Penerapan Algoritma Backtracking pada Proses Pembuatan Permainan Math Maze

Vincent Theophilus Ciputra - 13513005
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13513005@std.stei.itb.ac.id

***Abstract*—Dalam materi kuliah IF2211 Strategi Algoritma terdapat beberapa pokok bahasan. Pokok bahasan yang akan dibahas dalam makalah ini adalah algoritma *backtracking* dan *depth-first-search.* Algoritma *backtracking* ini akan diterapkan pada permainan *math maze*. Permainan *math maze* merupakan game sederhana yang bertujuan untuk menentukan jalur yang tepat agar sampai pada tujuan yang telah ditetapkan. Pada makalah ini, akan dibahas tentang proses pembuatan papan permainan *math maze.* Proses pembuatan papan permainan *math maze* itu terdiri dari proses pembangkitan sebuah labirin, pencarian solusi dari labirin yang sudah dibangkitkan, dan membuat papan permainan *math maze* baru dengan memanfaatkan labirin yang sudah diketahui solusinya. Dengan algoritma *backtracking*, papan permainan *math maze* ini dapat menghasilkan 1 solusi untuk setiap labirin dan labirin tidak akan memiliki loop atau ruang terbuang.**

Keywords*—Algoritma, *Backtracking*, *Depth-First-Search*, *Math maze

I. PENDAHULUAN

Game komputer merupakan salah satu aplikasi *software* yang saat ini banyak dikembangkan karena termasuk *software* yang diminati oleh berbagai pelanggan. Game komputer memiliki jenis yang bermacam-macam dengan tampilan yang menarik. Selain itu, game komputer juga dapat menjadi salah satu sarana *refreshing* yang cukup menyenangkan terutama bagi orang yang telah terbiasa menggunakan komputer.

Permainan *Math Maze* merupakan game sederhana yang bertujuan untuk menentukan jalur yang tepat agar sampai pada tujuan yang telah ditetapkan. Permainan ini memiliki perbedaan dari permainan-permainan labirin yang lain. Perbedaan permainan ini dari permainan labirin biasa adalah dalam permainan ini, pemain harus menemukan jalur pada labirin dengan menggunakan angka-angka pada bagian sisi kiri dan sisi atas sebagai indikasi berapa banyak kotak yang dilewati untuk setiap baris atau setiap kolom dan tidak menggunakan tembok penghalang seperti pada labirin biasa.

Makalah ini akan fokus pada pembuatan papan permainan *math maze* yang bisa ditentukan oleh pemakai baik dari penentuan ukuran papan permainan juga posisi garis *start* dan *finish* dengan menerapkan algoritma *backtracking*.

II. LANDASAN TEORI

A. Artificial Intelligence

Artificial Intelligence (AI), atau biasa disebut kecerdasan buatan adalah salah satu cabang ilmu pengetahuan yang berhubungan dengan pemanfaatan mesin untuk memecahkan suatu persoalan yang rumit dengan cara yang lebih manusiawi. Hal ini biasanya dilakukan dengan mengikuti atau mencontoh karakteristik dan analogi berpikir dari kecerdasan manusia, dan menerapkannya sebagai algoritma yang dikenal oleh komputer.

Agar komputer bisa bertindak seperti dan sebaik manusia, maka komputer juga harus diberi bekal pengetahuan dan mempunyai kemampuan untuk menalar. Untuk itu AI akan mencoba untuk memberikan beberapa metoda untuk membekali komputer dengan kedua komponen tersebut agar komputer bisa menjadi mesin pintar.

Dalam perkembangan teknologi AI, banyak sekali masalah yang dimiliki dan bisa dikategorikan ke dalam beberapa sub-masalah seperti :

- Kemampuan untuk mengambil keputusan,
- Representasi pemikiran,
- Perencanaan atau penjadwalan,
- Pembelajaran,
- Persepsi dan daya tanggap,
- Gerakan dan manipulasi objek,
- Kecerdasan sosial,
- Kreativitias mesin,
- Kecerdasan dasar.

