PENCARIAN RUTE TERPENDEK PADA LABIRIN MENGGUNAKAN METODE A*

SKRIPSI

Diajukan Untuk Memenuhi Sebagian

Persyaratan Memperoleh Gelar Sarjana

OLEH:

RENGGA DIONATA PUTRA

NIM: 0710633069

KEMENTRIAN PENDIDIKAN NASIONAL

UNIVERSITAS BRAWIJAYA

FAKULTAS TEKNIK

JURUSAN TEKNIK ELEKTRO

MALANG

2013

Abstrak

Rengga Dionata Putra, Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya, 2012, **"Pencarian Rute Terdekat Pada Labirin Menggunakan Metode A* "**, Dosen Pembimbing **Ir. Muhammad Aswin, MT**. Dan **Waru Djuriatno, ST., MT.**

Labirin adalah sebuah jaringan dari jalur jalur yang saling berhubungan untuk dilalui dari awal hingga akhir yang dimaksudkan untuk sebuah tantangan, manusia mungkin masih dapat menyelesaikan masalah pencarian ruang terdekat yang sederhana, tetapi jika jumlah rute yang ada sudah sedemikian banyaknya, maka kita akan mengalami kesulitan dan akan memakan waktu yang lama untuk menyelesaikannya.

Pencarian rute terdekat adalah usaha untuk mencari rute yang paling dekat dari possisi awal hingga akhir dengan beban paling ringan atau sedikit dibandingkan dengan seluruh rute yang ada. Pada skripsi ini akan dibuat suatu progam aplikasi untuk mencari rute terdekat pada labirin. Aplikasi ini menggunakan algoritma A* yang menerapkan suatu heuristik, dan penulis menggunakan algoritma Djikstra sebagai pembanding. Hasil dari aplikasi ini berupa rute terpendek yang dapat dilalui pada labirin

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Allah SWT, karena dengan rahmat, taufik dan hidayah-Nya lah skripsi ini dapat diselesaikan.

Skripsi berjudul " *pencarian rute terpendek pada labirin menggunakan metode A** " ini disusun untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Teknik di Jurusan Teknik Elektro Universitas Brawijaya.

Penulis menyadari bahwa penyusunan skripsi ini tidak terlepas dari bantuan berbagai pihak. Oleh karena itu, dengan ketulusan dan kerendahan hati penulis menyampaikan terima kasih kepada:

1. Ayah Abu Mansjur dan Ibu Sus Merry, Anita Olivia dan Victor H keluarga yang selalu memberi doa, semangat, kasih sayang, perhatian, serta dukungan baik materi maupun non-materi yang tak ternilai yang telah diberikan.
2. Bapak Dr. Ir. Sholeh Hadi Pramono, M.S. selaku Ketua Jurusan Teknik Elektro Universitas Brawijaya dan Bapak M. Aziz Muslim, ST., MT., Ph.D selaku Sekertaris Jurusan Teknik Elektro Universitas Brawijaya.
3. Bapak Waru Djuriatno ST., MT. selaku Ketua Kelompok Dosen Keahlian Elektronika Jurusan Teknik Elektro Universitas Brawijaya atas segala bimbingan, nasehat, pengarahan, motivasi, saran, dan masukan yang telah diberikan.
4. Bapak Waru Djuriatno ST., MT. Dan Bapak Ir. Muhammad Aswin MT. selaku Dosen Pembimbing atas segala bimbingan, nasehat, pengarahan, motivasi, saran, dan masukan yang telah diberikan.
5. Bapak dan Ibu dosen beserta staff dan karyawan Jurusan Teknik Elektro, baik secara langsung maupun tidak langsung telah membantu menyelesaikan skripsi ini.
6. Semua teman teman CORE 2007 yang sudah banyak membantu selama perkuliahan di jurusan elektro UB.
7. Ustadz Murdifin yang selalu memberi semangat, dan ilmu yang selalu di berikan selama penulisan skripsi ini.

8. ARS (anggun rosspita sari) terima kasih atas perhatian dan semangat yang pernah dibagi

Pada akhirnya, penulis menyadari bahwa skripsi ini masih belum sempurna. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang membangun. Penulis berharap semoga skripsi ini dapat bermanfaat bagi pembangunan ilmu pengetahuan dan teknologi.

Malang, Maret 2013

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR FLOWCHART

DAFTAR TABEL

BAB I
PENDAHULUAN

I. Judul

PENCARIAN RUTE TERPENDEK PADA LABIRIN MENGGUNAKAN ALGORITMA A*

II. Latar Belakang Masalah

Pencarian rute terdekat, adalah usaha untuk mencari rute yang paling dekat dari posisi awal hingga akhir dengan beban paling ringan atau sedikit dibandingkan dengan seluruh rute yang ada. Walaupun bagi manusia hal ini masih tergolong mudah, tetapi lain halnya dengan komputer, karena mereka tidak memiliki insting untuk menentukan jarak tanpa penggunaan algoritma.

Manusia mungkin masih dapat menyelesaikan masalah pencarian ruang terdekat yang sederhana, tetapi jika jumlah rute yang ada sudah sedemikian banyaknya, maka manusia akan mengalami kesulitan dan akan memakan waktu yang terlalu lama untuk menyelesaikannya. Inilah alasan mengapa banyak peneliti dan praktisi yang berusaha untuk menemukan metode yang paling efektif dan efisien bagi komputer untuk menggantikan manusia dalam pencarian rute terdekat.

Salah satu lingkungan yang banyak digunakan manusia sebagai permainan yang juga merupakan salah satu bentuk lingkungan dari pencarian rute terdekat yaitu adalah labirin. Labirin merupakan sebuah ruang yang memiliki banyak jalur dan persimpangan, dan pemain harus menemukan rute terdekat dari posisi awal hingga posisi akhir.

Algoritma A* adalah sebuah algoritma yang telah diperkaya dengan menerapkan suatu *heuristik*, algoritma ini membuang langkah-langkah yang tidak perlu dengan pertimbangan bahwa langkah-langkah yang dibuang sudah pasti merupakan langkah yang tidak akan mencapai solusi yang diinginkan.

Heuristik adalah nilai yang memberi harga pada tiap simpul yang memandu

A* mendapatkan solusi yang diinginkan. Dengan *heuristik* yang benar, maka A* pasti akan mendapatkan solusi (jika memang ada solusinya) yang dicari. Dengan kata lain, heuristik adalah fungsi optimasi yang menjadikan algoritma A* lebih baik dari pada algoritma lainnya

Algoritma A* menggabungkan jarak estimasi/heuristik $h(n)$ dan jarak sesungguhnya $g(n)$ dalam membantu penyelesaian persoalan, Algoritma ini akan mengunjungi setiap node, selama node tersebut merupakan nede yang terbaik. Jika node yang sedang dikunjungi tidak mengarah kepada solusi yang diinginkan, maka akan melakukan runut balik ke arah node sebelumnya untuk mencari node lainnya yang memberikan pilihan untuk melanjutkan ke node – node berikutnya . Bila tidak ada juga, maka akan terus mengulang mencari ke arah node akar sampai ditemukan simpul yang lebih baik.

III. Rumusan Masalah

Berdasarkan latar belakang di atas maka rumusan masalah dari skripsi ini adalah:

1. Mencari solusi pada labirin.
2. Pencarian solusi dilakukan dengan menggunakan algoritma A*

IV. Tujuan

Tujuan dari Tugas Akhir ini yaitu membangun sebuah aplikasi yang dapat menyelesaikan masalah pencarian rute terdekat untuk kasus labirin melalui algoritma A*.

V. Batasan Masalah

1. Lingkup permasalahan pencarian rute terdekat yang diangkat adalah pergerakan agen dari satu titik asal menuju satu titik tujuan (*node to node*).
2. Input labirin menggunakan Larik, kemudian diterjemahkan ke bentuk model (visual). Model yang digunakan adalah labirin dengan jalur-jalur yang saling berhubungan dan direpresentasikan dalam bentuk grid, bukan berupa area atau graf.
3. Tidak ada jalan buntu yang ak bersolusi pada labirin.
4. Koordinat finish telah diketahui.

5. Maksimum luas area labirin 16 blok x 16 blok

VI. Sistematika Penulisan

Sistematika penulisan laporan skripsi ini adalah sebagai berikut :

BAB I Pendahuluan

Memuat latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi pembahasan, dan sistematika pembahasan.

BAB II Tinjauan Pustaka

Membahas teori-teori yang mendukung dalam perancangan dan pembuatan aplikasi.

BAB III Metodologi Penelitian

Berisi tentang metode penelitian yang digunakan dalam perancangan dan pengujian sistem.

BAB VI Perancangan & Implementasi

Mengimplementasikan hasil perancangan ke dalam algoritma-algoritma dan kode-kode program dalam bahasa pemrograman.

BAB V Pengujian

Memuat hasil pengujian aplikasi terhadap masalah

.BAB VI Kesimpulan

Memuat kesimpulan dan saran saran

BAB II
TINJAUAN PUSTAKA

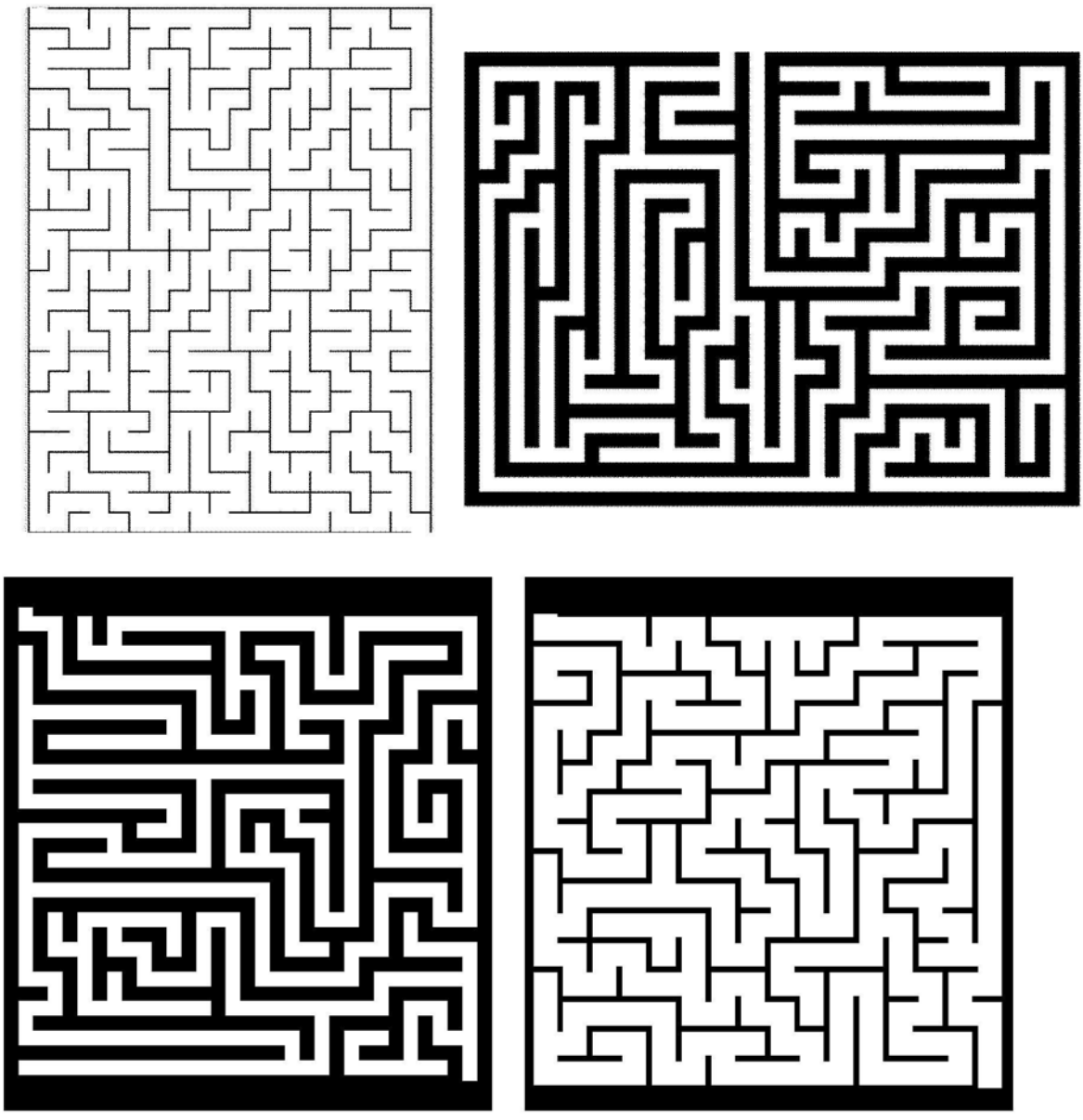
2.1 Labirin

Labirin adalah sebuah jaringan dari jalur jalur yang saling berhubungan untuk di lalui dari awal hingga akhir yang dimaksudkan untuk sebuah tantangan.
. Klasifikasi yang dimiliki sebuah labirin yaitu antara lain dimensi, hiperdimensi, topologi, struktur unit pembentuknya, rute, tekstur, dan fokus. Sebuah labirin memiliki gabungan dari seluruh klasifikasi yang ada, tetapi mungkin memiliki tipe yang berbeda beda. Di bawah ini beberapa jenis labirin

1. N-Ary atau labirin berbentuk pohon Pohon yang mempunyai akar yang setiap simpul cabangnya mempunyai paling banyak n buah anak disebut pohon n-ary . Dengan memodelkan labirin dengan n-ary, kita akan lebih mudah menemukan jalan keluar dalam permainan labirin.
2. Perfect yaitu labirin yang dipastikan tidak ada loop atau area yang tidak bisa di jangkau
3. Braid algoritma yang tidak mempunyai jalan buntu
4. *Unicursal* labirin yang tidak mempunyai persimpangan dan pemilihan jalur
5. *Weave* hampir sama dengan perfect tapi ada dinding yang tak terhubung dengan dinding lain

Sebuah labirin tidak harus memenuhi klasifikasi yang telah disebutkan di atas. Untuk labirin yang kompleks bisa memadukan satu atau lebih dari sifat sifat aatau klasifikasi yang telah disebutkan

Berikut beberapa contoh gambar labirin :



