Penerapan Strategi Algoritma *Branch and Bound* dan *DFS* pada *Micromouse*

Muhammad Tamiramin Hayat Suhendar 13519129
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13519129@std.stei.itb.ac.id

***Abstrak*—Perkembangan teknologi yang berjalan secara eksponensial membuat manusia berlomba untuk mengembangkannya.salah satu untuk mendorong kemajuannya dilakukannya ajang perlombaan atau kompetisi.salah satunya adalah kompetisi *micromouse*.*micromouse* merupakan permainan suatu robot tikus untuk mencapai kotak tujuan di dalam labirin dan Kembali ke titik awal.Makalah ini membahas mengenai salah dua algoritma yang dapat diterapkan terkait dengan kompetisi *micromouse*.algoritma yang diterapkan adalah algoritma *branch and bound* dan algoritma *backtracking*.**

Keywords—branch and bound; micromouse; algoritma;backtrack ; labirin

I. PENDAHULUAN

Perkembangan ilmu pengetahuan dan teknologi berjalan secara eksponensial.Ketika teknologi baru ditemukan maka tantangan kedepannya adalah membuat teknologi tersebut untuk lebih optimal. Optimal dapat dari segi waktu, harga, dan ruang.Riset-riset teknologi terbaru juga dijalankan untuk menemukan sesuatu yang baru.

Teknologi terbaru kebanyakan pertama kali dipegang oleh bagian militer.melihat kepada sejarah , ARPANET yang mengembangkan internet dan memakai internet untuk kepentingan militer olehAmerika Serikat.Setelah dirasa aman teknologi internet maka baru dibuka untuk khalayak umum.Begitu juga untuk teknologi-teknologi baru selanjutnya.

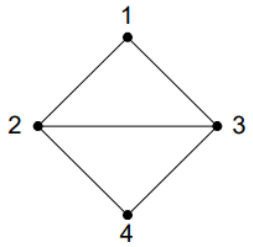
Terbukanya internet kepada publik membuat informasi dapat bergerak begitu cepat.Setelah perang dunia II usai setiap negara berlomba-lomba untuk menciptakan teknologi terbaru.Mulai dari Internet berkembang lagi teknologi terbaru salah satunya peta dunia digital.Peta dunia tersebut memberikan fitur untuk mencari tau jarak tempuh dari suatu titik ke titik lainnya.Pemakaian kendaraan juga dapat disesuaikan. Terdapat juga fitur untuk mencari rute tercepat *shortest path.*

Untuk mendorong kemajuan teknologi. Banyak kompetisi-kompetisi diadakan. Salah satu kompetisi tersebut adalah *micromouse*.*mocromouse* merupakan suatu kompetisi sudah ada sejak tahun 1970, mekanisme perlombaan *micromouse* adalah dengan membuat robot tikus kecil yang dijalankan pada sebuah labirin 16x16. Robot tikus ditempatkan pada titik awal yang sudah ditentukan dan bergerak menuju *goal* yang berada didalam maze dan Kembali ke titik awal tanpa dibantu.untuk mencapai tujuan tersebut pada robot tikus perlu diberi sutu algoritma pencarian jalan.salah dua dari algoritma tersebut adalah *Branch and Bound* dan *DFS*.Algoritma *DFS(Depth First Search)* merupakan algoritma yang dapat digunakan Ketika informasi dari tujuan tidak diketahui sehingga robot tikus mengeksplorasi labirin tersebut. Dan *Branch and Bound* digunakan ketika letak tujuan sudah diketahui.pemenang dari kompetsi *micromouse*_adalah robot tikus yang dapat menuju tujuan dan Kembali dengan waktu tercepat.

II. LANDASAN TEORI

A. *Graf*

Graf dipakai untuk merepresentasikan objek-objek dan hubungan antara objek tersebut.Graf terdiri dari simpul dan sisi.berikut adalah contoh dari graf



Gambar 1. Contoh Graf
Sumber : Graf (itb.ac.id), diakses pada 11 Mei 2021

Pada kasus *micromouse* graf digunakan untuk menganalisis peta labirin.beberapa terminologi pada graf yang sering dipakai adalah

1. Tetangga
   Dua simpul dikatakan bertetangga jika keduanya terhubung langsung(terdapat sisi).
2. Lintasan
   Panjang sisi dari simpul awal ke suatu simpul tujuan .
3. Sirkuit
   Lintasan yang diawali dan diakhiri dengan simpul yang sama (siklus).
4. Graf Berbobot

Graf yang setiap sisinya memiliki bobot atau diberi sebuah harga..

