Penggunaan Algoritma *Backtracking* pada Permainan *Mummy Maze*

Deddy Wahyudi

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jl. Ganesha 10, Bandung

Email : if15031@students.if.itb.ac.id

Abstrak

Permainan Mummy Maze adalah suatu permainan buatan GameHouse™ yang cukup menarik untuk dibahas, karena untuk menyelesaikannya kita dapat menerapkan beberapa algoritma yang populer seperti *Backtracking*, *Branch and Bou*nd dan Program dinamis. Dalam makalah ini, akan dibahas penyelesaian permainan *Mummy M*aze dengan algoritma *Backtracking*. Algoritma *Backtracking* digunakan untuk mencari solusi dari *maze*, dimana seorang pemain harus mencapai pintu keluar tanpa tertangkap oleh *Mummy* yang bergerak 2x kecepatannya.

Kata Kunci : *Mummy M*aze, *Backtracking*, *Branch and Bound*, Program Dinamis

1. Pendahuluan

Dewasa ini, perkembangan dunia *game* semakin luas dan cepat. Hal ini ditandai dengan bermunculannya banyak *game developer* berskala besar. Sebagian besar dari *developer* ini mementingkan aspek grafis pada permainan yang dibuatnya, sehingga terkadang aspek pemikiran dan latihan *problem solving* sering terlupakan. GameHouse™, merupakan salah satu *game develo*per yang tidak melupakan aspek tersebut. Salah satu produk GameHouse™ yang cukup menarik untuk dibahas adalah permainan *Mummy Maze*.

Sementara, *backtracking* adalah algoritma yang berbasis pada DFS untuk mencari solusi persoalan secara lebih mangkus. *backtracking*, yang merupakan perbaikan dari algoritma *brute-force*, secara sistematis mencari solusi persoalan di antara semua kemungkinan solusi yang ada.

2. Persoalan

Dalam *Mummy Maze*, pemain akan menggerakkan seorang penjelajah makam (selanjutnya disebut *explorer*) yang berada dalam sebuah pyramid. Tugas si *explorer* adalah untuk mendapatkan harta tersembunyi yang terdapat pada puncak *pyramid*. Namun pembuat *pyramid* telah menempatkan *Mummy-Mummy* penjaga untuk menjaga harta tersebut.

Dalam kegelapan *Pyramid*, *Mummy* dapat bergerak 2x lebih cepat dari *explorer*. Namun *Mummy-Mummy* ini tidak pintar, *Mummy* hanya akan mengambil jalur langsung untuk menangkap *explorer*, dan gerakannya memiliki pola, yaitu “Sesuaikan posisi horizontal terlebih dahulu, baru posisi vertikal”. Pola inilah yang harus dimanfaatkan *explorer* untuk lolos dari hadangan *Mummy*.

Dalam permainan sebenarnya, terdapat dua jenis *Mummy*, yaitu *Mummy* putih dengan spesifikasi seperti diatas, dan *Mummy* merah dengan sifat yang berkebalikan dengan *Mummy* putih, yaitu memiliki pola gerakan “Sesuaikan posisi vertikal terlebih dahulu, baru posisi horizontal”. Namun untuk penyederhanaan masalah, maka disini hanya akan membahas penyelesaian masalah dengan *Mummy* putih sebagai penghadang. Berikut dilampirkan gambar tentang bagaimana *Mummy* bergerak dalam permainan *Mummy Maze*.

Dalam permainan, juga terdapat beberapa objek yang dapat berinteraksi dengan explorer, yaitu *key*, *gate*, *trap* dan *ankh*. *Key* merupakan sebuah kunci emas yang dapat digunakan oleh *explorer* untuk membuka dan menutup *gate*. Sementara *trap* adalah tempat yang tidak boleh diinjak oleh *explorer*, karena terdapat jebakan yang dapat langsung membunuh *explorer*. Namun hal ini tidak berlaku untuk *Mummy*.

Ankh merupakan suatu indikator apakah suatu maze masih dapat diselesaikan atau tidak. Jika warna *ankh* berubah menjadi merah, maka berarti maze sudah tidak dapat diselesaikan dengan posisi explorer dan *mummy* saat itu. Jika hal ini terjadi, maka user harus melakukan *undo*, atau *reset*.

