IMPLEMENTASI ALGORITMA A*(*STAR*) PADA APLIKASI ANDROID DENGAN MENGGUNAKAN PETA *OFFLINE OPEN STREET MAP*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Aditya Bagus Setiawan
NIM: 135150109111004

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

PENGESAHAN

JUDUL SKRIPSI

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Aditya Bagus Setiawan
NIM: 135150109111004

Skripsi ini telah diuji dan dinyatakan lulus pada
8 Agustus 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Dr.Eng, Herman Tolle, ST., MT	Drs.Marji, MT.
NIK:197408232000121 001	NIK:19670801 199203 1 001

Mengetahui
Ketua Program Studi NamaProgramStudi

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 197105182003121001

PERNYATAAN ORISINALITAS

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 disitasi dalam naskah ini dan disebutkan dalam 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, 18 Agustus 2016

Aditya Bagus Setiawan

NIM: 135150109111004

KATA PENGANTAR

Assalamu'alaikum Wr.Wb.

Segala puji dan syukur penulis panjatkan kehadirat Allah SWT yang telah melipahkan berkat dan anugrahnya sehingga penulis dapat menyelesaikan skripsi yang sekaligus sebagai prasyarat untuk medapat gelar Sarjana Komputer Universitas Brawijaya dengan judul **"Implementasi Algoritma A*(*star*) Pada Aplikasi Android Dengan Menggukan Peta *Offline Open Street Map*".**

Pada penyusunan Skripsi ini tidak semata-mata hasil kerja penulis sendiri, melainkan juga berkat dan bimbingan dan dorongan dari pihak-pihak yang telah membantu, baik secara materi maupun non-materi.

Dalam hal ini penulis mengucapkan rasa terima kasih yang tak terhingga kepada :

1. Allah SWT atas semua karunia yang telah diberikan kepada penulis sehingga penulis dapat menyelesaikan skripsi ini dengan tepat waktu.
2. Bapak Dr. Eng Herman Tolle. S.T., M.T selaku dosen pembimbing I yang telah meberikan arahan dan bimbingan kepada penulis dalam penyusunan skripsi ini.
3. Bapak Drs.Mardji, M.T selaku dosen pembimbing II yang telah meberikan arahan dan bimbingan kepada penulis dalam penyusunan skripsi ini.
4. Bapak Achmad Basuki, S.T., M.MG., Ph.D selaku pembimbing akademik, terima kasih atas semua arahan yang diberikan.
5. Seluruh staf dan karyawan Fakultas Ilmu Komputer Universitas Brawijaya.
6. Bapak Bambang Sugeng Riyadi dan Ibu Sri Waluya Ningsih tercinta yang telah memberikan dukungan dengan sepenuh hati baik moril maupun materiil yang tidak ternilai harganya.
7. Teman-teman SAP 2013.

Harapan penulis, semoga laporan skripsi ini dapat bermanfaat dan dapat lebih disempurnakan lagi.Penulis menyadari bahwa laporan skripsi ini masih banyak terdapat kekurangan dan kelemahan karena keterbatasan kemampuan dan pengetahuan penulis, untuk itu penulis mengucapkan permohonan maaf.

Wassalamu'alaikum Wr.Wb

Malang, 18 Agustus 2016

Penulis

Adityabagussetiawan@outlook.co.id

ABSTRAK

Aditya Bagus Setiawan. 2013. Implementasi Algoritma A*(*star*) Pada Aplikasi Android Dengan Menggukan Peta *Offline Open Street Map*. Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya, Malang. Dosen pembimbing : Dr.Eng, Herman Tolle, S.T., M.T dan Drs. Marji, MT.

Pada era modern saat ini penggunaan perangkat bergerak tidak terbas hanya pada *short message service* (SMS) atau *telephone* saja namun pengguna perangkat bergerak dapat melakukan *explore* lebih jauh lagi pada perangkat bergerak yang dimilikinya salah satunya adalah penentuan posisi atau yang sering kita sebut sebagai *routing*. Fasilitas ini dirasa perlu ketika ingin menemukan jalur menuju tempat yang dituju dengan cepat dengan menggunakan perangkat perangkat bergerak dibandingkan dengan membaca peta.

Aplikasi yang dibangun berbasis *mobile* dengan yang digunakan adalah Android serta pengembangannya menggunakan Bahasa pemrograman Java.Dari kesemuanya diperoleh hasil sebuah aplikasi yang dapat menampilkan rute terpendek menuju tempat yang ingin dituju dengan menggunakan peta *Open Street Map* untuk wilayah Kota Malang.

Pada penelitian ini menggunakan Algoritma A*(*star*) untuk menentukan rute terpendek menuju tempat tujan. Algoritma A*(*star*) dipilih karena dianggap mampu untuk memecahkan masalah perhitungan terpendek dari sebuah rute. Berdasar pengujian yang telah dilakukan Algoritma A*(*star*) akurasinya mencapai 80% sehingga ini dapat digunakan sebagai acuan yang cukup akurat.

Katakunci: Algoritma A*(*star*), *open street map*, perangkat bergerak, rute terpendek

ABSTRACT

*Aditya Bagus Setiawan. 2013. Implementation of Algorithm A * (star) On Android Application Using Offline Map Open Street Map. Program Information Technology and Computer Science, University of Brawijaya, Malang. Supervisor: Dr. Eng Herman Tolle, ST., MT and Drs.Marji, MT.*

In today's modern era use of mobile devices terbas not only on short message service (SMS) or telephone only but mobile users can explore further on its mobile devices one of which is the positioning or what we often refer to as routing. This facility is necessary when you want to find a path to the destination quickly by using mobile devices than reading a map.

Mobile-based applications built with used is Android and its development using the Java programming language. Of all the result of an application that can display the shortest route to where you want to target by using the map Open Street Map for the city of Malang.

*In this study, using Algorithm A * (star) to determine the shortest route towards a destination place. Algorithm A * (star) were selected for being able to solve the problem of the calculation of the shortest route. Based on the testing that was done Algorithm A * (star) reached 80% accuracy so that it could be used as a fairly accurate.*

***Keywords**: Algorithm A * (star), open street map, the device moves, the shortest route*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar belakang

Pada jaman *modern* saat ini penggunaan perangkat bergerak tidak terbatas hanya pada *Short Message Service* (SMS) atau *telephone* saja namun pengguna perangkat bergerak dapat melakukan *explore* lebih jauh lagi pada perangkat yang dimilikinya salah satunya adalah penentuan posisi atau yang sering kita sebut sebagai *routing*. *Routing* disini biasanya digunakan untuk menuntun pengguna perangkat bergerak untuk menuju suatu tempat yang diinginkan dengan menggunakan media peta yang ada pada perangkat bergerak.

Peta pada perangkat bergerak masa kini terdapat dua macam, diantaranya peta *online* dan peta *offline* keduanya memiliki kelebihan dan kekurangan masing-masing. Penggunaan peta *online* saat ini lebih banyak digunakan karena praktis dalam segi penggunaan dan pemasangan seperti *Google Map, navigasi* jadi lebih mudah tanpa memikirkan kesulitan dalam pemasangan dan penggunaanya. Tapi terdapat beberapa masalah yang cukup menggangu dalam penggunaan peta *online* yaitu peta *online* membutuhkan *bandwidth* yang cukup besar untuk menampilkan peta pada perangkat bergerak dan penggunaan jaringan internet secara terus-menerus sehingga memerlukan biaya yang mahal dalam penggunaan data dan dapat mengguras daya baterai perangkat dengan cepat.

Pada saat ini penggunaan fasilitas *routing* pada perangkat bergerak masih menggunakan mode *online routing* dimana penggunaan mode *online routing* memiliki kekurangan yang cukup signifikan dari segi pemakaian kuota *bandwith* dan dari segi daya penggunan baterai dari perangkat tersebut, tidak jarang para pengguna fasilitas ini hampir tidak pernah menggunakannya sekalipun mereka memerlukannya.

Navigasi atau *routing* merupakan penentu arah kedudukan (*position*) dan arah perjalanan baik pada medan yang sebenarnya atau pada peta, dan oleh sebab itulah pengetahuan tentang pedoman arah (*compass*) dan peta beserta teknik penggunaannya harus dimiliki dan dipahami (Budi, 2009).

Sistem *routing* pertama kali dikenalkan tahun 2004 oleh tim peneliti yang terdiri dari Ryujiro Fujita, Hiroto Inoue, Naohiko Ichihara, Takehiko Shioda. Penelitian yang melibatkan tim peneliti tersebut, didanai oleh pemerintah Jepang yang bertujuan untuk mengatasi kepadatan lalu lintas yang terjadi di beberapa kota besar di Jepang (Setiawan, 2015). Sistem ini berfungsi untuk mencari semua rute perjalanan yang dapat dilalui oleh seorang pengendara kendaraan dalam mencapai tujuan. Sistem yang diperkenalkan oleh peneliti ini dahulunya menggunakan algoritma Dijkstra yang memiliki kekurangan dalam waktu pencarian yang kurang efektif dan dinilai lamban.

Open Street Map (OSM) merupakan sebuah proyek kolaborasi pembuatan peta dunia yang bebas disunting oleh siapapun. OSM berada di bawah lisensi *Open Data Commons Open Data License* (ODbl) sehingga kita bebas berbagi, menyalin,

mendistribusikan dan menggunakan *Databasenya*. (Open Street Map Indonesia 2014).

Untuk mengatasi permasalahan, pada penelitian ini akan dibahas penerapan sistem *routing* secara *offline* pada peta *Open Street Map* dengan menggunakan Algoritma A*(*star*) sebagai alternatif untuk mempercepat proses pencarian rute. Sistem *routing* yang dibangun oleh peneliti untuk diterapkan ke dalam aplikasi Pengembangan Modul *Routing* Peta *Offline Open Street Map* pada aplikasi Android menggunakan metode A*(*star*).

Pemilihan Algoritma A*(*star*) untuk penelitian ini dikarenakan Algoritma A*(*star*) ini dianggap akurat dalam penyelesaian masalah *routing* dan Algoritma A*(*star*) juga dapat dipadupadankan dengan peta *offline open street map,* selain itu manfaat yang bisa didapat adalah optimalisasi peta *offline* pada perangkat bergerak sehingga tidak mengguras daya baterai hingga *bandwith* dan sekaligus dapat membantu pengguna perangkat *mobile* menemukan jalur tercepat yang ingin ditempuh untuk bisa mencapai suatu tempat yang ingin dituju melalui bantuan *routing* pada perangkat *mobile* yang dimiliki.

1.2 Rumusan masalah

Melalui latar belakang yang telah diuraikan sebelumnya, maka dapat dirumuskan permasalahan sebagai berikut :

1. Bagaimana penerapan Algoritma A*(*star*) pada modul *routing* peta *offline*?
2. Bagaimana rancangan dan implementasi modul *routing*dengan menggunakan metode A**(star)* pada peta *offline open street map* pada aplikasi *Location Based Service* Android?
3. Bagaimana tingkat akurasi hasil implementasi metode Algoritma A*(*star*)?

1.3 Tujuan

Adapun tujuan yang ingin dicapai pada penelitian ini adalah :

1. Menerapkan Algoritma A*(*star*) pada pengembangan modul *routing* peta *offline.*
2. Menerapkan modul *routing* pada peta *offline open street map* pada aplikasi *Location Based Service* Android.

1.4 Manfaat

Manfaat yang diperoleh dari penelitian ini adalah :

1. Optimalisasi penggunaan peta *offline* pada perangkat bergerak.

2. Dapat membantu pengguna perangkat *mobile* menemukan jalur tercepat yang ingin ditempuh untuk bisa mencapai suatu tempat yang ingin dituju.

1.5 Batasan masalah

Adapun batasan-batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Aplikasi dibuat pada *platform* Android.
2. Aplikasi dibangun dengan menggunakan layanan berbasis lokasi.
3. Aplikasi ini menggunakan peta *offline opensource* yaitu Open Street Map (OSM).
4. Implementasi aplikasi menggunakan studi kasus wilayah Kota Malang.

1.6 Sistematika pembahasan

Pembuatan tugas akhir ini dilakukan dengan sistematika sebagai berikut :

1. BAB 1 PENDAHULUAN

Bab ini berisi latar belakang penelitian, rumusan masalah, batasan masalah. Tujuan penelitian, manfaat penelitian dan sistematika penulisan

2. BAB II TINJAUAN PUSTAKA

Bab ini berisi tentang teori dari berbagai pustaka yang menunjang untuk penelitian ini. Teori yang digunakan dalam penelitian ini antara lain *location based service* (LBS), Algoritma A*(*star*), Sistem operasi android, *OpenStreetMap* (OSM), SQLite, *Unified Modelling Language* (UML) dan pengujian perangkat lunak.

3. BAB III METODOLOGI

Bab ini berisi tentang metodologi atau alur perancangan sistem perangkat lunak yang digunakan dalam penelitian ini meliputi analisis perancangan sistem, rancang antar muka aplikasi dan rancangan penelitian.

4. BAB IV ANALISIS DAN PERANCANGAN

Berisi tentang analisa seluruh kebutuhan sistem untuk memastikan bahwa sistem yang dibuat bisa memenuhi seluruh kebutuhan dari pengembangan modul *routing* pada peta *offline open street map* dengan menggunakan Algoritma A*(*star*) ini.

5. BAB V IMPLEMENTASI DAN PEGNUJIAN

Bab ini berisi tentang segala hal yang berkaitan dengan implementasi sistem perangkat lunak yang digunakan untuk penelitian, meliputi implementasi kode program dan antar muka. Pada bab ini pula berisi tentang prosedur pengujian dari aplikasi yang dibangun beserta hasil analisa dari seluruh proses pengujian yang dilakukan terhadap aplikasi yang telah dibangun.

6. BAB VII KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dan saran yang bermanfaat dari aplikasi untuk pengembangan penelitian selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian pustaka

Penerapan peta *offline* pada aplikasi Android sebelumnya pernah di bangun oleh Vika Novita Novita sari dengan judul Rancang Bangun Aplikasi Pencarian Rute Terpendek untuk Menemukan SPBU Terdekat di Kota Malang dengan Menggunakan Algoritma Genetik Berbasis Android (Vika,2010). Peran Algoritma Genetika disini adalah sebagai penentu rute terpendek dengan mengedepankan penggaruh probabilitas *crossover* (Pc) dan probabilitas *mutasi* (Pm), selain itu Algoritma Genetika juga akan menentukan bagaimana penggaruh iterasi atau generasi dalam Algoritma Genetika dapat menentukan rute terpendek menuju SPBU terdekat.

Penerapan Algoritma A*(*star*) pernah diterapkan oleh Didik Setiawan dengan judul Perancagan dan Implementasi Algoritma A*(star) pada Aplikasi Angkot-Finder di Kota Bandung Untuk *Smartphone* Berbasis Android (Didik,2015). Dalam kasus ini penggunaan Algoritma A*(*star*) digunakan untuk membantu pengguna *smartphone* untuk menemukan jalur angkot yang tersebar di wilayah Kota Bandung dan digunakan pula untuk merekomendasikan pengguna aplikasi untuk untuk angkot dan jalur yang digunakan agar sampai ke tujuan dengan cepat dan biaya yang ringan.

Penerapan Algoritma A*(star) selanjutnya adalah pada penelitian dengan judul Perancangan Aplikasi Dalam Implementasi *Shortest Path Finder* Di Kawasan Kampus IT Telkom Berbasis Android Dengan Algoritma A*(*star*) (Achmad,2013). Dalam penelitian yang ditulis oleh Achmad Marizki Chilmi ini peran Algoritma A*(star) adalah sebagai penentu jalur tercepat untuk pencarian suatu tempat di kawasan kampus IT Telkom Bandung dan akurasi dari penerapan Algoritma ini mencapai 80%.

Tabel 2.1 Penelitian terkait

No	Nama	Tahun penelitian	Perbandingan	
			Peneliti	Penulis
1	Vika novitasari	2010	Menggunakan Algoritma Genetika untuk menentukan SPBU terdekat di seputaran Kota malang dan masih bersifat *online*.	Menggunakan Algoritma A*(star) yang berfungsi sebagai penentu jalur tercepat untuk menuju suatu tempat dan bersifat *offline*.
2	Didik setiawan	2015	Ruang lingkup yang digunakan hanya sebatas pada jalur angkot saja dan pengguannya masih bersifat *online routing*.	Ruang lingkup yang digunakan diseluruh jalanan di Kota Malang dan bersifat *offline routing.*

Tabel 2.2 Penelitian terkait (lanjutan)

No	Nama	Tahun penelitian	Perbandingan	
			Peneliti	Penulis
3	Achmad Marizki	2013	Ruang lingkup yang digunakan hanya sebatas pada kampus IT Telkom Bandung saja dan penggunanya bersifat*online routing.*	Ruang lingkup yang digunakan diselruh jalanan di Kota Malang dan bersifat *offline routing.*

2.2 Algoritma A*(*star*)

Algoritma A*(*star*) pertama kali dikenalkan pada tahun 1968 oleh Peter Hart, Nils Nilson dan Bertram Raphael (Hart et al., 1968). Dalam ilmu komputer A* (sering diucapkan dengan A *star*) merupakan salah satu algoritma pencarian *graph* terbaik yang mampu menemukan jalur dengan biaya pengeluaran paling sedikit dari titik permulaan yang diberikan sampai titik tujuan yang diharapkan (Pearl, 1984).

Algoritma ini menggunakan fungsi *distance - plus – cost* (biasanya dinotasikan dengan f(x)) untuk menentukan urutan kunjungan pencarian node di dalam tree. Gabungan jarak – plus – biaya merupakan penjumlahan dari dua fungsi yaitu fungsi *path – cost* (selalu dinotasikan dengan g(x), yang dimungkinkan bernilai *heuristic* ataupun tidak), dan sebuah kemungkinan penerimaan atas perkiraan heuristis jarak ke titik tujuan (dinotasikan dengan h(x)). Fungsi *path – cost* g(x) adalah jumlah biaya yang harus dikeluarkan dari node awal menuju node yang akan dituju. Dengan h(x) bagian dari fungsi f(x) yang harus dapat heuristis, yang mana tidak diperbolehkan untuk terlalu jauh memperkirakan jarak ke arah tujuan. Oleh kerena itu aplikasi seperti *routing*, h(x) mungkin mewakili garis lurus jarak ke titik tujuan (Setiawan, 2015).

