Penerapan Algoritma A* pada Aplikasi *Puzzle*

Latius Hermawan[1)], R. Kristoforus Jawa Bendi[2)]

Jurusan Teknik Informatika, Sekolah Tinggi Teknik Musi[1,2)]
Jl.Bangau No.60 Palembang

E-mail: fanytiuz@gmail.com[1)], kristojb@gmail.com[2)]

Abstrak

Perkembangan teknologi yang sangat cepat harus didukung dengan perkembangan dan implementasi algoritma yang baik, sehingga teknologi tersebut dapat memberikan hasil yang baik, efektif dan efisien. Puzzle yang digunakan adalah berukuran 3x3 yang berisi angka. Pemain harus menyusun kembali ubin yang teracak menjadi terurut. Untuk mengatasi masalah yang sedang terjadi maka dibuat Penerapan Algoritma A pada Aplikasi Puzzle yang menggunakan artificial intelligence yaitu mempelajari tentang bagaimana cara agar membuat komputer dapat melakukan pekerjan seperti dan sebaik yang dilakukan manusia. Algoritma A* yang termasuk dalam heuristik ini dan dibantu dengan perhitungan jarak Manhattan, akan membantu dalam pencarian pergeseran angka yang memungkinkan. Model Proses yang digunakan adalah Model Air Terjun (Waterfall) dengan pengujian white-box testing, black-box testing dan uji sampel berpasangan. Uji sampel berpasangan digunakan untuk menguji suatu hipotesis yang dibuat apakah dapat diterima atau tidak. Pengujian yang didapatkan dengan adanya penerapan algoritma A* pada aplikasi puzzle maka total pergeseran dan waktu yang digunakan dengan menggunakan algoritma A* akan meminimalkan total pergeseran dan waktu yang dihabiskan dibandingkan dengan bermain tanpa menggunakan algoritma A* yang dapat membantu pemain menyelesaikan permainan dengan mudah.*

Kata Kunci: *Algoritma A**, *Puzzle*

1. Pendahuluan

Perkembangan teknologi yang sangat cepat harus didukung dengan perkembangan dan implementasi algoritma yang baik, sehingga teknologi tersebut dapat memberikan hasil yang baik, efektif dan efisien. Pada saat ini, banyak algoritma yang berkembang untuk mencapai solusi dari sebuah permasalahan [1].

Permainan terdiri dari atas sekumpulan peraturan yang membangun situasi bersaing dari dua sampai beberapa orang atau kelompok dengan memilih strategi yang dibangun untuk memaksimalkan kemenangan sendiri atau pun untuk meminimalkan kemenangan lawan. Peraturan-peraturan menentukan kemungkinan tindakan untuk setiap pemain. Sejumlah keterangan diterima setiap pemain sebagai kemajuan bermain, dan sejumlah kemenangan atau kekalahan dalam berbagai situasi [2].

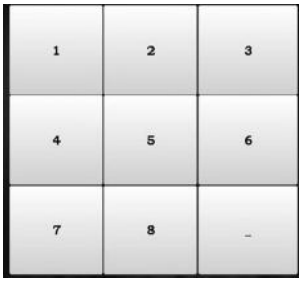
Puzzle merupakan salah satu jenis permainan yang cukup memeras otak untuk menyelesaikannya. Pemain ditantang untuk berpikir kreatif bagaimana untuk membuat semua bagian *puzzle* terletak pada posisi sebenarnya. Cara memainkannya cukup mudah, pemain hanya menggeser *puzzle* satu demi satu sampai akhirnya semua *puzzle* terletak pada posisi sebenarnya [3].

Algoritma A* merupakan algoritma pencari jalan terbaik dan merupakan gabungan dari algoritma Djikstra dan BFS. Ketiga algoritma ini menggunakan graf berbobot tidak berarah sebagai konsep dasar pencarian jejak [4]. Algoritma A* mengunjungi simpul dalam *graph* dengan cara mengunjungi simpul yang paling mendekati solusi yang dalam hal ini menganalisa algoritma A* dalam membantu mencari jalan pergeseran. Algoritma A* menerapkan *heuristic* untuk menemukan solusi yang paling optimum. *Heuristic* ini yang menyebabkan pohon ruang status tidak perlu dibangkitkan seluruhnya, hanya yang mendekati solusi terbaik saja. Pada kasus ini solusi terbaik dapat dicapai [5].

