Penerapan Algoritma Prim dan Kruskal Acak dalam Pembuatan Labirin

Jason Jeremy Iman 13514058
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13514058@std.stei.itb.ac.id

***Abstrak*—Labirin adalah suatu permainan yang mengharuskan pemain mencari jalan untuk mencapai pintu keluar. Dalam pembuatannya sebuah labirin dapat dibuat dengan komputer maupun secara manual. Dengan komputer labirin dapat direpresentasikan sebagai pohon merentang, maka untuk itu algoritma yang mampu membuat pohon merentang dapat juga membuat suatu labirin. Dalam hal ini akan dibahas algoritma Kruskal dan Prim yang digunakan dengan tanpa bobot dan secara acak dalam membuat sebuah labirin.**

***Keywords*—kruskal, labirin, pohon merentang, prim.**

I. PENGENALAN

Labirin merupakan permainan dimana pemain akan mulai bermain pada tempat awal dan nantinya harus berakhir pada suatu pintu keluar tertentu. Dengan kata lain seorang pemain harus dapat mencari jalan yang menghubungkan pintu masuk dan pintu keluar.

Tingkat kesulitan labirin berbeda-beda tergantung pada kompleksitas labirin tersebut. Biasanya, semakin besar dan banyaknya jalan pada suatu labirin tingkat kesulitannya pun akan meningkat.

Untuk itu pembuatan sebuah labirin yang berukuran besar secara manual dapat memakan waktu yang cukup lama. Dalam hal ini, pembuat labirin harus mampu membuat sebuah jalan yang menghubungkan pintu masuk dan keluar serta membuat jalan-jalan buntu.

Untuk mempermudah pembuatan, sebuah labirin yang sederhana dapat didefinisikan sebagai suatu graf. Sebuah labirin sederhana terdiri dari simpul dan sisi. Tempat dari setiap persegi dalam labirin adalah simpul dan jalan yang mungkin dilewati adalah sisinya. Karakteristik graf lainnya dari sebuah labirin adalah bahwa graf tersebut tidak boleh membentuk sirkuit. Dengan kata lain suatu labirin juga merupakan pohon.

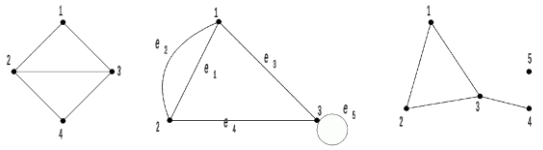
Dengan mengetahui hal tersebut, suatu cara pembuatan labirin adalah dengan membuat pohon merentang dari sebuah graf. Dalam hal ini, terdapat dua buah algoritma yang dapat dipilih, yaitu algoritma Kruskal dan algoritma Prim.

Yang menjadi perbedaan adalah bahwa pada umumnya, kedua algoritma tersebut akan menghasilkan sebuah pohon merentang minimum, namun dalam pembuatan labirin sisitidak berbobot. Jadi, penggunaan kedua algoritma tersebut akan sedikit berbeda dengan adanya pemilihan secara acak dan bukan terhadap bobot dari sisi graf.

II. TEORI GRAF

A. Definisi Graf

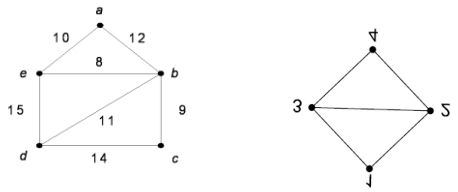
Graf adalah hubungan antara simpul dan sisi. Di mana terdiri dari minimal sebuah simpul. G = (V,E), dengan V adalah himpunan simpul yang tidak kosong dan E adalah himpunan sisi yang masing-masing anggotanya menghubungkan 2 buah simpul.



