

LAPORAN
Tugas Besar 2 IF 2211 Strategi Algoritma
Pengaplikasian Algoritma BFS dan DFS dalam Fitur People You May Know
Jejaring Sosial Facebook



DISUSUN OLEH

| | |
|-------------------------|--------------|
| Zaidan Naufal Sudrajat | 13518021 K03 |
| M. Reyhanullah Budiaman | 13519045 K01 |
| Muhammad Fikri .N | 13519069 K02 |

INSTITUT TEKNOLOGI BANDUNG
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
TEKNIK INFORMATIKA

2020/2021

BAB I

DESKRIPSI MASALAH

Bermain media sosial merupakan aktivitas yang sangat menyenangkan. Media sosial adalah sebuah media daring, dengan para penggunanya bisa dengan mudah berpartisipasi, berbagi, dan menciptakan isi blog, jejaring sosial, wiki, forum dan dunia virtual. Dari berbagai jenis media sosial, jejaring sosial merupakan jenis yang paling diminati oleh orang banyak. Popularitas jejaring sosial seperti facebook, twitter, Instagram, dan lainnya semakin meningkat dalam beberapa tahun terakhir. Pengguna media sosial pun berasal dari berbagai kalangan, mulai dari anak SD hingga orang tua berusia lanjut. Salah satu manfaat dari jejaring sosial adalah pengguna dapat berkomunikasi dengan orang-orang dari berbagai belahan dunia, baik orang yang sudah dikenal, maupun orang yang belum pernah dikenal sebelumnya.

Facebook sebagai salah satu pelopor jejaring sosial sejak tahun 2004 kini telah menembus angka 2,6 miliar pengguna. Fitur friend recommendation menjadi sangat penting karena banyaknya jumlah pengguna Facebook. Fitur bernama "People You May Know" ini dapat memberikan pengguna rekomendasi teman yang sebaiknya di-add, misalnya teman sekolah, teman kuliah, mantan pacar, atau orang yang kita kenal lewat suatu kegiatan tertentu. Faktor utama saran pertemanan melalui fitur tersebut adalah berdasarkan mutual friend yang dimiliki oleh kedua akun pengguna. Misalnya pengguna A dan C belum berteman di facebook, tetapi keduanya berteman dengan pengguna B, berarti A dan C memiliki mutual friend yang sama, yaitu B. Semakin banyak mutual friend yang dimiliki antara kedua akun, maka semakin tinggi rekomendasi akun tersebut untuk di-add. Selain itu, di setiap social media, termasuk Facebook, pengguna dapat mengeksplorasi akun-akun pengguna lainnya yang tidak memiliki mutual friends sama sekali. Akan tetapi, dengan menelusuri graf pertemanan antar akun sehingga kita dapat mengetahui 'jarak' antar akun agar bisa saling terhubung dan berteman.

Dalam tugas ini kami ditugaskan untuk membangun sebuah aplikasi GUI sederhana yang dapat memodelkan beberapa fitur dari People You May Know dalam jejaring sosial media (Social Network). Dengan memanfaatkan algoritma Breadth First Search (BFS) dan Depth First Search (DFS), sehingga dapat menelusuri social network untuk mendapatkan rekomendasi teman seperti pada fitur People You May Know. Selain untuk mendapatkan rekomendasi teman, diminta untuk mengembangkan fitur lain agar dua akun yang belum berteman dan tidak memiliki mutual friends sama sekali bisa berkenalan melalui jalur tertentu.

BAB II

LANDASAN TEORI

A. Traversal Graf

Traversal graf merupakan proses pencarian solusi dari suatu graf. Graf tersebut merupakan representasi dari sebuah persoalan yang ingin diselesaikan. Algoritma traversal graf adalah cara atau metode untuk mengunjungi simpul dengan cara yang sistematis. Contoh algoritma traversal graf adalah pencarian melebar (*Breadth First Search* / BFS) dan pencarian mendalam (*Depth First Search*) dengan asumsi graf terhubung.

Algoritma pencarian solusi :

1. Tanpa Informasi (*Uninformed* / *blind search*)

Algoritma pencarian solusi yang tidak memiliki informasi tambahan apapun.

Contoh : BFS, DFS, *Depth Limited Search*, *Iterative Deepening Search*, *Uniform Cost Search*.

2. Dengan informasi (*Informed search*)

Algoritma pencarian solusi yang berbasis heuristik.

Contoh : *Best First Search* dan A^*

Representasi graf dalam proses pencarian :

1. Graf Statis
2. Graf

B. BFS

BFS / *Breadth First Search* / Pencarian melebar merupakan algoritma pencarian secara melebar mulai dari titik awal pencarian hingga menemukan titik tujuan. Pencarian diawali dari titik awal kemudian mengunjungi titik-titik yang bertetangga/sejajar terlebih dahulu. Jika masih tidak ditemukan, dilanjutkan dengan mengunjungi titik berikutnya dengan syarat masih bertetangga.

C. DFS

DFS / *Depth First Search* / Pencarian mendalam merupakan suatu algoritma pencarian secara mendalam mulai dari titik awal pencarian hingga menemukan titik tujuan. Pencarian dilakukan diawali dari titik awal lalu menuju titik berikutnya yang akan ditelusuri hingga titik-titik yang lebih mendalam.

D. C# Desktop Application Development

Pada tugas besar ini digunakan framework *Windows Form App Project .Net*. *Windows Form App* dibuat pada *Windows Graphics Device Interface (GDI) Engine*. Framework memiliki fitur *drag and drop* yang memudahkan dalam pembuatan GUI. Pada pembuatan tugas ini dibuat GUI dan Algoritma secara paralel lalu kedua hal tersebut diintegrasikan pada akhir pembuatan.

BAB III

ANALISIS PEMECAHAN MASALAH

Persoalannya adalah untuk membangun sebuah aplikasi GUI yang dapat memodelkan fitur dari *People you may know* dalam jejaring sosial media (*social network*). Persoalan tersebut memanfaatkan algoritma BFS (*Breadth First Search*) dan DFS (*Depth First Search*). Program menerima input file eksternal yang berisi keterhubungan suatu node dengan node yang lain.

```
13
A B
A C
A D
B C
B E
B F
C F
C G
D G
D F
E H
E F
F H
```

Gambar 3.1 Contoh File eksternal

Sumber : File spek tubes

Baris pertama menyatakan banyaknya baris selanjutnya, kemudian baris setelahnya adalah keterhubungan antar satu node dengan node yang lain. Langkah-langkah pemecahan masalah adalah pertama dengan memodelkan suatu graf. Kami memodelkan graf dengan membuat sebuah class Graph yang memiliki atribut graphDict dengan tipe data SortedDictionary. Representasi tersebut memiliki key semua node graf dan untuk setiap key nya adalah tetangga node tersebut yang disimpan dalam sebuah list.

