Jurnal Mantik Penusa

Volume 2, No. 2, Desember 2018

e-ISSN 2580-9741
p-ISSN 2088-3943

PENERAPAN METODE BEST FIRST SEARCH PADA PERMAINAN TIC TAC TOE

Harvei Desmon Hutahaean

Teknik Informatika
STMIK Pelita Nusantara Medan. Jl. Iskandar Muda No. 1 Medan, Sumatera Utara, 20154 Indonesia

Email: harvei.hutahaean@gmail.com

ABSTRAK

Pencarian adalah proses pencarian solusi di dalam suatu permasalahan sampai solusi atau tujuan ditemukan, atau pergerakan di state-space untuk mencari lintasan dari initial-state ke goal-state. Dalam sebuah permainan TIC TAC Toe proses pencarian ruang keadaan tidak cukup untuk mengotomasikan tingkah laku pemecahan masalah, pada setiap situasi tersebut hanya terdapat sejumlah pilihan langkah yang terbatas yang boleh dilakukan oleh seorang pemain. Masalah-masalah yang akan dihadapi dapat dipecahkan dengan melakukan pencarian (search) dari antara pilihan-pilihan yang ada, terdukung oleh pandangan cara penyelesaian yang biasa dilakukan. Best First Serach bekerja dengan cara melakukan pencarian pada sebuah Graf Berarah yang setiap simpulnya menggambarkan sebuah titik di dalam suatu ruang problema.

Kata kunci: pencarian, best first serach, game

I. PENDAHULUAN

Dalam permainan digunakan pencarian ruang keadaan. Untuk menampilkannya diperlukan suatu metoda yang cocok sesuai dengan masalah yang dihadapi, maka pada masalah ini akan diterapkan metoda pencarian Best First Search yaitu pencarian terbaik pertama yang paling menjanjikan sesuatu.

Masalah-masalah yang akan dihadapi dapat dipecahkan dengan melakukan pencarian (search) dari antara pilihan-pilihan yang ada, terdukung oleh pandangan cara penyelesaian yang biasa dilakukan. Proses pencarian (search) merupakan proses yang amat penting dalam mendapatkan solusi problema sulit dimana tidak ada lagi teknik langsung yang dimungkinkan untuk dipakai.

Dalam kebanyakan problema (masalah), ruang keadaan bertambah secara luar biasa seiring dengan pertambahan jumlah keadaan yang dimungkinkan. Best First Search menangani kerumitan masalah dengan cara memandu proses pencarian pada sepanjang lintasan yang paling diharapkan. Agar dapat memecahkan problema (masalah) dalam pencarian tersebut, maka dipakailah apa yang disebut dengan proses pencarian Best First Search.

II. TEORI

A. Pencarian

Pencarian adalah proses pencarian solusi di dalam suatu permasalahan sampai solusi atau tujuan ditemukan, atau pergerakan di *state-space* untuk mencari lintasan dari *initial-state* ke *goal-state*. Dimana *state-space* itu sendiri adalah himpunan semua *state* yang dapat dicapai dari *state* awal sampai *state* tujuan melalui sederetan aksi. Sedangkan *initial-state* merupakan *state* awal yaitu darimana suatu pencarian akan dimulai. *Goal-state* merupakan *state* tujuan, seringkali tujuan hanya dinyatakan sebagai sifat yang harus dipenuhi. Lintasan (*path*) dalam *state-space* adalah sederetan aksi dari satu *state* ke *state* yang lain. Untuk banyak persoalan, lintasan mana yang diambil menentukan kualitas solusi.

B. Teknik Pencarian

Ada empat hal yang harus diperhatikan untuk membangun sistem atau memecahkan masalah tertentu :

1. Mendefenisikan masalah dengan jelas.
2. Menganalisis masalah.
3. Mengumpulkan dan mempresentasikan ilmu pengetahuan (*knowledge*).
4. Memilih teknik pemecahan masalah terbaik dan menggunakannya untuk masalah tertentu.

