Penggunaan Algoritma Depth First Search dalam Penyelesaian Maze

Faiz Muhammad Muflich / 13517093
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13517093@std.stei.itb.ac.id

***Abstract*—Algoritma Depth First Search atau DFS merupakan salah satu algoritma yang digunakan untuk melakukan penelusuran terhadap graf atau pohon berdasarkan tingkat kedalaman graf atau pohon tersebut. Permainan maze adalah permainan dimana pemain harus menemukan solusi atau jalan keluar sehingga keluar dari labirin tersebut. Hal yang menarik dari permainan maze adalah bagaimana cara menemukan jalan keluar dengan algoritma berjalan yang sedang dijalankan serta dengan waktu yang secepat mungkin. Makalah ini akan membahas mengenai penggunaan teori graf dalam algoritma DFS untuk menyelesaikan permainan maze.**

***Keywords*—Graf, Depth First Search Algorithm, Maze, Backtrack**

I. PENDAHULUAN

Permainan *maze* atau labirin merupakan permainan yang cukup populer sejak zaman dahulu. Menurut sejarah asal usul permainan labirin adalah dari zaman peradaban suku Aztec. Salah satu peninggalan suku Aztec yang masih dikenal hingga sekarang adalah *The Man in The Maze.* Dari peninggalan tersebut orang-orang pada zaman sekarang mengadaptasi hal tersebut kemudian menjadikannya menjadi sebuah permainan yang populer.

Permainan labirin merupakan kumpulan dinding yang membentuk jalan dari tempat awal permainan hingga tercapai jalan keluar. Namun dalam pencarian tersebut terdapat dinding-dinding yang mengakibatkan banyaknya kelokan dan bercabang-cabang, serta dapat ditemukan pula jalan buntu. Maka dari itu permainan ini membutuhkan kemampuan otak yang tidak biasa karena jika manusia memainkan permainan ini secara sungguhan akan sangat susah untuk dapat menyelesaikan labirin. Banyaknya kelokan dan jalan buntu yang harus dihafalkan tentu saja menjadi hal yang sangat berat jika benar-benar dimainkan oleh manusia sungguhan. Oleh sebab itu manusia mengadaptasi permainan ini menjadi permainan di atas kertas atau bahkan dewasa ini dibuat kompetisi robot pemecah labirin.

Tujuan dari permainan labirin adalah menemukan jalan keluar tercepat dari tempat mulai hingga tempat tujuan. Seperti yang sudah disebutkan sebelumnya bahwa banyak sekali jenis permainan labirin tergantung media yang digunakan akan tetapi setiap permainan labirin mempunyai tujuan yang sama yaitu mencari jalan keluar dari labirin.

Salah satu produk matematika diskrit yang memiliki keterkaitan dengan persoalan permainan labirin adalah teori graf. Graf mempunyai banyak implementasi terhadap kehidupan sehari-hari. Graf memudahkan proses visualisasi persoalan baik itu data, alur, bahkan dapat dihubungkan dengan permainan labirin.

Labirin dapat kita modelkan sebagai graf yang terhubung bahkan untuk beberapa kasus labirin dapat kita modelkan sebagai tree seperti yang kita ketahui bahwa tree sendiri salah satu bagian dari graf. Setiap node dapat direpresentasikan sebagai titik dimana ditemukan kelokan, kemudian edge dapat direpresentasikan sebagai jalan yang menghubungkan node. Jalan keluar termasuk kedalam node jika kita representasikan ke dalam tree node merupakan daun.

Dewasa ini perkembangan teknologi komputasi sangat luar biasa, termasuk diantaranya muncul banyak algoritma-algoritma yang membantu manusia dalam mengatasi permasalahan yang ada. Bahkan algoritma-algoritma tertentu dapat diklasifikan menjadi bebeapa jenis tergantung dari masalah di bidang apa yang sedang akan diselesaikan.

