Analisis Perbandingan Algoritma *Depth First Search* dan Algoritma *Breadth First Search* dalam Memecahkan Permainan Labirin

Ghebyon Tohada Nainggolan - 13520079
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13520079@std.stei.itb.ac.id

Abstrak — *Maze game* adalah permainan labirin dengan tujuan mencari sebuah target/pintu keluar dengan melewati banyak pilihan jalur, cabang, dan jalan buntu. *Maze game* merupakan salah satu dari banyak persoalan model matematika, dan untuk menyelesaikan persoalan matematika dibutuhkan cara yang disebut sebagai algoritma. Dalam ilmu informatika terdapat beberapa algoritma untuk menemukan solusi dari *Maze game* dua diantaranya adalah *Depth First Search* dan *Breadth First Search.*

Kata Kunci — *Maze Game, Depth First Search, Breadth First Search*

I. PENDAHULUAN

Game merupakan salah satu jenis hiburan yang disukai hampir oleh semua orang di tiap kalangan usia. Selain digunakan untuk menghilangkan kepenatan dalam beraktivitas, sebuah game juga dapat berfungsi untuk melatih pola piker seseorang untuk mencari solusi memecahkan suatu permasalahan yang ada di sebuah game. Banyaknya jumlah dari jenis game yang muncul, menyebabkan adanya pengelompokan genre dari game. Salah satu jenis game yang dikenal yaitu maze game atau permainan labirin.

Penulis sebagai seorang yang terjun ke dunia informatika tertarik dalam menemukan cara yang tepat untuk menyelesaikan maze game ini. Dalam dunia informatika permasalahan maze game ini dapat dibentuk dalam struktur data graf. Di dalam graf sendiri, terdapat banyak metode *searching* sesuai dengan kebutuhannya masing-masing. Salah satu metode searching yang dikenal ialah *Blind Search.* Metode ini memiliki banyak algoritma dan untuk mempersempit cakupan makalah, penulis membatasi analisis algoritma yang digunakan yakni Algoritma *Breadth First Search* dan Algoritma *Depth First Search.* Algoritma *Breadth First Search* adalah algortima pencarian yang dilakukan secara melebar, sedangkan Algoritma *Depth First Search* adalah algoritma pencarian yang dilakukan berdasarkan kedalaman suatu cabang.

Penulis membuat makalah ini bertujuan untuk menganalisis kasus maze seperti apa yang cocok untuk diselesaikan dengan Algoritma BFS dan kasus maze seperti apa yang cocok untuk diselesaikan dengan Algoritma DFS.

II. LANDASAN TEORI

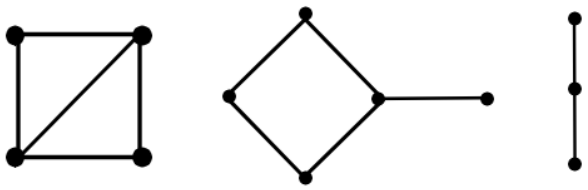
A. Graf

1. Definisi Graf

Graf adalah suatu struktur matematika berupa kumpulan titik(simpul/*vertex*) yang memiliki maupun tidak memiliki hubungan dengan titik lainnya, untuk titik yang memiliki hubungan dihubungkan oleh satu atau lebih garis(sisi/*edge*). Secara formal, suatu graf G didefinisikan sebagai G = (V,E). Dalam hal ini, $V = \{v1, v2, v3, \dots, vn\}$ atau himpunan tidak-kosong dari simpul-simpul; $E = \{e1, e2, e3, \dots, en\}$ atau himpunan sisi(*edges*) yang menggabungkan sepasang simpul. Sisi pada graf dinyatakan sebagai pasangan $e = (v1, v2)$ yang berarti sisi e merupakan sisi yang menghubungkan simpul v1 dengan simpul v2. Simpul dapat ditulis dengan huruf, angka, maupun kombinasi keduanya. Pada graf dimungkinkan adanya simpul yang memiliki sisi ke dirinya sendiri yang disebut dengan kalang (*loop*), kemudian dimungkinkan juga adanya pasangan simpul yang memiliki lebih dari satu sisi yang disebut dengan sisi ganda (*multiple edges*)

2. Jenis-Jenis Graf

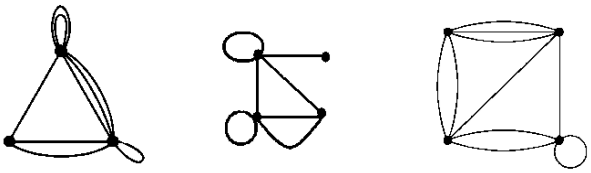
Berdasarkan ada tidaknya gelang atau sisi ganda pada suatu graf, graf dapat digolongkan menjadi dua jenis, yakni graf sederhana (*simple graph*), dan graf tak sederhana (*unsimple graph*). Graf sederhana ialah graf yang tidak memiliki sisi ganda atau sisi gelang.



