PENCARIAN JALAN KELUAR LABIRIN DENGAN METODE *WALL FOLLOWER*

Primanio

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Jalan Ganesha 10, Bandung
e-mail: if15027@students.if.itb.ac.id

ABSTRAK

Labirin (*maze*) adalah permainan yang sudah tidak asing di telinga kita. Labirin adalah jaringan jalan yang rumit dan berliku-liku. Sejak zaman dahulu, labirin telah digunakan dalam berbagai kepentingan, mulai dari proteksi keamanan hingga hiburan. Pada umumnya, labirin dibuat untuk tujuan hiburan. Dalam kehidupan nyata, labirin dapat ditemukan pada susunan jalan kecil atau gang-gang di kawasan perumahan. Sangat sulit bila seseorang yang asing dengan daerah tersebut untuk mencari jalan. Bila seseorang mengetahui metode untuk keluar dari sebuah labirin, maka mereka dapat dengan mudah mengatasi kesulitan yang dirasakan. Ada berbagai jenis metode pencarian jalan keluar untuk masalah labirin ini. Salah satu metode pencari jalan keluar yang sedehana adalah metode *wall follower*. Metode ini juga dikenal dengan aturan tangan kanan (*right-hand rule*) atau aturan tangan kiri (*left-hand rule*). Metode ini merupakan kombinasi dari algoritma runut-balik (*backtracking*) dan algoritma *greedy*. Metode ini akan mencari jalan sesuai dengan dinding labirin, baik itu ke kiri maupun ke kanan. Makalah ini akan mengulas tentang penggunaan metode *wall follower* utuk menemukan jalan keluar dalam sebuah labirin.

Kata kunci: algoritma *greedy*, *backtracking*, labirin, *left-hand rule*, *maze*, *maze solving*, *right-hand rule*. *wall follower*.

1. PENDAHULUAN

Labirin atau *maze* adalah sebuah *puzzle* dalam bentuk percabangan jalan yang kompleks dan memliki banyak jalan buntu. Tujuan permainan ini adalah pemain harus menemukan jalan keluar dari sebuah pintu masuk ke satu atau lebih pintu keluar. Bisa juga kondisi pemain menang yaitu ketika dia mencapai suatu titik di dalam labirin tersebut.

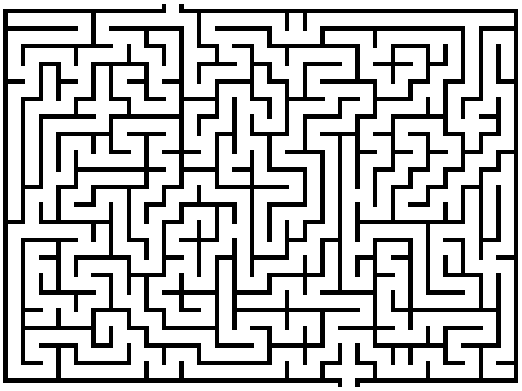
Labirin dalam dunia nyata banyak dibuat di taman atau ruangan-ruangan dengan pembatas berupa pagar tanaman, tembok atau pagar. Ukurannya bervariasi, tergantung ukuran ruangan atau taman tersebut. Labitin ini biasanya memang dirancang untuk menjadi sebuah atraksi permainan (misalnya rumah kaca) atau hanya sebagai hiasan saja.

Selain itu banyak labirin yang terbentuk secara "tidak sengaja". Contohnya jalan-jalan kecil atau gang-gang yang terbentuk diantara rumah-rumah pada kawasan pemukiman. Labirin-labirin ini secara tidak langsung "menyesatkan" orang asing yang masuk ke dalamnya.

Pada umumnya pembuatan labirin hanya untuk hiburan belaka. Namun, banyak bangunan yang menerapkan labirin sebagai salah satu sistem keamanan agar orang yang tidak berkepentingan atau tidak dikenal sulit untuk masuk ke dalam bangunan.

Labirin untuk permainan biasanya dicetak dalam sebuah kertas utuk diselesaikan oleh pemain. Permainan dilakukan dengan cara menuliskan jalan yang telah ditempuh menggunakan pensil atau hanya dengan menunjuk jalannya menggunakan jari.