Beberapa desain algoritma yang cukup populer digunakan dalam menyelesaikan permasalahan graf labirin ialah algoritma Breadth First Search (BFS) atau algoritma melebar, algoritma A* (A star), dan algoritma Depth First Search (DFS). Algoritma ini sudah diuji pada banyak labirin dengan variasi kompleksitas dan hasilnya dibandingkan berdasarkan waktu penyelesaian yang diperlukan dan panjang jalur yang dihasilkan dari setiap desain algoritma. Berdasarkan data tersebut disimpulkan bahwa BFS adalah yang paling akurat dalam menentukan jalur terpendek untuk setiap labirin. Akan tetapi hal tersebut dapat diketahui jika pemain mengetahui secara menyeluruh peta labirin. Sedangkan untuk setiap permainan belum tentu pemain diberikan peta labirin tersebut. Sehingga pada makalah ini akan dibahas mengenai algoritma DFS. Karena dalam beberapa kasus algoritma DFS memiliki waktu yang lebih cepat dalam menemukan jalan keluar, sebab saat program sudah menemukan jalan keluar maka program akan langsung berhenti.

Dalam makalah ini penulis akan membahas mengenai graf dan algortima DFS, implementasi graf pada maze, serta penggunaan algoritma DFS dalam menyelesaikan permainan labirin.

II. DASAR TEORI GRAF

A. Definisi Graf

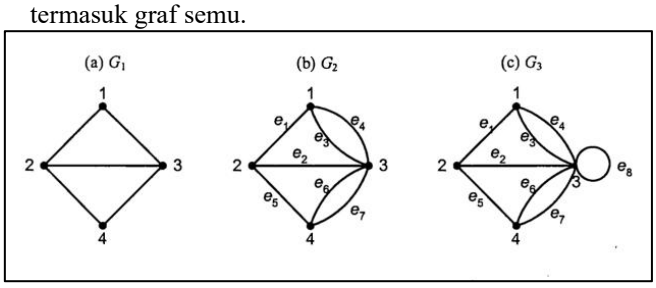
Graf mempunyai kegunaan untuk merepresentasikan objek-objek diskrit dan keterkaitannya antara objek-objek tersebut. Representasi secara visual dari graf adalah dengan menganggap objek sebagai suatu titik atau bulatan, sedangkan hubungan antara objek dapat dinyatakan dengan garis. Graf *G* secara matematis dapat didefinisikan sebagai pasangan himpunan (*V,E*) ditulis sebagai *G=(V,E),* dengan *V* diartikan sebagai himpunan tidak kosong dari simpul-simpul (*vertices)* sedangkan E adalah himpunan sisi-sisi (*edges*) yang menghubunkan sepasang simpul.[1]

Simpul pada graf biasanya dinamain dengan huruf(a,b,c,…), bilangan asli(1,2,3,…) atau gabungan keduanya. Sedangkan sisi yang menghubungkan simpu *a* dengan simpul *b* dinyatakan dengan pasangan (*a,b)* atau dilambangkan dengan lambang e_1, e_2, e_3, ... Dengan kata lain, jika *e* adalah sisi yang menghubungkan simpul *a* dengan simpul *b*, maka *e* dapat ditulis dalam bentuk[1]

$$e = (a,b)$$

Jenis graf dapat dibedakan berdasarkan ada tidaknya sisi ganda serta ada tidaknya orientasi arah pada sisi. Apabila dikategorikan berdasarkan ada tidaknya sisi ganda, maka graf dibedakan menjadi dua jenis :

