PENGAPLIKASIAN ALGORITMA BFS DAN DFS DALAM FITUR PEOPLE YOU MAY KNOW JEJARING SOSIAL FACEBOOK

LAPORAN TUGAS BESAR 2

Diajukan sebagai salah satu Tugas Besar 2

IF2211 Strategi Algoritma Semester II tahun 2020/2021

Oleh:

Mochammad Fatchur Rochman (13519009)

Yudi Alfayat (13519051)

Mgs. Tabrani (13519122)

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2021

DAFTAR ISI

BAB I

DESKRIPSI TUGAS

Dalam tugas besar ini, Anda akan diminta 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), Anda dapat menelusuri social network pada akun facebook untuk mendapatkan rekomendasi teman seperti pada fitur People You May Know. Selain untuk mendapatkan rekomendasi teman, Anda juga diminta untuk mengembangkan fitur lain agar dua akun yang belum berteman dan tidak memiliki mutual friends sama sekali bisa berkenalan melalui jalur tertentu.

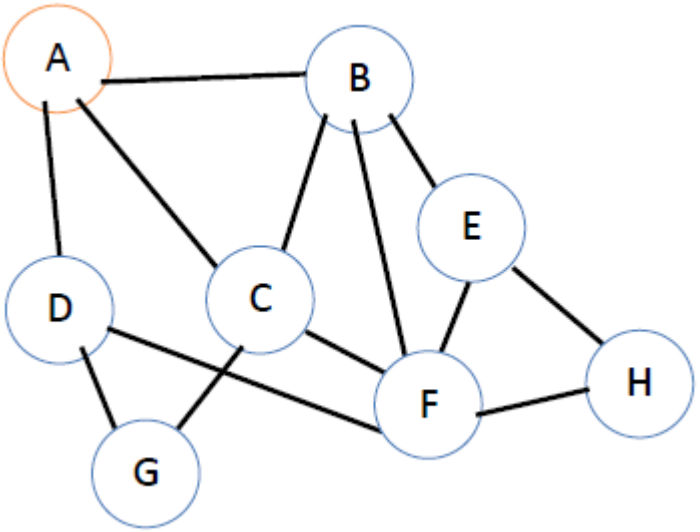
Contoh File Input dan Output

- File Input.txt

```
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 1. Contoh Input Berkas

- Visualisasi Graph File



Gambar 2. Visualisasi Graph Input File Berkas

Fitur Friend Recomendation

Misalnya pengguna ingin mengetahui daftar rekomendasi teman untuk akun A. Maka output yang diharapkan sebagai berikut

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

Nama akun: G
2 mutual friends:
C
D

Nama akun: E
1 mutual friend:
B
```

Gambar 3. Hasil Output Fitur Friend Recommendation yang diharapkan

Fitur Explore Friend

- Misalnya pengguna ingin mengetahui seberapa jauh jarak antara akun A dan H serta bagaimana jalur agar kedua akun bisa terhubung.

```
Nama akun: A dan H
2nd-degree connection
A → B → E → H
```

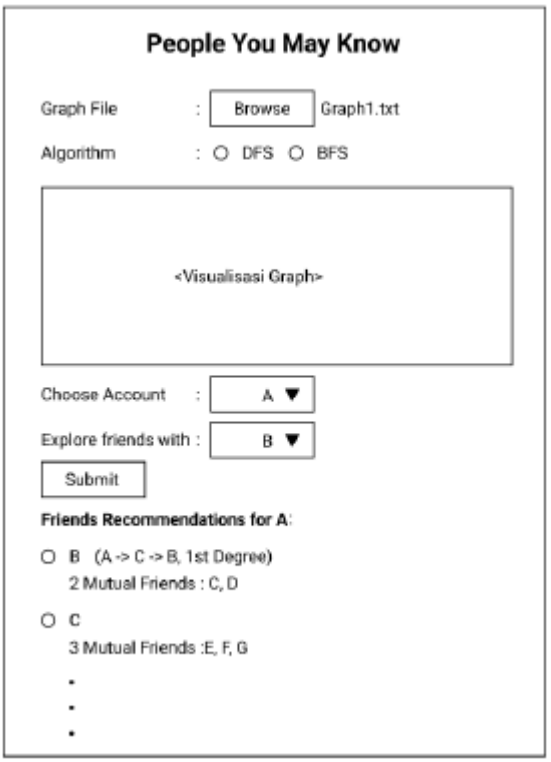
Gambar 4. Hasil Output Fitur Explore Friend yang diharapkan