Gambar 2.1 Graf sederhana
Sumber : Slide Materi Kuliah

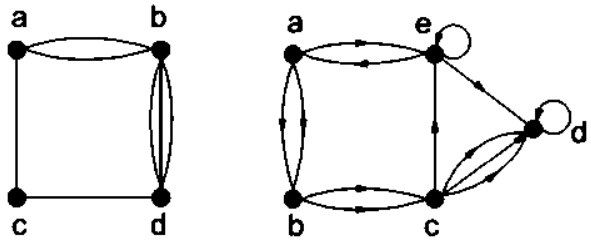
Sementara graf tidak sederhana dibagi lagi menjadi dua yakni graf ganda (*multi graph*) dan graf semu (*pseudo graph*). Graf ganda ialah graf yang memperbolehkan sisi ganda namun tidak sisi gelang. Sementara graf semu memperbolehkan sisi ganda dan sisi gelang.

Gambar 2.2 Graf Ganda
Sumber : Silde Materi Kuliah



Gambar 2.3 Graf Semu
Sumber : Slide Materi Kuliah

Berdasarkan orientasi arah pada sisi graf, graf dapat digolongkan menjadi dua jenis, yakni graf tak-berarah dan graf berarah. Graf tak-berarah adalah graf yang sisinya tidak mempunyai orientasi arah, dengan kata lain sisi $e1 = (v1, v2)$ akan serupa dengan sisi $e2 = (v2, v1)$. Sementara graf berarah adalah graf yang sisinya mempunyai orientasi arah, sisinya dikenal sebagai busur. Busur $(v1, v2)$ berarti sisi yang menghubungkan v1 dan v2 dengan v1 sebagai simpul asal dan v2 sebagai simpul tujuan.



Gambar 2.4 Graf tak-berarah(kiri) dan Graf berarah(kanan)
Sumber : Slide Materi Kuliah

3. Terminologi Graf

a. Ketetanggan (*Adjacent*)

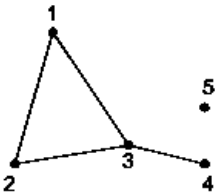
Dua buah simpul dikatakan bertetangga bila keduanya terhubung langsung dengan sebuah sisi(pada graf tak berarah).

b. Bersisian(*Incidenxy*)

Pada sembarang sisi $e = (v1, v2)$ dapat dikatakan e bersisian dengan simpul v1 dan 2 bersisian dengan simpul v2.

c. Simpul Terpencil

Simpul terpencil adalah simpul yang tidak mempunyai tetangga, dengan kata lain simpul tersebut tidak memiliki sisi yang bersisian dengannya.



Gambar 2.5 Graf dengan simpul terpencil (simpul 5)
Sumber : Slide Materi Kuliah

d. Graf Kosong (*null graph* atau *empty graph*)

Graf kosong adalah graf yang tidak memiliki sisi di dalamnya atau dengan kata lain setiak titik tidak memiliki hubungan apapun. V pada graf G = (V,E) adalah himpunan kosong.

e. Derajat (*Degree*)

Derajat suatu simpul v pada graf tak berarah adalah jumlah sisi yang bersisian dengan simpul tersebut atau dapat dikatakan sebagai banyak sisi yang terhubung dengan suatu simpul. Notasi derajat pada simpul v dituliskan *d(v)*. Kasus khusus, untuk simpul yang bersisian dengan sisi gelang derajat untuk sisi gelang tersebut berjumlah dua

f. Lintasan (*Path*)

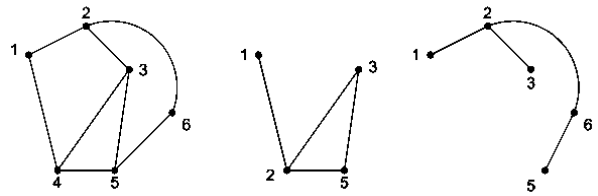
Lintasan dengan panjang n dari simpul awal ke simpul tujuan di dalam graf G adalah barisan berselang-seling simpul-simpul dan sisi-sisi yang berbentuk $v0, e1, v1, e2, v2, \ldots, en, vn$ sehingga $e1 = (v0, v1), e2 = (v1, v2), \ldots, en = (vn1, vn)$ adalah sisi-sisi dari graf. Pada graf sederhana, lintasan cukup dituliskan sebagai barisan dari simpulsimpul, tetapi untuk graf berarah perlu dituliskan pula sisinya untuk menghindari kerancuan. Selanjutnya, panjang lintasan merupakan jumlah sisi dalam lintasan tersebut. Lintasan yang berawal dan berakhir pada simpul yang sama disebut lintasan tertutup, sebaliknya disebut llintasan terbuka. Sirkuit (*Circuit*)

g. Keterhubungan (*Connected*)

