# PENERAPAN ALGORITMA BACKTRACKING PADA PERMAINAN MATH MAZE

Teneng, Joko Purwadi, Erick Kurniawan
Fakultas Teknik Program Studi Teknik Informatika
Universitas Kristen Duta Wacana Yogyakarta
Email: patmostos@yahoo.com, jokop@ukdw.ac.id, erick@ukdw.ac.id

**Abstrak :**

Permainan Math Maze merupakan game sederhana yang bertujuan menentukan jalur yang tepat untuk mencapai tujuan yang telah ditetapkan. Permainan ini hampir sama dengan permainan labirin (Maze) biasa. Perbedaannya adalah pada Math Maze kita harus menemukan jalur pada labirin dengan menggunakan angka-angka pada bagian sisi kiri dan sisi atas sebagai indikasi berapa banyak kotak yang dilewati untuk tiap baris atau tiap kolom dan tidak menggunakan tembok penghalang seperti pada labirin biasa. Pada penelitian ini akan dilakukan proses untuk membuat Papan Permainan Math Maze yang bisa dimainkan oleh pemakai dengan menerapkan algoritma *Backtracking*. Proses Pembuatan papan permainan Math maze itu terdiri dari beberapa langkah yaitu proses Pembangkitan sebuah maze, pencarian solusi dari maze yang sudah di bangkitkan, dan membuat papan permainan Math Maze baru dengan Memanfaatkan maze yang sudah di ketahui solusinya. Papan Permainan Math Maze dengan menerapkan algoritma backtracking dapat menghasilkan 1 solusi untuk setiap problem yang dibangkitkan dan maze yang dihasilkan dengan algoritma *backtracking* akan menghasilkan maze yang tidak memiliki loop dan ruang terbuang.

**Kata Kunci : *P**ermainan Math Maze, Algorima Backtracking***

## 1. Pendahuluan

Game komputer merupakan salah satu aplikasi software yang saat ini banyak dikembangkan. Dengan jenis yang bermacam-macam dan tampilan yang menarik, game komputer termasuk software yang diminati oleh berbagai kalangan. Selain karena tampilan dan aplikasinya relatif menarik, game komputer juga disinyalir 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 menentukan jalur yang tepat untuk mencapai tujuan yang telah ditetapkan. Permainan ini hampir sama dengan permainan labirin (maze) biasa. Perbedaannya adalah pada Math Maze kita harus

menemukan jalur pada labirin dengan menggunakan angka-angka pada bagian sisi kiri dan sisi atas sebagai indikasi berapa banyak kotak yang dilewati untuk tiap baris atau tiap kolom dan tidak menggunakan tembok penghalang seperti pada labirin biasa.

Penelitian ini akan fokus pada pembangunan papan permainan Math maze yang bisa ditentukan oleh pemakai baik dari penentuan ukuran papan permainan juga posisi pintu masuk dan pintu keluar dengan menerapkan metode Backtracking. Sebagai batasan dalam penelitian ini, ukuran *grid* papan yang dapat ditentukan oleh pemakai berjumlah 20 x 20.

## 2. Landasan Teori

### 2.1 Kecerdasan Buatan

Banyak sekali pendapat mengenai apa sebenarnya kecerdasan buatan itu. Pada intinya kecerdasan buatan adalah sebuah sistem yang dapat berpikir seperti manusia. Dengan definisi ini, kecerdasan buatan menawarkan baik media maupun uji teori kecerdasan. 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 pada permainan Math maze yang digunakan adalah algoritma backtracking.

### 2.2 Teori Algoritma Backtracking

Algoritma backtrack pertama kali diperkenalkan oleh D.H. Lehmer pada tahun1950. Dalam perkembangannya beberapa ahli seperti Rwalker, Golomb, dan Baumert menyajikan uraian umum tentang backtrack dan penerapannya dalam berbagai persoalan dan aplikasi.

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. Oleh karena algoritma ini berbasis pada algoritma Depth-First Search (DFS), maka pencarian solusi dilakukan dengan menelusuri suatu struktur berbentuk pohon berakar secara preorder. Algoritma DFS merupakan algoritma pencarian secara mendalam. Cara kerja algoritma ini adalah menelusuri salah satu kemungkinan yang ada hingga akar-akar maksimal, baru setelah itu menelusuri kemungkinan yang lain hingga akar-akar maksimal juga. Proses ini dicirikan dengan ekspansi simpul terdalam lebih dahulu sampai tidak ditemukan lagi suksesor dari suatu simpul.

