PERBANDINGAN BERBAGAI ALGORITMA PENYELESAIAN PERMASALAHAN LABIRIN

Muhammad Luthfi

Program Studi Teknik Informatika
Sekolah Teknik Elektro Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung 40135
email: luthfi@comlabs.itb.ac.id

ABSTRAK

Labirin atau Maze merupakan suatu jalur yang sengaja dibuat rumit sehingga seseorang yang masuk ke dalamnya tidak dapat keluar dengan mudah. Dalam menyelesaikan permasalahan ini, terdapat banyak sekali cara yang dapat digunakan. Beberapa cara seperti berjalan acak, sentuh tembok, lewati pintu lain, pencarian melebar, pencarian mendalam dan *cheese algorithm* terbukti dapat digunakan untuk menyelesaikan permasalahan labirin ini.

Diantara cara yang disebutkan diatas, ada beberapa cara yang memiliki metode yang menarik untuk diimplementasikan. Misalnya *cheese algorithm* yang melakukan pecarian pintu keluar dengan analogi tikus yang sedang mencari keju. *Cheese algorithm* ini tidak hanya mengadalkan tikus sebagai pencari keju, namun juga keju itu sendiri yang memberi tahu keberadaanya melalui intensitas bau yang dihasilkan.

Dalam makalah ini juga diterangkan beberapa algoritma lain yang sangat sederhana. Bahkan cara seperti berjalan acak mungkin tidak dikategorikan sebagai algoritma. Algoritma lain seperti berjalan menyentuh tembok dengan salah satu tangan dan melewati pintu lain yang belum dilalui juga merupakan cara sederhana yang kita dapatkan dari pengalaman sehari-hari.

Kata Kunci: labirin, maze, berjalan acak, sentuh tembok, lewati pintu lain, bfs, dfs, cheese algorithm.

1. PENDAHULUAN

Algoritma merupakan cara/urutan langkah penyelesaian masalah. Dengan mengikuti langkah-langkah penyelesaian masalah itu, kita dapat menemukan solusi dari permasalahan yang kita cari. Dalam membangun suatu algoritma, kita harus mengerti permasalahan yang kita hadapi dengan baik. Sehingga kita dapat menentukan langkah yang terbaik untuk diambil, dimana langkah itu mengacu pada optimasi solusi yang diinginkan. Dalam algoritma dikenal istilah kompleksitas. Kompleksitas algoritma ini dapat digunakan untuk menilai kemangkusan algoritma.

2. PERMASALAHAN LABIRIN

Labirin merupakan suatu ruang dengan jalan keluar yang sulit dan berliku. Permasalahan labirin merupakan salah satu permasalahan yang telah lama digunakan untuk menguji masalah optimasi algoritma. Banyak algoritma yang digunakan untuk menyelesaikan permasalahan ini, mulai dari algoritma yang mudah hingga algoritma yang rumit. Dalam makalah ini akan ditunjukkan beberapa algoritma yang digunakan untuk menyelesaikan permasalahan labirin.

3. ALGORITMA PENYELESAIAN

Bila suatu labirin telah diketahui dapat diselesaikan, maka kita dapat menyusun optimasi untuk algoritma yang akan kita bangun. Suatu algoritma sebenarnya tidak harus selalu rumit. Berikut akan diperkenalkan beberapa algoritma yang dapat digunakan untuk menyelesaikan permasalahan labirin, mulai dari yang sederhana hingga algoritma yang cukup rumit.

