Algoritma Prim sebagai *Maze Generation Algorithm*

Muhammad Ecky Rabani/13510037
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
ecky_dozha@yahoo.co.id

***Maze* secara kasar ditranslasikan ke dalam bahasa Indonesia oleh program berbasis *web* Google-Terjemahan kedalam kata "membingungkan". Jika kita melihat di kamus Inggris-Indonesia, maka *maze* disini berarti "jaringan jalan yang ruwet atau membingungkan". *Maze* adalah salah satu bentuk *tour-puzzle* dimana sang *solver* harus mencari rute dari titik awal ke titik akhir melalui jalan-jalan yang bercabang dan sulit untuk dinavigasi. Aplikasi *maze* biasanya ada dalam industri *game* baik digital maupun analog, bahkan *maze* ukuran besar dapat dijadikan *tourist attraction*. Dalam makalah ini akan dibahas suatu algoritma yang dapat digunakan oleh kita dalam membuat sebuah *maze* yang ideal.**

***Kata Kunci : Maze*, Algoritma Prim, Pohon Merentang Minimum, Graf**

I. PENDAHULUAN

Gambar 1 : *Maze* raksasa Longleat yang merupakan salah satu *tourist attraction* di Inggris

Maze adalah salah satu jenis *puzzle* yang menuntut sang pemain untuk melakukan perjalanan pada papan *puzzle*-nya meskipun tidak selamanya hal tersebut harus berbentuk dua dimensi atau disebut juga *tour-puzzle*. Pemain bisa saja menelusuri jalannya dengan representasi benda seperti dadu, menggunakan pensil atau *token* lainnya, namun tidak menutup kemungkinan pemain itu sendiri-lah yang harus menelusuri jalan tersebut. Pada *maze*, pemain biasanya diharuskan menelusuri jalan dari titik awal (*start*) sampai titik akhir (*finish*). Di dalam sebuah *maze*, jalan yang harus ditelusuri oleh pemain bercabang-cabang dan dirancang agar sulit untuk dinavigasi agar tercipta suatu *puzzle* yang menantang. Aplikasi dari *maze* ini banyak sekali, contohnya papan permainan untuk anak-anak maupun keluarga. Bahkan ada *maze* dalam ukuran besar yang memang dibuat sebagai suatu wahana permainan turis atau sebagai penghias taman saja. Dalam versi digital-nya, *maze* dapat diaplikasikan pada *game* ber-*genre puzzle*, atau hanya sebagai fitur pendukung agar *game* menjadi lebih menantang dan tidak membosankan.

Gambar 2 : *Maze* di dalam salah satu judul *video game*

Sebuah *maze* biasanya terdiri dari suatu ruangan dan tembok. Tembok inilah yang akan membagi ruangan tersebut menjadi jalan-jalan bercabang yang harus ditelusuri oleh pemain. Dalam pembuatan sebuah *maze*, yang pertama kali dilakukan adalah mendesain *layout* dari lintasan yang akan ditempuh dari titik *start* sampai titik *finish*, kemudian membuat jalan tambahan yang dibuat bercabang-cabang untuk meningkatkan tingkat kesulitan dari *puzzle*, kemudian membuat tembok berdasarkan *layout* lintasan yang telah dibuat dan membuat ruangan untuk ditelusuri oleh pemain sesuai lintasannya. Banyak cara untuk mempermudah kita dalam membuat sebuah *maze*. Salah satunya adalah menggunakan algoritma pembuat *maze* (*Maze Generation Algorithm*), algoritma ini bisa dilakukan manual dengan tangan maupun otomatis dilakukan dan dibuat oleh komputer. Termasuk di dalam algoritma pembuat *maze* adalah algoritma Kruskal dan algoritma Prim karena pembuatan *maze* berdasarkan

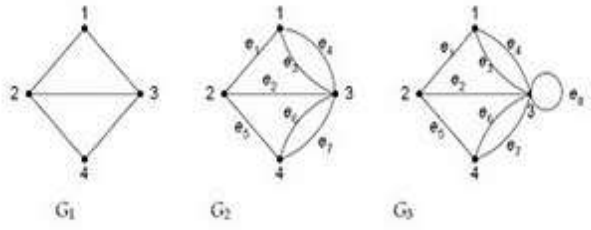
kepada teorema pembuatan pohon merentang minimum (*Minimum Spanning Tree*) dari suatu graf terhubung seperti yang sudah diajarkan di kuliah IF2091 Struktur Diskrit.

