Analisis Penerapan Algoritma *Backtracking* Pada Pencarian Jalan Keluar di Dalam Labirin

Andika Pratama
13505048

Alamat: Jl. Dago Asri Blok C No.16
e-mail: if15048@students.if.itb.ac.id

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2007

ABSTRAK

Setiap manusia ingin menyelesaikan permasalahan yang dihadapi dengan secepat-cepatnya dan mendapatkan keuntungan sebanyak-banyaknya dengan mengefisiensikan sumber daya yang dimiliki terhadap batasan-batasan yang ditemui pada suatu masalah. Saat ini permainan-permainan yang ada juga memberikan permasalahan-permasalahan pada penggunanya dan sangat mungkin pula terjadi di kehidupan nyata.

Dunia permainan juga adalah salah satu implementasi dari bidang informatika. Perkembangan permainan pada masa kini sudah sangat jauh, namun sebuah algoritma yang selalu menjadi dasar dari semua permainan adalah algoritma *backtracking* atau algoritma runut-balik. Algoritma runut-balik sendiri adalah algoritma yang berbasis pada *Depth First Search* untuk mencari solusi persoalan secara lebih mangkus.

Salah satu permainan yang dibahas di dalam mata kuliah ini adalah sebuah permainan yang sudah tidak asing lagi, yaitu permainan mencari jalan keluar dari sebuah labirin (*Maze Labirin*). Labirin adalah jaringan jalan yang ruwet dan berliku-liku. Kita diharuskan menemukan jalan mana yang harus dilalui untuk sampai pada jalan keluar.

Berbagai data dalam makalah ini penulis peroleh dari berbagai sumber yang berkaitan. Sumber utama yang menjadi referensi penulis adalah Diktat Kuliah IF2251 "Strategi Algoritmik" yang disusun oleh Ir. Rinaldi Munir,M.T. Selain dari sumber tersebut penulis juga mendapatkan referensi dari situs-situs yang berkaitan dengan penerapan algoritma yang telah kami pelajari untuk permainan Maze Labirin.

Kata kunci: Backtracking, Depth First Search, Maze, Algoritma

.

1. PENDAHULUAN

Setiap manusia ingin menyelesaikan permasalahan yang dihadapi dengan secepat-cepatnya dan mendapatkan keuntungan sebanyak-banyaknya dengan mengefisiensikan sumber daya yang dimiliki terhadap batasan-batasan yang ditemui pada suatu masalah. Saat ini permainan-permainan yang ada juga memberikan permasalahan-permasalahan pada penggunanya dan sangat mungkin pula terjadi di kehidupan nyata.

Salah satu bentuk permainan yang sering kita hadapi di permainan ataupun di dunia nyata adalah permainan *Maze Labirin.* Labirin adalah jaringan jalan yang ruwet dan berliku-liku. Kita diharuskan menemukan jalan mana yang harus dilalui untuk sampai pada jalan keluar.

Di Bandung, labirin banyak sekali ditemukan pada permukiman penduduk yang padat. Gang-gang sempit di dalam perkampungan tersebut berliku-liku dan sungguh memusingkan. Seseorang yang tersesat masuk ke sana dan asing dengan lingkungan di dalamnya pasti akan kesulitan menemui jalan keluar. Algoritma yang tepat untuk menemukan jalan keluar dari dalam labirin adalah algoritma runut-balik.

2. ALGORITMA RUNUT-BALIK

2.1 Deskripsi Umum Algoritma *Backtracking*

Algoritma *backtracking* pertama kali diperkenalkan oleh D.H. Lehmer pada tahun 1950. Dalam perkembangannya beberapa ahli seperti RJ Walker, Golomb, dan Baumert menyajikan uraian umum tentang *backtracking* dan penerapannya dalam berbagai persoalan dan aplikasi.

Runut-balik merupakan perbaikan dari algoritma *brute-force*, secara sistematis mencari solusi persoalan di antara semua kemungkinan yang ada. Hanya pencarian yang mengarah ke solusi saja yang dikembangkan, sehingga waktu pencarian dapat dihemat. Runut-balik lebih alami dinyatakan dalam algoritma rekursif.