Masalah utama dalam membangun sistem berbasis *Artificial Intelligence* (AI) adalah bagaimana mengkonversikan situasi yang diberikan ke dalam situasi lain yang diinginkan menggunakan sekumpulan operasi tertentu. Ada berbagai macam masalah yaitu :

1. *Single-state problems* yaitu satu aksi mengantarkan agen ke satu *state* yang lain.
2. *Multi-state problems* yaitu satu aksi dapat mengantarkan agen ke beberapa kemungkinan *state*.
3. *Contingency problems* yaitu hasil dari satu aksi sangat sukar untuk diprediksi, agen mengetahui efek apa yang mungkin ditimbulkan oleh aksi yang dilakukannya.

Umumnya menggunakan perencanaan (*planning*).

4. *Exploration problems* yaitu agen sama sekali tidak mempunyai informasi mengenai efek dari aksi yang dilakukannya. Agen perlu bereksperimen dan belajar atau menggunakan metoda *learning* yang ada.

C. Kriteria Pencarian

Terdapat empat kriteria dalam strategi pencarian yaitu :

1. *Completeness* : Apakah strategi tersebut menjamin penemuan solusi jika solusinya memang ada?
2. *Time complexity* :Berapa lama waktu yang diperlukan?
3. *Space complexity* :Berapa banyak memori yang diperlukan?
4. *Optimality*: Apakah strategi tersebut menemukan solusi yang paling baik jika terdapat beberapa solusi berbeda pada permasalahan yang ada?

D. Metoda Pencarian

Secara garis besar pencarian dapat dibedakan menjadi dua, yaitu Pencarian Buta (*Blind Uninformed Search*) dan Pencarian Terbimbing (*Informed Heuristics Search*). Pada *Blind Uninformed Search* tidak ada informasi mengenai jarak dari *current-state* ke *goal-state* sedangkan pada *Informed heuristic Search* ada informasi mengenai jarak dari *current-state* ke *goal-state*. Pada *Blind uninformed search* terdapat beberapa metoda, yaitu *Breadth First Search*, *Depth First Search*. Sedangkan pada *Informed Heuristics Search* yaitu : *Generate and Test*, Pendakatan Bukit (*Hill Climbing*), Pencarian Terbaik Pertama (*Best First Search*). Masing-masing metoda tersebut mempunyai karakteristik yang berbeda.

1. Breadth First Search (BFS)

Prosedur pencarian pelebaran pertama merupakan prosedur yang menjamin diperolehnya sebuah solusi jika solusi itu memang ada, dimana tersedia sejumlah pencabangan pohon (*tree*) yang berhingga. Jika terdapat sebuah solusi, maka akan ada sebuah lintasan dengan panjang berhingga dari keadaan awal ke keadaan tujuan. Pencarian pelebaran pertama ini akan mencari semua lintasan dengan panjang satu (yang panjangnya berhingga), dan kemudian akan mengamati semua lintasan dengan panjang dua (yang panjangnya juga berhingga). Proses ini terus dilanjutkan sampai semua lintasan diamati, sehingga solusi dapat diperoleh. Prosedur ini dijamin tidak hanya dapat menemukan solusi, namun juga solusi dengan lintasan yang terpendek dari tujuan.

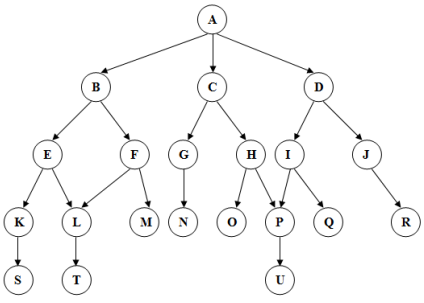
Terdapat tiga persoalan utama sehubungan dengan prosedur tersebut, yaitu :

Memerlukan memori yang besar. Jumlah simpul (*node*) di setiap tingkat dari pohon (*tree*) bertambah secara eksponensial terhadap jumlah tingkat, dan kesemuanya itu harus disimpan sekaligus. Membutuhkan sejumlah besar pekerjaan, khususnya jika lintasan solusi terpendek cukup panjang, karena jumlah simpul (*node*) yang diperlukan untuk diperiksa bertambah secara eksponensial terhadap panjang lintasan.