2. Tinjauan Pustaka

8 *puzzle* biasanya diwakili sebagai papan 3 * 3 yang ubinnya terdiri dari angka 1 sampai 8 yang disusun dari atas kiri sampai kanan bawah dari papan permainan (*goal board*). Permainan ini diwakili dengan papan n * n yang memiliki angka dari 0 sampai ($n^2 - 2$) sebagaimana pada masing-masing kolom dan baris merupakan indeks dari 0 (Gambar 2.1). Ubin yang tersisa (kanan bawah) adalah kosong, karena akan digunakan untuk pergeseran ubin yang lain yang berada dalam papan permainan. Teka-Teki dimainkan dengan menyusun ubin ke papan dan mencoba untuk menyelesaiannya dengan menciptakan kembali papan tujuan. Hal ini dicapai dengan menggerakkan ubin kembali ke posisi papan

tujuan dengan bergerak pada ruang kosong di sekitar papan. [6].



Gambar 1. Papan Permainan 8 *puzzle*

Ada empat operator yang dapat kita gunakan untuk menggerakan dari suatu keadaan ke keadaan yang baru [7].

1. Ubin kosong digeser kekiri.
2. Ubin kosong digeser kekanan.
3. Ubin kosong digeser keatas.
4. Ubin kosong digeser kebawah.

Permainan pergeseran angka 8 *puzzle* dapat diselesaikan dengan menggunakan bantuan struktur pohon pelacakan (*search tree*). Pohon pelacakan adalah suatu pohon (*tree*), dimana akar dari pohon berupa keadaan awal dan cabang berupa keadaan-keadaan yang mungkin terjadi dari keadaan sebelumnya serta daun merupakan keadaan akhir, yang dapat dijadikan sebagai solusi dari permasalahan. Namun, tidak semua daun dapat dijadikan sebagai solusi, dalam beberapa contoh kasus, ada beberapa atau semua daun bukan merupakan solusi dari permasalahan. Algoritma pencarian yang akan digunakan adalah algoritma *A**.

Algoritma ini pertama kali ditemukan pada tahun 1968 oleh Peter Hart, Nils Nilsson dan Bertram Raphael. Dalam tulisan mereka, algoritma ini dinamakan algoritma A. Penggunaan algoritma ini dengan fungsi heuristik yang tepat dapat memberikan hasil yang optimal, maka algoritma inipun disebut A* [8]. A*, merupakan salah satu contoh algoritma pencarian yang cukup popular di dunia. Jika kita mengetikkan Algoritma A* pada sebuah mesin pencari, seperti *google.com*, maka akan kita temukan lebih dari sepuluh ribu literatur mengenai algoritma A*. A* sebenarnya merupakan algoritma pengembangan dari BFS (*Breadth-First-Search*) untuk menemukan jalan dengan biaya terkecil dari titik awal ke titik tujuan (bisa ebih dari 1 titik tujuan). A adalah simpul yang sedang dijalankan dalam algortima pencarian jalan terpendek. Simpul adalah petak-petak kecil sebagai representasi dari area *pathfinding*. Bentuknya dapat berupa persegi, lingkaran, maupun segitiga. *Open list* adalah tempat menyimpan data simpul yang mungkin diakses dari starting point maupun simpul yang sedang dijalankan [9].

Heuristik yang paling umum digunakan adalah jarak Manhattan. Fungsi heuristik ini hanya akan menjumlahkan selisih nilai x dan nilai y dari dua buah titik. Heuristik ini dinamakan Manhattan mungkin karena di kota Manhattan di Amerika, jarak dari dua lokasi umumnya dihitung dari blok-blok yang harus dilalui saja dan tentunya tidak bisa dilintasi secara diagonal. Perhitungannya dapat ditulis sebagai berikut:

h(n) = abs(n.x-tujuan.x) + abs(n.y-tujuan.y) [10].