Teori-teori ini dapat dinyatakan dalam bahasa program komputer dan dibuktikan eksekusinya pada komputer nyata, contohnya adalah permainan pada komputer. Permainan pada komputer ini menggunakan teori kecerdasan berupa algoritma yang dapat digunakan untuk membuat *problem* atau mencari solusi dari *problem* yang ada. Pada proses membuat *problem* dalam permainan *Math Maze* yang digunakan adalah algoritma *backtracking.*

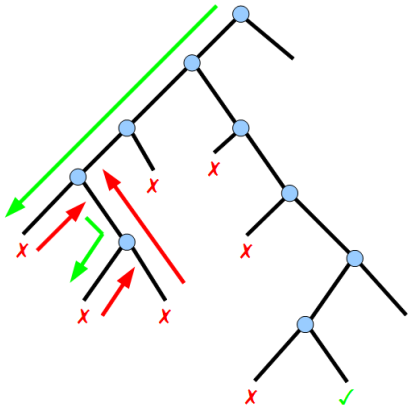
B. Algoritma Backtracking

Algoritma *backtracking* (runut balik) merupakan salah satu metode pemecahan masalah yang termasuk dalam strategi yang berbasis pencarian pada ruang status. Algoritma *backtracking* bekerja secara rekursif dan melakukan pencarian solusi persoalan secara sistematis pada semua kemungkinan solusi yang ada. Algoritma *backtracking* yang digunakan berbasis pada algoritma *Depth-First-Search* (DFS), maka pencarian solusi dilakukan dengan menelusuri suatu struktur berbentuk pohon berakar secara *preorder*.

Prinsip dasar algoritma *backtracking* adalah mencoba semua kemungkinan solusi yang ada. Perbedaan utamanya adalah pada konsep dasarnya, yaitu pada *backtracking* semua solusi dibuat dalam bentuk pohon solusi, dan kemudian pohon tersebut akan ditelusuri secara DFS sehingga ditemukan solusi terbaik yang diinginkan.

Seperti yang telah dijelaskan bahwa pencarian solusi dengan menggunakan algoritma *backtracking* ini berbasis pada DFS, maka kita menggunakan pohon ruang status. Langkah-langkah pencarian solusi dengan *backtracking* adalah sebagai berikut :

1. Solusi dicari dengan membentuk lintasan akar ke daun. Simpul yang sudah dilahirkan dinamakan simpul hidup dan simpul hidup yang diperluas dinamakan simpul-E (Expand-node).
2. Jika lintasan yang diperoleh dari perluasan simpul-E tidak mengarah ke solusi, maka simpul itu akan menjadi simpul mati dimana simpul itu tidak akan diperluas lagi.
3. Jika posisi terakhir ada di simpul mati, maka pencarian dilakukan dengan membangkitkan simpul anak yang lainnya dan jika tidak ada simpul anak, maka dilakukan *backtracking* ke simpul orang tua.
4. Pencarian dihentikan jika telah menemukan solusi atau tidak ada simpul hidup.

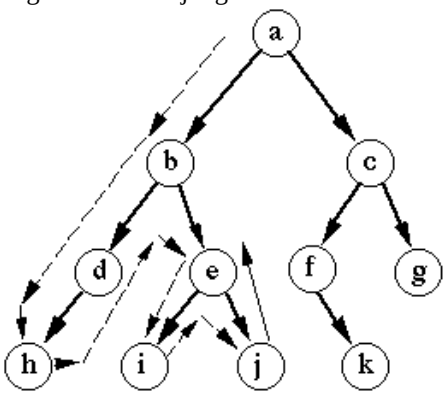


Gambar 1. Contoh algoritma *backtraking*
(Sumber : referensi [3])

C. Algoritma Depth-First-Search

Depth First Search adalah metode pencarian secara mendalam. Algoritma DFS ditunjukkan sebagai berikut. Misal simpul akar adalah simpul v.

1. Mengunjungi simpul v
2. Mengunjungi simpul w yang bertetangga dengan simpul v.
3. Ulangi langkah 1 dengan simpul w.
4. Ketika telah mencapai suatu simpul u dan tidak ada lagi simpul yang bertetangga dengannya, maka pencarian dirunut balik ke simpul terakhir yang dikunjungi sebelumnya.
5. Pencarian berhenti ketika tidak ada simpul lagi yang belum dikunjungi.