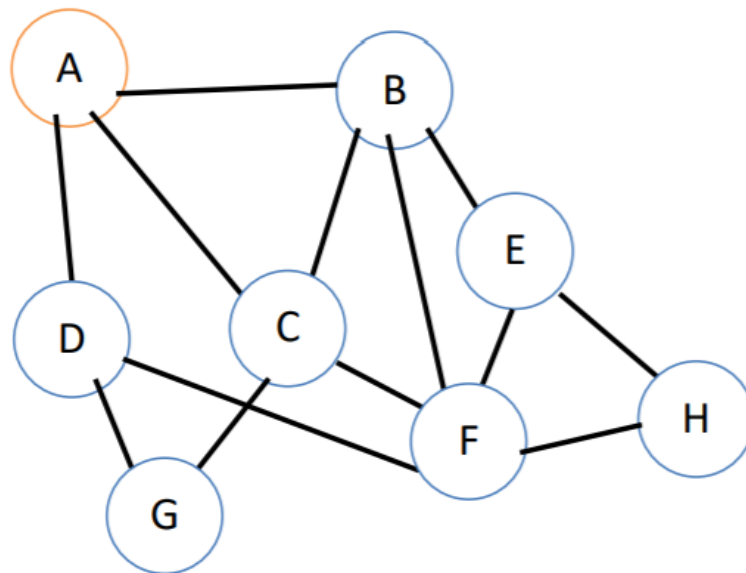
Class Graph

Class Graph adalah kelas yang digunakan untuk merepresentasikan persoalan keterhubungan pertemanan suatu jaringan sosial media. Tipe data yang digunakan adalah SortedDictionary.

Beberapa method yang ada pada kelas ini adalah sebagai berikut.

1. Graph()
-> Digunakan untuk konstruktor graf
2. AddEdge(string v1, string v2)
-> Digunakan untuk menambahkan sisi graf yang pada file eksternalnya memiliki 2 string pada 1 barisnya
3. AddEdge(string v1)
-> Digunakan untuk menambahkan sisi graf yang pada file eksternalnya hanya memiliki 1 string pada 1 barisnya (overloading AddEdge(string v1, string v2))

4. `sorting_value()`
-> Digunakan untuk mengurutkan value dari Dictionary nya (Graph) yang merupakan list of string dari tetangga vertex tersebut
5. `getGraph()`
-> Digunakan untuk mengambil graf
6. `PrintGraph()`
-> Menuliskan graf ke Console
7. `GetTotalVertices()`
-> Mengembalikan jumlah vertices(simpul) pada graf
8. `GetDegree()`
-> Mengembalikan jumlah simpul tetangga
9. `GetVertices()`
-> Mengembalikan semua vertices(simpul) pada graf
10. `GetListOfEdgesFrom()`
-> Mengembalikan seluruh simpul yang bertetangga dengan simpul yang diberikan pada parameter



Gambar 3.2 Visualisasi Graf

Pada gambar di atas adalah visualisasi graf yang terdapat pada file eksternal gambar 3.1

Proses mapping persoalan menjadi elemen - elemen algoritma BFS dan DFS adalah kami membuat kelas turunan graph yaitu BFS dan DFS.

1. Class BFS

BFS merupakan kelas turunan dari kelas Graph. Kelas ini memiliki atribut Ans dengan tipe list of string, antrian dengan tipe queue dengan elemen bertipe list of string, dan antrian_pair yang bertipe queue dengan elemen bertipe string. Ans merupakan atribut untuk menyimpan hasil pencarian pada graf dengan metode BFS. Atribut antrian digunakan untuk menyimpan state yang merupakan path dari node asal ke node tujuan. Atribut antrian_pair merupakan pasangan dari elemen pada antrian yang menyatakan node yang sedang dikunjungi pada path di atribut antrian.

Beberapa method yang ada pada kelas ini adalah sebagai berikut.

- a. BFS()
 - Konstruktors dari kelas BFS
- b. copyGraf(Graph g1)
 - Menyalin isi dari graph g1 ke objek saat ini.
- c. empty_antrian()
 - Mengosongkan isi atribut antrian dan antrian_pair
- d. Enqueue(List<string> kandidatSolusi, string node)
 - Melakukan enqueue kandidatSolusi pada atribut antrian dan enqueue node pada antrian_pair
- e. Dequeue()
 - Melakukan dequeue pada atribut antrian dan antrian_pair
- f. GetBFSAnswer(string from, string goal)
 - Implementasi algoritma *Breadth First Search*. Atribut Ans diisi dan dikembalikan jika terdapat jawaban. Akan mengembalikan list of string yang kosong jika tidak ditemukan.

2. Class DFS

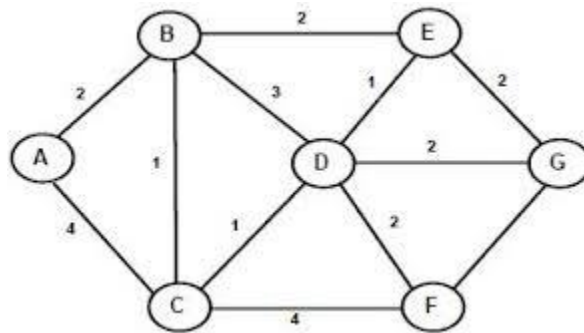
DFS merupakan kelas turunan dari kelas Graph. Kelas ini memiliki atribut visited, visited_temp, ans, dan ans_temp dengan tipe list of string. Atribut visited dan visited_temp digunakan untuk menyimpan node / node - node yang sudah pernah dilalui agar tidak mengunjungi node tersebut kembali. Untuk atribut visited digunakan untuk menyimpan node yang dikunjungi ketika menggunakan fitur explore friends. Untuk atribut visited_temp digunakan untuk menyimpan node yang dikunjungi ketika menggunakan fitur friends recommendation.