Algoritma *backtracking* (runut balik) merupakan salah satu metode pemecahan masalah yang termasuk dalam strategi yang berbasis pencarian pada ruang status. Algoritma *backtracking* 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. Proses ini dicirikan dengan ekspansi simpul terdalam lebih dahulu sampai tidak ditemukan lagi suksesor dari suatu simpul.

2.2 Prinsip Pencarian Solusi dengan Metode Runut-Balik

Langkah-langkah pencarian solusi pada pohon ruang status yang dibangun secara dinamis :

1. Solusi dicari dengan membentuk lintasan dari akar ke daun. Aturan pembentukan yang dipakai adalah mengikuti metode pencarian mendalam (DFS). Simpulsimpul yang sudah dilahirkan dinamakan **simpul hidup** (*live node*). Simpul hidup yang sedang diperluas dinamakan **simpul-E** (*Expand-node*). Simpul dinomori dari atas ke bawah sesuai dengan urutan kelahirannya.

2. Tiap kali simpul-E diperluas, lintasan yang dibangun olehnya bertambah panjang. Jika lintasan yang sedang dibentuk tidak mengarah ke solusi maka simpul-E tersebut "dibunuh" sehingga menjadi **simpul mati** (*dead node*). Fungsi yang digunakan untuk membunuh simpul- E adalah dengan menerapkan **fungsi pembatas** (*bounding function*). Simpul yang sudah mati tidak akan pernah diperluas lagi.

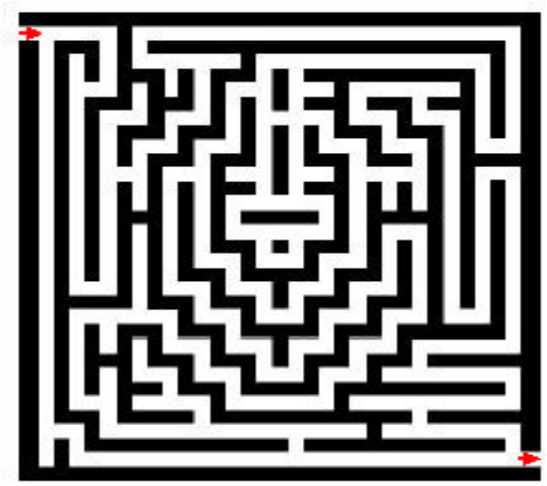
3. Jika pembentukan lintasan berakhir dengan simpul mati, maka proses pencarian diteruskan dengan membang-kitkan simpul anak yang lainnya. Bila tidak ada lagi simpul anak yang dapat dibangkitkan, maka pencarian solusi dilanjutkan dengan melakukan runutbalik ke simpul hidup terdekat (simpul orang tua). 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.

2.3 Penerapan Algoritma *Backtracking* dalam menyelesaikan Maze Labirin

Algoritma yang tepat untuk menemukan jalan keluar dari dalam labirin adalah algoritma runut-balik. Dengan algoritma ini, kita mencoba sebuah lintasan hingga menemui jalan buntu, lalu jejaki, (retrace) langkah sebelumnya sampai kita menemukan lintasan yang lain, lalu ulangi lagi lintasan tersebut. Pada akhirnya kita akan menemukan lintasan yang mengarah ke pintu keluar, atau mencoba semua lintasan dan memutuskan tidak terdapat solusinya.

Untuk menggambarkan algoritma runut-balik secara rinci, bagi lintasan menjadi sederetan langkah. Sebuah langkah terdiri dari pergerakan satu unit sel pada arah tertentu. Arah yang mungkin: ke atas (*up*), ke bawah (*down*), ke kiri (*left*), ke kanan (*right*). Algoritma runut baliknya secara garis besar adalah :



Gambar 1.Sebuah Labirin

```
while belum sampai pada tujuan do
  if terdapat arah yang benar sedemkian sehingga kita
     belum pernah berpindah ke sel pada arah tersebut
  then
     pindah satu langkah ke arah tersebut
  else
     backtrack langkah dampai terdapat arah seperti
     yang disebutkan di atas
  endif
endwhile
```