Gambar 2. Contoh algoritma *Depth First Search*
(Sumber : referensi [4])

Penelusuran secara DFS pada gambar 3 dimulai dari simpul a. Anak pertama dari simpul a yaitu b ditelusuri. Dari b, d ditelusuri dan dari d , h ditelusuri. Simpul h tidak memiliki anak simpul lagi, maka dilakukan *backtracking* ke simpul d. d juga tidak memiliki simpul anak, maka balik lagi ke b. b masih memiliki simpul e maka e ditelusuri. Begitu seterusnya sampai secara keseluruhan penelusuran tersebut akan menjadi a-b-d-h-e-i-j-c-f-k-g.

Kelebihan DFS adalah memakai memori yang cukup kecil karena hanya simpul yang aktif saja yang disimpan dan jika solusi berada pada level yang dalam dan paling kiri maka DFS akan menemukannya dengan cepat. Kelemahan DFS adalah jika pohon yang dibangkitkan memiliki level yang dalam (tak terhingga) dimungkinkan tidak ditemukan solusi. Selain itu, jika terdapat lebih dari satu solusi pada level yang berbeda, DFS tidak menjamin menemukan solusi yang optimal.

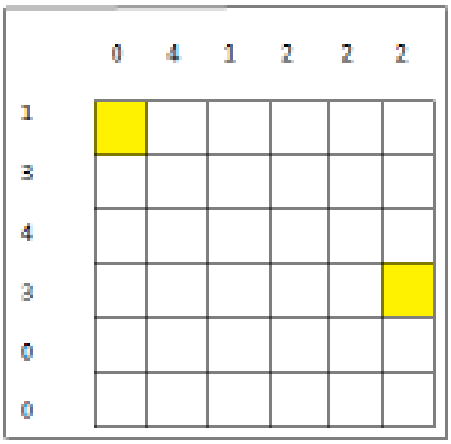
D. Math Maze

Permainan *Math Maze* adalah permainan yang dimainkan oleh 1 orang. Permainan ini merupakan game sederhana yang bertujuan untuk menentukan jalur yang tepat agar sampai pada tujuan yang telah ditetapkan. Permainan ini memiliki perbedaan dari permainan-permainan labirin yang lain. Perbedaan yang ada pada *math maze* adalah permainan ini menggunakan angka-angka yang ada pada sisi bagian kiri dan bagian atas yang

disebut dengan nilai-nilai pemandu. Angka-angka tersebut merupakan indikator yang menyatakan berapa banyak sel di dalam baris atau kolom tersebut yang dilalui oleh garis yang merupakan jalur menuju tujuan akhir.

Arah pergerakan jalur ditentukan dengan mengklik dari satu sel ke sel lain pada grid. Pilihan sel untuk di klik ada 4, yaitu atas, bawah, kirim dan kanan. Pemain tidak boleh mengklik sel lain dengan arah diagonal. Saat di klik, sel satu akan terhubung dengan sel selanjutnya dengan suatu garis. Apabila pemain berhasil mencapai pintu keluar atau *finish*, maka akan terbentuk jalur dari garis-garis tersebut.

Dengan nilai-nilai pemandu pada sisi kiri dan sisi atas papan permainan, maka pemain harus memperhitungkan sel mana yang harus diklik setelah pemain mulai keluar dari sel *start* atau pintu masuk. Pemain harus memperkirakan jalur solusi yang terbentuk nanti agar sesuai dengan nilai-nilai pemandu. Apabila jumlah nilai baris dan koom yang dilewati tidak sesuai sampai mencapai garis *finish*, maka dianggap gagal menemukan jalur yang tepat.



Gambar 3. Papan permainan *Math Maze* ukuran 6x6
(Sumber : referensi [5])

III. PENERAPAN METODE

A. Pembangkitan Maze

Proses pembangkitan sebuah maze yang dilakukan dengan *backtracking* adalah dengan prosedur :

1. Pilih satu sel pada sebuah *grid.*
2. Pilih secara acak sel terdekat atau sel tetangga dengan gerakan atas, bawah, kiri, atau kanan. Tidak diperbolehkan memilih sel tetangga secara diagonal.
3. Jika sel tetangga tersebut belum pernah kita datangi, maka pindah ke sel tersebut dan hapus pembatas antara sel tetangga tersebut dengan sel yang sebelumnya ditempati.
4. Jika tidak ditemukan sel tetangga yang belum didatangi, maka kita harus melakukan *backtracking* ke sel sebelumnya.
5. Langkah ini dilakukan sampai semua sel pada *grid* sudah didatangi.
6. Tentukan posisi *start* dan *finish*.