Beberapa terminology dasar yang terdapat pada algoritma A*(*star*) adalah (Setiawan, 2015) :

1. Simpul (*node*) adalah petak-petak kecil sebagai representasi dari area pencarian (*path finding*).
2. Simpul asal/mulai (*source node*) adalah sebuah terminologi untuk posisi awal sebuah benda.
3. Simpul ahir/tujuan (*destination node*) adalah tempat tujuan yang ingin dicapai pada pencarian.
4. Simpul sekarang (*current node*) adalah simpul terbaik sebelumnya yang dipilih dan menjadi titik acuan untuk membangkitkan simpul tetangga (*adjacent*).

5. Simpul tetangga (*neighbor node*) adalah simpul-simpul yang bertetangga dengan *current node*.
6. *Open list* adalah tempat menyimpan data simpul yang diakses dari simpul asal (*source node*) atau dari tetangga simpul sekarang (*current node*) yang belum pernah berada di *open list* maupun *closed list*.
7. *Cosed list* adalah tempat menyimpan sampul yang pernah menjadi sampul sekarang (*current node*).
8. *Came from* adalah tempat yang menyimpan data ketetanggaan dari suatu simpul, misalnya y *came from* x artinya *neighbor node*y dari *current node* x.
9. *Walkability* adalah sebuah atribut yang menyatakan apakah sebuah simpul dapat atau tidak dilalui oleh *current node*.

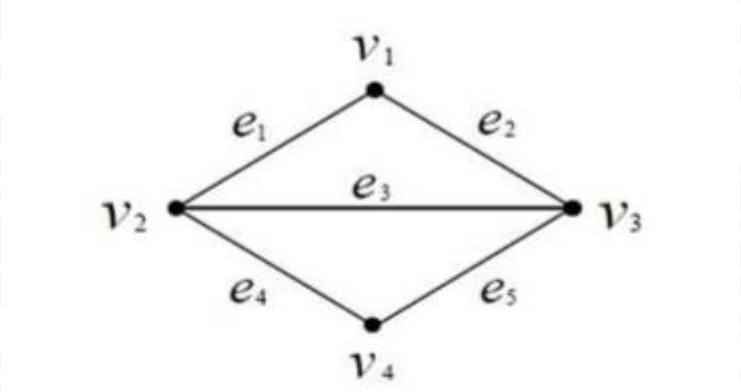
$$f(\times)=g(\times)+h(\times) \qquad (2.1)$$

Pada persamaan (2.1) tersebut, g(x) adalah biaya sebenarnya (*actual cost*), biasanya berupa jarak yang pernah ditempuh. Tapi dalam kasus tertentu g(x) dapat berupa penjumlahan antara jarak dengan biaya medan (*terrain cost*). Sedangkan h(x) adalah biaya heuristiknya (*heuristic cost*) berupa perkiraan jarak dari simpul sekarang (*current node*) ke simpul tujuan (*destination node*). Dan f(x) adalah total biaya dari biaya sebenarnya dan biaya *heuristik.*

Pada persamaan (2.1) diatas algoritma ini menggunakan fungsi biaya *heuristic* simpul x untuk menentukan urutan yang akan dilintasi dalam prioritas antrian (S.Rabin, 2000). Fungsi biaya dapat dibagi menjadi dua bagian, yaitu fungsi biaya jalan terahir dan fungsi biaya jalan masa depan (S.Rabin, 2000).

2.2.1 Graf

Graf adalah kumpulan dari simpul dan garis dimana pasang-pasangan simpul tersebut dihubungkan oleh segmen garis, simpul ini bisa disebut *vertex* dan segmen garis disebut *edge* (setiawan, 2015). Graf dapat digunakan untuk mempresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut. Seringkali graf digunakan untuk mempresentasikan suatu jaringan, misalkan jaringan jalan raya dimodelkan graf dengan kota sebagai simpul (*vertex*) dan jalan yang menghubungkan setiap kota sebagai sisi (*edge*) yang bobotnya (*weight*) adalah panjang jalan tersebut.



Gambar 2.1 Graf sederhana

Sumber : Setiawan (2014)

Gambar 2.1 diatas merupakan gambar graf sederhana yang memiliki dua komponen, yaitu *vertex*dan *edge*, sebuah graf G = (v,e). Terdapat beberapa istilah yang berkaitan dengan graf yaitu :

1. *Vertex* adalah himpunan simpul/titik pada sebuah graf.
2. *Edge* adalah himpunan garis yang menghubungkan tiap simpul/*vertex.*
3. *Adjacement* adalah sebuah titik yang berdekatan, dua buah titik dikatakan berdekatan jika dua buah titik tersebut terhubung dengan sebuah sisi.
4. *Path* adalah *walk* dengan setiap *vertex* yang berbeda. Sebuah *walk* didefinisikan sebagai urutan (tidak nol) *vertex* dan *edge*.
5. *Cycle* adalah siklus atau sirkuit lintasan yang berawal dan berahir pada simpul yang sama.

2.2.2 Shortest path

Dalam jurnal Achmad Mirizki Chilmi disebutkan bahwa lintasan terpendek (*shortest path*) merupakan lintasan minimum yang diperlukan untuk mencapai suatu titik dari titik tertentu. Dalam pencarian lintasan terpendek masalah yang akan dihadapi adalah mencari lintasan mana yang akan dilalui sehingga didapat lintasan terpendek dari suatu vertex yang lain. Ada beberapa maca persoalan lintasan terpendek diantaranya (Achmad,2013) :

1. Lintasan terpedek antara dua buah verteks.
2. Lintasan terpendek antara semua pasangan verteks.
3. Lintasan terpendek dari verteks tertentu ke semua verteks yang lain.
4. Lintasan terpendek antara dua buahverteks yang melalui beberapa verteks tertentu.

2.3 *SQLite*

SQLite merupakan sebuah proses *library* yang mengimplementasikan penyimpanan mandiri, *serverless*, tidak ada konfigurasi, mesin database SQL transaksional. Kode untuk *SQLite* dalam domain publik dan dengan demikian bebas untuk digunakan untuk tujuan apapun baik komersial maupun pribadi. SQLite sekarang ini termasuk yang banyak ditemukan dalam sebuah projek kecil maupun projek besar. Tidak seperti kebanyakan database SQL lainnya, *SQLite* memiliki proses server terpisah, SQLite membaca dan menulis langsung ke *file disk* biasa. Sebuah SQL database lengkap dengan beberapa *table, index, triggers,* dan *view* yang terkandung dalam sebuah *file disk* tunggal (Alfan, 2013).

Secara umum sebuah *SQLite* memiliki delapan fitur diantaranya (Alfan, 2013) :

1. *Serverless*, SQL tidak memerlukan proses pada *server* atau sistem untuk menjalankannya melainkan hanya sebuah file yang diakses oleh *library* SQLite
2. *Zero configuration,* tidak ada *server* berarti tidak perlu melakukan *setup.*
3. *Cross platform,* semua instan *database* berada dalam sebuah file yang *cross-platform* dan tidak memerlukan administrasi.
4. *Self-contained,* sebuah *library* mengandung keseluruhan dari sistem database, yang langsung terintergrasi pada sebuah aplikasi program.
5. *Small runtime footpoin,* untuk membangun database *SQLite* hanya membutuhkan kurang dari satu *megabyte library* (kode program) dan hanya membutuhkan beberapa *megabyte memory* saja.
6. *Transactional, SQLitetransactional* memperbolehkan aksi penyimpanan melalui beberapa proses *thred.*
7. *Full featured, SQLite*mendukung hampir sebagaian besar *standart* SQL92 (SQL2).
8. *Highly reliable,* tim pengembang *SQLite* mengembangkan melalui kode program yang telah melewati proses *testing*.

2.4 *Location Based Service* (LBS)

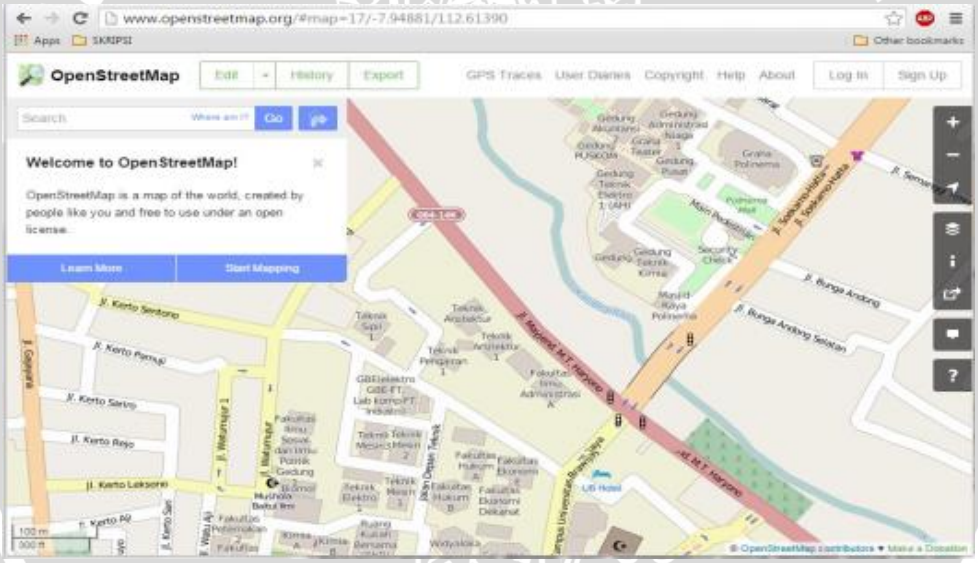
Location Based Service merupakan sebuah layanan informasi yang dapat diakses dengan perangkat bergerak melalui jaringan dan mampu menampilkan posisi secara geografis dimana keberadaan perangkat bergerak tersebut.(Wiliam, 2003).

Ada dua unsur utama dari *location bases service* diantaranya adalah (Wiliam, 2003):

1. *Location manager*, merupakan penyedia perangkat bagi sumber atau *source* untuk *location bases service*.
2. *Location provider*, menyediakan teknologi pencarioan lokasi yang digunakan oleh perangkat.

2.5 *Open Street Map* (OSM)

Open Street Map (OSM) adalah sebuah alat untuk membuat dan berbagi informasi dalam bentuk peta.Siapapun dapat berkontribusi untuk OSM, dan ribuan orang bebas menambahkan proyek setiap harinya. Pengguna dapat memanfaatkan OSM yang diberikan ijin untuk menambahkan detil informasi ke dalam peta. OSM bersifat digital, yang membuatnya sangat berguna untuk pengembangan aplikas LBS (HOT, 2012). Berikut tampilan peta digital dari *Open Street Map* yang ditunjukan pada Gambar 2.2 dibawah ini

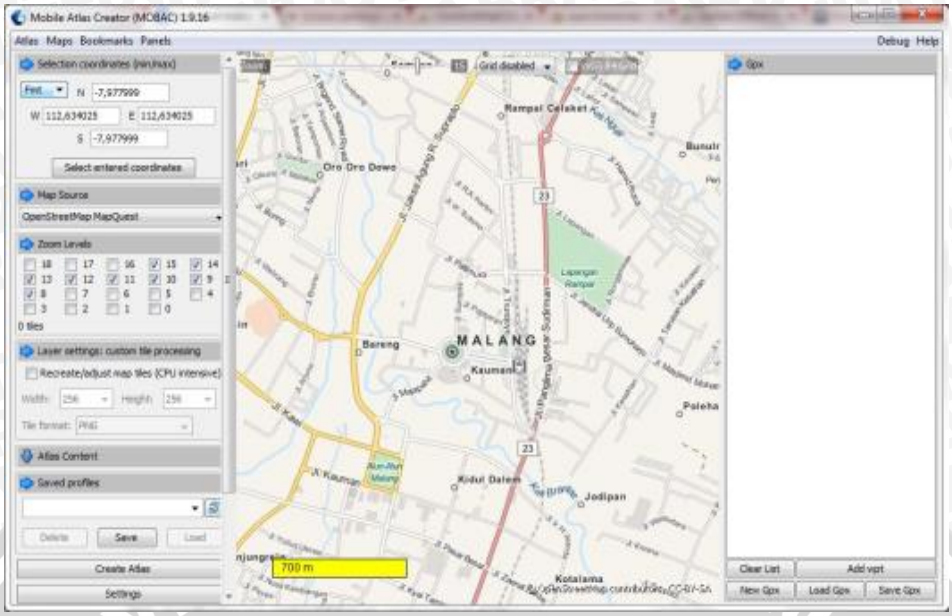


Gambar 2.2 Peta digital *Open Street Map*

Sumber :*Open Street Map* (2012)

2.6 *Mobile Atlas Creator* (MOBAC)

Mobile Atlas Creator (Mobac) adalah sebuah program atau aplikasi yang *open source* (GPL) yang dapat membuat peta secara *offline* untuk dipergunakan dan diterapkan pada aplikasi *mobile* sebagai penentu posisi. Seperti aplikasi *TrekBuddy*, dan *AndNav*.Dengan menggunakan MOBAC peta dapat disimpan sebagai satuan gambar PNG, File Zip, *SQLite*, dll.Sebagai sumbernya, MOBAC menggunakan sebagian besar peta *online* yang *open source* seperti *Open Street Map* dan penyedia peta *online* lainnya (Mobac, 2008). Tampilan aplikasi Mobac, ditunjukan pada Gambar 2.3



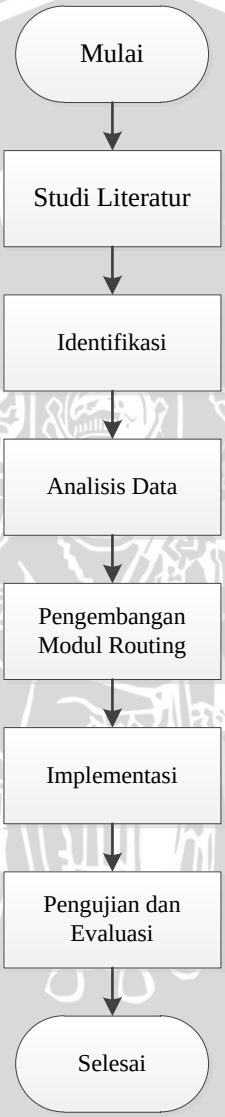
Gambar 2.3 Tampilan aplikasi MOBAC

2.7 OSM Droid *Library*

Osmdroid-android pada dasarnya adalah sebagai penganti kelas pada google *mapview*, sebuah *library* yang dapat membantu menampilkan peta pada *mapview* dalam memanfaatkan peta *offline* khususnya peta *offline* dari *Open Street Map* yang diterapkan pada aplikasi android (Kergall, 2014).

BAB 3 METODOLOGI

Metodologi yang digunakan untuk membangun aplikasi ini terdiri dari beberapa tahap diantaranya studi literatur, identifikasi, analisa data, pengembangan modul, implementasi, pengujian dan evaluasi. Gambar 3.1 dibawah ini merupakan diagram alir dari metodologi yang akan digunakan pada penelitian ini.



Gambar 3.1 Diagram alir metode penelitian

3.1 Studi Literatur

Tahapan yang pertama adalah Studi Literatur, dalam tahapan ini dilakukan untuk memperdalam dan memahami tentang teori dan konsep yang akan digunakan untuk pembuatan aplikasi "Pengembangan Modul *Routing* Peta *Open*

Stret Map pada Aplikasi Android Menggunakan Metode Algoritma A*(*star*)".

Bebrapa metode yang dipelajari dalam Studi Literatur ini adalah :

1. Algoritma A*(*star*).
2. Pemrograman Android.
3. *SQLite*.
4. Peta *offline Open Street Map* (OSM).
5. *Location Based Service* (LBS).

3.2 Identifikasi

Tahap yang kedua adalah identifikasi. Tahapan ini digunakan untuk mengkaji dan memberikan batasan masalah yang akan dibahas sekaligus yang akan diimplementasikan dalam aplikasi yang akan dibangun, dalam tahap ini pula dilakukan pengumpulan data yang berkaitan dengan konsep aplikasi yang akan dibuat. Dalam fase identifikasi ini akan menentukan keberhasilan dari setiap tahapan-tahapan berikutnya.

3.3 Analisis data

Tahap yang ketiga adalah analisis data.Tahap ini merupakan tahapan yang dilakukan pada semua data yang telah terkumpul. Setelah semua data terkumpul maka akan dilakukan pengujian terhadap sampel data yang ada dengan menggunakan Algoritma A*(*star*) untuk memecahkan permasalahan, selain melakukan pengujian menggunakan Algoritma A*(*star*) terhadap data sampel dilakukan juga analisis terhadap data jalan yang akan dilalui dikarenakan tidak semua data jalan akan digunakan. Pemilihan data jalan dikalukan berdasarkan fungsi dan lebar jalan. Penjelasan masing-masing jalan berdasar UU no.38 Tahun 2004 yaitu sebagai berikut (Novitasari,2014).

Jalan Arteri Primer adalah ruas jalan yang menghubungkan antar kota yang berdampingan atau menghubungkan kota jenjang kesatu dengan kota jenjang kedua.

Jika ditinjau dari peranan jalan maka persyaratan yang harus dipenuhi oleh jalan Arteri Primer adalah :

a. Kecepatan rencana > 60 km/jam.
b. Lebar badan jalan > 8,0 m.
c. Kapasitas jalan lebih besar dari volume lalu-lintas rata-rata.
d. Jalan masuk dibatasi secara efisien sehingga kecepatan rencana dan kapasitas jalan dapat tercapai.
e. Tidak boleh tertanggu oleh kegiatan lokal dan lalu lintas lokal.

f. Jalan primer tidak terputus walaupun memasuki kota.

Jalan Arteri Sekunder adalah ruas jalan yang menghubungkan kawasan perimer dengan kawasan sekunder kesatu atau menghubungkan kawasan sekunder kesatu dengan kawasan sekunder lainnya.

Jika ditinjau dari peranan jalan maka persyaratan yang harus dipenuhi oleh jalan Arteri Sekunder adalah :

a. Kecepatan rencana > 30 km/jam.
b. Lebar jalan > 8,0 m.
c. Kaqpasitas jalan lebih besar atau sama dari volume lalu lintas rata-rata.
d. Tidak boleh diganggu oleh lalu lintas lambat.

Jalan Kolektor Primer adalah ruas jalan yang menghubungkan antar kota kedua dengan kota jenjang kedua, atau kota jenjang kesatu dengan kota jenjang ketiga.

