220153808594
4 -99.93640136718750 22.930549621582031
5 -100.04859924316406 22.838890075683594
6 -100.01670074462891 22.803890228271484
7 -99.53056335449219 22.614999771118164
8 -99.53611755371094 22.726669311523438
9 -99.42443847656250 22.633430480957031
10 -99.37806701660156 22.679439544677734
11 -99.23111724853516 22.446109771728516
12 -98.87946319580078 22.340549468994141
13 -98.68611145019531 22.419170379638672
14 -98.61608886718750 22.418489456176758
15 -98.34557342529297 22.228330612182617
16 -98.46925354003906 22.017139434814453
17 -98.48136901855469 21.998680114746094
18 -98.49994659423828 21.974130630493164
19 -98.58856201171875 21.975269317626953
20 -98.57333374023438 21.941110610961914
21 -98.51862335205078 21.951389312744141
22 -98.55500793457031 21.933330535888672
23 -98.53723144531250 21.911659240722656
24 -98.56390308059375 21.884439468383789
25 -98.52111816406250 21.836669921875
26 -98.49028015136719 21.851940155029297
27 -98.45084381103516 21.782220840454101
28 -98.52528381347656 21.720550537109375
29 -98.56278991699219 21.728330612182617
30 -98.56278991699219 21.689439773559570
31 -98.61279296875 21.694719314575195
32 -98.64222717285156 21.608890533447266
33 -98.52443695068359 21.528329849243164
34 -98.51500701904297 21.398889541625977
35 -98.60582733154297 21.334440231323242
36 -98.62001037597656 21.215000152587891
37 -98.81111145019531 21.184999465942383
38 -98.90556335449219 21.216110229492188
39 -98.94306945800781 21.293609619140625
40 -99.04325866699219 21.268320083618164
41 -99.08778381347656 21.287220001220703
42 -99.20001220703125 21.643890380859375
43 -99.25527954101562 21.625829696655273
44 -99.29667663574219 21.563890457153320
45 -99.36695861816406 21.556659698486328
46 -99.41195678710938 21.461109161376953
47 -99.57972717285156 21.423889160156250
48 -99.69056701660156 21.554719924926758
49 -99.74305725097656 21.521659851074220
50 -99.79054260253906 21.419179916381836
51 -100.19319915771484 21.586259841918945
52 -100.29699707031250 21.648889541625977
53 -100.43250274658203 21.650550842285156
54 -100.54969787597656 21.515550613403320
55 -100.60810089111328 21.506389617919922
56 -100.75170135498047 21.568889617919922
57 -100.96859741210938 21.745000839233398
58 -101.20390319824219 21.767499923706055
59 -101.32279968261719 21.861389160156250
60 -101.42669677734375 21.834999084472656
61 -101.52490234375 21.856639862060547
62 -101.32859802246094 22.079170227050781
63 -101.36060333251953 22.395549774169922
64 -101.28859802246094 22.079170227050781
65 -101.31140136718750 21.856639862060547
66 -101.375 22.594169616699219
67 -101.48110198974609 22.618610382080078
68 -101.57080078125 22.598049163818359
69 -101.70829772949219 22.460830688476562
70 -101.87079620361328 22.493129730224609
71 -101.93440246582031 22.620550153563965
72 -102.08139801025390 22.755830764770508
73 -102.14279937744141 22.810279846191406
74 -102.24500274658203 23.002220153808594
75 -102.19450378417969 23.113050460815430
76 -102.28060150146484 23.217500686645508
77 -102.19360315625 23.333610534667969
78 -102.19280620291992 23.388610839843750
79 -102.05750274658203 23.373609542846680
80 -101.87030029296875 23.548049926757812
81 -101.73470306396484 23.610000610351562
82 -101.68530273437500 23.693889617919922
83 -101.40219879150339 23.898050308227754
84 -101.17310333251953 24.113330841064453
85 -100.98190307617188 24.398609161376953
86 -100.82389831542969 24.559999465942383
87 -100.58969879150339 24.290830612182617
88 -100.56220245361328 24.137500762939453
89 -100.60140228271484 23.959999084472656
90 -100.41719818115234 23.747499465942383
91 -100.46810150146484 23.610830307006836
92 -100.42780303955078 23.412220001220703
93 -100.45529937744141 23.278329849243164
94 -100.43440246582031 23.216390609741211
95 -100.37190246582031 23.194440841674805
96 -100.30329895019531 23.248050689697266
97 -100.05719757080078 23.241109848022461
29
0 -99.71816253366211 27.665857315063477
1 -99.91584777832031 27.523050308227754
2 -99.89499664630664 27.447500228881836
3 -99.80029296875 27.456939697265625
4 -99.74833679199219 27.413059234619141
5 -99.69917297363281 27.159160614013672
6 -99.73306274414062 26.913610458374023
7 -99.63390350341797 26.899169921875
8 -99.56639099121094 26.852509915527344
9 -99.63111877441406 26.662220001220703
10 -99.50110626220703 26.673049926757812
11 -99.41500854492188 26.628330230712891
12 -99.38250732421875 26.510000228881836
13 -99.44667053222656 26.453609466552734
14 -99.38722229003906 26.354999542236328
15 -99.31611633300781 26.360830307006836
16 -99.24861145019531 26.291109085083008
17 -99.17696380615234 26.290009915527344
18 -99.11373901367188 26.077779769897461
19 -99.01207733154297 26.095479965209961
20 -98.99649047851562 26.088720321655273
21 -98.89666748046875 25.993890762329102
22 -98.81529235839844 26.057779312133789
23 -98.58500671386719 26.040830612182617
24 -98.55639648437500 25.987220764160156
25 -98.56396808082031 25.520549774169922
26 -98.45056152343750 25.493049621582031
27 -98.44389343261719 25.422779083251953
28 -98.90471649169922 25.073749542236328
29 -98.92485046386719 25.076660156250000
30 -98.99278259277344 25.074169158935547
31 -99.04139709472656 25.122499465942383
32 -99.10611724853516 25.046670913696289
33 -99.15055847167969 25.051389694213867
34 -99.14250183105469 25.012220382690430
35 -99.19528198242188 24.876110076904297
36 -99.16388702392578 24.775829315185547
37 -99.25890350341797 24.804159164428711
38 -99.41388702392578 24.756660461425781
39 -99.56777954101562 24.643049240112305
40 -99.59278869628906 24.654439926147461
41 -99.73166656494141 24.528610229492188
42 -99.67417907714844 24.473890304565430
43 -99.62722778320312 24.497219085693359
44 -99.55724334716797 24.368049621582031
45 -99.61195373535156 24.217500686645508
46 -99.60639953613281 24.076940536499023
47 -99.49111938476562 23.993610382080078
48 -99.45056152343750 23.893610000610352
49 -99.49890136718750 23.894720077514648
50 -99.59777832031250 23.763330459594727
51 -99.83862304687500 23.746669769287109
52 -99.95639038085938 23.531669616699219
53 -99.88834381103516 23.372499465942383
54 -100.02420043945312 23.410270690917969
55 -100.07199859619141 23.354999542236328
56 -100.04280090332031 23.316110610961914
57 -100.05719757080078 23.241109848022461
58 -100.30329895019531 23.248050689697266
59 -100.37190246582031 23.194440841674805
60 -100.43440246582031 23.216390609741211
61 -100.45529937744141 23.278329849243164
62 -100.42780303955078 23.412220001220703
63 -100.46810150146484 23.610830307006836
64 -100.41719818115234 23.747499465942383
65 -100.60140228271484 23.959999084472656
66 -100.56220245361328 24.137500762939453
67 -100.58969879150339 24.290830612182617
68 -100.82389831542969 24.559999465942383
69 -100.78720092773438 24.892780303955078
70 -100.69830322265625 24.931110382080078
71 -100.82530212402344 25.038890838623047
72 -100.77310180664062 25.155550003051758
73 -100.70860290527344 25.198610305786133
74 -100.54360198974609 25.228330612182617
75 -100.43720245361328 25.211940765380859
76 -100.37470245361328 25.157499313354492
77 -100.22920270507812 25.213630676269531
78 -100.19090270996094 25.190929412841797
79 -100.25920104980469 25.254999160766602
80 -100.18969726562500 25.275829315185547
81 -100.30280303955078 25.325000762939453
82 -100.44170379638672 25.327220916748047
83 -100.58170318603516 25.444160464172363
84 -100.69170379638672 25.489999771118164
85 -100.67449951171875 25.536390304565430
86 -100.57779693603516 25.498609542846680
87 -100.57199859619141 25.528329849243164
88 -100.63449859619141 25.553329467773438
89 -100.64219665527344 25.608329772949219
90 -100.70999908447266 25.611660003662109
91 -100.81970214843750 25.743890762329102
92 -100.83249664306641 25.918050765991211
93 -100.91750335693359 25.987779617309570
94 -100.91339874267578 26.056940078735352
95 -100.94940185546875 26.111389160156250
96 -101.03579711914062 26.149440765380859
97 -101.21859741210938 26.370550155639648
98 -100.79419708251953 26.707780838012695
99 -100.69470214843750 26.628330230712891
100 -100.61579895019531 26.750829696655273
101 -100.56559753417969 26.771940231323242
102 -100.53279876708984 26.867219924926758
103 -100.54969787597656 27.030550003051758
104 -100.65859985351562 27.071109771728516
105 -100.69999694824219 27.010280609130859
106 -100.75920104980469 27.046670913696289
107 -100.79530334472656 27.026390075683594
108 -100.82279968261719 27.235279083251953
109 -100.58499908447266 27.395000457763672
110 -100.42829895019531 27.400829315185547
111 -100.31140136718750 27.709999084472656