B. Pencarian solusi dari Maze yang sudah dibangkitkan

Apabila *maze sudah dibangkitkan*, maka selanjutnya adalah mencari solusinya. Langkah-langkah untuk menentukan solusinya :

1. Dari sel *start*, pilih secara acak sel terdekat yang bisa dikunjungi. Sama seperti proses pembangkitan *maze*, hanya ada 4 gerakan yang diperbolehkan, yaitu atas, bawah, kiri, dan kanan.
2. Apabila sel belum pernah didatangi, maka pindah ke sel tersebut. Pilih lagi sel tetangga secara acak.
3. Apabila sel menemui tembok atau *wall*, maka lakukan *backtracking* ke sel sebelumnya.
4. Lakukan langkah-langkah ini sampai menemukan sel *finish.*

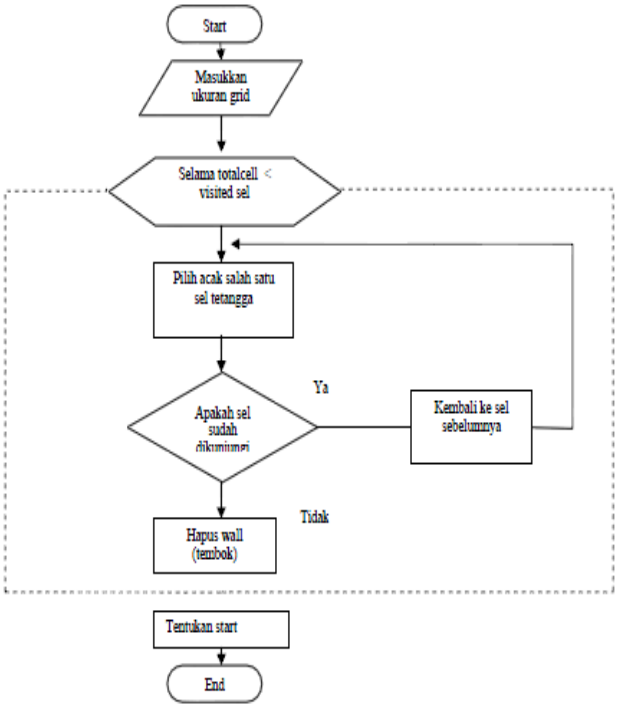
C. Pembuatan papan permainan Math Maze

Proses ini adalah proses untuk membuat papan permainan *Math Maze* dengan memanfaatkan *maze* yang sudah dibangkitkan dan solusi yang sudah ditemukan. Langkah-langkahnya adalah menghitung jumlah total sel pada baris dan kolom yang dilewati oleh garis atau jalur solusi. Jumlah sel yang dilewati pada baris dan kolom itulah yang nantinya akan menjadi nilai-nilai baris dan kolom sebagai panduan pada papan permainan. Papan permainan *Math Maze* sendiri akan dibentuk dengan menghilangkan tembok atau *wall* pada *maze* sempurna sehingga akan kembali berbentuk seperti *grid* biasa, hanya saja sudah disediakan sel *start* dan sel *finish* serta nilai-nilai pada baris dan kolom sebagai panduan untuk menemukan jalur solusi. Dengan demikian papan permainan sudah dapat dimainkan oleh pemain.

D. Algoritma Program

Langkah pertama dalam proses pembangkitan ini dilakukan dengan algoritma *backtracking* untuk membentuk sebuah *maze*. Langkah-langkah tersebut yaitu:

1. Masukkan ukuran *grid* (i x j).
2. Selama total sel pada *grid* < sel yang telah dikunjungi, lakukan langkah 3, 4, dan 5.
3. Pilih secara acak sel tetangga (atas, bawah, kiri, kanan).
4. Cek apakah sel sudah pernah dikunjungi, jika ya, kembali ke langkah 3, jika tidak, lakukan langkah 5.
5. Hapus *wall* atau pembatas pada sel.
6. Tentukan sel *start* dan sel *finish.*
7. Selesai.