Key	
Gate Closed	
Gate Opened	
Trap	
Ankh normal	
Ankh red	

Untuk mendapatkan gambaran yang lebih jelas tentang pergerakan *Mummy* terhadap *explorer*, perhatikan gambar dibawah ini

Gambar 1. Posisi awal Explorer sebagai berikut

Gambar 2. Explorer melangkah ke bawah

Dari kedua gambar diatas, dapat kita lihat bahwa *Mummy* bergerak dua langkah dengan berusaha menyesuaikan posisi horizontal terlebih dahulu, jika tidak menyesuaikan posisi horizontal tidak dapat dilakukan (sudah sesuai atau terhalang tembok), dan langkah masih tersisa, maka sesuaikan posisi vertikal.

3. Penyelesaian Masalah *Mummy Maze*

Penyelesaian masalah *Mummy* Maze ini sebetulnya dapat lebih mudah dimengerti jika permasalahan disederhanakan sebagai berikut. Pada setiap langkahnya, *explorer* memiliki lima pilihan untuk bergerak, yaitu atas, bawah, kiri, kanan atau diam sama sekali. Pilihan langkah ini dapat kita asumsikan sebagai suatu nilai integer yang nantinya akan dikumpulkan menjadi suatu himpunan solusi. Untuk perjanjian dalam makalah ini, maka diasumsikan :
1 : atas
2 : kanan
3 : bawah
4 : kiri
5 : diam
Sehingga solusi persoalan dapat direpresentasikan dengan *array of integer* yang berbentuk :

$$\{X_1, X_2, X_3, \ldots, X_n\}$$

Dimana setiap variabel *X* dapat berisi angka [1..5], dan *n* menyatakan jumlah langkah yang dibutuhkan untuk mencapai solusi. Dalam kasus ini nilai dari n tidak menentu, karena tidak terdapat jumlah langkah yang pasti untuk mencapai solusi (*explorer* mencapai pintu keluar dari *maze*). *Explorer* bisa saja memilih untuk berputar-putar dan mengulangi langkah yang sama, atau langsung menuju solusi, sehingga jumlah anggota dari himpunan solusi menjadi bervariasi. Untuk memperjelas konsep ini, berikut akan diberikan sebuah contoh kasus.

Gambar 3. Contoh kasus

Salah satu penyelesaian untuk maze diatas yaitu:

{1,1,4,4,1,2,2,2,1,1,4,4,3,4,4,4,2,2,2,1,2,2,3,3,
4,4,4,3,2,2,2,1,1,5,3,3,3,4,3,4,1,4,4,4}

Namun solusi diatas bukanlah satu-satunya solusi yang mungkin untuk menyelesaikan maze tersebut. Terdapat beberapa solusi lain, seperti himpunan dibawah ini :

{1,1,4,4,5,1,2,5,2,2,1,5,1,4,4,3,4,5,4,4,2,2,2,1,
2,2,3,3,5,4,4,4,3,2,2,2,1,1,3,3,3,3,5,4,3,4,1,4,4,
4}

Dan masih banyak lagi solusi yang mungkin untuk persoalan diatas.

Pengimplementasian algoritma *Backtracking* pada masalah *Mummy Maze* yaitu dengan mencoba semua kemungkinan langkah secara DFS, jika solusi dinyatakan tidak dapat dicapai, maka tekan tombol *undo* sampai solusi mungkin ditemukan kembali, lalu ulangi pencarian pada langkah-langkah yang belum dicoba.

Secara eksplisit, komponen algoritma *Backtracking* pada *Mummy Maze* dapat dinyatakan sebagai berikut :

1. **Solusi persoalan**
   Solusi persoalan pada masalah ini direpresentasikan dengan sebuah array of integer yang berjumlah n anggota.
   $\{X_1, X_2, X_3, \ldots, X_n\}$

2. **Fungsi pembangkit**
   Fungsi pembangkit pada masalah *Mummy Maze* akan membangkitkan nilai [1..5] yang merepresentasikan gerakan explorer.