Algoritma A* merupakan algoritma yang membantu menemukan solusi pencarian ruang keadaan dengan mempertimbangkan total biaya lintasan yang akan dilacak sesuai dengan node yang akan dilewati. Masalah ruang keadaan disebut juga dengan *state space problem*. Ruang keadaan (*state space*) merupakan suatu ruang yang berisi semua keadaan yang mungkin dalam suatu kasus AI. *State* adalah representasi suatu keadaan pada suatu saat ataupun dekripsi konfigurasi sistem. *State space* adalah semua *state* (keadaan) yang mungkin, dan biasanya digambarkan sebagai jaringan dengan *verteks* merupakan *state* dan *edge* merupakan perubahan yang mungkin. Kondisi dalam ruang keadaan meliputi:

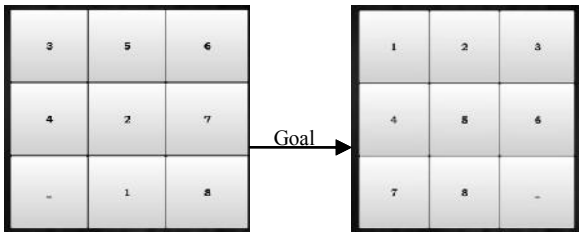
a. Keadaan awal (*initial state* / *start node*)
b. Keadaan tujuan, merupakan solusi yang dijangkau dan perlu diperiksa apakah telah mencapai sasaran.
c. Kaidah atau aturan yang memberikan bagaimana mengubah keadaan.

Keadaan direpresentasikan sebagai *node* (simpul), sedangkan langkah yang diperbolehkan atau aksi direpresentasikan dengan *arc* (busur). Representasi *state space* memungkinkan definisi formal suatu masalah sebagai persoalan mengubah status dengan menggunakan sekumpulan operator (*rule*) dan juga mendefinisikan masalah sebagai *search* yaitu mencari lintasan di dalam *state space* dari *initial state* ke *goal state*.

Manusia umumnya mempertimbangkan sejumlah alternatif strategi dalam menyelesaikan suatu masalah, dalam permainan catur misalnya, seorang pemain mempertimbangkan sejumlah kemungkinan tentang langkah-langkah berikutnya, memilih yang terbaik menurut kriteria tertentu seperti kemungkinan respon lawannya, atau memilih sejumlah langkah menurut suatu strategi global yang dirancangnya. Seorang pemain catur juga memperhitungkan pencapaian jangka pendek (seperti siasat untuk memperoleh keuntungan materi atau kualitas) [11].

3. Analisis

Proses permainan 8 *puzzle* secara umum dapat berlangsung apabila memiliki 1 pemain. Permainan ini dilakukan pada papan permainan yang berbentuk bujursangkar dengan ukuran 3x3 yang terdapat 1 ubin kosong sebagai tempat untuk menggeser ubin yang lain. Setiap ubin memiliki angka tersendiri, yaitu dari angka 1 sampai 8. Permainan ini dapat berlangsung apabila pemain telah menggerakan 1 ubin yang telah diacak untuk digeser. Terdapat banyak kemungkinan pergeseran yang akan membuat pemain merasa tidak bisa menyelesaikan permainan ini sehingga akan memerlukan banyak waktu untuk dapat menyelesaikan permainan ini. Pemain memerlukan suatu cara yaitu dengan menggunakan algoritma A* yang lebih cepat membantu menyelesaikan permainan. Pemain akan berhasil bila telah menyusun kembali ubin secara terurut seperti pada gambar 2.