Perhatikan busur antara akun A dan H, terbentuk salah satu jalur koneksi sebagai berikut: A-B-E-H (ada beberapa jalur lainnya, seperti A-D-F-H, dll, urutan simpul untuk ekspan diprioritaskan berdasarkan abjad). Akun A dan H tidak memiliki mutual friend, tetapi kedua akun merupakan 2nd-degree connection karena di antara A dan H ada akun B dan E yang saling berteman. Sehingga akun H dapat terhubung sebagai teman dengan jalur melalui akun B dan akun E. Jalur koneksi dari A ke H menggunakan BFS digambarkan dalam bentuk graf sebagai berikut.

- Apabila terdapat dua buah akun yang tidak bisa saling terhubung (tidak ada jalur koneksi), maka akan ditampilkan bahwa akun tersebut tidak bisa terhubung melalui jalur koneksi yang sudah dimilikinya sekarang, sehingga orang tersebut memang benar-benar harus memulai koneksi baru dengan orang tersebut

Spesifikasi Program

Aplikasi yang akan dibangun dibuat berbasis GUI. Berikut ini adalah contoh tampilan dari aplikasi GUI yang akan dibangun.



Gambar 5. Tampilan layout dari aplikasi dekstop yang dibangun

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.

Program yang dibuat harus memenuhi spesifikasi wajib sebagai berikut:

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 (lebih jelasnya akan diberikan contoh pada bagian 3).
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.

   Proses visualisasi ini boleh memanfaatkan pustaka atau kakas yang tersedia. Sebagai referensi, salah satu kakas yang tersedia untuk melakukan visualisasi adalah MSAGL (https://github.com/microsoft/automatic-graph-layout). Berikut ini adalah panduan singkat terkait penggunaan MSAGL oleh tim asisten yang dapat diakses pada : https://docs.google.com/document/d/1XhFSpHU028Gaf7YxkmdbluLkQgVl3MY6gt1t-PL30LA/edit?usp=sharing

4) Terdapat dua fitur utama, yaitu:
   a. Fitur friend recommendation
      a) Program menerima sebuah pilihan akun dari user yang hendak dicari rekomendasi temannya. Pemilihan nama akun akan diterima melalui GUI. Cara pemilihan dibebaskan, bisa input dari keyboard atau meng-klik langsung sebuah node dari graf.
      b) Program akan menampilkan daftar rekomendasi teman seperti pada fitur People You May Know facebook berupa

nama akun tersebut secara terurut mulai dari mutual friend terbanyak antar kedua akun beserta daftar nama akun mutual friend.

b. Fitur explore friends
    a) Dua akun yang tidak memiliki mutual friend, masih memiliki peluang untuk berteman jika kedua akun mempunyai common Nth degree connection, yaitu jalur yang menghubungkan kedua akun yang terpisah sejauh N akun (node pada graf).
    b) Program menerima pilihan dua akun yang belum berteman.
    c) Program akan menampilkan nilai N-th degree connection antar kedua akun dan memberikan jalur melalui akun mana saja sampai kedua aku bisa terhubung.
    d) Dari graph yang sudah dibentuk, aplikasi harus dapat menyusun jalur koneksi hasil explore friends antara akun satu dengan akun yang ingin dituju Aplikasi juga harus dapat menunjukkan langkah-langkah pencarian, baik dengan algoritma BFS maupun DFS.
    e) Jika tidak ditemukan jalur koneksi sama sekali antar kedua akun karena graf not fully connected, maka tampilkan informasi bahwa kedua akun tidak dapat terhubung.
    f) Mahasiswa tidak diperkenankan untuk melihat atau menyalin library lain dengan pemanfaatan BFS dan DFS.