Class ini memiliki beberapa method yaitu :

- a. DFS()
 - > Digunakan untuk meng construct DFS
- b. isExist(string vertex, List<string> list)
 - > Digunakan untuk mengetahui apakah vertex ada pada list atau tidak. Method ini akan return true atau false
- c. GetAnswerFR(string vertex)
 - > Digunakan untuk menginisialisasi pencarian friend recommendation.
- d. GetAnswer(string vertex, string goal)
 - > Digunakan untuk melanjutkan inisialisasi friend recommendation.

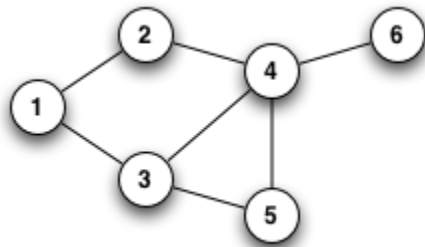
- e. recursiveFR(string vertex, string goal, List<string> visited, int iterasi, List<string> neighbour_vertex, SortedDictionary<string, List<string>> graphDFS)
-> Digunakan untuk melakukan rekursif dalam pencarian friend recommendation
- f. GetAnswerEF(string vertex, string goal)
-> Digunakan untuk menginisialisasi pencarian explore friends
- g. recursiveEF(string vertex, string goal, List<string> visited, int iterasi)
-> Digunakan untuk melakukan rekursif dalam pencarian explore friends
- h. print_ans_explore ()
-> Digunakan untuk print hasil explore friends yang disimpan pada atribut ans. Atribut ans diisi ketika pencarian rekursif explore friends (GetAnswerEF dan recursiveEF)
- i. print_ans_recommendation()
-> Digunakan untuk print hasil friends recommendation yang disimpan pada atribut ans_temp. Atribut ans diisi ketika pencarian rekursif explore friends (GetAnswerFR dan recursiveFR)
- j. copyGraf(Graph g1)
-> Digunakan untuk mengcopy graf

Contoh ilustrasi kasus graf yang berbeda dari yang gambar 3.1.



Dengan mengabaikan bobot dari setiap edge, maka isi file eksternal nya adalah :

12
A B
A C
B C
B D
B E
C D
C F
D E
D G
D F
E G
F G

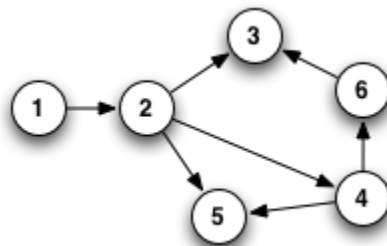


Maka isi file externalnya yaitu:

7
1 3
1 2
2 4
3 4
3 5
4 5
4 6

Sumber

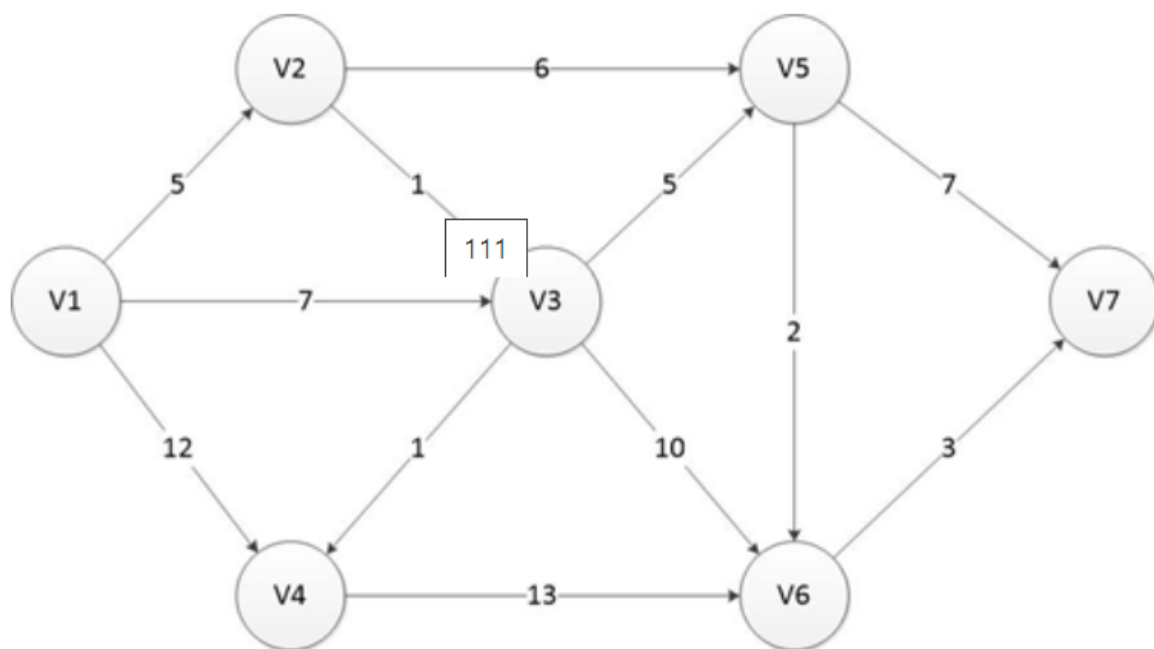
<http://think-like-a-git.net/sections/graph-theory/directed-versus-undirected-graphs.html>



Sumber : <http://think-like-a-git.net/sections/graph-theory/directed-versus-undirected-graphs.html>