Ketidakrelevanan operator akan menambah jumlah simpul (*node*) yang harus diperiksa dengan sangat besar.

Untuk mengimplementasikan pencarian pelebaran pertama digunakan daftar (*list*), baik yang *open* (terbuka) maupun yang *close* (tertutup) dalam menelusuri gerakan pencarian di dalam ruang keadaan. Prosedurnya adalah sebagai berikut :

Prosedur *Breadth First Search*
inisialisasi: open = [Start]; close = []
while open /= [] do
begin
hapuskan keadaan paling kiri dari kedaan open, sebutlah keadaan itu
dengan X;
jika X merupakan tujuan then return(sukses);
buatlah semua child dari X;
ambillah X dan masukkan pada closed;
eliminasilah setiap child X yang telah berada pada open atau closed,
yang akan
menyebabkan loop dalam search;
ambillah turunan di ujung kanan open sesuai urutan penemuannya;
end.



Gambar 1 : Breadth First search Graf

Bila menelusuri prosedur diatas pada graf gambar 1, maka dengan menganggap U sebagai keadaan tujuan yang diinginkan :
open = [A]; closed = []
open = [B,C,D]; closed = [A]
open = [C,D,E,F]; closed = [B,A]
open = [D,E,F,G,H]; closed = [C,B,A]
open = [E,F,G,H,I,J]; closed = [D,C,B,A]

open = [F,G,H,I,J,K,L]; closed = [E,D,C,B,A]

open = [G,H,I,J,K,L,M] (karena L telah open); closed = [F,E,D,C,B,A]

open = [H,I,J,K,L,M,N]; closed = [G,F,E,D,C,B,A]

dan seterusnya sampai U diperoleh atau open = []

Proses pencarian *Breadth First* mengamati setiap simpul (*node*) di setiap tingkat *Graf* sebelum bergerak menuju ruang yang lebih dalam, maka mula-mula semua keadaan akan dicapai lewat lintasan yang terpendek dari keadaan awal. Karena itu, proses pencarian ini menjamin ditemukannya lintasan terpendek dari keadaan awal ke keadaan tujuan.

2. Depth First Search (DFS)

Pada *Depth First Search*, proses pencarian akan dilakukan pada semua anaknya sebelum dilakukan pencarian ke node-node yang selevel. Pencarian dimulai dari node akar ke level yang lebih tinggi, proses ini diulangi terus hingga ditemukannya solusi. Proses pencarian ini akan cepat mencapai kedalaman ruang pencarian. Jika diketahui bahwa lintasan solusi problema akan panjang, maka pencarian *Depth First* tidak akan memboroskan waktu untuk melakukan pencarian sejumlah besar keadaan 'dangkal' kedalaman dalam *Graf*, tidak melihat lintasan lebih pendek menuju tujuan atau terjebak dalam lintasan dengan panjang tak-hingga (*infinite*) yang tidak mengarah ke tujuan. *Depth_First Search* jauh lebih efisien untuk ruang pencarian dengan banyak pencabangan, karena tidak perlu harus mengevaluasi semua simpul pada suatu tingkat tertentu pada daftar *Open*.

Depth First Search mempunyai keuntungan dan kelemahan :

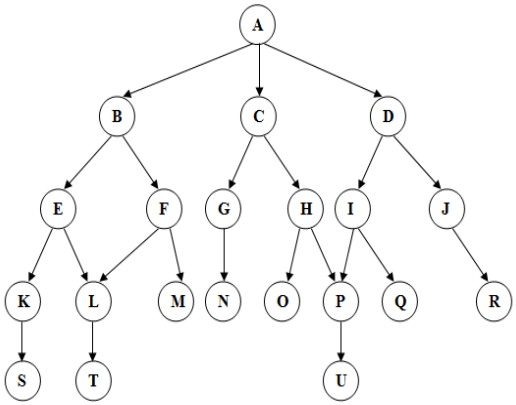
a. Keuntungan dari *Depth First Search*
   1. Membutuhkan memori yang relatif kecil, karena hanya node- node pada lintasan yang aktif saja yang disimpan.
   2. Secara kebetulan, metode depth-first search akan menemukan solusi tanpa harus menguji labih banyak lagi dalam ruang keadaan.