Jika ditinjau dari peranan jalan maka persyaratan yang harus dipenuhi oleh jalan Kolektor Primer adalah :

a. Kecepatan rencana > 40 km/jam.
b. Lebar badan jalan > 7,0 m.
c. Kapasitas jalan lebih besar atau sama dengan volume lalu lintas rata-rata.
d. Jalan masuk dibatasi secara efisien sehingga kecepatan rencana dan kapasitas jalan tidak terganggu.
e. Tidak boleh terganggu oleh kegiatan lokal dan lalu lintas lokal.
f. Jalan kolektor primer tidak terputus walaupun memasuki daerah kota.

Jalan Kolektor Sekunder adalah ruas jalan yang mengubungkan kawasan sekunder kedua dengan kawasan sekunder lainnya atau menghubungkan kawasan sekunder kedua dengan kawasan sekunder ketiga.

Jika ditinjau dari peranan jalan maka persyaratan yang harus dipenuhi oleh jalan Kolektor Sekunder adalah :

a. Kecepatan rencana > 20 km/jam.
b. Lebar jalan > 7,0 m.

Jalan Lokal Primer adalah ruas jalan yang menghubungkan kota jenjang kesatu dengan persil, kota jenjang kedua dengan persil, kota jenjang ketiga dengan kota jenjang ketiga lainnya, kota jenjang ketiga dengan kota jenjang dibawahnya.

Jika ditinjau dari peranan jalan maka persyaratan yang harus dipenuhi oleh Jalan lokal Primer adalah :

a. Kecepatan rencana > 20 km/jam.
b. Lebar badann jalan > 6,0 m.
c. Jalan lokal primer tidak terputus walaupun memasuki desa.

Data jalan yang diperolah akan disimpan dalam bentuk mode pada database SQlite. Node akan ditentukan dari percabangan/persimpangan jalan bukan dari nama jalan, hal tersebut dilakukan karena agar perhitungan total jarak sesuai dengan jalan yang dilalui.

3.4 Pengembangan modul *routing*

Tahap yang ke empat adalah Pengembangan modul *routing*. Tahap ini merupakan tahap dimana semua data yang diperlukan telah terkumpul dan dinyatakan biasa digunakan dalam pembangunan aplikasi. Pengembangan *routing* ini dilakukan dengan menggunakan Algoritma A*(*star*) sebagai solusi pemecahan masalah dalam pemilihan jalur tercepat dan terbaik dari sekian banyak jalur yang muncul. Penggunaan Algoritma A*(*star*) dirasa cukup membantu dikarenakan hasil ahir yang dihasilkan sangat akurat. Pada tahap ini akan lebih banyak membahas bagaimana mengolah data dan bagaimana sistem bekerja sesuai dengan inputan yang diterima dan hasil dari output yg menghasilkan jalur yang optimum.

Peranan Algoritma A*(*star*) disini guna menghasilkan rute terpendek untuk menuju lokasi yang dituju dengan cara menggolah data yang telah tersimpan pada database SQlite. Semua data yang disimpan akan diolah sesuai dengan tahap-tahap pada Algoritma A*(*star*) sehingga membentuk sebuah rute yang digambarkan pada peta *Offline Open Street Map* yang dicuplik untuk wilayah Kota Malang saja.

3.5 Implementasi

Tahap yang ke lima adalah implementasi. Dalam tahap ini apabila semua data yang diperoleh telah diproses dan pemecahan terhadap masalah yang muncul telah didapat makan akan masuk pada fase implementasi. Pada tahap ini aplikasi yang diimplentasikan berdasar pada tahap desain yang telah disetujui, pada tahap ini pula akan diberlakukan beberapa hal diantaranya *input, proses,* dan *output* yang sesuai dengan masukan yang diberikan oleh *user* dan yang telah diproses oleh aplikasi.

Aplikasi akan dibuat menggukana *software Eclipse* dengan Bahasa pemrograman *Java* dan menggunakan android SDK untuk *Windows*. Android SDK

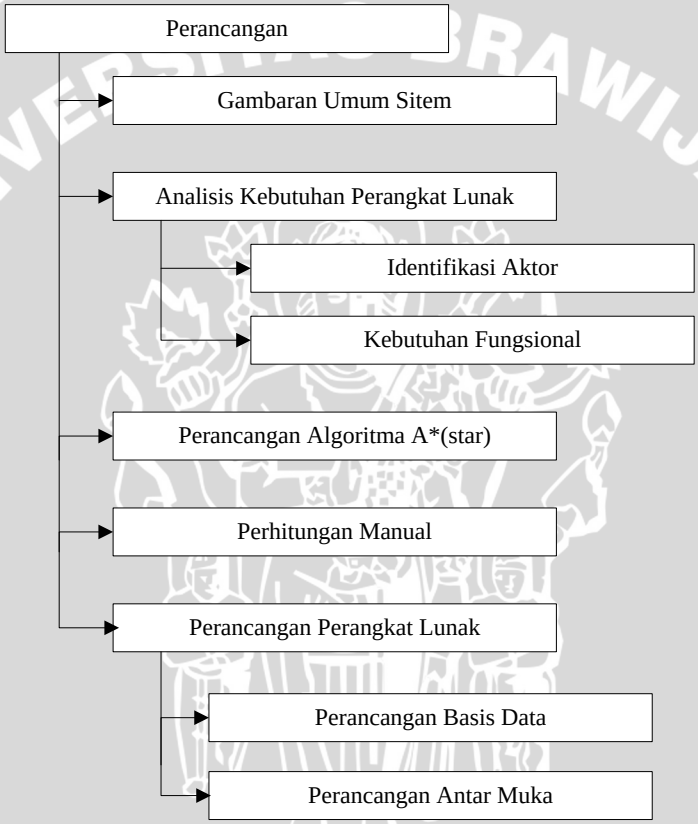
merupakan sebuah *Software Development Kit* untuk pengembangan aplikasi berbasis android yang digunakan untuk *Eclipse*.

3.6 Pengujian dan evaluasi

Tahap terahir adalah tahap pengujian dan evaluasi. Tahapan ini dilakukan jika semua tahapan sudah terpenuhi dan menghasilkan sebuah aplikasi tertentu. Pengujian dan evaluasi ini bertujuan untuk mengetahui kesalahan dan kekurang yang terdapat pada aplikasi yang telah dibangun. Pengujian yang dilakukan menggunakan *black-box testing* untuk memstikan bahwa aplikasi yang telah dibangun telah memenuhi seluruh kebutuhan yang telah didefinisikan pada bab perancangan. Pengujian juga dilakukan pada tahapan Algoritma A*(*star*) untuk menguji kesesuaian antara perhitungan yang dihasilakan oleh aplikasi sama dengan perhitungan yang dilakukan secara manual.

BAB 4 PERANCANGAN

Pada bab ini akan dibahas mengenai analisis kebutuhan dan perancangan terhadap sistem secara keseluruhan. Sebelum masuk pada tahap perancangan akan terlebih dahulu dilakukan analisis kebutuhan yang terdiri dari aktor dimana nantinya aktor ini akan berinteraksi secara langsung dengan aplikasi yang dibangun, daftar kebutuhan yang diperlukan dalam perhitungan manual Algoritma A*(*star*) yang diimplementasikan pada aplikasi. Selanjutnya seluruh kebutuhan tersebut akan dibentuk dalam suatu perancangan perangkat lunak yang terdiri dari perancangan basis data dan perancangan desain antar muka. Alur perancangan sistem ditunjukkan oleh Gambar 4.1**.**



Gambar 4.1 Diagram pohon perancangan

4.1 Gambaran umum sistem

Gambaran umum sistem merupakan sebuah representasi desain arsitektur yang dibuat secara umum. Pada tahapan ini merupakan awal dari perancangan sistem yang akan dibangun, gambaran umum sistem ditunjukkan pada Gambar 4.2.

Gambar 4. 2 Gambaran umum sistem

Pengguna masuk pada aplikasi, setelah masuk pengguna akan dihadapkan pada tiga menu utama diantaranya menu mulai, menu tentang, dan menu *exit*. Setelah pengguna memilih menu mulai maka pengguna melakukan set posisi, set posisi disini dilakukan secara manual dengan cara memilih marker jalan yang diinginkan pengguna yang ditampilkan pada peta *Open Street Map*. Marker jalan yang muncul pada aplikasi merupakan marker jalan yang telah tersimpan pada basis data *SQLite* dalam bentuk titik *longitude* dan *latitude*. Set posisi pengguna ini yang akan diproses lebih lanjut dengan menggunakan Algoritma A*(*star*). Hasil dari perhitungan akan ditampilkan pada layar perangkat bergerak berupa rute jalan menuju tempat yang ingin dituju di seputaran wilayah Kota Malang.

4.2 Analisis kebutuhan perangkat lunak

Perancangan aplikasi ini berupa aplikasi perangkat bergerak yang dikembangkan pada *platform* Android. Kegunaan aplikasi ini secara umum adalah sebagai modul *routing* yang membantu pengguna aplikasi untuk menuju tempat yang dituju dengan rute terdekat dan tercepat. Analisis kebutuhan mempunyai tujuan untuk mendapatkan semua kebutuhan yang diperlukan, sehingga semua kebutuhan yang harus dipenuhi pada aplikasi didasarkan pada kebutuhan dari pengguna aplikasi.

4.2.1 Identifikasi aktor

Aktor yang berinteraksi secara langsung pada perangkat lunak adalah pengguna aplikasi. Pengguna aplikasi disini mempunyai hak akses penuh pada seluruh fitur yang ada pada aplikasi.

4.2.2 Kebutuhan fungsional

Daftar kebutuhan berisi seluruh kebutuhan yang diperlukan oleh aplikasi yang akan dibangun. Kebutuhan fungsional yang terdiri dari fitur-fitur yang disediakan oleh aplikasi yang berguna untuk memenuhi kebutuhan pengguna.

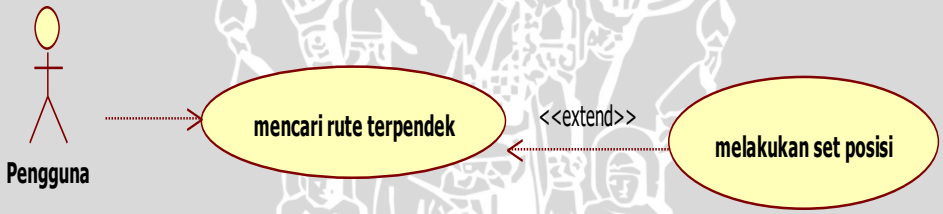
Kebutuhan fungsional ditunjukkan dengan menggunakan penomoran SRS(*Software Requirement Spesification*) serta digambarkan dengan menggunakan diagram *Use Case*. Daftar kebutuhan fungsional ditunjukkan pada Tabel 4.1

Tabel 4.1 Spesifikasi kebutuhan fungsional

Nomor SRS	Kebutuhan	Use Case
SRS_001	Aplikasi harus dapat menampilkan peta wilayah Kota Malang agar pengguna dapat melakukan set awal posisi secara manual pada setiap jalan yang ada di Kota Malang.	Melakukan set posisi.
SRS_002	Aplikasi harus dapat menampilkan rute jalan tercepat menuju tempat yang telah ditentukan dengan menggunakan Algoritma A*(*star*).	Mencari rute terpendek.

4.2.2.1 Diagram *Use Case*

Diagram *use case* merupakan diagram utama yang digunakan untuk menggambarkan perilaku sistem serta menggambarkan kebutuhan yang diperlukan oleh aktor dari suatu sistem atau aplikasi. Gambar 4.3 merupakan diagram *use case* dari daftar kebutuhan fungsional aplikasi.



Gambar 4.3 Diagram *use case*

4.2.2.2 Sekenario *use case*

Sekenario *use case* digunakan untuk menjelaskan secara detail tentang masing-masing kebutuhan fungsional yang terdapat pada diagram *use case*. Kebutuhan fungsional untuk melihat posisi pengguna pada peta *offline Open Street Map* serta pengguna dapat melakukan set posisi dengan memilih salah satu marker jalan untuk dikelola lebih lanjut dengan Algoritma A*(*star*) ditunjukkan oleh *use case* melakukan set posisi. Setiap titik jalan yang terdaftar pada basis data akan ditampilkan pada peta *offline open street map* dengan menggunakan marker. Pengguna melakukan set posisi secara manual dengan cara *click* nama jalan sesuai dengan yang diinginkan oleh pegguna. Posisi yang dipilih oleh pengguna akan menjadi lokasi atau titik awal untuk pencarian rute menuju lokasi yang dituju dengan menggunakan Algoritma A*(*star*). Sekenario *usecase* melakukan set posisi ditunjukkan pada Tabel 4.2 dibawah ini.

Tabel 4.2 Skenario *usecase* melakukan set posisi

Skenario *Use Case* Melakukan Set Posisi	
Kode SRS	SRS_001
Nama Use Case	*Use Case* Melakukan Set Posisi
Tujuan	Untuk melakukan set posisi pengguna.
Deskripsi	*Use Case* ini digunakan untuk melakukan set posisi pengguna secara manual dengan memilih marker pada jalan yang ada pada *Peta Open Street Map*.
Aktor	Pengguna
Pemicu	Pengguna memilih menu pencarian rute terdekat.
Kondisi Awal	Sistem menampilkan halaman utama aplikasi.
Skenario Utama (*Basic Flow*)	
Pengguna memilih menu pencarian rute terdekat. Sistem menampilkan peta *Open Street Map* beserta marker jalan. Sistem menampilkan pilihan set posisi.	
Skenario Bagian (*Sub Flow*)	
Set Posisi Manual	Pengguna memilih set posisi secara manual. Pengguna memilih salah satu marker jalan. Sistem menampilkan rute terpendek hasil perhitungan Algoritma A*(star).
Kondisi Akhir	Sistem menampilkan rute terpendek hasil perhitungan Algoritma A*(star).

Hasil perhitungan dengan menggukan Algoritma A*(*star*) dapat ditampilkan dalam bentuk rute pada peta *Open Street Map*. Kebutuhan fungsional untuk melihat rute hasil dari perhitungan dengan menggunakan Algoritma A*(*star*)

ditunjukkan oleh *use case* mencari rute terpendek. Sekenario *usecase* mencari rute terpendek ditunjukkan pada Tabel 4.3 di bawah ini.

Tabel 4.3 Skenario *usecase* mencari rute terpendek

Skenario *Use Case* Mencari Rute Terpendek	
Kode SRS	SRS_002
Nama *Use Case*	*Use Case* Mencari Rute Terpendek
Tujuan	Untuk melihat rute hasil perhitungan Algoritma A*(*star*).
Deskripsi	*Use Case* ini digunakan untuk melihat rute hasil perhitungan Algoritma A*(*star*) yang ditampilkan dalam bentuk rute pada peta offline *Open Street Map*.
Aktor	Pengguna
Pemicu	Pengguna meng *input* kan posisi.
Kondisi Awal	Sistem menampilkan halaman set posisi.
Skenario Utama (*Basic Flow*)	
1. Sistem menampilkan halaman set posisi. 2. Pengguna melakukan set posisi. 3. Sistem menampilkan rute terpendek hasil perhitungan Algoritma A*(*star*).	
Kondisi Akhir	Sistem menampilkan rute terpendek hasil perhitungan Algoritma A*(*star*).

4.3 Perancangan modul *routing*

Perancangan modul routing disini akan memuat mengenai perancangan Algoritma A*(*star*) dan perhitungan manual dari Algortima A*(*star*) yang digunakan untuk membangun aplikasi ini.

4.3.1 Perancangan Algoritma A*(*star*)

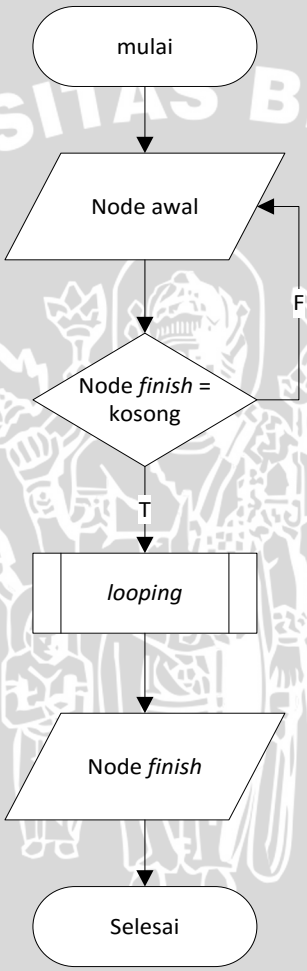
Fungsi Algoritma A*(*star*) pada aplikasi ini memiliki fungsi menentukan rute terbaik dan tercepat menuju point atau node yang dituju oleh pengguna. Proses Algoritma A*(*star*) dimulai ketika pengguna melakukan set posisi awal. Set posisi berupa id jalan dari sebuah node yang akan menjadi titik awal dari perhitungan Algoritma A*(*star*) yaitu untuk menentukan *cost* terpendek atau termurah dari set posisi yang sudah di tentukan oleh pengguna.

Algoritma A*(*star*) terdiri dari beberapa tahap diantaranya adalah :

1. Memasukkan node awal atau *start*.
2. Menentukan node *finish* atau tujuan.
3. Melakukan *looping* selama node *finish* belum ditemukan.
4. Menghasilkan node *finish* dengan jalur tercepat dan *cost* terpendek.

Jika node *finish* masih kosong maka proses akan terus berlangsung yaitu dengan melakukan *looping* sampai node *finish* terisi dan proses akan berhenti.

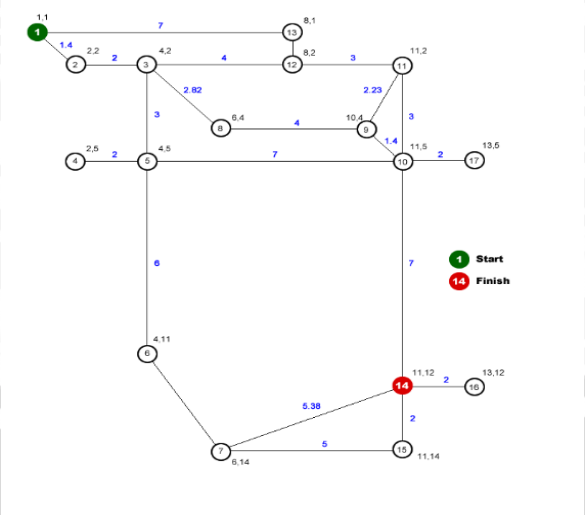
Langkah-langkah perhitungan menggunakan Algoritma A*(*star*) secara rinci ditunjukkan pada diagram alir Gambar 4.4 di bawah ini.