112 -100.18250274658203 27.794170379638672
113 -99.97306823730469 27.635280609130859
114 -99.80781555175781 27.770837783813477
115 -99.71816253366211 27.665857315063477
30
0 -97.1722183227539 25.95456886291504
1 -97.16862487792969 25.707496643066406
2 -97.29055786132812 25.432498931884766
3 -97.48028564453125 25.124164581298828
4 -97.58416748046875 24.784721374511719
5 -97.68806457519531 24.322498321533203
6 -97.72695922851562 23.789718627929688
7 -97.81806945800781 23.783885955810547
8 -97.75584411621094 23.765830993652344
9 -97.75334167480469 23.647499084472656
10 -97.72334289550781 23.750553131103516
11 -97.76640319824219 23.300830841064453
12 -97.74139404296875 22.905830383300781
13 -97.75556945800781 22.849441528320312
14 -97.75889587402344 22.924999237060547
15 -97.80584716796875 22.773609161376953
16 -97.87251281738281 22.73388671875
17 -97.84777832031250 22.708610534667969
18 -97.89028930664062 22.606666564941406
19 -97.86584472656250 22.581108093261719
20 -97.82806396484375 22.662220001220703
21 -97.84584045410156 22.518054962158203
22 -97.77687072753906 22.268054962158203
23 -97.77687072753906 22.268049240112305
24 -97.87612915039062 22.221309661865234
25 -97.92556762695312 22.271940231323242
26 -97.91335296630859 22.326389312744141
27 -98.10195922851562 22.382770538330078
28 -98.19306945800781 22.471389770507812
29 -98.29332733154297 22.468610763549805
30 -98.31390380859375 22.398330688476562
31 -98.49001312255859 22.440280914306641
32 -98.61608886718750 22.418489456176758
33 -98.68611145019531 22.419170379638672
34 -98.87946319580078 22.340549468994141
35 -99.23111724853516 22.446109771728516
36 -99.37806701660156 22.679439544677734
37 -99.36111145019531 22.419170379638672
38 -99.47946319580078 22.340549468994141
39 -99.23111724853516 22.446109771728516
40 -100.01670074462891 22.803890228271484
41 -100.04859924316406 22.838890075683594
42 -99.93640136718750 22.930549621582031
43 -99.90807342529297 23.002220153808594
44 -100.02580261230469 23.127220153808594
45 -100.08920288085938 23.120830535888672
46 -100.05719757080078 23.241109848022461
47 -100.04280090332031 23.316110610961914
48 -100.07199859619141 23.354999542236328
49 -100.02420043945312 23.410270690917969
50 -99.88834381103516 23.372499465942383
51 -99.95639038085938 23.531669616699219
52 -99.83862304687500 23.746669769287109
53 -99.59777832031250 23.763330459594727
54 -99.49890136718750 23.894720077514648
55 -99.45056152343750 23.893610000610352
56 -99.49111938476562 23.993610382080078
57 -99.60639953613281 24.076940536499023
58 -99.61195373535156 24.217500686645508
59 -99.55724334716797 24.368049621582031
60 -99.62722778320312 24.497219085693359
61 -99.67417907714844 24.473890304565430
62 -99.73166656494141 24.528610229492188
63 -99.59278869628906 24.654439926147461
64 -99.56777954101562 24.643049240112305
65 -99.41388702392578 24.756660461425781
66 -99.25890350341797 24.804159164428711
67 -99.19528198242188 24.876110076904297
68 -99.19528198242188 24.876110076904297
69 -99.14250183105469 25.012220382690430
70 -99.15055847167969 25.051389694213867
71 -99.10611724853516 25.046670913696289
72 -99.04139709472656 25.122499465942383
73 -98.99278259277344 25.074169158935547
74 -98.92485046386719 25.076660156250000
75 -98.90471649169922 25.073749542236328
76 -98.44389343261719 25.422779083251953
77 -98.45056152343750 25.493049621582031
78 -98.56916809082031 25.520549774169922
79 -98.55639648437500 25.987220764160156
80 -98.58500671386719 26.040830612182617
81 -98.81529235839844 26.057779312133789
82 -98.89666748046875 25.993890762329102
83 -98.99649047851562 26.088720321655273
84 -99.01207733154297 26.095479965209961
85 -99.11373901367188 26.077779769897461
86 -99.17696380615234 26.290009915527344
87 -99.24861145019531 26.291109085083008
88 -99.31611633300781 26.360830307006836
89 -99.38722229003906 26.354999542236328
90 -99.44667053222656 26.453609466552734
91 -99.38250732421875 26.510000228881836
92 -99.41500854492188 26.628330230712891
93 -99.50110626220703 26.673049926757812
94 -99.63111877441406 26.662220001220703
95 -99.56639099121094 26.852509915527344
96 -99.63390350341797 26.899169921875
97 -99.73306274414062 26.913610458374023
98 -99.69917297363281 27.159160614013672
99 -99.74833679199219 27.413059234619141
100 -99.80029296875 27.456939697265625
101 -99.89499664630664 27.447500228881836
102 -99.91584777832031 27.523050308227754
103 -99.71816253366211 27.665857315063477
104 -99.71449279785156 27.661558151245117
105 -99.54918670654297 27.612627029418945
106 -99.52674102783203 27.504283905029297
107 -99.49049377441406 27.490755081176758
108 -99.54358673095703 27.318653106689453
109 -99.46527099609375 27.269884109497070
110 -99.43715667724609 27.199197769165039
111 -99.45506286621094 27.028648376464844
112 -99.39271545410156 26.995550155963135
113 -99.39051818847656 26.946630477905273
114 -99.28552246093750 26.857360839843750
115 -99.16581726074219 26.579889297485352
116 -99.16867828369141 26.545728063964844
117 -99.10147094726562 26.488340377807617
118 -99.10672760009766 26.419530868530273
119 -98.93927001953125 26.395309448242188
120 -98.90889739990234 26.360328674316406
121 -98.81983184814453 26.375070571899414
122 -98.67791748046875 26.242055892944336
123 -98.59996795654297 26.260454177856445
124 -98.48851776123047 26.201543807983398
125 -98.45339202880859 26.220911026000977
126 -98.38452148437500 26.156030654907227
127 -98.34719085693359 26.158679962158203
128 -98.32793426513672 26.111646652221680
129 -98.29227447509766 26.132808685302734
130 -98.27135467529297 26.120895385742188
131 -98.29194641113281 26.098104467692871
132 -98.20069122314453 26.055376052856445
133 -98.08321380615234 26.065757751464844
134 -98.07634735107422 26.034626007080078
135 -98.04006958007812 26.258519677612305
136 -97.86743164062500 26.060140609741211
137 -97.64797210693359 26.023445129394531
138 -97.61292266845703 25.962001800537109
139 -97.57493591308594 25.954172134399414
140 -97.59008789062500 25.933231353759766
141 -97.43434906005859 25.845197677612305
142 -97.38564300537109 25.845361709594727
143 -97.38098907470703 25.917020797729492
144 -97.30443572998047 25.938663482666016
145 -97.30714416503906 25.965124130249023
146 -97.17221832275391 25.954568862915039
31
0 -93.58940124511719 17.375459671020508
1 -93.6269531250 17.3075008392334
2 -93.87342834472656 17.150279998779297
3 -94.32805633544922 17.172779083251953
4 -94.96721649169922 17.222499847412109
5 -95.00110626220703 17.335830688476562
6 -95.06916809082031 17.346660614013672
7 -95.25170898437500 17.594770431518555
8 -95.20558166503906 17.648189544677734
9 -95.21028137207031 17.733329772949219
10 -95.36418151855469 17.641389846801758
11 -95.43917846679688 17.633050918579102
12 -95.56001281738281 17.533050537109375
13 -95.72501373291016 17.501659393310547
14 -95.79028320312500 17.524719238281250
15 -95.91555786132812 17.779159545898438
16 -95.79945373535156 17.941110610961914
17 -95.80445861816406 18.051389694213867
18 -95.86222839355469 18.119159698486328
19 -96.09465026855469 18.164079666137695
20 -96.15028381347656 18.141670227050781
21 -96.16139221191406 18.185550689697266
22 -96.20529174804688 18.180269241333008
23 -96.23668670654297 18.213920593261719
24 -96.25473022460938 18.291660308837891
25 -96.35694885253906 18.388610839843750
26 -96.40583801269531 18.541109085083008
27 -96.67417907714844 18.681110382080078
28 -96.63555908203125 18.522220611572266
29 -96.75063323974609 18.430830001831055
30 -96.80751037597656 18.552780151367188
31 -97.03861999511719 18.476669311523438
32 -97.14418029785156 18.643329620361328
33 -97.27278137207031 18.632499694824219
34 -97.34529113769531 18.769439697265625
35 -97.24778747558594 18.887220382690430
36 -97.25083923339844 19.026390075683594
37 -97.26471710205078 19.159999847412109
38 -97.17001342773438 19.193609237670898
39 -97.07972717285156 19.183050155563965
40 -97.00167846679688 19.267499923706055
41 -97.05612182617188 19.307779312133789
42 -97.18556213378906 19.306659698486328
43 -97.24639892578125 19.373609542846680
44 -97.33445739746094 19.401390075683594
45 -97.35334777832031 19.538330078125000
46 -97.44000244140625 19.585830688476562
47 -97.35444641113281 19.619720458984375
48 -97.30889892578125 19.683889389038086
49 -97.28527832031250 19.75
50 -97.30944824218750 19.895549774169922
51 -97.13722229003906 20.117500305175781
52 -97.14639282226562 20.147220611572266
53 -97.38144683837891 20.263729095458984
54 -97.41223144531250 20.262220382690430
55 -97.47057342529297 20.239719390869141
56 -97.51500701904297 20.121110916137695