b. Kelemahan dari *Depth First Search*
   1. Memungkinkan tidak ditemukannya tujuan yang diharapkan.
   2. Hanya akan mendapatkan 1 solusi pada setiap pencarian.

Prosedur pencarian kedalaman pertama dapat diimplementasikan dengan melakukan proses pencarian sebagai berikut :

Prosedur Depth First Search
Inisialisasi open = [Start]; close = []
While open /= [] do
Begin
hapuskan keadaan berikutnya dari sebelah kiri open, sebutlah keadaan itu dengan X;
jika X merupakan tujuan then return(sukses);
buatlah semua child yang dimungkinkan dari X;
ambillah X dan masukkan pada closed;
eliminasilah setiap child X yang telah berada pada open atau closed, yang akan menyebabkan loop dalam search;
ambillah child yang tersisa di ujung kanan open sesuai urutan penemuannya;
end;

Untuk memeriksa algoritma diatas, bahwa keadaan-keadaan turunan (descendant) ditambahkan atau dihapuskan dari ujung kiri open. Hasil penerapan algoritma diatas ditunjukkan pada gambar



Gambar 2 : Garf setelah 6 iterasi *Depth Firs Search*
Sumber : Elearning.uin-suka.ac.id/attachment/pencarian_heuristik_bjyr6.pdf

Langkah-langkah penelusuran algoritma D*epth_First_Search* adalah :

open = [A]; closed = []
open = [B,C,D]; closed = [A]
open = [E,F,C,D]; closed = [B,A]
open = [K,L,F,C,D]; closed = [E,B,A]
open = [S,L,F,C,D]; closed = [K,E,B,A]
open = [L,F,C,D]; closed = [S,K,E,B,A]
open = [T,F,C,D]; closed = [L,S,K,E,B,A]
open = [F,C,D]; closed = [T,L,S,K,E,B,A]
open = [M,C,D]; closed = [F,T,L,S,K,E,B,A]
open = [C,D]; closed = [M,F,T,L,S,K,E,B,A]

dan seterusnya sampai diperoleh U atau open = [].

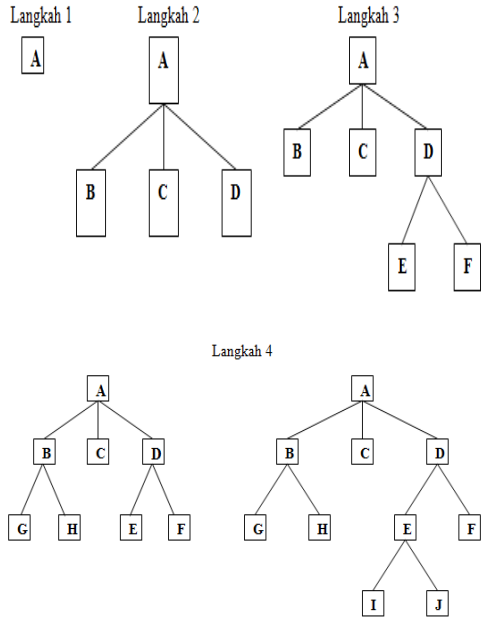
Algoritma pencarian *Depth First* dapat menyimpan sebuah *record parent* pada setiap keadaan, dan dapat merekonstruksi lintasan yang menghasilkan keadaan tujuan dari keadaan awal. Pencarian *Depth First* tidak menjamin ditemukannya lintasan terpendek menuju keadaan ketika pertama kali ditemuinya.