Gambar 2.1: Beberapa contoh graf

B. Jenis-Jenis Graf

1. Graf Sederhana
   Graf sederhana merupakan graf yang tidak mengandung 2 buah sisi yang menghubungkan simpul yang sama.
2. Graf Tak-Sederhana
   Graf tak-sederhana merupakan graf yang dapat mengandung 2 buah sisi yang menghubungkan simpul yang sama.
3. Graf Berarah
   Graf berarah adalah graf yang setiap sisinya memiliki arah.
4. Graf Tidak Berarah
   Graf tidak berarah adalah graf yang tidak memiliki sisi yang memiliki arah.
5. Graf Berbobot
   Graf berbobot adalah graf yang memiliki nilai-nilai dari tiap sisinya. Bobot dapat mengartikan banyak hal seperti jarak, biaya, waktu, dan hal-hal lain.
6. Graf Tidak Berbobot
   Graf tidak berbobot merupakan graf yang setiap sisinya tidak memiliki nilai tertentu, hanya hubungan antara simpul.



Gambar 2.2: Graf berbobot dan tidak berbobot

C. Terminologi Graf

1. Ketetanggaan

Simpul yang bertetangga atau memiliki ketetanggaan adalah dua simpul yang saling terhubung dengan sebuah sisi.

2. Bersisian

Sebuah sisi dianggap bersisian jika sisi tersebut berhubungan langsung dengan simpul tertentu.

3. Simpul Terpencil

Simpul terpencil adalah simpul yang tidak memiliki tetangga atau dengan kata lain tidak terhubung dengan sisi apapun.

4. Graf Kosong

Graf kosong adalah graf yang tidak memiliki sama sekali sisi.

5. Derajat

Derajat adalah jumlah sisi yang terkait dengan suatu simpul.

6. Lintasan

Lintasan adalah suatu jalan yang ditempuh dengan sisi-sisi yang berhubungan.

7. Sirkuit

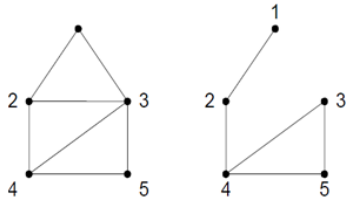
Sirkuit adalah lintasan yang berawal dan berakhir pada tempat yang sama.

8. Upagraf

Upagraf adalah bagian dari graf di mana simpul dan sisinya merupakan bagian dari graf utamanya.

9. Upagraf Merentang

Upagraf merentang adalah bagian dari graf (upagraf) yang mengandung semua simpul dari graf utamanya.

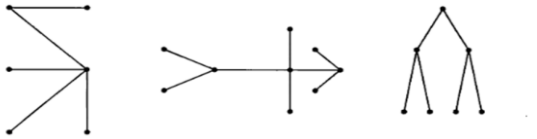


Gambar 2.3: Upagraf merentang

III. TEORI POHON

A. Definisi Pohon

Pohon adalah graf yang tidak memiliki sirkuit, tidak berarah, dan seluruhnya terhubung.



Gambar 3.1: Beberapa contoh pohon

B. Pohon Merentang

Pohon merentang atau *spanning tree* adalah upagraf merentang dari suatu graf yang tidak memiliki sirkuit atau dengan kata lain pohon yang merupakan upagraf merentang suatu graf.

Pembentukan pohon merentang dapat dilakukan dengan pemutusan sisi-sisi yang menyebabkan terjadi sirkuit pada graf utama.

Gambar 3.2: Contoh pohon merentang

Dalam teorinya, setiap graf terhubung memiliki paling sedikit sebuah pohon merentang.

Pada graf berbobot, upagraf merentangnya dapat dibuat sebagai pohon merentang minimum, di mana graf tersebut dibentuk menjadi pohon merentang, namun dengan jumlah tiap sisinya sekecil atau seminimum mungkin.

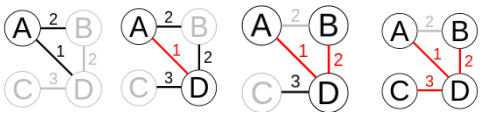
C. Algoritma Prim

Algoritma Prim merupakan algoritma yang digunakan untuk menemukan sebuah pohon merentang minimum dari sebuah graf. Algoritma ini berjalan dengan beberapa langkah sebagai berikut.