57 -97.56443512500000 20.106670379638672
58 -97.61473083496094 20.168329238891602
59 -97.69249725341797 20.176109313964844
60 -97.75279235839844 20.255279541015625
61 -97.75877298583984 20.439899444580078
62 -97.69333648681641 20.469720840454102
63 -97.62889099121094 20.417770385742188
64 -97.57064819335938 20.490299224853516
65 -97.57917785644531 20.588609695434570
66 -97.74223327636719 20.650550842285156
67 -97.73390197753906 20.793050765991211
68 -97.87390136718750 20.805000305175781
69 -97.88305664062500 20.706390380859375
70 -97.94862365722656 20.666660308837891
71 -97.96305847167969 20.519720077514648
72 -98.09818267822266 20.432170867919922
73 -98.04279327392578 20.506940841674805
74 -98.02944946289062 20.606939315795898
75 -98.03083801269531 20.642499923706055

76 -98.09529113769531 20.661659240722656
77 -98.33500671386719 20.434999465942383
78 -98.40223693847656 20.441110610961914
79 -98.45278930664062 20.359170913696289
80 -98.49474334716797 20.376390457153320
81 -98.56611633300781 20.501659393310547
82 -98.42445373535156 20.718610763549805
83 -98.49890136718750 21.712499618530273
84 -98.51083374023438 20.756389617919922
85 -98.42111206054688 20.790279388427734
86 -98.36695861816406 20.858610153198242
87 -98.23056030273438 20.830829620361328
88 -98.22000122070312 20.961940765380860
89 -98.17639160156250 21.027500152587890
90 -98.15306091308594 21.019439697265625
91 -98.13056945800781 21.074720382690430
92 -98.21250915527344 21.157220804541
93 -98.28816223144531 21.129779815673828
94 -98.27290344238281 21.183610916137695
95 -98.26306152343750 21.213050842285156
96 -98.29888916015625 21.233890533447266
97 -98.33750915527344 21.152219772338867
98 -98.41056823730469 21.154159545898438
99 -98.48722839355469 21.241939544677734
100 -98.47722625732422 21.352220053527832
101 -98.51500701904297 21.398889541625977
102 -98.52443695068836 21.528329849243164
103 -98.64222717285156 21.608890533447266
104 -98.61279296875 21.694719314575195
105 -98.56278991699219 21.689439773553957
106 -98.56278991699219 21.728330612182617
107 -98.52528381347656 21.720550537109375
108 -98.45084381103516 21.782220804541
109 -98.49028015136719 21.851940155029297
110 -98.52111816640625 21.836669921875
111 -98.56390380859375 21.884439468338379
112 -98.53723144453125 21.911659240722656
113 -98.55500793457031 21.933330535888672
114 -98.51862335205078 21.951389312744414
115 -98.57333374023438 21.941110610961914
116 -98.58856201171875 21.975269317626953
117 -98.49994659423828 21.974130630493164
118 -98.48136901855469 21.998680114746094
119 -98.46925354003906 22.017139434814453
120 -98.34557342529297 22.228330612182617
121 -98.61608886718750 22.418489456176758
122 -98.49001312255859 22.440280914306664
123 -98.31390380859375 22.398330688476562
124 -98.29332733154297 22.468610763549805
125 -98.19306945800781 22.471389770507812
126 -98.10195922851562 22.382770538330078
127 -97.91335296630859 22.326389312744414
128 -97.92556762695312 22.271940231323242
129 -97.87612915039062 22.221309661865234
130 -97.77687072753906 22.268049240112305
131 -97.77917480468750 22.157775878906250
132 -97.69917297363281 21.976943969726562
133 -97.55612182617188 21.774997711181641
134 -97.31723022460938 21.564163208007812
135 -97.32861328125 21.467777252197266
136 -97.41667175292969 21.271110534667969
137 -97.47666931152344 21.434165954589844
138 -97.38694763183594 21.471942901611328
139 -97.36973571777344 21.538055419921875
140 -97.62028503417969 21.789165496826172
141 -97.65390014648438 21.899166107177734
142 -97.78140258789062 22.088607788085938
143 -97.71473693847656 21.934719085693359
144 -97.67028808593750 21.671108245849609
145 -97.56806945800781 21.487777709960938
146 -97.48667907714844 21.483608245849609
147 -97.48333740234375 21.371944427490234
148 -97.20056152343750 20.812774658203125
149 -97.17140197753906 20.676109313964844
150 -96.67639160156250 20.157218933105469
151 -96.44778442382812 19.861663818359375
152 -96.27694702148438 19.314720153808594
153 -96.16694641113281 19.228610992431641
154 -96.11611938476562 19.223609924316406
155 -96.08445739746094 19.101665496826172
156 -96.03945922851562 19.060276031494141
157 -95.97056579589844 19.058330535888672
158 -95.90223693847656 18.871944427490234
159 -95.78028869628906 18.811940514992188
160 -95.75306701660156 18.803607940673828
161 -95.75750732421875 18.763332366943359
162 -95.94972229003906 18.863887786865234
163 -95.80889892578125 18.746109008789062
164 -95.87583923339844 18.754165649414062
165 -95.84611511230469 18.715553283691406
166 -95.77473449707031 18.744441986083984
167 -95.57167053222656 18.671943664550781
168 -95.73278808593750 18.750553131103516
169 -95.73167419433594 18.795555114746094
170 -95.57417297363281 18.717498779296875
171 -95.21278381347656 18.711109161376953
172 -95.05139160156250 18.613052368164062
173 -95.01889038085938 18.558052062988281
174 -94.80223083496094 18.522499084472656
175 -94.57978820800781 18.190330505371094
176 -94.47889709472656 18.146663665771484
177 -94.16806030273438 18.198608398437500
178 -94.13782501220703 18.209142684936523
179 -94.09445190429688 18.155830383300781
180 -94.09278869628906 18.068609237670898
181 -94.05139160156250 17.991659164428711
182 -94.07528686523438 17.880550385421484
183 -93.96806335449219 17.831390380859375
184 -93.92388916015625 17.745830535888672
185 -93.86473083496094 17.749719619750977
186 -93.85778808593750 17.722219467163086
187 -93.74195861816406 17.683610916137695
188 -93.68936157226562 17.561370849609375
189 -93.65390014648438 17.522499084472656
190 -93.66751098632812 17.452770233154297
191 -93.58940124511719 17.375459671020508

- Data Uji Coba Polyline

polyline
0
0 -98.13920593261719 22.552047729492188
1 -98.80698394775391 23.240459442138672
2 -98.86305999755859 23.284263610839844
3 -99.00114440917969 23.316827774047852
1
0 -99.10936737060547 19.805984497070312
1 -98.96969604492188 19.771375656127930
2 -98.98390197753906 19.680707931518555
3 -98.90819549560547 19.634996414184570
4 -98.89491271972656 19.675167083740234
5 -98.79182434082031 19.715181350708008
6 -98.62443542480469 19.862964630126953
7 -98.44770050048828 20.066602706909180
8 -98.26927185058594 20.168119430541992
9 -98.10771942138672 20.120254516601562
10 -98.03823089599609 20.220081329345703
11 -97.97956848144531 20.244550704956055
12 -97.93649291992188 20.370296478271484
13 -97.80111694335938 20.446706771850586
14 -97.54138946533203 20.500024795532227
15 -97.47367095947266 20.553863525390625
2
0 -103.55303955078125 19.380632400512695
1 -103.48767852783203 19.455810546875
2 -103.52696990966797 19.498273849487305
3 -103.51123046875 19.537647247314453
4 -103.45470428466797 19.530595779418945
5 -103.42726135253906 19.624608993530273
6 -103.48171234113086 19.661642074584961
7 -103.48139190673828 19.802940368652344
8 -103.51401519775391 19.853538513183594
9 -103.59980773925781 19.886968126709
10 -103.43490600585938 20.106674194335938
11 -103.45087432861328 20.275203704833984
3
0 -98.17977142333984 19.085229873657227
1 -98.61325836181641 19.352220535278320
2 -98.69737234652344 19.308343887329102
3 -98.92427825927734 19.295913696289062
4 -99.06818389892578 19.410888671875
5 -99.19351959228516 19.413570404052734
4
0 -99.24839782714844 19.048128128051758
1 -99.15768432617188 19.225410461425781
2 -99.22344207763672 19.332986831665039
3 -99.19351959228516 19.413570404052734
5
0 -103.80115509033203 19.008768066406250
1 -103.73291015625 19.244281768798828
2 -103.60496520996094 19.320915222167969
3 -103.55303955078125 19.380632400512695
6
0 -98.17977142333984 19.085229873657227
1 -98.88571929931641 19.037792205810547
2 -97.49718475341797 18.820150375366211
3 -97.38732910156250 18.887975907446289
4 -97.16186523437500 18.819280624389648
5 -96.94556427001953 18.909959793090820
6 -96.69460929663086 18.803607940673828
7 -96.65686035156250 18.754720687866211
8 -96.56147766113281 18.745605468750000
9 -96.31847381591797 18.8203125
10 -96.19523626205469 18.909311294555664
11 -96.12701416015625 19.021902084350586
7
0 -90.73926544189453 19.344089508056641
1 -90.73442840576172 19.186143875122070
2 -90.69801330566406 19.087060928344727
3 -90.74277496337891 19.093144989013672
4 -90.68738555908203 18.865163803100586
5 -90.75406646728516 18.662309249560547
6 -90.73566436767578 18.615751266479492
8
0 -98.17977142333984 19.085229873657227
1 -98.20897674560547 19.047203063964844
2 -98.32649230957031 19.012741088867188
3 -98.44714355468750 18.903293606911469
4 -98.45762634277344 18.597587585449219
9
0 -98.45762634277344 18.597587585449219