Maka isi file eksternal nya adalah :

```
7
1 2
2 3
2 5
2 4
4 5
4 5
6 3
```



12

v1 v2

v1 v4

v1 v3

V2 v3

V2 v5

V3 v4

V3 v5

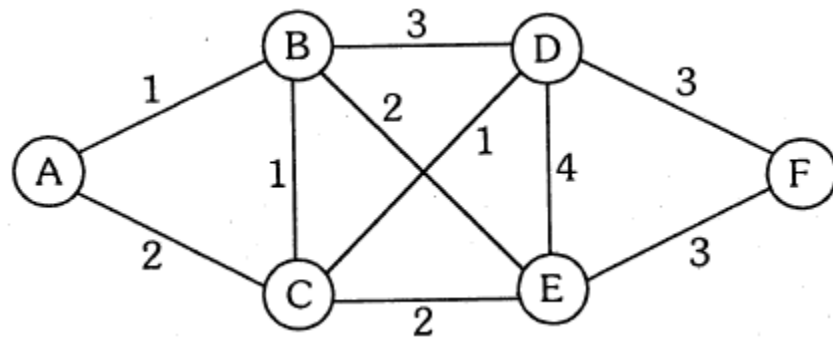
V3 v6

V4 v6

V5 v6

V5 v7

V6 v7



BAB IV

ANALISIS PEMECAHAN MASALAH

1. Implementasi program

Pada pengerjaan tugas besar ini digunakan Windows Form Application sehingga bentuk program utama benar benar berlingkup pada form. Berikut Pseudocode dari jalannya program.

```
//inialisasi
Graph grafGlobal = new Graph()
Microsoft.Msagl.Drawing.Graph graph = new
Microsoft.Msagl.Drawing.Graph("graph")
Form1.InitializeComponent().

If (button1.GetClicked) :
    string filepath = File.browse() //mendapat path dari file yang
di browse
    string file = open(filepath).readAll() // membaca semua file
    GrafGlobal = changeFiletoGraph(file) // mengubah file yang
dibaca menjadi sebuah Graf
    visualizeGraph(GrafGlobal) // menampilkan graf pada GUI
    if (bfs.getChecked):
        calculateFriendRecommendationBFS(Verteks X, Graph graf) //
membuat FR menggunakan BFS
        calculateExploreFriendBFS(Vertex from, Vertex to) //
membuat EF menggunakan BFS
        visualizeFriendRecommendationBFS() // menampilkan FR pada
GUI
        visualizeExploreFriendBFS() // menampilkan EF pada GUI
    else: //Dfs getChecked
        calculateFriendRecommendationDFS(Verteks X, Graph graf) //
membuat FR menggunakan DFS
        calculateExploreFriendDFS(Vertex from, Vertex to) //
membuat EF menggunakan DFS
        visualizeFriendRecommendationDFS() // menampilkan FR pada
GUI
        visualizeExploreFriendDFS() // menampilkan EF pada GUI
```

Berikut merupakan pseudocode dari BFS

```
USE Queue
USE List
function BFS (from, goal : string) → List of string
{ mencari solusi BFS }
KAMUS LOKAL
```

```

    Ans : List of string
    antrian : Queue of List of string
    antrian_pair : Queue of string
    temp_path : List of string
    curr_node : string
ALGORITMA
    Clear(antrian) { mengosongkan antrian }
    Clear(antrian_pair) { mengosongkan antrian }

    Enqueue(antrian, first)
    Add(temp_path, first)

    do
        temp_path ← Dequeue(antrian)
        curr_node ← Dequeue(antrian_pair)

        i traversal [{semua tetangga curr_node}]
        {implementasi foreach}
            if isContain(antrian, i) then
                Dequeue
            else
                if goals = i then
                    Ans ← temp_path
                    → Ans
                    Enqueue(antrian, i)
        while(isEmpty(antrian) ≠ false)

        {Tidak ditemukan solusi}
        Clear(Ans)
        →Ans

```

```

void GetDFSAnswerEF(string vertex, string goal) // explore
{
    int iterasi = 0;
    this.ans.Clear();
    recursiveEF(vertex,goal,this.visited, iterasi);
}

void recursiveEF(string vertex,string goal, List<string> visited,
int iterasi) // REKURSIF EXPLORE
{
    System.Diagnostics.Debug.WriteLine("iterasi : "
+iterasi);
    this.visited.Add(vertex);

    foreach (var v in ans)
    {
        System.Diagnostics.Debug.WriteLine(v + " | ");
    }
}

```

```

    }

    System.Diagnostics.Debug.WriteLine("Before");

    if (!(this.ans.Contains(vertex)))
    {
        this.ans.Add(vertex);
    }

    System.Diagnostics.Debug.WriteLine("after");

    foreach (var v in ans)
    {
        System.Diagnostics.Debug.WriteLine(v + " | ");
    }

    System.Diagnostics.Debug.WriteLine(" ");

    System.Diagnostics.Debug.WriteLine(vertex);
    if (this.graphDict.ContainsKey(vertex) &&
(this.ans.Count() > 0 || iterasi == 0)){

        iterasi ++;
        foreach (string next_vertex in
this.graphDict[vertex].Where(vertex => !visited.Contains(vertex))) {
            if (this.ans.Contains(goal))
            {

            }

            recursiveEF(next_vertex,goal,visited,
iterasi);
        }

        if (this.ans.Contains(goal))
        {
            System.Diagnostics.Debug.WriteLine("Skrg
di " + vertex);
            return;
        }

        System.Diagnostics.Debug.WriteLine("Count ans
" + this.ans.Count());

        if (this.ans.Count > 0)

```

```

        {
            //string temp = this.ans[this.ans.Count()
-1];
            this.ans.RemoveAt(this.ans.Count() - 1);
            //Console.WriteLine("Sudah kehapus" +
temp);
            System.Diagnostics.Debug.WriteLine("Count
ans " + this.ans.Count());
        }
        if (this.ans.Count != 0)
        {
            System.Diagnostics.Debug.WriteLine("BLM
NOL");
            recursiveEF(this.ans[this.ans.Count() -
1], goal, visited, iterasi);
        }

        System.Diagnostics.Debug.WriteLine("HALO");
        //Console.WriteLine("KOK HALO");
        foreach (var v in ans)
        {
            System.Diagnostics.Debug.WriteLine(v + "
| ");
        }
    }

    else
    {
        return;
    }
}

```

2. Penjelasan struktur data yang digunakan dalam program dan spesifikasi program

a. Struktur Data

1. Class Graf

Class Graf digunakan untuk menyimpan hubungan antara *Vertices* dan *Edge*.

Pada tugas besar ini digunakan class Dictionary yang mempunyai keys berupa string (menggambarkan *vertices*) dan *Edge* berupa list of string.

Graf mempunyai atribut :

- SortedDictionary<string, List<string>> graphDict;

Graf mempunyai method:

- `AddEdge(string v1, string v2());`
Menambahkan edge ke dictionary
- `void AddEdge(string v1);`
Overloading menambahkan edge ke dictionary apabila hanya satu vertices yang diberikan.
- `void sorting_value();`
Mersort dictionary berdasarkan graf.
- `SortedDictionary<string, List<string>> getGraph();` member
Mengembalikan Graf
- `void PrintGraph()`
Mengeluarkan output karakteristik dari graf
- `GetTotalVertices()`
Mengembalikan jumlah total vertices yang ada
- `List<string> GetVertices()`
Mendapatkan list of vertices
- `List<string> GetListOfEdgesFrom(string vertices)`
Mendapatkan list of edges dari suatu vertices yang diberikan.
- `void copyGraphV2(Graph global)`
Menyalin input graf ke atribut.
- `void copyGraph(Graph global)`
Mereference input graf.