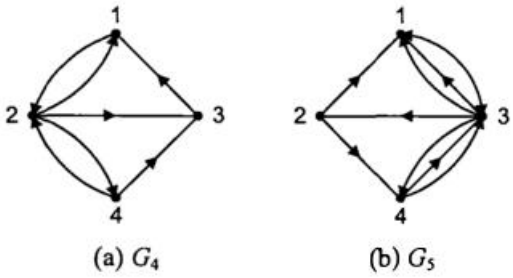
1. **Graf Sederhana,** adalah graf yang tidak mempunyai gelang maupun sisi-ganda. Pada graf sederhana, sisi dianggap sebagai pasangan tidak terurut karena tidak mempunyai arah, sehingga penulisan *(a,b)* dan (*b,a)* dianggap mempunyai arti yang sama.
2. **Graf Tidak Sederhana,** adalah graf yang mempunyai gelang atau sisi ganda ataupun keduanya. Graf tidak sederhana yang mengandung sisi ganda tanpa gelang disebut dengan graf ganda. Graf tidak sederhana yang mengandung gelang disebut graf semu. Graf tidak sederhana yang mengandung gelang dan sisi ganda termasuk graf semu.



Gambar 1.1 Visualisasi graf (a) graf sederhana, (b) graf ganda, dan (c) graf semu (Munir,2010)

Bila dikelompokan berdasar ada tidaknya orientasi arah pada sisi, graf dibagi menjadi dua jenis :

1. **Graf Tak Berarah,** adalah graf yang sisinya tidak mempunyai orientasi arah. Mirip dengan graf sederhana, sisi dianggap sebagai pasangan tidak terurut karena tidak mempunyai arah, sehingga penulisan *(a,b)* dan (*b,a)* dianggap mempunyai arti yang sama.
2. **Graf Berarah,** adalah graf yang setiap sisinya memiliki orientasi arah. Sisi yang berarah bisa disebut busur (*arc)*. Misal pada busur (*u,v*), *u* merupakan simpul asal sedangkan *v* merupakan simpul terminal. Graf berarah sering digunakan untuk menggambarkan aliran proses.



Gambar 1.2 Visualisasi graf (a) graf berarah dan (b) graf ganda berarah (Munir,2010)

B. Terminologi dalam Graf

Akan dijelaskan beberapa terminologi atau istilah dalam masalah graf yang akan digunakan dalam makalah ini. Perlu diperhatikan bahwasanya terminologi dalam graf tidak terbatas hanya pada terminologi yang tertulis di makalah ini.

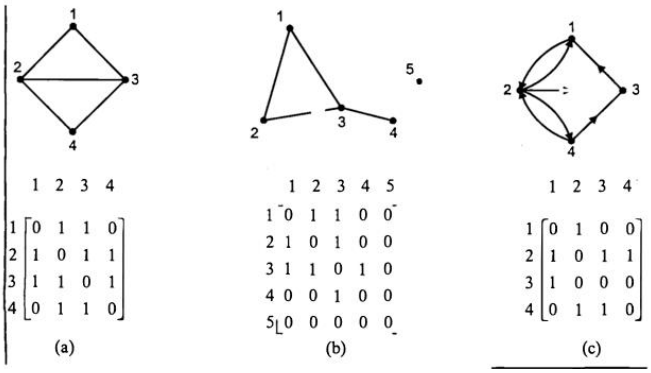
1. Bertetangga (Adjacent), dua buah simpul pada graf dikatakan bertetangga, jika keduanya terhubung langsung melalui suatu sisi.
2. Berisisian(*Incident)*, untuk sembarang sisi *E=(V,U),* sisi *E* dikatakan bersisian dengan simpul V dan simpul U.
3. Graf Kosong (*Null Graph*), suatu graf disebut graf kosong apabila himpunan setiap sisinya adalah himpunan kosong, sehingga graf tersebut sama sekali tidak memiliki sisi.
4. Derajat (*Degree*), pada graf tak berarah, derajat suatu simpul didapat dari banyak sisi yang bersisian dengan simpul tersebut. Sedangkan untuk graf berarah, derajat dibedakan menjadi dua, yaitu ada derajat masuk dan derajat keluar. Derajat masuk didefinisikan sebagai banyak busur yang menunjuk ke simpul tersebut, sedangkan untuk derajar keluar didefinisikan sebagai banyak busur yang meninggalkan dari simpul tersebut.
5. Lintasan(*Path),* lintasan dalam suatu graf diartikan sebagai barisan selang seling dari simpul dan sisi yang terhubung dan berbentuk v1, e1, v2, e2, ….. dari sini dapat diidentifikasi banyak sisi yang terlewati. Dan setiap sisi merupakan sisi-sisi dari graf G.
6. Graf terhubung (connected), graf G disebut graf terhubung jika untuk setiap pasangan simpul yang merupakan simpul di G, mempunyai lintasan yang menghubungkan keduanya. Jika G mempunyai paling tidak sedikitnya satu simpul yang tidak terhubung maka graf tersebut disebut graf tak terhubung.
7. Graf Berbobot (*Weighted Graph*) graf berbobot ialah graf yang setiap sisinya mempunyai nilai (bobot).[1]