3.1. Berjalan Acak

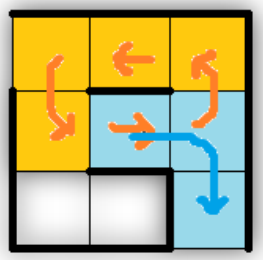
Berjalan acak merupakan cara penyelesaian permasalahan labirin yang paling sederhana. Bahkan, cara ini mungkin tidak disebut sebagai algoritma penyelesaian. Dengan berjalan acak, jalan keluar dicari dengan menyusuri petak yang ada pada labirin secara acak dari petak awal (pintu masuk) hingga didapatkan pintu keluarnya. Berjalan acak dapat diumpamakan seperti seseorang yang menerka-nerka jalan keluar dari labirin. Dengan menelusuri setiap petak secara acak, maka akan selalu ada kemungkinan untuk dapat menemukan pintu keluar dari labirin tersebut

3.2. Menyentuh Tembok

Algoritma penyelesaian dengan menyentuh tembok dilakukan dengan menyentuhkan selalu salah satu tangan kita

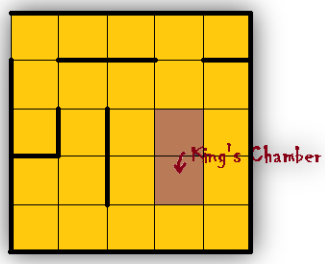
pada tembok labirin. Tangan yang kita sentuhkan pada labirin haruslah tangan yang sama. Bila kita memilih tangan kanan, maka kita harus selalu menggunakan tangan kanan untuk menyentuh tembok labirin.

Agoritma ini cukup efektif untuk digunakan mencari jalan keluar dari satu labirin. Namun, algoritma ini memiliki kelemahan. Jika labirin yang dilalui memiliki banyak jalan (multiple path labirinth), maka perlu dilakukan perbaikan algoritma. Perbaikan dilakukan dengan memberi tanda pada petak yang telah dilalui. Jika kita kembali pada petak yang telah dilalui tersebut, maka kita harus mengganti tangan yang digunakan untuk menyentuh tembok labirin.



Gambar1. Multiple Path Labirinth
(dimulai dari tengah, sentuh tembok atas dengan tangan kiri)

Walaupun telah dilakukan perbaikan, algoritma ini tetap tidak dapat menyelesaikan permasalahan labirin jika petak yang dituju (goal akhir) berada di tengah ruangan (king's chamber labirinth).



Gambar2. King's Chamber Labirinth

3.3. Lewat Pintu Lain

Lewati pintu lain atau *Least Recently Used Walk*[1] merupakan algoritma yang simple untuk menyelesaikan permasalahan labirin. Konsep utamanya adalah dengan menandai petak yang pernah dilalui. Namun, pada algoritma ini, bukan petak yang ditandai melainkan "pintu masuk" ke petak tersebut.

1. Tandai seluruh pintu dengan nilai 0,
2. Inisiasi *i* dengan 0,
3. Cek petak, jika petak tersebut merupakan tujuan akhir, berhenti pada petak tersebut,
4. Incremen *i* (*i*++),
5. Pilih pintu dengan nilai terndah,
6. Masuk dan tandai pintu tersebut dengan nilai *i*
7. Kembali ke langkah 3.

Konsep utamanya adalah melewati pintu dengan nilai terrendah (paling sedikit dilewati sebelumnya). Sehingga seolah-olah kita memilih pintu yang belum pernah dilewati sebelumnya dan memiliki kemungkinan lebih besar untuk dapat mencapai pintu keluar atau petak akhir yang kita cari lewat pintu tersebut.

3.4. Algoritma Pencarian Melebar

Pada algoritma pencarian melebar (*Breadth First Search*), pencarian dilakukan dengan mengunjungi petak mulai dari petak awal atau pintu masuk (sebagai root) lalu dilanjutkan dengan mengunjungi (membangkitkan) semua petak lain yang bertetangga dengan petak awal tersebut. Kemudian, dilanjutkan dengan mengunjungi petak-petak lain yang belum dikunjungi dan bertetangga dengan petak yang baru saja dikunjungi tersebut. Saat mengunjungi suatu petak baru, disana juga dilakukan pengecekan apakah petak tersebut merupakan petak akhir yang dituju. Bila petak yang dicek adalah petak akhir, maka pencarian akan dihentikan. Setelah itu, akan di-*generate* jalur solusi berdasarkan petak-petak yang telah dibangkitkan.