3. Best First Search (BFS).

a. Graf Or

Pencarian terbaik pertama (*Best First Search*) merupakan suatau cara yang menggabungkan keuntungan atau kelebihan dari pencarian *Breadth First* dan *Depth First*. Pada setiap langkah proses pencarian terbai pertama, kita memilih simpul-simpul (*node*) yang paling menjanjikan sesuatu. Hal ini dilakukan dengan menerapkan fungsi *Heuristik* yang memadai pada setiap simpul tersebut. Kemudian, kita mengembangkan simpul yang kita pilih itu dengan menggunakan aturan-aturan tertentu untuk menghasilkan penggantinya. Jika salah satu simpul tersebut merupakan sebuah solusi, kita berhenti. Jika bukan, semua simpul baru itu ditambahkan ke himpunan simpul yang sejauh ini telah dibuat. Setelah itu, kembali kita memilih simpul yang paling diharapkan dan melanjutkan proses. Namun, jika tidak bisa mendapatkan sebuah solusi, pencabangan itu akan mulai mencari simpul yang kurang menjanjikan sesuatu daripada cabang di tingkat puncak yang telah kita abaikan. Di titik itu, cabang yang saat ini lebih menjanjikan sesuatu dan sebelumnya sempat diabaikan, akan disimak. Namun cabang yang lebih tua tidak dilupakan. Simpul terakhirnya tetap berada di dalam himpunan yang berisi simpul-simpul yang telah dibuat namun tidak dikembangkan. Proses pencarian dapat dikembalikan padanya bila semua simpul lainnya tidak menjanjikan apa-apa dan justru simpul terakhir ini merupakan lintasan yang paling dapat diharapkan.

Gambar 3 berikut menunjukkan awal prosedur pencarian terbaik pertama (*Best_First*). Mula-mula hanya terdapat satu simpul yang kemudian dikembangkan menjadi tiga simpul baru. Dalam contoh ini, fungsi *Heuristik* yang merupakan pengukur biaya dalam memperoleh sebuah solusi dari sebuah simpul (*node*) tertentu, diterapkan pada simpul-simpul yang baru tersebut. Karena D merupakan simpul yang paling menjanjikan sesuatu, maka simpul ini yang kemudian dikembangkan dan menghasilkan dua simpul pengganti E dan F. lewat lintasan yang lain, katakanlah B tampak cukup dapat diharapkan, dan bila diproses akan didapatkan simpul G dan H. namun setelah dievaluasi, simpul-simpul baru ini tidak begitu dapat diharapkan dibandingkan dengan lintasan yang pertama tadi (lewat simpul D), sehingga perhatian kembali diarahkan ke simpul D menuju ke E. Bila E dikembangkan, diperoleh simpul I dan J. Selanjutnya, simpul J yang akan dikembangkan karena simpul ini yang tampaknya paling dapat diharapkan. Proses semacam ini akan terus dilanjutkan sampai diperoleh sebuah solusi.



Gambar 3 : Ilustrasi pencarian terbaik pertama (*best_first_search*)
Sumber : imamwardany.com/wp-content/uploads/2007/12/pencarianterbaikpertama.pdf

Implementasi pencarian *Best_First* dapat dilakukan dengan algoritma A*. Algoritma ini bekerja dengan cara melakukan pencarian pada sebuah *Graf* Berarah yang setiap simpulnya menggambarkan sebuah titik di dalam suatu ruang problema. Setiap simpul (*node*) akan berisi sebuah penunjuk seberapa besar simpul itu dapat diharapkan sebagai tambahan terhadap depenelitian keadaan problema yang direpresentasikannya, serta penghubung parent yang mengarah balik pada simpul terbaik dan sebuah daftar simpul yang dihasilkan darinya. Penghubung *Parent* ini akan memungkinkannya untuk menemukan kembali lintasan yang mengarah ke tujuan begitu tujuan tersebut ditemukan. Daftar pengganti atau penerusnya (*successor*) akan memungkinkannya untuk

melanjutkan pengembangan ke penggantinya jika ditemukan sebuah lintasan yang lebih baik dari sebuah simpul yang telah ada. Ada dua daftar simpul yang digunakan yaitu :

1) *Open* (Terbuka)

Simpul-simpul yang telah dihasilkan dan ada fungsi *Heuristik* yang terterap padanya, namun belum diperiksa. Daftar ini sebenarnya merupakan antrian yang diprioritaskan dengan elemen yang memiliki prioritas tertinggi sebagai nilai fungsi *Heuristik* yang paling dapat diharapkan.