Gambar 4.4 Diagram alir Algoritma A*(*star*)

4.3.2 Perhitungan manual

Pada bagian ini akan dijelaskan mengenai perhitungan manual dari Algoritma A*(*star*) dengan menggunakan data sampel. Data sampel ini diperolah dari sebagian data yang nantinya akan dijakikan sebagai data pada aplikasi yang dibangun. Gambar 4.4 dibawah ini merupakan suatu daerah di Kota Malang yang sudah dijadikan path jalan, lingkaran warna hijau merupakan titik awal dan lingkaran warna merah merupakan tujuan.



Gambar 4.5 Path jalan

Di dalam algoritma A*(star) terdapat 2 *list*, yaitu *openlist* dan *exploredlist.* Setiap perhitungan *cost*, diperlukan jarak lurus antara node dengan node *finish* menggunakan rumus *distance*. Untuk mempermudah setiap tahap algoritma, maka berikut telah dihitung jarak antara setiap node dengan node *finish* yang ditunjukkan oleh Tabel 4.4 dan Tabel 4.5 dibawah ini.

Tabel 4.4 Tabel jarak tiap node dengan node *finish*

Node	Koordinat	Jarak dengan Node 14
1	1,1	14.86
2	2,2	13.45
3	4,2	12.2
4	2.5	11.4
5	4.5	9.89
6	4.11	7.07
7	6.14	5.38
8	6.4	9.43
9	10.4	8.06
10	11.5	7
11	11.2	10
12	8.2	10.44

Tabel 4.5 tabel jarak tiap node dengan node *finish* (lanjutan)

Node	Koordinat	Jarak dengan node 14
13	8.1	11.4
15	11.14	2
16	13.12	10.19
17	13.5	7.28

Proses perhitungan menggunakan Algoritma A*(*star*) secara manual dengan menggukan data pada Tabel 4.4 dan Tabel 4.5 telah dijabarkan pada lampiran A.

4.4 Perancangan perangkat lunak

Setelah melalui analisa kebutuhan, selanjutnya kebutuhan-kebutuhan tersebut dibentuk dalam suatu perancangan perangkat lunak yang terdiri dari diagram perancangan basis data dan perancangan antar muka.

4.4.1 Perancangan basis data

Basis data berfungsi untuk menyimpan informasi atau data-data yang diperlukan oeleh sebuah sistem. Perancangan basis data digambarkan dalam bentuk ERD (*Entity Relationship Diagram*). Sistem ini menggunakan dua tabel yaitu untuk menyimpan data *node*(titik) dan tabel untuk menyimpan data *way* (jalan). Gambar 4.6 dibawah ini menunjukkan perancangan basis data yang akan digunakan untuk menyimpan data-data yang diperlukan.

Gambar 4.6 Perancangan basis data

Struktur tabel dan keterangan dari masing-masing *field* ditunjukkan pada Tabel 4.6 dan Tabel 4.7 dibawah ini.

Tabel 4.6 Struktur tabel node

No	Nama field	Tipe data	Keterangan	Contoh data
1	Id	*Numeric*	Id jalan	270392346
2	Lat	*Text*	Koordinat *latitude* jalan.	-7.9385083
3	Lon	*Text*	Koordinat *longitude* jalan.	112.6497439

Data node disimpan dengan menggunakan titik terdekat setelah pengguna memilih posisi awal atau titik *start.*

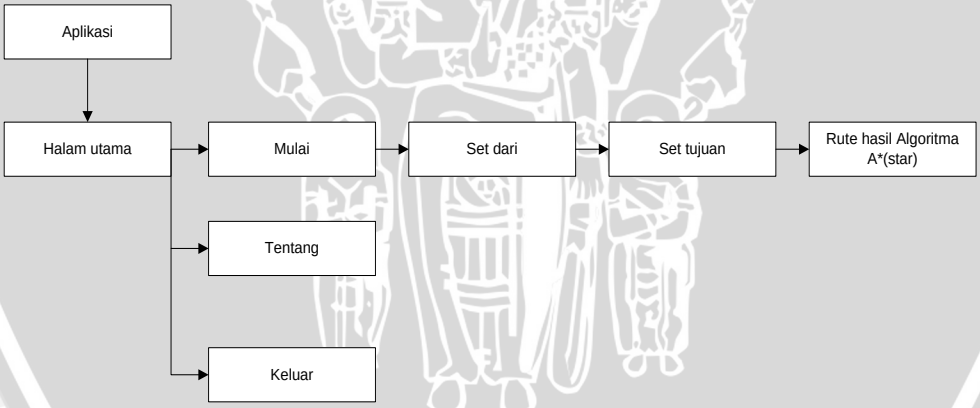
Tabel 4.7 Struktur tabel *way*

No	nama field	Tipe data	Keterangan	Contoh data
1	Id	*Numeric*	Id jalan	1
2	Node *from*	*Text*	Titik awal rute	1621495306
3	Node *to*	*Text*	Titik tujuan rute	1621495247

Tabel *way* digunakan untuk menyimpan rute yang akan digunakan oleh pengguna untuk dilewati dan yang akan digunakan untuk perhitungan dengan menggunakan Algoritma A*(*star*).

4.4.2 Perancangan antar muka

Perancangan antar muka berisi rancangan tampilan aplikasi yang dibangun. Perancangan berfungsi agar aplikasi yang dibuat mudah digunakan oleh pengguna. Untuk memudahkan menampilkan keseluruhan antar muka sistem yang dibangun maka digunakan *Site Map. Site Map* aplikasi ditunjukkan pada Gambar 4.7 dibawah ini.



Gambar 4.7 *Site map* aplikasi

4.4.2.1 Desain antar muka halaman utama aplikasi

Halaman utama merupakan halaman yang muncul pertama kali sebelum aplikasi dijalankan. Pada halaman utama terdapat tiga menu utama dalam aplikasi diantaranya adalah menu mulai, tentang dan tombol keluar yang digunakan untuk mengahiri aplikasi. Rancangan halaman utama aplikasi ditunjukkan pada Gambar 4.8.

Gambar 4.8 Desain antar muka halaman utama aplikasi

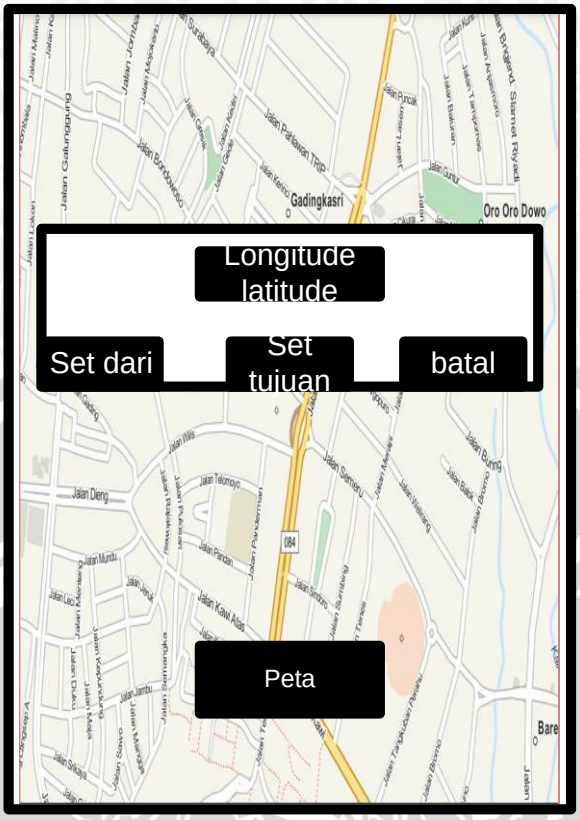
Untuk melakukan *routing* pengguna memilih menu yang pertama yaitu mulai seperti yang ditunjukkan Gambar 4.8**.** Keterangan Gambar 4.8 dijelaskan pada Tabel 4.8 dibawah ini**.**

Tabel 4.8 Keterangan desain antarmuka halaman utama aplikasi

Nama	Keterangan
Mulai	Menu ini merupakan menu untuk memulai aplikasi. Saat pengguna memilih menu ini makan pengguna akan langsung disuguhkan peta kota Malang secara keseluruhan. Setelah itu pengguna melakukan set posisi awal, set posisi awal ini dijadikan sebagai acuan untuk perhitungan dengan Algoritma A*(*star*) dan hasilkan akan ditampilkan oleh aplikasi.
Tentang	Menu ini digunakan untuk masuk ke halaman deskripsi tentang aplikasi.
Keluar	Menu ini berguna jika pengguna ingin keluar dari aplikasi.

4.4.2.2 Desain antarmuka halaman set lokasi

Halaman set lokasi merupakan halaman setelah menu pertama dipilih oleh pengguna, rancangan antarmuka halaman set lokasi ditunjukan Gambar 4.9.



Gambar 4.9 Desain antar muka halaman set lokasi

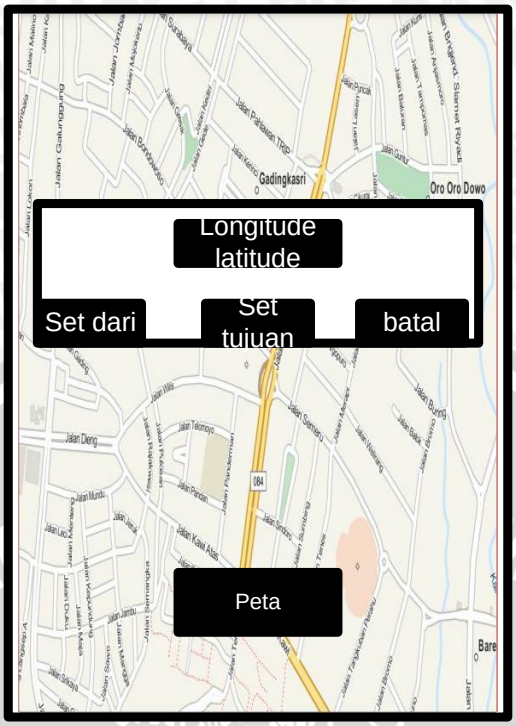
Pada halaman ini setelah pengguna menentukan titik awal lokasi maka akan muncul keterangan *longitude* dan *latitude* dari posisi yang dipilih oleh pengguna dan muncul tiga menu diantaranya set dari, set tujuan dan batal. *Input* yang dilakukan oleh pengguna akan secara otamatis mendeteksi titik terdekat dari titik yang di masukkan. Keterangan gambar ditunjukkan oleh Tabel 4.9 dibawah ini.

Tabel 4.9 Keterangan desain antarmuka halaman set lokasi

Nama	Keterangan
Longitude* dan *latitude	Menampilkan *longitude* dan *latitude* daerah dari titik yang sudah ditentukan oleh pengguna aplikasi.
Set dari	Merupakan menu untuk melakukan set posisi awal.
Set tujuan	Merupakan menu untuk melakukan set tujuan.
Batal	Menampilkan menu untuk membatalkan set posisi maupun set tujuan.
Peta	Menampilkan *peta offline open street map* Kota Malang.

4.4.2.3 Desain antarmuka set tujuan

Halaman set tujuan merupakan halaman setelah penguna melakukan set lokasi awal, rancangan halaman set tujuan ditunjukkan oleh Gambar 4.10 dibawah.



Gambar 4.10 Desain antarmuka halaman set tujuan

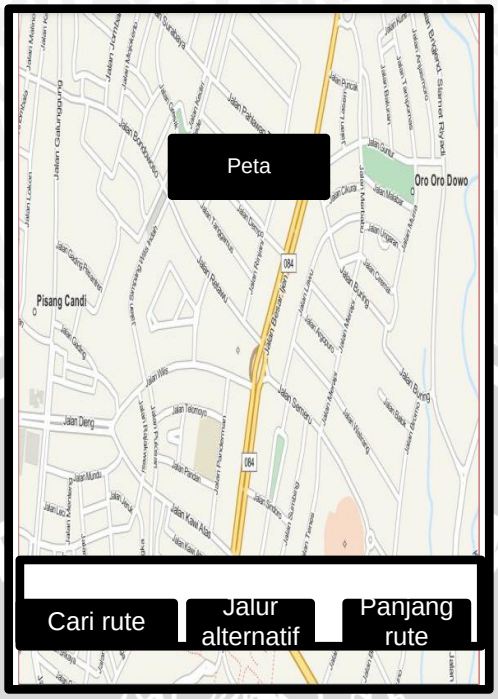
Pada halam ini setelah pengguna menentukan titik tujuan maka akan muncul keterangan *longitude* dan *latitude* dari posisi yang dipilih oleh pengguna dan muncul tiga menu diantaranya set dari, set tujuan dan batal. Input yang dilakukan oleh pengguna akan secara otamatis mendeteksi titik terdekat dari titik yang di *input*kan. Keterangan gambar ditunjukkan oleh Tabel 4.10 dibawah ini.

Tabel 4.10 Keterangan desain antarmuka halaman set tujuan

Nama	Keterangan
Longitude* dan *latiude	Menampilkan *longitude* dan *latitude* daerah dari titik yang sudah ditentukan oleh pengguna aplikasi.
Set dari	Merupakan menu untuk melakukan set posisi awal.
Set tujuan	Merupakan menu untuk melakukan set tujuan.
Batal	Menampilkan menu untuk membatalkan set posisi maupun set tujuan.
Peta	Menampilkan *peta offline open street map* Kota Malang.

4.4.2.4 Desain antarmuka halaman hasil

Halaman hasil mempunyai fungsi untuk menampilkan hasil dari perhitungan dengan menggunakan Algoritma A*(*star*) yang berupa rute terpendek menuju lokasi yang dituju. Halaman ini muncul setelah pengguna melakukan set tujuan pada halaman set tujuan. Rancangan halaman hasil ditunjukkan Gambar 4.11.



Gambar 4.11 Desain antarmuka halaman hasil

Rute hasil ditampilkan oleh aplikasi dalam bentuk jalanan yang harus dilalui untuk bisa mencapai tempat tujuan ditandai dengan memberi warna pada jalan mulai dari posisi awal hingga tempat yang dituju. Keterangan Gambar 4.11 diatas ditunjukkan pada Tabel 4.11 dibawah ini.

Tabel 4.11 Keterangan desain antarmuka halaman hasil

Nama	Keterangan
Peta	Nemapilkan rute hasil dari perhitungan Algoritma A*(*star*).
cari rute	Menu ini berfungsi untuk melakukan *routing*.
Jalur alternatif	Menu ini berfungsi untuk menampilkan beberpa rute lain sebagai pembanding.
Panjang rute	Menampilkan panjang rute yang telah ditentukan oleh pengguna.

BAB 5 IMPLEMENTASI DAN PENGUJIAN

Bab ini akan membahas hasil implementasi berdasarkan perancangan yang telah dibahas pada bab sebelumnya.

5.1 Lingkungan implementasi

Lingkungan implementasi yang dijabarkan pada sub bab ini meliputi lingkungan perangkat keras dan lingkungan perangkat lunak yang digunakan untuk pengembangan maupun untuk implementasi aplikasi pencarian rute terpendek dengan menggunakan metode Algoritma A*(*star*) di wilayah Kota Malang.

5.1.1 Lingkungan perangkat keras

Pada lingkungan perangkat keras ini yang digunakan untuk membangun aplikasi, instalasi aplikasi dan pengujian aplikasi terdiri dari dua macam diantaranya computer atau laptop yang digukan untuk pengembangan aplikasi sedangkan perangakat bergerak (*smartphone*) digunakan untuk instalasi dan pengujian aplikasi. Perangkat keras yang digunakan untuk pengembangan aplikasi ditujukan pada Tabel 5.1 dibawah ini. Sedangkan lingkungan perangkat keras dalam melakukan instalasi dan pengujian ditunjukkan pada Tabel 5.2**.**

Tabel 5.1 lingkungan perangkat keras pengembang aplikasi

Komponen	Spesifikasi
System model	Asus A450C
Processor	Intel Core i7 3537U. 2.0GHz - 4MB *Chache*
***Memory* (RAM)**	4 GB
Hardisk	750 GB
VGA	N vidia Geforce GT 720M

Tabel 5.2 lingkungan perangkat keras instalasi dan pengujian aplikasi

Komponen	Spesifikasi
System model	Advan X7
Processor	Intel Atom X3-C3230RK quad-core 1.0Ghz
Memory	1GB RAM 8 GB
Display	IPS LCD *capacitive touch screen*, 16M *colors* 1024 x 600 *pixels* (~170 ppi *pixels density*), 7 *inches*

5.1.2 Lingkungan perangkat lunak

Perangkat lunak yang digunakan untuk pengembangan aplikasi ditunjukkan pada Tabel 5.3**.** Sedangkan lingkungan perangkat lunak dalam melakukan istalasi dan pengujian ditunjukkan pada Tabel 5.4

Tabel 5.3 lingkungan perangkat lunak pengembangan aplikasi

Komponen	Spesifikasi
Operating system	Windows 8.1 pro 64-bit
IDE (*Integrated Development Enviroment*)	Eclipse Luna dengan ADT (*Android Development Tools*) *plugin*.
Perangkat lunak pendukung	SQL Database Browser

Tabel 5.4 lingkungan perangkat lunak instalaso dan pengujian aplikasi

Koponen	Spesifikasi
Platform	Android OS v5.1 Lollipop

5.2 Batasan implementasi

Dalam implementasi aplikasi ini terdapat beberapa batasan implementasi, batasan tersebut diantaranya :

1. Implementasi penggunaan peta pada aplikasi dan untuk melakukan set posisi hingga menampilkan hasil *routing* menggunakan peta *offline open street map* wilayah Kota Malang saja.
2. Set posisi secara manual dan dilakukan dengan mendeteksi titik atau node yang terdekat dari posisi yang dipilih.
3. Aplikasi hanya menampilkan rute yang terbaik dan tercepat sesuai perhitungan Algoritma A*(*star*).
4. Penggambaran rute pada peta *offline open street map* dilakukan dengan menarik garis antara satu titik ke titik lainnya sesuai dengan hasil perhitungan Algoritma A*(*star*) dan bukan merupakan penarikan garis dari titik awal menuju titik tujuan.

5.3 Implementasi basis data

Aplikasi ini dibangun dengan menggunakan basis data untuk penyimpanan seluruh data yang diperlukan. Data-data tersebut terdiri dari data node dan data *way.* Implementasi basis data sesuai dengan perancangan basis data yang

terdapat pada bab perancangan. Tabel 5.5 dibawah ini merupakan implementasi dari Tabel 4.6, Tabel 5.6 merupakan implementasi dari Tabel 4.7.