2. Class BFS

Class BFS digunakan untuk menyimpan algoritma BFS yang terdapat pada program ini. Pada program ini ada dua fitur yang menggunakan algoritma BFS yaitu friends recommendation dan friend explore

3. Class DFS

Class DFS digunakan untuk menyimpan algoritma DFS yang terdapat pada program ini. Pada program ini ada dua fitur yang menggunakan algoritma DFS yaitu friends recommendation dan friend explore.

DFS mempunyai atribut :

- `List<string> visited`
- `List<string> visited_temp`
- `List<string> ans`
- `List<string> ans_temp`

DFS memiliki method :

- `DFS()`
-> Digunakan untuk meng construct DFS
- `isExist(string vertex, List<string> list)`
-> Digunakan untuk mengetahui apakah vertex ada pada list atau tidak.
Method ini akan return true atau false
- `GetAnswerFR(string vertex)`
-> Digunakan untuk menginisialisasi pencarian friend recommendation.
- `GetAnswer(string vertex, string goal)`

- > Digunakan untuk melanjutkan inisialisasi friend recommendation.
- recursiveFR(string vertex, string goal, List<string> visited, int iterasi, List<string> neighbour_vertex, SortedDictionary<string, List<string>> graphDFS)
 - > Digunakan untuk melakukan rekursif dalam pencarian friend recommendation
- GetAnswerEF(string vertex, string goal)
 - > Digunakan untuk menginisialisasi pencarian explore friends
- recursiveEF(string vertex, string goal, List<string> visited, int iterasi)
 - > Digunakan untuk melakukan rekursif dalam pencarian explore friends
- print_ans_explore ()
 - > Digunakan untuk print hasil explore friends yang disimpan pada atribut ans. Atribut ans diisi ketika pencarian rekursif explore friends (GetAnswerEF dan recursiveEF)
- print_ans_recommendation()
 - > Digunakan untuk print hasil friends recommendation yang disimpan pada atribut ans_temp. Atribut ans diisi ketika pencarian rekursif explore friends (GetAnswerFR dan recursiveFR)
- copyGraf(Graph g1)
 - > Digunakan untuk mengcopy graf

b. Spesifikasi Program

Aplikasi yang akan dibangun dibuat berbasis GUI.

Spesifikasi GUI:

1. Program dapat menerima input berkas file eksternal dan menampilkan visualisasi graph.
2. Program dapat memilih algoritma yang digunakan.
3. Program dapat memilih akun pertama dan menampilkan friends recommendation untuk akun tersebut.
4. Program dapat memilih akun kedua dan menampilkan jalur koneksi kedua akun dalam bentuk visualisasi graf dan teks bertuliskan jalur koneksi kedua akun.
5. GUI dapat dibuat sekreatif mungkin asalkan memuat 4 spesifikasi di atas

Spesifikasi Wajib:

1. Buatlah program dalam bahasa C# untuk melakukan penelusuran social network facebook sehingga diperoleh daftar rekomendasi teman yang sebaiknya di-add. Penelusuran harus memanfaatkan algoritma BFS dan DFS.
2. Awalnya program menerima sebuah berkas file eksternal yang berisi informasi pertemanan di facebook. Baris pertama merupakan sebuah integer N yang adalah banyaknya pertemanan antar akun di facebook. Sebanyak N baris berikutnya berisi dua buah string (A, B) yang menunjukkan akun A dan B sudah berteman .
3. Program kemudian dapat menampilkan visualisasi graf pertemanan berdasarkan informasi dari file eksternal tersebut. Graf pertemanan ini merupakan graf tidak berarah dan tidak berbobot. Setiap akun facebook direpresentasikan sebagai

sebuah node atau simpul pada graf. Jika dua akun berteman, maka kedua simpul pada graf akan dihubungkan dengan sebuah busur.

2. Tata cara penggunaan program
 1. Buka file “Zref.exe”.
 2. Tekan tombol browse, akan muncul open file dialog. Kemudian pilih file txt dengan format isi sebagai berikut.
 3. Program akan menampilkan visualisasi graf berdasarkan file yang diinput seperti pada gambar dibawah berikut.
 4. Pilih metode pencarian pada radio button (berlabel DFS atau BFS) yang berada di kanan atas.
 5. Pilih akun.
 6. Program ini memiliki fitur bernama Friend Recommendation yang menampilkan daftar teman rekomendasi berdasarkan jumlah mutual friend, yaitu jumlah simpul tetangga yang sama.
 7. Fitur kedua pada program ini bernama Explore Friend. Fitur ini menampilkan hasil pencarian menggunakan algoritma BFS dan DFS
 8. Pilih akun kedua yang akan dicarikan keterhubungannya.
 9. Tekan tombol “Generate”
 10. Hasil pencarian akan terlihat pada Text Box kanan bawah.