Gambar 4. Flowchart pembangkitan *maze*

Setelah *maze terbentuk* dan sel star dan sel *finish* sudah ditentukan, maka dilakukan pencarian solusi pada *maze* tersebut. Pada tahap ini digunakan algoritma *backtracking*. Penggunaan algoritma *backtracking* ini terlihat pada proses penelusuran tiap jalur untuk mencapai tujuan yang diinginkan. Sejak komputer mulai mencari solusi, komputer akan menentukan jalur dengan menelusuri sembarang jalur. Ketika komputer menemukan jalan buntu, maka ia akan melakukan proses *backtrack* dengan cara kembali pada jalur sebelumnya sampai menemukan jalur baru yang belum pernah dilewati.

Pada proses pembangkitan *maze* dan pencarian solusi, arah gerakan yang diperbolehkan adalah atas, bawah, kiri, dan kanan. Selain keempat arah gerakan ini, tidak diperbolehkan melakukan gerakan lain seperti gerakan diagonal atau melompati sel. Setelah solusi ditemukan, maka dilakukan langkah ketiga, yaitu menghitung jumlah sel yang dilalui oleh jalur solusi pada bagian baris dan kolomnya. Jumlah total pada baris dan kolom yang dilalui jalur menjadi angka-angka pada sisi kiri dan atas papan permainan sebagai panduan pemain dalam mencari solusi. Langkah terakhir adalah menghilangkan tampilan *maze* sempurna dan solusinya sehingga papan permainan hanya menampilkan *grid* dengan sel *start* dan sel *finish* serta angka-angka pada sisi kiri dan atas papan permainan.

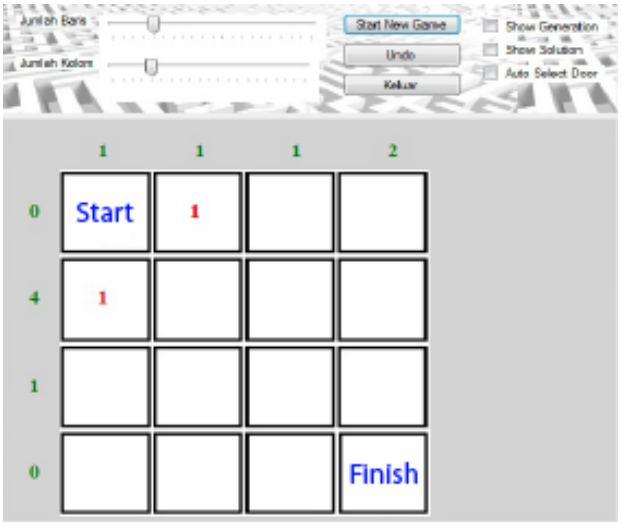
Algoritma dari program tersebut adalah sebagai berikut:

1. Mulai
2. Buat *stack generate* untuk struktur data
3. *Set totalcell* = jumlah sel pada *grid.*
4. Pilih satu sel secara acak sebagai *current cell*.
5. *Set visited* sel = 1.
6. Selama *visited cell* < *totalcell* lakukan langkah 7 sampai 9.
7. Cek sel tetangga dari *current cell*, jika lebih dari 1 lakukan langkah 8 sampai 11, jika = 1, lakukan langkah 9.
8. Pilih salah satu secara acak.
9. Simpan sebagai *visited cell*.
10. Masukkan data *visited cell* ke *stack generate*.
11. Jadikan sel tetangga tersebut menjadi *current cell* yang baru.
12. Selesai.

Pada algoritma ini, proses *backtracking* ditunjukan dengan mengeluarkan data dari *stack generate* di mana data tersebut mengarah ke sel yang telah dikunjungi kembali ke data di mana sel masih memiliki kemungkinan pengecekan sel tetangga lainnya.