Gambar 2.1.1 labirin

2.2 Routing

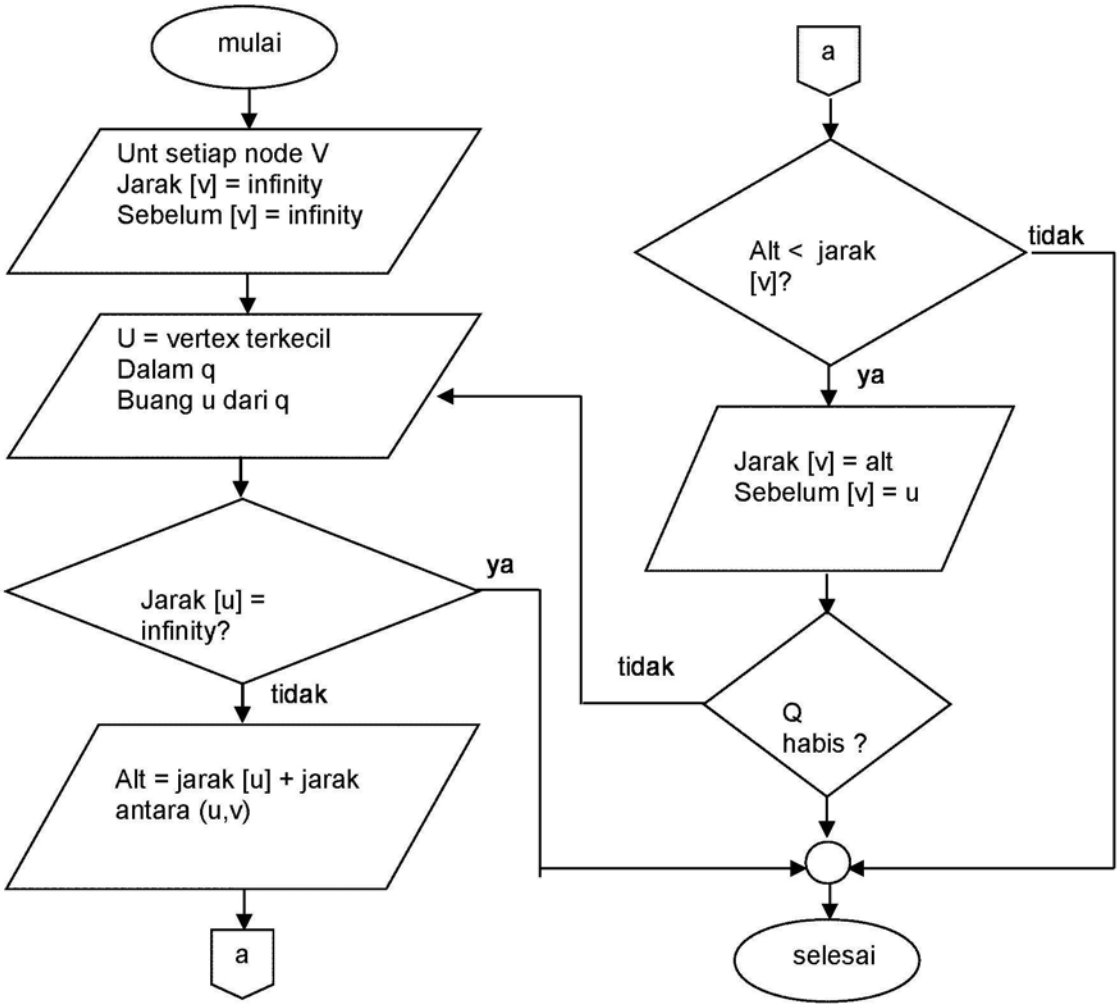
Routing/Perutean proses untuk memilih jalur (path) yang harus dilalui oleh paket. Semua routing bertujuan mencari rute tersingkat untuk mencapai tujuan. Dan masing-masing protokol mempunyai cara dan metodenya sendiri-sendiri.

Pada aplikasi yang di buat agen harus bisa mencapai garis finish sebagai alamat akhir yang harus dituju, dengan proses pemilihan dan perhitungan jalur pada setial langkahnya

2.3 Djikstra

Algoritma ini adalah salah satu bentuk algoritma greedy. Algoritma ini termasuk algoritma pencarian graf yang digunakan untuk menyelesaikan masalah lintasan terpendek. Dengan menggunakan fungsi biaya g(n) setiap simpul. Proses ini dilakukan berulang sampai simpul tujuan diperiksa. Algoritma Dijkstra memang menjamin didapatkannya jalur optimal, tetapi algoritma ini mempunyai kelemahan. Pemeriksaan simpul akan dilakukan ke segala arah yang dimungkinkan yang pada akhirnya seluruh simpul pada sebuah graf akan diperiksa. Hal ini menyebabkan algoritma ini bekerja dengan lambat dan menggunakan memori yang besar sehingga waktu yang diperlukan untuk menemukan solusi akan semakin besar pula.

Berikut flowchart algoritma Djikstra :



Flowchart 2.3 Djikstra

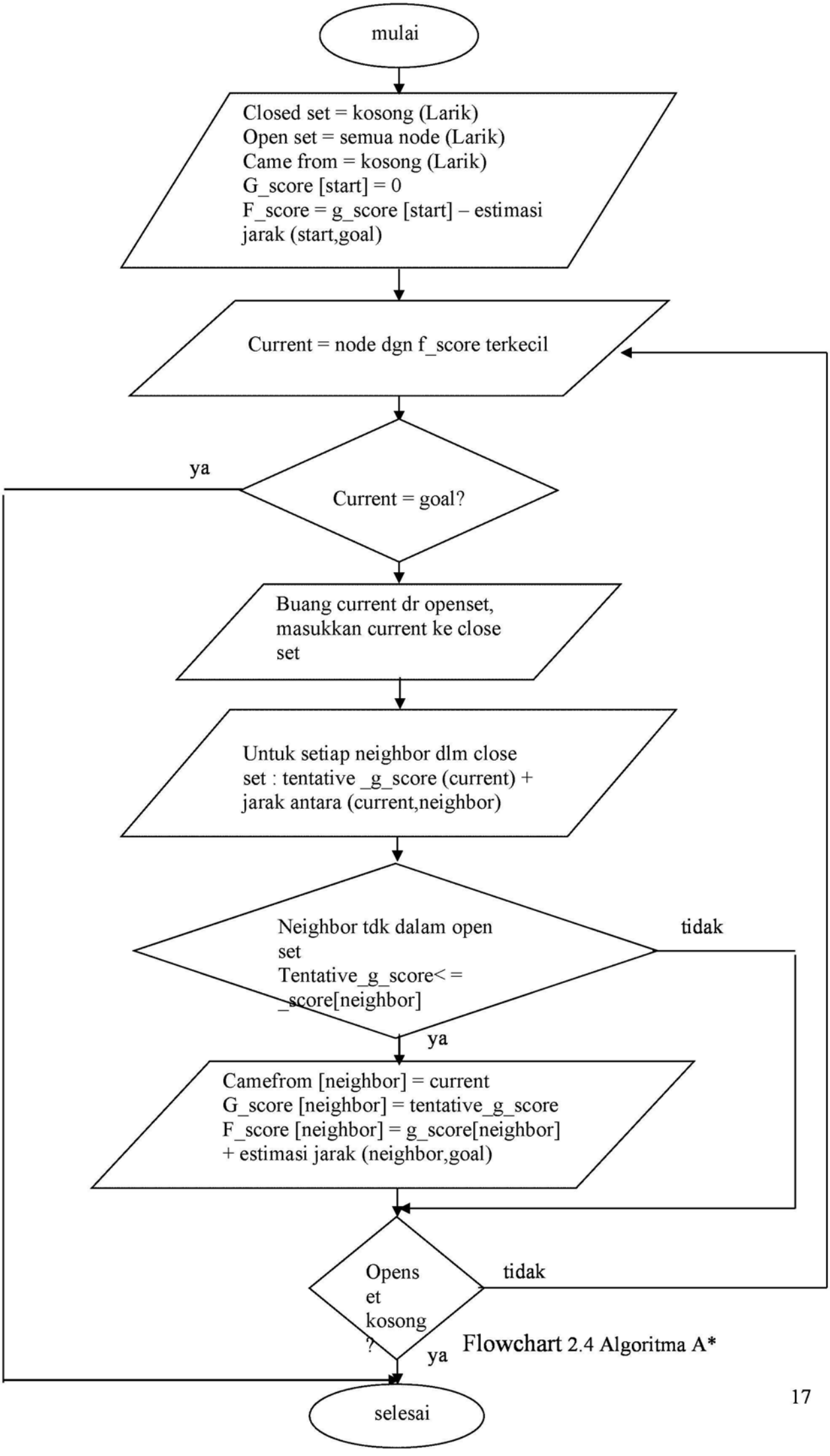
2.4 A*

Algoritma A* menyelesaikan masalah yang menggunakan graf untuk perluasan ruang statusnya. Dengan menerapkan suatu heuristik. Heuristik adalah nilai yang memberi nilai pada tiap simpul yang memandu A* mendapatkan solusi yang diinginkan. Dengan heuristik, maka A* pasti akan mendapatkan solusi (jika memang ada solusinya). Dengan kata lain, heuristik adalah fungsi optimasi yang menjadikan algoritma A* lebih baik dari pada algoritma lainnya.

Algoritma ini membuang langkah-langkah yang tidak perlu dengan pertimbangan bahwa langkah-langkah yang dibuang sudah pasti merupakan langkah yang tidak akan mencapai solusi yang diinginkan.

Algoritma ini akan mengunjungi secara mendalam selama simpul tersebut merupakan simpul yang terbaik. Jika simpul yang sedang dikunjungi ternyata tidak mengarah kepada solusi yang diinginkan, maka akan melakukan runut balik ke arah simpul akar untuk mencari simpul anak lainnya yang lebih menjanjikan dari pada simpul yang terakhir dikunjungi. Bila tidak ada juga, maka akan terus mengulang mencari ke arah simpul akar sampai ditemukan simpul yang lebih baik untuk dibangkitkan. A* baru berhenti ketika mendapatkan solusi yang dianggap solusi terbaik.

Berikut Algoritma A* :



Flowchart 2.4 Algoritma A*

BAB III
METODE PENELITIAN

Pada tahap ini akan dijelaskan mengenai langkah – langkah yang akan dilakukan untuk merancang dan mengimplementasikan perangkat lunak yang akan dibuat.adapun langkah – langkah yang akan dilakukan adalah sebagai berikut :

3.1 Studi Literatur

Dalam penyusunan skripsi ini, pengumpulan data dilakukan dengan melakukan studi literatur dengan sasaran tinjauan antara lain :

1. Pustaka Referensi
2. Informasi Internet
3. Pustaka Penunjang

Studi literatur yang dilakukan bertujuan untuk mengkaji hal – hal yang berhubungan dengan teori – teori yang mendukung dalam perencanaan dan perealisasian aplikasi. Adapun teori teori yang dikaji adalah sebagai berikut :

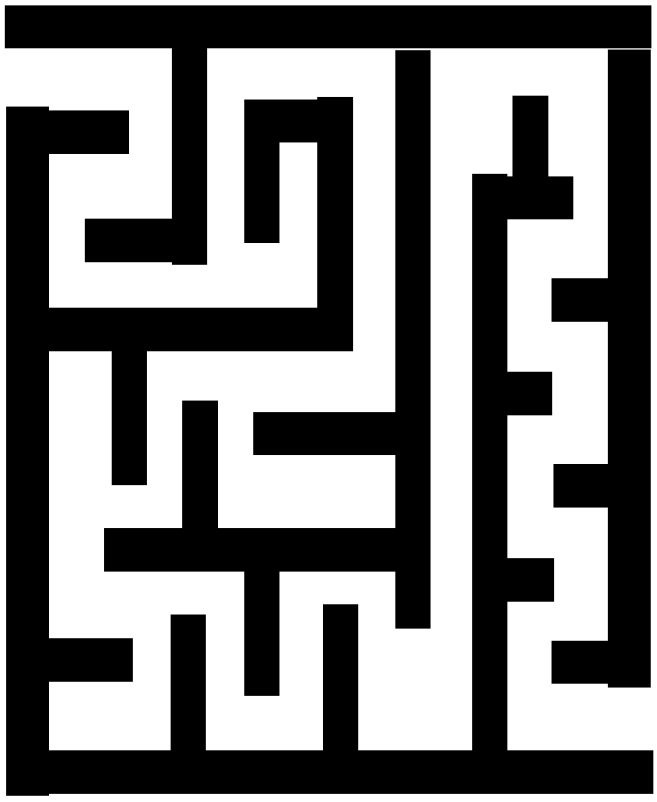
1. Representasi data labirin
2. penerapan algoritma A* dan algoritma Djikstra pada labirin
3. kompleksitas algoritma A* dan Djikstra

3.2 Representasi Data Labirin

Struktur data dari labirin akan menggunakan struktur data yang digunakan oleh citra PGM. Hal ini disebabkan karena repast memiliki fungsi untuk memproses dan memanipulasi citra dalam struktur PGM. Contoh dari struktur data labirin yang akan dibuat dapat dilihat pada gambar dibawah. Pada matriks data, angka "0" merupakan dinding labirin, angka 5 merpakan jalur pada labirin, angka 3 merupakan jalan masuk dan jalan keluar

Keterangan :

Nama	: Labirin
Panjang	: 17
Lebar	: 15
Kode Masuk/keluar	: 3
Kode Jalan	: 3
Kode Tembok	: 0



Gambar 3.2.1 Visualisasi data labirin

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	5	5	5	0	5	5	5	5	5	0	5	5	5	5	5	0
0	0	0	5	0	5	0	0	0	5	0	5	5	0	5	5	0
0	5	5	5	0	5	0	5	0	5	0	5	0	0	0	5	0
0	5	0	0	0	5	0	5	0	5	0	5	0	5	5	5	0
0	5	5	5	5	5	5	5	0	5	0	5	0	5	0	0	0
0	0	0	0	0	0	0	0	0	5	0	5	0	5	5	5	0
0	5	0	5	5	5	5	5	5	5	0	5	0	0	0	5	0
0	5	0	5	0	5	0	0	0	0	0	5	0	5	5	5	0
0	5	5	5	0	5	5	5	5	5	0	5	0	5	0	0	0
0	5	0	0	0	0	0	0	0	0	0	5	0	5	5	5	0
0	5	5	5	5	5	0	5	5	5	0	5	0	0	0	5	0
0	0	0	5	0	5	0	5	0	5	5	5	0	5	5	5	0
0	5	5	5	0	5	5	5	0	5	5	5	0	5	5	5	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

3.3 Penerapan Algoritma A* & Djikstra Pada Labirin

3.3.1 A* Pada Labirin

A* akan mencoba semua jalur/node pada labirin dan akan memilih nilai yang paling optimal / rute tercepat yaitu yang mempunyai nilai paling rendah (*lowest f score*). Dan jika terjadi ***close set*** maka A* akan merunut balik (backtrack) pada node sebelumnya sampai menemukan jalan percabangan / node yang memberi pilihan untuk jalan hingga jalan keluar.

