Strategi Menemukan Jalan Keluar Labirin dengan Waktu Tercepat Menggunakan Metode DFS

Mustika Rahmasuci*, Husnul Hotimatus S, Maulia Azizah*****
Putri Wulandari****, Diah Adistia A*******, Saiful Bukhori********
* ** *** **** ***** Student at University of Jember, ******Lecture at University of Jember
*mrahmasuci@gmail.com, **husnul238sahroh@gmail.com, ***maulinawolfson@gmailcom
**** putriwulandariduatiga@gmail.com, ***** diahdst@yahoo.com, ******saiful.ilkom@unej.ac.id

ABSTRACT

Depth-first search algorithm are blind search process and deepen search follow a single track until found goal. Maze is complicated network way and tortuous, also have many deadlock. Maze often become a challenge in a game like puzzle, which one there's an object in start position have find way out on specify position. in this journal we'll discuss about utilization DFS on maze and fastest time to find way out.

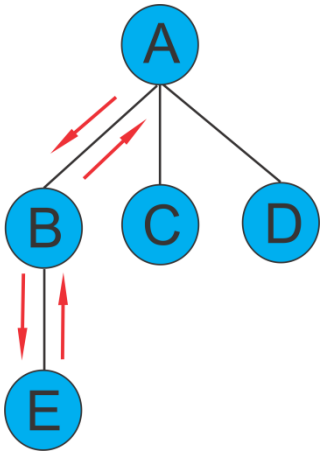
Keyword: DFS, Labirin.

1. Introduction

Labirin adalah jaringan jalan yang rumit dan berliku-liku, serta memiliki banyak jalan buntu. Sejak zaman dahulu labirin telah digunkan untuk berbagai kepentingan. Menurut sejarah, labirin dipercaya sebagian orang berasal dari zaman kuno dimana labirin merupakan struktur bangunan berliku yang dibuat Daedalus untuk raja Minos untuk memenjarakan Minotaur [1].

Pada umumnya labirin dibuat untuk tujuan hiburan. Dalam kehidupan nyata labirin banyak dibuat di taman atau ruangan dengan pembatas berupa pagar atau pun tembok dengan ukuran yang bervariasi. Labirin ini dirancang untuk menjadi sebuah atraksi permainan atau hanya sebagai hiasan saja. Ada beberapa negara yang membuat labirin dengan ukuran manusia dan menantang orang-orang untuk masuk kedalamnya, sebagai salah satu atraksi untuk menarik wisatawan [1]. Labirin-labirin ini secara tidak langsung menyesatkan orang asing yang masuk kedalamnya. Ada juga aplikasi labirin yang dapat dijumpai terutama dalam industry game, karena labirin menantang pemain untuk mencari jalan keluar dari titik yang ditentukan sebelumnya [1].

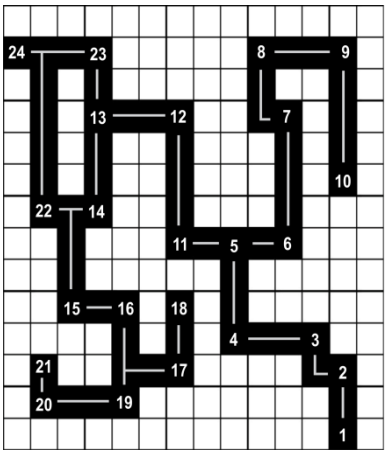
Depth First Search (DFS) adalah algoritma pencarian simpul dalam graf yang berjalan kedalam. Pada pencarian dengan algoritma Depth First Search, pencarian dimulai dari level paling pertama (level 0) kemudian dilanjutkan ke anak paling kiri pada level berikutnya (level 1) demikian seterusnya sampai tidak terdapat anak lagi atau level yang lebih dalam lagi. Jika pencarian sudah mencapai node atau anak paling dalam maka akan dilakukan penelusuran mundur atau backtracking untuk melakukan pencarian ke node anak berikutnya [2].



Gambar 1 Contoh penerepan DFS pada diagram pohon

ada Gambar 1, setiap lingkaran disebut node. Pencarian dilakukan mulai dari node A-B-E, pada node ini akan melakukan penelusuran mundur (backtracking) menuju E-B-A kemudian melanjutkan ke node anak berikutnya yaitu C. Demikian seterusnya sampai tujuan ditemukan.