Tabel 5.5 Tabel data node

No.	Kolom	Tipe data	Deskripsi	Contoh data
1	Id	*Numeric*	Id jalan.	270392346
2	Lat	*Text*	Koordinat *latitude* jalan.	-7.9385083
3	Lon	*Text*	Koordinat *longitude* jalan.	112.6497439

Pada Tabel 5.6 diatas menjelaskan tentang kolom-kolom dari tabel yang diimplementasikan. Contoh data merupakan salah satu bagian isi dari pada kolom tersebut.

Tabel 5.6 Tabel data *way*

No.	Kolom	Tipe data	Deskripsi	Contoh data
1	Id	*Numeric*	Id jalan	1
2	Node *from*	*Text*	Titik awal rute.	1621495306
3	Node *to*	*Text*	Titik tujuan rute.	1621495247

5.4 Implementasi kode program

Pada implementasi kode program akan dijelaskan dan ditampilkan mengenai hasil implementasi aplikasi sesuai dari perancangan yang telah dijelaskan pada bab sebelumya. Aplikasi pengembangan modul *routing* dengan menggunakan Algoritma A*(*star*) ini terdiri dari beberapa *method* atau fungsi, pada penulisan penelitian ini akan dijelaskan beberapa *method* utama yang digunakan untuk membangun aplikasi sehingga tidak seluruh *method* akan dijelaskan secara rinci pada bagian ini. *Method-method* tersebut diantara lain beberapa *method* pada implementasi Algoritma A*(*star*) dan beberapa *method* yang berfungsi sebagai pendukung pada pembuatan aplikasi ini.

5.4.1 Kode program database

Bagian ini menjelaskan potongan kode program dari implementasi untuk *creator* SQL database. Kode 1.1 dibawah ini merupakan potongan program database yang digunakan untuk menampilkan dan membuat database yang diperlukan sekaligus sebagai media penyimpanan seluruh data jalan yang digunakan pada aplikasi yang dibangun. Pada pembangunan aplikasi ini penyimpanan data menggunakan SQLite.

```
1  public class SQLiteDatabaseCreator {
2      public static void main(String[] args) {
3          Connection c;
4          Statement  stmt;
5          try {
6
7  //Membuka database malang.db, jika tidak ada maka akan otomatis terbuat
8              Class.forName("org.sqlite.JDBC");
9              c= DriverManager.getConnection("jdbc:sqlite:malang.db");
10             c.setAutoCommit(false);
11
12             //Membuat Tabel
13             stmt = c.createStatement();
14             String sql = "CREATE TABLE node " +
15 "(idINTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,"+
16                         " lat  REAL NOT NULL, " +
17                         " lon  REAL NOT NULL)";
18             String sql2 = "CREATE TABLE way " +
19                    "(idINTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,"+
20                         " node_from  REAL NOT NULL, " +
21                         " node_to    REAL NOT NULL)";
22             stmt.executeUpdate(sql);
23             stmt.executeUpdate(sql2);
24
25             c.commit();
26             stmt.close();
27             c.close();
28
29             System.out.println("Sukses");
30         } catch (ClassNotFoundException ex) {
31             System.out.println("Class Notfound Ex: " + ex.getMessage());
32         } catch (SQLException ex) {
33             System.out.println("SQL Ex: " + ex.getMessage());
34         }
35
36     }
37 }
```

Kode 1.1 Implementasi *creator* SQL database

5.4.2 Kode program implementasi Algoritma A*(*star*)

Bagian ini akan menjelaskan potongan kode program dari implementasi Algoritma A*(*star*). Algoritma A*(*star*) pada aplikasi ini terdiri dari *method* dan *class* diantaranya *getDistance*() dan *buildWaypoint*(). *Methot* dan *class* tersebut disajikan dalam bentuk potongan kode program yaitu pada Kode 1.2 dan Kode 1.3

```
1  publicstaticfloat getDistance(Node n1, Node n2) {
2  doubledLat=Math.toRadians(n2.location.getLatitude()n1.location.getLatitude());
3  doubledLng=Math.toRadians(n2.location.getLongitude()-
4  n1.location.getLongitude());double a = Math.sin(dLat/2) * Math.sin(dLat/2) +
5  Math.cos(Math.toRadians(n1.location.getLatitude()))*
6  Math.cos(Math.toRadians(n2.location.getLatitude()))*Math.sin(dLng/2)          *
7  Math.sin(dLng/2)double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
8  float dist = (float) (6371000 * c);
9        return dist;
10       }
```

Kode 1.2 Implementasi *method getDistance*()

Kode 1.3 dibawah ini adalah *class* yang digunakan untuk menggabungkan semua poin-poin rute jalan sehingga rute jalan berubah menjadi point yang sudah terhubung dan dapat dihitung dengan menggunakan Algoritma A*(*star*).

```
1  public class GraphSearch {
2
3          public ArrayList<Node> waypoint;
4          public boolean found = false;
5          public  void  graphDFSByRecursion(Node  currentNode,Node
6  targetNode) {if (null == currentNode) {return; // back track}
7                  visitNode(currentNode);
8                  waypoint.add(currentNode);
9                  currentNode.visited = true;
10                 //check if we reached out target node
11                 if (currentNode.id == targetNode.id)
12  {found = true;return;
13  // we have found our target nodeV.
14                 }
15                 //recursively visit all of unvisited neighbors
16               for (Node neighbor : currentNode.neighbors) {
17                 if (!found)
18                 if (!neighbor.visited) {
19                 graphDFSByRecursion(neighbor, targetNode);
20 }
21               }
22         }
```

Kode 1.3 Implementasi *class waypoint()*

5.5 Implementasi antarmuka

Bagian ini menampilkan hasil implementasi antarmuka aplikasi pengembangan modul *routing offline* secara keseluruhan.

5.5.1 Implementasi antarmuka halaman utama

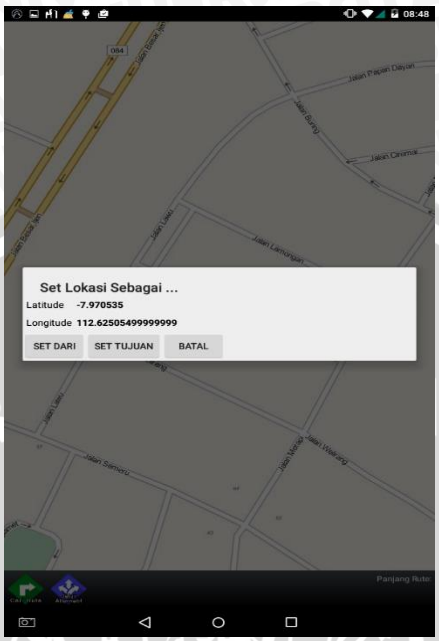
Halaman utama aplikasi merupakan halaman pertama yang akan muncul ketika aplikasi mulai dijalankan pertama kali. Halaman ini terdiri dari tiga menu utama yaitu menu mulai, menu tentang, dan menu untuk mengahiri aplikasi atau keluar. Halaman utama ini juga sekaligus sebagai tanda jika aplikasi siap untuk dijalankan. Implementasi halaman utama aplikasi ditunjukkan pada Gambar 5.4

Gambar 5.1 implementasi antarmuka halaman utama

5.5.2 Implementasi antarmuka halaman set lokasi

Halaman set lokasi merupakan halaman dimana pengguna aplikasi melakukan set posisi awal atau lokasi awal pada peta yang tertera. Halaman ini akan muncul

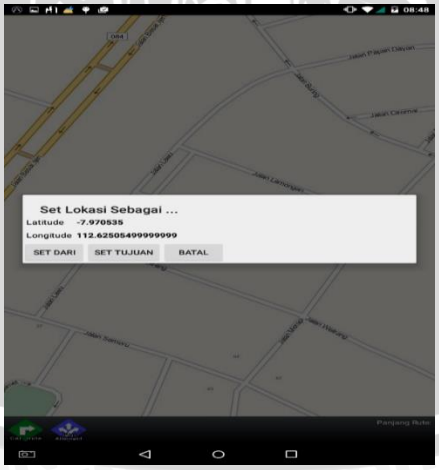
saat pengguna menekan atau memilih tombol mulai pada halaman utama aplikasi, pemiliham lokasi ini dilakukan secara manual dan hanya terbatas di wilayah kota Malang saja. Implementasi halaman set lokasi akan ditunjukkan pada Gambar 5.5.



Gambar 5.2 Implementasi antarmuka set lokasi

5.5.3 Implementasi antarmuka halaman set tujuan

Halaman set tujuan merupakan halaman ketiga setelah pengguna menentukan posisi awal, halaman ini tidak jauh berbeda dengan halaman set lokasi namun peruntukan di halaman ini hanya untuk melakukan tujuan dari posisi semula. Implementasi halaman set tujuan ini ditunjukkan pada Gambar 5.6.

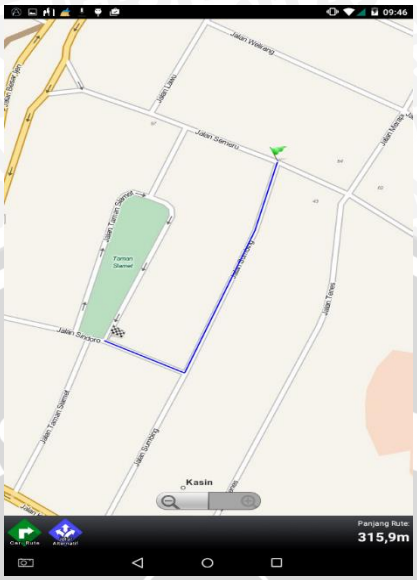


Gambar 5.3 Implementasi antarmuka halaman set tujuan

5.5.4 Implementasi antarmuka halaman hasil

Halaman terahir dari proses menentukan lokasi awal dan tujuan adalah halaman hasil, halaman ini berisi hasil kalkulasi rute dan akan muncul marker jalan

sebagai penunjuk arah sesuai dengan yang telah ditentukan oleh pengguna aplikasi. Implementasi halaman hasil ditunjukkan pada Gambar 5.7.



Gambar 5.4 Implementasi antarmuka halaman hasil

5.6 Pengujian

Pada bagian ini akan menjelaskan tentang pengujian dan analisis yang akan dilakukan pada aplikasi yang telah dibangun. Pengujian dilakukan dengan menggunakan metode *Black Box Testing*, pengujian yang dilakukan diantaranya adalah pengujian validasi dan pengujian terhadap Algoritma A*(*star*) itu sendiri. Setelah melalui tahap pengujian maka akan dilakukan tahap analisis terhadap hasil dari pengujian yang telah dilakukan, analisis dilakukan bertujuan untuk memperoleh kesimpulan dari hasil setiap pengujian yang telah dilakukan.

5.6.1 Pengujian validasi fitur

Pengujian validasi dilakukan untuk memastikan bahwa aplikasi yang dibangun telah memenuhi semua kebutuhan fungsional sesuai pada bab perancangan. Pengujian validasi merupakan pengujian yang tidak memperhatikan alur algoritma program atau sering disebut dengan *black box testing,* namun lebih ditekankan pada kesesuaian antara sistem yang telah dibangun dengan daftar kebutuhan fungsional.

5.6.1.1 Kasus uji validasi fitur

Untuk melakukan pengujian validasi sebelumnya ditentukan kasus uji yang sesuai dengan daftar kebutuhan pada bab perancangan. Berikut merupakan kasus uji berdasarkan daftar kebutuhan fungsional. Kasus uji melakukan set posisi ditunjukkan pada Tabel 5.7 dan kasus uji mencari rute tercepat ditunjukkan pada Tabel 5.8**.**

Tabel 5.7 Kasus uji melakukan set posisi

Nomor kasus uji	**UVL_01**
Nama kasus uji	Kasus uji melakukan set posisi
Objek uji	Kebutuhan fungsional SRS_001
Tujuan pengujian	Pengujian dilakukan bertujuan untuk memastikan bahwa set posisi dapat dilakukan secara manual secara baik dengan memilih marker jalan.
Data	Manual : memilih marker dengan id=4
Prosedur uji	1. Memilih menu mulai pada halaman awal aplikasi. 2. Masuk pada halaman kedua aplikasi. 3. Melakukan set posisi sebagai awalan atau titik start pada marker jalan. 4. Menekan tombol cari rute. 5. Sistem akan melakukan perhitungan dengan id jalan yang sesuai dengan posisi yang telah ditentukan.
Hasil yang diharapkan	Pengguna dapat melakukan set posisi.

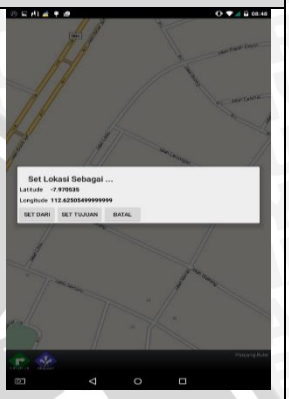
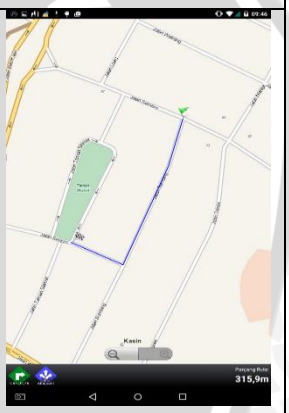
Tabel 5.8 Kasus uji mencari rute terpendek

Nomor kasus uji	**UVL_02**
Nama kasus uji	Kasus uji mencari rute terpendek
Objek uji	Kebutuhan fungsional SRS_002
Tujuan pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat menampilkan rute hasil perhitungan dengan Algoritma A*(*star*) pada peta *offline open street map*
Data	Manual : memilih marker dengan id = 4
Prosedur pengujian	1. Pengguna aplikasi melakukan set posisi. 2. Sistem menampilkan rute hasil perhitungan dengan menggunkan Algoritma A*(*star*) pada peta *offline open street map.*
Hasil yang diharapkan	Sistem dapat menampilkan rute hasil perhitungan dengan menggunakan Algoritma A*(*star*) pada peta *offline open street map.*

5.6.1.2 Hasil pengujian validasi fitur

Pada Tabel 5.9 dibawah ini menunjukkan hasil dari pengujian validasi fitur berdasar kasus uji yang telah dibuat sebelumnya.

Tabel 5.9 Hasil pengujian validasi fitur

No	Nomor kasus uji	Hasil yang diharapkan	Hasil yang didapat	Status validasi	*Screenshot*
1	UVL_01	Pengguna dapat melakukan set posisi pada marker jalan yang tertera di peta.	Pengguna sudah dapat melakukan set posisi dimanapun pada marker jalan yang ada di peta.	**VALID**	
2	UVL_02	Sistem dapat menampilkan rute hasil perhitungan dengan Algoritma A*(*star*) pada peta *offline open street map.*	Sistem sudah dapat menampilkan rute hasil perhitungan dengan Algoritma A*(star) pada peta *offline open street map.*	**VALID**	

5.6.1.3 Analisis hasil pengujian validasi fitur

Analisis hasil pengujian validasi dilakukan dengan membandingkan kesesuaian hasil kerja sistem yang telah dibangun dengan semua daftar kebutuhan yang telah didefinisikan pada bab perancangan. Berdasarkan hasil pengujian yang telah dilakukan dapat diketahui bahwa hasil implementasi sistem telah sesuai dengan semua fitur pada daftar kebutuhan yang telah didefinisikan pada bab perancangan dan hasil perhitungan yang dilakukan oleh sistem dengan menggunakan perhitungan Algoritma A*(*star*) juga sudah sesuai dengan perhitungan manual yang telah dibuat.

5.6.2 Pengujian Algoritma A*(*star*)

Pengujian Algoritma A*(*star*) dilakukan untuk pengujian validasi hasil apakah hasil perhitungan oleh aplikasi sama dengan hasil perhitungan manual dalam memecahkan permasalahan pencarian rute tercepat. Prosedur pengujian dilakukan dengan melakukan perhitungan manual yang terlampir pada bab

lampiran setelah itu hasil perhitungan dipadupadankan dengan hasil yang dihasilkan oleh aplikasi yang dibangun.

5.6.2.1 Sistematika pengujian Algoritma A*(*star*)

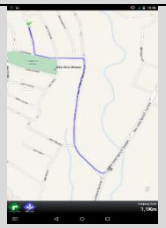
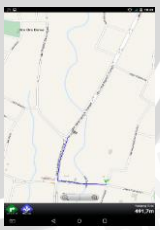
Pengujian dilakukan dengan melakukan pengacakan pada posisi awal atau titik *start* pada marker jalanan wilayah Kota Malang, setelah ditentukannya titik awal akan dilanjutkan dengan perhitungan secara manual yang sudah terlampir pada bab lampiran.

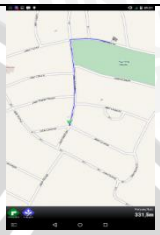
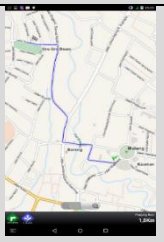
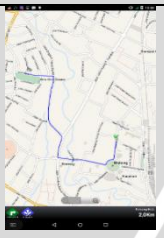
Pada aplikasi akan dilakukan cara yang sama yaitu dengan memilih marker jalan yang sama dengan perhitungan manual yang dilakukan. Setelah didapat hasil maka hasil tersebut akan dipadupadankan kesesuainnya.

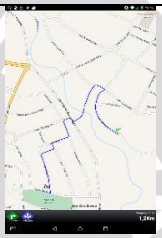
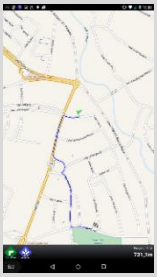
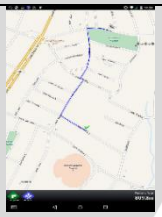
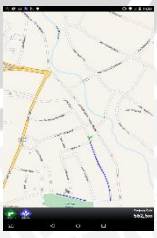
5.6.2.2 Hasil dan analisis pengujian validasi hasil

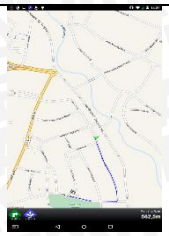
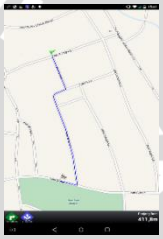
Pengujian validasi hasil dilakukan dengan menggunakan 17 titik awal yang sama antara perhitungan manual dan perhitungan yang dilakukan oleh aplikasi. Titik awal yang digunakan diantaranya node 1, node 12, node 7, node node 4, node 5, node 3, node 2, node 11, node 15 node 10, node 8, node 13, node 17, node 6, node 16 dan node 9. Hasil pengujian baik secara manual maupun pada aplikasi ditunjukkan pada Tabel 5.10 dan pada lampiran B pada halaman 59.