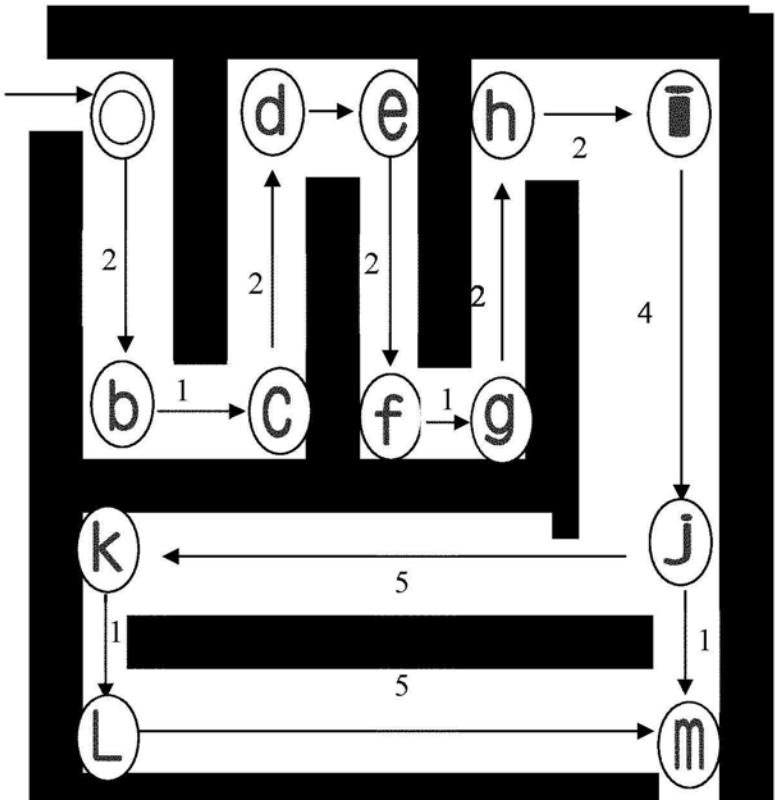
Pemilihan node yang akan diekspansi g(n), yaitu nilai yang digunakan untuk mencapai sebuah node, dan h(n), yaitu nilai yang diperlukan untuk mencapai tujuan dari node tersebut.

$$\mathbf{F(n) = g(n) + h(n)}$$

Cara ini sangat baik digunakan karena jika nilai **h(n)** memenuhi syarat tertentu maka cara ini akan lengkap dan optimal. A* disebut optimal jika **h(n)** bersifat *admissible heuristic*, yaitu tidak pernah memiliki nilai yang terlalu melampaui perkiraan untuk mencapai tujuan.

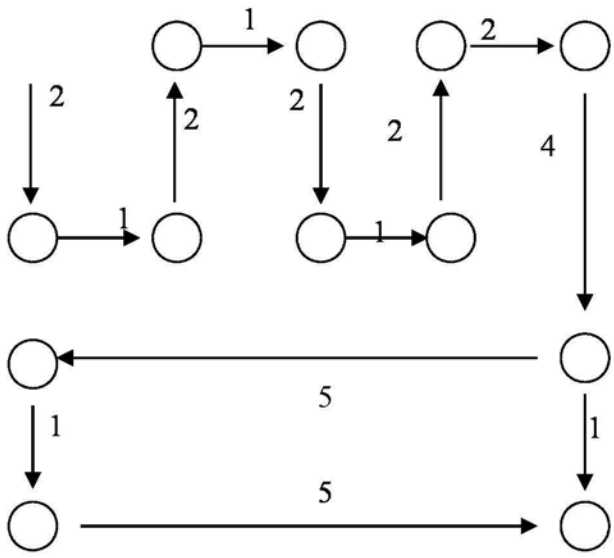
Nilai **h(n)** bisa didapatkan melaui beberapa cara, antara lain dengan menarik garis lurus antara node saat ini dengan node akhir dan menghitung jaraknya atau dengan menghitung jumlah besar minimum yang perlu dilalui untuk mencapai node akhir dari node saat ini.

Pada ilustrasi di bawah ini, sebuah node berada pada tiap persimpangan, atau di setiap sudut labirin.



Gambar 3.2.1 penerapan node pada labirin

Setelah di rubah menjadi bentuk vertex dan edge menjadi seperti di bawah ini



Gambar 3.2.2 Ilustrasi node node pada labirin

Keterangan jumlah jarak:

h(b) = 2, h(c) = 3, h(d) = 5, h(e) = 6,

h(f) = 8, h(g) = 9, h(h) = 11, h(i) = 13, h(j) = 17,

h(k) = 22, h(l) = 23, h(m) = 28,

f(n) = g(n) + h(n)

Maka di peroleh hasil :

Melewati jalur b – c – d – e – f – g – h – i – j – m = 33

Jika, melewati jalur b – c – d – e – f – g – h – i – j – k – l – m = 43

Hasil Algoritma A*	
Jalur yang dilewati	b – c – d – e – f – g – h – i – j – m

Tabel 3.3.1 Hasil Algoritma A*

ketika menggunakan algoritma A*, dan pada persimpangan (J). A* akan menghitung jarak ke arah jalan keluar, yaitu melewati node (k) atau (m). Jika ke arah node (K) maka memerlukan jarak (5). Maka secara otomatis A* akan memilih langsung ke Node (m) karena lebih optimal. Karena menggunakan algoritma A* maka hasil yang dipilih haya hasil yang paling optimal, hasil yang kurang optimal tidak masuk dalam tabel hasil.

3.3.2 Djikstra Pada Labirin

Dengan menggunakan ilustrasi yang sama dengan diatas "**Ilustrasi node node pada labirin"** ketika berada pada persimpangan (j), agen djikstra akan terpecah menjadi dua bagian, yaitu ke arah (M) dan kearah (K), walaupun sudah menemukan jalan yang lebih optimal Djikstra tetap menghitung hingga semua node yang ada dilewati dan di hitung, Pada algoritma Djikstra dipastikan dapat menemukan jalan yang optimal, akan tetapi semua kemungkinan di coba. Hal ini yang membuat Djikstra memakan banyak memori dan berjalan lambat,
maka di peroleh tabel hasil :

Jalur yang bisa dilalui	
Jalur 1	b – c – d – e – f – g – h – i – j – m
Jalur 2	b – c – d – e – f – g – h – i – j – k – l – m

Tabel 3.3.2 Hasil Algoritma Djikstra

3.4 **Kompleksitas Algoritma**

Kompleksitas algoritma A* tergantung pada heuristic. Pada kasus yang rumit, misalnya labirin mempunyai banyak percabangan perhitungan solusi akan menjadi fungsi exponensial. Dan akan menjadi fungsi polinomial ketika labirin hanya mempunyai satu jalan keluar

$$|h(x) - h^*(x)| = O(\log h^*(x))$$

Seperti rumus diatas, dimana H adalah solusi optimal *x* adalah hasil yang tepat mencapai tujuan/ garis finish. Dengan kata lain jika H error atau tidak mendapatkan hasil hasilnya tidak akan melebihi H*, dan akan mengembalikannya ke jalan keluar

Sedangkan pada algorima Djikstra, jarak awal di tentukan dari angka 0, jika ada node yang kosong, S akan mengunjungi node terdekat dari dari S. Pada algoritma Djiksra kompleksitasnya berlaku : **O(M log N)**

Dimana M adalah jarak dan N adalah nodenya. UM[x] adalah total jarak antara node pertama dan jalan keluar, S adalah node yang dipilih sebelumnya

3.5 **Pengujian dan Analisis**

Pengujian dilakukan agar dapat menunjukkan bahwa aplikasi mampu bekerja sesuai dengan perancangan yang dilakukan dengan tingkat kesalahan yang kecil. Pengujian yang akan dilakukan melalui beberapa macam labirin yang akan di persiapkan sebelumnya.

3.6 **Pengambilan Kesimpulan dan Saran**

Pada tahap ini, diambil kesimpulan dari hasil pengujian dan analisis terhadap aplikasi pencarian rute. Tahap selanjutnya adalah pembuatan saran yang dimaksudkan untuk memperbaiki kesalahan kesalahan yang terjadi dan menyempurnakan penulisan serta untuk memberikan pertimbangan atas pengembangan aplikasi selanjutnya

BAB IV
PERANCANGAN DAN IMPLEMENTASI

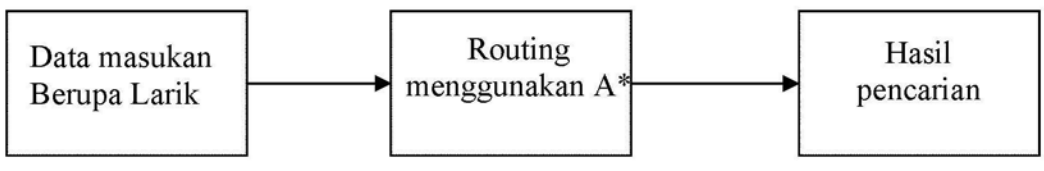
Bab ini menjelaskan mengenai langkah-langkah yang akan dilakukan untuk merancang dan mengimplementasikan aplikasi pencarian rute terdekat pada labirin. Perancangan dan implementasi dikerjakan dengan beberapa tahap. Meliputi identifikasi Larik menjadi bentuk gambar labirin, penghitungan dan percarian rute pada labirin. perancangan diawali dengan penggambaran blok diagram kerja sistem yang menunjukkan cara kerja aplikasi secara umum

4.1 Perancangan Secara Umum

Perancangan aplikasi secara umum merupakan tahap awal dalam melakukanperancangan alpikasi perncarian rute yang akan dibuat. Perancangan diawali dengan penggambaran blok diagram kerja sistem yang menunjukkan cara kerja aplikasi umum.

4.1.1 Blok Diagram Sistem

Pembuatan diagram sistem merupakan dasar dari perancangan sistem agar perancangan dan perealisasian aplikasi berjalan secara sistematis. Data masukan adalah LARIK yang sebelumnya sudah kita buat. Blok diagram dapat ditunjukkan seperti gambar :



Gambar 4.1.1 Blok Diagram

Fungsi dari masing-masing bagian pada diagram blok ini akan dijelaskan sebagai berikut :

1. Data masukan merupakan Larik yang kita siapkan sebelumnya yang akan di gunakan sebagai input sistem.
2. komputer digunakan sebagai sarana untuk proses pengolahan data, mulai dari data awal dalam bentuk Larik, identifikasi Larik menjadi bentuk labirin dan pencarian rute pada labirin
3. dari beberapa proses tahap sebelumnya maka akan didapatkan informasi yang menampilkan hasil yang sesuai dengan tujuan perancangan dan pembuatan aplikasi ini

4.2 Cara Kerja Aplikasi

Cara kerja dari aplikasi pencarian rute terdekat pada labirin ini dimulai dengan membuat data awal berbentuk Larik yang kita persiapkan dahulu sesuai yang kita inginkan. Kemudian data ini menjadi masukan utama untuk aplikasi pencarian rute terdekat.

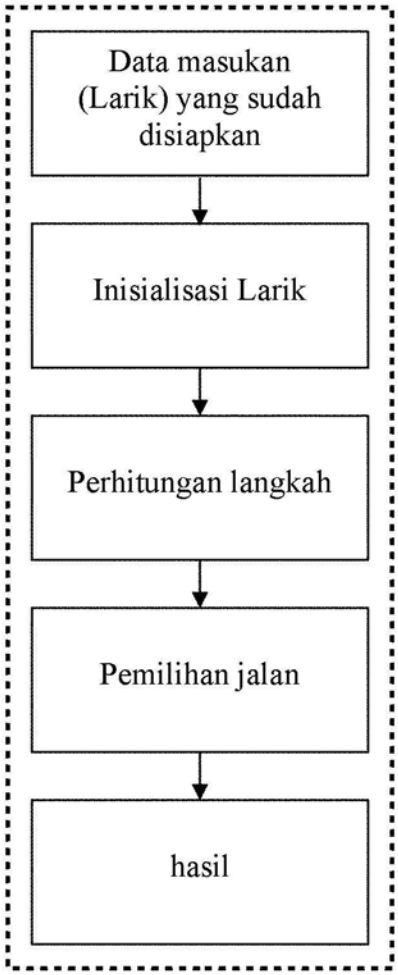
Setelah memasukkan data utama, aplikasi siap memprosesnya menjadi labirin untuk selanjutnya dihitung dan dicari jalan tercepat menuju finish.

4.2.1 Perancangan Perangkat Lunak

Pada bagian perancangan ini perangkat lunak yang akan dibuat menggunakan bahasa pemrogaman Visual Basic.NET 2010 dan sistem yang digunakan untuk membangun perangkat lunak ini dirancang dengan spesifikasi mampu melakukan hal-hal sebagai berikut:

1. merubah data masukkan yang berupa Larik menjadi bentuk labirin
2. menghitung dan mencari jalan tercepat pada labirin