2) *Close* (Tertutup)

Simpul-simpul yang telah diperiksa. Berikut ini adalah *Algoritma A** :

Mulailah dengan open yang hanya berisi simpul awal. Buatlah nilai g simpul ini sama dengan 0, nilai h'nya berapa saja, serta nilai f'nya sama dengan h' + 0, atau h'. buatlah closed untuk suatu daftar kosong.

Prosedur berikut akan diulangi sampai simpul tujuan didapatkan :

Jika tidak ada simpul pada open, dilaporkan adanya kesalahan. Jika ada, simpul pada open diberi nilai f' terendah , kemudian simpul ini disebut Simpul_Terbaik. Setelah itu, simpul ini dipindahkan dari open dan diletakkan pada closed. Kemudian simpul ini ditelaah apakah merupakan simpul tujuan?. Jika ya, maka ke luar (exit) dan melaporkan solusi. Jika tidak, buatlah pengganti Simpul Terbaik. Pada keadaan yang pasti, algoritma A* menunjukkan hasil yang optimal dalam menghasilkan jumlah simpul tersedikit dalam proses menemukan solusi sebuah problema.

Adapun algoritma (*pseudocode*) heuristik (*Best First Search*) sebagai berikut :

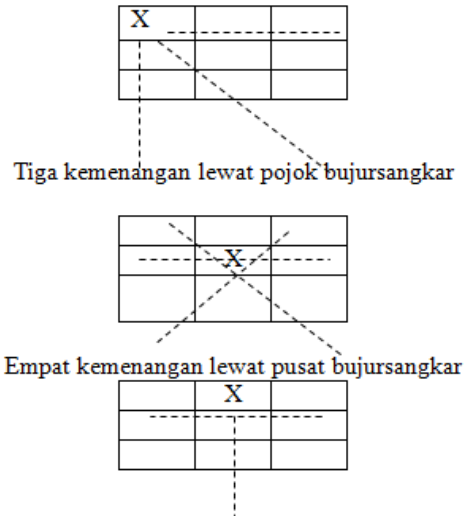
```
Procedure Best First Search
Inisialisasi : open = [Mulai]; closed = []
    While open /= do
    begin
      ambillah keadaan berikutnya dari open, sebutlah ini dengan X;
      jika X merupakan tujuan the return lintasan solusi yang menghasilkan X;
      proseslah X; buatlah semua child-nya;
      untuk setiap child X
            do case
      child belum berada pada open atau closed :
      begin
      tentukan nilai heuristik untuk keadaan child;
      tambahkan keadaan child pada open;
      end;
      child telah berada pada open :
      jika telah sampai pada open lewat lintasan yang lebih pendek dari
      keadaan saat ini then berikan nilai lintasan yang lebih pendek ini pada
      open child telah berada pada closed :
      jika child telah sampai pada closed lewat lintasan yang lebihpendek dari keadaan saat ini
      then
      begin
      berikan nilai lintasan yang lebih pendek ini pada
      closed; move keadaan ini dari closed ke open
      end;
      end;
      letakkan X pada closed;
      atur ulang keadaan-keadaan pada open menurut heuristik
      (nilai yang terbaik pada urutan pertama)
      end;
      return (gagal); {open telah selesai diperiksa seluruhnya}
      end.
```

III. PEMBAHASAN

A. Implementasi metoda pencarian terbaik pertama (*Best First Search*) untuk penyelesaian permainan Tic Tac Toe, hal mendasar yang diperhatikan dalam permainan ini yaitu bagaimana permainan tersebut melakukan pencarian dan dapat diselesaikan dengan cepat.