1 -98.88558959960938 18.758331298828125
2 -99.07452392578125 18.959896087646484
3 -99.24839782714844 19.048128128051758
10
0 -99.24839782714844 19.048128128051758
1 -99.26949310302734 19.032897949218750
2 -99.22885894775391 18.880485534667969
3 -99.24325561523438 18.777820587158203
4 -99.30345916748047 18.650505065917969
5 -99.40689849853516 18.575290676993164
11
0 -101.79046630859375 19.799131393432617
1 -101.84362030029297 19.724540710449219
2 -101.83450317382812 19.579763412475586
3 -101.87545776367188 19.491081237792969
4 -102.00881195068359 19.462226867675781
5 -102.0605468750 19.398839950561523
6 -102.06324005126953 19.086585998535156
7 -102.14235687255859 18.828720092773438
8 -102.25086212158203 18.729322433471680
9 -102.08269500732422 18.460737228393555
10 -102.11546325683594 18.445804959472656
11 -102.10710144042969 18.420745849609375
12 -102.29084777832031 18.356105804443359
13 -102.25686645507812 18.217784881591797
14 -102.28009796142578 18.144607543945312
12
0 -99.40689849853516 18.57529067993164
1 -99.44404602050781 18.548276901245117
2 -99.43357849121094 18.478021621704102
3 -99.53810882568359 18.360361099243164
4 -99.50730133056641 18.302539825439453
5 -99.55483245849610 18.177537918090820
6 -99.54829406738281 18.027479171752930
7 -99.59439849853516 17.948101043701172
8 -99.52498626770898 17.572341918945312
9 -99.46488952636719 17.469566345214844
10 -99.48026275634766 17.299699783325195
11 -99.53599548339844 17.167741775512695
12 -99.58931732177734 17.177837371826172
13 -99.76406097412109 17.072593688964844
14 -99.78825378417969 16.973054885864258
13
0 -98.45762634277344 18.597587585449219
1 -98.19299316406250 18.231063842773438
2 -98.12873077392578 18.223941802978516
3 -98.07994842529297 18.178592681884766
4 -98.05541229248047 18.206939697265625
5 -97.97665405273438 18.141006469726562
6 -97.74546051025390 17.748630523681641
7 -97.47534179687500 17.651411056518555
8 -97.43719482421875 17.576333999633789
9 -97.22795104980469 17.447423934936523
10 -97.16970062255859 17.334009170532227
11 -97.08241271972656 17.304653167724609
12 -97.04243469238281 17.220714569091797
13 -96.95446014404297 17.285148620605469
14 -96.87515258789062 17.280509948730469
15 -96.80374908447266 17.230773925781250
16 -96.72885131835938 17.077991485595703
17 -96.33345031738281 16.908538818359375
18 -96.29170989990234 16.689620971679688
19 -96.36012268066406 16.565196990966797
20 -96.31735229492188 16.554065704345703
21 -96.24795532226562 16.428565979003906
22 -96.18653106689453 16.519721984863281
23 -96.06200408935547 16.483879089355469
24 -96.02327728271484 16.540174484252930
25 -95.90851593017578 16.499414443969727
26 -95.82138061523438 16.396528244018555
27 -95.57236480712891 16.407924652099609
28 -95.49495697021484 16.443704605102539
29 -95.42388916015625 16.427877426147461
30 -95.37094116210938 16.362220764160156
31 -95.22793579101562 16.318861007690430
14
0 -115.48423004150391 32.663681030273438
1 -115.44100189208984 32.616191864013672
15
0 -116.88575744628906 32.439590454101562
1 -116.73117065429688 32.470195770263672
2 -116.62241363525391 32.571735382080078
3 -116.48696136474609 32.568511962890625
4 -116.13248443603516 32.473213195800781
5 -116.03031921386719 32.547771453857422
6 -116.03651428222656 32.572635650634766
7 -115.99660491943359 32.542396545410156
8 -115.66409301757812 32.599739074707031
9 -115.57735443115234 32.596683502197266
10 -115.44100189208984 32.616191864013672
16
0 -116.88575744628906 32.439590454101562
1 -117.03643035888672 32.543209075927734
17
0 -115.30187988281250 32.15225601196289
1 -115.31527709960938 32.179637908935547
2 -115.33641815185547 32.364883422851562
3 -115.44100189208984 32.616191864013672
18
0 -116.38530731201172 31.896617889404297
1 -116.44403839111328 31.968904495239258
2 -116.83031463623047 32.305484771728516
3 -116.88227081298828 32.431987762451172
4 -116.88575744628906 32.439590454101562
19
0 -106.48795318603516 31.766572952270508
1 -106.43035125732422 31.648981094360352
20
0 -114.99444580078125 31.094306945800781
1 -115.06958007812500 31.459945678710938
2 -115.13468933105469 31.725408554077148
3 -115.22736358642578 32.0
4 -115.30187988281250 32.15225601196289
21
0 -106.43035125732422 31.648981094360352
1 -106.47203826904297 31.510700225830078
2 -106.45437622070312 31.392599105834961
3 -106.50205230712891 31.284790039062500
4 -106.46198272705078 30.914659500122070
22
0 -110.94062805175781 31.335670471191406
1 -110.97500610351562 31.201580047607422
2 -110.85015869140625 30.956420898437500
3 -110.85311126708984 30.784162521362305
23
0 -110.85311126708984 30.784162521362305
1 -110.70431518554688 30.898881912231445
2 -110.64499664306641 30.877582550048828
3 -110.56174468994141 30.997287750244141
4 -110.20428466796875 31.015897750854492
5 -110.12483215332031 31.066158294677734
6 -110.07715606689453 31.159839630126953
7 -109.95024108886719 31.184343338012695
8 -109.81684875488281 31.269309997558594
9 -109.71903991699219 31.268726348876953
10 -109.55690765338086 31.322494506835938
11 -109.38133239746094 31.229496002197266
12 -109.35474395751953 31.126077651977554
13 -109.29721832275391 31.078344345092773
14 -109.22355651855469 30.825305938720703
15 -109.15032196044922 30.861402511596680
16 -109.05789947509766 30.732305526733398
17 -109.00684356689453 30.745899200439453
18 -109.01060485839844 30.788085937500000
19 -108.93089294433594 30.901699066162109
20 -108.38481140136719 30.756286621093750
21 -108.27399444580078 30.870306015014648
22 -108.14411926269531 30.924531936645508
23 -108.13675689697266 31.035377502441406
24 -107.89217376708984 31.171463012695312
25 -107.77793121337891 31.299739837646484
26 -107.74542236328125 31.383419036865234
27 -107.62738037109375 31.382989883422852
28 -107.30525207519531 31.480352401733398
29 -106.89985656738281 31.623353248901367
30 -106.85609436035156 31.671960830688477
31 -106.59384918212891 31.714187622070312
32 -106.43035125732422 31.648981094360352
24
0 -111.11242675781250 30.541179656982422
1 -110.88816070556641 30.679807662963867
2 -110.85311126708984 30.784162521362305
25
0 -115.44100189208984 32.616191864013672
1 -115.28126525878906 32.551559448242188
2 -115.00129699707031 32.519332885742188
3 -114.12458038330078 32.272617340087891
4 -113.88725280761719 32.129676818847656
5 -112.85962677001953 31.887037277221680
6 -112.86524963378906 31.723276138305664
7 -112.51583862304688 31.270942687988281
8 -112.36339569091797 30.969396591186523
9 -112.16850280761719 30.730306625366211
10 -112.05503082275391 30.680528640747070
11 -111.82955932617188 30.711977005004883
12 -111.55710601806641 30.598764419555664
13 -111.11242675781250 30.541179656982422
26
0 -106.46198272705078 30.914659500122070
1 -106.51606750488281 30.751783370971680
2 -106.52896118164062 30.375762939453125
27
0 -106.52896118164062 30.375762939453125
1 -106.35691070556664 29.966569900512695
2 -106.38706970214844 29.863655090332031
3 -106.32178497314453 29.562791824340820
4 -106.37527465820312 29.533700942993164
5 -106.41990661621094 29.435903549194336
6 -106.33234405517578 29.030767440795898
7 -106.21121215820312 28.852624893188477
8 -106.20551300048828 28.785900115966797
9 -106.05220794677734 28.627302169799805
10 -105.96237182617188 28.588750839233398
11 -105.93405151367188 28.517448425292969
12 -105.71402740478516 28.447113037109375
13 -105.48714447021484 28.273399353027344
14 -105.36421203613281 28.245090484619141
15 -105.36412811279297 28.175878524780273
16 -105.28820800781250 28.086439132690430
17 -105.29029846191406 27.980943679809570
18 -105.19203186035156 27.853679656982422
19 -105.15859985351562 27.671977996826172
20 -105.10816192626953 27.616222381591797
28
0 -105.10816192626953 27.616222381591797
1 -104.94274902343750 27.433376312255859
2 -104.92945098876953 27.127773284912109
29
0 -99.505256652832031 27.495365142822266
1 -99.764335632324219 26.973493576049805
30
0 -99.764335632324219 26.973493576049805
1 -99.854614257812500 26.798080444335938
2 -100.170555114746094 26.511772155761719
3 -100.086715698242188 26.266239166259766
4 -100.118904113769531 26.003196716308594
5 -100.274177551269531 25.802551269531250
6 -100.315490722656250 25.661518096923828
31
0 -104.92945098876953 27.127773284912109
1 -104.34567260742188 26.766773223876953
2 -103.69494628906250 26.087995529174805
3 -103.49382019042969 25.586135864257812
32
0 -108.09827423095703 25.472293853759766
1 -108.30030822753906 25.495544433593750
2 -108.81982421875 25.816797256469727
3 -108.98059082031250 25.820027674560547
4 -109.05595397949219 25.957237243652344
5 -109.01709747314453 26.020257949829102