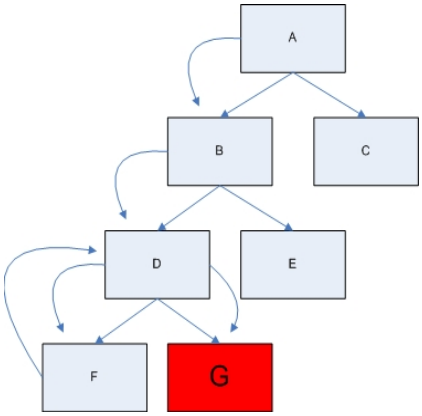
Prinsip dasar algoritma Backtracking adalah mencoba semua kemungkinan solusi yang ada. Perbedaan utamanya adalah pada konsep dasarnya,yaitu pada backtracking semua solusi dibuat dalam bentuk pohon solusi (tree), dan kemudian pohon tersebut

akan ditelusuri secara DFS (Depth First Search) sehingga ditemukan solusi terbaik yang diinginkan.

Seperti yang telah dijelaskan diatas 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 dari 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 child maka dilakukan backtracking ke simpul parent
4. Pencarian dihentikan jika kita telah menemukan solusi atau tidak ada simpul hidup yang dapat di diperlukan.




**Gambar 1**. Contoh algoritma backtracking

### 2.3 Depth First Search (DFS)

Algoritma DFS pertama kali diperkenalkan oleh Tarjan dan Hopcroft 20 tahun yang lalu. Mereka menunjukkan bagaimana DFS dapat digunakan untuk membangun sejumlah algoritma graph yang efisien. Metode Depth First Search (DFS) merupakan metode pencarian secara mendalam. Cara kerjanya adalah menelusuri salah satu kemungkinan yang ada hingga akar-akar yang maksimal, baru setelah itu menelusuri kemungkinan yang lain hingga akar-akar maksimal juga.

Perlu diperhatikan bahwa kebutuhan waktu dalam proses pencarian dengan DFS, sejalan dengan kedalaman maksimum pohon pencarian. Jika kedalaman pohon tidak terbatas, maka dapat dimungkinkan bahwa algoritma tidak akan berhenti. Hal ini dapat terjadi jika ruang

pencarian tidak terbatas, atau jika ruang pencarian mengandung siklus keadaan. Dengan demikian algoritma DFS tidak menunjukkan sifat komplit, hanya sebagian ruang pencarian yang ditempuh.

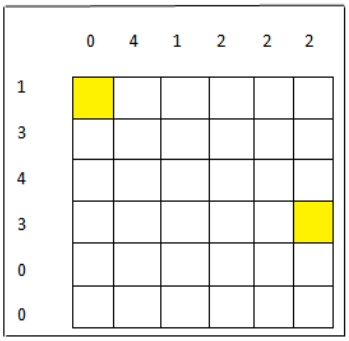
### 2.4 Math Maze

Permainan Math Maze hanya dimainkan oleh 1 orang. Permainan Math Maze merupakan game sederhana yang bertujuan menentukan jalur yang tepat untuk mencapai tujuan yang telah ditetapkan. Pada dasarnya Math Maze adalah permainan yang merupakan perkembangan dari permainan Maze (labirin). 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, kanan, kiri dan bawah. 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 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 dari sel Start atau Pintu Masuk. Pemain harus memperkirakan jalur solusi yang terbentuk nanti harus sesuai dengan nilai- nilai pemandu. Apabila jumlah nilai baris dan kolom yang dilewati tidak sesuai sampai mencapai garis selesai maka dianggap gagal menemukan jalur yang tepat menuju pintu keluar atau selesai. Berikut ini akan dijelaskan proses permainan Math maze.

Pada awal permainan akan disediakan papan permainan berupa grid atau sel-sel dengan ukuran n x m . Di antara sel tersebut terdapat pintu masuk dan pintu keluar.




**Gambar. 2**. Papan permainan Math Maze ukuran 6x6

**3. Penerapan Metode**

Sistem yang akan dibuat dengan penerapan algoritma *backtracking* bukan untuk mencari solusi pada Papan permainan Math maze yang sudah siap, tapi penerapan yang dilakukan adalah proses pembuatan atau pembangkitan papan permainan Math Maze itu sendiri. Proses Pembuatan papan permainan Math maze itu terdiri dari beberapa langkah yaitu: proses Pembangkitan sebuah maze, pencarian solusi dari Maze yang sudah di bangkitkan, dan membuat papan permainan Math Maze baru dengan memanfaatkan Maze yang sudah di ketahui solusinya.