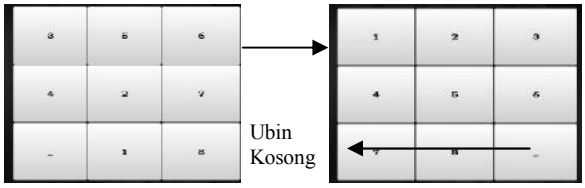


Gambar 2. Tujuan Akhir Permainan 8 *Puzzle*

Analisis kebutuhan sistem merupakan proses identifikasi dan evaluasi permasalahan-permasalahan yang ada, sehingga sistem yang dibangun sesuai dengan kriteria yang diharapkan. Aturan permainan 8 *puzzle* dan cara berpikir komputer akan diterapkan pada permainan 8 *puzzle*. Oleh karena itu, aplikasi yang dihasilkan, harus memenuhi kebutuhan sebagai berikut.

1. Aplikasi akan menerima *input* berupa nama *user*.
2. Aplikasi harus mampu menjalankan permainan 8 *puzzle* sesuai dengan aturan.
3. Aplikasi harus memiliki *hint* dan pencarian solusi. Algoritma berpikir komputer ini dibuat dengan mengikuti Algoritma A* .
4. Aplikasi harus mampu mencatat *history* berupa langkah-langkah yang diambil oleh *user* atau komputer.
5. Aplikasi dapat menyimpan dan membuka kembali permainan yang pernah disimpan sebelumnya ke *database*.
6. Aplikasi harus mampu mengatur level suara audio.

Papan permainan 8 *Puzzle* yang digunakan di dalam aplikasi akan terlihat seperti pada gambar 3. Pada saat awal permainan ubin akan diacak secara random. Pemain akan dapat memulai permainan setelah ubin teracak. Pemain harus menyusun kembali ubin yang telah diacak dengan cara menggeser ubin ke kanan, ke kiri, ke atas, ke bawah dengan melewati ubin yang kosong. Ubin kosong digunakan sebagai ruang untuk dapat menggeser ubin agar ubin dapat tersusun kembali secara berurutan. Jika Ubin telah berada pada posisi, maka pemain telah berhasil menyelesaikan permainan ini.

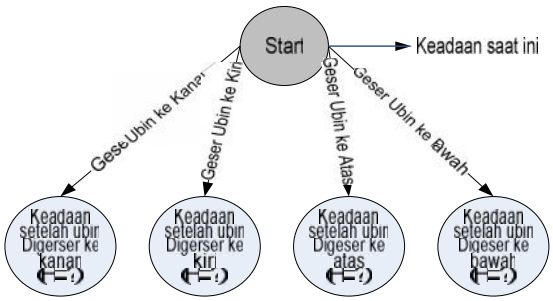


Gambar 3. Aturan Permainan 8 *Puzzle*

Aturan permainan 8 *puzzle* yang diterapkan pada aplikasi adalah sebagai berikut:

1. Pada awal permainan, ubin akan diacak sehingga ubin menjadi tidak terurut,
2. Pemain menggeser ubin ke atas atau ke bawah atau ke kiri atau ke kanan,
3. Bila pemain dapat menggeser ubin secara terurut kembali, maka pemain telah menyelesaikan permainan.

Algoritma berpikir komputer dirancang dengan menggunakan Algoritma A*. Rumusan A* ini dibuat sedemikian rupa sehingga pemain dapat melihat solusi dari permainan yang telah dijalankan. Implementasi penerapan metoda A* pada algoritma berpikir komputer dapat dilihat pada gambar berikut.



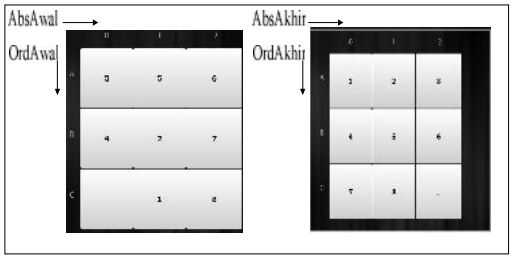
Gambar 4. Algoritma Berfikir Komputer

Pada keempat keadaan tersebut komputer akan memilih aksi yang memiliki nilai heuristik (H) yang paling kecil. Bila ada yang mempunyai nilai heuristik yang sama, maka bidak pertama yang dipilih. Rumusan untuk menghitung nilai heuristik pada masing-masing ubin adalah sebagai berikut:

Nilai Heuristik(H) = |(AbsAwal – OrdAkhir)| + |(OrdAwal – AbsAkhir)| (1)

Dimana AbsAwal adalah Posisi kolom angka

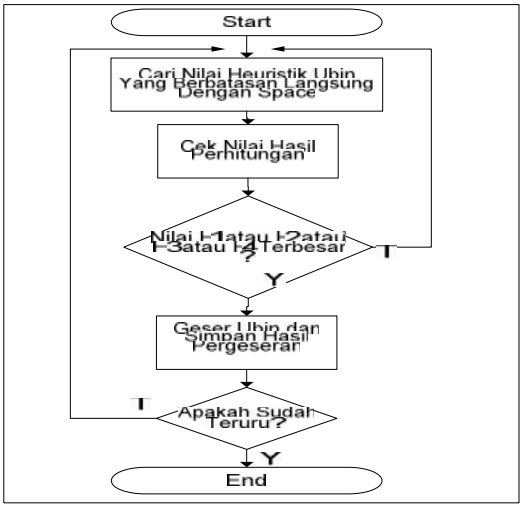
yang teracak sebelum digeser, OrdAkhir adalah Posisi baris angka yang teracak sebelum digeser, OrdAwal adalah Posisi kolom angka yang benar dan AbsAkhir adalah Posisi baris angka yang benar



Gambar 5. Perhitungan heuristik yang akan dilakukan

Algoritma A* dapat dijelaskan sebagai berikut.

1. Cari nilai heuristik untuk kotak angka yang berbatasan langsung dengan space
2. Kotak dengan nilai heuristik paling besar yang akan dipilih lalu dimasukkan ke dalam list. Di sini list befungsi sebagai pencatat status agar tidak ada pengulangan terhadap gerakan.
3. Apabila terdapat dua buah kotak dengan heuristik sama, yang digerakkan adalah kotak yang pertama.
4. Alokasi next list lalu ulangi langkah di atas dengan kondisi terakhir (setelah kotak digerakkan)
5. Setelah ditemukan solusi yang benar, maka proses pencarian solusi akan dihentikan.



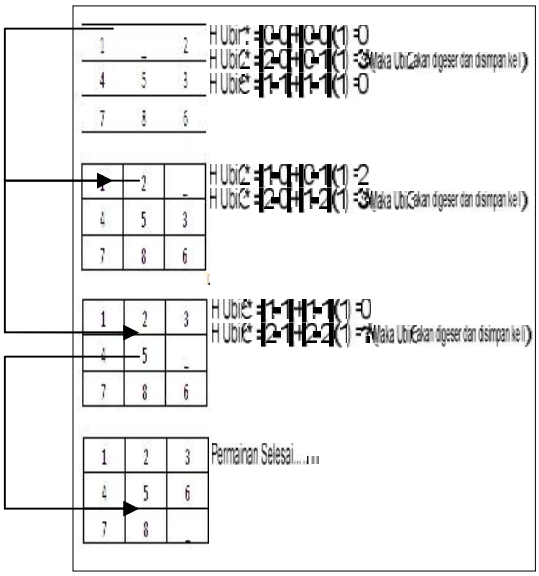
Gambar 6. Alur Algoritma A*

Gambar 7 dan Gambar 8 merupakan contoh dari penggunaan algoritma A*.