Pada algoritma ini, dikenal istilah laras atau ting-katan. Sehingga, setiap petak yang dibangkitkan secara bersamaan akan berada pada laras yang sama. Jika dimodelkan dalam bentuk pohon, maka semua petak pada laras *t* akan dikunjungi terlebih dahulu sebelum petak-petak pada laras *t* + 1 dikunjungi.

3.4.1. Langkah Penyelesaian secara BFS

Secara umum, algoritma pencarian melebar (BFS) secara iteratif adalah sebagai berikut:

1. mulai *traversal* dari petak root awal, masukkan petak ini ke dalam antrian
2. periksa petak yang ada diantrian
3. kunjungi semua petak lain yang bertetangga dengan petak tersebut dan masukkan ke dalam antrian.
5. ulangi langkah 2 dan 3 hingga solusi berhasil ditemukan atau tidak ada petak lain yang dapat dibangkitkan.
6. bila solusi ditemukan, *generate* solusi dangan melakukan trace dari petak-petak yang telah dibangkitkan.

3.5. Algoritma Pencarian Mendalam

Seperti halnya pada algoritma pencarian melebar (BFS), *traversal* titik pada algoritma pencarian mendalam (*Depth*

First Search) dimulai dari petak awal lalu dilanjutkan dengan mengunjungi (membangkitkan) petak yang bertetangga deng-an petak awal tersebut. Berbeda dengan BFS, pada DFS ini tidak semua petak yang bertetangga dibangkitkan secara bersamaan. Hanya salah satu dari sejumlah keadaan (*state*) yang telah ditentukan saja yang akan dibangkitkan. Pembangkitan simpul tersebut dapat dilakukan berdasakan suatu prioritas maupun secara acak.

Bila solusi belum ditemukan, pencarian kemudian dilanjutkan dengan membangkitkan satu petak lain. Pada pembangkitan petak ini, bila menemukan jalan buntu, tidak ada arah lain selain arah datang, maka akan dilakukan *backtrack. Backtrack* dilakukan hingga titik percabangan yang terakhir dilewati, kemudian dilanjutkann dengan ber-jalan melalui percabangan yang belum pernah dilewatinya. Hal ini dilakukan secara rekursif dan baru berhenti jika solusi ditemukan atau semua jalan telah dilewati.

3.5.1. Langkah Penyelesaian secara DFS

Secara umum, algoritma pencarian mendalam (DFS) secara rekursif ini adalah:

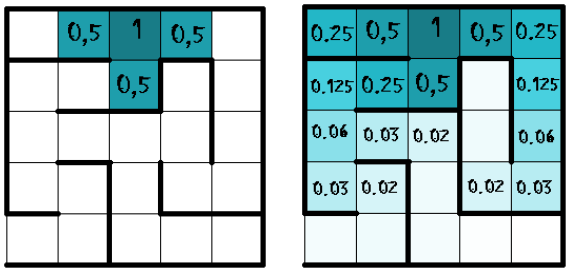
1. mulai traversal dari petak awal.
2. periksa apakah petak akhir (basis)
3. bila solusi belum ditemukan, kunjungi petak lain yang bertetangga dengan petak yang sebelumnya diperiksa.
4. panggil kembali fungsi DFS dan terapkan pada petak baru tersebut.
5. bila solusi tidak ditemukan, mundur ke petak sebelumnya.

3.6. Cheese Algorithm

Pada algoritma-algoritma yang dibahas sebelumnya, kita cenderung berfokus pada bagaimana seseorang dalam labirin dapat menemukan jalan keluar ataupun petak yang dicari dalam labirin tersebut. Berbeda dengan algoritma-algoritma tersebut, *cheese algorithm* menawarkan fokus yang berbeda. Pada algoritma ini, jalan keluar atau petak yang dicari, juga dipergunakan untuk menemukan solusi dengan lebih cepat.