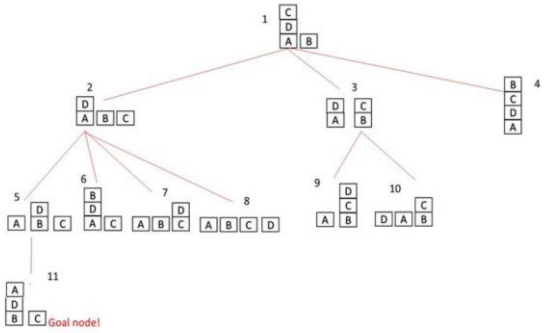
B. *Algoritma DFS(Depth First Search)*

Algoritma *DFS* merupakan algoritma pencarian simpul.pada graf tanpa mengetahui informasi apapun terkait graf/ tidak dengan menggunakan heuristik.Pada peta labirin *micromouse*, setiap petak dapat dianggap sebagai simpul dan setiap perbatasan petak dengan petak lainnya dapat dianggap sebagai sisi. Sehingga algoritma DFS dapat dipakai.Pada DFS dilakukan pembentukan pohon ruang status dinamis untuk mencapai tujuan.

Pada kasus ini traversal dimulai dari titik awal atau i.pada DFS umumnya simpul yang dikunjungi akan disimpan pada *stack* dan menggunakan konsep LIFO(*Last* In *First Out)* Langkah-langkah algortima DFS adalah sebagai berikut.

1. Kunjungi simpul i
2. Kunjungi simpul j yang bertetangga dengan simpul i dengan urutan prioritas barat,selatan,timur,dan utara.
3. Megulangi Langkah DFS mulai dari simpul j
4. Ketika mencapai suatu simpul k, dan tetangga dari simpul k semuanya sudah dikunjingi. Maka pencaharian dirunut balik ke simpul terakhir dikunjungi sebelumnya dan memiliki simpul tetangga w yang belum dikunjungi
5. Pencarian berakhir Ketika semua simpul di dalam labirin telah dikunjungi atau Ketika simpul tujuan telah tercapai

Karena pada labirin didalamnya terdapat simpul tujuan dan algoritma DFS merupakan algoritma yang komplit, maka pemberhentian dari algoritma DFS adalah saat bertemu/mengunjungi simpul tujuan.



Gambar 2. Contoh DFS

Sumber : PowerPoint Presentation (itb.ac.id), diakses pada 12 Mei 2021

C. *Algoritma Beanch and Bound*

Algoritma *Branch and Bound* adalah algoritma BFS(*Breadth First Search*) ditambah dengan *least cost search.Branch and Bound* atau disingkat dengan B&B digunakan untuk menyelesaikan persoalan optimasi.Pada B&B setiap simpul diberi nilai *cost*(c(i)) yaitu nilai taksiran lintasan termurah ke simpul status tujuan yang melewati simpul i.pada kasus minimasi maka simpul yang di ekspan adalah simpul yang memiliki *cost* terkecil.

B&B memiliki fungsi pembatas.dan memangkas semua jalur yang tidak mengarah kepada solusi.pada B&B simoul yang dikunjungi dapat disimpan dalam bentuk *queue* antrian yang berkonsep FIFO(First In First Out).Kriteria Pemangkasan adalah nilai simpul yang tidak lebih baik dari simpul yang di ekspan saat ini.

Langkah-langkah pemecahan persoalan menggunakan B&B adalah sebagai berikut.

