Perbandingan Algoritma *Depth-First Search* dan Algoritma *Hunt-and-Kill* dalam Pembuatan Labirin

Arie Tando - 13510018
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13510018@std.stei.itb.ac.id

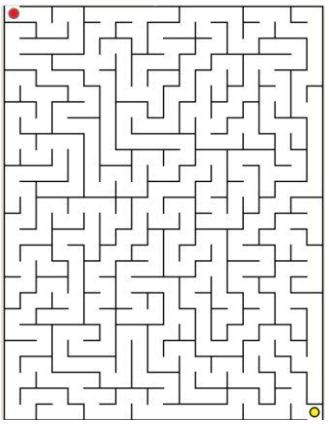
***Abstract*—Labirin adalah sebuah permainan yang direpresentasikan sebagai tempat yang terdiri dari kumpulan jalan yang rumit, bercabang, dan buntu yang dirancang membentuk sebuah teka-teki dimana pemain harus mencari jalan keluar dari tempat tersebut. Salah satu tingkat kesulitan dalam menyelesaikan permainan labirin ditentukan dari ukuran maze tersebut. Semakin besar ukuran suatu labirin maka akan meningkatkan kerumitan dalam pembuatan labirin tersebut. Oleh karena itu sudah terdapat beberapa teknik pembuatan labirin yang dapat dilakukan secara otomatis. Salah satu teknik yang sering digunakan adalah algoritma *Depth-first search* (DFS). Seiring berjalannya waktu, masalah pembuatan labirin dapat deselesaikan secara otomatis dengan beberapa algoritma, salah satunya adalah algoritma *Hunt-and-kill*. Dalam makalah ini akan membahas mengenai perbandingan algorima *Depth-first search* dengan algoritma *Hunt-and-kill* dalam pembuatan labirin.**

***Index Terms*—Labirin, algorima *Depth-first search* (DFS), algoritma *Hunt-and-kill*.**

I. PENDAHULUAN

Labirin atau maze merupakan suatu persoalan yang sangat terkenal di dunia. Dahulu labirin merupakan suatu tempat dengan dinding-dinding yang tinggi, dimana dinding-dinding tersebut ada di sekitar anda dan membuat anda tersesat. Labirin pertama kali sangatlah sederhana yaitu berpola spiral, dimana anda dapat berjalan mengikuti arah hingga masuk kepusat dan keluar lagi ke titik awal. Labirin jenis lainnya adalah hasil kompilasi dari labirin spiral yaitu dengan 1 jalan menuju pusat tanpa ada percabangan dan jalan tersebut berbelok-belok hingga menuju pusat. Seiring dengan berjalannya waktu, labirin bercabang pun dibuat. Labirin bercabang ini dapat membuat anda merasa cukup panjang untuk berjalan, dan labirin ini dapat membuat anda benar-benar tersesat didalamnya.

Pada zaman sekarang ini, sudah banyak sekali permainan yang menggunakan labirin. Permainan labirin ini adalah permainan dimana pada suatu gambar labirin akan terdapat 1 titik awal sebagai tempat bagi pemain untuk mulai menjelajah labirin, dan 1 titik akhir sebagai tujuan akhir yang harus dicapai oleh pemain. Untuk mencapai tujuan maka pemain harus melewati jalan bercabang yang tersedia dan tidak boleh melewati dinding yang biasanya digambarkan dengan garis hitam sebagai pembatas dari jalan. Gambar 1 di bawah ini merupakan contoh dari permainan labirin dengan garis hitam melambangkan dinding-dinding dimana pemain tidak dapat melintasi atau menembus dinding tersebut, titik berwarna merah adalah titik awal dari pemain, dan titik berwarna kuning merupakan titik akhir yang harus dicapai.



***Gambar 1.* Contoh gambar maze**

Pembuatan labirin adalah kegiatan merancang tata letak dari dinding-dinding dalam labirin. Ada banyak pendekatan yang berbeda-beda untuk menghasilkan sebuah labirin. Algoritma komputasi komputer yang paling banyak digunakan dalam pembuatan labirin secara otomatis adalah dengan algoritma *Depth-first search*. Algoritma lain yang dapat digunakan dalam pembuatan labirin ini adalah algoritma *Hunt-and-kill*. Algoritma *Hunt-and-kill* merupakan algoritma yang hampir mirip dengan algoritma *Depth-first search*. Kedua algoritma ini akan dibandingkan dalam pembuatan labirin yang lebih optimum.

