Penerapan Algoritma Breadth First Search Pada Circular Maze

Muhammad Akyas David Al Aleey - 13520011
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): akyasdavid007@gmail.com

***Abstract*—Circular Maze atau labirin melingkar merupakan bentuk modifikasi dari labirin biasa yang setiap dinding dan jalannya berbentuk kurva melingkar dengan banyaknya simpangan yang dilalui. Perbedaan lainnya, circular maze terkadang diawali dari titik pusat labirin kemudian mencari jalan keluar ataupun sebaliknya. Terdapat berbagai algoritma yang dapat digunakan untuk mencari solusi dari puzzle rumit tersebut. Pada makalah ini, akan dibahas algoritma Breadth First Search yang dapat digunakan untuk menentukan apakah circular maze yang diberikan memiliki solusi untuk mencapai jalan keluar atau tidak dengan bantuan matriks ketetanggaan dalam menyimpan keterhubungan simpul satu dengan yang lain.**

Keywords—circular maze, breadth first search, matriks ketetanggaan, labirin

I. PENDAHULUAN

Perkembangan teknologi yang begitu pesat tentu memengaruhi berbagai aspek yang terlibat di dalamnya. Manusia berlomba-lomba dalam menciptakan suatu teknologi yang lebih canggih dan lebih praktis dari teknologi yang sudah ada. Hal tersebut beriringan dengan berkembangnya ilmu pengetahuan dewasa ini. Pemikiran-pemikiran manusia sekarang begitu canggih dan kreatif akibat zaman yang berubah. Berbagai algoritma yang lebih efektif dan efisien dari sebelumnya muncul sebagai buah hasil dari perkembangan ilmu pengetahuan tersebut. Terkadang algoritma tersebut dibangkitkan dari beberapa algoritma dasar atau konsep umum yang telah ada.

Adanya algoritma yang lebih efektif dan efisien mempermudah manusia dalam menyelesaikan persoalan-persoalan yang mereka hadapi mulai dari yang tergolong mudah hingga sulit. Dan seiring dengan masuknya zaman yang serba digital seperti ini, tak jarang permasalahan tersebut dapat diselesaikan dengan bantuan teknologi informasi atau komputasi. Dengan membuat suatu program komputer, permasalahan tersebut dapat diselesaikan dengan waktu yang singkat untuk setiap kasus yang mungkin. Setiap permasalahan tersebut memiliki cara penyelesaian berbeda-beda yang bersesuaian dengan suatu algoritma baik algoritma dasar maupun algoritma yang telah dikembangkan.

Persoalan labirin merupakan persoalan yang begitu erat kaitannya dengan materi graf. Sebuah sistem jalur yang rumit, berliku-liku, dan memiliki banyak jalan buntu tersebut begitu menarik untuk dimainkan. Labirin tersebut dapat berupa permainan di atas kertas, di layar, maupun dibuat dengan skala besar dengan menggunakan tanaman atau tembok yang cukup besar dan tinggi. Setiap peserta yang masuk ke dalam labirin ditantang untuk dapat melewati setiap jalan yang ada hingga menemukan jalan keluarnya. Balik lagi, seiring dengan berkembangnya kreativitas manusia, muncullah sebuah labirin yang berbentuk lingkaran atau *circular maze*. Di dalam *circular maze* tersebut, jalan labirin diatur sedemikian rupa sehingga peserta bergerak maju mundur melintasi bentuk melingkar melalui serangkaian kurva, berakhir di jantung atau pusat labirin. Dengan mengetahui desain dari *circular maze* tersebut, kita dapat membuat suatu program dengan algoritma sesuai yang menentukan apakah labirin tersebut memungkinkan untuk diselesaikan atau tidak.

Oleh karena itu, pada makalah ini, penulis akan membahas pengimplementasian algoritma Breadth-First Search untuk menentukan apakah peserta dapat melewati jalur melingkar tersebut sedemikian sehingga pusat labirin dapat tercapai.