Pertama, pilihlah suatu simpul sebagai titik mulai dari proses pengerjaan. Simpul yang dipilih bebas dikarenakan sebuah pohon merentang harus mengandung semua simpul yang ada.

Kedua, pilihlah sisi dengan bobot terkecil dari simpul-simpul yang sudah terpilih. Dalam hal ini apabila pada langkah satu dipilih simpul 1, maka akan dicari sisi yang berbobot minimum yang berhubungan dengan 1.

Ketiga, ulangi langkah kedua sampai seluruh simpul telah terhubung dengan satu sisi, dengan persyaratan pada langkah kedua tidak boleh terjadi pembentukan sirkuit.



Gambar 3.3: Contoh pengerjaan algoritma Prim

Dengan kata lain algoritma Prim akan secara berurutan membentuk upagraf pohon dari graf utama dengan simpul yang terus-menerus ditambah.

C. Algoritma Kruskal

Algoritma Kruskal juga merupakan algoritma yang digunakan untuk menemukan sebuah pohon merentang minimum dari sebuah graf. Algoritma ini berjalan dengan beberapa langkah sebagai berikut.

Pertama, daftarkan semua sisi yang ada beserta nilai bobotnya.

Kedua, urutkan setiap sisi berdasarkan bobot sehingga sisi terurut dengan bobot yang kian meningkat.

Ketiga, bentuklah sebuah graf sesuai urutan yang telah dibuat di langkah kedua dengan batasan bahwa apabila sisi yang dibentuk menghasilkan sirkuit maka sisi tersebut dilewat dan dilanjutkan ke sisi berikutnya.

Sisi	(2,3)	(1,4)	(1,3)	(2,4)	(1,2)	(3,4)
Bobot	2	4	5	8	19	20

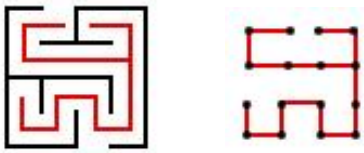
Tabel 3.1: Contoh tabel terurut Kruskal

Dengan kata lain pengerjaan algoritma Kruskal adalah sebagai berikut. Contoh pada tabel 3.1. Mulai dengan membuat sisi (2,3). Dilanjutkan dengan membuat sisi (1,4). Apabila (1,4) menghasilkan sebuah sirkuit, maka sisi (1,4) tidak dibuat dan dilanjutkan dengan sisi (1,3). Hal ini akan diulangi terus sampai semua simpul telah memiliki sisi atau sampai seluruh tabel telah dilewati.

IV. Pembuatan Labirin

A. Labirin dalam Graf Pohon

Sebuah labirin dapat direpresentasikan sebagai sebuah graf. Untuk merepresentasikan labirin dalam bentuk graf yang dilakukan adalah dengan menandai *path* atau jalan yang dapat dilalui. Jalan tersebut merupakan kumpulan dari sisi-sisi graf tersebut. Simpul dari sebuah labirin ditandai dengan kotak-kotak atau *cell* sebuah labirin pada labirin segiempat. Pada labirin yang lebih kompleks simpul dapat diartikan sebagai suatu tempat tanpa percabangan atau suatu tempat tanpa belokan.



Gambar 4.1: Labirin dan representasi grafnya

Seperti pada gambar 4.1 representasi dari labirin adalah graf pohon. Hal ini terjadi karena adanya batasan dari sebuah labirin yaitu bahwa pada labirin tidak ada suatu sirkuit. Sirkuit dari labirin tidak akan menambah kesulitan atau pun menguranginya. Sirkuit bahkan mungkin akan membuat labirin tidak dapat diselesaikan.

B. Pembentukan Labirin dari Pohon

Sebagaimana suatu labirin dapat direpresentasikan dalam bentuk pohon, maka suatu pohon juga dapat dibuat menjadi suatu labirin.

