PENGGUNAAN ALGORITMA RUNUT BALIK DALAM PENCARIAN SOLUSI *THREE DIMENSIONAL MAZE*

Ferdian Thung (13507127)

Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Jalan Ganesha No. 10, Bandung 40132
e-mail: if17127@students.if.itb.ac.id

ABSTRAK

Makalah ini membahas tentang pencarian solusi dalam *Three Dimensional Maze*. Berbeda dengan maze yang ditemui pada umumnya, *Three Dimensional Maze* merupakan sebuah maze dengan dimensi tiga di mana ia memiliki enam arah penelusuran, yakni utara, timur, selatan, barat, atas, dan bawah. Algoritma runut balik digunakan untuk mencari solusi *Three Dimensional Maze* di mana langkah demi langkah ditelusuri dan jika terdapat jalan buntu maka proses runut balik dilakukan guna mencari kemungkinan solusi yang lain. Dengan ini, pada akhirnya solusi akan ditemukan atau diperoleh fakta bahwa maze yang ditelusuri tidak mempunyai solusi, yakni ketika pencarian kembali pada titik awal.

Kata kunci: algoritma *backtracking*, *three dimensional maze*.

1. PENDAHULUAN

1.1 Algoritma Runut Balik (*Backtracking*)

Runut-balik (*backtracking*) adalah algoritma yang berbasis DFS (*Depth First Search*) 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. Dengan metode ini, kita tidak perlu memeriksa semua kemungkinan solusi yang ada, hanya pencarian yang mengarah pada solusi saja yang dipertimbangkan sehingga waktu pencarian dapat diminimalisir. *Backtracking* secara harfiah dinyatakan dalam algoritma rekursif. Kadang-kadang disebutkan bahwa runut-balik merupakan bentuk tipikal dari algoritma rekursif.

Saat ini algoritma runut-balik banyak diterapkan untuk program *games* (seperti permainan tic-tac-toe, pencarian jalan, catur, dll.) dan masalah-masalah pada bidang kecerdasan buatan (*artificial intelligence*). Untuk menerapkan metode runut-balik, properti berikut didefinikan: [1]

1. Solusi persoalan
   Solusi dinyatakan sebagai vektor dengan n-tuple:
   X = (x1, x2, …, xn), xi ∈ himpunan berhingga Si
   Mungkin saja S1 = S2 = … = Sn
   Contoh: Si = {0, 1}, xi = 0 atau 1
2. Fungsi pembangkit nilai xk
   Dinyatakan sebagai T(k)
   T(k) membangkitkan nilai untuk xk, yang merupakan komponen vektor solusi.
3. Fungsi pembatas (pada beberapa persoalan fungsi ini dinamakan fungsi kriteria)
   Dinyatakan sebagai B(x1,x2,…,xk)
   Fungsi pembatas menentukan apakah (x1, x2, …, xk) mengarah ke solusi. Jika mengarah, maka pembangkitan nilai untuk xk+1 dilanjutkan, tetapi jika tidak, maka (x1, x2, …, xk) dibuang dan tidak dipertimbangkan lagi dalam pencarian solusi.

Langkah-langkah pencarian solusi adalah sebagai berikut:

1. Solusi dicari dengan membentuk lintasan dari akar sampai daun. Aturan pembentukan mengikuti metode DFS. Simpul-simpul yang sudah dilahirkan dinamakan simpul hidup. Simpul hidup yang sedang diperluas dinamakan simpul-E. Simpul dinomori dari atas ke bawah sesuai dengan urutan kelahiran.
2. Tiap kali simpul-E diperluas, lintasan yang dibangun olehnya bertambah panjang. Jika lintasan yang sedangdibentuk tidak mengarah ke solusi, maka simpul-E tersebut dibunuh sehingga menjadi simpul mati. Fungsi yang digunakan untuk membunuh simpul-E adalah dengan menerapkan fungsi pembatas. Simpul yang sudah mati tidak akan pernah diperluas lagi.
3. Jika pembentukan lintasan berakhir dengan simpul mati, maka proses pencarian diteruskan dengan membangkitkan simpul anak yang lainnya. Bila tidak ada lagi simpul anak yang dapat dibangkitkan, maka pencarian solusi dilanjutkan dengan melakukan runut-balik ke simpul hidup terdekat. Selanjutnya simpul ini menjadi simpul-E yang baru. Lintasan baru dibangun kembali sampai lintasan tersebut membentuk solusi.
4. Pencarian dihentikan bila kita telah menemukan solusi atau tidak ada lagi simpul hidup untuk runut-balik.