C. Representasi Graf

Untuk memudahkan penyelesaian masalah dalam penggunaan graf, dibutuhkan representasi graf yang mudah untuk dipahami. Akan dibahas representasi graf dengan menggunakan matriks ketetanggaan (adjacency matrix).

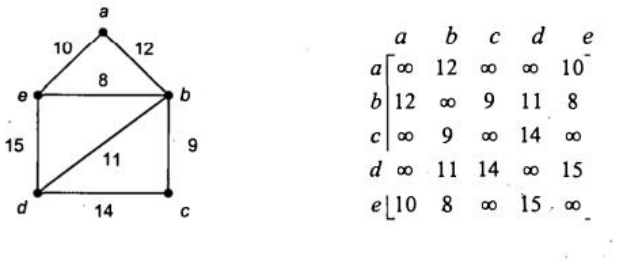
Matriks ketetanggan G adalah matriks dwimatra yang berukuran n x n. Dengan n merupakan banyaknya simpul. Bila matriks tersebut dinamakan A = [a_{ij}], maka a_{ij} = 1 jika simpul i

dan j bertetangga, sebaliknya jika 0 maka simpul i dan j tidak bertetangga.[1]



Gambar 1.3 contoh penggunaan matriks ketetanggaan (Munir,2010)

Untuk graf berbobot, nilai a_{ij} diisi dengan bobot tiap sisi yang menghubungkan kedua simpul. Bila tidak berhubungan, diberikan nilai tak hingga.[1]



Gambar 1.4 contoh penggunaan matriks ketetanggaan untuk graf berbobot (Munir,2010)

III. DASAR TEORI ALGORITMA DFS

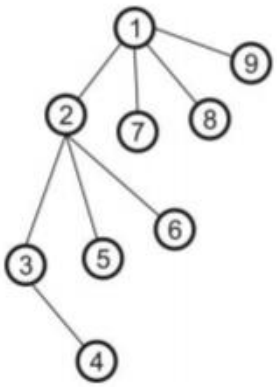
A. Algortima Depth First Search

Algoritma pencarian mendalam, atau lebih dikenal dengan DFS, merupakan algoritma traversal graf yang melakukan penelusuran simpul dengan pendekatan mendalam. Secara informal, algoritma DFS mengunjungi simpul dari akar ke simpul anak (*child node*) terus menerus hingga ke daun terlebih dahulu, kemudian runut balik (backtrack) ke simpul bapak *(parent node)* bila tidak menemukan solusi, lalu ditelusuri ulang simpul anak yang belum dikunjungi terus menerus seterusnya hingga sudah tidak ada yang belum dikunjungi, atau sampai ditemukan solusi.[2]

Secara formal, algoritma DFS diartikan sebagai berikut. Misalkan terdapat graf G yang mempunyai n buah simpul. Traversal (penelusuran) dimulai dari simpul v. Simpul yang berikutnya dikunjungi ialah simpul w yang bertetangga dengan simpul v, lalu pencarian mendalam dimulai lagi secara rekursif dari simpul w. Ketika mencapai pada sebuah simpul, misalnya u, sedemikian sehingga semua simpul yang bertetangga dengannya telah dikunjungi, pencarian dirunut-balik ke simpul terakhir yang dikunjungi sebelumnya dan mempunyai simpul w yang belum dikunjungi. Pencarian mendalam kemudian dimulai lagi dari w. Pencarian berakhir bila tidak ada lagi simpul yang belum dikunjungi yang dapat dicapai dari simpul yang telah dikunjungi.[3]