.

II. DASAR TEORI

1. Graf

Teori graf merupakan pokok bahasan yang sudah tua usianya namun memiliki banyak terapan sampai saat ini. Graf digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut. Representasi visual dari graf adalah dengan menyatakan objek sebagai noktah, bulatan, atau titik, sedangkan hubungan antara objek dinyatakan sebagai garis.



Gambar 3 : Contoh berbagai bentuk graf

Secara matematis, suatu graf G didefinisikan sebagai pasangan himpunan (V,E) yang dalam hal ini :

V = himpunan tidak kosong dari simpul-simpul (*vertices* atau *node*) = $\{v_1,v_2,...,v_n\}$, dan
E = himpunan sisi (*edges* atau *arcs*) yang menghubungkan sepasang simpul = $\{e_1,e_2,...,e_n\}$.

Atau dapat ditulis juga secara singkat notasinya menjadi G = (V,E).

Pada contoh gambar tiga graf diatas, graf G_1, G_2, dan G_3 dapat kita nyatakan himpunan simpul V dan himpunan sisi E-nya sebagai :

G_1 = (V,E) dengan
V = {1,2,3,4}
E = { (1,2), (1,3), (2,3), (2,4), (3,4) }

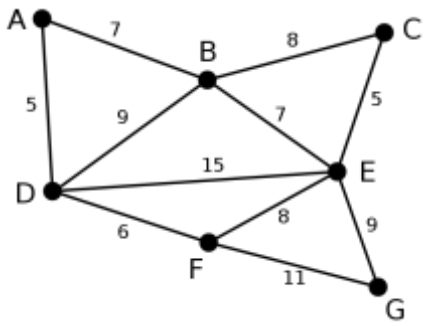
G_2 = (V,E) dengan
V = {1,2,3,4}
E = { (1,2), (2,3), (1,3), (1,3), (2,4), (3,4), (3,4) }
= { $e_1,e_2,e_3,e_4,e_5,e_6,e_7$ }

G_3 = (V,E) dengan
V = {1,2,3,4}
E = { (1,2), (2,3), (1,3), (1,3), (2,4), (3,4), (3,4), (3,3) }
= { $e_1,e_2,e_3,e_4,e_5,e_6,e_7,e_8$ }

Graf juga dapat dikelompokkan menjadi beberapa kategori atau jenis bergantung pada sudut pandang pengelompokannya. Sebagai contoh pengelompokkan graf berdasarkan ada tidaknya gelang atau sisi ganda pada suatu graf, yaitu graf sederhana (*simple graph*) dan graf tak-sederhana (*unsimple-graph*). Graf sederhana tidak mengandung gelang maupun sisi ganda seperti pada graf G_1 pada gambar diatas. Sebaliknya, graf yang mengandung sisi ganda atau gelang dinamakan graf tak-sederhana seperti pada graf G_2 dan G_3 pada gambar diatas.