Tabel 5.10 Hasil pengujian validasi hasil

No	Perhitungan Manual		Aplikasi			Validasi
	Titik *start*	Rute hasil	Titik *start*	*Rute hasil*	*screenshot*	
1	**1**	**1**-2-3-5-5-7-**14**	**1**	**1**-2-3-5-6-7-**14(1,1 KM)**		**VALID**
2	**12**	**12**-11-10-**14**	**12**	**12**-11-10-**14(491,7 M)** **12**-11-9-10-**14(498,2 M)** **12**-11-10-17-**14(514,6 M)** **12**-11-10-5-10-**14(629,0 M)** **12**-11-10-9-**14(511,9 M)**		**VALID**

3	**7**	**7**-6-5-10-**14**	**7**	**7**-15-**14(371,0 M)** **7**-6-5-10-17-10-**14(543,0 M)** **7**-6-5-10-**14(337,6 M)** **7**-6-5-3-8-9-10-**14(575,4 M)** **7**-6-5-3-12-11-9-10-**14(805,9 M)**		**TIDAK VALID**
4	**4**	**4**-5-10-**14**	**4**	**4**-5-10-**14(331,5 M)** **4-5-3-12-11-9-10-14(720,5 M)**		**VALID**
5	**5**	**5**-10-**14**	**5**	**5**-10-**14(978,5 M)** **5**-10-17-10-**14(1,2 KM)** **5**-3-12-11-9-10-**14(1,4 KM)** **5**-3-8-9-10-**14(1,7 KM)** **5**-3-8-9-10-**14(1,0 KM)**		**VALID**
6	**3**	**3**-8-9-10-**14**	**3**	**3**-8-9-10-**14(1,8 KM)**		**VALID**
7	**2**	**2**-3-8-9-10-**14**	**2**	**2**-3-8-10-**14(2,0 KM)**		**VALID**
8	**11**	**11**-10-**14**	**11**	**11**-10-**14(536** M) **11**-12-3-8-9-10-**14(808,5 M)** **11**-12-3-5-6-7-**14(867,1 M)** **11**-12-3-2-3-5-6-7-**14(873,7 M)** **11**-9-8-3-5-4-6-7-**14(776,9 M)**		**VALID**

9	**15**	**15-14**	**15**	**15-14 (275,8 M)** **15**-7-6-5-4-5-10-**14(1,1 KM)** **15**-7-6-5-10-**14(920,6 M)** **15**-6-5-3-8-9-10-**14(715,2 M)** **15**-7-6-5-3-8-9-10-**14(826,3 M)**		**VALID**
10	**10**	**10**-17-**14**	10	**10**-17-**14(453,1 M)** **10**-5-6-7-15-**14(810,9 M)** **10**-9-11-12-3-5-6-7-15-**14(1,1 KM)**		**VALID**
11	**8**	**8**-9-10-**14**	**8**	**8**-9-10-**14(1,2 KM)** **8**-9-11-10-**14(1,3 KM)** **8**-3-12-11-10-**14(1,7 KM)**		**VALID**
12	**13**	**13**-12-11-9-10-**14**	**13**	**13**-12-11-10-**14(731,1 M)** **13**-12-11-9-10-**14(697,6 M)** **13**-1-2-3-12-11-10-**14(873,7 M)** **13**-12-3-5-6-7-**14(1,1 KM)** **13**-12-11-10-9-8-3-5-6-7-**14(1,4 KM)**		**TIDAK VALID**
13	**3**	**3-8-9-10-14**	**3**	**3**-8-9-10-**14(809,8 M)** **3**-12-11-9-10-**14(1,1 KM)** **3**-2-1-13-12-11-10-**14(1,5 KM)**		**VALID**
14	**17**	**17-10-14**	**17**	**17**-10-**14(562,5 M)** **17**-10-5-6-7-**14(1,0 KM)** **17**-10-9-8-3-5-6-7-**14(1,1 KM)** **17**-10-11-12-3-5-6-7-**14(1,2 KM)**		**VALID**

15	**6**	**6-5-14**	**6**	**6**-5-**24(320 M)** **6**-5-10-**14(677,7 M)** **6**-3-12-11-10-**14(962,6 M)**		**VALID**
16	**16**	**16-14**	**16**	**16-14(151,8 M)** **16**-14-10-**14(591,2 M)** **16**-14-10-11-9-10-**14(655,8 M)** **16**-14-15-**14(509,5 M)** **16**-14-7-**14(650,4 M)**		**VALID**
17	**9**	**9-10-17-14**	**9**	**9**-10-17-**14(411,8 M)** **9**-10-5-6-7-**14(642,3 M)** **9**-8-3-5-6-7-**14(908,4 M)** **9**-10-5-6-7-**14(579,0 M)** **9**-11-12-3-5-6-7-**14(1,2 KM)**		**VALID**

Dari hasil yang telah ditujukkan pada Tabel 5.10 diatas dapat disimpulkan bahwa keluaran yang dihasilkan oleh aplikasi dan perhitungan manual secara keseluruhan telah sesuai dan dari hasil kalkulasi aplikasi menunjukkan bahwa hampir seluruh *routing* akurat baik dalam hal pemilihan jarak maupun arah tujuan dan tingkat keakuratan *routing* menggunakan Algoritma A (*star*) ini dapat dikatakan hampir sempurna. Namun pada pengujian nomor 3 dan 13 percobaan yang dilakukan dinyatakan tidak valid dikarenakan hasil yang dihasilkan oleh aplikasi dan perhitungan manual terdapat ketidak cocokan. Secara keseluruhan didapat hasil 11.7% dari total percobaan yang dinyatakan valid maka dapat disimpulkan penggunaan Algoritma A*(*star*) untuk proses routing mencapai kesempurnaan 88%.

BAB 6 PENUTUP

6.1 Kesimpulan

Berdasar hasil perancangan, implementasi dan pengujian yang telah dilakukan maka dapat diambil kesimpulan sebagai berikut:

1. Penerapan Algoritma A*(*star*) pada peta *offline* khususnya modul *routing* dirasa dapat dapat membantu pengguna perangkat bergerak untuk menemukan tempat yang dituju dengan pilihan jalur yang terpendek sehingga akan mempersingkat waktu perjalanan dan menghemat penggunaan daya batrei serta *bandwith* dari perangkat bergerak yang dimiliki serta tingkat akurasinya yang dirasa cukup baik.
2. Rancangan dan implementasi modul *routing* dengan menggunakan Algoritma A*(*star*) secara umum mengarah kepada asas-asas *user friendly* sehingga pengguna aplikasi tidak merasa bingung dalam penggunaanya. Selain itu aplikasi ini dibangun dengan menggunakan *SQLite* sebagai media atau *platform* penyimpanan seluruh data yang diperlukan oleh aplikasi.
3. Tingkat akurasi yang dihasilkan oleh Algoritma A*(*star*) pada implementasi untuk menentukan jalur terpendek mencapai sempurna secara keseluruhan, tingkat akurasi ini mencapai 88%. Tingkat akurasi ini jika dibandingkan dengan menggunakan Algoritma lain sangat berbeda dikarenakan cara kerja Algoritma A*(*star*) ini adalah mencoba seluruh kemungkian jalur yang ada setelah itu akan ditentukan *cost* yang terkecil lalu dijadikan sebagai jalur tercepat.

6.2 Saran

Saran yang dapat diberikan untuk pengembangan aplikasi ini adalah sebagai berikut :

1. Perlunya dilakukan pengembangan lebih lanjut pada aplikasi khususnya untuk mengatasi *loading* data yang lama saat proses *routing* saat titik awal dan tujuan yang ditentukan memiliki jarak yang jauh.
2. Perlunya pengembangan aplikasi dengan menggunakan format lain selain menggunakan pemrograman *eclipse*.
3. Dari segi aplikasi perlu adanya pengembangan lebih lanjut dengan menambahkan beberapa faktor lain yang mendukung baik dari segi penerapan peta, algoritma dan arah jalan yang telah di perbaharui.

DAFTAR PUSTAKA

Alfan, Mochammad.,2013. *Analisis Performasi DBMS SQLite sebagai Mobile Database*.[e-book].Universitas Telkom Bandung.S1.Tersedia di <https://openlibrary.telkomuniversity.ac.id/pustaka/95480/analisis-performansi-dbms-sqlite-sebagai-mobile-database.html> [Diakses 10 Desember 2015]

Achmad, Marizki, C., 2013. *Perancangan aplikasi dalam implementasi shortest path finder di kawasan kampus IT Telkom berbasis android dengan Algoritma A**.S1.[e-book]. Fakultas Elektro dan Komunikasi, Universitas Telkom Bandung.Tersedia di http://openlibrary.telkomuniversity.ac.id/pustaka/93152[Diakses 3 Mei 2015]

Budi.,2009.Pengertian navigasi, [online]. Tersedia di <http://jawaratracker.com/pengertian-navigasi-dan-contohnya/> [Diakses 3 Mei 2015].

Setiawan,Didik., 2015. *Perancangan dan implementasi Algoritma A* pada aplikasi angkot-finder di Kota Bandung untuk smartphone berbasis android*. S1.[e-book]. Fakultas Teknik, Universitas Telkom Bandung.Tersedia di <https://openlibrary.telkomuniversity.ac.id/pustaka/102733> [Diakses 3 Mei 2015]

Hart, P. E.; Nilsson, N. J.; Raphael, B. 1968. *"A Formal Basis for the Heuristic Determination of Minimum Cost Paths"*. IEEE Transactions on Systems Science and Cybernetics SSC4 4 (2): 100–107. doi:10.1109/TSSC.1968.300136

HOT., 2012. *Mengumpulkan data spasial dengan OpenStreetMaps*. [online]. Tersedia di <http://openstreetmap.id/docs/Unit1_Mengumpulkan_data_spasial_dengan_OSM_Tingkat_Dasar.pdf> [Diakses 25 Mei 2015]

Kergall,John., 2014. “How to use the OSMDroid Library” [online]. Tersedia di <https://github.com/osmdroid/osmdroid/wiki/How-to-use-the-osmdroid-library> [Diakses 1 Oktober 2015].

Mobac., 2008. “*Mobile Atlas creator*”.[online]. Tersedia di <www.mobac.sourceforge.net/> [Diakses 3 Mei 2015]

Novitasari,Vika., 2014.*, Rancang Bangun Aplikasi Pencarian Rute Terpendek untuk Menemukan SPBU Terdekat di Kota Malang dengan Menggunakan Algoritma Genetik Berbasis Android.*,S1.Fakultas Ilmu Komputer, Universitas Brawijaya, Malang.

Open Street Map Indonesia., 2014. *Pengenalan OpenStreetMap*. [online] Tersedia di <www.slideshare.net/hotosmid/03-pengenalan-openstreetmap> [Diakses 5 Mei 2015]

Open Street Map., 2012. *OpenStreetMap*. [image online]. Tersedia di <http://www.openstreetmap.org/#map=18/-8.09917/112.17807> [Diakses 25 Mei 2015]

Pearl, Judea., 1984. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley. ISBN 0-201-05594-5

S. Rabin., 2000 “*A* Speed Optimizations.” In Game Programming Gems, edited by Mark Deloura. Hingham,* M A : Charles River Media.

Wiliam., 2003. *Location Based Service,* [online]. Tersedia di <http://supeeerblog.blogspot.com/2013/05/location-based-services-lbs.html> [Diakses 25 Desember 2015]

LAMPIRAN A PERHITUNGAN MANUAL CONTOH KASUS PADA BAB 4

1. Masukkan node awal ke *openlist*.
   Node awal adalah node *start*, yaitu node 1.

   gscore = cost dari node pertama = 0.

   fscore = cost dari node dengan node *finish* = 0.

2. Ulangi langkah langkah berikut selama node *finish* belum ditemukan atau *openlist* tidak kosong:
   Loop 1.

   a. Jadikan node saat ini adalah node pertama dari *openlist*. Untuk looppertama ini, maka node saat ini adalah node 1.
   b. Tambahkan node saat ini ke *exploredlist*.
   c. Hapus node saat ini dari *openlist*.
   d. Periksa apakah node saat ini = node *finish*.
      Jika ya maka node *finish* telah ditemukan, loop berhenti, jika tidak maka lanjutkan proses.
   e. Pada setiap node tetangga, lakukan proses berikut:
      - Node 2 :
        Hitung *cost* : Jarak antara node 2 dan node 1 = 1.4

        Hitung g*score* sementara: g*score* node saat ini + *cost* = 0 + 1.4 = 1.4

        Hitung f*score* sementara: gscore sementara + jarak node 2 dengan *finish* = 1.4 + 13.45 = 14.85

        Jika di *exploredlist* terdapat node 2 **dan f*score* node** 2**< f*score* sementara maka *skip***. Node 2 **f*score*** = 0, belum ada di dalam *exploredlist*, maka lanjutkan (*no skip*).

        Jika dalam node 2 tidak terdapat dalam *openlist* atau f*score* node 2 > f*score* maka:

        Jadikan *parent* node 2 = node sekarang = node 1

        g *score* node 2 = g *score* sementara = 1.4

        f*score* node 2 = f*score* sementara = 1.485

        hapus node 2 dari *openlist* jika sudah ada didalamnya

tambahkan node 2 ke dalam *openlist*

- **Node 13 :**
  Cost = 7

  g*score* sementara = 0 + 7 = 7

  f*score* sementara = 7 + 11.4 = 18.4

  Node 13 tidak terdapat dalam *exploredlist*, maka lanjutkan

  Node 13 tidak terdapat dalam *openlist* maka *parent* node

  13=node 1, g*score*=7, f*score*=18.4

  Hapus Node 13dari *openlist* jika ada

  Tambahkan Node 13ke *openlist*

- **Saat ini *openlist* berisi Node 2 dan Node 13**.
- **Urutkan *openlist* berdasarkan f*score* terkecil**, maka Node 2 berada paling atas, node 2 akan menjadi node sekarang di loop selanjutnya, dan demikian seterusnya.

Loop 2

a. Node sekarang: Node2.
b. Tambahkan Node2 ke dalam *exploredlist*.
c. Hapus node 2 dari *openlist*.
d. Node2 =Node14 ? *False*, Lanjut.
e. Setiap node tetangga Node2:
   - **Node 1**
     - *Cost*= 1.41
     - g*score* sementara= 1.41 + 1.41=2.82
     - f*score* sementara= 2.82 + 14.87=17.69
     - Node1 ada dalam *exploredlist* dan nilai f*score*nya < f*score* sementara, maka Lanjut
   - **Node 3**
     - *Cost*= 2.0
     - g*score* sementara= 1.41 + 2.0=3.41
     - f*score* sementara= 3.41 + 12.21=15.62
     - Node3 belum ada dalam *openlist*
     - **MAKA:**
       Nilai Node3: *parent*=Node2, g*score*=3.41, f*score*=15.62
       Tambahkan Node3 ke dalam *openlist*.

Saat ini openlist berisi: Node3, Node13, urutkan berdasarkan f*score* terkecil, maka node teratas: Node3.

Loop 3

a. Node sekarang : node 3

b. Tambahkan Node3 ke dalam *exploredlist.*

c. Hapus node Node3 dari *openlist*.

d. Node3 = Node14 ? *False*, Lanjut.

e. Setiap node tetangga Node3:

- **Node 2**
  - *Cost*= 2.0
  - g*score* sementara= 3.41 + 2.0=5.41
  - f*score* sementara= 5.41 + 13.45=18.86
  - Node2 ada dalam *exploredlist* dan nilai f*score*< f*score* sementara, maka Lanjut
- **Node 5**
  - *Cost*= 3.0
  - g*score* sementara= 3.41 + 3.0=6.41
  - f*score* sementara= 6.41 + 9.9=16.31
  - Node5 belum ada dalam *openlist*
  - **MAKA:**

    Nilai Node5: *parent*=Node3, g*score*=6.41, f*score* =6.31

    Tambahkan Node5 ke dalam *openlist.*
- **Node 8**
  - *Cost*= 2.83
  - g*score* sementara= 3.41 + 2.83=6.24
  - f*score* sementara= 6.24+ 9.43=15.67
  - Node8 belum ada dalam *openlist*
  - **MAKA:**

    Nilai Node8: *parent*=Node3, g*score* =6.24, f*score* =15.67

    Tambahkan Node8 ke dalam *openlist.*
- **Node 12**
  - *Cost*= 4.0

- g*score* sementara= 3.41 +4.0=7.41
- f*score* sementara= 7.41 + 10.44=17.85
- Node 12 belum ada dalam *openlist*
- **MAKA:**

  Nilai Node 12 : *parent*=Node3, g*score* =7.41, f*score* = 7.85

  Tambahkan Node 12 ke dalam *openlist*.

Saat ini *openlist* berisi: Node8, Node 12, Node5, Node13, urutkan berdasarkan f*score* terkecil, maka node teratas: Node8.

Loop 4

a. Node sekarang : node 8
b. Tambahkan Node 8 ke dalam *exploredlist*
c. Hapus node Node 8 dari *openlist*
d. Node 8 = Node14 ? *False,* Lanjut
e. Setiap node tetangga Node 8 :

- **Node 3**
  - *Cost*= 2.83
  - g*score* sementara= 6.24 + 283=9.07
  - f*score* sementara= 9.07 + 12.21=21.28
  - Node3 ada dalam *exploredlist* dan nilai f*score*< f*score* sementara, maka Lanjut
- **Node 9**
  - *Cost*= 4.0
  - g*score* sementara= 6.24 + 4.0=10.24
  - f*score* sementara= 10.24 + 8.06=18.3
  - Node9 belum ada dalam *openlist*
  - **MAKA :**

    Nilai Node9: *parent*=Node8, g*score* =10.24, f*score* =18.3

    Tambahkan Node9ke dalam *openlist*.

Saat ini openlist berisi: Node5, Node12, Node13, Node9, urutkan berdasarkan fscore terkecil, maka node teratas: Node5.