6 -109.01828765869140 26.024868011474609
7 -108.84106445312500 26.279146194458008
8 -108.96424102783203 26.492893218994141
9 -109.35879516601562 26.810253143310547
10 -109.43181610107422 26.944801330566406
11 -109.45054626464844 27.104482650756836
12 -109.93235778808594 27.481643676757812
13 -109.92048645019531 27.590358734130859
14 -109.95774841308594 27.617017745971680
15 -110.08898162841797 27.599201202392578
16 -110.29584503173828 27.642541885375977
17 -110.28295135498047 27.750015258789062
18 -110.36265563964844 27.873189926147461
19 -110.54379272460938 27.952892303466797
20 -110.85173797607422 28.036218643188477
21 -110.96472167968750 28.098985671997070
22 -111.03927612304688 28.279699325561523
23 -111.05003356933594 28.524341583251953
24 -110.95777893066406 28.817974090576172
25 -110.95870971679688 29.039720535278320
26 -110.84043884277344 29.035095214843750
27 -110.83592987060547 29.105138778686523
28 -110.86141204833984 29.140159606933594
29 -110.91542053222656 29.114822387695312
30 -111.05889892578125 29.808158874511719
31 -111.11550140380860 29.913309097290039
32 -111.08987426757812 30.429050445556641
33 -111.11242675781250 30.541179656982422
33
0 -100.98526763916016 25.423786163330078
1 -100.90292353898438 25.576700210571289
2 -100.64228820800781 25.693258285522461
3 -100.52172088623047 25.695953369140625
4 -100.42491912841797 25.645479202270508
5 -100.31549072265625 25.661518096923828
34
0 -100.50785827636719 28.687894821166992
1 -100.60336303710938 28.576074600219727
2 -100.85127258300781 28.375846862792969
3 -100.83728027343750 28.342655181884766
4 -101.03605651855469 28.260690689086914
5 -101.11153411865234 28.157880783081055
6 -101.20772552490234 27.938106536865234
7 -101.11875915527344 27.838094711303711
8 -101.18605041503906 27.651119232177734
9 -101.22555541992188 27.202919006347656
10 -101.42337036132812 26.902362823486328
11 -101.42291259765625 26.790660858154297
12 -101.34906768798828 26.589435577392578
13 -101.35840606689453 26.057336807250977
14 -101.21865844726562 25.925846099853516
15 -101.15544128417969 25.920883178710938
16 -101.10898590088789 25.776947021484375
17 -101.10185241699219 25.682954440673828
18 -100.98936462402344 25.516914367675781
19 -100.98526763916016 25.423786163330078
35
0 -103.49382019042969 25.586135864257812
1 -103.32716369628906 25.767545700073242
2 -103.11321258544922 25.785413742065430
3 -102.96733856201172 25.756607055664062
4 -102.87482452392578 25.628334045410156
5 -102.71221923828125 25.582761764526367
6 -102.41402435302734 25.679468154907227
7 -102.00797271728516 25.647136688232422
8 -101.56401824951172 25.665313720703125
9 -101.29737854003906 25.445991151611328
10 -101.04431915283203 25.457778930664062
11 -100.98526763916016 25.423786163330078
36
0 -100.31549072265625 25.661518096923828
1 -100.04393005371094 25.296283721923828

2 -99.90374755859375 25.198331832885742
3 -99.82538604736328 25.182058334350586
37
0 -107.29000091552734 24.67160987854004
1 -107.56788635253906 24.99049949645996
2 -107.8890151977539 25.224885940551758
3 -107.94305419921875 25.335002899169922
4 -108.09827423095703 25.472293853759766
38
0 -97.5024642944336 25.88355255126953
1 -97.51649475097656 25.820112228393555
2 -97.66690063476562 25.559734344482422
3 -97.80072784423828 25.473478317260742
4 -97.84380340576172 25.405941009521484
5 -97.87290954589844 25.351694107055664
6 -97.86331176757812 25.272974014282227
7 -98.06756591796875 25.070154190063477
8 -98.1362533569336 24.86134910583496
9 -98.25823211669922 24.734111785888672
10 -98.30854797363281 24.52762222290039
11 -98.48269653320312 24.215221405029297
39
0 -104.66083526611328 24.024341583251953
1 -104.65049743652344 24.032331466674805
2 -104.57369995117188 24.091697692871094
3 -104.59442138671875 24.139606475830078
4 -104.47254943847656 24.225914001464844
5 -104.3762435913086 24.366437911987305
6 -104.02713775634766 24.498624801635742
7 -103.9247817993164 24.58612823486328
8 -103.84362030029297 24.73193359375
9 -103.7396240234375 24.78122901916504
10 -103.70000457763672 24.856517791748047
11 -103.7080078125 24.934017181396484
12 -103.77145385742188 25.026826858520508
13 -103.76675415039062 25.104204177856445
14 -103.69775390625 25.194581985473633
15 -103.61746978759766 25.237924575805664
16 -103.68609619140625 25.479076385498047
17 -103.49382019042969 25.586135864257812
40
0 -104.92945098876953 27.12777328491211
1 -105.6423568725586 26.92787742614746
2 -105.60066223144531 26.880971908569336
3 -105.60587310791016 26.83406639099121
4 -105.5433349609375 26.688135147094727
5 -105.4912109375 26.448392868041992
6 -105.30879974365234 26.370216369628906
7 -105.16287231445312 26.307674407958984
8 -104.57393646240234 25.541542053222656
9 -104.63648223876953 25.312223434448242
10 -104.57914733886719 25.20798683166504
11 -104.49054718017578 25.009939193725586
12 -104.4697036743164 24.858797073364258
13 -104.53224182128906 24.75977325439453
14 -104.63648223876953 24.67638397216797
15 -104.69902038574219 24.46269989013672
16 -104.69902038574219 24.254228591918945
17 -104.6885986328125 24.103086471557617
18 -104.66083526611328 24.024341583251953
41
0 -110.23724365234375 23.87031364440918
1 -110.31592559814453 24.143749237060547
2 -110.37770080566406 24.09360122680664
3 -110.46720123291016 24.117935180664062
4 -110.57307434082031 24.05215072631836
5 -110.64275360107422 24.093900680541992
6 -110.72150421142578 24.088165283203125
7 -110.7870101928711 24.143245697021484
8 -110.91849517822266 24.161293029785156
9 -111.02162170410156 24.3626766204834
10 -111.46446990966797 24.587080001831055
11 -111.77418518066406 25.249635696411133
12 -111.56360626220703 25.387353897094727
13 -111.40067291259766 25.57955932617187 5
14 -111.24371337890625 25.673662185668945
15 -111.35665130615234 25.81456756591797
16 -111.37118530273438 25.999162673950195
17 -111.36796569824219 26.063720703125
18 -111.5215072631836 26.311281204223633
19 -111.68695068359375 26.507709503173828
20 -111.7552719116211 26.538803100585938
21 -111.93224334716797 26.724468231201172
22 -111.90767669677734 26.847898483276367
23 -112.02608489990234 26.917139053344727
24 -112.15581512451172 27.144161224365234
25 -112.2258529663086 27.192501068115234
26 -112.28253936767578 27.29570198059082
27 -112.32091522216797 27.390583038330078
28 -112.62644958496094 27.414722442626953
29 -112.80167388916016 27.319917678833008
30 -113.05819702148438 27.29704475402832
31 -113.26209259033203 27.429309844970703
32 -113.38542175292969 27.65825843811035
33 -113.45347595214844 27.724733352661133
34 -113.87832641601562 27.90898895263672
35 -113.87986755371094 28.058761596679688
36 -113.81422424316406 28.385597229003906
37 -113.89217376708984 28.50867462158203
38 -114.00157928466797 28.678245544433594
39 -114.10566711425781 28.805994033813477
40 -114.10575103759766 28.827322006225586
41 -114.15645599365234 28.929380416870117
42 -114.15248107910156 29.236906051635742
43 -114.38101959228516 29.383621215820312
44 -114.67650604248047 29.721818923950195
45 -114.72701263427734 29.734533309936523
46 -114.87886047363281 29.894935607910156
47 -115.16743469238281 29.965478897094727
48 -115.288330078125 30.060726165771484
49 -115.4301528930664 30.068340301513672
50 -115.40644836425781 30.13590431213379
51 -115.42684936523438 30.14731216430664
52 -115.57411193847656 30.151155471801758
53 -115.63922882080078 30.30689239501953
54 -115.64725494384766 30.314380645751953
55 -115.82939910888672 30.369352340698242
56 -115.86131286621094 30.388498306274414
57 -115.91827392578125 30.77372932434082
58 -115.93451690673828 30.857912063598633
59 -116.11912536621094 31.09864044189453
60 -116.12593078613281 31.14302635192871
61 -116.17733764648438 31.268016815185547
62 -116.3192367553711 31.392353057861328
63 -116.35529327392578 31.499412536621094
64 -116.38809967041016 31.569332122802734
65 -116.41693115234375 31.691055297851562
66 -116.39434051513672 31.822072982788086
67 -116.38530731201172 31.896617889404297
42
0 -98.48269653320312 24.215221405029297
1 -98.6183853149414 24.11323356628418
2 -98.83592224121094 24.083219528198242
3 -99.12969970703125 23.731660842895508
43
0 -99.82538604736328 25.182058334350586
1 -99.67584991455078 24.888042449951172
2 -99.54924011230469 24.851356506347656
3 -99.5304946899414 24.598148345947266
4 -99.55218505859375 24.55371856689453
5 -99.48703002929688 24.475383758544922
6 -99.5032882690429 7 24.35395050048828
7 -99.37472534179688 24.201631546020508
8 -99.12210845947266 24.06793212890625
9 -99.09986114501953 23.8656215667724 6
10 -99.12969970703125 23.731660842895508
44