Labirin terbagi menjadi beberapa kategori sesuai jenisnya, yaitu Labirin 2 dimensi, 3 dimensi, bentuk segitiga, sigma, dan masih banyak lagi. Sebagai batasan materi, makalah ini hanya akan membahas tentang pencarian jalan dalam labirin 2 dimensi.



Gambar 1. Contoh labirin 2 dimensi.

2. METODE

Ada banyak metode pencarian jalan keluar untuk persoalan labirin ini. Ada metode yang berfokus pada labirinnya seperti D*ead-end filler*, *Cul-de-sac filler*, *Blind alley filler* dan *Blind alley sealer*. Ada juga metode yang berfokus pada pemain seperti algoritma rantai, algoritma *pledge*, *wall follower*, runut-balik rekursif, Algoritma *Tremaux* dan masih banyak lagi

Metode yang penulis pilih adalah metode *wall follower*. Metode ini juga dikenal dengan aturan tangan kanan (*right-hand rule*) atau aturan tangan kiri (*left-hand rule*). Metode ini adalah metode yang sederhana dan paling mudah untuk diaplikasikan dalm dunia nyata ataupun pada program komputer.

Algoritma ini dapat dikategorikan sebagai algoritma runut-balik (*backtracking*). Bila pemain menemukan jalan buntu, maka pemain akan merunut-balik sampai ke percabangan terakhir dan kembali melakukan pencarian jalan keluar.

Namun pada beberapa bagian, algoritma ini bersifat *greedy* karena dalam setiap keputusannya, metode ini mengambil pilihan yang terbaik tanpa memperhatikan konsekuensi ke depannya dan berharap dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global.

Metode ini berfokus pada pemain, dengan algoritma yang sangat sederhana. Prinsipnya adalah selalu berbelok ke kiri atau ke kanan dengan konsisten apabila kita menemukan percabangan. Bila pemain menemukan jalan buntu, pemain harus berbalik arah. Kita akan mengetahui apakah labirin memiliki jalan keluar atau tidak apabila kita akhirnya kembali ke pintu masuk. *Pseudo-code* dari algoritma ini adalah:

```
function SolveMaze (M : Maze)→ boolean
{Menyelesaikan maze M, mereturn true
bila maze memiliki solusi dan mereturn
false bila maze tidak memiliki solusi,
menerapkan metode wall follower
berorientasi kiri}

Deklarasi:
   Selesai, Ada_solusi : boolean

Algoritma:
   Selesai ← false
   Ada_solusi ← true

   while not Selesai and Ada_solusi do
      if (Pintu_Masuk) then
         Ada_solusi ← false
      else if (Pintu_Keluar) then
         Selesai ← true
      else if (Percabangan) then
         {Belok kiri}
      else if (Jalan_Buntu) then
         {Berbalik Arah (backtrack)}
      else
         {Maju 1 sel sesuai jalan}
      endif
   endwhile
   return Ada_solusi
```

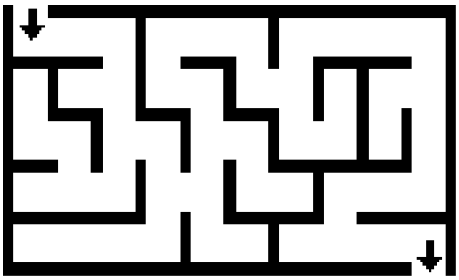
Algoritma ini tidak memerlukan memori penyimpan data karena tidak ada *state* yang disimpan. Kita hanya perlu mengetahui kondisi setiap langkah kita saja dengan mendapatkan status dari *maze*-nya sendiri. Selain itu, algoritma ini tidak dapat diterapkan pada labirin yang mempunya *goal* di tengah labirin, karena algoritma ini akan terus melakukan backtrack higga ke pintu, baik di wal maupun di akhir.

Algoritma ini juga bisa diterapkan pada labirin 3 dimensi yang memakai tangga. Dengan menganggap bahwa tangga naik adalah arah serong-kiri (barat-laut) atau serong-kanan (timur-laut), maka algoritma ini dapat digunakan.