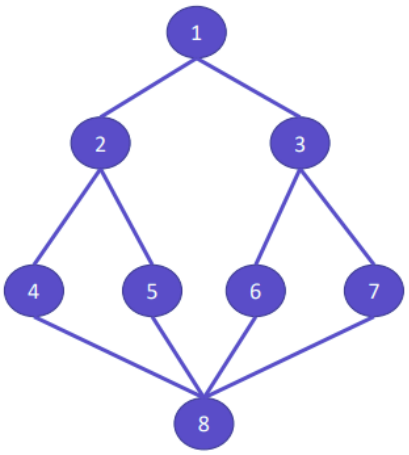
BAB II
LANDASAN TEORI

A. Graph Traversal

Algoritma traversal graf adalah algoritma yang bertujuan mengunjungi simpul suatu graf dengan cara yang sistematik. Algoritma traversal graf ini terdiri dari pencarian melebar (*breadth first search* atau BFS) dan pencarian mendalam (*depth first search* atau DFS) dengan asumsi graf terhubung.

B. *Breadth FIrst Search* (BFS)

Traversal pada *breadth first search* dimulai dari simpul awal atau v. Kemudian kunjungi semua simpul yang bertetangga dengan simpul v terlebih dahulu. Kunjungi simpul yang belum dikunjungi dan bertetangga dengan simpul-simpul yang tadi dikunjungi, demikian seterusnya. Berikut ini ilustrasi pengurutan menggunakan *breadth first search*.



Iterasi	V	Q	dikunjungi							
			1	2	3	4	5	6	7	8
Inisialisasi	1	{1}	T	F	F	F	F	F	F	F
Iterasi 1	1	{2,3}	T	T	T	F	F	F	F	F
Iterasi 2	2	{3,4,5}	T	T	T	T	T	F	F	F
Iterasi 3	3	{4,5,6,7}	T	T	T	T	T	T	T	F
Iterasi 4	4	{5,6,7,8}	T	T	T	T	T	T	T	T
Iterasi 5	5	{6,7,8}	T	T	T	T	T	T	T	T
Iterasi 6	6	{7,8}	T	T	T	T	T	T	T	T
Iterasi 7	7	{8}	T	T	T	T	T	T	T	T
Iterasi 8	8	{}	T	T	T	T	T	T	T	T

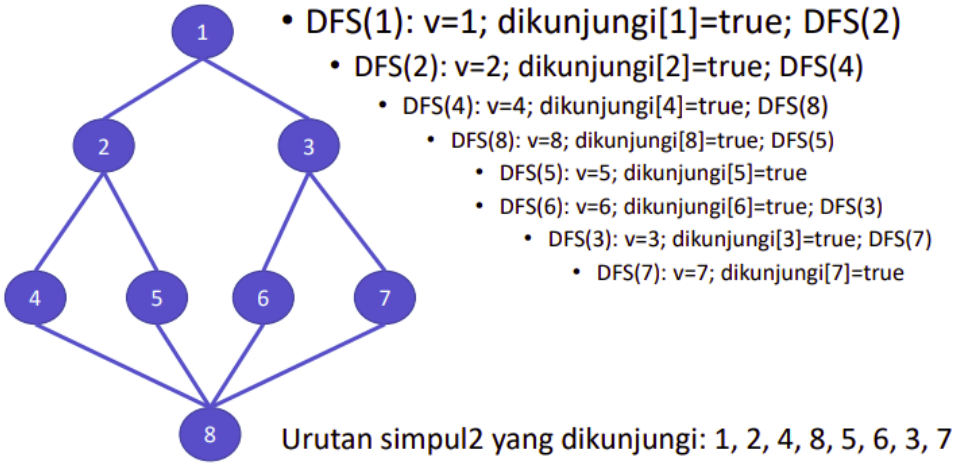
Urutan simpul2 yang dikunjungi: 1, 2, 3, 4, 5, 6, 7, 8

Gambar 1.1 Ilustrasi *breadth first seacrh*

C. *Depth First Search* (DFS)

Traversal pada *breadth first search* dimulai dari simpul awal atau v. Kunjungi simpul v. Kemudian kunjungi simpul w yang bertetangga dengan simpul v. Ualngi DFS mulai dari simpul w. Ketika mencapai simpul u sedemikian sehngga semua simpul yang bertetangga dengannya telah dikunjungi, pencarian dirunut-balik (*backtrack*) ke simpul terakhir yang dikunjungi sebelumnya dan

mempunyai simpul w yang belun dikunjungi. Pencarian berakhir bila tidak ada lagi simpul yang belum dikunjungi yang dapat dicapai dari simpul yang telah dikunjungi. Berikut ini ilustrasi pengurutan menggunakan *depth first search*.