II. TEORI DASAR

A. *Graf*

Graf merupakan struktur data matematika non-linear yang merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut. Graf terdiri dari pasangan himpunan (V, E) dengan notasi G = (V, E) yang dalam hal ini, V merupakan himpunan tak-kosong dari simpul-simpul (vertices). V = {v_1, v_2, v_3, …, v_n}. Sementara E merupakan himpunan sisi (edges) yang menghubungkan sepasang simpul. E = {e_1, e_2, e_3, …, e_n}.

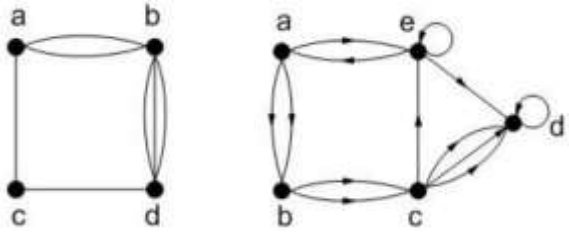
Graf digolongkan menjadi dua jenis berdasarkan ada tidaknya gelang atau sisi ganda pada suatu graf, diantaranya:

1. Graf sederhana (simple graph) yang tidak mengandung gelang maupun sisi ganda.
2. Graf tak-sederhana (unsimple graph) yang mengandung sisi ganda atau gelang. Graf tak-sederhana dibedakan lagi menjadi:
    - Graf ganda (multi-graph) yang mengandung sisi ganda.
    - Graf semu (pseudo-graph) yang mengandung sisi gelang.

Gambar 1. Graf ganda dan Graf semu

Selain itu, berdasarkan orientasi arah pada sisi, graf dibedakan lagi kedalam dua jenis, yaitu:

1. Graf tak-berarah (undirected graph) yang tidak memiliki orientasi arah.
2. Graf berarah (directed graph) yang setiap sisinya diberikan orientasi arah.

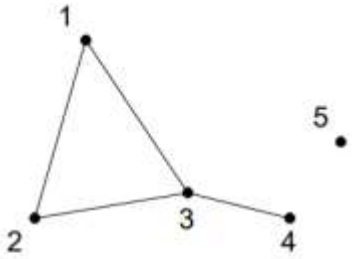


Gambar 2. Graf tak-berarah dan Graf berarah

Terdapat tiga macam cara sendiri dalam merepresentasikan graf. Diantaranya yaitu:

1. Matriks Ketetanggaan (adjacency matrix)

   Matriks ketetanggaan suatu graf G adalah suatu matriks yang baris dan kolomnya bernilai 0 atau 1 jika graf tersebut tak berbobot atau tiap sisinya tidak bernilai. Untuk setiap A = [a_{ij}], a_{ij} akan bernilai 1 jika simpul i dan j bertetangga. Sebaliknya, a_{ij} akan bernilai 0 jika simpul i dan j tidak bertetangga. Berikut contoh representasi graf menggunakan matriks ketetanggaan.

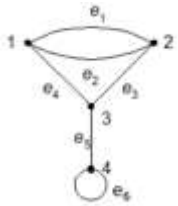


	1	2	3	4	5
1	0	1	1	0	0
2	1	0	1	0	0
3	1	1	0	1	0
4	0	0	1	0	0
5	0	0	0	0	0

Gambar 3. Representasi Matriks Ketetanggaan

2. Matriks Bersisian (incidency matrix)

   Pada matriks bersisian, untuk setiap A=[a_{ij}], a_{ij} akan bernilai 1 jika simpul i bersisian dengan sisi j. Sedangkan a_{ij} akan bernilai 0 jika simpul i tidak bersisian dengan sisi j. Berikut contoh representasi graf menggunakan matriks bersisian.

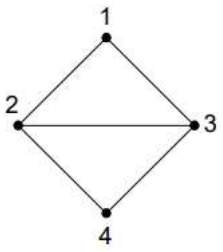


	e1	e2	e3	e4	e5	e6
1	1	1	0	1	0	0
2	1	1	1	0	0	0
3	0	0	1	1	1	0
4	0	0	0	0	1	1

Gambar 4. Representasi Matriks Bersisian

3. Senarai Ketetanggaan (adjacency list)

   Pada senarai ketetanggaan, keterhubungan antar simpul didefinisikan dengan menggunakan linked list. Setiap simpul memiliki list dari simpul-simpul yang terhubung dengan simpul tersebut. Berikut contoh representasi graf menggunakan senarai ketetanggaan.