II. DASAR TEORI

2.1 Algoritma *Depth-first search (DFS)*

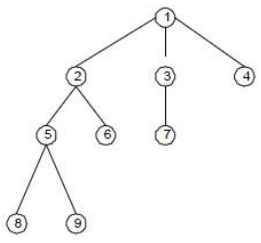
Depth-first search merupakan sebuah algoritma untuk melakukan pencarian solusi pada sebuah pohon. Struktur pohon berakar banyak digunakan dalam merepresentasikan semua kemungkinan solusi dari suatu persoalan agar memudahkan untuk pencarian solusi. Pencarian solusi dilakukan dengan mengunjungi simpul-simpul di dalam pohon. Setiap simpul akan diperiksa apakah solusi telah dicapai. Jika sudah mencapai solusi, maka pencarian solusi selesai, tetapi jika belum sama, maka periksa simpul berikutnya.

Pada umumnya, pohon dibentuk secara dinamis selama pencarian solusi berlangsung. Pohon dinamis membangun simpul-simpul secara bersamaan dengan pencarian solusi. Bila sebuah simpul yang dibentuk merupakan simpul yang buntu atau tidak mengarah ke solusi, maka pencarian solusi dilanjutkan dengan membentuk simpul berikutnya. Metode yang digunakan untuk membentuk pohon selama pencarian solusi salah satunya adalah metode pencarian mendalam (*Depth-first search* atau *DFS*).

Pencarian solusi dengan *DFS* dicirikan dengan ekspansi simpul-simpul terdalam lebih dulu. Algoritma pencarian solusi dengan DFS dimulai dari simpul v:

1. Kunjungi simpul v
2. Kunjungi simpul w yang bertetangga dengan simpul v
3. Ulangi *DFS* mulai dari simpul w
4. Ketika mencapai simpul u sedemikian sehingga semua simpul yang bertetangga dengannya telah dikunjungi, maka dilakukan pencarian runut-balik (*backtrack*) ke simpul terakhir yang memiliki simpul w yang belum dikunjungi
5. Pencarian berakhir bila tidak ada lagi simpul yang belum dikunjungi atau telah mencapai simpul solusi.

Perhatikan pada gambar 2, mula-mula simpul akar dibangkitkan pertama kali yaitu simpul 1, kemudian mencari simpul tetangga dari simpul 1 yaitu simpul 2. Ulangi langkah *DFS* yang dimulai dari simpul 2 lalu mengunjungi simpul 5 dan kemudian simpul 8. Ketika mencapai simpul 8 sedemikian sehingga tidak ada lagi simpul yang dapat dikunjungi maka dilakukan pencarian runut-balik (*backtrack*) ke simpul terakhir yang memiliki tetangga yang belum dikunjungi yaitu simpul 5. Kemudian dari simpul 5 dilakukan pencarian kembali ke simpul 9 dan seterusnya. Hasil penulusuran DFS pada gambar 2 adalah 1,2,5,8,9,6,3,7,4



Gambar 2. **Pohon**

2.2 Algoritma *Hunt-and-Kill*

Algoritma *Hunt-and-Kill* merupakan sebuah algoritma untuk membuat labirin. Cara kerja algoritma ini seperti memburu dan membunuh. Langkah pertama yang dilakukan algoritma ini adalah membuat jalan secara acak sampai tidak ada jalan lain untuk membuat jalan lalu proses ini dibunuh, kemudian algoritma ini memburu sebuah titik yang akan digunakan untuk membuat jalan yang baru dan begitu seterusnya dengan melakukan bunuh (*kill*) dan berburu (*hunt)*.