Gambar 1.2 Ilustrasi *depth first search*

D. C# *Desktop Application Development*

C# *desktop application development* adalah pengembangan aplikasi *desktop* dengan menggunakan bahasa C#. C# adalah bahasa turunan C yang dikembangkan oleh Microsoft. C# biasa digunakan dalam pengembangan aplikasi berbasis *desktop*. Pengembangan aplikasi *desktop* dengan C# biasa dilakukan menggunakan *Integrated Development Environment* tertentu, yaitu Visual Studio. Visual Studio juga merupakan produk buatan Microsoft. Visual Studio dapat dijalankan di sistem operasi Windows dan sistem operasi MacOS.

BAB III
ANALISIS PEMECAHAN MASALAH

A. Langkah - Langkah Pemecahan Masalah

Program ini dibuat dengan beberapa langkah. Pertama, saat *file* .txt dimasukkan ke program, program akan membaca *file* tersebut, dan mengonversinya menjadi graf, sesuai dengan *edge* yang ada di *file*. Setelah program mengubah *file* menjadi graf, akan muncul simpul-simpul dari graf tersebut. Kemudian pengguna dapat menjalankan fitur *explore friend, friends recommendation* sesuai dengan algoritma yang diinginkan, yaitu *breadth first search* dan *depth first search*.

B. Proses Mapping Persoalan

1. Explore Friends Breadth First Search

Pada algoritma *explore friends* dengan pendekatan BFS, simpul yang menjadi asal akan mencari simpul yang menjadi tujuan. Simpul asal akan maju secara rekursif ke simpul yang bertetangga denganya. Simpul-simpul yang bertetangga ini akan dianggap tidak bisa dikunjungi saat pengunjungan simpul selanjutnya jika bukan simpul-simpul tersebut yang dikunjungi. Jika ketemu simpul tujuan, pencarian akan berhenti. Namun jika tidak, pencarian akan terus dilakukan sampai suatu simpul tidak bisa mengunjungi simpul yang bertetangga dengannya, karena simpul-simpul tetangga tersebut sudah dikunjungi. Namun, pengunjungan simpul dapat mundur jika memang tidak ada simpul setelahnya.

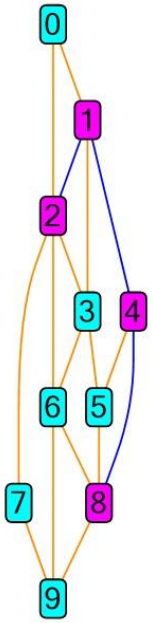
2. Explore Friends Depth First Search

Pada algoritma *explore friends* dengan pendekatan DFS, simpul yang menjadi asal akan mencari simpul yang menjadi tujuan. Simpul asal akan maju secara rekursif ke simpul yang bertetangga denganya. Jika ketemu simpul tujuan, pencarian akan berhenti. Namun jika tidak, pencarian akan terus dilakukan sampai suatu simpul tidak bisa mengunjungi simpul yang bertetangga dengannya, karena simpul-simpul tetangga tersebut sudah dikunjungi. Namun, pengunjungan simpul dapat mundur jika memang tidak ada simpul setelahnya.

3. *Friend Recommendation*

Pada *friend recommendation*, yang dicari adalah *friend recommendation* dari simpul awal yang dipilih di *explore friend*. Algoritma ini mulanya akan mencari simpul-simpul tetangga dari simpul awal. Kemudian, simpul-simpul tetangga tersebut akan mencari simpul-simpul tetangga mereka sebagai *friend recommendation* untuk simpul awal. Kemudian simpul-simpul yang dijadikan *friend recommendation* tadi akan diurutkan berdasarkan *mutual friend* dengan simpul awal atau jumlah simpul tetangga yang sama dengan simpul awal.