Terminologi atau istilah-istilah dalam graf yang penting dan sering dipakai salah satunya adalah bertetangga (*adjacent*), bersisian (*incident*), lintasan (*path*), dan sirkuit (*circuit*). Dua buah simpul pada suatu graf G dikatakan bertetangga bila keduanya terhubung langsung dengan sebuah sisi. Sebagai contoh pada graf G_1 pada gambar diatas, simpul 1 bertetangga dengan simpul 2 dan 3, tetapi tidak bertetangga dengan simpul 4. Untuk sembarang sisi $e = (v_i,v_j)$, sisi e dikatakan bersisian dengan simpul v_i dan simpul v_j. Sebagai contoh pada graf G_1, sisi (2,3) bersisian dengan simpul 2 dan simpul 3, begitu juga sisi (2,4) bersisian dengan simpul 2 dan simpul 4, tetapi sisi (1,2) tidak bersisian dengan simpul 4. Menuliskan lintasan pada graf cukup dituliskan sebagai barisan simpul-simpul saja pada graf sederhana. Sebagai contoh pada graf G_1, lintasan 1,2,3,4 adalah lintasan dengan barisan sisi (1,2), (2,4), (4,3). Lintasan yang berawal dan berakhir pada simpul yang sama disebut sirkuit atau siklus. Pada graf G_1 pada gambar diatas, lintasan 1,2,3,1 adalah sebuah sirkuit.

Salah satu jenis graf khusus adalah graf berbobot (*weighted graph*). Graf berbobot adalah graf yang setiap sisinya diberi sebuah harga atau bobot seperti contoh pada gambar dibawah ini.



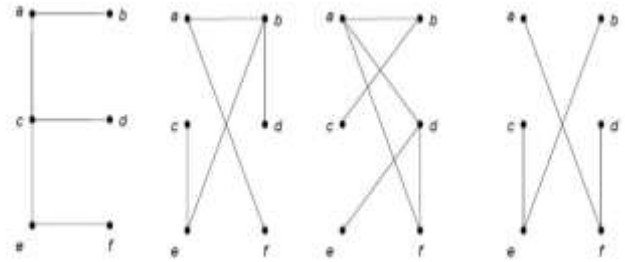
Gambar 4 : Contoh graf berbobot (*weighted graph*)

Bobot pada setiap sisi dapat menyatakan jarak antara dua buah simpul, biaya perjalanan antara dua buah kota, waktu tempuh pesan, ongkos produksi, dan sebagainya.

2. Pohon

Diantara sekian banyak konsep dalam teori graf, konsep pohon (*tree*) mungkin merupakan konsep yang paling penting, khususnya bagi siapapun yang tertarik dengan penerapan graf. Pohon termasuk kedalam graf jenis khusus. Definisi pohon adalah graf tak-berarah terhubung yang tidak mengandung sirkuit. Menurut definisi, ada dua

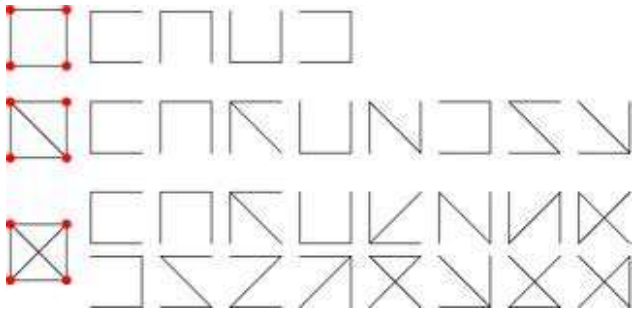
sifat penting pada pohon yaitu terhubung dan tidak mengandung sirkuit. Berikut adalah gambar dari beberapa graf yang dapat disebut pohon dan juga tidak.



Gambar 5 : Contoh graf berupa pohon dan bukan pohon

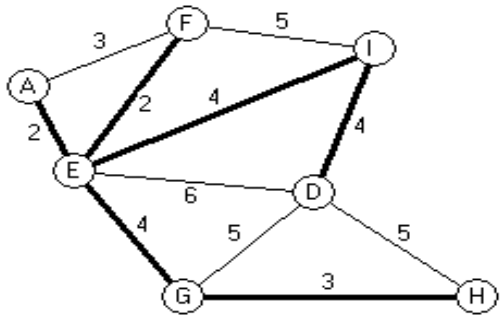
Dua graf paling kiri pada gambar diatas adalah graf berupa pohon karena tidak mengandung sirkuit dan semua simpul terhubung. Sedangkan, pada dua graf paling kanan pada gambar diatas bukanlah berupa pohon karena pada graf ketiga dari kiri terdapat sirkuit walaupun semua simpul terhubung yaitu sirkuit a,d,f,a. Pada graf keempat dari kiri, tidak semua simpul terhubung meskipun tidak ada sirkuit di dalamnya, graf dengan simpul c,e,b tidak terhubung dengan graf dengan simpul a,f,d.