3. Hasil pengujian

A. Test Case 1

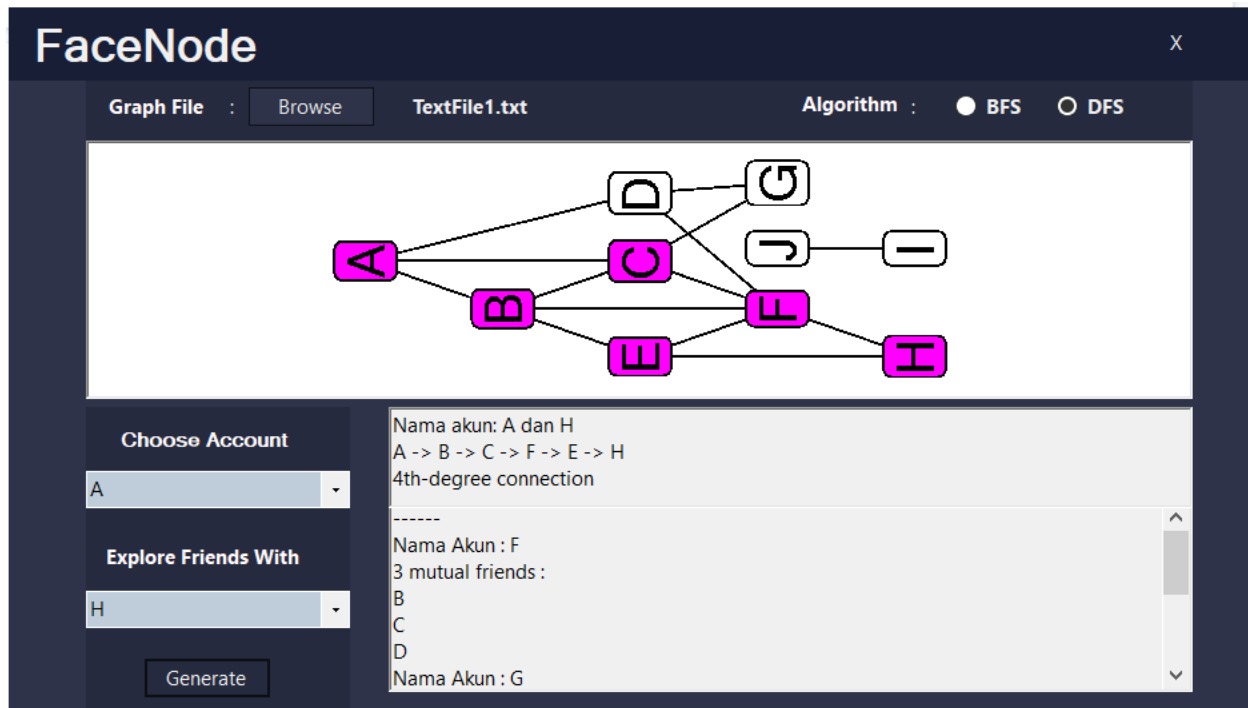
1. BFS dari A ke H

The screenshot shows the FaceNode application interface. At the top, the title bar says "FaceNode". Below it, there's a "Graph File" section with a "Browse" button and the filename "TextFile1.txt". To the right, the "Algorithm" section has two radio buttons: "BFS" (selected) and "DFS". The main area displays a graph with nodes A through I. Nodes A, B, E, and H are highlighted in pink. The graph structure is as follows: A is connected to B, C, and D. B is connected to C and E. C is connected to D, E, and F. D is connected to G. E is connected to F and H. F is connected to G and H. G is connected to I. I is connected to J. J is connected to I. The bottom section contains a "Choose Account" dropdown with "A" selected, an "Explore Friends With" dropdown with "H" selected, and a "Generate" button. To the right of these dropdowns, the search results are displayed in a text box:

```
Nama akun: A dan H
Connection Degree : 3
A --> B --> E --> H

Daftar rekomendasi teman untuk akun A:
Nama Akun: F
3 mutual friends:
B
C
D
```