**3.1 Pembangkitan Maze**

Proses Pembangkitan sebuah maze yang dilakukan dengan *backtracking* adalah dengan prosedur berikut.

- Pilih satu sel pada sebuah grid
- Pilih secara acak sel terdekat atau sel tetangga dengan gerakan yang di perbolehkan adalah atas, bawah, kiri, kanan. Tidak diperbolehkan memilih sel tetangga secara diagonal.
- Jika sel tetangga tersebut belum pernah kita datangi, maka pindah ke sel tersebut dan hapus pembatas atau *wall* antara sel tetangga tersebut dengan sel yang sebelumnya di tempati.
- Jika Tidak ditemukan sel tetangga yang belum didatangi maka kita harus melakukan *backtracking* ke sel sebelumnya.
- Langkah ini dilakukan sampai semua sel pada grid sudah didatangi.
- Tentukan posisi *Start* dan *Finish*.

**3.2 Pencarian solusi dari Maze yang sudah dibangkitkan.**

Apabila *maze* sudah di bangkitkan maka selanjutnya dalah mencari solusinya. Langkah-langkahnya adalah sebagai berikut

- Dari sel Start, pilih secara acak sel terdekat yang bisa di kunjungi. Sama seperti proses pembangkitan *maze*, hanya ada 4 gerakan yang diperbolehkan yaitu atas, bawah, kiri dan kanan.
- Apabila Sel belum pernah di datangi maka pindah ke sel tersebut. Pilih lagi sel tetangga secara acak.

- Apabila Sel menemui tembok atau *wall*, maka lakukan *backtracking* ke sel sebelumnya.
- Lakukan langkah-langkah ini sampai menemukan sel yang menjadi selesai atau pintu keluar.

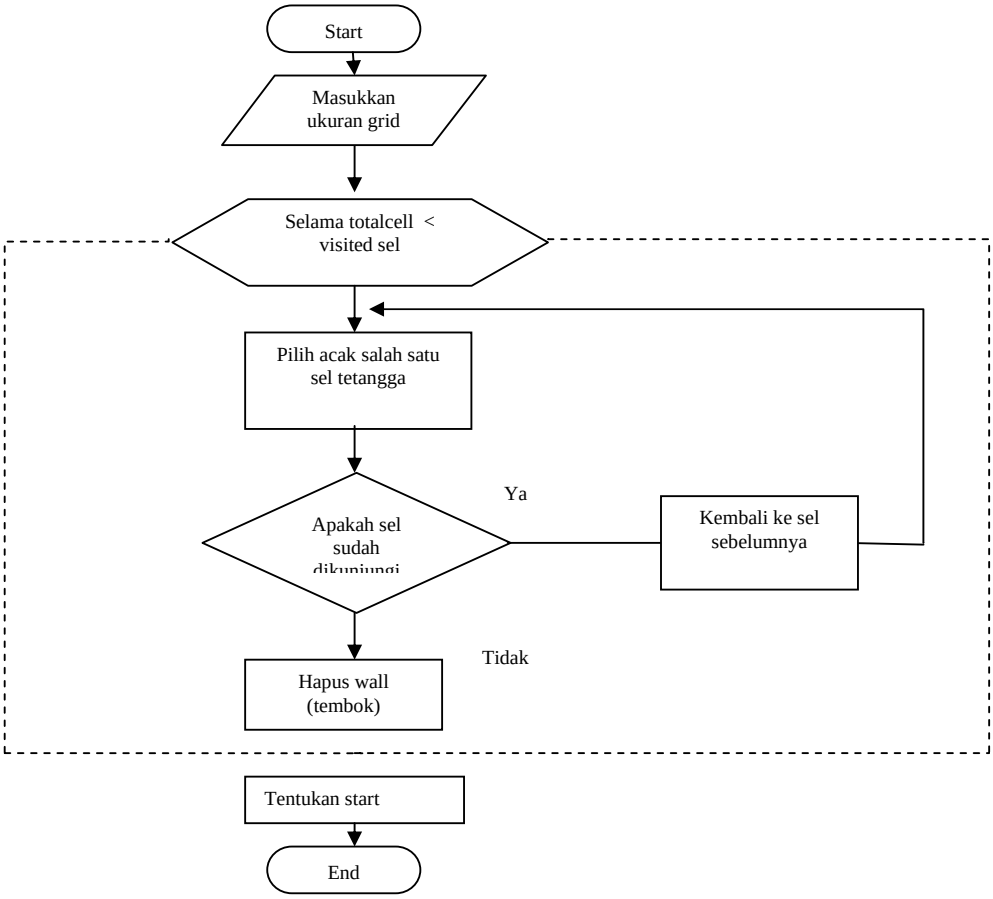
**3.3 Pembuatan papan permainan Math Maze**