1.2 *Three Dimensional Maze*

Sebelum berbicara mengenai *three dimensional maze*, kita perlu mengetahui karakteristik maze secara umum. Maze pada umumnya dapat dibagi dalam tujuh klasifikasi:[2]

1. Dimensi

Pada dasarnya merupakan banyaknya dimensi ruang yang ditangani maze. Dimensi dibagi menjadi:

- 2D, dua dimensi, memiliki empat arah gerak yakni utara, timur, selatan, dan barat.
- 3D, tiga dimensi, memiliki tambahan dua arah yakni atas dan bawah.
- Dimensi yang lebih tinggi, yakni dimensi di atas tiga.
- *Weave*, pada dasarnya merupakan maze 2D, tetapi memungkinkan bagian-bagiannya tumpang tindih satu sama lain.

2. *Hyperdimension*

Merupakan dimensi dari obyek yang bergerak di dalam maze. *Hyperdimension* dibagi menjadi:

- *Non-hypermaze*, di mana dimensi obyek berupa titik.
- *Hypermaze*, di mana dimensi obyek berupa garis.
- *Hyperhypermaze*, di mana dimensi obyek berupa bidang.

3. Topologi

Menggambarkan geometri ruang maze. Topologi dibagi menjadi:

- *Normal*, merupakan maze dalam ruang Euclides.
- *Planair*, mengacu pada maze dengan topologi abnormal, misalnya maze pada permukaan kubus.

4. *Tesselation*

Menggambarkan geometri dari sel-sel individual pembentuk maze. *Tesselation* dibagi menjadi:

- *Ortogonal*, di mana sel individual berbentuk persegi.
- *Delta*, di mana sel individual berbentuk segitiga.
- *Sigma*, di mana sel individual berbentuk heksagonal.
- *Theta*, terbentuk dari lingkaran konsentris di mana awal atau akhir berada di tengah lingkaran.
- *Upsilon*, di mana sel individual berbentuk oktagon.
- *Zeta*, pada dasarnya persegi tetapi membolehkan jalur dengan sudut 45 derajat.
- *Omega*, mengacu pada semua *non-orthogonal tessellation* yang konsisten.
- *Crack*, merupakan maze amorf tanpa *tessellation* yang konsisten.
- *Fractal*, yakni maze yang terbentuk dari maze yang lebih kecil.

5. *Routing*

Menggambarkan jalur dalam maze. *Routing* dibagi menjadi:

- *Perfect*, yakni maze tanpa loop dan daerah yang tidak dapat diakses.
- *Braid*, yakni maze tanpa jalan buntu.
- *Unicursal*, yakni maze tanpa persimpangan.
- *Sparseness*, yakni maze dengan daerah yang tidak dapat diakses.
- *Partial Braid*, yakni maze dengan loop dan jalan buntu di dalamnya.

6. Tekstur

Menggambarkan pola jalur dalam maze. Tekstur dibagi menjadi:

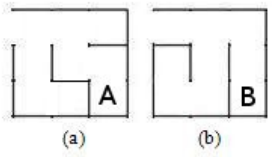
- *Bias*, yakni maze cenderung mengarah pada satu arah dibanding arah lain.
- *Run*,yakni seberapa jauh jalur cenderung lurus sebelum berbalik arah.
- *Elite*, yakni panjang solusi dibandingkan dengan ukuran maze.
- *Symmetric*, yakni memiliki jalur simetrik.
- *River*, yakni maze yang tampak mengalir.

7. Fokus

Menunjukkan bahwa pembuatan maze dapat dipandang dengan berbagai cara. Fokus dibagi menjadi:

- *Wall adders*, yakni pembuatan maze dipandang sebagai pembuatan tembok.
- *Passage carvers*, yakni pembuatan maze dipandang sebagai pembuatan jalur.
- *Template*, yakni pembuatan maze dipandang sebagai gabungan *wall adders* dan *passage carvers*.