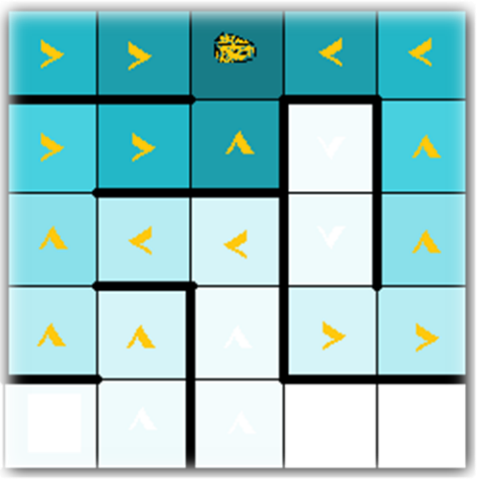
Secara umum, cara kerja algoritma ini mirip dengan algoritma pencarian melebar (BFS) setelah didapatkan himpunan solusi menuju petak akhir. Pada algoritma BFS, jalur solusi kemudian di-*generate* dari himpunan solusi tadi (akhir-awal), sehingga didapatkan jalur solusi yang sebenarnya (awal-akhir). Pada *cheese algorithm*, kita tidak mencari himpunan solusi terlebih dahulu, disini kita berperan sebagai tikus yang mencari keju dalam labirin. Petak akhir atau yang kita anngap sebagai keju inilah yang memberitahukan keberadaan berdasarkan intensitas bau yang menyebar dalam labirin sesuai dengan jaraknya terhadap tempak keju (petak akhir) berada.

Intensitas bau bergantung pada jarak kita terhadap keju. Sehingga makin dekat kita terhadap keju (petak akhir) akan semakin besar nilai intensitasnya. Katakanlah nilai intensitas ini berada antara 0 dan 1, dimana 1 adalah nilai petak dimana keju itu berada. Misal intensitas petak berikutnya kita tetapkan bernilai setengah dari nilai kotak yang lebih dekat dengan keju. Maka kita akan mendapatkan nilai petak-petak tersebut semakin kecil ketika menjauhi petak tempat keju berada.



Gambar3. Contoh persebaran intensitas pada labirin

Persebaran intensitas tersebut akan membentuk suatu pola yang mengarah pada petak tempat keju berada. Dengan cara ini, jika kita telah menemukan suatu rute maka kita dapat menggunakan rute tersebut untuk dapat menemukan petak akhir dengan lebih cepat.



Gambar4. Pola arah yang terbentuk dari persebaran intensitas pada labirin.

4. KESIMPULAN

Berbagai algoritma dapat diterapkan untuk menyelesaikan permasalahan labirin. Ada algoritma yang sulit namun efektif untuk digunakan, ada pula algoritma yang cenderung mudah dan sederhana, namun memerlukan waktu yang cukup lama untuk menyelesaikan permasalahan tersebut. Untuk itu, kita perlu memilih dengan baik algoritma apa yang hendak

digunakan agar mendapatkan solusi yang optimal dari permasalahan yang diajukan.

Untuk menyelesaikan permasalahan labirin ini, algoritma pencarian melebar (BFS) dan algoritma pencarian mandalam (DFS) lebih cocok digunakan. Selain keduanya dapat digunakan untuk menyelesaikan berbagai permasalahan labirin, algoritma ini juga cukup mangkus dan mudah dimengerti.

REFERENSI

[1] Suits, David B. Paper, "Playing With Maze", *Ro-chester Institute of Technology*, 1994, Rochester, New York 14623, hal.9-15.

[2] Munir, Rinaldi. *Diktat Kuliah IF3051 Strategi Algoritma*, Program Studi Teknik Informatika ITB: Bandung, 2005.

.