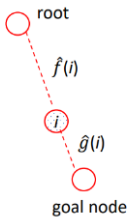
1. Masukan simpul akar ke dalam antrian, jika akar merupakan simpul solusi , maka solusi sudah ditemukan dan *stop*
2. Jika antrian kosong maka tidak ada solusi dan *stop*
3. Pilih dari antrian simpul i yang memiliki nilai cost(c(i)) paling kecil. Jika terdapat nilai yang sama maka pemilihan simpul menggunakan prioritas barat,selatan,timur,utara.
4. Jika simpul i merupakan solusi maka pencarian dihentikan, jika tidak maka bangkitkan semua anak anaknya(tetangga). Jika tidak memiliki tetangga maka Kembali pada Langkah kedua.
5. Untuk setiap tetangga dari simpul i , hitung nilai *cost* kemudian masukan ke dalam antrian.
6. Kembali pada Langkah nomor 2

Untuk *micromouse* nilai dari cost yang digunakan adalah

$$C(i) = f(i) + g(i) \text{ .....(1)}$$

C(i) adalah nilai dari cost, f(i) adalah ongkos untuk mencapai simpul i dari simpul akar, g(i) adalah ongkos dari simpul i ke simpul tujuan.Perhitungan g(i) menggunakan *manhatan distance*.

$$g(i) = |xi\text{-}xj| + |yi\text{-}yj| \text{ ……(2)}$$



Gambar 3. Visualisasi fungsi *cost*

Sumber : PowerPoint Presentation (itb.ac.id), diakses pada 12 Mei 2021

D. *Struktur Labirin*

Pada kompetisi *micromouse* menggunakan labirin dengan ukuran 16x16. Namun untuk kemudahan dalam perhitungan dan visualisasi maka digunakan peta labirin berukuran 4x4.

III. STRATEGI ALGORITMA

Penjelasan cara kerja kedua algoritma dalam penerapan robot tikus dibahas pada bab ini.Sebelumnya masukan yang berupa gambar labirin perlu di proses terlebih dahulu agar menjadi sebuah graf dan disimpan setiap tetangga dari suatu simpul.dicatat juga titik awal mulai.untuk graf dan simpul mungkin memerlukan struktur data tersendiri yang tidak dijelaskan didalam makalah ini.

A. Pendekatan Algoritma DFS

Diasumsikan dari masukan sudah di proses menjadi sebuah graf yang terdiri dari simpul dan sisi.secara teori algoritma DFS ini tidak optimal.Algoritma DFS memiliki kompleksitas waktu O(b^m) dengan b adalah jumlah maksimum percabangan pada graf. Dan m merupakan maksimum kedalaman dari ruang status.Keunggulan dari DFS adalah dari segi kompleksitas ruang.kompleksitas ruangnya adalah O(bm).Permasalahan dalam DFS karena DFS merupakan algoritma pencaharian tanpa informasi maka jika tidak di ketahui ada/tidaknya solusi pada graf. Maka algoritma DFS akan lama karena harus mencek semua simpul yang ada pada graf.namun pada kasus ini sudah diketahui bahwa solusi pasti berada didalam graf.

```
procedure DFS(vertice i, Graph G)
    if goal(i)or G.isAllvisited() then
        break
    else
// tandai simpul i telah dikunjungi
        i.visited ← true
// mengunjungi semua tetangga dari
// simpul i
        if !i.alladjvisitedvisited then
// menyimpan simpul ekspan dalam stack
            Lintasan.push(i)
            foreach vertice j in
i.adjacent :
                if j.visited = false then
                    DFS(j)
                endif
            endfor
        else
// semua tetangganya sudah dikunjungi
            Lintasan.pop()
        endif
    endif
```

Gambar 4. Pseudocode DFS
Sumber : dokumen pribadi

Pada Pseudocode diatas vertice merupakan sebuah strukrur data dan terdapat fungsi goal yang menerima masukan vertice dan mengembalikan nilai benar atau salah, benar jika simpul i merupakan simpul tujuan dan salah Ketika simpul i bukan simpul tujuan.kemudian mentranslasikan semua tetangganya..dalam DFS umumnya lintasan berupa stack, Lintasan merupakan stack penyimpanan lintasan dalam bentuk variabel global.