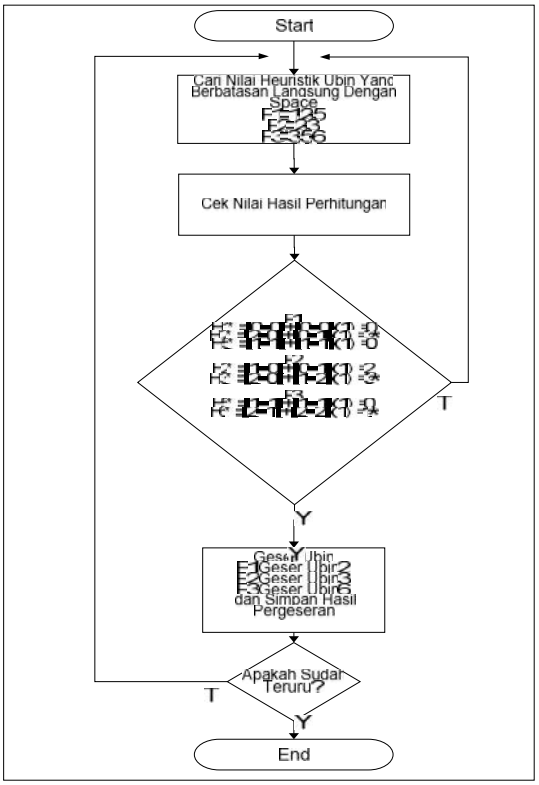
4. Hasil dan Pembahasan

Gambar 9 merupakan tampilan papan permainan 8 puzzle.

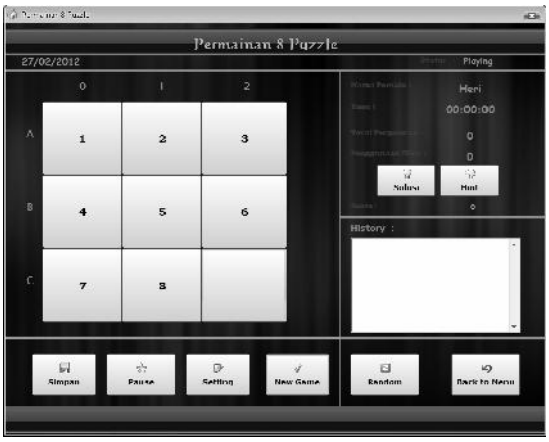
Pengujian *white box* dilakukan untuk memastikan bahwa semua jalur independen dari suatu modul telah dilalui paling sedikit satu kali. Pengujian *white box* yang dilakukan adalah pengujian *basis path* dan *Cyclomatic Complexitynya*. Setelah *flowchart* (Gambar 10) dibuat, maka *flowchart* tersebut dapat diubah menjadi *flowgraph (Gambar 11)*.



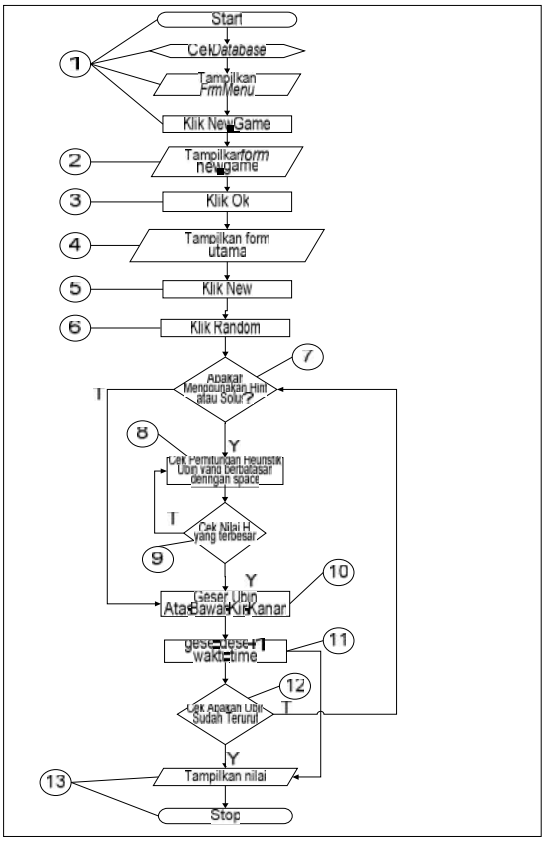
Gambar 7. Penerapan Algoritma A*



Gambar 8. *Flowchart* Penerapan Algoritma A*



Gambar 9. Permainan *Puzzle*



Gambar 10. *Flowchart* modul permainan

Setelah *flowchart* dan *flowgraph* dibuat, langkah selanjutnya adalah membuat rangkaian jalur yang dapat dilalui pada modul ruangan. Berikut ini adalah basis set yang terbentuk berdasarkan *flowgraph* modul ruangan:

Penghitungan *Cyclomatic Complexity*

E= 16,N= 13

V(G) = E - N + 2

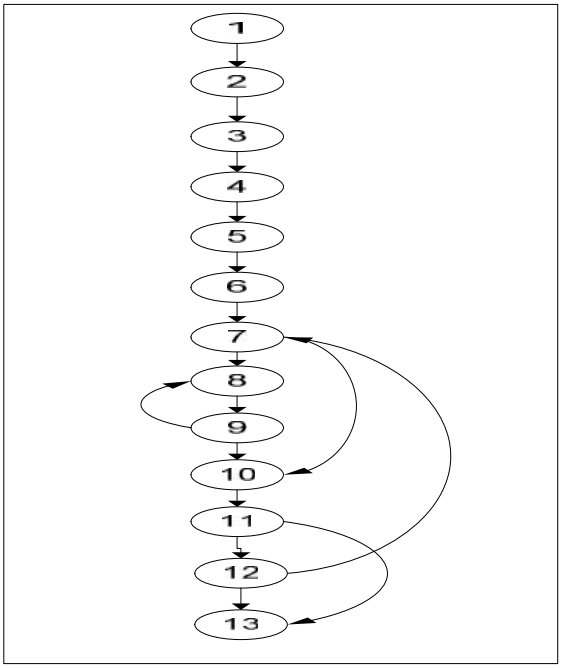
V(G) = 16-13+2= 5

Keterangan:

E = Edge, N = Node

V(G) = *Cyclomatic Complexity*

Setelah dihitung nilai *Cyclomatic Complexitnya*, ternyata didapat bahwa V(G)= 5. Maka dapat ditarik kesimpulan bahwa penerapan algoritma A* pada aplikasi *puzzle* ini memiliki struktur baik, prosedur stabil dan resikonya rendah.



Gambar 11. *Flowgraph* modul permainan

Setelah program selesai akan dilakukan pengujian algoritma yang sudah dibuat, yaitu dengan mencoba menyelesaikan kasus *puzzle* yang telah diacak sebanyak 100 kali untuk melihat waktu dan total pergeseran yang dihasilkan didapatkan data berikut:

1. TG*=3245, W*=2494, d=-2781, d^2=117689.
2. TG=6026, W=7987, d=-5493, d^2=373641.

Dengan menggunakan uji sampel berpasangan, akan diuji apakah total pergeseran dan waktu yang digunakan sebelum dan sesudah menggunakan algoritma A* mengalami kemajuan dengan perhitungan ($\alpha = 5\%$) sebagai berikut:

$TG_{1=}$ Total Geser*(Solusi)

$TG_{2=}$ Total Geser (*User*)

H_0: $TG_1 > TG_2$

H_1: $TG_1 < TG_2$

$$\bar{d} = \frac{\Sigma d}{n} = \frac{-2781}{100} = -27.81$$

$$s_d^2 = \frac{\Sigma d^2}{n-1} = \frac{117689}{100-1} = 1188,78$$

$$s_d = 34.67$$

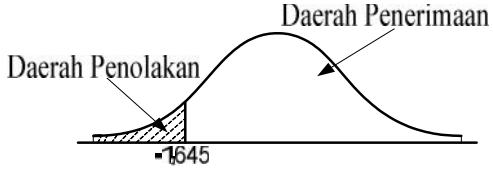
$$t_{hit} = \frac{\bar{d}}{s_d/\sqrt{n}}$$
$$= \frac{-27.81}{34.67/\sqrt{100}} = -8,02$$