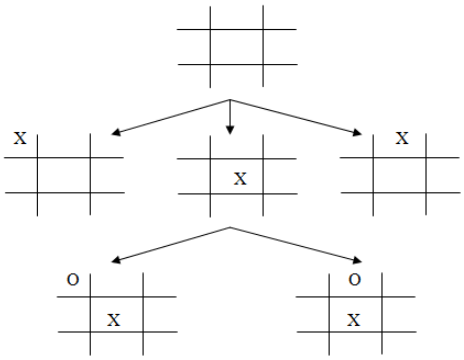
B. Pada penelitian ini permainan yang dimaksud adalah permainan Tic Tac toe yaitu permainan yang dimainkan oleh dua orang yang akan mengklick sehingga muncul nilai atau gambar sesuai yang dibuat (O dan X). Jika nilai X atau O-nya membentuk tiga nilai yang sama baik secara mendatar, menurun atau melalui diagonalnya maka pemain tersebut yang menang.

C. Pemain pertama dapat melakukan gerakan dimana X memilki garis yang paling diharapkan untuk mencapai kemenangan. Pencarian *Best First* akan memilih dan melakukan gerakan menuju keadaan dengan nilai *Heuristik* tertinggi. Dalam hal ini, X diletakkan di pusat papan. Seperti gambar berikut :



Gambar 4 : Paling mungkin untuk menang yang diterapkan pada permainan tic tac toe

Setelah langkah pertama, pemain ke-dua dapat memilih dua alternatif. Manapun yang dipilih pemain ke-dua, *Best First search* dapat tetap diterapkan dan melakukan pemilihan langkah dari langkah-langkah yang dimungkinkan seperti gambar berikut :



Gambar 5 : Ruang keadaan yang tereduksi secara *heuristik*

Permasalahan yang telah dibahas sebelumnya dicoba merancang sistem dengan membuat aplikasi permainan tersebut.

IV. IMPLEMENTASI

Implementasi merupakan percobaan suatu sistem yang telah jadi ke dalam permasalahan yang sebenarnya. Pengimplementasian suatu program akan berpengaruh pada spesifikasi komputer yang digunakan, agar program bisa berjalan dengan baik maka spesifikasi perangkat keras dan perangkat lunak harus sesuai.

A. Form Menu Utama

Form menu utama merupakan *form* yang berisi pilihan-pilihan menu utama atau tampilan pertama saat program dijalankan.

Gambar 5 : Form Menu Utama

B. Form Permainan (Game)

Form permainan merupakan tampilan dari permainan.

Gambar 6. Form Game

V. KESIMPULAN

1. Algoritma Best First Search dapat diimplementasikan dalam penentuan pencarian rute terpendek, penetuan langkah dan juga dalam kasus TSP.
2. Algoritma Best First Search pada game tic tac toe cukup bagus dengan nilai heuristiknya yang relatif kecil daripada kesempatan yang disediakan. Sehingga, skor yang diperoleh dapat lebih tinggi.

VI. REFERENSI

[1] Sandy S. Artificial Intelligence. Yogyakarta : Andy offset. 1993.
[2] Sony Daniswara, Ryan. Mencari dan memperbaiki Handphone. Depok: kawan pustaka. 2005
[3] Adipranata, dkk, 2007. Aplikasi Pencarian Rute Optimum pada Peta guna Meningkatkan Efisiensi Waktu Tempuh Pengguna Jalan Dengan Metode A* dan Best First Search. Jurnal Informatika Vol. 8, N0. 2
[4] Kusumadewi, Sri. 2003. Artificial Intelligence (Teknik dan Aplikasinya), yogyakarta, Graha Ilmu.
[5] Kusumadewi, Sri. 2003. Pengantar Kecerdasan Buatan. Yogyakarta, Graha Ilmu.
[6] Zi, Nurullina. 2011. Implementasi Konsep Kecerdasan Buatan dengan Metode Best First Search (BFS) untuk Pembuatan Game Ular Tangga Modifikasi. Universitas Sumatera Utara.