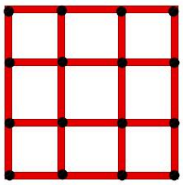
Dengan mengubah setiap sisi dari pohon menjadi *path* atau jalan kosong dan yang tidak terkena sisi menjadi tembok dari labirin akan terbentuk suatu labirin tanpa posisi awal dan posisi akhir.

Penempatan posisi awal dan akhir akan dilakukan dengan menambahkan suatu simpul diluar labirin dan menghubungkannya pada dua buah simpul berderajat satu.

C. Algoritma Prim Acak dalam Labirin

Algoritma Prim merupakan algoritma yang digunakan untuk membentuk sebuah pohon merentang minimum dari sebuah graf. Karena sebuah labirin juga merupakan pohon merentang, maka penggunaan algoritma Prim dapat juga membuat suatu labirin.

Karena algoritma Prim membentuk pohon dari sebuah graf akan digunakan sebuah graf yang saling berhubungan dengan simpul sekitarnya.



Gambar 4.2: Graf yang digunakan dalam labirin 4x4

Dari suatu graf seperti gambar 4.2, algoritma Prim akan membentuk sebuah pohon merentang. Namun, yang dihasilkan bukan pohon merentang minimum karena pada graf yang digunakan bukanlah graf yang berbobot. Dalam hal ini algoritma Prim acak yang digunakan adalah sebagai berikut.

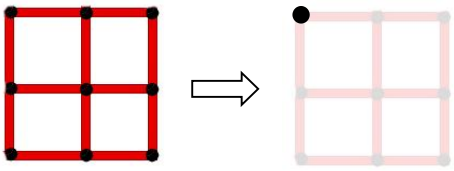
Pertama, akan dipilih sebuah simpul acak pada graf yang akan digunakan.

Kedua, tambahkan sisi pada simpul yang sudah dipakai, dalam hal ini sisi yang dibentuk akan dipilih secara acak. Batasan yang digunakan sama seperti algoritma Prim biasa, yaitu tidak diperbolehkannya pembuatan sirkuit.

Ketiga, pengulangan instruksi kedua sampai terbentuk pohon merentang atau semua simpul telah digunakan.

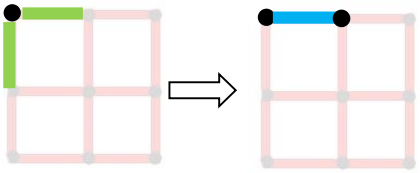
Untuk memperjelas algoritma tersebut akan dicontohkan algoritma Prim dalam kasus labirin 3x3.

Pertama, ambil sebuah simpul bebas pada graf.



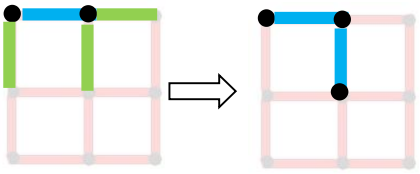
Gambar 4.2: Langkah algoritma Prim acak

Lalu, tambahkan suatu sisi secara acak. Sisi yang akan ditambahkan harus berupa sisi yang berhubungan dengan simpul dari sisi-sisi yang sudah dibuat. Dalam contoh ini, warna hijau menandakan kemungkinan sisi yang dapat ditambahkan.



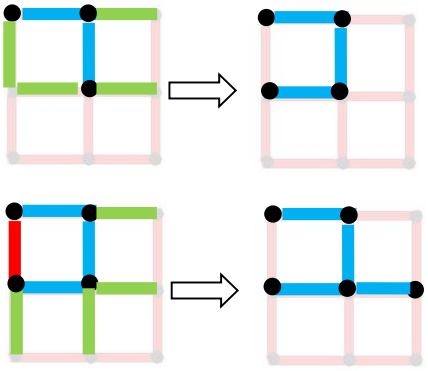
Gambar 4.3: Langkah algoritma Prim acak

Lanjutkan, tambahkan suatu sisi secara acak.