Untuk desain aplikasi secara umum akan ditunjukkan pada gambar 4.2.1

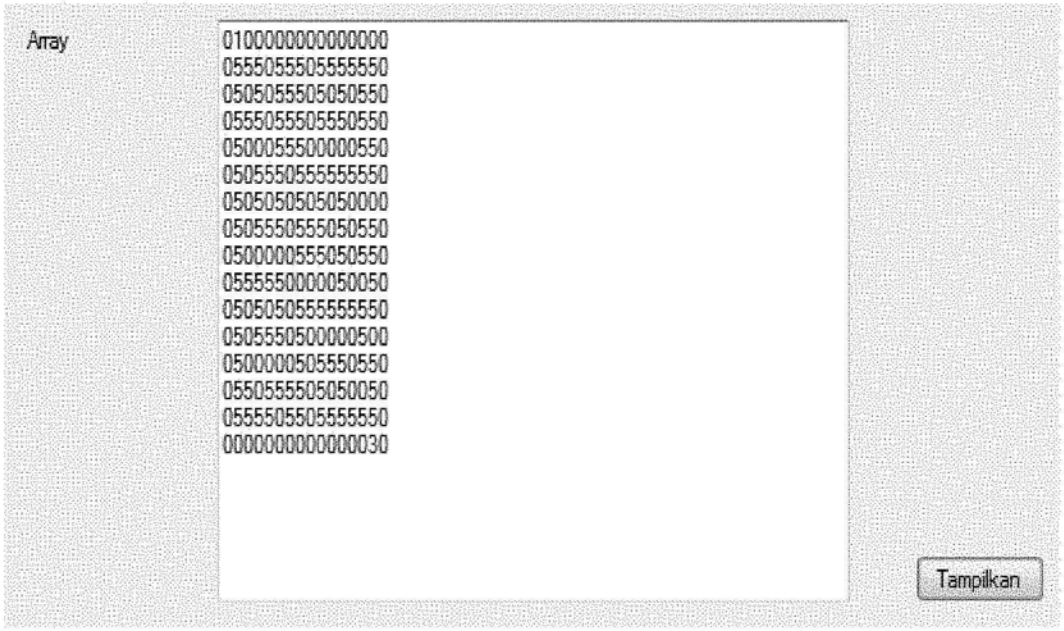


Gambar 4.2.1 Desain Aplikasi

4.2.2 Inisialisasi Larik

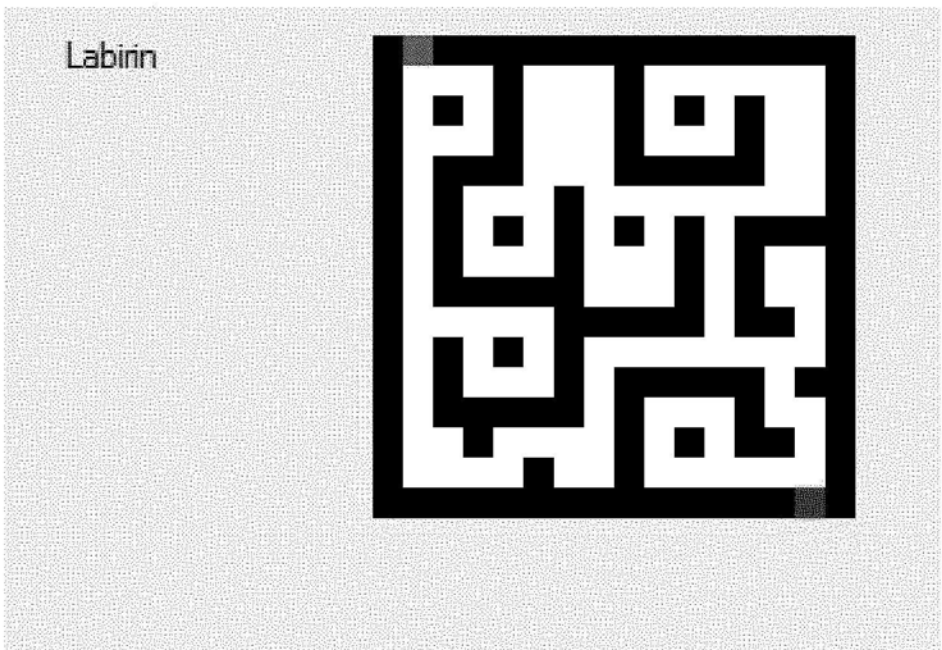
Pada tahap ini Larik yang kita buat sebelumnya, menjadi masukan utama yang akan di inisialisasi pada kotak Larik. Proses inisialisasi Larik ini sendiri bertujuan untuk merubah data awal menjadi labirin secara utuh dan siap untuk

dicari dan dihitung langkahnya yang bertujuan untuk menemukan jalan keluar/finish, berikut gambar proses pemasukan data dan inialisasinya.



Gambar 4.2.2.1 Data kotak Larik

Textbox Larik menjadi tempat masukan utama pada alpikasi ini, sebelum di inialisasi data masih berbentuk Larik dan belum menjadi sebuah labirin utuh.



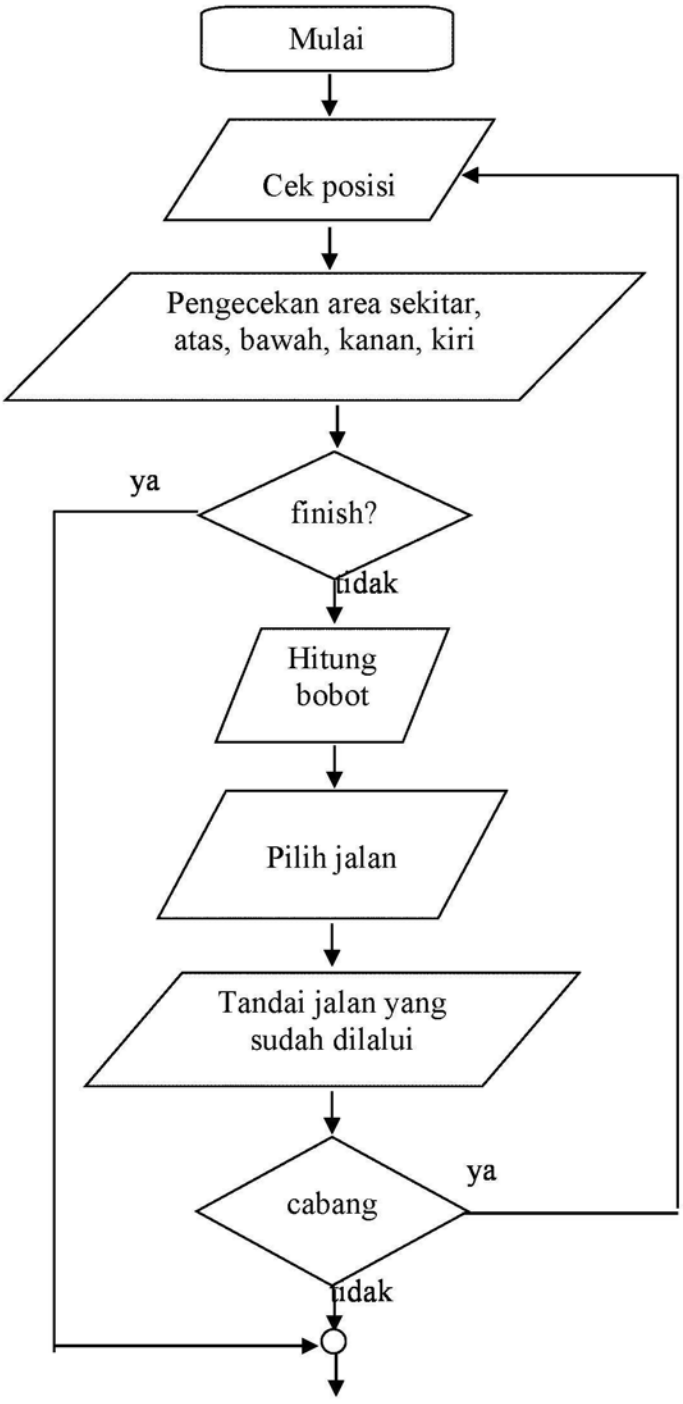
Gambar 4.2.2.2 hasil inisialisasi

Setelah proses inisialisasi data yang sebelumnya berbentuk Larik berubah menjadi sebuah labirin utuh, yang bisa kita lihat dan bedakan dengan mudah antara tembok, jalan masuk, dan jalan keluar. Labirin yang sudah di inisialisasi ini kemudian siap diproses untuk dicari jalan keluar/finish dan dihitung langkahnya

4.2.3 Pemilihan Langkah

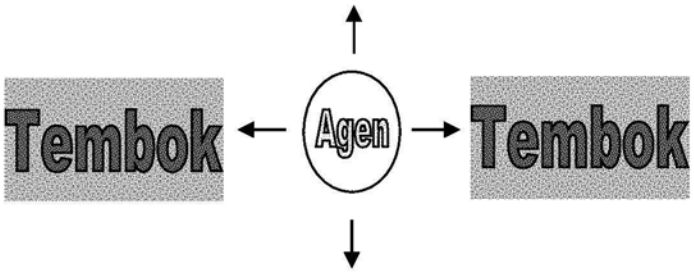
Pada tahap pemilihan langkah ini, dilakukan setelah data Larik di inisialisasi dan menjadi labirin secara utuh. Proses ini dilakukan dengan cara membandingkan area sekitar agen ter lebih dahulu, yaitu : area atas, bawah, kanan dan kiri labirin. agen akan menginisialisasi area sekitar dan membandingkan antara tembok, jalan taukah jalan buntu.

Proses ini diawali pada angka "1" yaitu kode yang mewakili jalan masuk pada labirin, pemilihan langkah pertama dilakukan dengan cara menseleksi/membandingkan area sekitar agen, area yang di bandingkan yaitu tembok,jalan yang telah dilalui/buntu.


Mulai
Cek posisi
Pengecekan area sekitar,
atas, bawah, kanan, kiri
ya
finish?
tidak
Hitung
bobot
Pilih jalan
Tandai jalan yang
sudah dilalui
ya
cabang
tidak

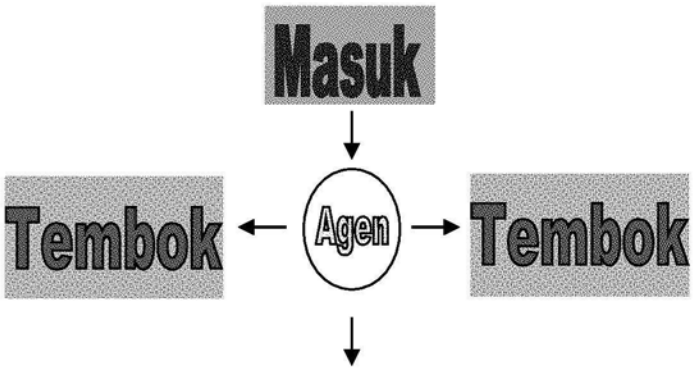
selesai

Flowchart 4.2.3.1 pemilihan langkah



Gambar 4.2.3.1 Pilih jalan

Pada Gambar 4.2.3.1 dijelaskan, agen harus bisa membandingkan area sekitar, terdiri dari tembok dan posisi awal sehingga agen tidak menggantikan/menabrak posisi tembok atau mengulangi langkah kebelakang. Proses membandingkan area sekitar ini adalah kemampuan dasar yang harus dimiliki agen, dan proses ini akan di ulang selama agen belum menemukan jalan keluar/finish, dan akan berhenti ketika menemukan garis finish



Gambar 4.2.3.2 Agen Bergerak Maju

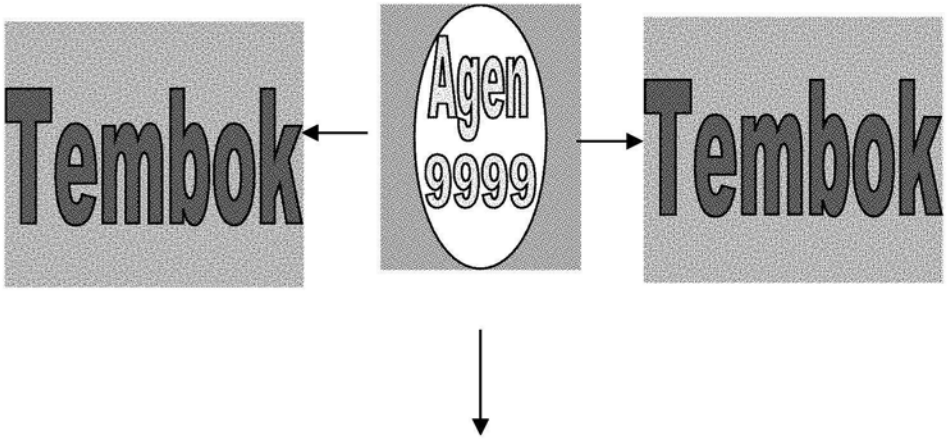
Pada ilustrasi gambar 4.2.3.2 agen bergerak menjauhi posisi awal (jalan masuk). Setelah agen mampu membandingkan area sekitar, agen harus bergerak maju/menjauhi jalan masuk dan menandainya, hal ini bertujuan agar agen tidak melakukan langkah kebelakang (untuk jalan yang tidak buntu) dan menhindari terjadinya looping/ mengulangi langkah yang sama secara berulang.

Pada aplikasi pensarian rute terdekat ini, jalan yang sudah dilalui di inisialisasi dengan nama "bekas". Hal ini akan terus dilakukan agen jika tak ada jalan buntu. Semua jalan akan di tandai hal ini dilakukan agar agen tetap bergerak maju/tidak mundur dan tidak melakukan gerakan yang sama untuk berulang kali.

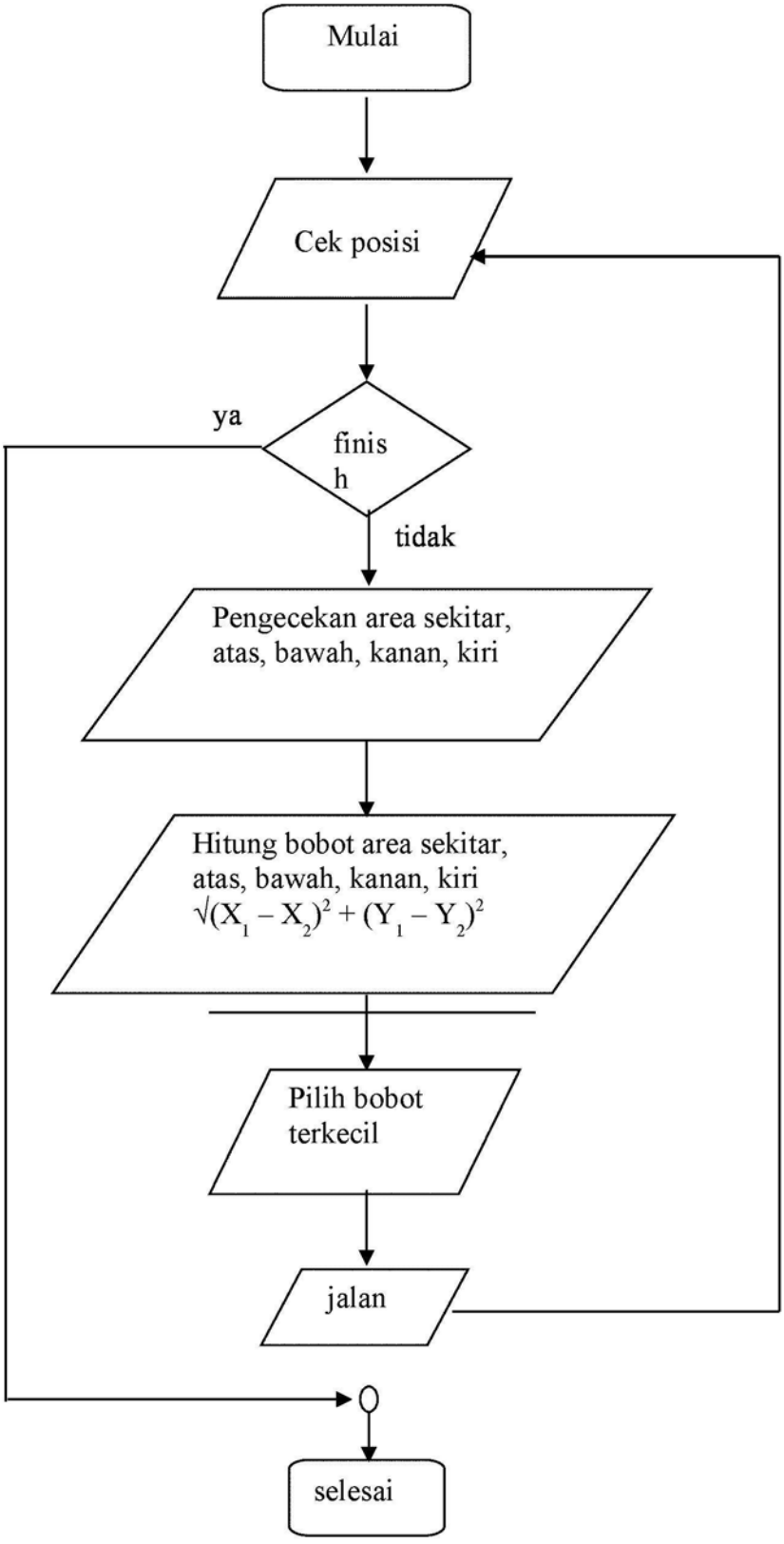
4.2.4 Perhitungan Langkah

Perhitungan akan langkah dilakukan ketika agen berada pada posisi awal (jalan masuk) hal ini bertujuan agar agen sudah bisa memilih langkah pada awal posisi. Pada aplikasi pencari rute terdekat ini menggunakan pytagoras untuk menghitung jarak.