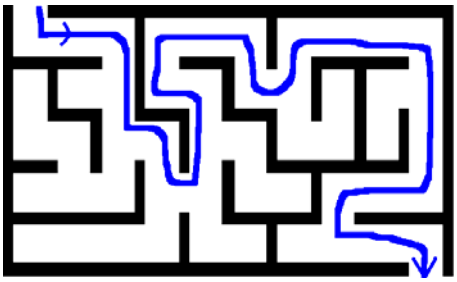
3. APLIKASI

Uji coba algoritma ini dilakukan terhadap 3 buah labirin. Ketiga labirin ini memiliki karakteristik berbeda. Berikut ini adalah ketiga labirin tersebut beserta penyelesaiannya dengan menggunakan metode *wall follower* berorientasi kiri.

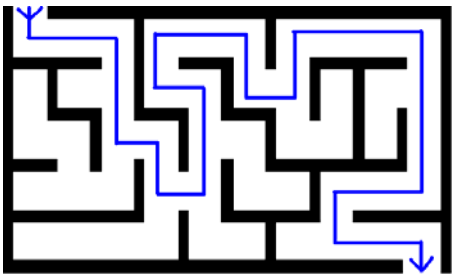
3.1 Labirin Dengan Solusi Optimum



Gambar 2. Labirin 1 – Kondisi Awal



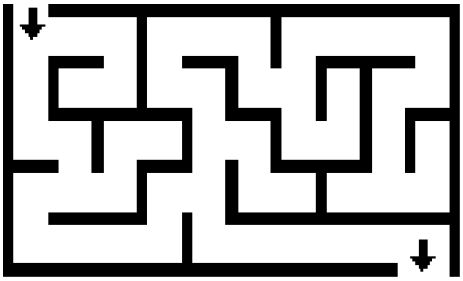
Gambar 3. Labirin 1 – Solusi *Wall Follower*



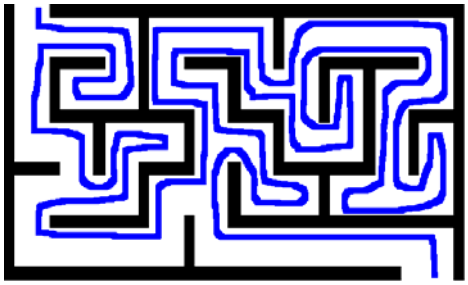
Gambar 4. Labirin 1 – Jalur Optimum

Dapat dilihat bahwa solusi yang didapatkan dengan mengaplikasikan metode *wall follower* sama dengan jalur optimum dari labirin ini.

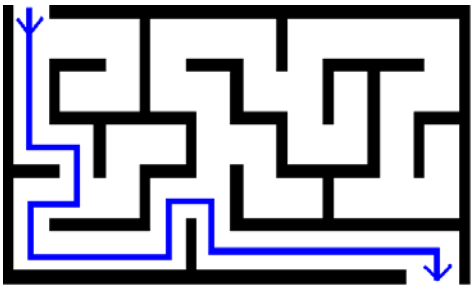
3.2 Labirin Dengan Solusi Tidak Optimum



Gambar 5. Labirin 2 – Kondisi Awal



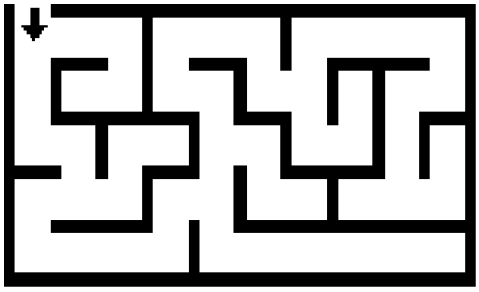
Gambar 6. Labirin 2 – Solusi *Wall Follower*



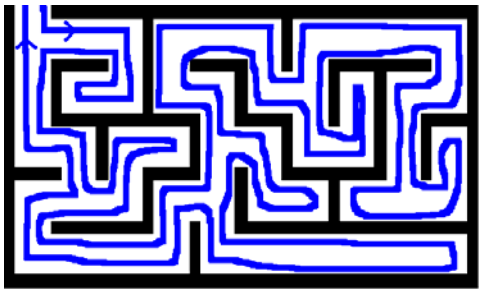
Gambar 7. Labirin 2 – Jalur Optimum

Pada kasus kedua ini, jalan keluar yang didapatkan algoritma ini jauh berbeda dengan solusi optimum. Jalur yang didapatkan harus memutar dan melalui hampir semua jalan yang ada pada labirin.