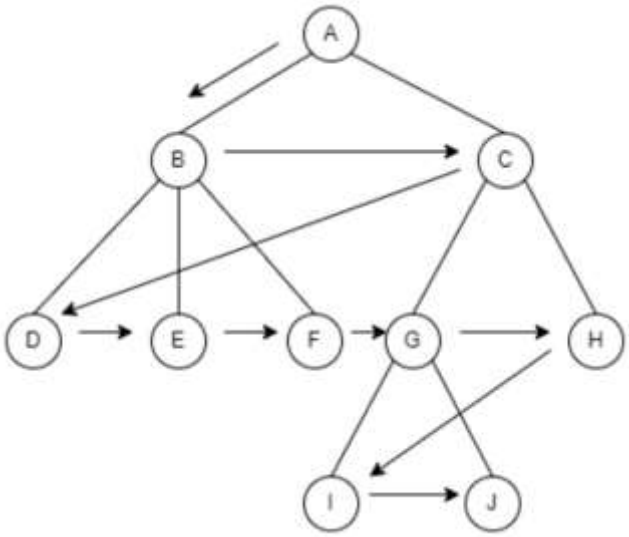


Simpul	Simpul Tetangga
1	2, 3
2	1, 3, 4
3	1, 2, 4
4	2, 3

Gambar 5. Representasi Senarai Ketetanggaan

B. Breadth-First Search

Algoritma Pencarian Melebar (*Breadth-First Search* atau BFS) merupakan salah satu algoritma pencarian pada graf yang dapat diterapkan baik pada graf berarah maupun tidak berarah. Algoritma ini merupakan algoritma yang menjadi pola dasar dan fundamental bagi banyak algoritma graf. Sesuai namanya, *Breadth-First Search* akan mengunjungi tiap simpul dengan cara yang sistematik secara melebar. Artinya, algoritma ini akan melakukan pencarian dengan mengunjungi setiap simpul / *vertices* pada sebuah graf secara preorder yaitu mengunjungi suatu simpul kemudian mengunjungi seluruh tetangga dari simpul tersebut. Pencarian tersebut dilakukan terus menerus hingga ditemukan simpul solusinya. Berikut merupakan ilustrasi penelusuran suatu graf dengan menggunakan algoritma *Breadth-First Search*.



Gambar 6. Ilustrasi Alur Pencarian pada Algoritma Breadth-First Search

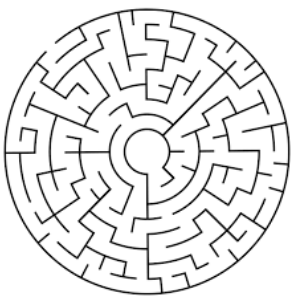
Secara lebih detail dan sistematis, algoritma dari pencarian *Breadth-First Search* dapat dilakukan seperti berikut.

1. Kunjungi suatu simpul v sebagai simpul awal.
2. Kunjungi semua simpul yang merupakan tetangga dari simpul v terlebih dahulu.
3. Kunjungi simpul yang bertetangga dengan simpul tadi dan belum pernah dikunjungi.
4. Lakukan proses 2 dan 3 seterusnya hingga semua simpul telah dikunjungi atau simpul solusi telah tercapai.

Beberapa struktur data yang dibutuhkan dalam algoritma pencarian *Breadth-First Search* diantaranya:

1. Matriks ketetanggaan (adjacency matrix) A = $[a_{ij}]$ yang berukuran nxn dengan aij bernilai 1 jika simpul i dan simpul j saling bertetangga, serta bernilai 0 jika simpul i dan simpul j tidak bertetangga.
2. Queue untuk menyimpan simpul yang telah dikunjungi.
3. Array of Boolean berukuran n yang berisi *true* jika simpul ke-indeks sudah dikunjungi serta *false* jika belum dikunjungi.