Keterhubungan digunakan untuk mendeskripsikan keterhubungan dua buah simpul maupun keterhubungan suatu graf. Simpul v1 dan v2 dinyatakan terhubung jika terdapat lintasan dari v1 ke v2. Suatu graf G dinyatakan sebagai graf terhubung jika untuk setiap pasang simpulnya terhubung.

h. Upagraf (*Subgraf*) dan Komplemen Upagraf

Graf G1 = (V1,E1) adalah sebuah upagraf dari G = (V,E) jika setiap anggota elemen himpunan V1 terdapat di V dan setiap anggota elemen himpunan V2 terdapat di E. Sedangkan komplemen upagraf G1 adalah adalah graf G2 = (V2, E2) dengan E2 = E – E1 dan V2 adalah simpul-simpul yang bersisian dengan anggota E2.



Gambar 2.7 Graf (kiri), upagraf(tengah), dan komplemen upagraf(kanan)
Sumber : Slide Materi Kuliah

i. Upagraf Merentang

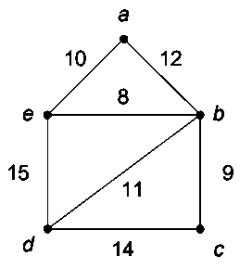
Upagraf merentang dari G = (V,E) adalah upagraf G1 = (V1, E1) yang mengandung semua simpul dari G, dengan kata lain V = V1.

j. Cut-Set

Cut-set adalah himpunan sisi yang jika dibuang dari graf terhubungnya akan menyebabkan graf tersebut menjadi tidak terhubung lagi dan menghasilkan dua komponen upagraf.

k. Graf Berbobot (*Weighted Graph*)

Graf berbobot adalah graf yang setiap sisinya diberi nilai atau bobot. Bobot dapat merepresentasikan berbagai hal tergantung permasalahannya



Gambar 2.5 Graf dengan sisi berbobot
Sumber : Slide Materi Kuliah

B. Blind Search

Blind search merupakan metode pencarian yang digunakan dalam *Visitation Problem*/Masalah Pengunjungan. *Blind search* adalah pencarian asal atau pencarian yang dilakukan tanpa ada informasi awal yang digunakan dalam proses pencarian. Jika solusi sudah ditemukan, maka pencarian akan dihentikan. Apabila dibuat skemanya, *blind search* hanya mengenal bagian [masalah]-[pencarian]-[solusi]. *Blind Searching* memiliki tiga ciri-ciri utama yaitu:

- Membangkitkan simpul berdasarkan urutan
- Kalau ada solusi maka solusi akan ditemukan
- Hanya memiliki informasi tentang simpul yang telah dikunjungi

Algoritma umum yang termasuk *Blind Search* diantaranya :

- *Breadth First Search* (BFS)
- *Depth First Search* (DFS)
- *Uniform Cost Search* (UCS)
- *Depth-Limited Search* (DLS)
- *Interative-Deeping Search* (IDS)
- *Bi-directional Search* (BDS)

C. Breadth First Search

Breadth First Search adalah algoritma pencarian yang dimulai dengan mencari target pada seluruh simpul yang berada di level lebih dari 1 dari simpul sebelumnya terlebih dahulu. BFS harus memenuhi persyaratan berikut :

1. Jika v(target) dapat dicapai dari s(awal), dan v != s, maka terdapat beberapa simpul u yang merupakan elemen simpul yang terhubung dengan v (yang mana u dikunjungi sebelum mengunjungi v).
2. Jika u dan v dicapai oleh s(dalam hal ini u dan v bertetangga), dan jika d(u) < d(v), maka u harus dikunjungi terlebih dahulu

Dalam pengaplikasiannya, BFS memanfaatkan struktur data *queue*. Berikut adalah contoh pseudocode-nya :

Alternatif 1

```
for all v in V[G] do
    visited[v] := false
end for
Q := EmptyQueue
Enqueue(Q,s)
while not Empty(Q) do
    u := Dequeue(Q)
    if not visited[u] then
        visted[u] := true
        for all w in Adj[u] do
            if not visited[w] then
                Enqueue(Q,w)
            end if
        end for
    end if
end while
```