Langkah-langkah algoritma *Hunt*-and-*Kill* :

1. Menginisialisasi semua sel menjadi belum dikunjungi
2. Memilih lokasi awal secara acak dan menandai sel menjadi telah dikunjungi
3. Memilih arah secara acak lalu pindahkan sel yang berdekatan ke arah itu kecuali sel tersebut telah dikunjungi atau sel tersebut adalah perbatasan
4. Ulangi langkah 3 hingga tidak ditemukan lagi arah yang dapat dikunjungi dan lanjut ke langkah 5.
5. Mulailah melakukan pencarian secara berurutan dari sel yang ada sampai menemukan ada sel yang telah dikunjungi tetapi arahnya masih belum lengkap, lanjutkan ke langkah 3
6. Jika semua sel memiliki arah lengkap maka pembuatan selesai.

III. PEMBAHASAN

Penerapan algoritma Depth-first search dan algoritma Hunt-and-kill bisa dilakukan untuk pembuatan labirin secara otomatis. Pada bagian ini akan dijelaskan bagaimana cara kerja dari kedua algoritma tersebut dalam pembuatan labirin secara otomatis dan perbandingan dari kedua algoritma tersebut.

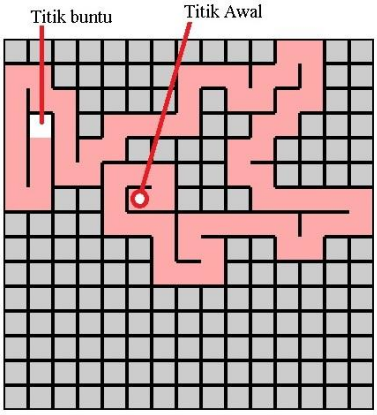
3.1 Algoritma *Depth-first search* (*DFS*)

Langkah pertama dalam membuat sebuah labirin adalah sekumpulan persegi-persegi yang membentuk sebuah persegi besar dengan tujuan sebagai awal dari sebuah labirin. Cara membuat labirin paling sederhana adalah dengan cara menghilangkan atau menghapus garis-garis hitam pada kotak persegi tersebut sedikit demi sedikit sehingga lama kelamaan akan menjadi suatu jalan yang dibatasi dengan garis hitam pada sisinya sebagai dinding yang tidak dapat dilewati.

Penerapan algoritma Depth-first search (*DFS*) dalam pembuatan labirin yang sering diimplementasikan adalah menggunakan metode rekursif atau menggunakan *stack*. Cara ini merupakan salah satu cara tersingkat untuk membentuk labirin menggunakan komputer. Langkah pertama yang dilakukan dengan algoritma *DFS* adalah menentukan simpul v pertama kali sebagai akar. Simpul v ini dapat dipilih secara acak maupun dapat ditentukan dari titik awal. Simpul v yang telah dibuat akan disimpan dalam sebuah *stack* dan titik yang menjadi posisi pada simpul v tersebut akan diberi tanda bahwa titik tersebut

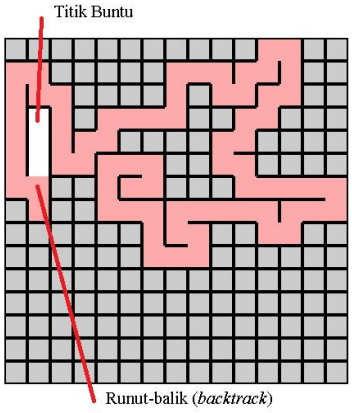
telah dikunjungi.

Langkah selanjutnya dari algoritma *DFS* adalah mencari simpul yang bertetangga dengan v dengan memilih salah satu titik dari keempat arah (atas, bawah, kiri, dan kanan). Kemudian jika telah menentukan titik tersebut secara acak, maka garis pembatas dari titik tersebut akan dihapus dan titik tersebut akan disimpan didalam *stack.* Titik yang telah dilalui akan diberi tanda yang sama seperti titik sebelumnya sebagai penanda bahwa titik tersebut telah dikunjungi. Langkah ini akan dilakukan berulang-ulang hingga suatu saat menemukan suatu jalan dimana tidak ada jalan lagi yang dapat dibangun seperti pada gambar 3.