IV. IMPLEMENTASI DAN PEMBAHASAN

A. Form Permainan Math Maze



Gambar 5. Tampilan Papan permainan *Math Maze*

Pada *form* permainan *Math Maze* (gambar 4), pemain harus menemukan jalur dari sel *start* sampai ke sel *finish*. Beberapa komponen yang ada adalah :

1. *Trackbar* Baris
   Trackbar baris ini merupakan komponen yang menerima masukan dari pemain berupa jumlah baris papan permainan yang ingin dimainkan. Jumlah baris diwakili dengan turus-turus yang ada pada *trackbar* dengan jumlah maksimal 20.
2. *Trackbar* Kolom
   Trackbar kolom ini merupakan komponen yang menerima masukan dari pemain berupa jumlah kolom papan permainan yang ingin dimainkan. Jumlah kolom diwakili dengan turus-turus yang ada pada *trackbar* dengan jumlah maksimal 20.
3. Tombol *Start New Game*
   Tombol ini akan membuat papan permainan yang baru sesuai dengan masukan pemakai yaitu jumlah

baris dan kolom yang diinginkan.

4. Tombol *Undo*
   Tombol *Undo* ini adalah tombol untuk mundur atau menghapus langkah tersebut dan kembali ke langkah sebelumnya apabila pada saat pemain salah menentukan langkah yang dipilih.
5. *Checkbox Show Generation*
   Apabila *checkbox* ini dicentang, maka pada saat pemain mengklik tombol *Start New Game* akan diperlihatkan proses sistem saat membangkitkan papan permainan.

B. Algoritma Pembangkitan Maze

Algoritma *backtracking* adalah algoritma berbasis pada metode *depth first search*. Metode inilah yang diterapkan pada proses untuk membentuk sebuah *maze*. Langkah-langkahnya adalah :

1. Buat papan permainan awal berupa *grid* atau sel-sel yang temboknya masih tertutup satu sama lain.
2. Pilihlah salah satu sel pada *grid* secara acak sebagai titik mulai.
3. Pilih lagi secara acak sel yang terdekat atau sel tetangganya.
4. Jika sel terdekat yang dipilih belum pernah dikunjungi, hapus tembok yang dilewati antara sel tersebut dan jadikan sebagai titik mulai yang baru.
5. Jika sel yang dipilih sudah pernah dikunjungi, maka kembali lagi ke sel sebelumnya, yaitu sel titik mulai. Apabila sampai pada titik dimana semua sel sudah dikunjungi, pilih secara acak salah satu titik pada jalur yang sudah dilewati dan kembali ke titik tersebut, lalu cek apakah ada sel yang belum dilewati.
6. Ulangi langkah 3, 4, dan 5 pada semua sel di *grid*.

Penerapan *backtracking* dilakukan dengan mengeluarkan data dari *stack* sehingga struktur data hanya akan melakukan *backtrack* ke posisi sel yang tidak mengarah ke sel yang sudah dikunjungi.

C. Algoritma Pencarian Solusi pada Maze yang Dibangkitkan

Algoritma yang diterapkan untuk pencarian solusi pada *maze* yang telah dibangkitkan adalah sebagai berikut :

1. Pilih sel pertama kali secara acak pada *grid*, akan dikenali sebagai *current cell*.
2. Cek pada sel tetangga, apakah sel sudah pernah dikunjungi. Gunanya untuk menentukan *current cell* berikutnya. Karena ini adalah sel pertama maka belum ada sel yang dikunjungi. Data sel disimpan dalam *Stack Generate.*
3. Cara pengecekan yaitu dengan gerakan yang sudah ditentukan yaitu atas (x, y-1), bawah (x, y+1), kiri (x-1, y), dan kanan (x+1, y).
4. Proses ini dilakukan terus menerus sampai kondisi total sel = sel yang dikunjungi dan *current cell* = sel *finish*. Apabila ditemui sel yang sudah dikunjungi maka dilakukan proses *backtracking* yaitu dengan mengeluarkan data dari *Stack Generate* yang mana data tersebut mengarah ke sel yang telah dikunjungi kembali ke data di mana sel masih memiliki kemungkinan pengecekan sel tetangga lainnya.