0 -99.00114440917969 23.31682777404785
1 -98.97016143798828 23.511117935180664
2 -99.12969970703125 23.731660842895508
45
0 -102.95545959472656 23.30149269104004
1 -102.9338150024414 23.60133934020996
2 -102.9615249633789 23.71196746826172
3 -103.21302032470703 24.0
4 -103.40582275390625 24.281028747558594
5 -103.39749145507812 24.38106346130371
6 -103.33726501464844 24.495948791503906
7 -103.3469467163086 24.598241806030273
8 -103.22075653076172 24.685823440551758
9 -103.2825698852539 24.72873878479004
10 -103.2769775390625 24.785429000854492
11 -103.35833740234375 24.810911178588867
12 -103.46866607666016 24.983570098876953
13 -103.56199645996094 25.03872299194336
14 -103.48467254638672 25.355119705200195
15 -103.49382019042969 25.586135864257812
46
0 -104.66083526611328 24.024341583251953
1 -104.37261962890625 23.99486541748047
2 -104.33887481689453 23.899738311767578
3 -104.23825073242188 23.8420467376709
4 -104.1634521484375 23.837915420532227
5 -104.04396057128906 23.75186538696289
6 -103.8519287109375 23.776721954345703
7 -103.66780853271484 23.67523765563965
8 -103.44869995117188 23.663448333740234
9 -103.3384780883789 23.625690460205078
10 -103.22837829589844 23.550045013427734
11 -103.12907409667969 23.533430099487305
12 -102.95545959472656 23.30149269104004
47
0 -110.23724365234375 23.87031364440918
1 -110.03675842285156 23.815990447998047
2 -109.87773895263672 23.700368881225586
3 -109.69879913330078 23.691715240478516
4 -109.66838073730469 23.485506057739258
5 -109.5396499633789 23.4091854095459
6 -109.49009704589844 23.316251754760742
7 -109.43186950683594 23.29551124572754
48
0 -106.16822814941406 23.224891662597656
1 -105.98648834228516 23.329090118408203
2 -105.90454864501953 23.4315185546875
3 -105.92548370361328 23.467355728149414
4 -105.84908294677734 23.53226661682129
5 -105.87178802490234 23.546981811523438
6 -105.83731842041016 23.563587188720703
7 -105.8425521850586 23.632104873657227
8 -105.75765228271484 23.642335891723633
9 -105.67019653320312 23.734167098999023
10 -105.52033233642578 23.750202178955078
11 -105.48424530029297 23.72580909729004
12 -105.24266052246094 23.863384246826172
13 -105.0251235961914 23.881601333618164
14 -104.99124908447266 23.852272033691406
15 -104.9439468383789 23.933324813842773
16 -104.85275268554688 23.93770980834961
17 -104.66083526611328 24.024341583251953
49
0 -106.16822814941406 23.224891662597656
1 -106.17130279541016 23.22785758972168
2 -106.2863540649414 23.37474822998047
3 -106.3225326538086 23.40803337097168
4 -106.5860366821289 23.64920425415039
5 -106.6233139038086 23.87137222290039
6 -106.6844711303711 23.8924503326416
7 -106.83820343017578 24.100831985473633
8 -107.0107421875 24.166582107543945
9 -107.0919418334961 24.296493530273438
10 -107.09757995605469 24.450788497924805
11 -107.29000091552734 24.67160987854004
50
0 -109.43186950683594 23.29551124572754
1 -109.52449035644531 23.14082145690918
2 -109.6731948852539 23.070077896118164
51
0 -109.6731948852539 23.070077896118164
1 -109.71731567382812 22.997831344604492
2 -109.79331970214844 22.943349838256836
3 -109.88160705566406 23.056535720825195
4 -109.98159790039062 23.376068115234375
5 -110.03376770019531 23.454322814941406
6 -110.2120590209961 23.637311935424805
7 -110.23724365234375 23.87031364440918
52
0 -102.6067886352539 22.86092185974121
1 -102.24700927734375 23.367021560668945
2 -101.74700927734375 23.827104568481445
3 -101.67671203613281 23.97753143310547
4 -101.58979797363281 24.022008895874023
5 -101.44953918457031 24.218183517456055
6 -101.37340545654297 24.4580078125
7 -101.35704803466797 24.586679458618164
8 -101.3817367553711 24.617488861083984
9 -101.17000579833984 24.74240493774414
10 -101.07658386230469 24.855642318725586
11 -101.06624603271484 25.038427352905273
12 -101.11184692382812 25.10611343383789
13 -101.09042358398438 25.27130889892578
14 -100.98526763916016 25.423786163330078
53
0 -102.95545959472656 23.30149269104004
1 -102.6067886352539 22.86092185974121
54
0 -100.1446304321289 22.59044075012207
1 -99.72090911865234 22.99807357788086
2 -99.68816375732422 23.164762496948242
3 -99.52647399902344 23.381364822387695
4 -99.38218688964844 23.414674758911133
5 -99.28857421875 23.62146759033203
6 -99.19879150390625 23.620635986328125
7 -99.17501831054688 23.711944580078125
8 -99.12969970703125 23.731660842895508
55
0 -100.98526763916016 25.423786163330078
1 -100.79963684082031 25.430450439453125
2 -100.78533172607422 25.28691864013672
3 -100.81652069091797 25.228416442871094
4 -100.5152587890625 25.000123977661133
5 -100.32122039794922 24.801101684570312
6 -100.26264190673828 24.586952209472656
7 -100.28932189941406 24.499141693115234
8 -100.39363861083984 24.381418228149414
9 -100.3774719238281 2 24.099626541137695
10 -100.6440658569336 23.652328491210938
11 -100.47207641601562 22.963512420654297
12 -100.4053955078125 22.928789138793945
13 -100.24234771728516 22.691181182861328
14 -100.16357421875 22.648630142211914
15 -100.1446304321289 22.59044075012207
56
0 -98.13920593261719 22.552047729492188
1 -98.09541320800781 22.714353561401367
2 -98.10276794433594 22.83111000061035
3 -98.04094696044922 22.871112823486328
4 -98.06747436523438 22.913448333740234
5 -98.00936889648438 23.21078109741211
6 -98.03092193603516 23.462678909301758
7 -98.26515197753906 24.001861572265625
8 -98.38079071044922 24.06028175354004
9 -98.48269653320312 24.215221405029297
57
0 -100.77020263671875 22.397239685058594
1 -100.68870544433594 22.44231605529785
2 -100.642822265625 22.567306518554688
3 -100.52845001220703 22.625837326049805
4 -100.45498657226562 22.603906631469727
5 -100.33978271484375 22.640716552734375
6 -100.26361083984375 22.565732955932617
7 -100.28025817871094 22.433073043823242
8 -100.18157958984375 22.484569549560547
9 -100.1446304321289 22.59044075012207
58
0 -100.95858764648438 22.150075912475586
1 -100.77020263671875 22.397239685058594
59
0 -102.6067886352539 22.86092185974121
1 -101.662353515625 22.588581085205078
2 -101.46702575683594 22.422155380249023
3 -101.16500854492188 22.314558029174805
4 -100.95858764648438 22.150075912475586
60
0 -100.95858764648438 22.150075912475586
1 -100.86759948730469 22.074323654174805
2 -100.79095458984375 21.894302368164062
3 -100.68473052978516 21.752676010131836
4 -100.6839828491211 21.699159622192383
5 -100.72975158691406 21.646310806274414
61
0 -106.16822814941406 23.224891662597656
1 -106.12933349609375 23.187335968017578
2 -106.0928726196289 23.161916732788086
3 -105.84413146972656 22.98541831970215
4 -105.79251861572266 22.836627960205078
5 -105.6044692993164 22.703088760375977
6 -105.59260559082031 22.567676544189453
7 -105.3669662475586 22.49898910522461
8 -105.21295928955078 22.12059211730957
9 -105.21853637695312 21.944927215576172
10 -105.11966705322266 21.832077026367188
11 -105.05359649658203 21.676145553588867
12 -104.9773178100586 21.639310836791992
13 -104.89228057861328 21.493213653564453
62
0 -100.72975158691406 21.646310806274414
1 -100.76134490966797 21.609832763671875
2 -100.75370025634766 21.536937713623047
3 -100.68702697753906 21.37828826904297
4 -100.60530853271484 21.327749252319336
5 -100.57846069335938 21.15589141845703
63
0 -89.56354522705078 20.96061897277832
1 -89.4361801147461 20.903982162475586
2 -89.25377655029297 20.88625144958496
3 -89.21172332763672 20.868940353393555
64
0 -103.401123046875 20.661514282226562
1 -103.28433227539062 20.742765426635742
2 -103.33417510986328 20.81891441345215
3 -103.26213073730469 20.837465286254883
4 -103.1856918334961 20.965343475341797
5 -103.1902084350586 21.210662841796875
6 -103.09886932373047 21.460262298583984
7 -103.10826110839844 21.513769149780273
8 -103.06024932861328 21.61578369140625
9 -103.00123596191406 21.61690902709961
10 -102.98516845703125 21.646087646484375
11 -102.97770690917969 21.785720825195312
12 -102.89657592773438 21.883697509765625
13 -102.87651824951172 22.028127670288086
14 -102.82454681396484 22.135408401489258
15 -102.8905029296875 22.340126037597656
16 -102.72725677490234 22.63717269897461
17 -102.6067886352539 22.86092185974121
65
0 -104.89228057861328 21.493213653564453
1 -104.69812774658203 21.355833053588867
2 -104.66280364990234 21.302743911743164
3 -104.65523529052734 21.19856071472168
4 -104.53585052490234 21.06281089782715
5 -104.38997650146484 21.023954391479492
6 -104.23029327392578 21.078365325927734
7 -103.9741439819336 20.915454864501953
8 -103.8384780883789 20.921545028686523
9 -103.84114837646484 20.890975952148438
10 -103.76957702636719 20.869661331176758
11 -103.64501190185547 20.75267219543457