C. Contoh Ilustrasi



Gambar 3.1 Graf ilustrasi

Pada pencarian *explore friend* dengan pendekatan *breadth first search*, urutan pencarian dari 2 ke 8, akan menjadi 2 -> 1 -> 4 -> 8. Sedangkan pada pendekatan *depth first search*, pencarian dari 2 ke 8 akan menjadi 2 -> 1 -> 3 -> 5 -> 4 -> 8. Kemudian *friends recommendation* dari simpul 2 akan ditampilkan, yaitu simpul 9 dengan 2 *mutual friends* (simpul 6 dan 7), simpul 5 dengan 1 *mutual friend* (simpul 3), simpul 8 dengan 1 *mutual friend* (simpul 6), dan simpul 4 dengan 1 *mutual friend* (simpul 1).

BAB IV

IMPLEMENTASI DAN PENGUJIAN

A. Implementasi Program

1. BFS *Explore Friend*

```
function bfsExploreFriend(Graph g, int nodeForm, int nodeTo, int[] dikunjungi,
int[] dibfs, bool timeToReset, int[] hasil) : boolean
KAMUS DATA
        boolean isExist
ALGORITMA
        if (dikunjungi[nodeForm] == 0) then
                dikunjungi[nodeForm] == 1
                hasil.addElement(nodeForm)
        {reset node yang terhubung ke nodeFrome, yang menandakan node-node
tersebut dapat dikunjungi dari nodeForm}
        if (timeToReset) then
                i traversal[0..g.getNumOfConnectedNode(nodeForm)]
                        dibfs[g.getConnectedNode(nodeForm, i)] == 0
        {node yang dicari sudah ditemukan}
        if (nodeForm == nodeTo) then
                return true
        else
                {Node tidak mempunyai cabang}
                if (g.getNumOfConnectedNode(nodeForm) == 0)
                        return false
                else
                        isExist = false
                        i = 0
                        {mengambil semua node yang terhubung ke nodeForm}
                        connectedNode = g.getConnectedNode(nodeForm)
                        {Mencari node yang bisa dikunjungi dari nodeForm, node
yang bisa dikunjungi adalah  node yang seblumnya belum pernah dikunjungi dan
atau bukan tetangga dari nodeForm}
```

```
                    i traversal[0 .. connectedNode.Length]
                        if (dikunjungi[connectedNode[i]] == 0 and
dibfs[connectedNode[i]] == 0) then
                            j traversal[0 .. connectedNode.Length]
                                if (i != j) then
                                    dibfs[connectedNode[j]] = 1
                            isExist = true
                            break
                    if (isExist) then
                        return bfsExploreFriend(g, connectedNode[i],
nodeTo, dikunjungi, dibfs, false, hasil)
                    else
                        {jika tidak ada lagi yang bisa dikunjungi dari
nodeForm, maka dilakukan backtracking}
                        hasil.removeLastElement()
                        {ketika posisi sudah di titik asal, dan tidak ada lagi
yang bisa dikunjungi, berarti tidak ada jalur yang bisa dilalui untuk menuju tujuan
nodeTo}
                        if (hasil.Length == 0) then
                            return false
                        else
                            return bfsExploreFriend(g,
hasil.getLastElement, nodeTo, dikunjungi, dibfs, true, hasil)
```