Gambar 3. **Pembentukan Labirin ketika menemukan jalan buntu dengan algoritma *DFS***

Pada gambar 3, posisi awal dipilih secara acak yaitu titik awal dan dilakukan pembuatan jalan sedemikian rupa hingga menemukan jalan buntu yaitu titik buntu. Dikatakan titik buntu karena pada keempat arahnya (atas, bawah, kiri, dan kanan) telah dilalui sehingga titik tersebut akan dianggap telah mati. Proses yang akan dilakukan komputer dengan algoritma ini adalah melakukan runut-balik (*backtrack*) dari jalur yang pernah dilalui sebelumnya. Setiap kali melakukan runut-balik maka titik-titik tersebut akan dilakukan pengecekan apakah tetangganya ada yang belum dikunjungi, dari titik itulah akan diteruskan untuk pembentukan jalan selanjutnya seperti pada gambar 4 yaitu titik runut-balik yang diteruskan ke arah bawah.



Gambar 4. **Proses runut-balik dalam pembuatan labirin dengan algoritma *DFS***

Langkah-langkah tersebut akan diulang-ulang hingga seluruh kotak pada labirin tersebut akan dikunjungi semua. Dengan ini maka seluruh bagian dari kotak akan ditelusuri dengan menggunakan runut-balik (*backtrack*) hingga mencapai titik awal yang berarti bahwa seluruh isi *stack* telah habis dan program akan berhenti. Jika seluruh isi *stack* telah habis maka dapat dipastikan labirin telah selesai dibuat.

Proses rekursif pada pembentukan labirin dengan menggunakan algoritma *DFS* terdapat pada bagian pembuatan jalur labirin. Langkah-langkah yang dilakukan algoritma DFS adalah :

1. Mulai dari suatu kotak yang dipilih secara acak menjadi titik awal
2. Memberi tanda bahwa kotak tersebut telah dikunjungi, kemudian mencari tetangga dari kotak yang sedang ditempatinya
3. Jika kotak dari tetangga tersebut belum pernah dikunjungi, maka pilih kotak tetangga tersebut dan dibuat jalan dengan menghapus dinding yang membatasi antara kotak sekarang dengan kotak tetangga yang dipilih sebelumnya. Kemudian lakukan langkah ini secara rekursif dengan kotak awal adalah kotak tetangga yang dipilih sebelumnya.

Berdasarkan langkah-langkah diatas, algoritma ini memiliki proses rekursif yang cukup dalam jika ukuran dari labirin yang dibuat cukup besar.

Proses runut-balik terjadi ketika suatu jalan telah buntu. Langkah-langkah yang dilakukan pada proses runut-balik adalah :

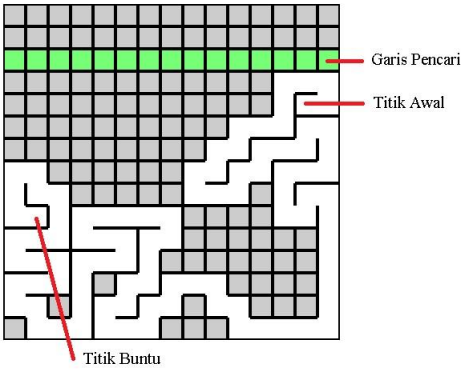
1. Ketika suatu kotak dipilih, maka akan ditandai bahwa kotak itu sudah dikunjungi
2. Ketika suatu kotak yang sedang diduduki tidak memiliki tetangga lagi maka diambil kotak sebelumnya yang telah dilalui
3. Kotak yang telah diambil tersebut dijadikan kotak yang memiliki posisi sekarang. Jika kotak tersebut memiliki tetangga maka lakukan kembali langkah 1 dengan kotak pada posisi sekarang, jika tidak maka kembali ke langkah 2 dengan memilih kotak pada langkah sebelumnya.