12 -103.49161529541016 20.7181453704834
13 -103.401123046875 20.661514282226562
66
0 -101.38312530517578 20.645681381225586
1 -101.32845306396484 20.823408126831055
2 -101.33781433105469 20.879549026489258
3 -101.68869018554688 21.118148803710938
4 -101.8945465087890 6 21.356746673583984
5 -102.09103393554688 21.497100830078125
6 -102.21735382080078 21.51581382751465
7 -102.2641372680664 21.642131805419922
8 -102.28752899169922 22.329858779907227
9 -102.37641906738281 22.516996383666992
10 -102.47467041015625 22.540388107299805
11 -102.4793472290039 22.732202529907227
12 -102.6067886352539 22.86092185974121
67
0 -89.21172332763672 20.868940353393555
1 -88.83197784423828 20.712621688842773
2 -88.58587646484375 20.69756507873535
3 -88.50109100341797 20.636905670166016
4 -88.41143798828125 20.627832412719727
5 -88.11685943603516 20.70218849182129
6 -87.9283447265625 20.665300369262695
7 -87.64276885986328 20.88195037841797
8 -87.51373291015625 20.8813419342041
9 -87.14608001708984 21.01816177368164
10 -86.95065307617188 21.14484977722168
11 -86.87403106689453 21.153467178344727
68
0 -100.57846069335938 21.15589141845703
1 -100.44445037841797 21.00667381286621
2 -100.4107666015625 20.91497802734375
3 -100.45613861083984 20.82143783569336
4 -100.24408721923828 20.693626403808594
5 -100.25408172607422 20.57135009765625
69
0 -98.13920593261719 22.552047729492188
1 -98.09376525878906 22.500865936279297
2 -97.98959350585938 22.471405029296875
3 -97.87422943115234 22.337139129638672
4 -97.80486297607422 22.172544479370117
5 -97.6563949584961 21.538063049316406
6 -97.72059631347656 21.264650344848633
7 -97.733642578125 21.068328857421875
8 -97.68749237060547 20.943687438964844
9 -97.55590057373047 20.809070587158203
10 -97.47367095947266 20.553863525390625
70
0 -100.25408172607422 20.57135009765625
1 -100.38563537597656 20.597322463989258
2 -100.6351318359375 20.523988723754883
3 -101.01043701171875 20.50799560546875
4 -101.22809600830078 20.5889949798584
5 -101.29039001464844 20.654287338256836
6 -101.38312530517578 20.645681381225586
71
0 -99.21683502197266 20.483980178833008
1 -99.28591918945312 20.480146408081055
2 -99.31599426269531 20.525039672851562
3 -99.34258270263672 20.574460983276367
4 -99.34333801269531 20.74517822265625
5 -99.22985076904297 20.857513427734375
6 -99.20613098144531 20.999046325683594
7 -99.03311920166016 21.0657901763916
8 -99.00680541992188 21.126605987548828
9 -98.96502685546875 21.13289451599121
10 -98.92668151855469 21.25800323486328
11 -98.9542236328125 21.254196166992188
12 -98.99579620361328 21.36895751953125
13 -98.98230743408203 21.41301918029785
14 -98.90227508544922 21.43268585205078
15 -98.9993667602539 21.51763153076172
16 -98.93707275390625 21.87303352355957
17 -99.0124282836914 21.994199752807617
18 -98.97478485107422 22.05678367614746
19 -98.96803283691406 22.17074966430664
20 -99.08534240722656 22.544601440429688
21 -99.01578521728516 22.61835479736328
22 -99.01811218261719 22.691831588745117
23 -98.96390533447266 22.729034423828125
24 -98.95389556884766 22.806900024414062
25 -99.01751708984375 22.840503692626953
26 -99.02102661132812 22.94092559814453
27 -99.11678314208984 23.116315841674805
28 -99.09803771972656 23.23636817932129
29 -99.00114440917969 23.31682777404785
72
0 -101.38312530517578 20.645681381225586
1 -101.50562286376953 20.46150016784668
73
0 -103.401123046875 20.661514282226562
1 -103.33509826660156 20.630971908569336
2 -103.30232238769531 20.648969650268555
3 -103.2841567993164 20.605377197265625
4 -103.20783996582031 20.565752029418945
5 -103.13758087158203 20.568527221679688
6 -103.0443344116211 20.62670135498047
7 -102.79298400878906 20.542062759399414
8 -102.53390502929688 20.536897659301758
9 -102.41826629638672 20.564647674560547
10 -102.13066101074219 20.439716339111328
11 -102.10057830810547 20.36758804321289
12 -102.03356170654297 20.341768264770508
13 -101.50562286376953 20.46150016784668
74
0 -103.401123046875 20.661514282226562
1 -103.5513916015625 20.42495346069336
2 -103.45087432861328 20.275203704833984
75
0 -86.87403106689453 21.153467178344727
1 -86.84359741210938 21.156888961791992
2 -86.89388275146484 20.83197593688965
3 -87.10062408447266 20.615272521972656
4 -87.28062438964844 20.47867202758789
5 -87.43292236328125 20.2454776763916
76
0 -99.2267837524414 20.213205337524414
1 -99.19342803955078 20.248193740844727
2 -99.20977020263672 20.364768981933594
3 -99.16930389404297 20.44626235961914
4 -99.21683502197266 20.483980178833008
77
0 -100.25408172607422 20.57135009765625
1 -99.74978637695312 20.284482955932617
2 -99.59596252441406 20.237417221069336
3 -99.51145935058594 20.250240325927734
4 -99.3521728515625 20.176902770996094
5 -99.2267837524414 20.213205337524414
78

0 -99.10936737060547 19.805984497070312
1 -99.2215576171875 20.064897537231445
2 -99.2267837524414 20.213205337524414
79
0 -103.45087432861328 20.275203704833984
1 -103.42488098144531 20.221593856811523
2 -103.17111206054688 20.140546798706055
3 -103.0404052734375 20.16670036315918
4 -102.87830352783203 20.14495086669922
5 -102.8317642211914 20.095582962036133
6 -102.71932220458984 20.064245223999023
7 -102.7182388305664 19.989131927490234
8 -102.55269622802734 20.023218154907227
9 -102.4228744506836 19.954713821411133
10 -102.31571197509766 19.95716094970703
11 -102.28990173339844 19.990507125854492
12 -102.18460845947266 19.863182067871094
13 -101.79046630859375 19.799131393432617
80
0 -99.19351959228516 19.413570404052734
1 -99.10420227050781 19.518840789794922
2 -99.21527862548828 19.542396545410156
3 -99.21771240234375 19.78343391418457
4 -99.10936737060547 19.805984497070312
81
0 -90.73926544189453 19.344089508056664
1 -90.69364166259766 19.4439640045166
2 -90.6437759399414 19.75701332092285
3 -90.5402603149414 19.83722496032715
4 -90.4654541015625 19.854726791381836
5 -90.37574005126953 19.815162658691406
6 -90.25191497802734 19.823213577270508
7 -90.22781372070312 20.029460906982422
8 -90.05072021484375 20.320905685424805
9 -90.07521820068336 20.471540451049805
10 -90.04251861572266 20.574766159057617
11 -89.89502716064453 20.6456298828125
12 -89.78489685058594 20.838844299316406
13 -89.6489257812525 20.953683853149414
14 -89.56354522705078 20.96061897277832
82
0 -101.79046630859375 19.799131393432617
1 -101.47267913818336 19.624349594116211
2 -101.23119354248047 19.704282760620117
3 -101.02166748046875 19.687667846679688
4 -100.96308898925781 19.647865295410156
5 -100.83731079101562 19.682077407836914
6 -100.78861236572266 19.629674911499023
7 -100.73139190673828 19.670913696289062
8 -100.70640563964844 19.627231597900039
9 -100.72633361816406 19.582883834838867
10 -100.67932891845703 19.512527465820312
11 -100.55101776123047 19.439210891723633
12 -100.496826171875 19.452835083007812
13 -100.49028015136719 19.491588592529297
14 -100.37702941894531 19.496438980102254
15 -100.34862518310547 19.426845550537109
16 -100.28139495849609 19.45433998107910
17 -100.07939910888672 19.399707794189453
18 -99.87431335449219 19.424066543579102
19 -99.71531677246094 19.288778305053711
20 -99.35968780517578 19.30224609375
21 -99.19351959228516 19.413570404052734
83
0 -97.47367095947266 20.553863525390625
1 -97.4253921508789 20.525909423828125
2 -97.23793029785156 20.587448120117188
3 -97.17060089111328 20.540451049804688
4 -97.16871643066406 20.47327995300293
5 -97.09900665283203 20.456907272338867
6 -97.0291519165039 20.487619400024414
7 -96.98255920410156 20.465518951416016
8 -96.79888916015625 20.264511108398438
9 -96.62389373779297 20.054998397827515
10 -96.62471771240234 19.997724533081055
11 -96.45243072509766 19.856475830078125
12 -96.42369842529297 19.768434524536133
13 -96.35450744628906 19.395088195800781
14 -96.38668823242188 19.359386444091797
15 -96.49320983886719 19.3381404876709
16 -96.48246002197266 19.308132171630859
17 -96.36607360839844 19.260932922363281
18 -96.33944702148438 19.197391510009766
19 -96.1310043334961 19.13465690612793
20 -96.09725189208984 19.103948593139648
21 -96.12701416015625 19.021902084350586
84
0 -104.89228057861328 21.493213653564453
1 -104.90027618408203 21.383428573608398
2 -104.93565368652344 21.333410263061523
3 -104.89377593994141 21.209827423095703
4 -104.96092224121094 21.147600173950195
5 -104.96789550781250 21.152858734130859
6 -105.15679931640625 20.844644546508789
7 -105.01023101806641 20.679756164550781
8 -105.04686737060547 20.514865875244141
9 -105.17511749267578 20.469062805175781
10 -105.13847351074219 20.075160980224609
11 -104.95526885986328 19.791185379028320