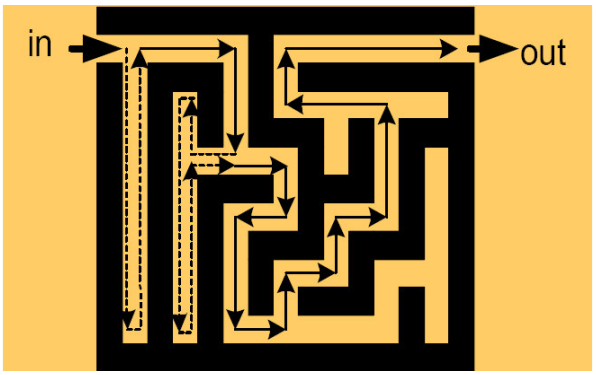
Namun terdapat permasalahan, bagaimana mengetahui langkah yang mana yang perlu dijejaki kembali? Ada dua solusi untuk masalah ini: pertama, simpan semua langkah yang pernah dilakukan, atau kedua, gunakan rekursi (yang secara implisit menyimpan semua langkah). Rekursi adalah solusi yang lebih mudah.

Algoritma runut-balik persoalan labirin adalah sebagai berikut. Parameter fungsi adalah labirin M yang menyimpan status labirin (labirin dapat diimplementasikan sebagai matriks 0/1, yang dalam hal ini jalan dinyatakan sebgai rangkaian 0, sedangkan dinding/tembok dinyatakan sebagai rangkaian 1):

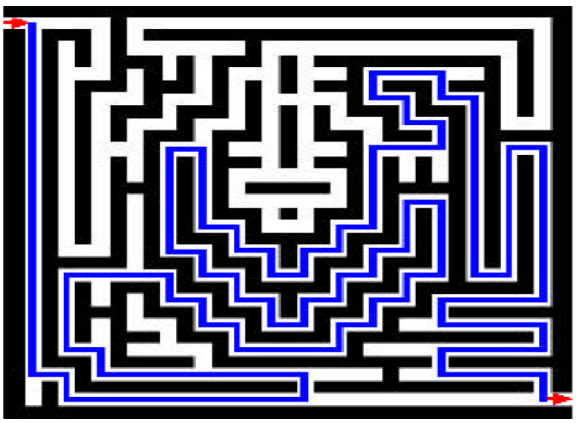
```
function Solve Maze (input M : labirin) => boolean
{ true jika solusi ditemukan, false jika tidak }

Deklarasi
        Arah : integer { up = 1, down, 2, left = 3, right = 4 }
Algoritma:
        If solusi sudah ditemukan then
           return true
        else
          for tiap arah gerakan  (up, down, left, right) do
                  move (M, arah) { pindah satu langkah (satu sel) sesuai arah tersebut}
                  if  SolveMaze(M) then
                      return true
                  else
                      unmove (M, arah ) {backtrack}
                  endif
          endfor
          return false {semua arah sudah dicoba, tetapi tetap buntu, maka kesimpulannya: tidak ada solusi}
        endi f
```

Algoritma runut-balik di atas hanya memberikan apakah ada solusi (yaitu, ada lintasan dari titik masuk ke titik keluar labirin) atau tidak ada solusi. Algoritma tersebut tidak memberikan lntasan yang dilalui. Masalah ini dapat diatasi dengan pengamanan berikut. Jika pemanggilan rekursif *SolveMaze(M)* mengembalikan *true,* ini berarti bahwa perpindahan langkah yang dilakukan mengarah ke solusi. Oleh karena itu *output*-kan gambar labirin yang dihasilkan ke layar. Masalah kecil muncul karena perpinfahan langkah yang dihasilkan akan dicetak dalam urutan terbalik sebagaimana pemanggilan rekursif *return*. Untuk memecahkan ini, diperlukan langkah penyimpanan perpindahan langkah ke dalam stack dan mencetak seluruh langkah setelah pemanggilan *SolveMaze* dibandingkan mencetaknya segera.