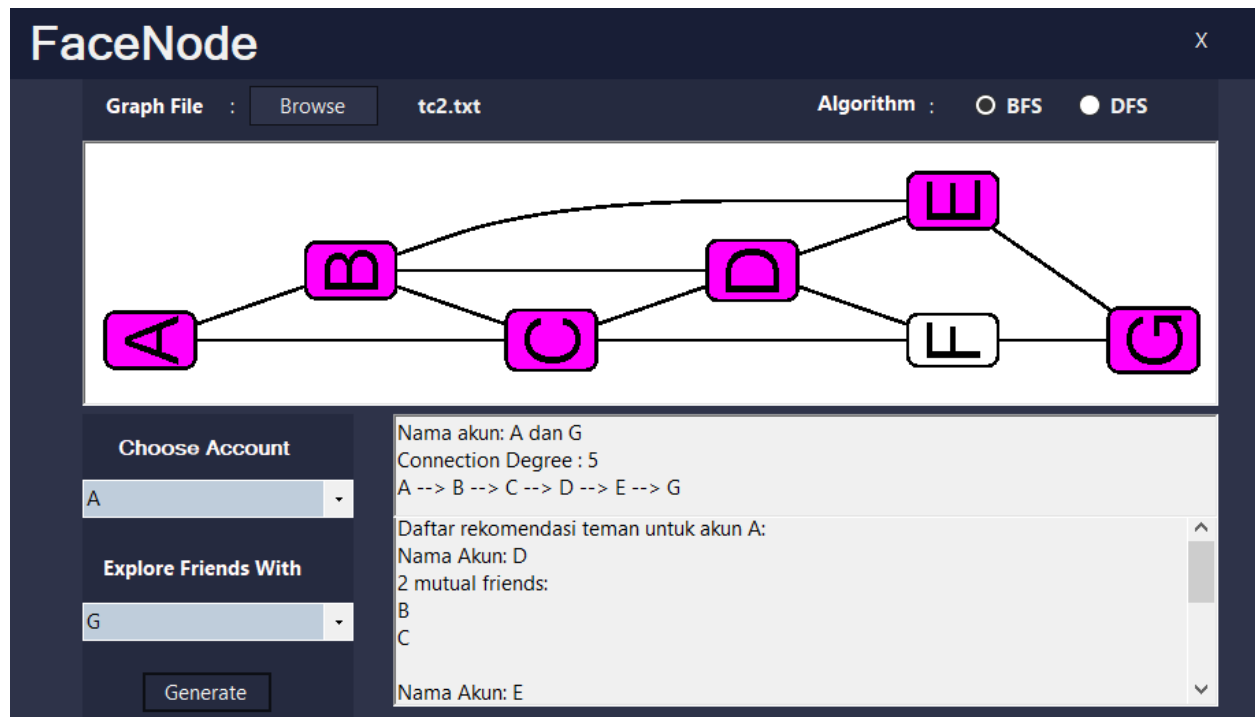
2. Test Case 1 DFS dari A ke H



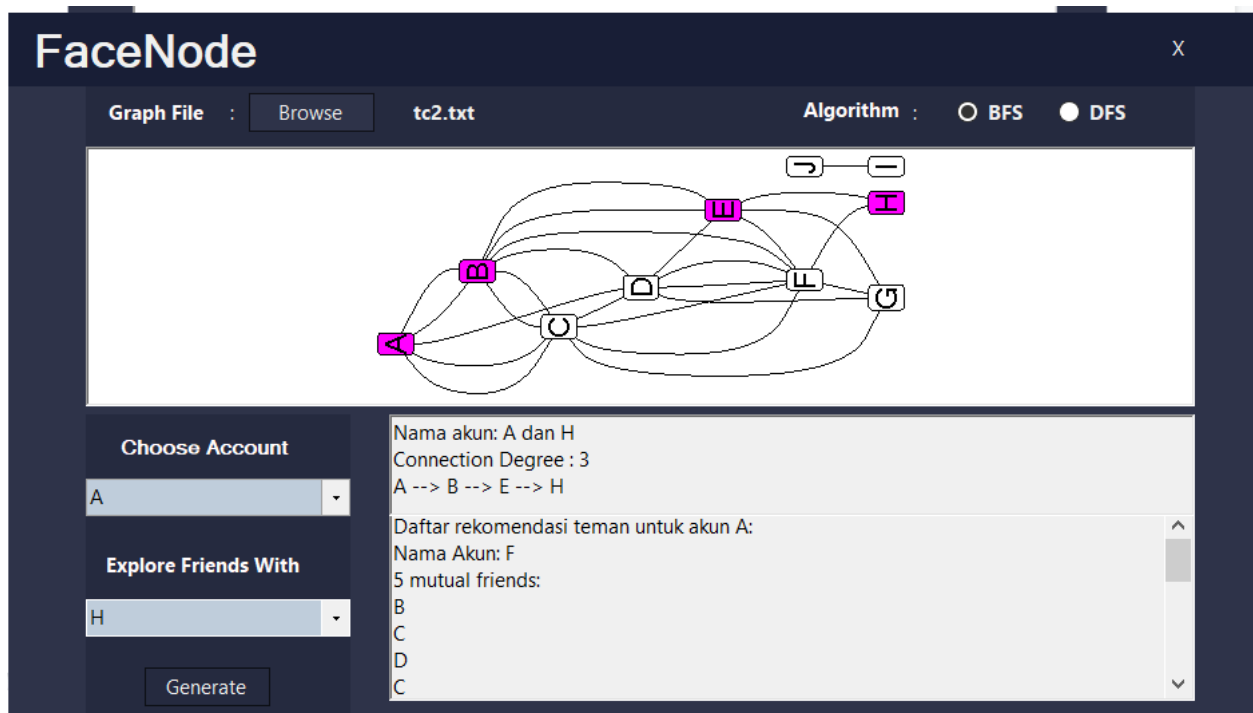
Pada test case ini dapat dilihat bahwa algoritma BFS lebih optimal dalam pencarian node tujuan daripada DFS dalam fitur explore friends. Untuk friend recommendation hasil yang diperoleh tidak berbeda.