B. Pendekatan Algoritma Branch and Bound

Pada algoritma *Branch and Bound* terdapat fungsi pembatas dan fungsi *cost*.Ekspansi simpul diambil dari simpul yang memiliki nilai *cost* paling kecil.nilai *cost* setiap simpul dibangun Ketika simpul mengeksplorasi tetangganya dan memasukan simpul tetangga kedalam antrian.nilai *cost* merupakan jarak dari titik awal ke simpul tersebut ditambah dengan jarak dari simpul tersebut ke simpul tujuan. Karena pada labirin setiap petaknya berukuran sama maka jarak antar petak diseragamkan bernilai satu.Kemudian jaeak dari simpul ke simpul tujuan dihitung dengan menggunakan Manhattan *distance*.

```
procedure B_and_B(Graph G,vertice goal)
    Q ← new Queue
    fn ← -1
// tandai simpul start yang pertama kali
// dikunjungi
    G.start.visited ← true
    Q.add(G.start)
    while (isnotEmpty(Q)):
// mengambil simpul yang memiliki nilai
// cost paling kecil
        X = Q.min()
// fn merupakan jumlah Langkah yang
// sudah dilewati
        fn ← fn + 1
        foreach vertice j in x.adjacent :
            if !j.visited then
                j.hitungcost(fn,G.start)
                j.visited ← true
                Q.add(j)
// menyimpan lintasan, arr adalah
// variabel global
        Savepath(j,arr)
            endif
        endfor
```

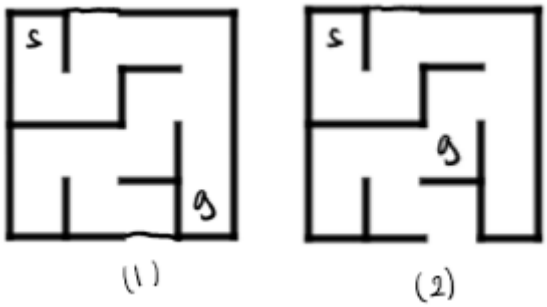
```
    endwhile
function hitungcost(fn.G.start)
    → fn + abs(this.x-G.start.x) +
abs(this.y – G.start.y)
```

Gambar 5. Pseudocode DFS
Sumber : dokumen Pribadi

Proses Ekspansi dari algoritma *branch and bound* menggunakan mekanisme BFS(*Breadth First Search*).dengan menggunakan antrian atau *queue*.Pada Pesudocode Gambar 5 diasumsikan telah ada kelas graf, vertice atau simpul, queue. Sehingga dapat langsung dipakai.terdapat juga fungsi untuk menghitung nilai *cost* untuk setiap simpul yang akan diekspan.untuk lintasan disimpan dalam variabel global *array*.fungsi Savepath digunakan untuk menyimpan jalur lintasan yang dilalui oleh *micromouse* sehingga untuk Kembali ke titik awal *micromouse* dapat menggunakan lintasan sebelumnya.

IV. Hasil Eksperimen

Untuk mempermudah Simulasi maka penulis menggunakan peta berukuran 4x4. Ukuran tersebut merupakan satu perempat dari ukuran kompetisi *micromouse* yang sebenarnya, diharapkan hasil yang didapat bisa mencerminkan jika dilakukan dengan ukuran yang penuh.

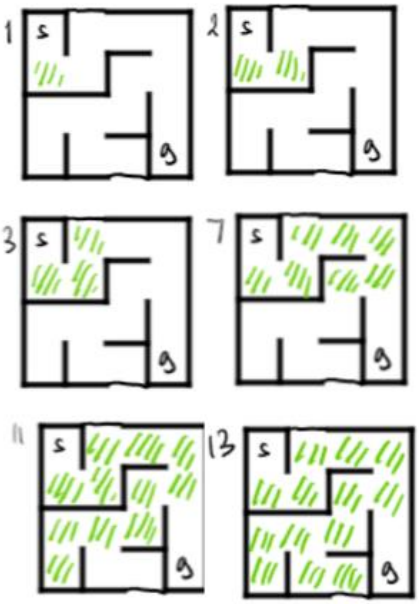


Gambar 4. Labirin yang digunakan
sumber : dokumen pribadi

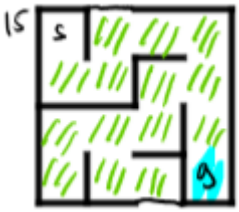
S merupakan titik awal robot tikus dan g adalah tempat tujuan dari tikus.

A. *Algoritma DFS*

Berikut adalah langkah Langkah yang diambil jika menggunakan algoritma DFS.Langkah digambarkan dalam bentuk gambar, warna hijau merupakan lintasan yang diambil.