Gambar 2 memperlihatkan lintasan yang dibentuk dengan algoritma runut-balik pada sebuah labirin. Runut Balik diperlihatkan dengan garis putus-putus. Sedangkan Gambar 3 memperlihatkan lintasan yang benar yang dilalui di dalam labirin pada gambar 1.



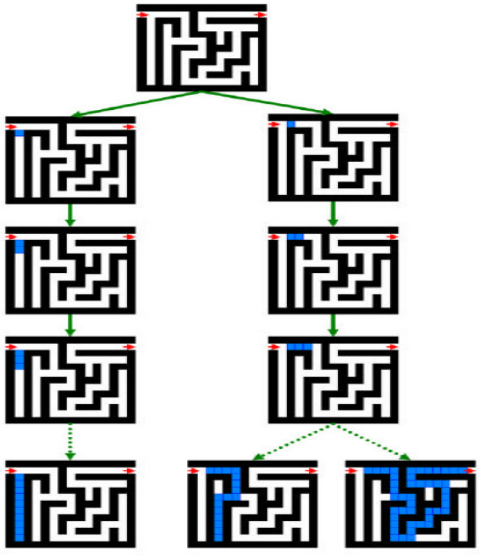
Gambar 2.Contoh runut-balik pada sebuah labirin



Gambar 3.Contoh runut-balik pada labirin gambar 1

Algoritma runut-balik pada persoalan labirin dapat dipandang sebagai pembentukan pohon ruang status. Akar dari pohon adalah labirin awal, dan anak-anaknya adalah labirin yang dihasilkan dari pergerakan satu langkah dari labirin semula. Simpul-simpul pada pohon dibentuk dengan skema traversal DFS.

Simpul daun merepresentasikan apakah status pergerakan langkah sampai pada titik buntu atau pintu keluar sudah ditemukan. Jikan pergerakan langkah mencapai titik buntu, maka simpul daun yang merepresentasikan status labirin mencapai titik buntu dimatikan dan lintasan pencarian dirunut-balik ke simpul-simpul si atasnya.Demikian seterusnya, pohon ruang status diekspansi sampai solusi ditemukan atau tidak terdapat solusi.



Gambar 4.Sebagian pohon ruang status pada persoalan labirin pada gambar 2

3. PERBANDINGAN ALGORITMA RUNUT-BALIK DENGAN ALGORITMA LAIN

Matematikawan Leonhard Euler adalah salah satu dari sekian banyak orang yang pertma kali menganalisa labirin-labirin ini secara matematis. Dengan melakukan hal tersebut, beliau telah juga membangun sebuah cabang ilmu baru yaitu *topologi.*

Salah satu algoritma lain yang digunakan untuk menyelesaikan *Maze Labirin* adalah dengan Algoritma *Wall Follower* .Algoritma ini diterapkan dengan terus berjalan mengikuti jalan di dalam labirin hingga menemukan percabangan yang berbelok ke arah kiri. Setiap menemukan percabangan jalan yang memiliki jalan yang berbelok ke arah kiri, kita langsung berbelok kiri. Namun jika pada suatu percabangan tidak terdapat belok kiri maka kita hanya perlu terus berjalan lurus atau mengikuti jalan yang ada. Jika menemui jalan buntu, kita hanya perlu berbalik arah dan terus melanjutkan perjalanan dengan mengikuti aturan-aturan sebelumnya. Hal ini juga berlaku untuk arah kanan. Kita dapat mengganti semua kata kiri di atas dengan kata kanan dan akan mendapatkan hasil yang sama asalkan kita selalu konsisten.

Algoritma ini mirip seperti algoritma *brute force* dikarenakan seperti mencoba semua kemungkinan jalan yang ada namun sebenarnya berbeda, karena dengan algoritma ini kita terus menelusuru semua jalan yang ada namun hanya yang perlu kita telusuri saja.