Setelah mengetahui definisi dari pohon, salah satu istilah dalam konsep pohon adalah pohon merentang (*spanning tree*). Misalkan G = (V,E) adalah graf tak-berarah terhubung yang bukan pohon, yang berarti di graf G terdapat beberapa sirkuit, G dapat diubah menjadi pohon T = (V_1,E_1) dengan cara memutuskan sirkuit-sirkuit yang ada. Hal ini dilakukan berulang-ulang hingga tidak ada sirkuit lagi di dalam graf G, maka setelah hal tersebut selesai dilakukan, graf G menjadi sebuah pohon T yang dinamakan pohon merentang.



Gambar 6 : Contoh graf terhubung (paling kiri) dan semua pohon merentangnya

Pada graf berbobot yang terhubung, dikenal istilah pohon merentang minimum (*minimum spanning tree*). Jika G adalah graf berbobot, maka bobot pohon merentang T dari G didefinisikan sebagai jumlah bobot semua sisi di T. Pohon merentang yang berbeda memiliki bobot yang berbeda pula. Di antara semua pohon merentang di G, pohon merentang yang berbobot minimum dinamakan pohon merentang minimum. Pohon merentang minimum merupakan pohon merentang yang paling penting dan memiliki terapan yang luas dalam prakteknya. Seperti contoh pada graf berbobot yang terhubung pada gambar dibawah ini, pohon merentang minimum ditandai dengan garis tebal, dan bobot dari pohon sebesar 19. Besar bobot pohon merentang tersebut jika dibandingkan dengan pohon merentang lain dari graf dibawah merupakan yang terkecil, besar bobot pohon merentang yang lainnya dari graf dibawah ini adalah 25, 27, 28, dan lain-lain.



Gambar 7 : Pohon merentang minimum dari graf berbobot yang terhubung

3. Algoritma Prim

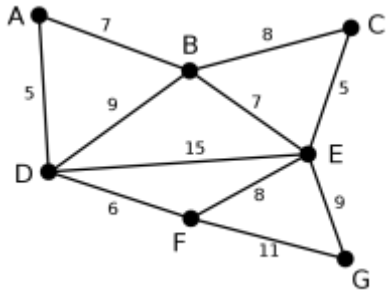
Dalam membangun sebuah pohon merentang minimum dari suatu graf berbobot yang terhubung, dikenal dua algoritma yang banyak digunakan oleh para *engineer*. Yang pertama adalah algoritma Prim dan yang kedua adalah algoritma Kruskal. Pada makalah ini hanya akan dibahas mengenai algoritma Prim, karena yang akan digunakan sebagai *maze generation algorithm* pada penelitian dalam makalah ini adalah algoritma Prim saja.

Algoritma Prim membentuk pohon merentang minimum langkah per langkah. Pada setiap langkah kita mengambil sisi dari suatu graf G yang mempunyai bobot minimum namun terhubung dengan pohon merentang minimum T yang telah terbentuk. Berikut adalah algoritma lengkapnya,

- Ambil sisi dari graf G yang berbobot minimum, masukkan dalam T.
- Pilih sisi (u,v) yang mempunyai bobot minimum dan bersisian dengan simpul di T, tetapi (u,v) tidak membentuk sirkuit di T. Tambahkan (u,v) ke dalam T.
- Ulangi langkah kedua sebanyak n-2 kali dengan n adalah jumlah simpul pada graf.