Dari definisi formal diatas, jika disesuaikan dengan permasalahan labirin maka algoritma akan berhenti saat sudah ditemukan jalan keluar labirin. Sehingga peneulusuran tidak dilanjutkan. Akan tetapi jika node terdalam yang tercapai ternyata tidak mempunyai tetangga lagi dan bukan merupakan jalan keluar maka algoritma akan backtrack ke node sebelumnya dan mencari anak lain yang belum dikunjungi hingga ditemukan jalan keluar.

Jika algoritma digambarkan secara visual maka akan tampak seperti berikut.



Gambar 2.1 visualisasi algoritma DFS (Atmadja,2015)

Simpul nomor 1 (simpul akar) merupakan simpul yang paling pertama dikunjungi, maka urutan pengunjungan simpul bersesuaian dengan nomor simpul yang tertera pada graf. [2]

B. Algoritma Runut Balik

Ketika algoritma DFS menemukan simpul daun yang tidak memberikan solusi, maka algoritma DFS melakukan runut balik (backtracking). Algoritma runut balik ini berbasis pada DFS untuk mencari solusi persoalan yang lebih efisien.[3] Algoritma runut balik dapat dilakukan dengan dua pendekatan, yaitu pendekatan iteratif dan pendekatan rekursif. Pada makalah ini, penulis akan membahas hanya algoritma runut balik dengan pendekatan rekursif.[2]

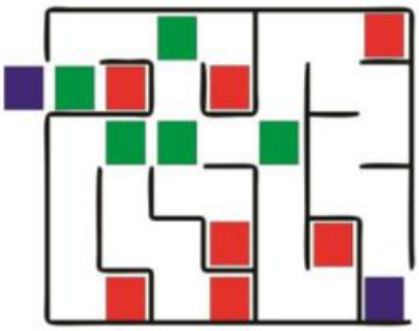
Secara umum, algoritma runut balik rekursif berguna untuk persoalan yang membutuhkan langkah-langkah yang tidak sedikit dalam menemukan jawabannya. Seperti contoh penggunaan algoritma ini dalam algoritma DFS, atau cara kerja mesin prolog yang mencari nilai kebenaran dari fakta dan rule yang diberikan, ataupun mesin untuk men-generate bilangan permutasi.

Jelas begitu bermanfaatnya algoritma runut balik dalam berbagai permasalahan. Termasuk dalam penggunaan algoritma DFS, algoritma runut balik mempunyai peranan penting. Karena tanpa adanya algoritma runut balik maka program dapat berjalan terus tanpa henti karena ada kasus dimana labirin menjadikan program berjalan secara terus menerus tanpa henti jika tidak digunakan algoritma runut balik.

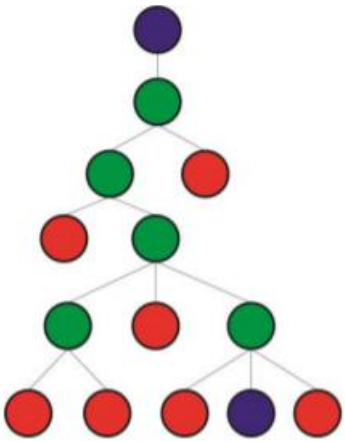
IV. PEMBAHASAN

A. Representasi Labirin menjadi Graf

Saat akan menyelesaikan permainan maze terdapat beberapa pendekatan. Salah satu pendekatan yang terbaik adalah dengan graf. Labirin dapat direpresentasikan menjadi graf. Pada makalah ini, graf dibuat sedemikian rupa sehingga simpul mewakili titik persimpangan, baik itu pertigaan, perempatan serta jalan buntu, dan sisi mewakili jalan yang menghubungkan antara simpul tersebut. Oleh karena itu simpul dapat dibedakan menjadi tiga macam, yaitu simpul pintu masuk/keluar, simpul persimpangan, serta simpul jalan buntu.