C. Circular Maze



Gambar 7. Contoh Circular Maze

Circular Maze atau labirin berbentuk lingkaran merupakan salah satu modifikasi dari labirin berbentuk persegi yang sering kali dilihat. Labirin sendiri dapat dideskripsikan sebagai suatu struktur puzzle atau jalan berpola yang diatur sedemikian pula sehingga jalan tersebut saling berliku-liku dan simpang siur, banyak belokan yang salah dan jalan buntu. Biasanya labirin memiliki satu jalan masuk dan satu jalan keluar. Dengan banyaknya simpangan di dalamnya, ada beberapa pilihan yang harus ditentukan untuk memecahkan permasalahan tersebut. Pilihan yang salah tentunya akan menghasilkan jalan buntu.

Sama halnya dengan prinsip dasar labirin, *circular maze* juga memiliki jalan berlika-liku dan simpangan yang begitu banyak. Hanya saja, bentuknya lingkaran dan peserta akan bergerak secara melingkar. Penentuan posisi start dapat dari pusat labirin atau pintu masuk yang ada di lapisan terluar labirin. Begitupun juga untuk penentuan posisi finish.

III. Implementasi Breadh-First Search Pada Penyelesaian Circular Maze

A. Asumsi Persoalan

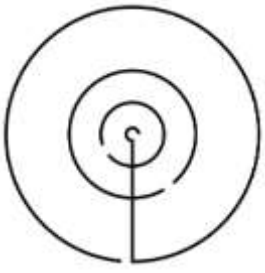
Untuk menyelesaikan persoalan ini, diasumsikan bahwa *circular maze* terdiri dari *n* lapisan atau dinding. Setiap dinding dapat berupa garis lurus yang menandakan bahwa garis tersebut merupakan dinding lurus dan garis kurva melingkar yang menandakan bahwa garis tersebut merupakan lapisan dari *circular maze*.

Berangkat dari hal tersebut, untuk dinding berbentuk kurva melingkar akan digambarkan dengan jari-jari *r* yang merupakan jarak dinding dari pusat, dan dua sudut θ_1 dan θ_2 yang menggambarkan sudut awal dan akhir dinding searah jarum jam. Perhatikan bahwa menukar kedua sudut akan mengubah dinding.

Kemudian, untuk dinding lurus digambarkan oleh sudut θ yang merupakan arah dinding, dan dua jari-jari $r_1 < r_2$ yang menggambarkan panjang awal dan akhir dinding.

Dalam penentuan arah, sudut diukur dalam derajat dengan spesifikasi yaitu sudut 0 sesuai dengan arah menunjuk ke arah utara atau searah dengan sumbu-y positif dan sudut tersebut meningkat searah jarum jam (maka arah timur sesuai dengan sudut 90).

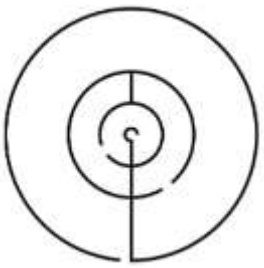
Berikut merupakan contoh gambaran *circular maze* berdasarkan spesifikasi yang telah diberikan di atas yang berturut-turut dapat dan tidak dapat diselesaikan.



Gambar 8. Contoh Circular Maze sesuai spesifikasi

Terdapat 5 data garis yang dibutuhkan dalam penggambaran circular maze tersebut, diantaranya:

1. Garis melingkar beradius 1 dengan sudut awal 180 dan berakhir di sudut 90.
2. Garis melingkar beradius 5 dengan sudut awal 250 dan berakhir di sudut 230.
3. Garis melingkar beradius 10 dengan sudut awal 150 dan berakhir di sudut 140.
4. Garis melingkar beradius 20 dengan sudut awal 185 dan berakhir di sudut 180.
5. Garis lurus dihitung dari radius 1 hingga radius 20 dengan arah 180.



Gambar 9. Contoh Circular Maze sesuai spesifikasi

Terdapat 6 data garis yang dibutuhkan dalam penggambaran circular maze tersebut dengan 5 data pertama sama seperti data pada gambar 8 dengan tambahan sebuah garis lurus dihitung dari radius 5 hingga radius 10 dengan arah 0.