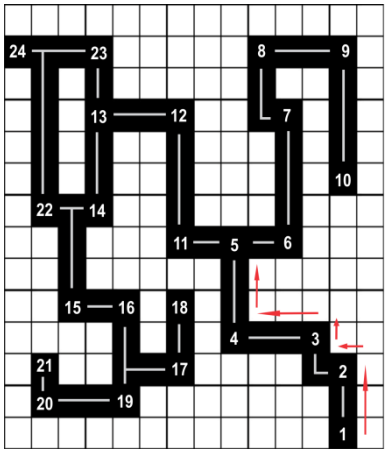
2. Research Method



Gambar 2 Contoh Labirin

Gambar 2 merupakan labirin yang akan digunakan untuk menjelaskan bagaimana menemukan jalan keluar dengan waktu tercepat menggunakan metode DFS. Seperti yang terlihat pada Gambar 2, kotak hitam bernomor merupakan jalan yang akan di lewati. Sedangkan kotak putih merupakan tembok atau penghalang. Setiap jalur yang berbeda diberi angka yang berbeda juga, untuk memudahkan menerapkan DFS pada labirin. Pada labirin tersebut terdapat angka 1 yang merupakan pintu masuk atau bisa disebut dengan start. Jika diterapkan pada pohon seperti pada Gambar 1 maka angka 1 bisa dianggap sebagai node A yang juga start pada pohon itu. Sedangkan angka 24 merupakan pintu keluar dari labirin itu atau bisa disebut sebagai tujuan. Jadi jurnal ini akan menjelaskan bagaimana menerapkan DFS pada labirin untuk mencari jalan keluar dengan waktu tercepat dari angka 1 hingga angka 24.

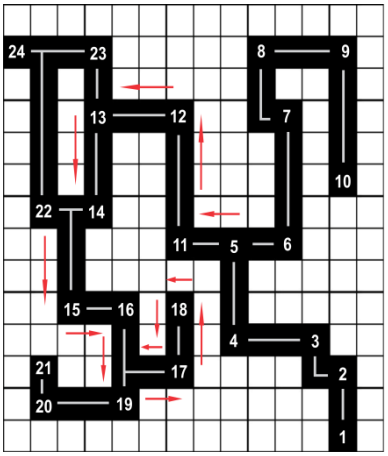
Tahap 1



Gambar 3 Strategi Pencarian pada Tahap 1

Bisa dilihat pada Gambar 3, angka 1-2-3-4 merupakan jalan satu arah (tidak bercabang atau bukan pertigaan atau perempatan). Sehingga DFS akan langsung menuju angka 2 dan seterusnya sampai pada angka 4. Hal itu terjadi karena DFS tidak memiliki jalan lain selain melewati angka 1-2-3-4. Berbeda dengan angka 5, yang memiliki cabang atau pertigaan. Pada angka 5, DFS akan menentukan angka 6 atau angka 11 yang dipilih untuk dilewati terlebih dahulu.

Tahap 2

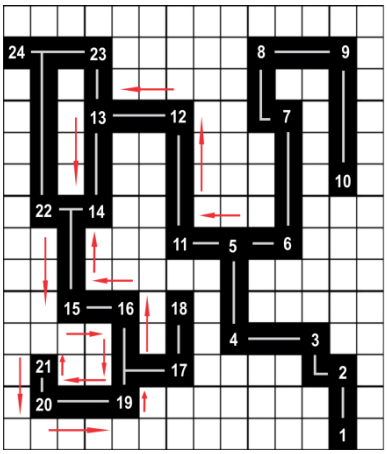


Gambar 4 Strategi Pencarian pada Tahap 2

Bisa dilihat pada Gambar 4, pada angka 5 menuju ke angka 11, bukan ke angka 6 karena DFS akan memulai pencarian dari sebeleh kiri. Setelah dari angka 11 akan langsung menuju angka 12-13 yang merupakan jalan satu arah. Pada angka 13 DFS menemukan pertigaan antara angka 23 dan angka 14. Disini DFS akan memilih angka 14 untuk dilewati karena angka 14 berada disebelah kiri. Pada angka 14 juga merupakan jalan pertigaan antara angka 22 dan angka 15. DFS akan memilih angka 15 untuk di lewati karena berada di sebelah kiri dari angka 14. Setelah melewati angka 15 DFS akan menuju angka 16 yang merupakan jalur pertigaan antara angka 17 dan 19. Angka 17 adalah jalan yang dilewati terlebih dahulu oleh DFS karena berada disebelah kiri terhadap angka 16. Setelah melewati angka 17 akan bertemu dengan angka 18 yang merupakan jalan buntu. Sehingga DFS tidak bisa melanjutkan pencarian menuju angka 24. Disaat itu DFS menerapkan backtracking.