Alternatif 2

```
for all v in V[G] do
    visited[v] := false
end for
Q := EmptyQueue
visited[s] := true
Enqueue(Q,s)
while not Empty(Q) do
    u := Dequeue(Q)
    for all w in Adj[u] do
        if not visited[w] then
            visited[w] := true
            Enqueue(Q,w)
        end if
    end for
end while
```

D. Depth First Search

Depth First Search adalah algoritma pencarian yang dimulai dengan mencari target pada salah satu simpul yang berada di level lebih dari 1 dari simpul sebelumnya terlebih dahulu. DFS harus memenuhi persyaratan berikut :

1. s(awal) dikunjungi terlebih dahulu
2. Jika v(target) dapat dicapai dari s, dan v != s, maka terdapat beberapa simpul u yang merupakan elemen simpul yang terhubung dengan v (yang mana u dikunjungi sebelum mengunjungi v).
3. Jika u dan v adalah simpul dari graf dapat dijangkau dari s dan jika v tidak dapat dijangkau dari u, dan jika u dikunjungi terlebih dahulu sebelum v, maka semua simpul yang *reachable* dari u akan dikunjungi sebelum v dikunjungi

Dalam pengaplikasiannya, DFS memanfaatkan struktur data *stack.* Berikut adalah contoh pseudocode-nya :

Alternatif 1

```
for all v in V[G] do
    visited[v] := false
end for
S := EmptyStack
Push(S,s)
while not Empty(S) do
    u := Pop(S)
    if not visited[u] then
        visted[u] := true
        for all w in Adj[u] do
            if not visited[w] then
                Push(S,w)
            end if
        end for
    end if
end while
```

Alternatif 2

```
for all v in V[G] do
    visited[v] := false
end for
S := EmptyStack
visited[s] := true
```

```
Push(S,s)
while not Empty(S) do
    u := Pop(S)
    if there is at least one unvisited vertex in Adj[u] then
        Pick w to be any unvisited vertex in Adj[u]
        Push(S,u)
        visited[w] := true
        Push(S,w)
    end if
end while
```

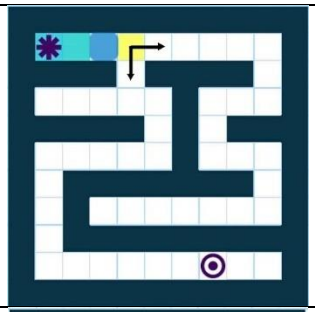
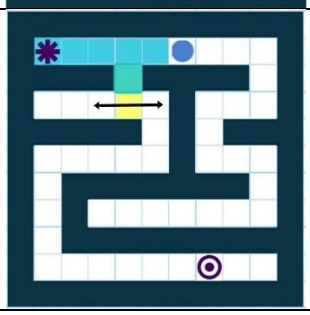
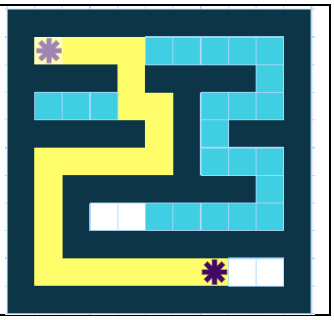
III. Aplikasi dan Analisis Algoritma BFS dan DFS pada Maze Game

Dalam analisis digunakan webapp https://clementmihailescu.github.io/Pathfinding-Visualizer/# untuk mempermudah visualisasi tiap algoritma. Sebagai keterangan simbol '*' sebagai *start state* dan 'o' sebagai final state. Sebagai contoh digunakan Map Maze berikut