B. Ide Penyelesaian

Untuk menyelesaikan kasus ini, pertama-tama penulis membuat batasan-batasan data yang boleh dimasukkan pada masukan, diantaranya:

1. Jumlah dinding labirin tidak lebih dari 5000 ($1 \leq n \leq 5000$).
2. Banyaknya lapisan tidak lebih dari 20 atau radius labirin kurang dari 20 ($1 \leq r \leq 20$).
3. Sudut dinding antara 0 hingga 359 derajat. Jika dinding berbentuk circular, maka sudut awal dan akhir tidak boleh sama.

Berikut merupakan potongan kode program dari penyelesaian persoalan *circular maze* menggunakan algoritma *Breadth-First Search*.

```
#include<bits/stdc++.h>
#define N 200001

using namespace std;
int circular[30][400];
int straight[30][400];
int head[N],to[N],nex[N],idx;
int f[N];
int x,y,z;
void addedge(int a,int b)
{
    nex[++idx] = head[a];
    head[a] = idx;
    to[idx] = b;
}
void solve()
{
    int n;
    scanf("%d",&n);
    char ch;
    idx = 0;
    for(int i = 0; i < 20; i++) {
        for(int j = 0; j <= 360; j++) {
            circular[i][j] = 0;
            straigt[i][j] = 0;
        }
    }
    for(int i = 0; i < N; i++) {
        f[i] = 0;
        head[i] = 0;
    }
    for(int i = 1; i <= n; i++) {
        cin >> ch;
        scanf("%d %d %d",&x,&y,&z);
        if (ch == 'C') {
            for(int i = y; i != z; i = (i+1)%360) {
                circular[x][i] = 1;
            }
        }
        else {
            if(x > y) {
                swap(x, y);
            }
            for(int i = x; i < y; i++) {
                straight [i][z] = 1;
            }
        }
    }
    for(int i = 0; i <= 20; i++) {
        for(int j = 0; j < 360; j++) {
            if(!straight[i][(j+1)%360]) {
                addedge(i*360+j, i*360+(j+1)%360);
                addedge(i*360+(j+1)%360, i*360+j);
            }
        }
    }
    for(int i = 1; i <= 20; i++) {
        for(int j = 0;j < 360; j++) {
            if(!circular[i][j]) {
                addedge((i-1)*360+j, i*360+j);
                addedge(i*360+j, (i-1)*360+j);
            }
        }
    }
    queue <int> q;
    q.push(0);
    f[0] = 1;
```

```
    while(!q.empty()) {
        int x = q.front();
        q.pop();
        for(int i = head[x]; i ; i = nex[i]){
            if(!f[to[i]]) {
                f[to[i]] = 1;
                q.push(to[i]);
            }
        }
    }
    int ans = 0;
    for(int i = 0; i < 360; i++) {
        ans |= f[20*360+i];
    }
    if(ans) {
        printf("YES\n");
    }
    else {
        printf("NO\n");
    }
}
int main() {
    int query;
    scanf("%d", &query);
    while(query--) {
        solve();
    }
    return 0;
}
```

Berdasarkan kode program yang telah dituliskan di atas, setiap titik-titik pada circular maze tersebut akan ditranslasikan kedalam sebuah matriks dengan jumlah baris yang merepresentasikan jumlah lapisan maze yang dalam hal ini berukuran 20 dan jumlah kolom yang merepresentasikan sudut dinding dari 0 hingga 359 derajat.

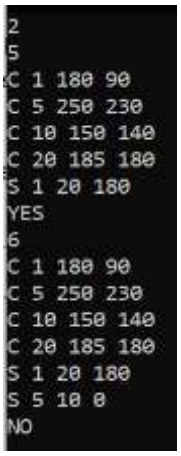
Karena pada labirin tersebut dinding terbagi menjadi dua jenis, yakni dinding circular dan dinding lurus, maka dibuat 2 buah matriks (matriks circular dan straight) dengan ukuran yang sudah dijelaskan di atas yang akan bernilai 1 jika pada radius dan derajat tersebut telah terisi oleh garis baik garis melingkar maupun garis lurus.