$W_{1=}$ Total Geser* (Solusi)
$W_{2=}$ Total Geser (*User*)
H_0: $W_1 > W_2$
H_1: $W_1 < W_2$

$$\bar{d} = \frac{\Sigma d}{n} = \frac{-5493}{100} = -54.93$$
$$s_d^2 = \frac{\Sigma d^2}{n-1} = \frac{373641}{100-1} = 3774,15$$
$$s_d = 61.23$$
$$t_{hit} = \frac{\bar{d}}{s_d/\sqrt{n}}$$
$$= \frac{-54.93}{61.23/\sqrt{100}} = -8,9$$
$$t_{hit} = \frac{\bar{d}}{s_d/\sqrt{n}}$$
$$= \frac{-54.93}{61.23/\sqrt{100}} = -8,9$$

$\alpha = 5\% = 0.05$
df = n-1, 100 – 1 = 99

Kriteria Penerimaan



Dari hasil yang didapat, dilihat bahwa:
1. -8,02 < -1,645 berarti H_1 untuk $TG_1 < TG_2$ diterima.
2. -8,90 < -1,645 berarti H_1 untuk $W_1 < W_2$ diterima.

Hal ini berarti dengan menggunakan algoritma A* dapat membantu *user* yaitu dengan menghemat waktu permainan dan lebih sedikit melakukan pergeseran.

5. Kesimpulan

Berdasarkan penjelasan yang telah diuraikan sebelumnya, maka dapat disimpulkan sebagai berikut:
1. Total pergeseran dan waktu yang digunakan dengan menggunakan algoritma A* akan meminimalkan total pergeseran dan waktu yang dihabiskan dibandingkan dengan bermain tanpa menggunakan algoritma A*.
2. Penerapan Algoritma A* Pada Aplikasi *Puzzle* ini dapat membantu pemain dalam memainkan permainan secara cepat dan tanpa menghabiskan waktu yang cukup lama.
3. Penggunaan metode A* dapat mencari solusi masalah pergeseran dengan baik.

Daftar Pustaka

[1] Tilawah,H.2010. **Penerapan Algoritma A-star (A*) Untuk Menyelesaikan Masalah Maze.** Jurusan Teknik Informatika, ITB.
[2] Himmanudin. (2011). **Rancang Bangun Game Novel Cisual Menggunakan Ren'py v.6.11.2 .** Yogyakarta: Jurusan Teknik Informatika Fakultas Teknik Industri Universitas Islam Indonesia.
[3] Rezan A. (2009). **Penerapan Pohon dan Algoritma Heuristic dalam Menyelesaikan Sliding Puzzle .** ITB.
[4] Ecky,P .(2009). **Penerapan Algoritma A* Sebagai Algoritma Pencari Jalan Dalam Game.** Diambil dari Jurusan Teknik Informatika, Institut Teknologi Bandung.
[5] Prasetyo, Igor. **Penyelesaian Permasalahan 8 Puzzle dengan Menggunakan Algoritma A* (A Star).** Diambil dari Jurusan Teknik Informatika, Institut Teknologi Bandung.
[6] Hayes, R. (2001). ***The Sam Loyd 15-Puzzle.***
[7] Kusumadewi, S. (2003), ***Artificial Intelligence* (Teknik dan Aplikasinya)**. Penerbit Graha Ilmu, Yogyakarta.
[8] Della,P. (2010). **Algoritma Pencarian A* dengan Fungsi Heuristik Jarak Manhattan.** Diambil dari Jurusan Teknik Informatika, ITB.
[9] Victor, Ivan, Diko . **Algoritma A* (A Star) Sebagai Salah Satu Contoh Metode Pemrograman Branch and Bound.** Diambil dari *Library* ITB.
[10] Riftadi,M. (2007). **Variasi Penggunaan Fungsi Heuristik Dalam Pengaplikasian Algoritma A*.** Diambil dari Jurusan Teknik Informatika, Institut Teknologi Bandung.
[11] Desiani, A. dan Arhami, M. (2006). **Konsep Kecerdasan Buatan**, Penerbit Andi, Yogyakarta.