12 -104.89114379882812 19.818666458129883
13 -104.79953765869141 19.672098159790039
14 -104.54304504394531 19.470565795898438
15 -104.44227600097656 19.498046875
16 -104.35066986083984 19.397281646728516
17 -104.09417724609375 19.223232269287109
18 -103.97509002685547 19.168268203735352
19 -103.84683990478516 19.049180984497070
20 -103.80115509033203 19.008768081665039
85
0 -87.43292236328125 20.245477676391602
1 -87.58403778076172 20.177200317382812
2 -87.63883209228516 20.018737792968750
3 -87.779296875 19.945472717285156
4 -88.028533935546875 19.605962753295898
5 -88.04049682617188 19.463672637939453
6 -88.13240814208984 19.258794784545898
7 -88.17449951171875 18.956121444702148
8 -88.36308288574219 18.737714767456055
9 -88.4677505493164 18.526744842529297
86
0 -88.4677505493164 18.526744842529297
1 -88.30876922607422 18.498857498168945
87
0 -90.735664367675781 18.615751266479492
1 -90.44006347656250 18.571599960327148
2 -90.34901428222656 18.654247283935547
3 -89.98442840576172 18.700626373291016
4 -89.82604980468750 18.666135787963867
5 -89.45017242431641 18.505243301391602
6 -89.20916748046875 18.450397491455078
7 -88.70173645019531 18.476087570190430
8 -88.46775054931641 18.526744842529297
88
0 -103.80115509033203 19.008768081665039
1 -103.60866546630859 18.858489532470703
2 -103.49874114990234 18.783525466918945
3 -103.28804779052734 18.462907791137695
4 -102.28009796142578 18.144607543945312
89
0 -93.3669204711914 18.000158309936523
1 -93.35328674316406 18.050104141235352
2 -93.39549255371094 18.167369842529297
3 -93.34413090991211 18.232862472534180
4 -93.21426391601562 18.275938034057617
5 -93.11488342285156 18.321666717529297
6 -93.07499694824219 18.408590316772461
7 -92.66696166992188 18.424953460693359
8 -92.43190765338806 18.655305862426758
9 -92.22096252441406 18.616743087768555
10 -91.80078125 18.368707656860352
11 -91.48069763183594 18.403478622436523
12 -91.17185974121094 18.644823074340820
13 -91.11758422851562 19.022382736206055
14 -90.84870147705078 19.168416976928711
15 -90.73926544189453 19.344089508056641
90
0 -93.3669204711914 18.000158309936523
1 -93.23297119140625 18.031494140625
2 -93.15468597412109 17.995742797851562
3 -92.97396850585938 17.995391845703125
91
0 -93.57559204101562 17.986190795898438
1 -93.36692047119141 18.000158309936523
92
0 -94.91053009033203 17.939884185791016
1 -94.94849395751953 18.077737808227539
2 -95.14989471435547 18.127485275268555
3 -95.27522277832031 18.202795028686523
4 -95.32044982910156 18.195468902587891
5 -95.35001373291016 18.255165100097656
6 -95.40853881835938 18.282190856191406
7 -95.32292938232422 18.422756195068359
8 -95.35499572753906 18.40625
9 -95.43688201904297 18.466917037963867
10 -95.43888854980469 18.588508605957031
11 -95.52920532226562 18.634838104248047
12 -95.57472229003906 18.648183822631836
13 -95.86427307128906 18.698785781860352
14 -96.12701416015625 19.021902084350586
93
0 -94.91053009033203 17.939884185791016
1 -94.75673675537109 17.935884475708008
2 -94.64873504638672 17.995174407958984
3 -94.55835723876953 17.989706039428711
4 -94.44979095458984 18.092994689941406
5 -94.13571929931641 17.991004943847656
6 -93.82499694824219 17.969497680664062
7 -93.57559204101562 17.986190795898438
94
0 -92.97396850585938 17.995391845703125
1 -92.85096740722656 17.947151184082031
2 -92.54805755615234 17.685811996459961
3 -92.03568267822266 17.685146331787109
4 -91.83281707763672 17.793981552124023
5 -91.73429870605469 17.963949203491211
6 -91.26213836669922 18.335466384887695
7 -91.02465057373047 18.639890670776367
8 -90.73566436767578 18.615751266479492
95
0 -102.28009796142578 18.144607543945312
1 -102.2437515258789 18.133129119873047
2 -102.13381958007812 18.151449203491211
3 -101.78572082519531 18.123968124389648
4 -101.44678497314453 17.812509536743164
5 -101.15364837646484 17.656780242919922
6 -101.04371643066406 17.537693023681641
7 -100.95211029052734 17.482730865478516
8 -100.88799285888672 17.418607711791992
9 -100.65897369384766 17.345323562622070
10 -100.512405395507812 17.381965637207031
11 -100.44828796386719 17.299520492553711
12 -100.20095062255859 17.152952194213867
13 -100.10018157958984 17.107149124145508
14 -100.05438232421875 17.125469207763672
15 -99.83452606201172 16.988061904907227
16 -99.78825378417969 16.973054885864258
96
0 -94.91053009033203 17.939884185791016
1 -95.06185913085938 17.722455978393555
2 -95.04934692382812 17.391521453857422
3 -95.08525085449219 17.310979843139648
4 -94.98432922363281 17.039844512939453
5 -95.02125549316406 16.789190292358398
6 -94.95346832275391 16.653491973876953
7 -94.94334411621094 16.549036026000977
8 -95.02043151855469 16.429950714111328
97
0 -95.02043151855469 16.429950714111328
1 -94.7871322631836 16.480514526367188
2 -94.61346435546875 16.558389663696289
3 -94.37394714355469 16.483865737915039
4 -94.21722412109375 16.373004913330078
98
0 -95.22793579101562 16.318861007690430
1 -95.02043151855469 16.429950714111328
99
0 -99.78825378417969 16.973054885864258
1 -99.49559020996094 16.878135681152344
2 -99.40398406982422 16.933099746704102
3 -99.32154083251953 16.841493606567383
4 -99.1383285522461 16.868974685668945
5 -98.68030548095703 16.685764312744141
6 -98.04225158691406 16.332382202148438
7 -97.83628845214844 16.292741775512695
8 -97.72063446044922 16.132320404052734
9 -97.31538391113281 16.062849044799805
10 -96.99476623535156 16.081169128417969
11 -96.56421661376953 15.861316680908203
12 -96.38101196289062 15.879637718200684
13 -96.17887878417969 15.879637718200684
14 -96.13547551586914 15.901526451110840
15 -96.07284545898438 15.933111190795898
16 -95.80686187744140 15.903185844421387
17 -95.69635772705078 15.977557182312012
18 -95.63095855712891 15.97562026977539
19 -95.37104797363281 16.112514495849609
20 -95.25 16.179622650146484
21 -95.21418762207031 16.199068069458008
22 -95.22793579101562 16.318861007690430
100
0 -92.00285339355469 15.679814338684082
1 -91.95990753173828 15.749850273132324
2 -91.95658111572266 15.834765434265137
3 -91.98797607421875 15.876571655273438
4 -92.01018524169922 16.050710678100586
5 -92.12388610839844 16.249223709106445
6 -92.30889892578125 16.500331878662109
7 -92.51536560058594 16.571319580078125
8 -92.51871490478516 16.618787765502930
9 -92.63381195068359 16.742097854614258
10 -92.79602050781250 16.745964050292969
11 -92.84206390380859 16.714258193969727
12 -92.95408630371094 16.760284423828125
13 -93.00904083251953 16.702497482299805
14 -93.01667785644531 16.739128112792969
15 -93.15063476562500 16.748275756835938
16 -93.24286651611328 16.801481246948242
17 -93.32888031005859 16.748332977294922
18 -93.42066192626953 16.776559829711914
19 -93.50341796875000 16.740020751953125
20 -93.53736114501953 16.682996749877930
21 -93.71958160400391 16.683010101318359
22 -93.89389038085938 16.534420013427734
23 -94.21722412109375 16.373004913330078
101
0 -92.00285339355469 15.679814338684082
1 -91.97331237792969 15.624648094177246
2 -91.86822509765625 15.609670639038086
3 -91.78380584716797 15.456043243408203
4 -91.60852050781250 15.384429931640625
5 -91.58666229248047 15.338549613952637
102
0 -94.21722412109375 16.373004913330078
1 -94.12313079833984 16.33350944519043
2 -94.01518249511719 16.352426528930664
3 -93.52812957763672 15.887414932250977
4 -93.18031311035156 15.672866821289062
5 -93.01281738281250 15.472093581533203
6 -92.95328521728516 15.476068496704102
7 -92.89689636230469 15.417993545532227
8 -92.66707611083984 15.326546669006348
9 -92.54611968994140 15.173183441162109
10 -92.44663238525390 15.127095222473145
103
0 -92.44663238525390 15.127095222473145
1 -92.36122894287109 15.046827316284180
2 -92.30233001708984 15.073133468627930
3 -92.32604980468750 15.172205924987793
4 -92.27683250856400 15.121696472167969
5 -92.24508666992188 15.132232666015625
6 -92.18874359130859 15.221604347229004
7 -92.22986602783203 15.236886024475098
8 -92.20713043212891 15.258617401123047
9 -92.23545074462891 15.310029983520508
10 -92.11560058593750 15.444288253784180
11 -92.11124420166016 15.519154548645020
12 -92.15285491943360 15.581016540527344
13 -92.14109802246094 15.660291671752930
14 -92.08804321289062 15.700028419494629
15 -92.00285339355469 15.679814338684082
104
0 -92.44663238525390 15.127095222473145
1 -92.36617279052734 14.922773361206055
2 -92.23624420166016 14.887837409973145
3 -92.14919281005859 14.968704223632812
4 -92.09090423583984 14.911367416381836
5 -92.09738159179688 14.708067893981934
6 -91.79783630371094 14.694200515747070
7 -91.77190399169922 14.571367263793945
8 -91.64316558837891 14.573575019836426