Loop 5

a. Node sekarang : node 5

b. Tambahkan Node 5 ke dalam *exploredlist*

c. Hapus node Node 5 dari *openlist*

d. Node 5 = Node14 ? *False*, Lanjut

e. Setiap node tetangga Node 5:

- **Node 3**
  - *Cost*= 3.0
  - g*score* sementara= 6.41 + 3.0=9.41
  - f*score* sementara= 9.41 + 12.21=21.62
  - Node3 ada dalam *exploredlist* dan nilai f*score*< f*score* sementara, maka Lanjut
- **Node 4**
  - *Cost*= 2.0
  - g*score* sementara= 6.41 + 2.0=8.41
  - f*score* sementara= 8.41 + 11.4=19.81
  - Node4 belum ada dalam *openlist*
  - **MAKA :**

    Nilai Node4: *parent*=Node5, g*score* =8.41, f*score*=19.81

    Tambahkan Node4 ke dalam *openlist*.
- **Node 6**
  - *Cost*= 6.0
  - g*score* sementara= 6.41 + 6.0=12.41
  - f*score* sementara= 12.41 + 7.07=19.48
  - Node6 belum ada dalam *openlist*
  - **MAKA :**

    Nilai Node6: *parent*=Node5, g*score* =12.41, f*score* =19.48

    Tambahkan Node6 ke dalam *openlist*.
- **Node 10**
  - *Cost*= 7.0

- g*score* sementara= 6.41 + 7.0=13.41
- f*score* sementara= 13.41 + 7.0=20.41
- Node 10 belum ada dalam *openlist*
- **MAKA :**

  Nilai Node 10: *parent*=Node5, g*score*=13.41, f*score*=20.41

  Tambahkan Node 10 ke dalam *openlist*.

Saat ini openlist berisi: Node12, Node9, Node13, Node4, Node6, Node10, urutkan berdasarkan fscore terkecil, maka node teratas: Node12.

Loop 6

a. Node sekarang : node 12
b. Tambahkan Node 12 ke dalam *exploredlist*
c. Hapus node Node 12 dari openlist
d. Node 12 = Node14 ? *False*, Lanjut
e. Setiap node tetangga Node 12 :

- **Node 3**
  - *Cost*= 4.0
  - g*score* sementara= 7.41 + 4.0=11.41
  - f*score* sementara= 11.41 + 12.21=23.62
  - Node3 ada dalam *exploredlist* dan nilai f*score*< f*score* sementara, maka Lanjut
- **Node 13**
  - *Cost*= 1.0
  - g*score* sementara= 7.41 + 1.0=8.41
  - f*score* sementara= 8.41 + 11.4=19.81
- **Node 11**
  - *Cost*= 3.0
  - g*score* sementara= 7.41 + 3.0=10.41
  - f*score* sementara= 10.41 + 10.0=20.41
  - Node11 belum ada dalam *openlist*
  - **MAKA :**

Nilai Node11: *parent*=Node12, g*score*=10.41, f*score*=20.41

Tambahkan Node11 ke dalam *openlist.*

Saat ini *openlist* berisi: Node9, Node6, Node13, Node4, Node10, Node11,urutkan berdasarkan f*score* terkecil, maka node teratas: Node9.

Loop 7

a. Node sekarang: Node9

b. Tambahkan Node9 ke dalam *exploredlist*

c. Hapus node Node9 dari *openlist*

d. Node9 = Node14 ? *False*, Lanjut

e. Setiap node tetangga Node9:

- **Node 8**
  - *Cost*= 4.0
  - g*score* sementara= 10.24 + 4.0=14.24
  - f*score* sementara= 14.24 + 9.43=23.67
  - Node8 ada dalam *exploredlist* dan nilai f*score*< f*score* sementara, maka Lanjut
- **Node 10**
  - *Cost*= 1.41
  - g*score* sementara= 10.24 + 1.41=11.65
  - f*score* sementara= 11.65 + 7.0=18.65
  - f*score* sementara Node10 lebih kecil daripadaf*score* 10nya
  - **MAKA:**

    Nilai Node10: *parent*=Node9, g*score* =11.65, f*score*=18.65

    Tambahkan Node10ke dalam *openlist.*
- **Node 11**
  - *Cost*= 2.24
  - g*score* sementara= 10.24 + 2.24=12.48
  - f*score* sementara= 12.48 + 10.0=22.48

Saat ini *openlist* berisi: Node13, Node10, Node11, Node4, Node6, urutkan berdasarkan f*score* terkecil, maka node teratas: Node13.

Loop 8

a. Node sekarang: Node13

b. Tambahkan Node13 ke dalam *exploredlist*

c. Hapus Node13 dari *openlist*

d. Node13 = Node14 ? *False*, Lanjut

e. Setiap node tetangga Node13:

- **Node 1**
    - *Cost*= 7.0
    - gscore sementara= 7.0 + 7.0=14.0
    - fscore sementara= 14.0 + 14.87=28.87
    - Node1 ada dalam exploredlist dan nilai fscorenya < fscore sementara, maka Lanjut.
- **Node 12**
    - cost= 1.0
    - g*score* sementara= 7.0 + 1.0=8.0
    - f*score* sementara= 8.0 + 10.44=18.44
    - Node12 ada dalam *exploredlist* dan nilai f*score*< f*score* sementara, maka Lanjut

Saat ini *openlist* berisi: Node10, Node6, Node11, Node4, urutkan berdasarkan f*score* terkecil, maka node teratas: Node10.

Loop 9

a. Node sekarang: Node10

b. Tambahkan Node10 ke dalam *exploredlist*

c. Hapus node Node10 dari *openlist*

d. Node10 = Node14 ? *False,* Lanjut

e. Setiap node tetangga Node10:

- **Node 5**
    - *Cost*= 7.0
    - g*score* sementara= 11.65 + 7.0=18.65
    - f*score* sementara= 18.65 + 9.9=28.55

Saat ini loop sudah berhenti, maka jalur yang ditemukan dapat ditelusuri mulai dari node 9 adalah node 10 node 17 dan node 14

- Node5 ada dalam *exploredlist* dan nilai f*score*< f*score* sementara, maka Lanjut

- **Node 9**
  - *Cost*= 1.41
  - g*score* sementara= 11.65 + 1.41=13.06
  - f*score* sementara= 13.06 + 8.06=21.12
  - Node9 ada dalam *exploredlist* dan nilai f*score*< f*score* sementara, maka Lanjut.
- **Node 11**
  - *Cost*= 3.0
  - g*score* sementara= 11.65 + 3.0=14.65
  - f*score* sementara= 14.65 + 10.02=24.65
- **node 17**
  - *cost*= 2.0
  - g*score* sementara= 11.65 + 2.0=13.65
  - f*score* sementara= 13.65 + 7.28=20.93
  - Node17 belum ada dalam *openlist*
  - **MAKA :**

    Nilai Node17: *parent*=Node10, g*score*=13.65, f*score*=20.93

    Tambahkan Node17 ke dalam *openlist.*

Saat ini *openlist* berisi: Node6, Node4, Node11, Node17, urutkan berdasarkan f*score* terkecil, maka node teratas: Node6

Loop 10

a. Node sekarang: Node6

b. Tambahkan Node6 ke dalam *exploredlist*

c. Hapus node Node6 dari *openlist*

d. Node6 = Node14 ? *False*, Lanjut

e. Setiap node tetangga Node6:

- **Node 5**
  - *Cost*= 6.0

- g*score* sementara= 12.41 + 6.0=18.41
- f*score* sementara= 18.41 + 9.9=28.31
- Node5 ada dalam *exploredlist* dan nilai f*score*< f*score* sementara, maka Lanjut.

- **Node 7**
  - *Cost*= 3.61
  - g*score* sementara= 12.41 + 3.61=16.02
  - f*score* sementara= 16.02 + 5.39=21.41
  - **MAKA :**

    Nilai Node**7** : *parent*=Node6, g*score*=16.02, f*score*=21.41

    Tambahkan Node**7** ke dalam *openlist.*

Saat ini *openlist* berisi: Node4, Node17, Node11, Node7, Urutkan berdasarkan f*score*terkecil, maka node teratas: Node4.

Loop 11

a. Node sekarang : node 14
b. Tambahkan Node4 ke dalam *exploredlist*
c. Hapus node Node4 dari *openlist*
d. Node4 = Node 14 ? *False*, Lanjut
e. Setiap node tetangga Node4:

- **Node 5**
  - *Cost*= 2.0
  - g*score* sementara= 8.41 + 2.0=10.41
  - f*score* sementara= 10.41 + 9.9=20.31
  - Node5 ada dalam *exploredlist* dan nilai f*score*< f*score* sementara, maka Lanjut

Saat ini *openlist* berisi: Node11, Node17, Node7, urutkan berdasarkan fscore terkecil, maka node teratas: Node11.

Loop 12

a. Node sekarang: Node11
b. Tambahkan Node11 ke dalam *exploredlist*
c. Hapus node Node11 dari *openlist*

d. Node11 = Node14 ? *False*, Lanjut

e. Setiap node tetangga Node11:

- **Node 9**
    - *Cost*= 2.24
    - g*score* sementara= 10.41 + 2.24=12.65
    - f*score* sementara= 12.65 + 8.006=20.71
    - Node9 ada dalam *exploredlist* dan nilai f*score*< f*score* sementara, maka Lanjut
- **Node 10**
    - *Cost*= 3.0
    - g*score* sementara= 10.41 + 3.0=13.41
    - f*score* sementara= 13.41 + 7.0=20. 41
    - Node10 ada dalam *exploredlist* dan nilai f*score*< f*score* sementara, maka Lanjut
- **Node 12**
    - *Cost* = 3.0
    - g*score* sementara = 10.41 + 3.0=13.41
    - f*score* sementara = 13.41 + 10.44=23.85
    - Node12 ada dalam *exploredlist* dan nilai f*score*< f*score* sementara, maka Lanjut

Saat ini *openlist* berisi: Node17, Node7, urutkan berdasarkan f*score* terkecil, maka node teratas: Node17.

Loop 13

a. Node sekarang: Node17

b. Tambahkan Node17 ke dalam *exploredlist*

c. Hapus node Node17 dari *openlist*

d. Node17 = Node14 ? *False*, Lanjut

e. Setiap node tetangga Node17:

- **Node 10**
    - *Cost*= 2.0

- g*score* sementara= 13.65 + 2.0=15.65
- f*score* sementara= 15.65 + 7.0=22.65
- Node10 ada dalam *exploredlist* dan nilai f*score*< f*score* sementara, maka Lanjut

Saat ini *openlist* berisi: Node7, urutkan berdasarkan f*score* terkecil, maka node teratas: Node7.

Loop 14

a. Node sekarang: Node7

b. Tambahkan Node7 ke dalam *exploredlist*

c. Hapus node Node7 dari *openlist*

d. Node7 =Node14 ? *False*, Lanjut

e. Setiap node tetangga Node7:

- **Node 6**
  - *Cost*= 3.61
  - g*score* sementara= 16.02 + 3.61=19.63
  - f*score* sementara= 19.63 + 7.07=26.7
  - Node6 ada dalam *exploredlist* dan nilai f*score*< f*score* sementara, maka Lanjut
- **Node 14**
  - *Cost*= 5.39
  - g*score* sementara= 16.02 + 5.39=21.41
  - f*score* sementara= 21.41 + 0.0=21.41
  - Node14 belum ada dalam *openlist*
  - **MAKA:**

    Nilai Node14: *parent*=Node7, g*score*=21.41, f*score* =21.41

    Tambahkan Node14 ke dalam *openlist*.
- **Node 15**
  - Node sekarang : node 14
  - Tambahkan Node14 ke dalam *exploredlist*
  - Hapus node Node14 dari *openlist*

- Node14 = Node14 ? *True*, *Finish*
- Setiap node tetangga Node14

- **node 10**
  - *cost* = 7.0
  - g*score* sementara= 21.41 + 7.0 =28.41
  - f*score* sementara= 28.41 + 7.0 =35.41
  - Node10 ada dalam *exploredlist* dan nilai f*score* nya < f*score* sementara, maka Lanjut
- **Node 15**
  - *Cost*= 2.0
  - g*score* sementara= 21.41 + 2.0=23.41
  - f*score* sementara= 23.41 + 2.0=25.41
- **Node 16**
  - *Cost*= 2.0
  - g*score* sementara= 21.41 + 2.0=23.41
  - f*score* sementara= 21.41 + 2.0=23.41
  - Node16 belum ada dalam *openlist*
  - **MAKA :**

    Nilai Node16: *parent*=Node14, g*score*=23.41, f*score* =23.41

    Tambahkan Node16 ke dalam *openlist.*

Saat ini *loop* sudah berhenti, maka jalur yang ditemukan dapat ditelusuri mulai dari node14 ke stiap *parent*nya, yaitu:
Node14 *parentnya* Node7
Node7 *parentnya* Node6
Node6 *parentnya* Node5
Node5 *parentnya* Node3
Node3 *parentnya* Node2
Node2 *parentnya* Node1

Jadi, jalur dari Node 1 ke Node 14 terdekat adalah Node 1 – Node 2 – Node 3 – Node 5 – Node 6 – Node 7 – Node 14.

LAMPIRAN B PERHITUNGAN MANUAL PENGUJIAN VALIDASI HASIL

Perhitungan manual 1

Node *start* = 12

Tetangga dari node 12 = node 3, node 11 dan node 13

Loop 1

Node 3

- *Cost*= 4.0
- g *score*= 0+4=4
- *fscore*= 4 + 12.2= 16.2

Node 11

- *Cost*= 3.0
- g*score*= 0 + 3.0= 3
- *fscore*= 3 + 10= 13

Node13

- *Cost*= 7
- g*score*= 0 + 7 = 7
- *fscore*= 7 + 4.4 = 11.4

MAKA : node terkecil adalah node 11

Loop2

Tetangga dari node 11 adalah node 12, node 9 dan node 10

Node 12

- *Cost*= 30
- g*score*=3+3.0=6
- *fscore*= 6 + 10.44=16.44

Node 9

- *Cost*=2.23
- g*score*=13+2.23=5.23
- *fscore*=5.23+8.06=13.29

Node 10

- *Cost*=3.0
- g*score*=3+3.0= 6
- *fscore*=6+7= 13

Maka : node terkecil adalah node 10

Loop 3

Tetangga dari node 10 adalah node 17, node 5 dan node 14

Node 17

- *Cost*= 2.0

- g*score*=6+2.0=8
- *fscore*=8+7.28=15.28

Loop 5

- *Cost*= 7.0
- g*score*= 6+7.0=13
- *fscore* =13+9.89=22.89

Node 14

- *Cost*=7
- g*score*=6+7= 13
- *fscore* =13+0= 13

Maka : node terkecil adalah node 14

Loop4

Tetangga dari node 14 adalah node 10, node 16 node 10 dan node 15

Node 10

- *Cost*= 7.0
- g*score*=13 +7.0=2.0
- *fscore*=2.0+7 = 27

Node 16

- *Cost*= 2.0
- g*score*=13+2.0=15
- *fscore*=15+10.19 =15.19

Node 15

- *Cost*=2.0
- g*score* =13+2.0=15
- *fscore* =15+2=17

Saat ini loop sudah berhenti, maka jalur yang ditemukan dapat ditelusuri mulai dari node 12 adalah node 12- node 11 - node 10 – node 14.

Perhitungan manual 2

Node start 7

Loop1

Tetangga dari node 7 = node 6, node 15, node 14

Node 6

- *cost*= 8
- g *score*= 0+8= 8
- f *score*= 8+9.43 = 17.43

Node 15

- *cost* = 5
- g score = 0 + 5 = 5
- f score = 5 + 9.89 = 14.89

Node 14

- *cost* = 5.38
- g *score*=0+3.38 = 3.38
- f score =5.38+0=5.38

loop2

tetangga dari node 15 = node 7 dan node 14

Node 7

- *cost* =5
- g *score*=5+5=10
- f score=10+5.38=15.38

Node 14

- *cost*= 2
- g *score*=5+2=7
- f score=7+0=7

loop3

tetangga dari node 14 = node 10, node 15, node 7

Node 16

- *cost*= 2
- g *score* =7+2=9
- f score =9+10.19=19.19

Node 15

- *cost*= 2
- g *score* = 7+ 2 = 9
- f score = 9 + 2=11

Node 17

- *cost*= 5.38
- g *score* =7+5.38 =12.38
- f score=12.38+5.38=17.76

Saat ini loop sudah berhenti, maka jalur yang ditemukan dapat ditelusuri mulai dari node 7 adalah node 15, node 14

Perhitungan manual 3

Node start 4
Loop 1
Tetangga node 4 = node 5
Node 5

- *cost* = 2
- g *score* = 0 + 2 = 2
- f *score* = 2 + 9.89 = 11.89

loop 2
tetangga node 5 = node 3, node 10, node 6
Node 3
- *cost* = 3
- g *score* = 11.89 + 2 = 14.89
- f *score* = 14.89 + 12.2 = 27.89

Node 10
- *cost* = 7
- g *score* = 11.89 + 7 = 18.89
- f *score* = 18.89 + 5.38 = 24.27

Node 6
- *cost* = 6
- g *score* = 11.89 + 6 = 17.89
- f *score* = 17.89 + 7.07 = 24.96

loop 3
tetangga node 10 = node 9, node 11, node 17 ,node 14
Node 9
- *cost* = 1.4
- g *score* = 11.89 + 1.4 = 13.29
- f *score* = 13.29 + 8.06 = 21.35

Node 11
- *cost* = 3
- g *score* = 11.89 + 3 = 21.89
- f *score* = 21.89 + 10 = 31.89

Node 17
- *cost* = 2
- g *score* = 11.89 + 2 = 20.89
- f *score* = 20.89 + 7.28 = 28.17

Node 14
- *cost* = 7
- g *score* = 11.89 + 7 = 25.89
- f *score* = 25.89 + 0 = 25.89

Saat ini loop sudah berhenti, maka jalur yang ditemukan dapat ditelusuri mulai dari node 4 adalah node 5, node 10, node 14

Perhitungan manual 4
Node start 5

Loop 1
Tetangga node 5 = node 3, node 4, node 10, node 6
Node 3

- *cost* = 3
- g *score* = 0 + 3 = 3
- f *score* = 3 + 12.2 = 15.2

Node 4

- *cost* = 2
- g *score* = 0 + 2 = 2
- f *score* = 2 + 11.4 = 13.4

Node 10

- *cost* = 7
- g *score* = 0 + 7 = 7
- f *score* = 7 + 7 = 14

Node 6

- *cost* = 6
- g *score* = 0 + 6 = 6
- f *score* = 6 + 2.07 = 13.07

Loop 2

Tetangga node 10 = node 9, node 11, node 17, node 14

Node 9

- *cost* = 1.4
- g *score* = 7 + 1.4 = 8.4
- f *score* = 8.4 + 8.6 = 17

Node 11

- *cost* = 3
- g *score* = 7 + 3 = 10
- f *score* = 10 + 10 = 20

Node 17

- *cost* = 2
- g *score* = 7 + 2 = 9
- f *score* = 9 + 7.28 = 16.28

Node 14

- *cost* = 7
- g *score* = 7 + 7 = 14
- f *score* = 14 + 0 = 14

Saat ini loop sudah berhenti, maka jalur yang ditemukan dapat ditelusuri mulai dari node 5 adalah node 10, node 14