Jadi jalan yang sudah dilewati oleh DFS yaitu angka 5-11-12-13-14-15-16-17-18. Sedangkan bakctrakingnya di mulai dari angka 18-17-16. Kenapa backtracking berhenti di angka 16? Karena DFS menerapkan pencarian ke dalam. Angka 16 merupakan pertigaan paling akhir yang dilalui yang memiliki pertigaannya merupakan angka 17 dan 19. Setelah memilih angka17 dan tidak bisa melanjutkan pencarian karena angka 18 merupakan jalan buntu. Maka setelah melukan backtracking. DFS akan memulai pencarian lagi dari angka 16 dan memilih jalan yang beum dilewati (selain angka 17) yaitu angka 19.

Tahap 3

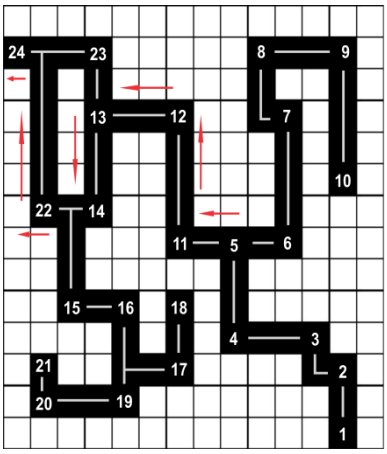


Gambar 5 Strategi Pencarian pada Tahap 3

Setelah melakukan backtracking pada tahap 2, DFS memulai lagi pencariannya dari angka 16 menuju angka 19. Setelah menuju angka 19, DFS langsung melaui jalan di angka 20 dan angka 2,1 karena jalan tersebut merupakan jalan satu arah. Setelah berada di angka 21, DFS tidak bisa melanjutkan perjalannya menuju angka 24 karena angka 21 merupakan jalan buntu. Setelah menemukan jalan buntu DFS akan menerapkan backtracking lagi.

Jadi angka yang di lewati pada tahap ini yaitu 5-11-12-13-14-15-16-19-20-21. Sedangkan backtrackingnya yaitu dari angka 21-20-19-18-17-16-15-14. Kenapa backtracking berhenti di angka 14 bukan di angka 16 ? Karena pertigaan di angka 16 sudah dilewati semua oleh DFS, yaitu angka 17 dan 19 dan menemui jalan buntu, sehingga tidak bisa melanjutkan pencarian menuju angka 24. Sedangkan angka 14 merupakan pertigaan paling akhir setelah melakukan backtracking. Maka pencarian akan dilanjutkan dari angka 14.

Tahap 4



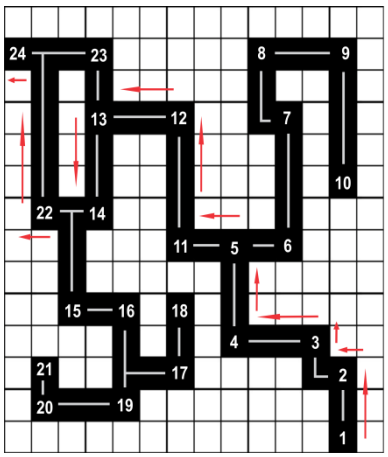
Gambar 6 Strategi Pencarian pada Tahap 4

Setelah melakukan backtracking di tahap 3. DFS akan memulai lagi pencariannya untuk menemukan angka 24 sebagai tujuannya, di mulai dari angka 14 yang merupakan pertigaan paling akhir setelah pertigaan di angka 16 sudah dilewati semua. Pada pertigaan angka 14 terdapt angka 15 dan 22, angka yang belum dilewati yaitu angka 22. Jadi DFS memilih angka 22 untuk dilewati. Ternyata angka 22 merupakan pertigaan dari angka 23 dan angka 24. Karena angka 24 berada disebelah kiri dari angka 22. Maka DFS memilih angka 24 untuk melanjutkan pencarian. Ternyata angka 24 yang dipilih DFS merupakan tujuan atau pintu keluar dari labirin. Maka proses pencarian akan berhenti.