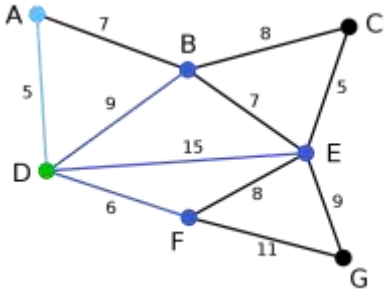
Jumlah langkah seluruhnya di dalam algoritma Prim adalah 1+(n-2) = n-1. Untuk keperluan pembuatan *maze*, algoritma Prim ini akan dimodifikasi sedikit, pada langkah awal dari algoritma Prim, kita tidak mengambil sisi dari graf G yang berbobot minimum, melainkan memilih satu simpul secara *random* dari graf G, kemudian melihat semua sisi yang berbobot minimum dan bersisian dengan simpul tersebut, kemudian menambahkannya kedalam

pohon T. Sebagai contoh pada graf berbobot pada gambar nomor 4 diatas, akan kita buat pohon merentang minimum-nya menggunakan algoritma Prim. Berikut adalah susunan gambar-gambar yang menunjukkan langkah per langkah membangun pohon rentang minimum dari graf pada gambar diatas.



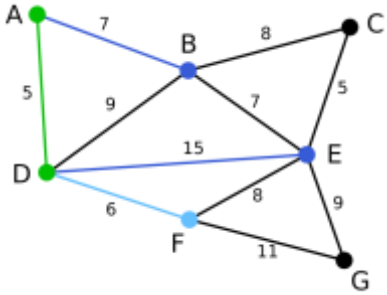
Gambar 8 : Membuat pohon merentang minimum (langkah-0)

Ini adalah kondisi awal dari graf berbobot pada gambar nomor 4 diatas. Graf ini bukanlah sebuah pohon karena terdapat sirkuit di dalamnya. Sekarang kita pilih simpul D secara sembarang sebagai titik awal.



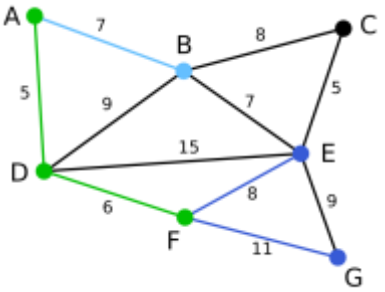
Gambar 9 : Membuat pohon merentang minimum (langkah-1)

Sisi pertama yang akan kita tambahkan ke dalam pohon merentang minimum T adalah sisi yang berbobot minimum dan bersisian dengan simpul D. Dari keempat sisi yang ada, sisi (D,A) berbobot paling kecil yaitu sebesar 5, karena itu sisi tersebut kita tambahkan ke dalam T (ditanda warna biru muda).



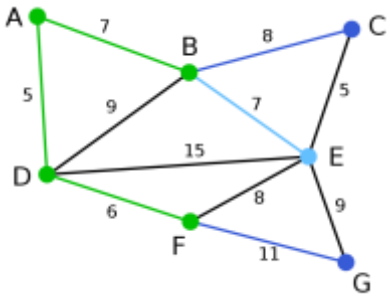
Gambar 10 : Membuat pohon merentang minimum (langkah-2)

Langkah diatas kita ulangi, tetapi kali ini kita mencari sisi berbobot minimum yang bersisian dengan simpul D dan A, yaitu sisi (D,F) yang memiliki besar bobot 6.



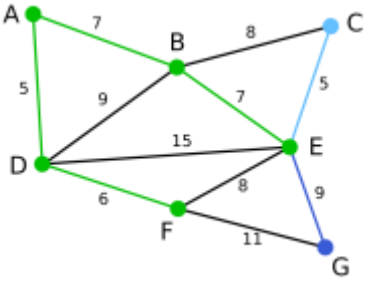
Gambar 11 : Membuat pohon merentang minimum (langkah-3)

Langkah berikutnya dalam algoritma Prim ini sama seperti sebelumnya. Kita mencari sisi yang memiliki bobot minimum dan bersisian dengan semua simpul dalam T (A, D, dan F). 7 bernilai paling kecil dalam hal ini, maka sisi (A,B) ditambahkan ke dalam T.