Berdasarkan karakteristik di atas, pada makalah ini yang dimaksud dengan *three dimensional maze* adalah maze dengan dimensi tiga, *non-hypermaze*, topologi normal, *orthogonal tessellation*, dan *partial braid routing*. Sementara itu, tekstur dan fokus dapat berupa tipe yang mana pun. *Three Dimensional Maze* dapat pula dipandang sebagai maze dua dimensi yang bertingkat sehingga memiliki bagian atas atau bawah.



Gambar 1 *Three Dimensional Maze* dua tingkat dengan (a) tingkat 1 dan (b) tingkat 2

Gambar di atas merepresentasikan *Three Dimensional Maze* dua tingkat dengan tanda A berarti ada jalan ke atas dan B ada jalan ke bawah. Adapun tanda AB dapat digunakan untuk menunjukkan keberadaan jalan ke atas dan ke bawah.

2. METODE

2.1 Algoritma Penelusuran Maze

Algoritma penelusuran maze dengan *backtracking* pada umumnya memiliki dua solusi masalah: pertama, menyimpan semua langkah yang dilalui, kedua, menggunakan teknik rekursi (yang secara implisit menyimpan semua langkah). Rekursi adalah solusi yang lebih mudah.

Pada *Three Dimensional Maze*, hal yang dialami tidaklah berbeda. Hanya saja di sini diperlukan penelusuran dengan tambahan dua arah, yakni ke atas dan ke bawah. Berikut *pseudocode* untuk algoritma penelusuran *Three Dimensional Maze* yang dibuat secara rekursif.

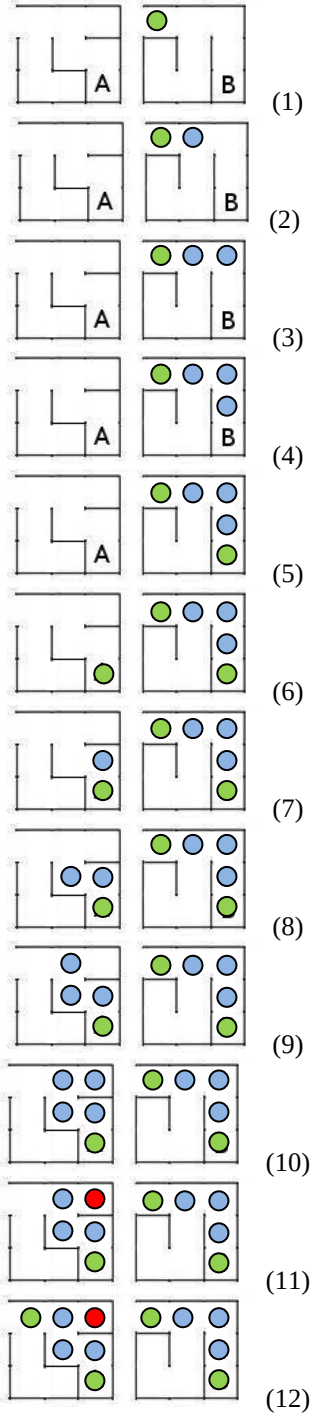
```
function Solve3DMaze(input M:3DMaze)
{mengembalikan true jika solusi ditemukan, false
jika tidak; M terdefinisi }

Kamus lokal
arah : integer {utara = 1, timur = 2, selatan =
3, barat = 4, atas = 5, bawah = 6}

Algoritma
begin
  if solusi telah ditemukan then
    return true
  else
    for semua kemungkinan arah penelusuran do
      Move(M,arah)
      if (Solve3DMaze(M)) then
        return true
      else
        unMove(M,arah)
      endif
    endfor
    return false
end
```

Berdasarkan *pseudocode* di atas, representasi 3DMaze dapat berupa tipe bentukan dari koordinat keluar maze dan array tiga dimensi yang tiap-tiap indeksnya menyatakan koordinat pada ruang tiga dimensi. Nilai yang disimpan dalam array tiga dimensi dapat berupa boolean 0 yang menyatakan dinding dan 1 yang menyatakan jalur sehingga penggunaan memori efektif dan efisien. Pada kasus tersebut, solusi ditemukan ketika koordinat obyek yang menelusuri 3DMaze sama dengan koordinat keluar maze.