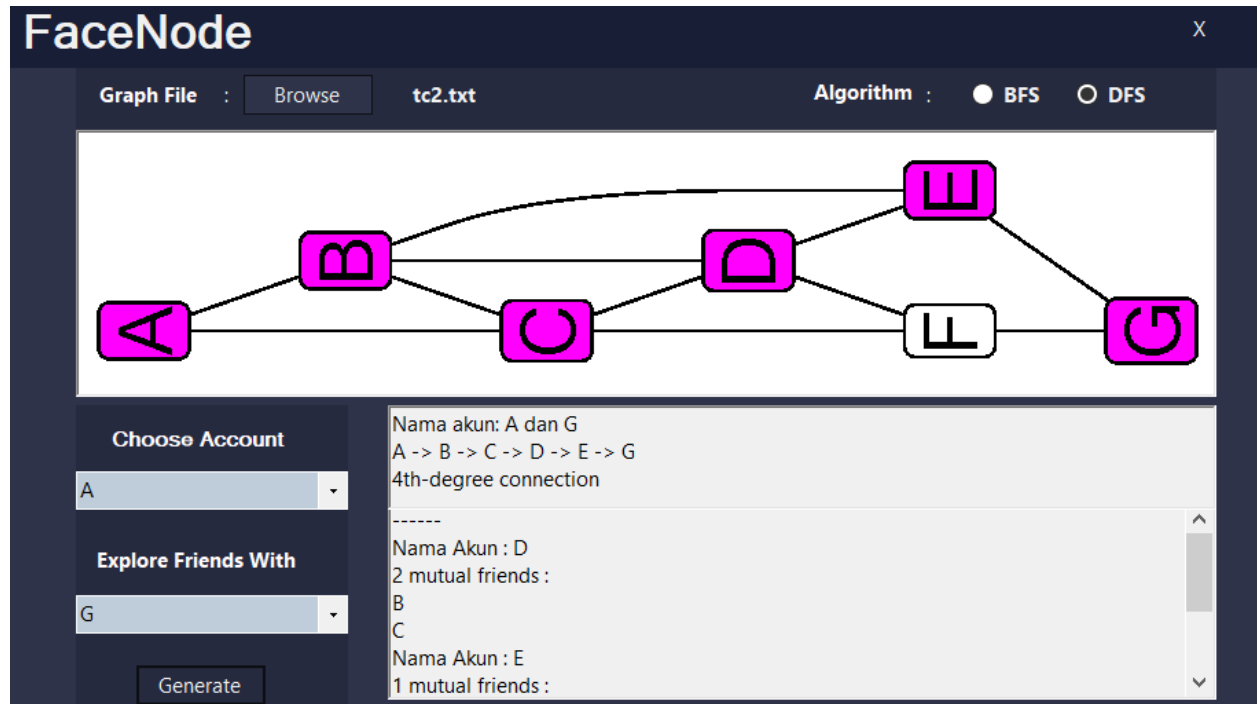
B. Test Case 2

1. BFS dari A ke G



2. DFS dari A ke G

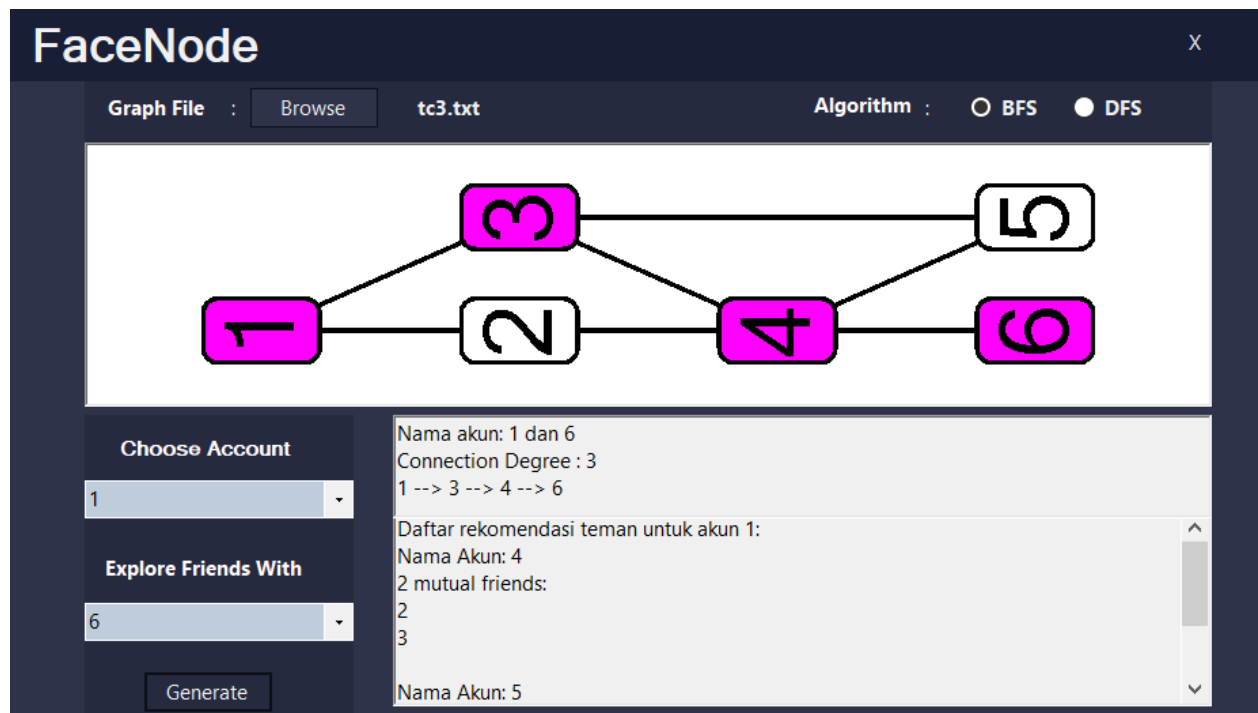




Pada test case ini dapat dilihat bahwa algoritma BFS lebih optimal dalam pencarian node tujuan daripada DFS dalam fitur explore friends. Untuk friend recommendation hasil yang diperoleh tidak berbeda.

C. Test Case 3

1. BFS dari 1 ke 6



2. DFS dari 1 ke 6

FaceNode

Graph File : tc3.txt Algorithm : ☒ BFS ☐ DFS

Choose Account: 1

Explore Friends With: 5

Nama akun: 1 dan 5
1 -> 3 -> 4 -> 5
2nd-degree connection

Nama Akun : 4
2 mutual friends :
3
2

Nama Akun : 5
1 mutual friends :

D. Test Case 4

1. BFS dari 1 ke 6

FaceNode

Graph File : tc4.txt Algorithm : ☐ BFS ☒ DFS

Choose Account: 1

Explore Friends With: 6

Nama akun: 1 dan 6
Connection Degree : 3
1 ---> 2 ---> 3 ---> 6

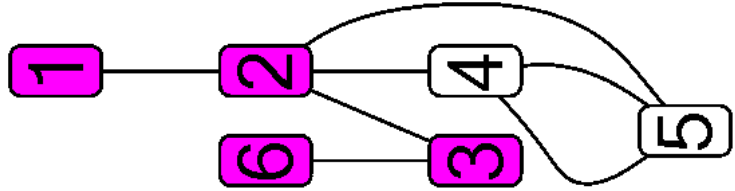
Daftar rekomendasi teman untuk akun 1:
Nama Akun: 3
1 mutual friends:
2

Nama Akun: 5
1 mutual friends:

2. Test Case 4 DFS dari 1 ke 6

FaceNode X

Graph File : tc4.txt Algorithm : ☒ BFS ☐ DFS



```
graph LR; 1 --- 2; 2 --- 3; 2 --- 4; 2 --- 5; 3 --- 6; 4 --- 5; 5 --- 6; style 1 fill:#ff00ff; style 2 fill:#ff00ff; style 3 fill:#ff00ff; style 6 fill:#ff00ff; style 4 fill:#ffff00; style 5 fill:#ffff00;
```

Choose Account
1 ▾

Explore Friends With
6 ▾

Nama akun: 1 dan 6
1 -> 2 -> 3 -> 6
2nd-degree connection

Nama Akun : 3
1 mutual friends :
2

Nama Akun : 5
1 mutual friends :
2

E. Test Case 5

1. BFS dari 1 ke 6

FaceNode

Graph File : tc5.txt Algorithm : ☐ BFS ☒ DFS

Choose Account

v2

Explore Friends With

v6

Nama akun: v2 dan v6
 Connection Degree : 4
 v2 --> v1 --> v4 --> v3 --> v6

Daftar rekomendasi teman untuk akun v2:
 Nama Akun: v4
 1 mutual friends:
 v1

Nama Akun: v3
 1 mutual friends:

2. DFS dari 1 ke 6

FaceNode

Graph File : tc5.txt Algorithm : ☒ BFS ☐ DFS

Choose Account

v2

Explore Friends With

v6

Nama akun: v2 dan v6
 v2 -> v1 -> v4 -> v3 -> v6
 3rd-degree connection

Nama Akun : v4
 1 mutual friends :
 v1

Nama Akun : v3
 1 mutual friends :
 v1

BAB V

Kesimpulan, Saran, Refleksi dan Komentar

5.1 Kesimpulan

Kami telah berhasil membuat sebuah Aplikasi WindowsForm dengan menggunakan bahasa C#. Program membaca file eksternal yang berisi baris pertama menyatakan banyaknya baris selanjutnya, kemudian baris setelahnya adalah keterhubungan antar satu node dengan node yang lain. Setelah dibaca direpresentasikan ke class Graph, class DFS, dan class DFS. Setelah itu dilakukan pencarian menggunakan algoritma BFS atau DFS. Kemudian, hasilnya ditampilkan pada WindowsForm.

5.2 Saran

Penulis juga menyadari jika program yang dibuat masih memiliki banyak kekurangan dan jauh dari sempurna. Oleh karena itu, saran untuk pengembangan aplikasi ini kedepannya yaitu dengan mengoptimasi algoritma BFS dan DFS yang dibangun serta GUI lebih diperindah.

5.3 Refleksi

Dengan diberikannya tugas ini, kami dapat lebih lanjut menjelajahi dan mendalami materi-materi terkait traversal graf terutama BFS dan DFS yang sudah maupun belum dipelajari serta mengeksplor dan membuat program terkait hal tersebut. Selain itu, kami mempelajari dan mencoba C# yang terkesan “baru” bagi kami juga meningkatkan kemampuan memprogram dan bekerja dalam tim.

DAFTAR PUSTAKA

<https://github.com/microsoft/automatic-graph-layout>

<http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag1.pdf>

<http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag2.pdf>