3.2 Algoritma *Hunt-and-Kill*

Algoritma ini cukup menarik dalam pembentukan sebuah labirin. Sesuai dengan namanya, algoritma ini menggunakan metode seperti berburu dan membunuh proses dalam pembuatan labirin pada komputer. Algoritma ini memiliki kemiripan dengan algoritma *DFS* tetapi berbeda dalam pembentukan labirinnya. Langkah-langkah yang dilakukan algoritma *Hunt-and-Kill* pada pembuatan labirin adalah :

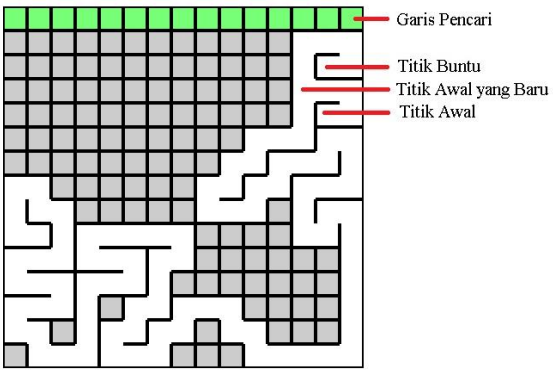
1. Tentukan titik awal secara acak, lalu pilih secara acak arah yang akan dilalui
2. Mengahapus garis pembatas antara titik yang sekarang dengan titik yang dipilih pada langkah sebelumnya

3. Membuat titik yang dipilih tadi menjadi titik yang sekarang
4. Lakukan berulang-ulang hingga tidak ada lagi arah yang dapat dilalui
5. Lakukan pencarian dari batas atas labirin hingga batas bawah labirin, cari titik pertama yang ditemukan dimana titik tersebut memiliki arah yang belum dikunjungi
6. Membuat jalur dari titik yang ditemukan pertama kali dan menggantikan titik tersebut sebagai titik sekarang
7. Ulangi kembali langkah 1 sampai tidak ada lagi titik yang memiliki tetangga lagi.



Gambar 5. **Pembentukan labirin jika mengalami jalan buntu pada algoritma *Hunt-and-Kill***

Perhatikan pada gambar 5, titik awal merupakan titik pertama kali dalam pembuatan jalur. Pembuatan jalur mirip dengan teknik pada algoritma *DFS* tetapi jalur yang telah dilalui tidak disimpan secara rekursif tetapi hanya dilakukan pengulangan. Pembuatan jalur dilakukan secara acak hingga ditemukan titik buntu. Setelah ditemukan titik buntu maka dilakukan pencarian dari atas ke bawah yang dilakukan oleh garis pencari hingga menemukan titik pertama yang memiliki tetangga tetapi belum dikunjungi. Selanjutnya dari titik hasil pencarian tersebut dilakukan pembuatan jalur yang baru dengan titik awal baru seperti pada gambar 6.



Gambar 6. **Pembuatan Labirin dengan titik awal yang baru pada algoritma *Hunt-and-Kill***

Proses dari algoritma *Hunt-and-Kill* ini cukup mudah untuk diimplementasikan. Implementasi yang pertama kali dilakukan adalah menentukan titik awal secara acak dan kemudian melakukan pengulangan yang dijalankan dengan dua fase, yaitu fase berjalan dan fase berburu. Program dengan algoritma ini akan berhenti jika dalam fase berburu sudah tidak ditemukan lagi titik awal yang belum memiliki tetangga yang dikunjungi. Implementasi fase berjalan hampir mirip dengan implementasi pencarian jalur pada algoritma *DFS*. Fase berjalan mengembalikan nilai dari posisi x dan y yang berada di posisi sekarang. Jika dalam fase berjalan mendapatkan nilai kembalian berupa tidak ada jalan yang dapat dibuat lagi, maka program akan menggantikan fase menjadi fase berburu yang akan mencari titik awal baru dengan kembalian berupa titik awal yang baru. Setelah mendapatkan nilai dari titik awal yang baru maka program akan melanjutkan pembuatan jalur dan proses ini akan berulang-ulang hingga fase berburu tidak menghasilkan titik yang baru.