Perhitungan jarak dilakukan dengan menghitung jarak atar koordinat saat ini (x,y) dan finish (x,y). Perhitungan terus dilakukan ketika agen bergerak hingga agen menemukan jalan keluar/finish



Gambar 4.2.4.1 bobot agen

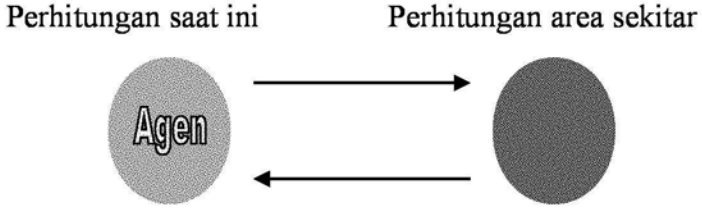


Flowchart 4.2.4.1 perhitungan langkah

Pada posisi awal (jalan masuk), agen mempunyai bobot awal "9999" hal ini bertujuan agar agen bisa bergerak menjauhi posisi awal/jalan masuk.

Agen berjalan ketika perhitungan jarak area sekitar (kanan, kiri, atas, bawah) dengan garis finis telah dilakukan, hasil selanjutnya dibandingkan dengan bobot awal yaitu "9999"

Angka "9999" dipakai dengan asumsi bahwa pada posisi awal tidak ada bobot perhitungan yang di atas "9999", dengan begitu agen akan selalu dapat bergerak menjauh dari posisi awal/jalan masuk.

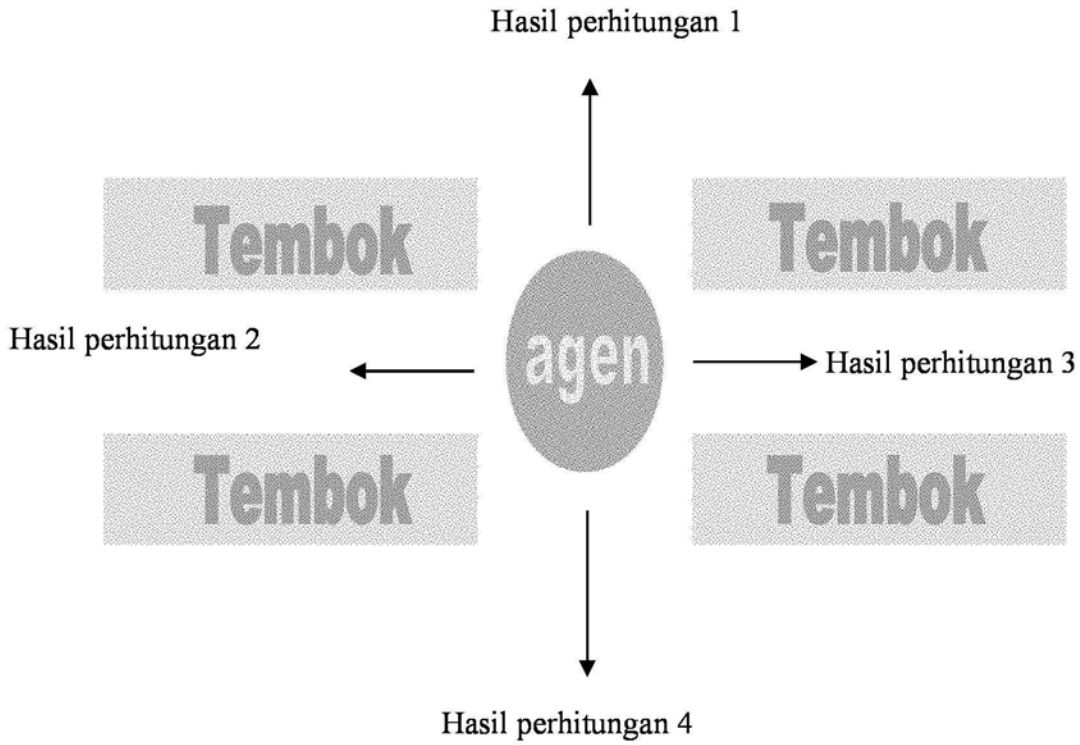


Gambar 4.2.4.2 perbandingan bobot

Setelah agen mampu bergerak dari posisi awal dan mempunyai bobot "baru" yang lebih kecil dari "9999" maka agen akan bergerak maju dan mempunyai koordinat baru/posisi baru. Perhitungan jarak area sekitar dengan letak jalan keluar/finish akan tetap dilakukan dengan tujuan untuk memperoleh nilai baru.

Pada gambar 4.2.4.2 digambarkan, bobot yang diperoleh agen pada koordinat saat ini akan di bandingkan dengan hasil perhitungan area sekitar yang memberi kemungkinan bagi agen untuk bergerak, perbandingan ini akan di

lakukan pada area sekitar agen (kanan, atas, kiri, bawah) hingga diperoleh bobot yang lebih kecil dari bobot sebelumya.

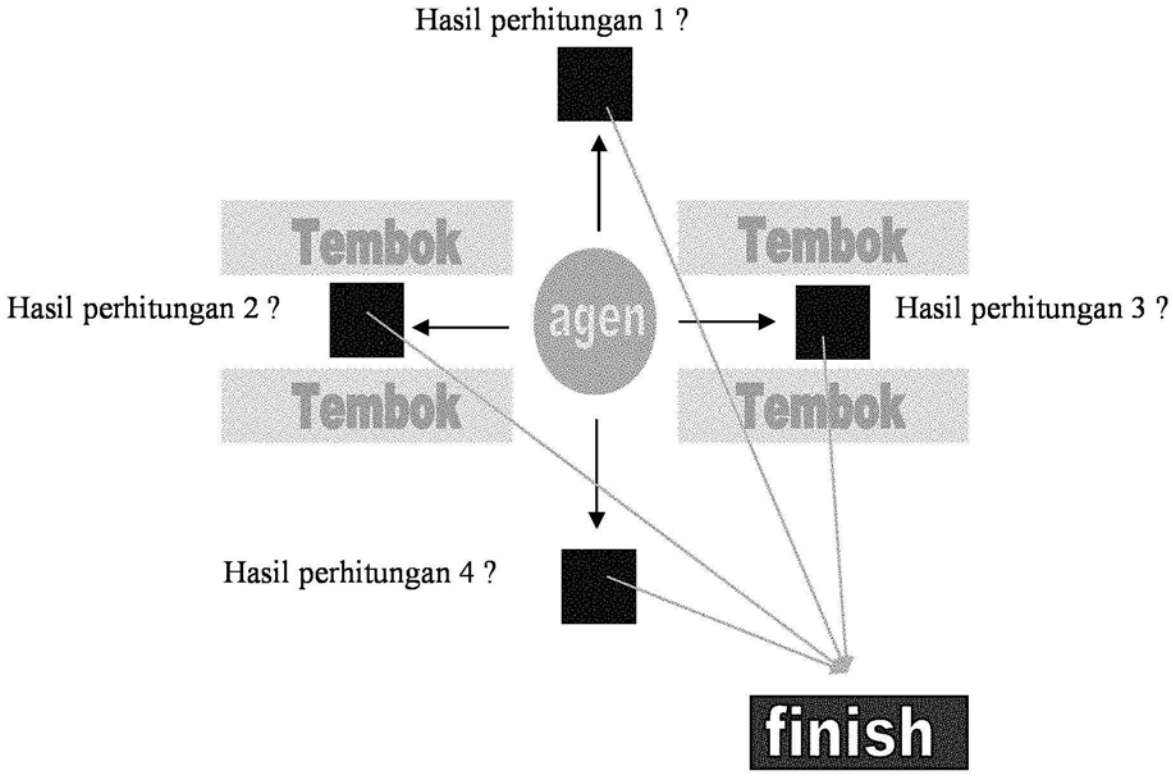


Gambar 4.2.4.3 Pemilihan langkah berdasar bobot

Pada gambar 4.2.4.3 agen berada pada posisi tengah, dimana area sekitar (atas, bawah, kanan dan kiri) memberikan pilihan untuk bergerak. Pada posisi ini agen akan menghitung bobot dari masing masing jalan, dan dipilih bobot yang paling kecil dari ke empat pilihan tersebut.

Pada gambar 4.2.4.3 di jelaskan setiap pilihan dihitung menggunakan rumus pytagoras antara kordinat saat ini, dengan posisi jalan keluar/finish yang

sudah diketahui oleh agen, setelah mendapatkan hasil agen akan bergerak ke posisi bobot yang lebih kecil



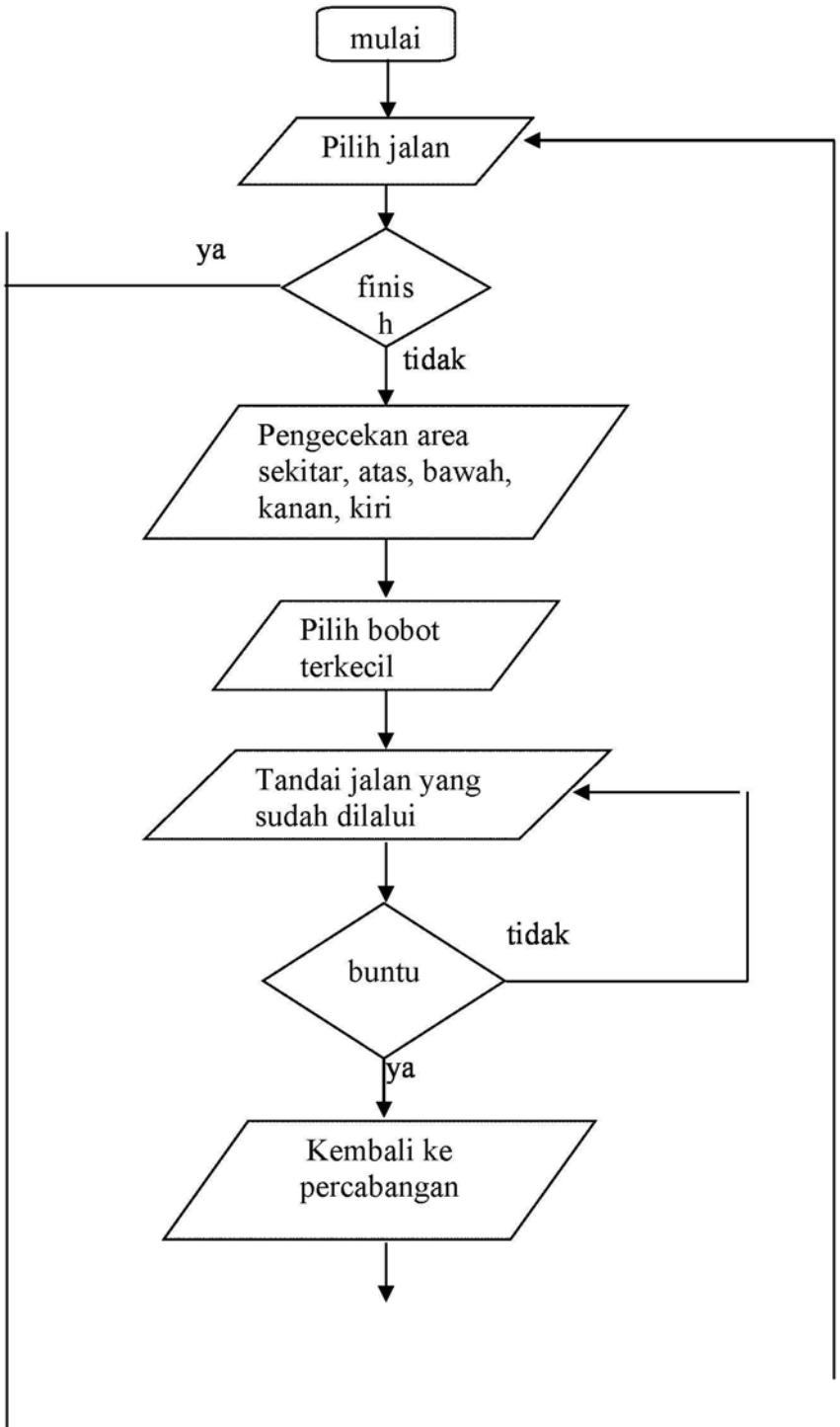
Gambar 4.2.4.4 perhitungan langkah

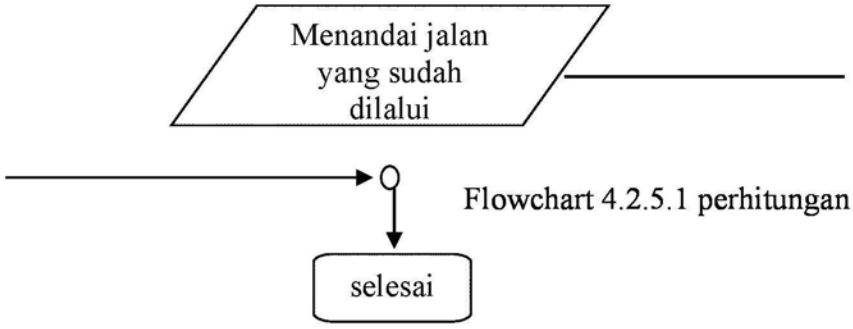
Dengan menggunakan rumus $\sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2}$ maka akan diperoleh perhitungan saat ini, yaitu hasil dari perhitungan area atas, bawah, kanan, kiri yang dihitung jaraknya dengan garis finish. Hasil dari perhitungan ini selanjutnya akan di bandingkan dan akan di pilih bobot yang paling ringan

Pada aplikasi ini A* berperan pada proses pemilihan langkah dan perhitungan, dimana hasil pemilihan langkah dan perhitungan dilakukan secara real time. Berbeda dengan djikstra yang perhitungan semua area dilakukan di awal berjalannya aplikasi sehingga terkesan lambat dan berat.