Algoritma ini tetaplah memiliki kelemahan, bahwa jika pemain memulai permainan dalam labirin terebut bukanlah dari pintu masuk namun dari tengah-tengah labirin, dan labirin tersebut memiliki jalur berputar di tengahnya dapat menyebabkan pemain terus berputar-putar di dalam labirin tanpa pernah bisa keluar dari labirin. Hal ini telah disempurnakan oleh algoritma *Pledge* dimana arah pandang kita terus diperhatikan. Ke arah manakah kita menghadap(utara, selatan, timur, barat), terus diperhitungkan dan akan memberikan kita jalur yang tepat agar kita bisa keluar dari labirin.

Sehingga secara kompleksitas waktu algoritma ini jelas jauh lebih cepat dibandingkan algoritma brute force dan hampir sama dengan algoritma runut-balik.

Terdapat pula algoritma Tremaux's. Algoritma Tremaux's adalah sebuah metode yang efektif yang memerlukan kita untuk menggambar garis pada lantai untuk menandai jalur yang dipastikan akan bekerja untuk semua labirin. Ketika tiba di sebuah percabangan, pilih salah satu arah dan tandai dan dari arah manakah kamu memasuki arah tersebut. Ketika kamu tiba di sebuah percabangan yang telah ditandai pilih jalur yang belum ditandai jika mungkin. Jika tidak bias, masuki kembali jalur yang sudah ditandai dan beri tanda lagi (kamu bisa memasuki jalur darimana kamu datang). Jangan pernah

masuki jalur yang sudah pernah ditandai 2 kali, kamu tidak akan pernah perlu mengambil jalur manapun lebih dari dua kali. Jika tidak ada jalan keluar, cara ini akan membawa kamu kembali ke pintu masuk.

Kelebihan dari algoritma-algoritma ini adalah kerelevanannya untuk diterapkan dalam kehidupan sehari-hari dibandingkan dengan algoritma runut-balik. Hal ini dapat memudahkan kita untuk menemukan jalan kita tersesat di dunia nyata.

IV. KESIMPULAN

Algoritma *backtracking* merupakan algoritma yang cukup mangkus untuk menyelesaikan berbagai persoalan. Hal ini disebabkan karena pada prinsipnya tidaklah diperlukan untuk memeriksa semua kemungkinan solusi yang ada. Pencarian hanya mengarah pada solusi yang dipertimbangkan saja. Oleh karena algoritma ini cukup efektif, maka *backtracking* banyak diterapkan dalam berbagai permainan dan persoalan yang berkaitan dengan bidang kecerdasan buatan (*artificial inteligence*).

Hasil analisis kemampuan algoritma *backtracking* dalam menemukan jalan keluar *Maze Labirin* menunjukkan bahwa algoritma ini cukup efektif untuk mendapatkan solusi persoalan tersebut. Sistem kerja algoritma *backtracking* yang sistematis dan ciri khasnya yang hanya memeriksa kemung-kinan solusi yang memang dapat dipertimbangkan untuk menjadi solusi akhir, diperkirakan dapat menjadi solusi yang efektif dan efisien untuk persoalan ini. Secara kompleksitas algoritma pun metode *backtracking* jauh lebih cepat dari metode *brute force* dan dapat memberikan hasil yang unik.

Melalui studi literatur dan analisis yang telah dilakukan, penulis berpendapat bahwa algoritma *backtracking* cukup efektif meskipun masih terdapat beberapa algoritma lain yang dapat digunakan untuk menyelesaikan permasalahan ini dengan kompleksitas algoritma yang juga relative kecil.

REFERENSI

[1] Munir,Rinaldi.2004.Diktat Kuliah Matematik Diskrit.Departemen Teknik Informatika ITB

[2] Munir,Rinaldi.2005.Diktat Kuliah Strategi Algoritmik.Departemen Teknik Informatika ITB

[3] Preiss, Bruno. 1997. Abstract Backtracking Solvers. *http://www.brpreiss.com/books*. Diakses tanggal 21 Mei 2007 pukul 22.00

[4] http://en.wikipedia.org/wiki/Maze. Diakses tanggal 21 Mei 2007 pukul 20.00

[4] http://en.wikipedia.org/wiki/Backtrack. Diakses tanggal 21 Mei 2007 pukul 20.00