Gambar 3.1 Contoh Maze yang Dianalisis
Sumber : Dokumen Penulis

A. Analisis Pencarian Solusi Menggunakan Algoritma Breadth First Search

Tabel 3.1 Analisis Contoh Kasus dengan BFS

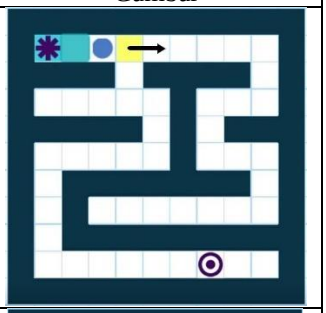
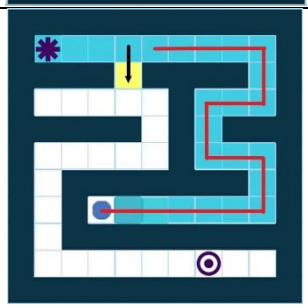
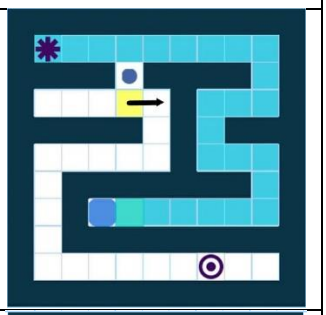
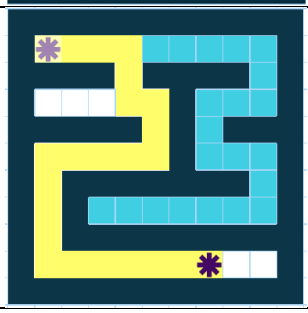
Keterangan	Gambar
Pada simpul kuning terdapat dua jalur selanjutnya. Pengunjungan ke simpul selanjutnya dilakukan sekaligus (kanan dan bawah)	
Pada simpul kuning terdapat dua jalur selanjutnya. Pengunjungan ke simpul selanjutnya dilakukan sekaligus (kiri dan kanan)	
Pengunjungan dilakukan seterusnya mengikuti jalur sisa karena tidak terdapat lagi simpul yang bercabang. Hingga didapatkan target simpul, maka pencarian dihentikan. Jalur kuning merupakan solusi permainan	

Hasil yang didapatkan :

- Simpul yang dikunjungi berjumlah : 45 simpul
- Panjang jalur solusi : 22 langkah

B. Analisis Pencarian Solusi Menggunakan Algoritma Depth First Search

Tabel 3.2 Analisis Contoh Kasus dengan DFS

Keterangan	Gambar
Pada simpul kuning terdapat dua jalur selanjutnya. Pengunjungan ke simpul selanjutnya dilakukan ke kanan terlebih dahulu	
Pengunjungan dilakukan sampai mendapatkan jalan simpul ujung (jalan buntu). Kemudian dilakukan pop pada stack yang menyimpan alamat simpul yang ditandai oleh garis merah hingga mencapai simpang awal. Selanjutnya pengunjungan dilakukan ke arah bawah	
Pada simpul kuning terdapat dua jalur selanjutnya. Pengunjungan ke simpul selanjutnya dilakukan ke kanan terlebih dahulu	
Pengunjungan dilakukan seterusnya mengikuti jalur hingga didapatkan target simpul, maka pencarian dihentikan. Jalur kuning merupakan solusi permainan	

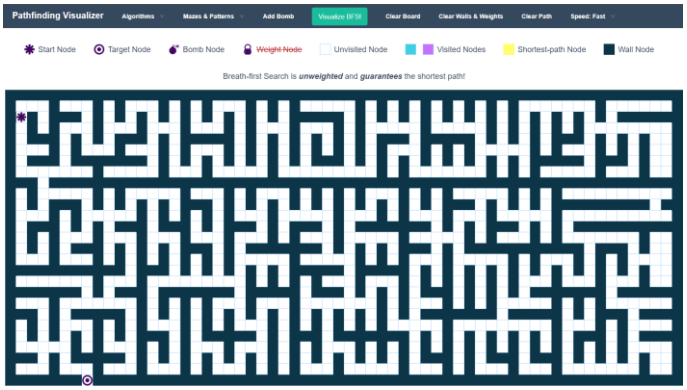
Hasil yang didapatkan :

- Simpul yang dikunjungi berjumlah : 44 simpul
- Panjang jalur solusi : 22 langkah

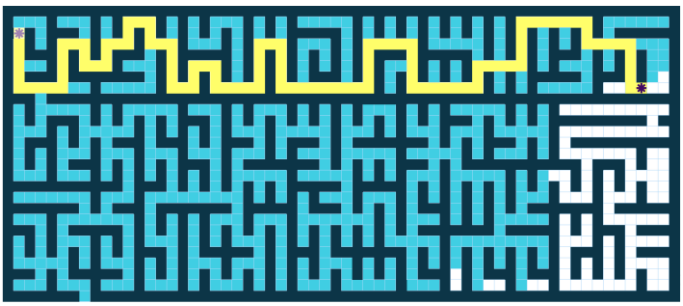
IV. STUDI KASUS

Beberapa contoh kasus di-*generate* secara random dengan menggunakan fitur website (Maze & Patterns)

A. Contoh Kasus 1



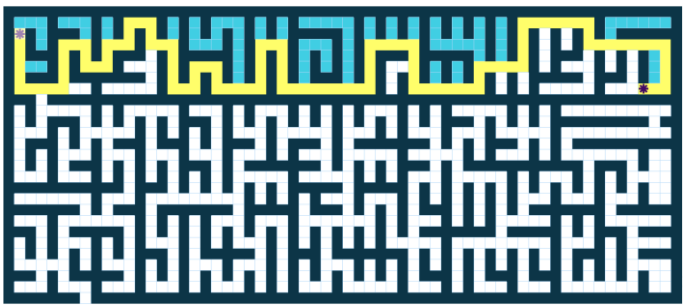
Gambar 4.1 Kasus 1
Sumber : Dokumen Penulis