Gambar 5. Langkah algoritma DFS labirin 1
sumber : dokumen pribadi



Gambar 6. Hasil akhir Algoritma DFS labirin 1
sumber : dokumen pribadi

Algoritma DFS secara tidak langsung menggunakan *backtrack* atau runut balik.dapat dilihat untuk langkah 1,2 dan 3 algoritma DFS melakukan ekspansi petak yang tersedia saja hingga pada langkah ke 6.terdapat dua simpul/petak yang dapat diekspan. Namun karena algoritma DFS didesain untuk memprioritaskan barat,selatan,timur,baru kemudian utara, maka pada langkah ke 7 simpul/petak yang diekspan adalah petak yang berada di sebelah barat.begitu juga yang terjadi pada langkah ke-9.

Kasus labirin 1 merupakan kasus terburuk pada algoritma DFS karena pencarian dilakukan pada setiap petak, dan petak solusi adalah petak terakhir yang di kunjungi.

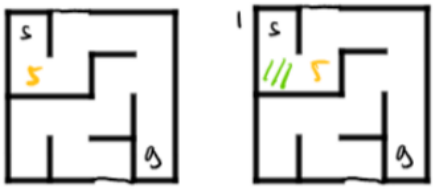
Pada Kasus Labirin kedua merupakan kasus yang baik karena simpul solusi yang ditemukan/dikunjungi bukan meupakan simpul terakhir yang dikunjungi.

Gambar 7. Hasil akhir Algoritma DFS labirin 2
sumber : dokumen pribadi

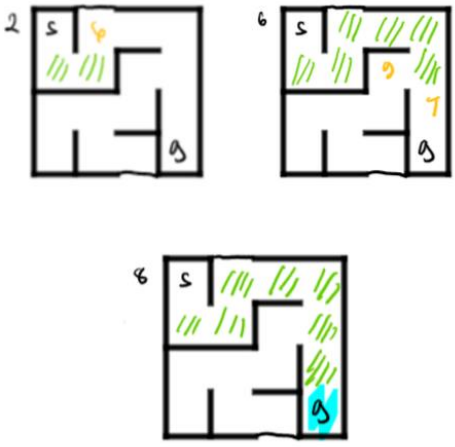
Dapat dilihat pada Labirin kedua, hanya memerlukan 8 langkah untuk algoritma DFS mencapai simpul solusinya.Delapan Langkah dari 15 langkah, dapat juga dianggap sebagai langkah rata rata pada algoritma DFS.

B. Algoritma Branch and Bound

Berikut Langkah-langkah yang diambil dengan menggunakan algoritma *branch and bound*.petak yang diberi warna hijau adalah petak yang diambil atau lintasan. Dan angka yang berwarna kuning merupakan nilai dari *cost*. Nilai dari *cost* dibangkitkan Ketika simpul sekarang ingin mengekspan node.

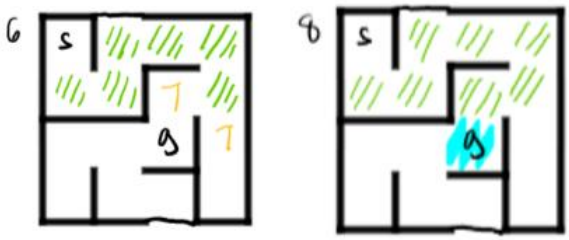


Gambar 8. Langkah 0 dan langkah 1
sumber : dokumen pribadi



Gambar 9. Langkah 0 dan langkah 1 labirin 1
sumber : dokumen pribadi

Hal yang menarik adalah ketika petaknya bertemu pada percabangan. maka untuk kasus labirin 1 ekspan simpul ditentukan oleh nilai *cost* yang paling minimum.sehingga lebih efisien untuk mencapai tujuan dan tidak membuang Langkah.hanya membutuhkan delapan Langkah untuk mencapai petak tujuan.



Gambar 10. Langkah 0 dan langkah 1 labirin 2
sumber : dokumen pribadi

Pada Labirin yang kedua, ketika simpul tujuan dipindahkan pada posisi (3,2) algoritma *branch and bound* juga membutuhkan delapan langkah.