Sebelum memasuki algoritma utama, akan disimpan simpul-simpul bersebelahan yang tidak dilalui oleh suatu garis. Simpul-simpul yang saling bersebelahan tersebut merepresentasikan jalur kosong yang dapat dilewati. Karena ketetanggaan antar simpul disimpan dalam *adjacency matrix*, maka terdapat 4 cara yang dapat dilalui oleh pemain, yaitu jalan ke depan, kanan, kiri, dan belakang. Matriks circular akan menghubungkan simpul-simpul tetangga di depan dan belakang sedangkan matriks straight akan menghubungkan simpul-simpul tetangga di samping kanan dan kiri jika elemen pada matriks bernilai 0.

Pada saat pencatatan simpul yang bertetangga, seluruh simpul tersebut direpresentasikan lagi menjadi suatu array satu dimensi dengan ukuran 20*360+360. Simpul awal akan disimpan dalam array head dan simpul tetangganya akan disimpan dalam array to. Array nex akan menyimpan simpul yang akan ditelusuri sesuai dengan algoritma *Breadth-First Search* yaitu menyebar ke tetangga terdekat terlebih dahulu.

Proses penelusuran akan dilakukan terus menerus menggunakan algoritma *Breadth-First Search* yang memanfaatkan struktur data Queue atau antrian hingga mencapai lapisan ke-20 meskipun dinding yang ada tidak mencapai radius sebesar itu. Akan diperiksa untuk sekeliling garis melingkar di lapisan ke-20 mulai dari sudut 0 hingga sudut 359 derajat terdapat jalan keluar atau tidak. Dasar penentuan solusi akhir tersebut telah melewati pertimbangan bahwa untuk setiap *circular maze* beradius kurang atau sama dengan 20 yang memiliki jalan keluar, proses penelusuran akan tetap berjalan hingga mencapai radius terluar. Dan jika jalan keluar terletak di dinding dengan radius kurang dari 20, penentuan solusi akhir akan tetap benar.

Berikut merupakan hasil implementasi kode program yang telah dibuat dalam menentukan ada tidaknya jalan keluar pada kedua *circular maze* di atas.



Gambar 10. Contoh input dan output program

Dapat dilihat bahwa *circular maze* pertama memiliki jalan keluar dan *circular maze* kedua tidak memiliki jalan keluar.

IV. KESIMPULAN

Berdasarkan pembahasan di atas, algoritma *Breadth-First Search* merupakan algoritma pencarian dasar pada graf yang dapat digunakan dalam memecahkan persoalan *circular maze* atau labirin melingkar. Dalam kasus ini, matriks ketetanggaan diterapkan untuk mempresentasikan simpul-simpul tetangga yang merujuk ke indeks satuan jalan dan dinding penyusun labirin. Penelusuran diawali dari pusat lingkaran dan berakhir di lapisan beradius 20 meskipun dinding terluar labirin beradius kurang dari 20. Penentuan solusi didapatkan dari pengecekan keliling lapisan terluar apakah terdapat indeks satuan di antara itu yang telah dikunjungi.

ACKNOWLEDGMENT

Pertama dan yang utama, penulis ingin mengucapkan puji syukur kehadirat Tuhan Yang Maha Esa karena atas izin, rahmat, dan karunia-Nya, penulis dapat menyelesaikan makalah ini dengan baik dan tepat waktu. Penulis juga hendak

menyampaikan terima kasih kepada Dr. Nur Ulfa Maulidevi, S.T., M.Sc.selaku dosen pengampu mata kuliah IF2211 khususnya kelas K02 serta Tim Dosen IF2211 Strategi Algoritma yang senantiasa mengajar kami dengan sabar dan bersedia memberikan ilmu yang sangat bermanfaat sehingga penulis dapat tercerahkan dalam menyusun makalah ini. Dan juga penulis mengucapkan terima kasih kepada orang tua serta teman-teman penulis yang selalu mendukung secara maksimal baik selama proses pembelajaran maupun pembuatan makalah ini Tak lupa, penulis memohon maaf jika terdapat kesalahan penulisan, penyampaian, atau apapun itu baik yang disengaja maupun tidak.

REFERENCES

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf diakses pada tanggal 21 Mei 2022.

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag1.pdf diakses pada tanggal 21 Mei 2022.

[3] https://codeforces.com/problemset/problem/1662/O diakses pada tanggal 22 Mei 2022.

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, 23 Mei 2022

Muhammad Akyas David Al Aleey dan 13520011