3. **Fungsi pembatas**
   Fungsi pembatas akan bekerja jika langkah yang dilakukan *explorer* salah, sehingga membawanya tidak bisa lagi menyelesaikan *maze*. Fungsi pembatas pada masalah ini dapat diidentifikasi dengan mudah karena terdapat fitur *ankh*. Saat solusi tidak dapat dicapai lagi, maka warna ankh akan berubah menjadi merah. Pada saat ini, user harus melakukan *backtrack* dengan menekan tombol *undo* hingga *ankh* kembali berwarna kuning, lalu mencoba langkah lain yang belum pernah dicoba sebelumnya.

Berikut ini akan ditampilkan pseudo-code dari algoritma *Backtracking* yang diimplementasikan untuk mempermudah pemahaman :

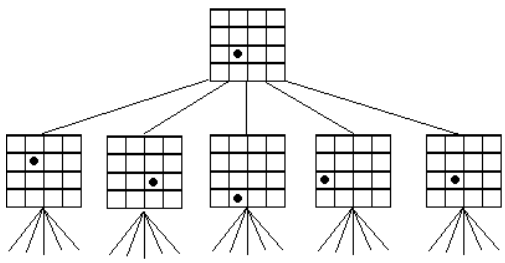
```
procedure backtrack (i :
integer)
{mencari solusi persoalan
dengan algoritma backtracking
skema rekursif}
for tiap X[i] yang belum dicoba
sedemikian sehingga
      ((X[k] ← T(k)) and
T(x[1], x[2], …, x[i]) = true)
do
      if (X[1], X[2], …, X[i])
adalah lintasan dari akar ke
daun then
            cetakSolusi(X)
      endif
      backtrack(i+1)
endfor
```

4. Analisis

Secara sederhana, dapat disebutkan bahwa penyelesaian permainan Mummy Maze dengan Backtracking dinyatakan sebagai berikut :

1. Jika explorer berada pada gerbang, maka solusi telah ditemukan
2. Jika tidak, maka coba suatu langkah yang belum pernah dicoba sebelumnya
3. Jika explorer berada pada gerbang, maka solusi ditemukan, jika tidak maka ada 2 pilihan
4. a. jika posisi explorer terhadap Mummy masih aman (ankh masih berwarna kuning), maka kembali ke langkah 2.
   b. jika posisi explorer sudah tidak bisa lari dari Mummt (ankh berubah warna menjadi merah), maka lakukan backtrack (dengan menekan tombol undo), lalu kembali ke langkah 2.

Dengan langkah-langkah seperti diatas, maka solusi dari permainan Mummy Maze pasti akan ditemukan, mengingat developer game sudah memastikan bahwa setiap maze pasti memiliki solusi.
Solusi yang diciptakan dapat dinyatakan sebagai pohon 5r.



Penyelesaian permainan Mummy Maze menggunakan Backtracking dapat dikatakan cukup mangkus, mengingat kompleksitas algoritmanya bernilai O(n).

5. Kesimpulan

Penerapan algoritma *Backtracking* pada permainan *Mummy maze* memang cocok untuk diterapkan dibandingkan algoritma lainnya, mengingat domain persoalan yang tidak terlalu luas, dan jumlah solusi yang relatif dapat diingat bahkan secara visual sekalipun. Namun pada prakteknya, mencoba seluruh kemungkinan dengan algoritma *Backtracking* sekalipun tidak terlalu efektif, karena pada dasarnya permainan ini dibuat untuk diselesaikan dengan pola dan logika pemain, sehingga pergerakan yang didasarkan atas perhitungan visual yang cermat dan logika terbukti dapat menyelesaikan masalah dengan lebih cepat. Namun tentu algoritma *Backtracking* ini cocok untuk diterapkan pada domain persoalan yang lebih besar, misalnya jika ukuran maze adalah 100 x 100, dimana membuat suatu pola dengan daya ingat visual menjadi sulit.

6. Referensi

Munir, Rinaldi. 2007. Strategi Algoritmik. Teknik Informatika ITB : Bandung