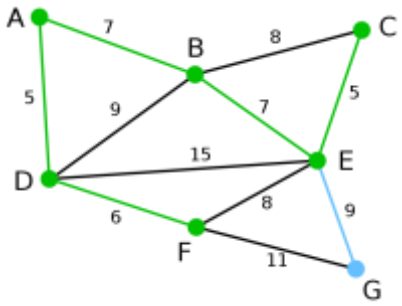
Gambar 12 : Membuat pohon merentang minimum (langkah-4)

Pada langkah ini, kita mengambil sisi (B,E) untuk ditambahkan ke dalam T karena memiliki besar bobot paling kecil yaitu 7.



Gambar 13 : Membuat pohon merentang minimum (langkah-5)

Pada langkah ini, kita mengambil sisi (E,C) untuk ditambahkan ke dalam T karena memiliki besar bobot paling kecil yaitu 5.

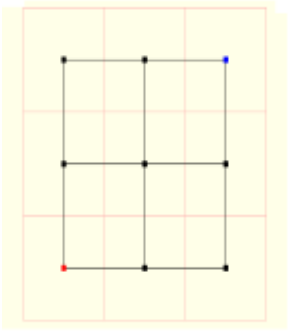


Gambar 14 : Membuat pohon merentang minimum (langkah-6)

Ini adalah langkah terakhir karena setelah langkah ini, jumlah sisi di dalam pohon T akan berjumlah n-1 (dalam hal ini graf memiliki 7 simpul, maka n-1 = 7-1 = 6). Pada kasus ini sisi yang berbobot minimum dan bersisian dengan semua simpul dalam pohon T adalah sisi yang memiliki besar bobot 8, namun kita tidak bisa menambahkan sisi (F,E) maupun (B,C) ke dalam pohon T karena dapat menyebabkan terjadinya sirkuit dalam pohon T, karena itu kita memilih sisi yang memiliki besar bobot 9. Sisi yang kita ambil adalah sisi (E,G) dan bukan (B,D) karena dapat menyebabkan terjadinya sirkuit. Maka selesailah langkah kita untuk menentukan pohon rentang minimum dari graf pada gambar nomor 4 di atas. Pohon rentang T yang telah kita hasilkan memiliki besar bobot 39 dan merupakan besar bobot yang paling minimum.

III. PEMBUATAN *MAZE* MENGGUNAKAN ALGORITMA PRIM

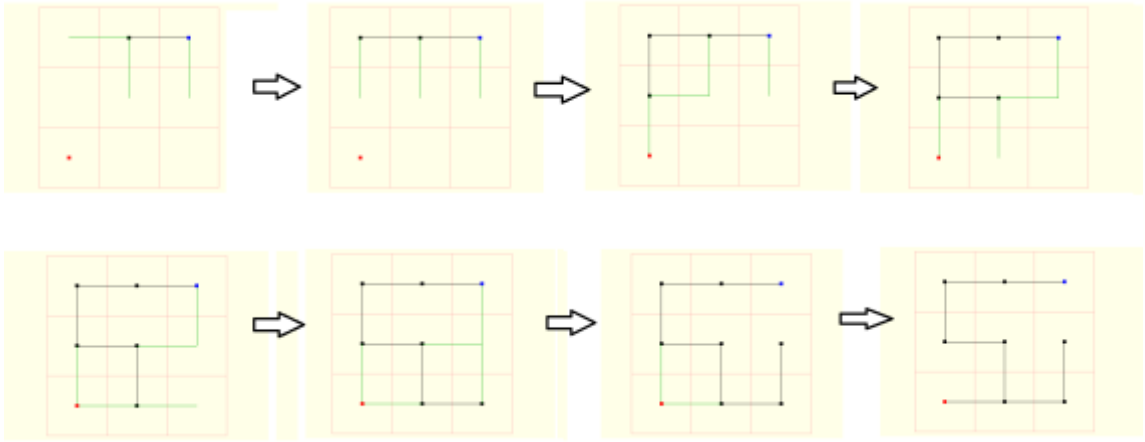
Seperti yang telah dijelaskan sebelumnya tentang *maze*, sebuah *maze* biasanya terdiri dari suatu ruangan dan tembok. Pertama-tama kita buat sebuah ruangan penuh tembok yang membagi luas dari satu ruangan besar tersebut menjadi beberapa ruangan lagi (luasnya tidak harus sama besar), akan terbentuk sebuah *grid* dengan ukuran bermacam-macam tergantung dari cara kita membagi ruangan tersebut. Sekarang anggap satu *cell* ruangan sebagai simpul, maka kita akan mendapat jumlah simpul sebanyak n, dimana n adalah total ruangan yang telah kita buat berdasarkan pembagian ruangan yang telah dilakukan tadi. Kemudian setiap simpul kita hubungkan dengan simpul diatasnya, di kanan dan kiri-nya, juga dibawahnya dengan cara membuat sebuah sisi, dengan begini akan terbentuk sebuah graf terhubung dan bukan berupa pohon. Graf ini disebut sebagai graf dasar untuk lintasan dari *maze* (sebut saja graf Dm). Sebagai contoh akan kita aplikasikan pada *maze* yang masih sederhana dengan ukuran *grid* 3x3 seperti pada gambar dibawah ini.