4.2.5 **Back Track**

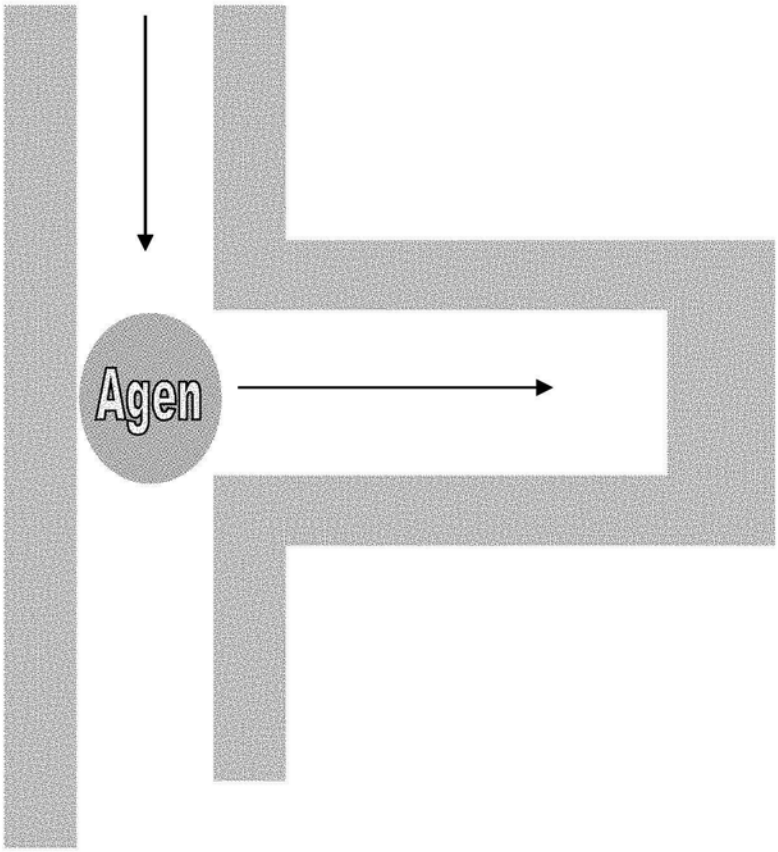
Proses backtrack ini hanya digunakan untuk labirin khusus, yang mempunyai jalan buntu atau rumit sehingga agen bisa bergerak mundur dan memilih jalan yang memberikan pilihan untuk jalan.





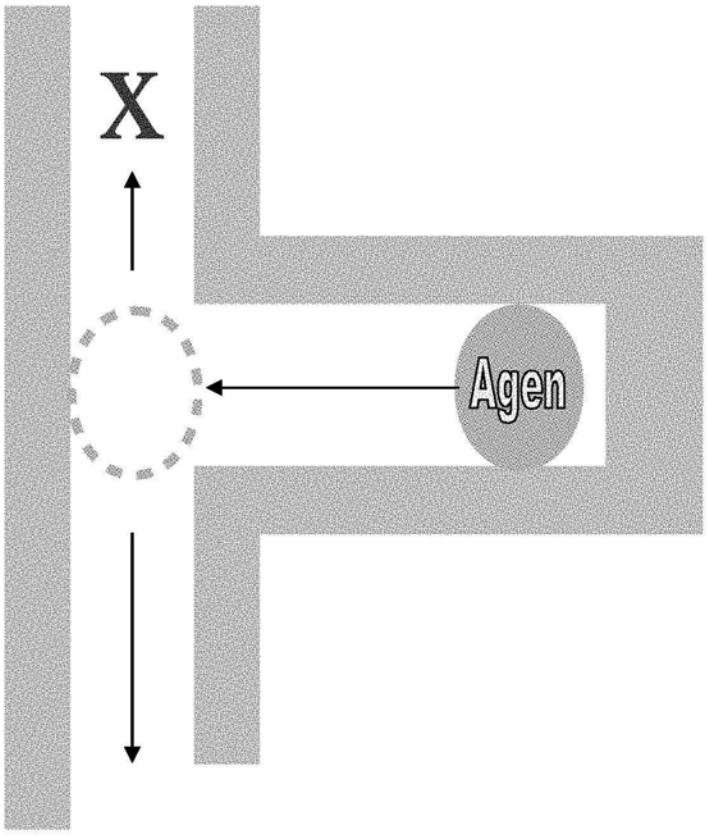
Flowchart 4.2.5.1 perhitungan langkah

pada proses sebelumnya, "pelmilihan langkah" jalan yang sudah dilalui deberi tanda "bekas" agar agen tak bergerak ke posisi sebelumnya secara berulang, begitu juga dengan proses backtrack, agen harus bisa menandai setiap persimpangan yang ada untuk melakukan proses backtrack



Gambar 4.2.5.1 Agen melalui jalan buntu

Pada gambar 4.2.5.1 agen berada pada persimpangan yang memberi dua pilihan untuk jalan, ke arah kanan merupakan jalan buntu. Pada ilustrasi diatas agen akan memilih jalan buntu untuk dilalui, tapi sebelum melalu persimpangan agen harus mampu menandai persimpangan untuk melakukan proses backtrack



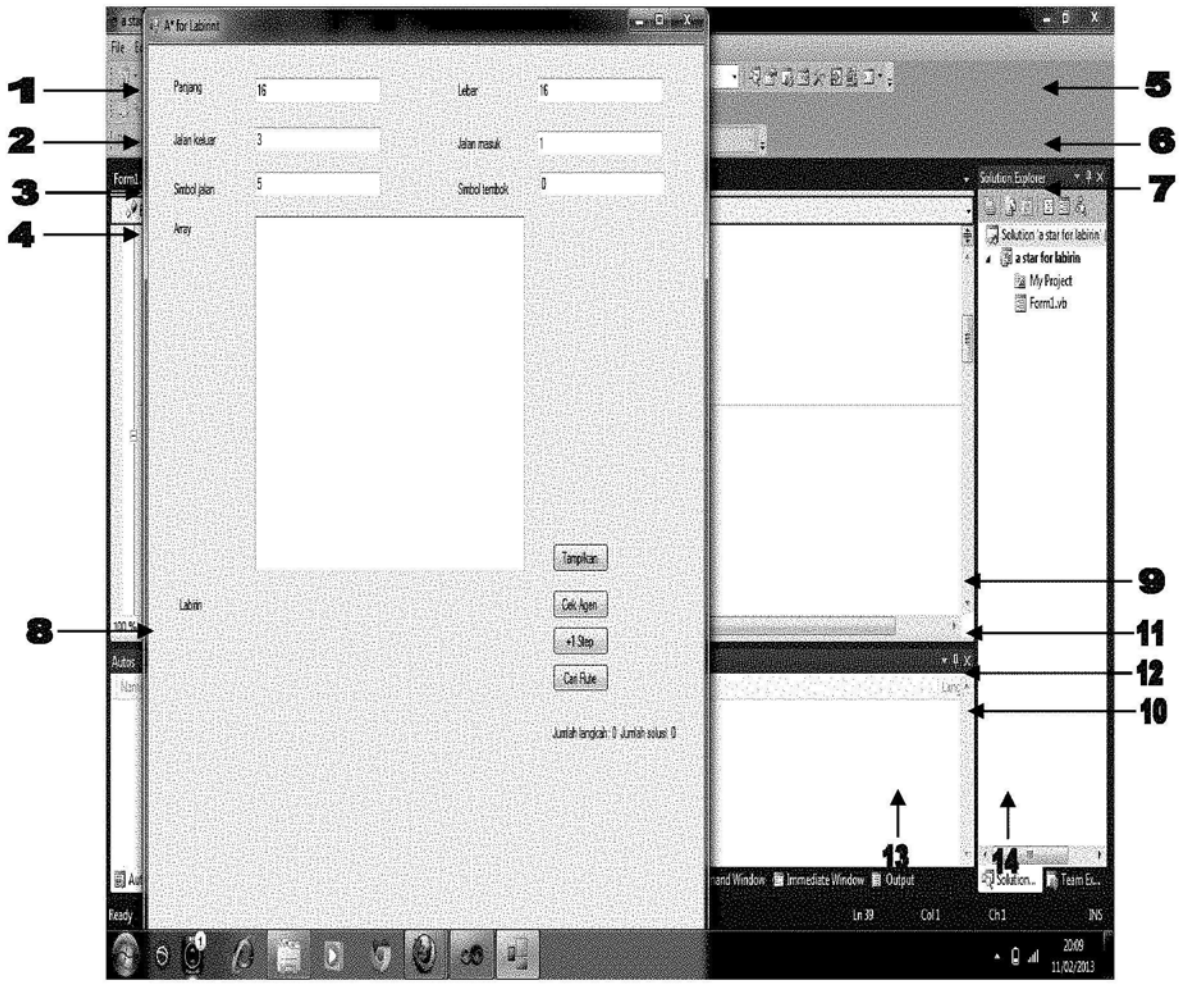
Gambar 4.2.5.2 kembali ke persimpangan

Setelah agen mengetahui bahwa jalan yang telah dilalui adalah jalan buntu, maka agen akan menghapus jalan yang sudah dilalui (bekas) untuk kembali pada jalan yang memberikan agen pilihan untuk melanjutkan perjalanan menemukan jalan keluar.

Agen akan tetap mencari jalan dengan cara mundur per kotak(per koordinat) dan mencari kemungkinan untuk keluar dari jalan buntu hingga di temukan persimpangan/jalan yang memberi pilihan untuk melanjutkan perjalanan hingga finish

Ketika pada persimpangan, agen akan membandingkan area sekitar. Jika jalan sudah pernah dilalui (bekas) maka agen akan memilih jalur baru yang belum pernah dilewati. Hal ini akan terus dilakukan agen ketika melewati jalan buntu.

4.6 Desain Interface



Gambar 4.6.1 Desain Interface

Design aplikasi ini terdiri dari 1 jendela, yang berisi beberapa menu untuk mendukung kerja dari aplikasi ini, diantaranya :

1. Panjang : yang berisi informasi tentang panjang labirin, toolbox ini juga berfungsi sebagai batasan panjang labirin.
2. Jalan Keluar : kode angka untuk jalan keluar, sebagai pembeda dan tanda yang mudah dikenali pada aplikasi ini jalan keluar berwana merah
3. Simbol Jalan : simbol jalan pada aplikasi ini diwakili dengan angka "5" dan bisa disesuaikan dengan kemauan kita. Simbol jalan ini sendiri adalah jalan yang akan dilalui agen untuk mencapai garis finish
4. Larik : Larik merupakan masukan utama pada aplikasi ini. Pada kolom textbox inilah akan dimasukkan serangkaian Larik untuk selanjutnya diproses lebih lanjut
5. Lebar : yang berisi informasi tentang lebar labirin, toolbox ini juga berfungsi sebagai batasan lebar labirin.
6. Jalan Masuk : kode angkak untuk jalan masuk paad aplikasi ini adalah "1" atau bisa kita rubah sesuai dengan keinginan kita, diwakili dengan warna hijau sebagai jalan masuk, kode ini adalah awal mula agen sebelum bergerak
7. Simbol Tembok : diwakili dengan angka "0", kode ini berfungsi sebagai batasan/halangan yang ada pada labirin.
8. Labirin : pada picturebox labirin inilah Larik dari masukan utama akan di tampilkan dalam bentuk labirin utuh dan siap dicari jalan keluar/finish dan dihitung.
9. Tampilkan : button "tampilkan ini berguna untuk proses inisialisasi. Dari data masukan utama berupa Larik dirubah kedalam bentuk labirin utuh pada picturebox labirin
10. Cari Rute : pada button ini, adalah proses utama dari aplikasi ini. Terdapat proses memilih langkah, menghitung langkah hingga menemukan jalan keluar/finish pada labirin secara langsung
11. Cek Agen : button ini berguna untuk mengecek posisi dari agen, berisi kordinst ysng menginformasikan keberadaan agen

12. +1 atep : button ini berfungsi hampir sama dengan button cari rute, perbedaannya pada button ini agen bergerak satu demi satu kotak, hingga terlihat proses pencarian rutenya
13. Jumlah Langkah : label box ini meng informasikan pada kita tentang jumlah langkah yang di tempuh agen secara keseluruhan, termasuk perhitungan langkah backtrack
14. Jumlah Solusi : labelbox ini memberi informasi pada kita tentang jumlah langkah yang di perlukan agen untuk mencapai jalan keluar. Dan hanya dihitung langkah solusinya saja

BAB V
PENGUJIAN

Rancangan yang telah diimplementasikan dalam bentuk nyata memerlukan suatu pengujian. Uji coba ini adalah cara untuk mengetahui hasil dari percobaan yang dilakukan sekaligus sebagi sarana pemunculan ide ide bagi proses pengembangan selanjutnya. Pengujian yang dilakukan dalam bab ini adalah sebagai berikut :

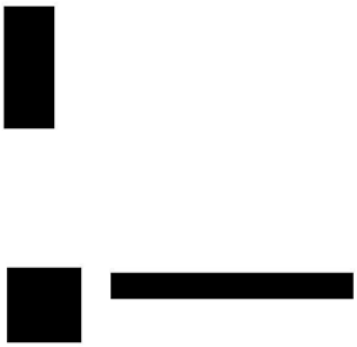
1. labirin tanpa cabang
2. labirin bercabang
3. labirin buntu bersolusi
4. labirin buntu
5. labirin 20 x 20 bercabang, buntu bersolusi

Pada pengujian ini, labirin dibuat dengan bantuan MAZESMITH, yang diperoleh dari laman dengan alamat di mazesmith.sourceforge.net/

5.1 Pengujian Labirin Tanpa Cabang

Lab[illegible] ng hanya mempunyai satu jalan atau labirin ya[illegible] tidak memberikan pilihan jalan untuk dipilih. Labirin ini relatif leb[illegible] muda dari pada labirin be[illegible]g

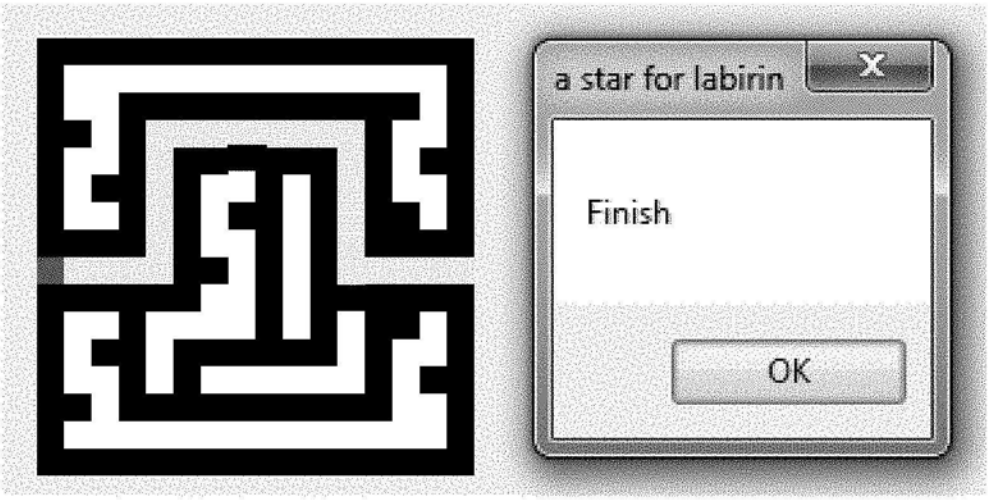
pengujian :



Gambar 5.1.1 Pengujian labirin tanpa cabang

Pada pengujian peratma yang dilakukan di labirin tidak bercabang ini, agen berhasil menemukan jalan keluar/finish dengan 29 langkah.