Gambar 3.1 labirin yang disertai dengan blok warna (Atmadja,2015)



Gambar 3.2 Graf hasil representasi labirin (Atmadja,2015)

Dari graf tersebut warna yang berbeda-beda menandakan jenis node dari titik yang berbeda-beda pula. Warna merah diartikan sebagai jalan buntu, warna hijau diartikan sebagai persimpangan jalan, dan warna ungu menandakan pintu masuk serta pintu keluar. Perlu diperhatikan bahwa simpul persimpangan atau simpul hijau selalu mempunyai anak simpul, berbeda dengan simpul merah yang merupakan jalan buntu. Tentu saja sudah tidak mempunyai anak simpul lagi atau bisa kita anggap sebagai daun(jika dalam tree). Sedangkan untuk simpul awal dan akhir mempunyai keistimewaan sendiri. Hal ini digunakan untuk mengenali bahwa program berjalan dari mana dan akan menuju node akhir (jalan keluar) yang dikenali dari keistimewaannya.[2]

B. Penyelesaian Labirin Menggunakan Algoritma DFS

Jika labirin dapat direpresentasikan menjadi graf seperti di atas maka tentu saja labirin dapat diseleaikan dengan algoritma DFS.

Pada mulanya program berjalan dari titik awal atau pintu masuk. Kemudian menelusuri ke simpul anak terus menerus dengan orientasi anak paling kiri terlebih dahulu. Selama simpul masih mempunyai anak maka akan dikunjungi sang anak terus menerus. Apabila ternyata program mengunjungi jalan buntu (node merah) maka program akan melakukan backtracking kemudian kembali ke simpul bapak dan mengunjungi simpul anak yang belum dikunjungi. Hal tersebut dilakukan terus menerus hingga ditemukan jalan keluar. Karena sebelumnya sudah diperlakukan berbeda untuk jalan keluar, maka program akan dengan mudah menyadari bahwa program sudah mencapai jalan keluar atau belum.

Hal tersebut dirasa mudah dalam penyelesaiannya akan tetapi per;u diperhatikan bahwa penjelasan tersebut hanya digunakan untuk labirin yang sudah direpresentasikan menjadi graf.

Berikut akan dibahas bagaimana algoritma pengunjungan simpul dan pembangkitan simpul menggunakan algoritma DFS.

- Program mulai berjalan dari titik awal (hanya node awal yang sudah terdefinisi sejak awal).
- Apabila simpul yang dikunjungi simpul persimpangan, periksa bentuk simpangannya dan jalan baru yang bisa dilalui oleh benda. Setidaknya simpul persimpangan mempunyai tiga atau empat belokan termasuk arah masuk persimpangan.
- Apabila dari arah masuk persimpangan terdapat belokan ke kiri, bangkitkan/kunjungi simpul yang terhubung dengan jalan tersebut. Ulangi dari langkah 1 dengan posisi sekarang ialah simpul ynag baru dibangkitkan. Bila tidak ditemukan belokan ke kiri lanjut ke tahap selanjutnya.
- Apabila dari arah masuk persimpangan terdapat jalan lurus, bangkitkan / kunjungi simpul yang terhubung dengan jalan lurus tersebut. Ulagi dari langkah 1 dengan posisi sekarang ialah simpul yang baru dibangkitkan atau dikunjungi. Bila tidak ditemukan jalan lurus, lanjutkan ke tahap selanjutnya.
- Apabila dari arah masuk persimpangan terdapat belokan ke kanan, bangkitkan / kunjungi simpul yang terhubung dengan jalan tersebut. Ulangi dari langkah 1 dengan posisi sekarang ialah simpul yang baru dibangkitkan atau dikunjungi. Apabila tidak ada belokan ke kanan lanjut ke tahap selanjutnya.
- Apabila simpul yang dikunjungi merupakan simpul jalan buntu, maka berhenti.
- Apabila simpul yang dikunjuni merupakan simpul akhir (jalan keluar) dianggap sebagai jalan buntu namun diberikan tanda atau pewarnaan yang berbeda untuk menunjukan bahwa node tersebut merupakan jalan keluar.
- Backtracking ke simpul persimpangan sebelumnya lalu kunjungi node yang belum dikunjungi.
- Apabila simpul sekarang merupakan simpul awal, maka berhenti. [2]