Gambar 4.2 Solusi Kasus 1 dengan BFS
Sumber : Dokumen Penulis

Hasil yang didapatkan :

- Simpul yang dikunjungi berjumlah : 422 simpul
- Panjang jalur solusi : 32 langkah

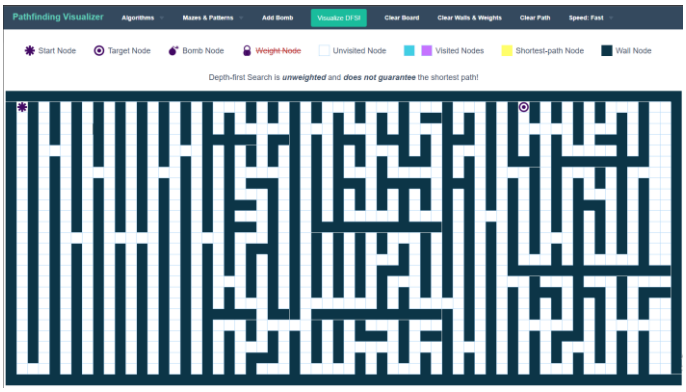


Gambar 4.3 Solusi Kasus 1 dengan DFS
Sumber : Dokumen Penulis

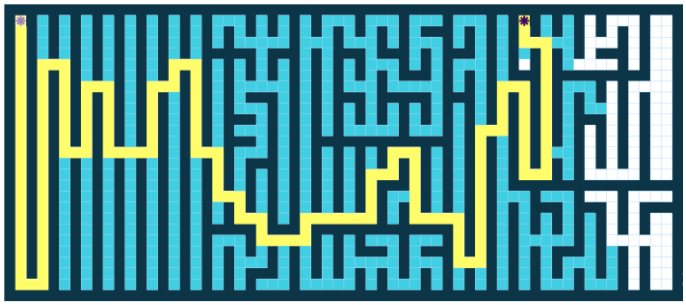
Hasil yang didapatkan :

- Simpul yang dikunjungi berjumlah : 202 simpul
- Panjang jalur solusi : 114 langkah

B. Contoh Kasus 2



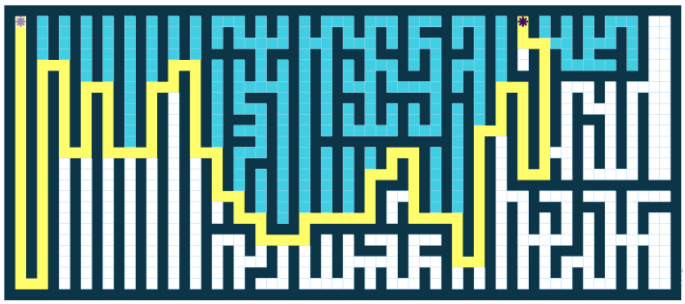
Gambar 4.4 Kasus 2
Sumber : Dokumen Penulis



Gambar 4.5 Solusi Kasus 2 dengan BFS
Sumber : Dokumen Penulis

Hasil yang didapatkan :

- Simpul yang dikunjungi berjumlah : 695 simpul
- Panjang jalur solusi : 195 langkah

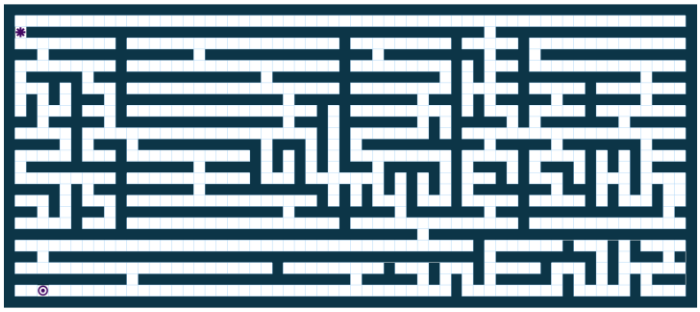


Gambar 4.6 Solusi Kasus 2 dengan DFS
Sumber : Dokumen Penulis

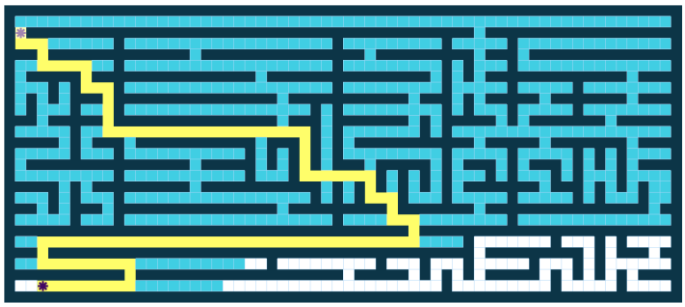
Hasil yang didapatkan :