Perlu diketahui bahwa algoritma di atas tidak secara eksplisit menyimpan jalur yang menjadi solusi. Jika ingin dilakukan penyimpanan jalur solusi, hal ini dapat dilakukan dengan mudah melalui penambahan stack koordinat tiga dimensi ke dalam kode. Isi stack ini ditambah setiap kali dilakukan pergerakan dan dikurangi ketika terjadi pembatalan gerakan. Dengan begitu, pada akhir pemanggillan fungsi, stack akan menyimpan jalur solusi yang terbentuk jika solusi berhasil ditemukan.

2.2 Pengujian Algoritma Penelusuran Maze

Untuk pembuktian kebenaran algoritma penelusuran maze yang digunakan, akan dilakukan pengujian dengan contoh kasus diambil dari gambar 1. Jalan masuk diambil dari tingkat 2. Berikut merupakan proses yang terjadi selama berjalannya algoritma.



Gambar 2 Contoh Kasus Penelusuran *Three Dimensional Maze* dua tingkat

3. ANALISIS

Prioritas arah yang digunakan pada penelusuran *Three Dimensional Maze* pada gambar 2 yakni utara, timur, selatan, barat, atas, dan bawah. Berikut merupakan proses penelusuran maze pada tiap langkahnya.

(1) Kondisi maze awal, obyek berada pada pintu masuk di level 2.
(2) Obyek memilih bergerak ke timur karena arah utara berupa tembok.
(3) Obyek kembali bergerak ke timur dengan alasan yang serupa.
(4) Obyek kini bergerak ke arah selatan karena arah utara dan timur berupa tembok.
(5) Obyek memiliki dua arah gerak yang mungkin, yakni utara dan selatan. Akan tetapi, arah utara berarti kembali ke posisi sebelumnya sementara masih ada jalur yang bisa diambil. Oleh karena itu, arah selatan diambil.
(6) Di sini obyek memilih bergerak ke bawah karena arah utara berarti kembali ke posisi sebelumnya sehingga obyek kini berada pada tingkat 1.
(7) Obyek langsung bergerak ke utara.
(8) Obyek bergerak ke barat karena arah utara dan timur berupa tembok sementara arah selatan berarti kembali.
(9) Obyek langsung bergerak ke utara.
(10) Obyek memilih bergerak ke timur karena arah utara berupa tembok.
(11) Arah utara, timur, dan selatan berupa tembok sehingga satu-satunya pilihan adalah kembali ke posisi sebelumnya. Pada tahap ini, obyek melakukan runut balik terhadap langkah yang telah diambilnya.
(12) Arah utara berupa tembok sementara arah timur dan selatan akan membawa obyek kembali ke posisi yang pernah ditempati sebelumnya, karena itu dipilih arah barat. Ternyata posisi ini merupakan pintu keluar maze.

Penelusuran di atas dilakukan dengan menguji setiap arah gerak di mana obyek akan bergerak ke arah tersebut jika tidak ditemukan penghalang. Proses penelusuran ini ternyata sangat mirip dengan penelusuran maze biasa, hanya saja ditambahkan dua arah gerak yang mungkin ditelusuri, yakni arah gerak atas dan bawah. Proses ini tampak tidak akan banyak berubah untuk dimensi berapa pun. Perubahan hanya mencakup penambahan arah gerak yang ingin ditelusuri.

4. KESIMPULAN

Berdasarkan pengujian dan analisis yang dilakukan, dapat ditarik kesimpulan berikut:

4.1 Algoritma *backtracking* dapat memberikan solusi untuk permasalahan *Three Dimensional Maze*.
4.2 Algoritma penelusuran maze dapat diperluas dengan mudah untuk dimensi yang lebih besar di mana cukup dilakukan penambahan arah yang akan ditelusuri.

REFERENSI

[1] Munir, Rinaldi. 2005. Strategi Algoritmik. Bandung : Penerbit Informatika.
[2] *http://www.astrolog.org/labyrnth/algrithm.htm*.Diakses pada 2 Januari 2010.