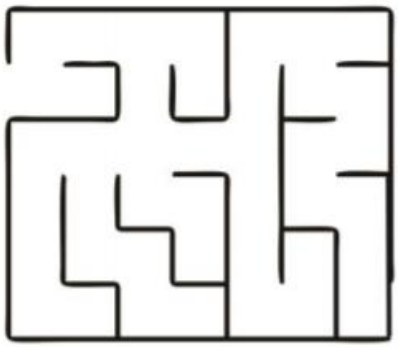
Dengan menjalankan algoritma di atas maka labirin sudah

dapat diproses selayaknya graf. Kemudian untuk menyelesaikan graf tersebut dijalankan algoritma DFS seperti yang sudah dijelaskan sebelumnya. Perlu diketahui disini penulis menggunakan orientasi arah kiri. Karena setiap proses selalu diproses bagian kiri secara mendalam terlebih dahulu. Akan tetapi perlu digaris bawahi bahwasanya algoritma DFS tidak terbatas hanya kepada orientasi kiri saja namun bisa orientasi arah yang berbeda.

C. Pengecualian

Labirin dengan satu pintu mempunyai jalan masuk dan keluar yang sama tentu saja dengan begitu algoritma DFS tidak dapat menyelesaikan persoalan karena tidak akan ditemukan jalan keluar.

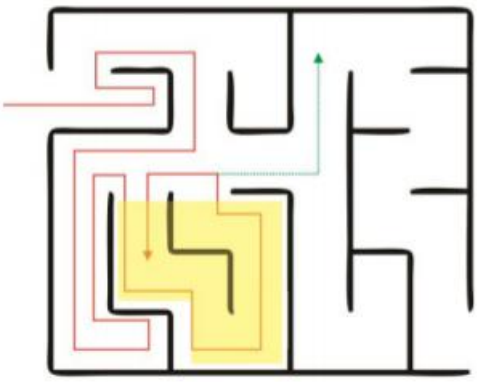
Seperti yang sudah disepakati di awal bahwa pintu masuk dan pintu keluar merupakan simpul yang dianggap berbeda maka solusi dari algoritma DFS tidak akan membuahkan hasil. Jika dicerna lebih jauh lagi permainan labirin jika tidak ditemukan pintu keluar dirasa kurang sempurna maka memang semestinya labirin harus mempunyai pintu keluar.



Gambar 3.3 Labirin dengan satu pintu (Atmadja,2015)

Terlihat jelas labirin ini tidak dimungkinkan selesai dengan menggunakan algoritma DFS maka perlu diperiksa ulang sebelum digunakan algoritma DFS.

Selain labirin dengan satu pintu, jika beruntung dapat ditemukan pula kasus dimana labirin mepunyai suat sirkuit, yaitu apabila dalam penelusuran simpul persimpangan tidak ditemukan jalan buntu atau jalan keluar dan hanya ditemukan persimpangan lainnya. Maka hal ini jika tidak dilakukan pencatatan akan terjadi *infinite loop* karena sifat dfs yang mencari hingga mendalam terlebih dahulu.