Gambar 15 : Sebuah *maze* dengan 9 ruangan (ukuran 3x3) dan graf dasar lintasannya

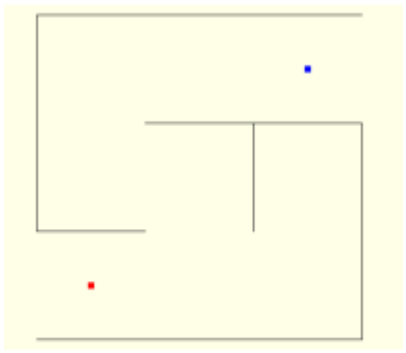
Setelah membuat graf Dm, kita dapat membuat lintasan untuk *maze* yang nantinya akan digunakan oleh sang pemain dalam bernavigasi mencari titik *finish* dari titik *start*. Lintasan tersebut kita buat dengan menerapkan algoritma Prim dan membuat pohon rentang minimum dari graf Dm. Dalam hal ini kita memakai *randomized-version* dari algoritma Prim karena kita anggap semua bobot dari sisi memiliki besar yang sama, sehingga kita memilih sisi secara random untuk ditambahkan ke dalam pohon lintasan *maze* (sebut saja pohon Tm) selama tidak membentuk sirkuit di dalam pohon Tm. Pertama, seperti dalam algoritma Prim, pilih satu simpul secara *random* sebagai titik *start* dan tambahkan sisi berbobot minimum yang bersisian dengan simpul tersebut ke dalam pohon Tm, kemudian pilih juga titik *finish*, dan tandai simpul yang bertindak sebagai titik *finish*. Pada gambar diatas, simpul sebagai titik *start* ditandai dengan warna biru, sedangkan simpul yang bertindak sebagai titik *finish* ditandai dengan warna merah. Gambar dibawah akan menjelaskan langkah demi langkah membuat pohon Tm dari graf terhubung Dm.

Pohon lintasan Tm ditandai dengan garis berwarna hitam pada gambar dibawah ini. Untuk garis-garis berwarna hijau, adalah sisi yang bersisian dengan semua simpul pada pohon Tm. Karena terdapat 9 simpul pada graf Dm, maka total langkah dari algoritma Prim untuk kasus ini berjumlah 9-1 = 8 kali. Setiap sisi yang diambil sifatnya *random* karena semua sisi besar bobotnya sama selama sisi yang diambil tidak membuat sirkuit di dalam pohon Tm, sehingga dari *grid* ukuran 3x3 ini bisa dibuat pohon lintasan Tm yang berbeda yang akan berbobot sama, hal ini tentunya mengakibatkan bentuk *maze* yang bermacam-macam. Selesailah sudah pengaplikasian algoritma Prim sebagai algoritma pembentuk *maze* atau *maze generation algorithm* yang dapat membuat berbagai bentuk *maze*.