Proses ini adalah proses untuk membuat papan permainan Math Maze dengan memanfaatkan *maze* yang sudah dibangkitkan dan solusi yang sudah di temukan dari *maze* tersebut. Langkah-langkahnya adalah menghitung jumlah total sel pada baris dan kolom yang di lewati 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 pintu masuk dan pintu keluar serta nilai-nilai pada baris dan kolom sebagai panduan untuk menemukan jalur solusi. Dengan demikian papan permainan sudah dapat dimainkan oleh pemakai.

**3.4 Algoritma Program**

Langkah pertama dalam Proses pembangkitan ini dilakukan dengan Algoritma *Backtracking* untuk membentuk sebuah *Maze*. Mulai

a. Masukkan ukuran *grid* ( i x j)
b. Selama Total sel pada *grid* < sel yang telah di kunjungi lakukan langkah d, e, dan f.
c. Pilih secara acak sel tetangga (kanan, kiri , atas, bawah)
d. Cek apakah sel sudah pernah dikunjungi, jika ya kembali ke langkah d, jika tidak lakukan langkah f.
e. Hapus *wall* atau pembatas pada sel
f. Tentukan pintu masuk (start) dan goal (pintu keluar)
g. Selesai




**Gambar 3**. Flowchart pembangkitan maze.

Setelah *maze* terbentuk dan pintu masuk dan pintu keluar 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 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 misalnya bergerak secara 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 mencari solusi. Langkah terakhir adalah menghilangkan tampilan *maze* sempurna dan solusinya sehingga papan permainan hanya menampilkan *grid* dengan sel start dan sasaran serta angka-angka pada sisi kiri dan atas papan permainan.

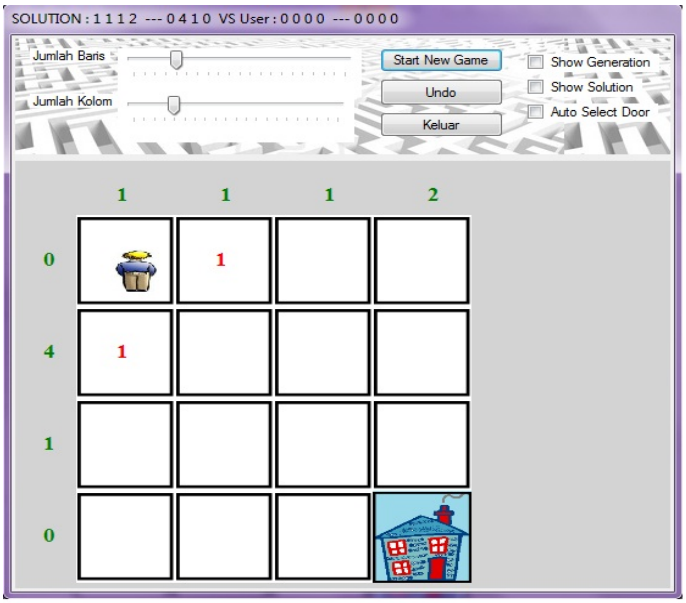
Algoritma dari program tersebut adalah sebagai berikut :

a. Mulai
b. Buat Stack generate untuk sturktur data
c. Set totalcell = jumlah sel pada grid
d. Pilih satu sel secara acak sebagai current sel
e. Set visited sel=1
f. Selama Visited sel < totalcell lakukan langkah g sampai i
g. Cek sel tetangga dari current sel, jika lebih dari 1 lakukan langkah h sampai k, jika =1 lakukan langkah l
h. pilih salah satu secara acak.
i. Simpan sebagai visited sel
j. Masukan (push) data visited sel ke stack generate.
k. Jadikan sel tetangga tersebut menjadi current sel yang baru
l. Selesai

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

## 4. Implementasi dan Pembahasan

### 4.1 Form Permainan Math Maze




**Gambar 4.** Tampilan Form Permainan Math Maze

Pada Form permainan Math maze (Gambar 4), pemain harus menemukan jalur dari sel

START sampai ke Sel Finish yang diwakili oleh gambar rumah. Beberapa komponen yang ada di dalam form Permainan Math Maze adalah :

a. Trackbar Baris

   Trackbar baris ini merupakan komponen yang menerima masukan dari pemakai berupa jumlah baris papan permainan yang ingin dimainkan oleh pemakai. Jumlah baris diwakili dengan turus-turus yang ada pada track bar dengan jumlah maksimal 20.