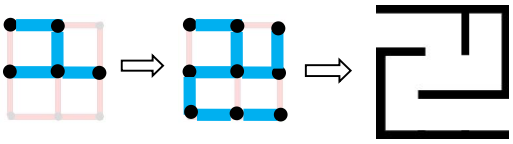
Gambar 4.4: Langkah algoritma Prim acak

Ulangi langkah tersebut, tambahkan suatu sisi secara acak. Pada gambar kedua terdapat batasan pembuatan sirkuit, maka dalam proses pemilihan sisi, sisi tersebut tidak dapat dipilih



Gambar 4.5: Langkah algoritma Prim acak

Ulangi langkah tersebut, sampai seluruh simpul memiliki minimal satu sisi. Setelah itu ubah pohon menjadi suatu labirin.



Gambar 4.6: Langkah algoritma Prima acak

D. Algoritma Kruskal Acak dalam Labirin

Sama halnya dengan Algoritma Prim, Algoritma Kruskal juga merupakan algoritma yang digunakan untuk membuat suatu pohon merentang minimum dari suatu graf yang tersambung. Karena pohon merentang dapat dibentuk menjadi labirin, maka Algoritma Kruskal juga dapat digunakan untuk menghasilkan labirin.

Dalam hal ini, sama seperti pada algoritma Prim untuk pembuatan labirin 4x4 akan digunakan suatu graf terhubung seperti pada gambar 4.2.

Algoritma Kruskal di sini juga merupakan algoritma Kruskal acak yang berarti dari graf tersebut tidak akan terbentuk suatu pohon merentang minimum. Hal ini dikarenakan sisinya yang memang tidak memiliki bobot. Oleh sebab itu, algoritma Kruskal yang digunakan tidak mengurutkan sesuai bobot dari sisi melainkan mengurutkan secara acak. Dalam hal ini algoritma Kruskal acak yang digunakan adalah sebagai berikut.

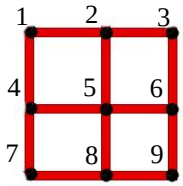
Pertama, dibuat semua kemungkinan sisi yang ada pada suatu tabel.

Kedua, tambahkan sisi, dalam hal ini sisi yang dibentuk akan dipilih secara acak. Batasan yang digunakan sama seperti algoritma Kruskal biasa, yaitu tidak diperbolehkannya pembuatan sirkuit.

Ketiga, pengulangan instruksi kedua sampai terbentuk pohon merentang atau semua simpul telah digunakan.

Untuk memperjelas algoritma tersebut akan dicontohkan algoritma Kruskal dalam kasus labirin 3x3.

Pertama, bentuk sebuah tabel dari semua kemungkinan sisi..

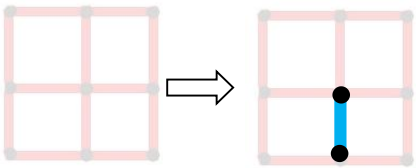


Gambar 4.7: Langkah algoritma Kruskal acak

(5,8)	(6,9)	(1,4)	(4,5)	(3,6)	(1,2)
(8,9)	(4,7)	(2,5)	(5,6)	(2,3)	(7,8)

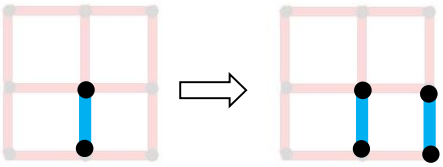
Tabel 4.1: Langka algoritma Kruskal acak

Lalu, bentuklah suatu sisi berurutan dari tabel, dalam contoh berarti akan dibuat sisi (5,8)



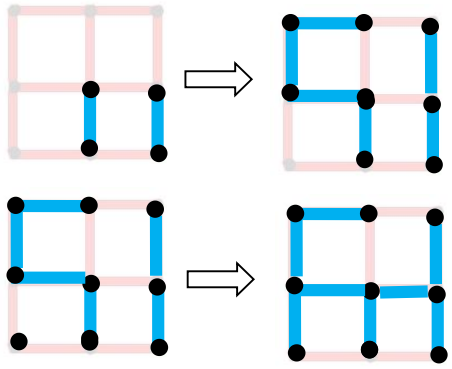
Gambar 4.8: Langkah algoritma Kruskal acak

Lanjutkan, tambahkan suatu sisi kedua.