- Simpul yang dikunjungi berjumlah : 443 simpul
- Panjang jalur solusi : 195 langkah

C. Contoh Kasus 3



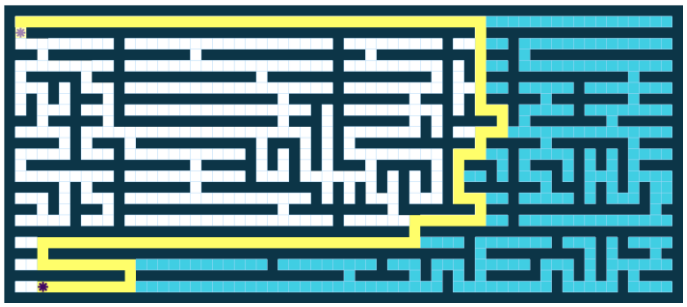
Gambar 4.7 Kasus 3
Sumber : Dokumen Penulis



Gambar 4.8 Solusi Kasus 3 dengan BFS
Sumber : Dokumen Penulis

Hasil yang didapatkan :

- Simpul yang dikunjungi berjumlah : 693 simpul
- Panjang jalur solusi : 109 langkah



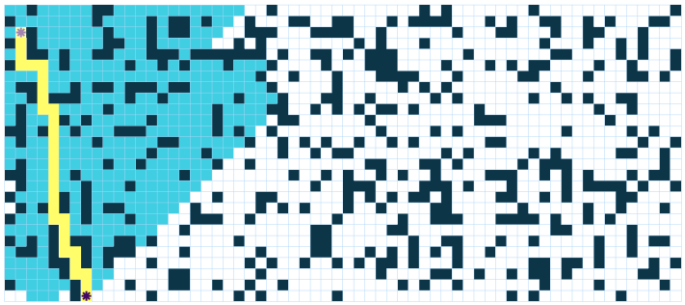
Gambar 4.9 Solusi Kasus 3 dengan DFS
Sumber : Dokumen Penulis

Hasil yang didapatkan :

- Simpul yang dikunjungi berjumlah : 422 simpul
- Panjang jalur solusi : 131 langkah

D. Contoh Kasus 4

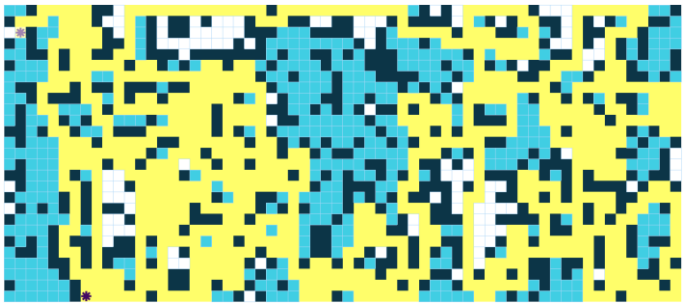
Gambar 4.10 Kasus 4
Sumber : Dokumen Penulis



Gambar 4.11 Solusi Kasus 4 dengan BFS
Sumber : Dokumen Penulis

Hasil yang didapatkan :

- Simpul yang dikunjungi berjumlah : 367 simpul
- Panjang jalur solusi : 30 langkah

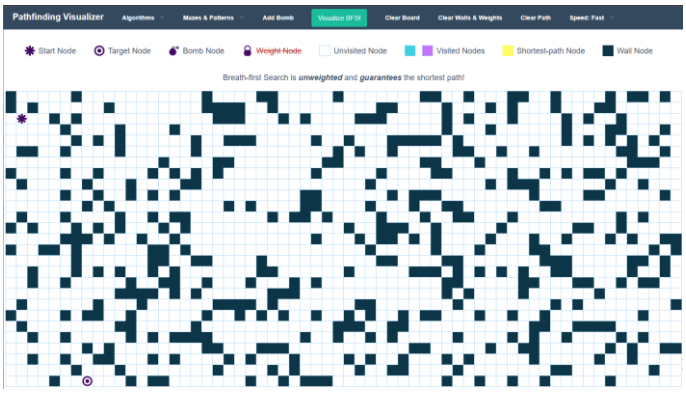


Gambar 4.12 Solusi Kasus 4 dengan DFS
Sumber : Dokumen Penulis

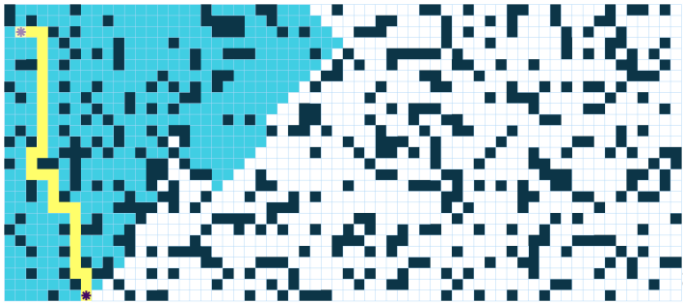
Hasil yang didapatkan :