3.3 Labirin Tanpa Solusi



Gambar 8. Labirin 3 – Kondisi Awal



Gambar 9. Labirin 2 – Keadaan Akhir

Pada kasus ini, metode *wall follower* terbukti dapat menangani labirin yang tidak memiliki jalan keluar. Metode ini akan membawa kembali pemain ke jalan masuk agar pemain tidak “tersesat”.

4. ANALISIS

Dari uji coba yang telah dilakukan, didapatkan bahwa algoritma *well follower* ini tidak selalu menghasilkan solusi yang optimum. Hal ini disebabkan oleh beberapa hal, antara lain:

1. Algoritma ini masih bersifat *greedy*. Setiap langkah dilakukan tanpa memperhitungkan konsekuensi ke depannya, apakan langkah ini merupakan langkah menuju jalan terbaik atau bukan.
2. Keputusan yang diambil bukan berdasarkan jalan yang terdekat, akan tetapi selalu mengambil jalan yang tetap, yaitu ke kiri atau ke kanan secara teus menerus.

Dengan diketahuinya hal-hal penghambat tersebut, maka kita bisa menyempurnakan algoritma ini dengan menghitung jarak terpendek dari suatu titik sebelum mengambil keputusan – meskipun hal ini tidak dapat dilakukan di dunia nyata. Hal ini akan membuat pengambilan keputusan menjadi lebih tepat dan solusi yang dihasilkan menjadi lebih optimum.

Akan tetapi, algoritma ini telah terbukti akan selalu menemukan jalan keluar dalam labirin, meskipun jalan keluar tersebut hanya satu dan merupakan jalan masuknya. Seberapa rumit pun labirin yang ada, pasti algoritma ini menemukan jalan keluar bagi pemain, sehingga pemain tidak mungkin "tersesat" dalam labirin.

Aplikasi metode ini akan lebih terasa efektifnya di dunia nyata bila dibandingkan dengan metode lain yang berfokus pada labirin. Metode ini berfokus pada pemain sehingga pemain tidak perlu mengetahui bentuk keseluruhan labirin. Kita hanya perlu bermain dan melihat kondisi saat kita mengambil keputusan. Hal ini sangat mencerminkan kejadian dalam labirin di dunia nyata. Berbeda dengan metode yang berfokuskan labirinnya, di mana kita harus mengetahui jalan mana yang merupakan jalan keluar, percabangan atau jalan buntu secara keseluruhan sebelum kita bermain.

Selain itu, bila dibandingkan algoritma runut-balik yang biasa (secara rekursif), algoritma ini lebih mudah diterapkan karena kita tidak perlu memberikan tanda di setiap percabangan sehingga kita tidak perlu mengingat jalur yang telah kita tempuh.

5. KESIMPULAN

Metode *wall follower* adalah salah satu metode pencarian jalan keluar dalam labirin yang pasti dapat menemukan solusi dari permasalahan labirin dengan syarat tujuan dari labirin tersebut adalah sebuah jalan keluar, bukan suatu area atau tempat di dalam labirin. Algoritma ini memang tidak selalu dapat menemukan jalur terpendek, namun algoritma ini dapat dengan mudah diaplikasikan dalam kehidupan sehari-hari dibandingkan dengan metode pencarian jalan keluar lain karena algoritma ini tergolong sederhana. Algoritma ini juga tidak membutuhkan memori karena tidak menyimpan *state* sebelumnya.

REFERENSI

[1] Munir, Rinaldi. 2007. *Diktat Kuliah IF2251 Strategi Algoritmik.* Bandung: Program Studi Teknik Informatika, STEI ITB.

[2] ---. 21 Mei 2007. *Maze Algorithms.* http://www.astrolog.org/labyrnth/algrithm.htm

[3] ---. 21 Mei 2007. *Maze – Wikipedia the free encyclopedia.* http://en.wikipedia.org/wiki/Maze

[4] ---. 22 Mei 2007. *Binus - Bluejackets Forum, Sehari-hari Thread.* http://bluejack.binus.ac.id/BJ/Default.aspx?tabid=63&g=topics&f=44

[5] ---. 22 Mei 2007. *Echo – Indonesian Hacker & Open Source Forum, Algoritma & Pemrograman Thread.* http://echo.or.id/forum//viewforum.php?f=1&sid=60075215e808ad5b3574421d78f84853