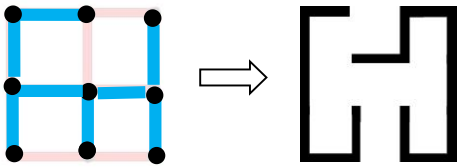
Gambar 4.9: Langkah algoritma Kruskal acak

Ulangi langkah tersebut sampai semua simpul telah memiliki sisi, tambahkan suatu sisi menurut urutan tabel. Perhatikan bahwa apabila penambahan sisi mengakibatkan sirkuit, maka sisi tersebut akan dilewat. Contoh, apabila kita menambahkan (5,6) lalu sisi selanjutnya (8,9), sisi (8,9) tidak akan dibuat.



Gambar 4.10: Langkah algoritma Kruskal acak

Terakhir ubahlah pohon merentang tersebut ke dalam bentuk labirin. Tambahkan titik mulai dan titik akhir.



Gambar 4.11: Langkah algoritma Kruskal acak

V. KESIMPULAN

Algoritma pembuatan pohon merentang dapat digunakan dalam pembuatan labirin. Hal ini dikarenakan sebuah labirin dapat direpresentasikan dalam bentuk sebuah pohon. Dalam hal ini terdapat dua buah algoritma yang biasa digunakan dalam pembuatan pohon merentang, yaitu algoritma Prim dan algoritma Kruskal yang sedikit diubah.

Perbedaan penggunaan algoritma Prim serta Kruskal biasa dengan algoritma yang digunakan pada pembuatan labirin adalah urutan yang disebabkan tidak digunakannya bobot pada pembuatan labirin. Jadi, kedua algoritma tersebut digunakan bukan dengan urutan bobot, tetapi digunakan dengan pemilihan sisi serta simpul yang acak.

Penerapan kedua algoritma tersebut dalam ukuran yang lebih besar dapat menghasilkan labirin yang jauh lebih kompleks dalam waktu yang lebih sedikit dari pembuatan secara manual.

VII. UCAPAN TERIMA KASIH

Pertama, penulis ingin mengucapkan terima kasaih kepada Tuan yang Maha Esa karena hanya atas berkat-Nya penulis dapat menyelesaikan makalah ini. Penulis juga ingin menyampaikan terima kasih kepada orang tua yang tanpanya penulis tidak akan mampu mendapat pengetahuan sejauh ini. Penulis juga tidak lupa ingin mengucapkan terima kasih kepada Dr. Ir. Rinaldi Munir, MT serta Harlili S, M.Sc. yang telah memberikan pengajaran mengenai materi yang dibahas. Selain, itu penulis juga ingin mengucapkan terima kasih kepada teman-teman serta keluarga yang telah membantu melalui masukan, dukungan, serta doa.

REFERENCES

[1] Munir, Rinaldi. 2005. Matematika Diskrit Edisi Ketiga. Bandung : Penerbit Informatika.

[2] Foltin, Martin. Automated Maze Generations and Human Interaction, dari: is.muni.cz/th/143508/fi_m/thesis.pdf diakses: 8 Desember 2015. Pukul 15:32.

[3] Buck, James. Mazes for Programmers, dari: : weblog.jamisbuck.org/2011/1/10/maze-generation-prim-s-algorithm diakses: 8 Desember 2015. Pukul 12:22.

[4] Buck, James. Mazes for Programmers, dari: : weblog.jamisbuck.org/2011/1/10/maze-generation-kruskal-s-algorithm diakses: 8 Desember 2015. Pukul 12:32.

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, 8 Desember 2015

Jason Jeremy Iman - 13514058