3.3 Perbandingan Algoritma *DFS* dan Algoritma *Hunt-and-Kill* pada Pembentukan Labirin

Algoritma *DFS* dan algoritma *Hunt-and-Kill* merupakan algoritma yang dapat membuat labirin secara otomatis dengan prinsip dan cara kerja yang berbeda. Algoritma yang sangat dibutuhkan dalam pembuatan labirin adalah algoritma yang cepat dan efisien dalam hal waktu pembentukan labirin. Faktor-faktor yang perlu diperhatikan dalam pembuatan labirin adalah jumlah dari hasil komputasi dalam dilakukannya. Berikut adalah perbandingan antara algoritma *DFS* dan algoritma *Hunt-and-Kill* dalam pembentukan labirin.

Algoritma *DFS* adalah algoritma yang melakukan pembentukan labirin dengan membentuk pohon ruang status secara mendalam hingga mencapai simpul buntu dan melakukan runut-balik hingga menemukan simpul yang belum buntu dan meneruskan pembuatan simpul dari simpul yang telah diperoleh pada saat pencarian runut-balik. Proses menggunakan algoritma *DFS* ini melakukan pengulangan setiap kali membuat 1 kotak jalan baru menggunakan proses rekursif dan jika terdapat jalan buntu maka secara otomatis akan melakukan proses runut-balik dari proses rekursif yang telah dibuat. Proses dengan algoritma *DFS* ini memakan waktu yang cukup lama pada bagian proses runut-balik karena jika telah mencapai titik buntu maka program harus mengecek ulang untuk setiap jalur yang telah dilalui sebelumnya.

Algoritma *Hunt-and-Kill* adalah algoritma yang melakukan pembentukan labirin dengan membentuk jalan pada labirin yang mirip dengan algoritma *DFS* tetapi tidak dilakukan secara rekursif tetapi hanya dilakukan dengan pengulangan. Ketika mencapai titik buntu maka fase akan berpindah menjadi fase *hunt* yang melakukan pencarian kotak dengan melakukan iterasi dari batas atas labirin ke batas bawah labirin. Jika ditemukan kotak yang dapat menjadi titik awal yang baru maka fase *hunt* berakhir dan dilanjutkan dengan pembuatan jalan.

Proses pembentukan jalur labirin pada algoritma *DFS* dan algoritma *Hunt-and-Kill* memiliki hasil komputasi

yang tidak berbeda jauh. Tetapi pada proses fase *hunt* pada algoritma *Hunt-and-Kill* lebih cepat dibandingkan dengan proses runut-balik pada algoritma *DFS* karena pada algoritma *Hunt-and-Kill* cukup mencari kotak secara iterasi dari atas sedangkan pada algoritma *DFS* harus melakukan runut-balik yang memerlukan waktu yang lebih lama.

IV. KESIMPULAN

Dari hasil pembahasan di atas mengenai pembentukan labirin secara otomatis menggunakan algoritma *DFS* dan algoritma *Hunt-and-Kill*, maka didapat beberapa kesimpulan sebagai berikut :

1. Pembentukan labirin dengan algoritma *Hunt-and-Kill* lebih cepat dibandingkan dengan algoritma *DFS*
2. Pembentukan labirin dengan algoritma *DFS* dan algoritma *Hunt-and-Kill* memiliki kekurangan pada pembuatan jalur labirin karena dengan kedua algoritma tersebut hanya membuat jumlah percabangan jalur yang sedikit
3. Dalam pembentukan labirin terdapat beberapa algoritma lain selain algoritma *DFS* dan algoritma *Hunt-and-Kill*.

DAFTAR PUSTAKA

[1] Munir, Rinaldi. Diktat *Kuliah IF3051 Strategi Algoritma*. 2010.

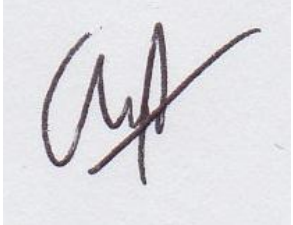
[2] http://weblog.jamisbuck.org/2011/1/24/maze-generation-hunt-and-kill-algorithm
Tanggal Akses: 16 Desember 2012

[3] http://weblog.jamisbuck.org/2010/12/27/maze-generation-recursive-backtracking
Tanggal Akses: 16 Desember 2012

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 20 Desember 2012



Arie Tando - 13510018