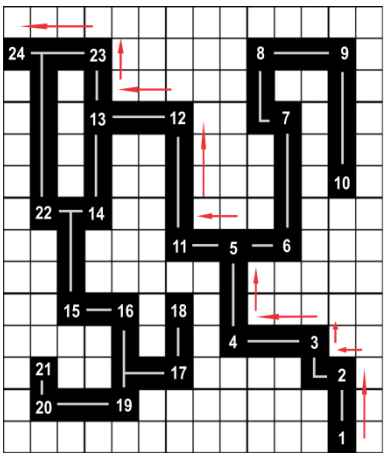
3. Result and Analysis

Setelah DFS melakukan pencarian dan menemukan tujuannya atau jalan kelaur pada labirin. Maka proses pencarian akan berhenti. Jadi hasil akhir dari pencarian tersebut bisa dilihat pada Gambar 7a. Maka

hasil akhir yang di dapatkan oleh DFS untuk waktu tercepat menemukan jalan keluar menggunakan metode DFS yaitu melewati angka 1-2-3-4-5-11-12-13-14-22-24, terdapat 11 jalur. Jika di amati lebih lanjut, ternyata hasil pencarian DFS untuk menemukan waktu tercepat ternyata memiilik jalur lebih panjang dari Gambar 7b



Gambar 7a Labirin dengan hasil akhir dari Strategi Pencarian



Gambar 7b Labirin dengan hasil akhir berbeda

Pada Gambar 7b juga menemukan hasil pencarian dari angka 1 sebgai pintu masuk hingga angka 24 sebagai pintu keluar labirin. Tetapi jalur yang dilewati sedikit berbeda yaitu dari angka 1-2-3-4-5-11-12-13-23-24, terdapat 10 jalur yang dilewati. Bisa dilihat perbedaan jalur yang dilewati pada Gambar 7a dan Gambar 7b yaitu pada pertigaan angka 13. Pada Gambar 7a DFS memilih angka 14 yang berada disebelah kiri angka 13 untuk dilewati sedangkan pada Gambar 7b jalur yang dipilih yaitu angka 23 yang berada disebalah kanan angka 13.

Jika menggunakan Gambar 7b sebagai hasil akhir waktu tercepat, maka itu adalah hal yang salah. Karena hasil akhir pada Gambar 7b membutuhkan waktu yang lebih lama. Kenapa bisa begitu? Karena pencarian menggunakan metode DFS harus dimulai dari kiri. Sedangkan angka 23 berada di sebelah kanan pertigaan angka 13. Jadi secara otomatis DFS akan memilih jalur disebalah kiri dan melakukan pencarian kedalam. Seperti yang dilakukan pada BAB 2. Tetapi bisa menggunakan hasil akhir pada Gambar 7b jika mencari rute terpendek pada labirin meskipun membuthkan waktu yang lebih lama.

4. Conclusion

Setelah melakukan pancarian jalan keluar dengan waktu tercepat pada labirin seperti yang dilakukan pada BAB 2. Maka hasil yang benar yaitu pada Gambar 7a. Sedangkan Gambr 7b digunakan untuk mencari jalan keluar dengan rute terpendek pada labirin. Setelah menemukan hal tersebut maka dapat disumpulkan bahwa DFS memiliki kekurangan dan kelibihan.

Kelemahanya yaitu hasil akhir yang ditemukan tidak selalu optimal, karena yang didahulukan dalam pencarian adalah menuruni pohon[3]. Kelebihannya yaitu metode depth-first search akan menemukan solusi tanpa harus menguji lebih banyak node atau angka yang di pakai dalam pembahasan kali ini [4].

Acknowledgements

Terimakasih kami sampaikan untuk teman teman yang telah membantu hingga tersusunnya artikel ini yang merupakan output dari salah satu mata kuliah dengan dosen pembimbing Prof. Dr. Saiful Bukhori, ST., M.Kom.

References

[1] MazeGeneratorDenganMenggunakanAlgoritmaDepth-First-Search. Octarapribadi, S and Kom. 2015, Jurnal TIMES, Vol. 1, pp. 1-5.

[2] Evaluasi dan Usaha Optimalisasi Algoritma Depth First Search dan Breadth First Search dengan Penerapan pada Aplikasi Rat Race dan Web Peta. Kandaga, Tjatur and Hapendi, Alvin. 2008.

[3] Analisis dan Penyelesaian Permainan River Crossing Ultimate Menggunakan Algoritma BFS dan DFS. Aprilia, Ira and Naskah, Terima. 2, 2016, Edisi, Vol. 6. 2088-4591.

[4] Robot Micromouse dengan Menggunakan Algoritma Depth-First Search. Anita, Nur. 2010.