Hasil proses ini akan menemukan solusi dari *maze* yang sudah dibangkitkan. Tabel 1 dan tabel 2 menunjukkan beberapa data tentang hasil percobaan yang dilakukan dengan membandingkan posisi *start* dan *goal* yang berbeda-beda. Tabel 1 merupakan hasil percobaan dimana posisi *start* dan *finish* ditentukan oleh pemakai, dan tabel 2 ditentukan secara acak oleh komputer. Percobaan dilakukan pada *grid* dengan ukuran 6x6. Tabel 1 dan tabel 2 seperti di bawah ini.

Tabel 1. Pencarian solusi dengan *Start* dan *Finish* ditentukan pemain

Start (baris , kolom)	Finish (baris , kolom)	Apakah Solusi Di temukan?	Jumlah Solusi
0,0	0,5	ya	1
0,1	0,4	ya	1
0,2	0,3	ya	1
0,3	0,2	ya	1
0,4	0,1	ya	1
0,5	0,0	ya	1

Tabel 2. Pencarian solusi dengan *Start* dan *Finish* ditentukan komputer

Start (baris , kolom)	Finish (baris , kolom)	Apakah Solusi Di temukan?	Jumlah Solusi
0,0	0,2	ya	1
0,5	0,1	ya	1
0,3	0,3	ya	1
0,0	0,5	ya	1
0,2	0,0	ya	1
0,5	0,0	ya	1
0,3	0,4	ya	1
0.1	0,4	ya	1
0,3	0,1	ya	1
0,5	0,4	ya	1

Dapat dilihat bahwa setiap *start* dan *finish* dipindah pada sel-sel yang berbeda akan selalu ditemukan solusi dan jumlah solusi hanya 1 pada setiap *start* dan *finish* yang dibuat. Dari hasil percobaan yang dilakukan sampai 10 kali, didapatkan hasil yang sama yaitu hanya ada 1 solusi dengan *start* dan *finish* yang berbeda-beda.

Dari hasil percobaan tersebut, apabila digabungkan dengan konsep pembangkitan *maze* dengan metode *backtracking* dapat disimpulkan :

1. Proses pembangkitan *maze* yang bentuk selnya akan saling terhubung, pasti akan menghasilkan solusi dengan aturan *start* dan *finish* harus

diletakan di bagian sisi kiri dan kanan papan permainan.
2. Hanya ada 1 solusi pada setiap satu kali pemilihan posisi *start* dan *finish*.

Kekurangan dari analisis ini adalah tidak melakukan analisa apakah hasil pencarian solusi dan jumlah solusi akan sama apabila *start* dan *finish* diletakan pada sisi atas dan bawah.

V. KESIMPULAN

Berdasarkan hasil penerapan algoritma *backtracking* pada permainan *Math Maze*, diperoleh bahwa algoritma ini dapat digunakan untuk menghasilkan *maze* yang tidak memiliki *loop* dan ruang tertutup atau terbuang. Terkait dengan solusi terhadap *maze* yang terbentuk, algoritma *backtracking* juga dapat menghasilkan solusi yang pasti ada pada setiap *problem* dan hanya 1 solusi untuk setiap *problem*nya.

REFERENSI

[1] Munir, Rinaldi. 2003. Matematika Diskrit Edisi Kedua. Bandung: Penerbit Informatika.

[2] http://rehulina.wordpress.com/2009/08/05/pengertian-kecerdasan-buatan/
Diakses pada tanggal 2 Mei 2015.

[3] http://www.dreamincode.net/forums/topic/266147-backtracking-using-this-pseudocode/
Diakses pada tanggal 3 Mei 2015.

[4] http://www.cse.unsw.edu.au/~billw/Justsearch.html
Diakses pada tanggal 3 Mei 2015.

[5] Bond, C.(1981)Maze generator. Diakses dari http://www.atarimagazines.com/compute
Diakses pada tanggal 4 Mei 2015.

[6] Feronato, E. (2008). Step by step perfect maze generation. Diakses dari http://www.emanueleferoanato.com/my-works/
Diakses pada tanggal 4 Mei 2015.

[7] Loy, J. (2002). Math maze. Diakses dari http://www.jimloy.com/puzz/
Diakses pada tanggal 4 Mei 2015.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 4 Mei 2015

Vincent Theophilus Ciputra
13513005