2. DFS *Explore Friend*

```
funtion dfsExploreFriend(Graph g, int nodeForm, int nodeTo, int[] dikunjungi,
int[] hasil) : boolean
KAMUS
    boolean isExist
ALGORITMA
    {Check apakah nodeForm sudah pernah dikunjungi}
    if (dikunjungi[nodeForm] == 0) then
```

```
            dikunjungi[nodeForm] == 1
            hasil.addElement(nodeForm)
        {jika node yang dituju sudah ditemukan}
        if (nodeForm == nodeTo) then
            return true
        else
            {jika tidak ada node yang terhubung ke nodeForm}
            if (g.getNumOfConnectedNode(nodeForm) == 0) then
                return false
            else
                isExist = false;
                i = 0
                {mengambil node-node yang terhubung ke nodeForm}
                connectedNode = g.getConnectedNode(nodeForm);

                {mencari node yang bisa belum pernah dikunjungi dari
nodeForm, node diurutnkan berdasarkan abjad}
                i traversal[0 .. connectedNode.Length]
                    if (dikunjungi[connectedNode[i]] == 0) then
                        isExist = true
                        break
                if (isExist) then
                    return dfsExploreFriend(g, connectedNode[i],
nodeTo, dikunjungi, hasil)
                else
                    {Melakukan backtracking}
                    hasil.removeLastElement()
                    {jika nodeForm sudah kembali ke titik asal dan
semua node sudah dikunjungi, maka jalur dari nodeForm ke nodeTo tidak
ditemukan}
                    if (hasil.Length == 0) then
                        return false
```

else

return dfsExploreFriend(g, hasil.getLastElement(), nodeTo, dikunjungi, hasil)

3. Friend Recommendation

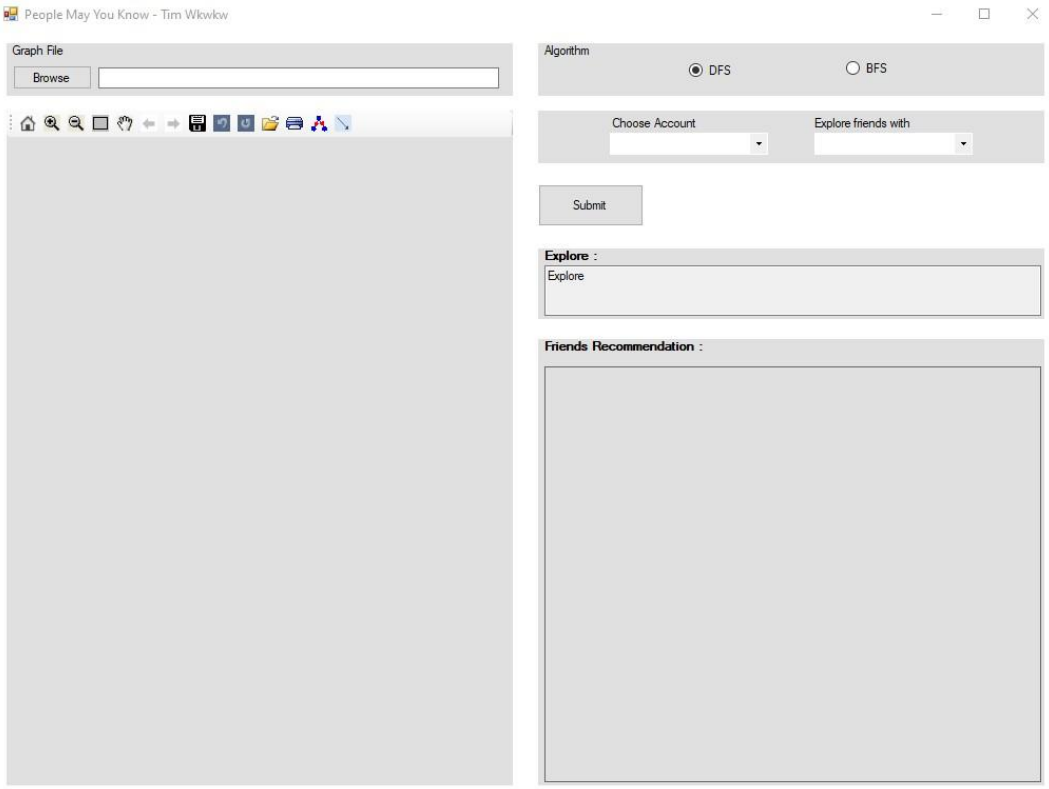
```
procedure dfsRecommendation (Graph g, int search)
KAMUS
ALGORITMA
    search1 = search
    {mengambil semua node yang terhubung ke node yang dicari,  artinya node-node ini berteman dengan node search di jejaring sosial}
    connectedNode = g.getConnectedNode(search1)
    i traversal[0..connectedNode.Length]
        search2 = connectedNode[i]
```

B. Struktur Data dan Spesifkasi Program

Struktur data yang digunakan untuk merepresentasikan graf berupa kelas, yang memiliki atribut *numOfNode* yang bertipe *integer*, *node* yang bertipe *list of string*, *numOfConnectedNode* yang bertipe *list of integer*, dan *connectedNode* yang bertipe *list of list of integer*.