V. KESIMPULAN

Penerapan dari ilmu strategi algoritma sangat meluas.salah satunya dapat dipakai untuk pencarian solusi pada kompetisi *micromouse*.dari eksperimen yang dilakukan untuk algoritma DFS dapat dipakai untuk mencari petak tujuan namun jumlah langkah yang ditempuh tidak optimal tetapi pada DFS *micromouse* dapat melakukan *backtrack*, karena penyimpanan lintasan yang berupa stack. Algoritma DFS cocok digunakan untuk pencarian ketika simpul tujuan nya pasti ada namun tidak diketahui letaknya.

Pada algoritma *branch and bound* pencarian solusi memiliki Langkah yang lebih sedikit disbanding algoritma DFS.algoritma *branch and bound* cukup optimal untuk menyelesaikan masalah pencarian simpul tujuan.namu kelemahan yang dimiliki adalah ketika labirin memiliki banyak percabangan dan ukurannya besar maka proses perhitungan cukup memakan waktu.sehingga untuk ukuran labirin yang kecil penggunaan algoritma *branch and bound* dapat dipertimbangkan.kelemahan yang kedua adalah letak dari simpul tujuan harus sudah diketahui sebelum memulai kompetisi dikarenakan lokasi dari petak tujuan dimanfaatkan untuk menghitung nilai *cost* pada bagian heuristik.

Untuk kasus terbaik algoritma DFS dapat menyaigi algoritma *branch and bound* yang tergambarkan pada kasus labirin 2.namun kedua algoritma DFS dan *branch and bound* tidak dapat dibandingkan begitu saja, karena algoritma DFS adalah algoritma tanpa informasi dan B&B adalah algoritma dengan informasi .Selain kedua algoritma tersebut dapat juga dipertimbangkan algoritma A star dan flood fill untuk menyelesaikan perosoalan pada kompetisi *micromouse*.

VI. PENUTUP

Puji Syukur Alhamdulillah penulis ucapkan kepada Allah Swt.,yang telah memberikan nikmat ilmu, nikmat sehat secara jasmani,rohani dan nikmat lainnya yang tidak bisa disebutkan.terimakasih penulis ucapkan juga kepada teman-teman seperjuangan dan orang tua yang selalu mendukung dalam penyelesaian makalah ini.Sehingga penulis dapat menyelesaikan makalah dengan tepat waktu.terimakasih juga

saya sampaikan kepada Dosen Strategi Algoritma K03,Dwi Hendratmo Widyantoro yang telah menyampaikan ilmu dan membimbing mahasiswa untuk menyelesaikan mata kuliah strategi algoritma.Dengan adanya makalah "Penerapan Strategi Algoritma *Branch and Bound* dan DFS pada *Micromouse"* Penulis berharap pembaca dapat lebih tersadarkan luasnya implementasi dari ilmu strategi algoritma, dan strategi B&B dan DFS dapat dimodifikasi untuk memenangkan kompetisi *micromouse*.

PRANALA VIDEO DI YOUTUBE

https:/youtu.be/vaD45ZrH1FE

REFERENSI

[1] Micromouseonline.2017.Introduction - Micromouse Online.Diambil dari www.micromouseonline/micromouse-book/introduction pada tanggal 10 Mei 2021.

[2] Munir,R.2021.Breadth/Depth First Search (BFS/DFS) bagian 1. Diambil dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag1.pdf diakses pada 10 Mei 2021.

[3] Munir,R.2020.Graf(Bag.1).Diambil dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf diakses pada 10 Meri 2021

[4] Munir,R.2021.Algoritma Branch and Bound Bagian 1.Diambil dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Branch-and-Bound-2021-Bagian1.pdf diakses pada 12 Mei 2021

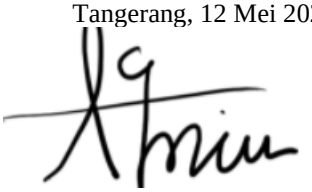
[5] Munir,R.2021. Breadth/Depth First Search (BFS/DFS) bagian 2.Diambil dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag2.pdf diakses pada 12 Mei 2021

[6] Service,JGB.2021.Maze Generator.Diambil dari http://www.mazegenerator.net/ diakses pada 12 Mei 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.

Tangerang, 12 Mei 2021



Muhammad Tamiramin Hayat Suhendar 13519129