Gambar 3.4 Labirin dengan sirkuit (Atmadja,2015)

Jelas terlihat bahwa labirin di atas akan mengakibatkan *infinite loop* hal ini juga diakibatkan jika direpresentasikan menjadi graf, maka tidak akan muncul pohon. Oleh sebab itu tidak semua labirin dapat diselesaikan dengan menggunakan algoritma DFS. Hanya saja jika setiap node yang sudah dikunjungi dicatat maka program tidak akan menemui *infinite loop*.[2]

V. Penerapan Algoritma Depth First Search

Algoritma Depth First Search termasuk salah satu algoritma graf yang sangat berguna. Ada beberapa manfaat dari penggunaan algoritma DFS diantaranya sebagai berikut :

- Penyelesaian permasalahan permainan labirin seperti yang sudah dibahas di makalah ini. Ternyata algoritma DFS termasuk ke dalam algoritma yang mangkus dalam penyelesaian masalah ini.
- Untuk permainan sudoku, salah satu alternatif penyelesaiannya dapat juga digunakan algoritma graf DFS.
- Dapat digunakan untuk membuat sebuah search engine lokal.
- Dan masih banyak lagi.

VI. Kesimpulan

Teori graf mempunyai banyak sekali kebermanfaatan untuk manusia. Salah satunya penyelesaian permainan labirin. Dengan bantuan teori graf, permainan labirin dapat diselesaikan dengan mangkus bahkan untuk kasus terbaik tidak perlu ada usaha untuk mengingat. Selain itu pemodelan permasalahan menjadi bentuk yang lebih sederhana juga membantu manusia dalam menyelesaikan permasalahan yang dihadapi. Mengubah bentuk labirin menjadi graf merupakan bukti nyata pemodelan sederhana mempunyai cara yang mudah dan teratur dalam penyelesaiannya. Akan tetapi untuk penggunaan algoritma DFS dalam permainan labirin tidak semua labirin dapat diselesaikan termasuk dalam kemampuan mengingat lintasan, persimpangan dan jalan buntu. Sehingga hanya labirin yang dapat dimodelkan menjadi graf yang berbentuk pohon saja dapat diselesaikan sesuai dengan penjelasan di atas.

VII. Ucapan Terima Kasih

Penulis mengucapkan syukur alhamdulillah kepada Allah SWT atas karunia dan hidayah-Nya serta kemudahan-kemudahan yang tiada hentinya selalu tercurah sehingga penulis dapat menyelesaikan tugas penulisan makalah ini. Penulis juga mengucapkan terima kasih sebesar-besarnya kepada Dr. Judhi Santoso M.Sc., Dra. Harlili M.Sc., dan Dr. Ir. Rinaldi Munir, M.T. selaku dosen mata kuliah IF 2120 Matematika diskrit yang telah membimbing dan mengajarkan kepada penulis perkuliahan matematika diskrit serta telah memberikan tugas ini sebagai pemacu penulis untuk meningkatkan kapasitasnya sebagai mahasiswa yang bermanfaat bagi sesama terkait kebermanfaatan keilmuan informatika yang sudah ditekuni sejauh ini.

Pada akhirnya, saya ucapkan terima kasih kepada Bapak, Ibu,

serta keluarga yang selalu mendukung saya dalam menyelesaikan perkuliahan ini. Saya harap makalah ini mempunyai nilai kebermanfaatan untuk sesama.

REFERENCES

[1] R. Munir, Matematika Diskrit, 3rd ed. Bandung : Penerbit INFORMATIKA Bandung, 2010, ch.10.

[2] J. Atmadja, "Implementasi Intuitif Penyelesaian Labirin dengan Algoritma Depth-First Search"

[3] Sullivan, Ph. D., D. G. (2012, Fall). Recursion and Recursive Backtracking. United States: Harvard Extention School. Dipetik Desember 9, 2018, dari http://www.fas.harvard.edu/~cscie119/lectures/recursion.pdf

[4] Kleinberg, J., & Tardos, E. (2006). Algorithm Design. Massachussets: Pearson Education, Inc.

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, 10 Desember 2018

Faiz Muhammad Muflich / 13517093