- Simpul yang dikunjungi berjumlah : 1120 simpul
- Panjang jalur solusi : 729 langkah

E. Contoh Kasus 5



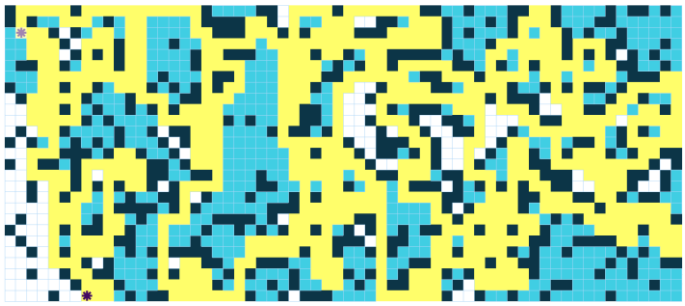
Gambar 4.13 Kasus 5
Sumber : Dokumen Penulis



Gambar 4.14 Solusi Kasus 5 dengan BFS
Sumber : Dokumen Penulis

Hasil yang didapatkan :

- Simpul yang dikunjungi berjumlah : 422 simpul
- Panjang jalur solusi : 32 langkah



Gambar 4.15 Solusi Kasus 5 dengan DFS
Sumber : Dokumen Penulis

Hasil yang didapatkan :

- Simpul yang dikunjungi berjumlah : 1131 simpul
- Panjang jalur solusi : 703 langkah

V. KESIMPULAN

1. Algoritma *Breadth First Search* dan *Depth First Search* dapat dimanfaatkan dalam *maze game*
2. Algoritma *Breadth First Search* dan *Depth First Search* cocok digunakan pada maze karena tidak memperhatikan bobot tiap simpul
3. Algoritma *Breadth First Search* cocok menggunakan struktur data *queue*
4. Algoritma *Depth First Search* cocok menggunakan struktur data *stack*
5. Algoritma *Breadth First Search* akan menghasilkan solusi dengan jalur terpendek selama pencarian
6. Algoritma *Depth First Search* akan menghasilkan solusi dengan jalur sesuai dengan cara aksesnya menuju simpul target

VI. UCAPAN TERIMA KASIH

Penulis mengucapkan puji syukur kepada Tuhan Yang Maha Esa atas berkat dan rahmatnya, penulis bisa menyelesaikan tugas makalah ini. Penulis juga mengucapkan terimakasih kepada Ibu Harili selaku dosen mata kuliah Matematika Diskrit, yang selama ini membimbing pembelajaran Matematika Diskrit yang sangat membantu pengerjaan makalah ini dan Bapak Rinaldi Munir, yang selama satu semester ini menyediakan website yang dapat dengan mudah diakses berisi materi-materi kuliah, Latihan-latihan soal untuk kuis dan ujian, dan semua dokumen pembelajaran, soal dan lainnya yang tentunya berguna dalam proses pembelajaran Matematika Diskrit.

REFERENCES

[1] Munir Rinaldi https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf, diakses 14 Desember 2021

[2] G. M. Emil, dkk. "Perbandingan Algoritma *Breadth First Search dan Depth First Search pda Game Mummy Maze Delux*". https://ejournal.unikadelasalle.ac.id/realtech/article/view/46/9, diakses 14 Desember 2021

[3] Rietno Anggun, "Makalah *Blind Search*". https://www.scribd.com/document/332880455/Makalah-Blind-Search, diakses 14 Desember 2021

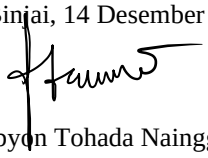
[4] http://rosyid.lecturer.pens.ac.id/kecerdasaran%20komputasional/Bab%204%20Algoritma%20Pencarian.pdf , diakses 14 Desember 2021

[5] https://clementmihailescu.github.io/Pathfinding-Visualizer/#H, diakses 14 Desember 2021

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.

Bintaj, 14 Desember 2021



Ghebyon Tohada Nainggolan
NIM : 13520079