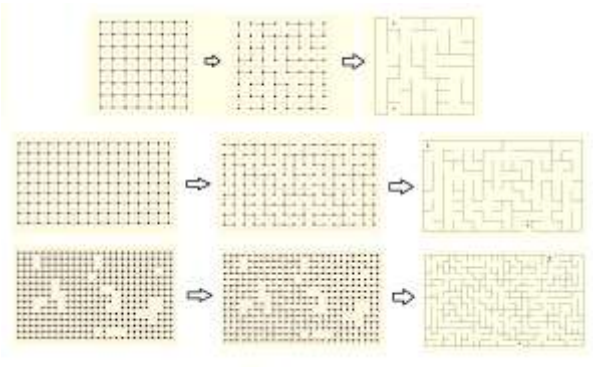
Gambar 16 : Langkah-langkah membuat pohon lintasan Tm dari graf Dm menggunakan *randomized-version* algoritma Prim

Setelah terbentuk pohon lintasan Tm, maka selesai sudah tahap mendesain *layout* dari lintasan dan juga jalan-jalan bercabang yang merupakan unsur utama dari *maze*, terakhir kita bentuk *maze* secara utuh dengan *layout* lintasan yang tadi kita buat menggunakan algoritma Prim, alhasil jadilah *maze* dengan ukuran *grid* 3x3 seperti dibawah ini. Sang *solver* atau pemain tinggal menelusuri jalan tersebut mulai dari titik biru ke titik merah.



Gambar 17 : *Maze* yang telah berhasil diciptakan

Dalam pengembangan tingkat lanjutnya, *maze generation algorithm* bisa dilakukan secara otomatis oleh komputer dengan cara membuat *source-code*-nya. Bahkan dengan *source-code* yang sudah canggih, kita dapat membuat *maze* dengan ukuran *grid* berapa saja (bisa meminta masukan dari *user*) atau membuat *maze* dengan kondisi khusus seperti ada satu ruangan atau simpul yang tidak boleh dilewati oleh pemain. Beberapa contoh program yang sudah jadi ada di beberapa situs internet dan bisa kita pelajari lebih dalam. Aplikasi dari algoritma pembuat *maze* ini banyak sekali, kebanyakan diaplikasikan pada permainan bertipe *puzzle* atau *adventure/role-playing* yang menuntut sang pemain untuk menelusuri *maze* untuk mendapatkan *achievement* tertentu atau naik ke level atau tingkat berikutnya, permainan ini bisa berupa digital maupun analog (*board-game*).



Gambar 18 : *Maze* berbagai ukuran *grid* dan *maze* kondisi khusus dibuat menggunakan algoritma Prim

IV. KESIMPULAN

Membuat suatu *maze* sangatlah mudah, kita hanya butuh mengaplikasikan algoritma Prim yang sudah dimodifikasi sedikit saja. Algoritma Prim adalah salah satu bentuk algoritma pembuat *maze* (*maze generation algorithm*). Dengan algoritma Prim dapat dibuat *maze* dengan ukuran *grid* berapa saja.

REFERENSI

[1] Munir, Rinaldi, Diktat Kuliah IF-2091 Struktur Diskrit, Program Studi Teknik Informatika, STEI, ITB, 2008.

[2] http://www.cut-the-knot.org/ctk/Mazes.shtml (Akses: 10-12-2011)

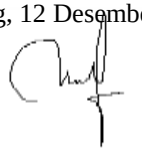
[3] http://weblog.jamisbuck.org/2011/1/10/maze-generation-prim-s-algorithm (sda)

[4] http://en.wikipedia.org/wiki/Prim%27s_algorithm (sda)

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, 12 Desember 2011



Muhammad Ecky Rabani/13510037