b. Trackbar Kolom

   Trackbar kolom ini merupakan komponen yang menerima masukan dari pemakai berupa jumlah kolom papan permainan yang ingin dimainkan oleh pemakai. Jumlah kolom diwakili dengan turus-turus yang ada tpada track bar dengan jumlah maksimal 20.

c. Tombol Start New Game

   Tombol ini akan membuat papan permainan yang baru sesuai dengan masukan pemakai yaitu jumlah baris dan kolom yang diinginkan.

d. Tombol Undo

   Tombol Undo ini adalah tombol untuk mundur apabila pada saat pemakai salah menentukan langkah yang dipilih maka dengan mengklik tombol Undo akan menghapus langkah tersebut dan kembali ke langkah sebelumnya.

e. Checkbox Show Generation

   Apabila checkbox ini di centang maka pada saat pemakai mengklik tombol Start New Game maka akan diperlihatkan proses sistem saat membangkitkan papan permainan.

**4.2 Algoritma Pembangkitan Maze**

Algoritma *Backtracking* adalah algoritma yang berbasis pada metode Depth first search (DFS). Metode ini lah yang di terapkan 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 di pilih 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 di mana semua sel sudah dikunjungi, pilih secara acak salah satu titik pada jalur yang sudah dilewati dan kembali lagi ke titik tersebut, lalu cek apakah ada sel yang belum dilewati.
6. Ulangi langkah 3,4,5 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.

### 4.3 Algoritma Pencarian Solusi pada Maze yang Dibangkitkan

Algoritma yang diterapkan untuk pencarian solusi pada Maze yang terlah dibangkitkan adalah sebagai berikut :

1. Pilih sel pertama kali secara acak pada *grid*, akan di kenali sebagai *current* sel (ptCurrent).
2. Cek pada sel tetangga, apakah sel sudah pernah dikunjungi. Gunanya untuk menentukan *current* sel berikutnya. Karena ini adalah sel pertama maka belum ada sel yang dikunjungi. Maka data sel disimpan di Stack generate.
3. Cara pengecekan yaitu dengan gerakan yang sudah ditentukan yaitu atas ( x, y-1 ), kanan ( x+1, y ), bawah ( x, y+1 ), kiri ( x-1, y ) .
4. Proses ini dilakukan terus menerus sampai kondisi total sel = sel yang dikunjungi dan ptCurrent sel = ptEnd sel. Apabila ditemui sel yang sudah di kunjungi maka dilakukan proses *backtracking* yaitu dengan mengeluarkan data dari *stack generate*d 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 di bangkitkan. 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 di mana posisi start dan finish ditentukan oleh pemakai, dan Tabel 2 di tentukan secara acak oleh Komputer. Percobaan dilakukan pada ukuran *grid* 6x6.

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

| 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 di temukan solusi dan jumlah solusi hanya 1 pada setiap start dan *finish* yang di buat. 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 di gabungkan dengan konsep pembangkitan maze dengan metode *Backtracking* dapat di simpulkan :

- 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.
- 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 dan jumlah solusi akan sama apabila Start dan Finish diletakan pada sisi atas dan bawah.

## 5. Kesimpulan

Berdasar hasil penerapan algoritma Backtracking pada permainan Math Maze dalam penelitian ini, diperoleh bahwa algoritma Backtracking dapat digunakan untuk menghasilkan maze yang tidak memiliki loop dan ruang yang 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 problemnya.

**Daftar Pustaka**


[1] Bond, C. (1981) Maze generator. Diakses dari http://www.atarimagazines.com/compute.

[2] Cornell, G., Morrison, J.. (2002). Programming VB.NET: Guide for experienced programmers. A! Press, New York.

[3] Feronato, E. (2008). Step by step perfect maze generation. Diakses dari http://www.emanueleferoanato.com/my-works/

[4] Halvorson, M. (2008). Microsoft visual basic 2008. Microsoft Press, Washington.

[5] Loy, J. (2002). Math maze. Diakses dari http://www.jimloy.com/puzz/

[6] Jones, M. T. (2003). AI aplication programming**.** Charles River Media, Inc, Massachusetts.

[7] Santoso, H. (2005). Membangun aplikasi .NET menggunakan VB.NET 2005. PT Elex Media Komputindo, Jakarta.

[8] Setiawan, S. (1993). Artificial intelligence. Andi Offset, Yogyakarta.

[9] Suparman. (1991). Mengenal artificial Intelligence. Andi Offset, Yogyakarta.