Perhitungan manual 5

Node start 3

Loop 1

Tetangga node 3 = node 2, node 12, node 5, node 8

Node 2

- *cost* = 2
- g *score* = 0 + 2 = 2

- f *score* = 2 + 13.45 = 15.45

Node 12

- *cost* = 4
- g *score* = 0 + 4 = 4
- f *score* = 4 + 10.44 = 14.44

Node 5

- *cost* = 3
- g *score* = 0 + 3 = 3
- f *score* = 3 + 9.89 = 12.89

Node 8

- *cost* = 2.82
- g *score* = 0 + 2.82 = 2.82
- f *score* = 2.82 + 9.43 = 12.25

Loop 2

Tetangga node 8 = node 9, node 3

Node 9

- *cost* = 4
- g *score* = 2.82 + 4 = 6.82
- f *score* = 6.82 + 8.06 = 14.48

Node 3

- *cost* = 2.82
- g *score* = 2.82 +2.82= 5.64
- f *score* = 5.64 + 12.2 = 17.84

Loop 3

Tetangga node 9 = node 8, node 11, node 10

Node 8

- *cost* = 4
- g *score* = 14.48 + 4 = 18.48
- f *score* = 18.48 + 9.43 = 27.91

Node 11

- *cost* = 2.23
- g *score* = 14.48 + 2.23 = 16.71
- f *score* = 16.71 + 10 = 26.71

Node 10

- *cost* = 1.4
- g *score* = 14.48 + 1.4 = 15.88
- f *score* = 15.88 + 7 = 22.88

Loop 4

Tetangga node 10 = node 9, node 11, node 17, node 5, node 14

Node 9

- *cost* = 1.4
- g *score* = 22.88 + 1.4 = 24.28

- f *score* = 24.28 + 8.06 = 32.34

Node 11

- *cost* = 3
- g *score* = 22.88 + 3 = 25.28
- f *score* = 25.28 + 10 = 35.88

Node 17

- *cost* = 2
- g *score* = 22.88 + 2 = 24.88
- f *score* = 24.88 + 7.28 = 32.16

Node 5

- *cost* = 7
- g *score* = 22.88 + 7 = 29.88
- f *score* = 29.88 + 9.89 = 39.77

Node 14

- *cost* = 7
- g *score* = 22.88 + 7 = 29.88
- f *score* = 29.88 + 0 = 29.88

Saat ini loop sudah berhenti, maka jalur yang ditemukan dapat ditelusuri mulai dari node 3 adalah node 8, node 9, node 10 dan node 14

Perhitungan manual 6

Node start 2
Loop 1
Tetangga node 2 = node 1, node 3
Node 1

- *cost* = 1.4
- g *score* = 0 + 1.4 = 1.4
- f *score* = 1.4 + 14.86 = 16.26

Node 3

- *cost* = 2
- g *score* = 0 + 2 = 2
- f *score* = 2 + 12.2 = 14.2

Loop 2
Tetangga node 3 = node 8, node 12, node 5, node 2
Node 8

- *cost* = 2.82
- g *score* = 2 + 2.82 = 4.82
- f *score* = 4.82 + 9.43 = 14.25

Node 12

- *cost* = 4
- g *score* = 2 + 4 = 4
- f *score* = 4 + 10.44 = 16.44

Node 5

- *cost* = 3
- g *score* = 2 + 3 = 5
- f *score* = 5 + 12.22 = 17.2

Node 2

- *cost* = 2
- g *score* = 2 +2 = 4
- f *score* = 4 + 13.45 = 17.45

Loop 3

Tetangga node 8 = node 3, node 9

Node 3

- *cost* = 2.82
- g *score* = 4.82 + 2.82 = 7.64
- f *score* = 7.64 + 12.2 = 19.84

Node 9

- *cost* = 4
- g *score* = 4.82 + 4 = 8.82
- f *score* = 8.82 + 8.06 = 16.88

Loop 4

Tetangga node 4 = node 8, node 11, node 10

Node 8

- *cost* = 4
- g *score* = 8.82 + 4 = 12.82
- f *score* = 12.82 + 9.43 = 22.25

Node 11

- *cost* = 2.23
- g *score* = 8.82 + 2.23 = 11.05
- f *score* = 11.05 + 10 = 21.05

Node 10

- *cost* = 1.4
- g *score* = 8.82 + 1.4 = 10.22
- f *score* = 10.22 + 7 = 17.22

Loop 5

Tetangga node 10 = node 9, node 11, node 17, node 5

Node 9

- *cost* = 1.4
- g *score* = 10.22 + 1.4 = 11.62
- f *score* = 11.62 + 8.06 = 19.68

Node 11

- *cost* = 3
- g *score* = 10.22 + 3 = 13.22
- f *score* = 13.22+ 10 = 23.22

Node 17

- *cost* = 2
- g *score* = 10.22 + 2 = 12.22
- f *score* = 12.22+ 2 = 14.22

Node 5

- *cost* = 7
- g *score* = 10.22 + 7 = 17.22
- f *score* = 17.22 + 9.89 = 27.11

Saat ini loop sudah berhenti, maka jalur yang ditemukan dapat ditelusuri mulai dari node 2 adalah node 3, node 8, node 9, node 10 dan node 17

Perhitungan manual 7

Node start 11
Loop 1
Tetangga node 11 = node 12, node 9, node 10
Node 12

- *cost* = 3
- g *score* = 0 + 3 =3
- f *score* = 3 + 10.44 = 13.44

Node 9

- *cost* = 2.23
- g *score* = 0 + 2.23 = 2.23
- f *score* = 2.23 + 8.06 = 10.29

Node 10

- *cost* = 3
- g *score* = 0 + 3 = 3
- f *score* = 3 +7 = 10

Loop 2
Tetangga node 10 = node 17, node 9, node 11, node 14
Node 17

- *cost* = 2
- g *score* = 3 + 2 = 5
- f *score* = 5 + 7.28 = 12.28

Node 9

- *cost* = 1.4
- g *score* = 3 + 1.4 = 4.4
- f *score* = 4.4 + 8.06 = 12.46

Node 11

- *cost* = 3
- g *score* = 3 + 3 = 6
- f *score* = 6 + 10 = 16

Node 14

- *cost* = 7
- g *score* = 3 + 7 = 10

- f *score* = 10 + 0 = 10

Saat ini loop sudah berhenti, maka jalur yang ditemukan dapat ditelusuri mulai dari node 11 adalah node 3, node 10 dan node 14

Perhitungan manual 8

Node start 15
Loop 1
Tetangga node 15 = node 7 dan node 14
Node 7
- *cost* = 5
- g *score* = 0 + 5 =5
- f *score* = 5 + 5.38 = 10.38

Node 14
- *cost* = 2
- g *score* = 0 + 2 = 2
- f *score* = 2 + 0 = 0

Saat ini loop sudah berhenti, maka jalur yang ditemukan dapat ditelusuri mulai dari node 15 adalah node 7 dan node 14

Perhitungan manual 9

Node start 10
Loop 1
Tetangga node 10 = node 17, node 9, node 5, node 10
Node 17
- *cost* = 2
- g *score* = 0 + 2 =2
- f *score* = 2 + 7.28 = 9.28

Node 9
- *cost* = 1.4
- g *score* = 0 + 1.4 =1.4
- f *score* = 1.4 + 8.06 = 9.46

Node 11
- *cost* = 3
- g *score* = 0 + 3 =3
- f *score* = 3 + 12.2 = 15.2

Node 5
- *cost* = 7
- g *score* = 0 + 7 =7
- f *score* = 7 + 9.89 = 18.89

Loop 2
Tetangga node 17 = node 10

Node 10

- *cost* = 2
- g *score* = 2 + 2 =4
- f *score* = 4 + 7 = 11

Saat ini loop sudah berhenti, maka jalur yang ditemukan dapat ditelusuri mulai dari node 10 adalah node 17 dan node 14

Perhitungan manual 9

Node start 8
Loop 1
Tetangga node 8 = node 3 dan node 9
Node 3

- *cost* = 2.82
- g *score* = 0 + 2.82 = 0.82
- f *score* = 0.82 + 12.2 = 13.2

Node 9

- *cost* = 4
- g *score* = 0 + 4 = 4
- f *score* = 4 + 8.06 = 12.06

Loop 2
Tetangga node 9 = node 8 node 11 dan node 10
Node 8

- *cost* = 4
- g *score* = 4 + 4 = 8
- f *score* = 8 + 9.43 = 16.3

Node 11

- *cost* = 2.23
- g *score* = 4 + 2.23 = 6.23
- f *score* = 6.23 + 10 = 16.23

Node 10

- *cost* = 1.4
- g *score* = 4 + 1.4 = 5.4
- f *score* = 5.4 + 7 = 12.4

Loop 3
Tetangga node 10 = node 9 node 11 node 17 dan node 10
Node 9

- *cost* = 1.4
- g *score* = 5.4 + 4 = 9.4
- f *score* = 9.4 + 8.06 = 17.46

Node 11

- *cost* = 3
- g *score* = 5.4 + 3 = 8.4
- f *score* = 8.4 + 10 = 18.4

Node 17

- *cost* = 2
- g *score* = 5.4 + 2 = 7.4
- f *score* = 7.4 + 7.28 = 14.68

Node 14

- *cost* = 7
- g *score* = 5.4 + 7 = 12.4
- f *score* = 12.4 + 0 = 12.4

Saat ini loop sudah berhenti, maka jalur yang ditemukan dapat ditelusuri mulai dari node 8 adalah node 3 node 9 dan node 10

Perhitungan manual 10

Node start 13
Loop 1
Tetangga node 13 = node 1 dan node 12
Node 1

- *cost* = 7
- g *score* = 0 + 7 = 7
- f *score* = 7 + 14.86 = 21.86

Node 12

- *cost* = 1
- g *score* = 0 + 1 = 1
- f *score* = 1 + 10.44 = 11.44

Loop 2
Tetangga node 12 = node 13 node 3 dan node 11
Node 13

- *cost* = 1
- g *score* = 1 + 1 = 2
- f *score* = 2 + 11.4 = 13.4

Node 3

- *cost* = 4
- g *score* = 1 + 4 = 5
- f *score* = 5 + 12.2 = 17.2

Node 11

- *cost* = 3
- g *score* = 1 + 3 = 4
- f *score* = 4 + 10 = 14

Loop 3
Tetangga node 11 = node 9 node 10 dan node 12
Node 9

- *cost* = 2.23
- g *score* = 4 + 2.23 = 6.23
- f *score* = 6.23 + 8.06 = 14.29

Node 10

- *cost* = 3

- g *score* = 4 + 3 = 7
- f *score* = 7 + 7 = 14

Node 12

- *cost* = 3
- g *score* = 4 + 3 = 7
- f *score* = 7 + 10.44 = 17.44

Loop 4

Tetangga node 10 = node 9 node 11 node 17 node 5 dan node 14

Node 9

- *cost* = 2.23
- g *score* = 7 + 2.23 = 9.23
- f *score* = 9.23 + 8.06 = 17.29

Node 11

- *cost* = 3
- g *score* = 7 + 3 = 10
- f *score* = 10 + 10 = 20

Node 17

- *cost* = 2
- g *score* = 7 + 2 = 9
- f *score* = 9 + 7.28 = 16.28

Node 5

- *cost* = 7
- g *score* = 7 + 7 = 14
- f *score* = 14 + 9.89 = 23.89

Node 14

- *cost* = 7
- g *score* = 7 + 7 = 14
- f *score* = 14 + 0 = 14

Saat ini loop sudah berhenti, maka jalur yang ditemukan dapat ditelusuri mulai dari node 13 adalah node 12 node 11 node 10 dan node 10

Perhitugan manual 11

Node start 3

Loop 1

Tetangga node 3 = node 2, node 12, node 5, node 8

Node 2

- *cost* = 2
- g *score* = 0 + 2 = 2
- f *score* = 2 + 13.45 = 15.45

Node 12

- *cost* = 4
- g *score* = 0 + 4 = 4
- f *score* = 4 + 10.44 = 14.44

Node 5

- *cost* = 3
- g *score* = 0 + 3 = 3
- f *score* = 3 + 9.89 = 12.89

Node 8

- *cost* = 2.82
- g *score* = 0 + 2.82 = 2.82
- f *score* = 2.82 + 9.43 = 12.25

Loop 2

Tetangga node 8 = node 9, node 3

Node 9

- *cost* = 4
- g *score* = 2.82 + 4 = 6.82
- f *score* = 6.82 + 8.06 = 14.48

Node 3

- *cost* = 2.82
- g *score* = 2.82 +2.82= 5.64
- f *score* = 5.64 + 12.2 = 17.84

Loop 3

Tetangga node 9 = node 8, node 11, node 10

Node 8

- *cost* = 4
- g *score* = 14.48 + 4 = 18.48
- f *score* = 18.48 + 9.43 = 27.91

Node 11

- *cost* = 2.23
- g *score* = 14.48 + 2.23 = 16.71
- f *score* = 16.71 + 10 = 26.71

Node 10

- *cost* = 1.4
- g *score* = 14.48 + 1.4 = 15.88
- f *score* = 15.88 + 7 = 22.88

Loop 4

Tetangga node 10 = node 9, node 11, node 17, node 5, node 14

Node 9

- *cost* = 1.4
- g *score* = 22.88 + 1.4 = 24.28
- f *score* = 24.28 + 8.06 = 32.34

Node 11

- *cost* = 3
- g *score* = 22.88 + 3 = 25.28
- f *score* = 25.28 + 10 = 35.88

Node 17

- *cost* = 2
- g *score* = 22.88 + 2 = 24.88
- f *score* = 24.88 + 7.28 = 32.16

Node 5

- *cost* = 7
- g *score* = 22.88 + 7 = 29.88
- f *score* = 29.88 + 9.89 = 39.77

Node 14

- *cost* = 7
- g *score* = 22.88 + 7 = 29.88
- f *score* = 29.88 + 0 = 29.88

Saat ini loop sudah berhenti, maka jalur yang ditemukan dapat ditelusuri mulai dari node 3 adalah node 8, node 9, node 10 dan node 14

Perhitungan manual 12

Node start 17

Loop 1

Tetangga node 17 = node 10

Node 10

- *cost* = 2
- g *score* = 0 + 2 = 2
- f *score* = 2 + 7 = 9

Loop 2

Tetangga node 10 = node 9, node 11, node 17, node 5, node 14

Node 9

- *cost* = 1.4
- g *score* = 2 + 1.4 = 3.4
- f *score* = 3.4 + 8.06 = 11.46

Node 11

- *cost* = 3
- g *score* = 2 + 3 = 5
- f *score* = 5 + 10 = 15

Node 17

- *cost* = 2
- g *score* = 2 + 2 = 4
- f *score* = 4 + 7.28 = 11.28

Node 5

- *cost* = 7
- g *score* = 2 + 7 = 9
- f *score* = 9 + 9.89 = 18.89

Node 14

- *cost* = 7
- g *score* = 2 + 7 = 9
- f *score* = 9 + 0 = 9

Saat ini loop sudah berhenti, maka jalur yang ditemukan dapat ditelusuri mulai dari node 3 adalah node 8, node 9, node 10 dan node 14

Perhitungan manual 13

Node start 6
Loop 1
Tetangga node 6 = node 5 dan node 7
Node 5
- *cost* = 6
- g *score* = 0 + 6 = 6
- f *score* = 6 + 9.89 = 15.89

Node 7
- *cost* = 2
- g *score* = 0 + 2 = 2
- f *score* = 2 + 12.2 = 14.2

Loop 2
Tetangga node 7 = node 15 dan node 14
Node 15
- *cost* = 5
- g *score* = 2 + 5 = 7
- f *score* = 7 + 2 = 9

Node 14
- *cost* = 0
- g *score* = 2 + 0 = 2
- f *score* = 0 + 0 = 0

Saat ini loop sudah berhenti, maka jalur yang ditemukan dapat ditelusuri mulai dari node 6 adalah node 7, node 15 dan node 14

Perhitungan manual 14

Node start 16
Loop 1
Tetangga node 16 = node 14
Node 14
- *cost* = 2
- g *score* = 0 + 2 = 2
- f *score* = 2 + 0 = 2

Saat ini loop sudah berhenti, maka jalur yang ditemukan dapat ditelusuri mulai dari node 16 adalah node 14

Perhitungan manual 15

Node start 9
Loop 1

Tetangga node 9 = node 11 node 10 dan node 8

Node 11

- *cost* = 2.23
- g *score* = 0 + 2.23 = 2.23
- f *score* = 2.23 + 10 = 12.23

Node 10

- *cost* = 1.4
- g *score* = 0 + 1.4 = 1.4
- f *score* = 1.4 + 7 = 8.4

Node 8

- *cost* = 4
- g *score* = 0 + 4 = 4
- f *score* = 4 + 9.43 = 13.43

Loop 2

Tetangga node 10 = node 9 node 11 node 17 dan node 5

Node 9

- *cost* = 1.4
- g *score* = 1.4 + 1.4 = 2.8
- f *score* = 2.8 + 8.06 = 10.86

Node 11

- *cost* = 3
- g *score* = 1.4 + 3 = 4.4
- f *score* = 4.4 + 10 = 14.4

Node 17

- *cost* = 2
- g *score* = 1.4 + 2 = 3.4
- f *score* = 3.4 + 7.28 = 10.68

Node 5

- *cost* = 7
- g *score* = 1.4 + 7 = 8.4
- f *score* = 8.4 + 9.89 = 18.29

Loop 3

Tetangga node 17 = node 10 node 14 dan node 5

Node 10

- *cost* = 2
- g *score* = 3.4 + 2 = 5.4
- f *score* = 5.4 + 7 = 12.4

Node 5

- *cost* = 7
- g *score* = 3.4 + 7 = 10.4
- f *score* = 10.4 + 4.89 = 20.89

Node 14

- *cost* = 7
- g *score* = 3.4 + 7 = 10.4
- f *score* = 10.4 + 0 = 10.4