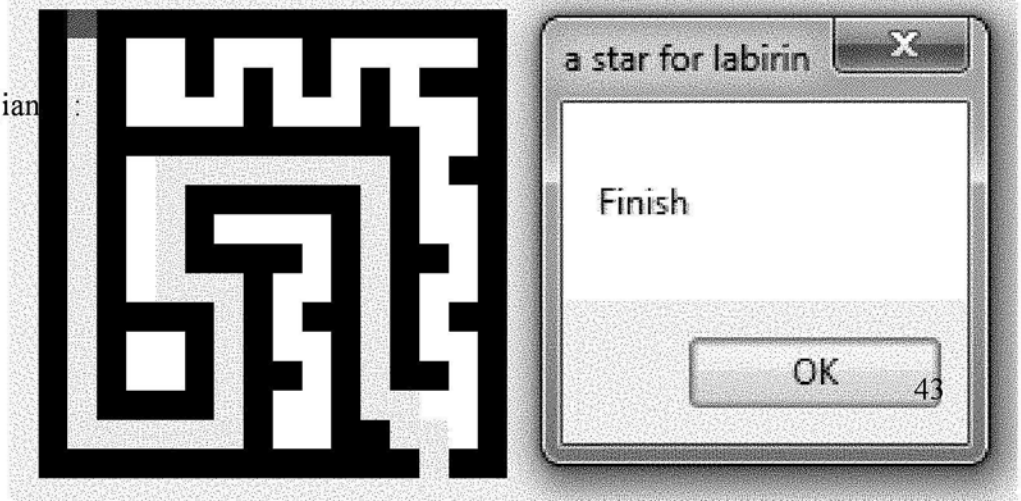
Pengujian 2 :



Gambar 5.1.2 Pengujian labirin tanpa cabang

Pada pengujian kedua agen berhasil mencapai jalan keluar dengan 25 langkah.

Pengujian :



Gambar 5.1.3 Pengujian labirin tanpa cabang

Di pengujian ketiga agen memerlukan 49 langkah menuju jalan keluar

Pengujian 4:

Gambar 5.1.4 Pengujian labirin tanpa cabang

Pada pengujian keempat agen memerlukan 42 langkah menuju jalan keluar.

5.2 Labirin Bercabang

sehi

pilih

tidal

Peng

Gambar 5.2.1 Pengujian labirin bercabang

Pada pengujian pertama labirin bercabang ini agen memerlukan 56 langkah untuk mencapai jalan keluar

Pengujian 2 :

Gambar 5.2.2 Pengujian labirin bercabang

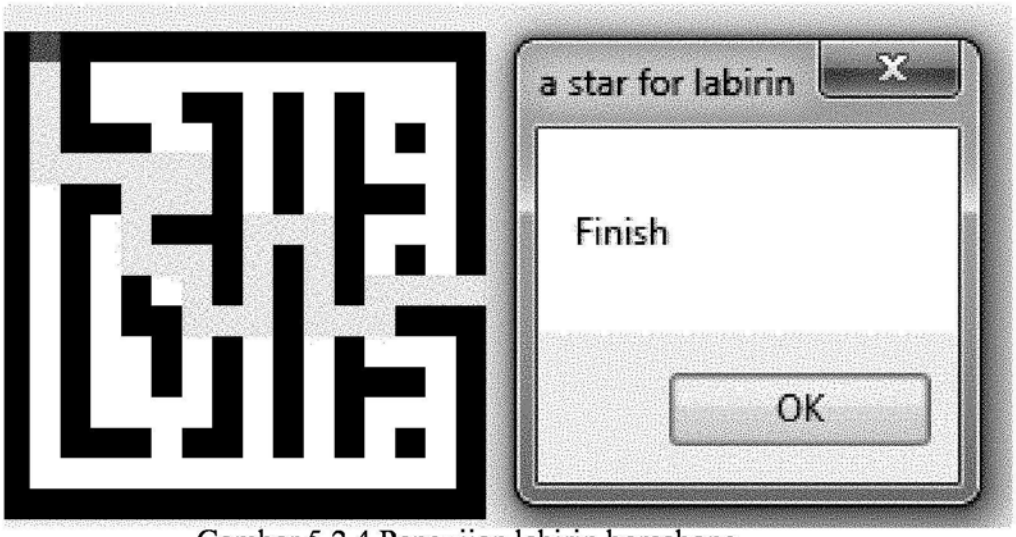
Pada pengujian kedua agen memerlukan 21 langkah untuk mencapai jalan keluar

Pengujian 3

Gambar 5.2.3 Pengujian labirin bercabang

Agen memerlukan 29 langkah untuk mencapai jalan keluar

Pengujian 4 :

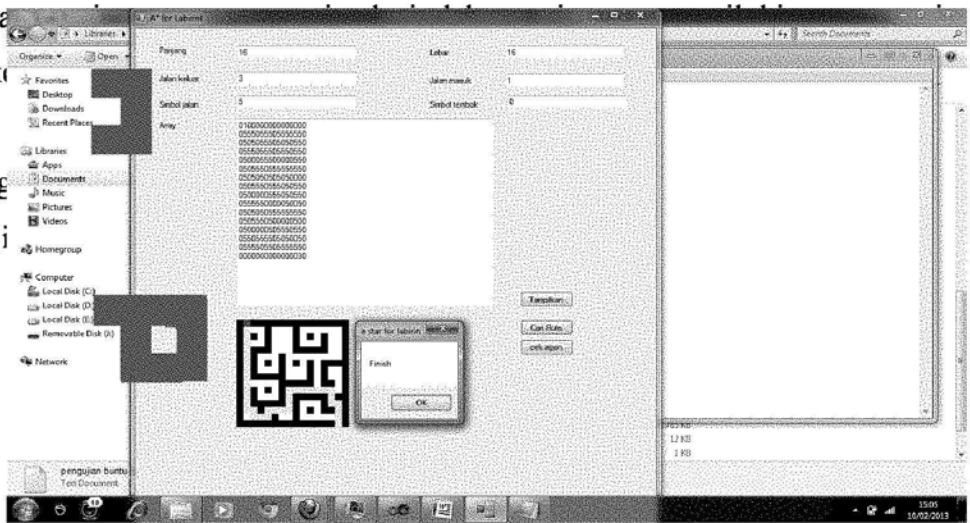


Gambar 5.2.4 Pengujian labirin bercabang

Pada pengujian keempat ini agen memerlukan 39 langkah untuk mencapai jalan keluar.

5.3 Labirin Buntu Bersolusi

Labirin buntu bersolusi adalah labirin yang mempunyai jalan buntu, bercaba
jalan k
labirin
sehingg
Penguji



Gambar 5.3.1 Pengujian labirin buntu bersolusi

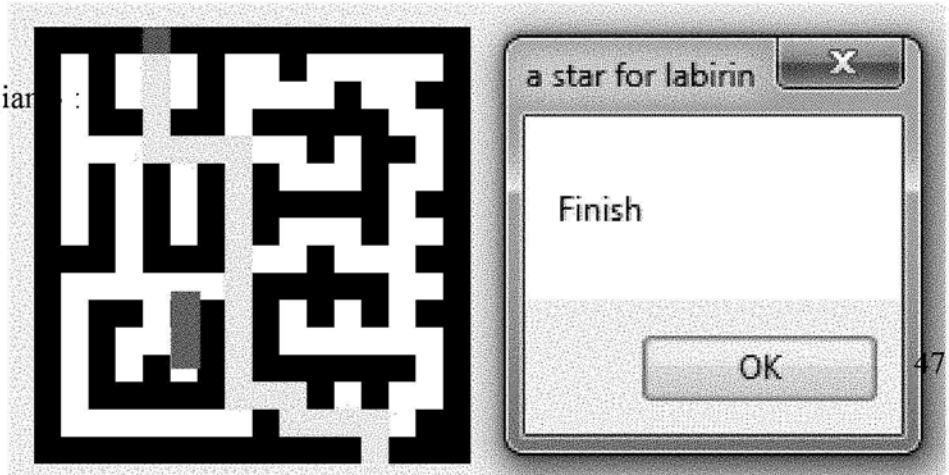
Pada pengujian pertama pada labirin buntu bersolusi ini agen memerlukan 36 langkah untuk menemukan jalan keluar, pada gambar labirin diatas agen melakukan proses backtrack, proses ini di tandai dengan jalan berwarna biru pada gambar

Pengujian 2 :

Gambar 5.3.2 Pengujian labirin buntu bersolusi

Pada pengujian kedua agen memerlukan 24 langkah untuk menemukan jalan keluar

Pengujian :



Gambar 5.3.3 Pengujian labirin buntu bersolusi

Pada percobaan ketiga agen melakukan proses backtrack yang di tandai dengan warna biru, dan memerlukan 23 langkah untuk mencapai jalan keluar

Pengujian 4 :

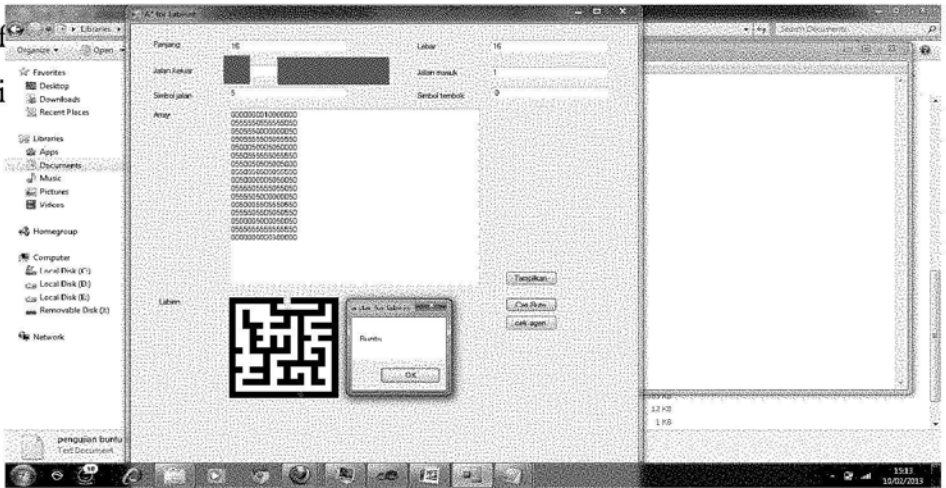
Gambar 5.3.4 Pengujian labirin buntu bersolusi

Pada pengujian ke 4 agen memerlukan 24 langkah, dan melakukan backtrack (pada jalan yang ditandai warna biru) untuk menemukan jalan keluar.

5.4 Labirin Buntu

Labirin buntu adalah labirin yang mempunyau jalan buntu, percabangan tetapi tidak mempunyai jalan keluar/ agen tidak bisa menemukan jalan keluar/f

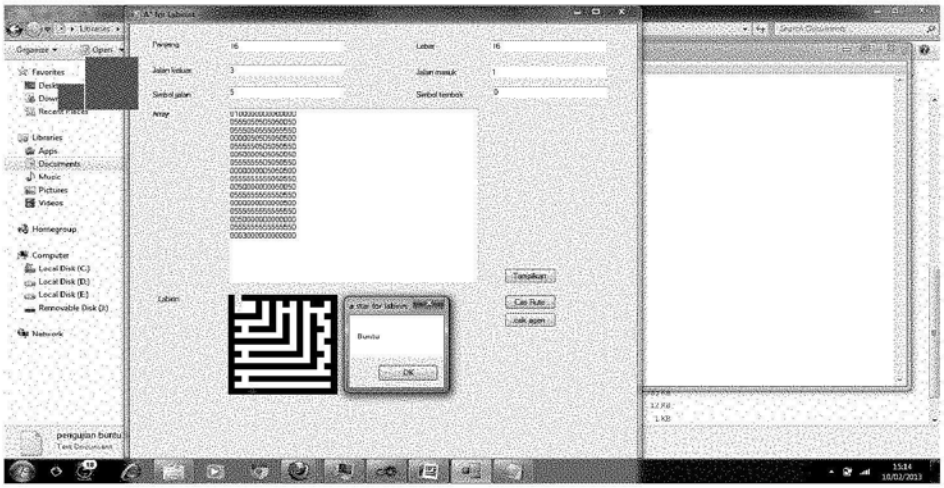
Penguji



Gambar 5.4.1 Pengujian labirin buntu

Pada pengujian pertama ini agen melakukan backtrack, dan memerlukan 12 langkah untuk kembali ke posisi awal, karena tak mendapatkan solusi.

Pengujian ke 2 :



Gambar 5.4.2 Pengujian labirin buntu

Pada pengujian kedua ini agen mencoba untuk mencari solusi, dan melakukan 36 langkah untuk backtrack ke posisi awal karena tak menemukan solusi

5.5 **labirin 20 x 20 bercabang, Buntu Bersolusi**

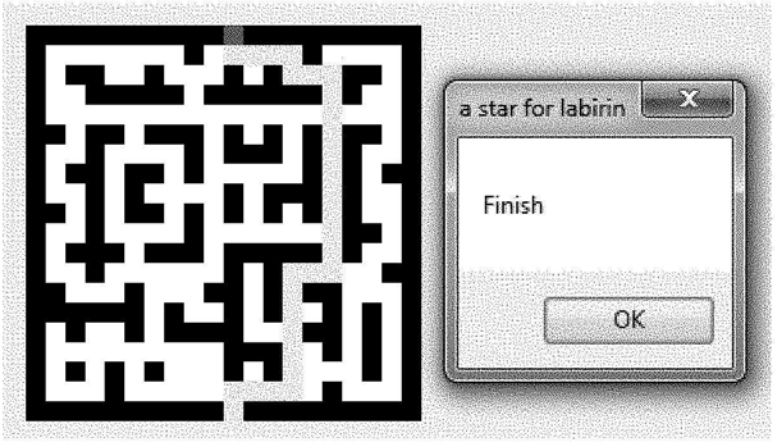
Labirin 20 x 20 adalah labirin yang mempunyai panjang 20 kotak dan lebar 20 kotak. Labirin ini mempunyai percabangan dan jalan buntu sehingga lebih rumit dari pengujian pengujian sebelumnya

pengujian 1 :

Gambar 5.5.1 Pengujian labirin 20x20 bercabang dan buntu bersolusi

agen melakukan 62 langkah untuk menemukan jalan keluar

Pengujian 2 :



Gambar 5.5.2 Pengujian labirin 20x20 bercabang dan buntu bersolusi

Pada pengujian kedua ini agen memerlukan 31 langkah untuk menemukan jalan keluar.

5.6 Tabel Hasil Pengujian

5.6.1 Labirin Tidak Becabang

Pengujian ke	Langkah Solusi Untuk Labirin			
	Tidak Bercabang 1	Tidak Bercabang 2	Tidak Bercabang 3	Tidak Bercabang 4
1	29	25	49	42
2	29	25	49	42
3	29	25	49	42

Tabel 5.6.1Labirin tidak bercabang

5.6.2 Labirin Bercabang

Pengujian ke	Langkah Solusi Untuk Labirin			
	Bercabang 1	Bercabang 2	Bercabang 3	Bercabang 4
1	56	21	29	39
2	56	21	29	39
3	56	21	29	39

Tabel 5.6.2.1 Labirin tidak bercabang

5.6.3 Labirin Buntu Bersolusi

Pengujian ke	Langkah Solusi Untuk Labirin			
	Buntu Bersolusi 1	Buntu Bersolusi 2	Buntu Bersolusi 3	Buntu Bersolusi 4
1	36	23	29	24
2	36	23	29	24
3	36	23	29	24

Tabel 5.6.3.1 Labirin Buntu bersolusi

5.6.4 Labirin Buntu

Pengujian ke	Langkah Solusi Untuk Labirin	
	Buntu 1	Buntu 2
1	2	2
2	2	2
3	2	2

Tabel 5.6.4.1 Labirin Buntu

5.6.5 Labirin 20x20 Bercabang dan Buntu Bersolusi

Pengujian ke	Langkah Solusi Untuk Labirin	
	Buntu 1	Buntu 2
1	62	29
2	62	29
3	62	29

Tabel 5.6.5.1 Labirin 20x20

BAB VI
PENUTUP

6.1 Kesimpulan

Berdarsarkn hasil perancangan, implementasi dan pengujian yang dilakukan, didapatkan kesimpulan :

A* tidak menjamin selalu mendapatkan rute terbaik (bobot terkecil) dari semua rute yang ada, hal ini terjadi karena A* hanya menghitung area yang dilalui saja. Area yang tidak dilalui di abaikan

VI.2 Saran

Saran yang dapat diberikan untuk pengembangan pencarian rute terdekat pada labirin menggunakan metode A* antara lain :

1. untuk pengembangan lebih lanjut pencarian rute terdekat ini, menggunakan labirin yang lebih besar dan lebih rumit
2. ditambahkan kriteria atau jenis jenis labirin yang lebih kompleks

DAFTAR PUSTAKA

Kusumadewi, Ida dan Hari Purnomo. 2005. *Penyelesaian Masalah Optimisasi dengan Teknik-teknik Heuristik.* Yogyakarta : Graha Ilmu

Pratama, Rian Putra. 2011. *Perbandingan Algoritma A* dan Dijkstra Berbasis Web GIS untuk Pencarian Rute Terpendek.* Skripsi. Bandung: Universitas Pendidikan Indonesia

Rudy Adipranata, et al. 2007. Aplikasi Pencari Rute Optimum pada Peta Guna Meningkatkan Efisiensi Waktu Tempuh Pengguna Jalan dengan Metode A* dan Best First Search

Aini, Dewi Yusra. 2011. Analisis Algoritma A Star (A*) dan Implementasinya dalam Pencarian Jalur Terpendek pada Jalur Lintas Sumatera di Provinsi Sumatera Utara. Skripsi.

Felzenszwalb, Pedro F and D. McAllester. 2007. *The Generalized A* Architecture*. Journal Of Artificial Intelegence Research : hal. 153-190.

Harlianto Tanudjaja, et al.2008.*Sistem Perencanaan Jalur Dengan Metode A* (A Star) pada Perangkat Bantu Tuna Netra*

Catatanteknisi.com, pengertian routing, table routing dan protocol routing. www.catatanteknisi.com/2011/05/pengertian-routing-tabel-routing.html#.USCRJ_LVeG8

Wikipedia. 2012. Complexity. wikipedia.org/wiki/A*_search_algorithm#Complexity

Algorithmist. 2012. Dijkstra's_algorithm. algorithmist.com/index.php /Dijkstra%27 s_
algorithm

A*_search_algorithm. 2012. http://en.wikipedia.org/wiki/A*_search_algorithm

Thesolidsnake. 2012. http://thesolidsnake.wordpress.com/2012/01/23/pencarian-jarak-terpendek-dengan-algoritma-a/

Alihasyim. 2012. pencarian-rute-terpendek-menggunakan. http://alihasyim.blogspot.com/2012/10/pencarian-rute-terpendek-menggunakan.html

LAMPIRAN

```
Public Class Form1

    Dim Labirin(1, 1) As String
    Dim Bekas(1, 1) As String
    Dim xRute(10000) As String
    Dim yRute(10000) As String
    Dim nLangkah As Integer
    Dim Bobot(10000) As Double
    Dim xMasuk As Integer
    Dim yMasuk As Integer
    Dim xKeluar As Integer
    Dim yKeluar As Integer
    Dim buntu As Integer = 0
    Dim finish As Integer = 0
    Dim xagen As Integer
    Dim yagen As Integer
    Dim xatas As Integer
    Dim yatas As Integer
    Dim xkanan As Integer
    Dim ykanan As Integer
    Dim xkiri As Integer
    Dim ykiri As Integer
    Dim xbawah As Integer
    Dim ybawah As Integer
    Dim jumlahLangkah As Integer

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
        'inisialisasi panjang dan lebar array
        Dim xPanjang As Integer = (TextBoxPanjang.Text)
        Dim xLebar As Integer = (TextBoxLebar.Text)
        ReDim Labirin(xPanjang, xLebar)
        ReDim Bekas(xPanjang, xLebar)
        ReDim Bobot(10000)
        nLangkah = 0
        jumlahLangkah = 0
        LabelStatus.Text = "Jumlah langkah: " & jumlahLangkah & "  Jumlah solusi: " & nLangkah

        'mengisi array'
        For i As Integer = 0 To xLebar - 1 'pengisian lebar array, dikurangi 1 karena mulai dari 0'
            For j As Integer = 0 To xPanjang - 1 'pengisian panjang array, dikurangi 1 karena dimulai dari 0'
                Labirin(j, i) = TextBoxArray.Text((i * (xPanjang + 2)) + j)
            Next
        Next

        'menggambar labirin
        Dim xgraphic As Graphics = PictureBoxLabirin.CreateGraphics 'inisialisasi picture box untuk menggambar
labirin'

        For i As Integer = 0 To xPanjang - 1 'untuk pengisian lebar labirin'
            For j As Integer = 0 To xLebar - 1 'untuk pengisian panjang labirin'
                If Labirin(i, j) = TextBoxJalan.Text Then 'simbol jalan, inputan dari text box panjang berwarna putih'
                    xgraphic.FillRectangle(Brushes.White, i * 10, j * 10, 10, 10)
                ElseIf Labirin(i, j) = TextBoxJalanMasuk.Text Then 'pengisian jalan masuk, dari text box jalan masuk
berwarna hijau'
                    xgraphic.FillRectangle(Brushes.Green, i * 10, j * 10, 10, 10)
                    xMasuk = i
                    yMasuk = j
```

```
            ElseIf Labirin(i, j) = TextBoxJalanKeluar.Text Then 'simbol jalan keluar, inputan dari text box panjang
berwarna merah'
                xgraphic.FillRectangle(Brushes.Red, i * 10, j * 10, 10, 10)
                xKeluar = i
                yKeluar = j
            Else
                xgraphic.FillRectangle(Brushes.Black, i * 10, j * 10, 10, 10) ' jika buukan jalan masuk, jalan keluar dan
jalan labirin, maka digambar/warna hitam/tembok'
            End If
        Next
    Next
    xgraphic.Dispose()
    xagen = xMasuk
    yagen = yMasuk
    buntu = 0
    finish = 0
End Sub

Private Sub ButtonStep_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
ButtonStep.Click
    If finish = 1 Then
        MsgBox("Finish")
    ElseIf buntu = 1 Then
        MsgBox("Buntu")
    Else
        gerak()
    End If

    LabelStatus.Text = "Jumlah langkah: " & jumlahLangkah & "  Jumlah solusi: " & nLangkah
End Sub

Sub gerak()
    Dim jaraktemp As Double = 9999
    nLangkah = nLangkah + 1
    jumlahLangkah = jumlahLangkah + 1

    xatas = xagen
    yatas = yagen - 1
    xkanan = xagen + 1
    ykanan = yagen
    xbawah = xagen
    ybawah = yagen + 1
    xkiri = xagen - 1
    ykiri = yagen


    Dim xgraphic As Graphics = PictureBoxLabirin.CreateGraphics

    If xatas >= 0 And xatas <= CInt(TextBoxLebar.Text) - 1 And yatas >= 0 And yatas <=
CInt(TextBoxPanjang.Text) - 1 Then 'batasan agen untuk bergerak ke atas'
        If (Labirin(xatas, yatas) <> TextBoxTembok.Text) And (Bekas(xatas, yatas) <> 1) Then 'batasan agen bukan
tembok, maka agen bisa berjalan'
            If (HitungJarak(xatas, yatas) > Bobot(nLangkah) Or HitungJarak(xatas, yatas) = 0) And HitungJarak(xatas,
yatas) < jaraktemp Then 'perbandingan nilai sekarang, dengan jarak temp'
                xagen = xatas
                yagen = yatas
```

```
                jaraktemp = HitungJarak(xatas, yatas) 'jika jarak sekarang < jarak temp, maka jarak lebih kecil yang di
pakai'

            End If
        End If
    Else
    End If

    If xkanan >= 0 And xkanan <= CInt(TextBoxLebar.Text) - 1 And ykanan >= 0 And ykanan <=
CInt(TextBoxPanjang.Text) - 1 Then 'batasan agen untuk bergerak ke kanan'
        If (Labirin(xkanan, ykanan) <> TextBoxTembok.Text) And (Bekas(xkanan, ykanan) <> 1) Then 'batasan
agen bukan tembok, maka agen bisa berjalan'
            'MsgBox(HitungJarak(xkanan, ykanan) & " " & Bobot(nLangkah))
            If (HitungJarak(xkanan, ykanan) > Bobot(nLangkah) Or HitungJarak(xkanan, ykanan) = 0) And
HitungJarak(xkanan, ykanan) < jaraktemp Then 'perbandingan nilai sekarang, dengan jarak temp'
                xagen = xkanan
                yagen = ykanan
                jaraktemp = HitungJarak(xkanan, ykanan) 'jika jarak sekarang < (jarak sebelumnya) maka agen bergerak
ke bobot yang lebih kecil'
            End If
        End If
    End If

    If xbawah >= 0 And xbawah <= CInt(TextBoxLebar.Text) - 1 And ybawah >= 0 And ybawah <=
CInt(TextBoxPanjang.Text) - 1 Then 'batasan agen untuk bergerak ke bawah'
        If (Labirin(xbawah, ybawah) <> TextBoxTembok.Text) And (Bekas(xbawah, ybawah) <> 1) Then 'batasan
agen bukan tembok, maka agen bisa berjalan'
            If (HitungJarak(xbawah, ybawah) > Bobot(nLangkah) Or HitungJarak(xbawah, ybawah) = 0) And
HitungJarak(xbawah, ybawah) < jaraktemp Then 'perbandingan nilai sekarang, dengan jarak temp'
                xagen = xbawah
                yagen = ybawah
                jaraktemp = HitungJarak(xbawah, ybawah) 'jika jarak sekarang < (jarak sebelumnya) maka agen
bergerak ke bobot yang lebih kecil'

            End If
        End If
    End If

    If xkiri >= 0 And xkiri <= CInt(TextBoxLebar.Text) - 1 And ykiri >= 0 And ykiri <=
CInt(TextBoxPanjang.Text) - 1 Then 'batasan agen untuk bergerak ke bawah'
        If (Labirin(xkiri, ykiri) <> TextBoxTembok.Text) And (Bekas(xkiri, ykiri) <> 1) Then 'batasan agen bukan
tembok, maka agen bisa berjalan'
            If (HitungJarak(xkiri, ykiri) > Bobot(nLangkah) Or HitungJarak(xkiri, ykiri) = 0) And HitungJarak(xkiri,
ykiri) < jaraktemp Then 'perbandingan nilai sekarang, dengan jarak temp'
                xagen = xkiri
                yagen = ykiri
                jaraktemp = HitungJarak(xkiri, ykiri) 'jika jarak sekarang < (jarak sebelumnya) maka agen bergerak ke
bobot yang lebih kecil'

            End If
        End If
    End If

    If jaraktemp < 9999 Then
        xgraphic.FillRectangle(Brushes.Yellow, xagen * 10, yagen * 10, 10, 10) 'jika setelah perhitungan sudah
dilakukan, maka agen akan bergerak kearah yang lebih kecil, dan berwarna kuning'
        Bekas(xagen, yagen) = "1"
```

```
            xRute(nLangkah) = xagen
            yRute(nLangkah) = yagen
            Bobot(nLangkah) = jaraktemp
        Else
            xgraphic.FillRectangle(Brushes.White, xagen * 10, yagen * 10, 10, 10) 'backbackback'
            Bekas(xagen, yagen) = "0"
            Bobot(nLangkah) = 0
            nLangkah = nLangkah - 2
            xagen = xRute(nLangkah)
            yagen = yRute(nLangkah)
        End If
        xgraphic.Dispose()

        If xagen = xKeluar And yagen = yKeluar Then
            finish = 1
            MsgBox("Finish")

        End If

        If xagen = xMasuk And yagen = yMasuk Then
            buntu = 1
            MsgBox("Buntu")
        End If
    End Sub



    Function HitungJarak(ByVal x As Integer, ByVal y As Integer)
        Dim jarak As Double
        jarak = Math.Pow((xKeluar - x), 2) + Math.Pow((yKeluar - y), 2) 'rumus perhitungan jarak dengan rumus
pitagoras'
        jarak = Math.Pow(jarak, 0.5)
        Return jarak
    End Function

    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button3.Click
        MsgBox(xagen & ", " & yagen)

    End Sub

    Private Sub ButtonCariRute_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
ButtonCariRute.Click

        While buntu = 0 And finish = 0
            gerak()
        End While
        LabelStatus.Text = "Jumlah langkah: " & jumlahLangkah & "  Jumlah solusi: " & nLangkah
    End Sub
End Class
```