C. Tata Cara Penggunaan Program

Program ini memilik fitur menampilkan *node* dari *file input* yang berekstensi .txt, memiliki fitur *explore friend* yang diimplementasikan melalui algoritma DFS maupun BFS, fitur *friend recommendation*. Berikut ini adalah *user interface* dari program.



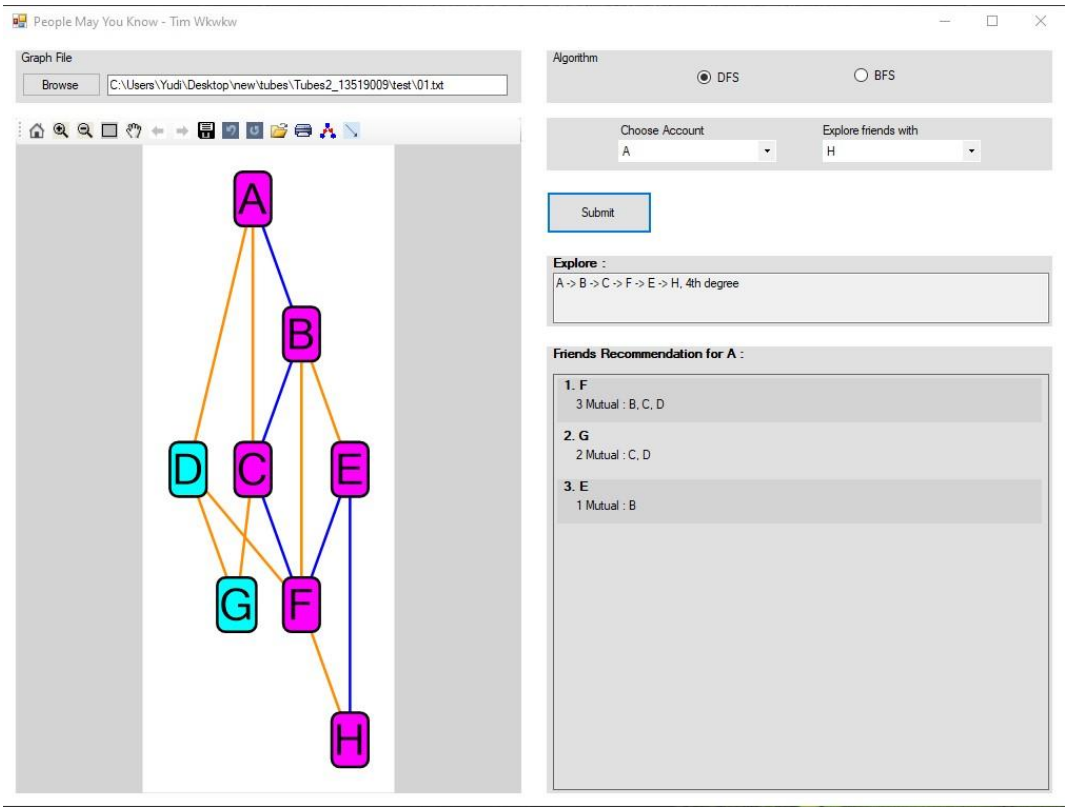
Gambar 4.1 *Screenshot User Interface*

D. Hasil Pengujian

1. *File* 01.txt

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

Berikut ini hasil program yang muncul dengan masukan 01.txt.



Gambar 4.2 Hasil pengujian *file* 01.txt

Pada gambar di atas, dilakukan *explore friend* dari A ke H dengan algoritma DFS menghasilkan A->B->C->F->E->H. *Friends recommendation* untuk A juga ditampilkan di bawahnya.

2. *File* 02.txt

15
tabrani,fatur
tabrani,yudi
tabrani,pak munir
fatur,yudi
fatur,pak rinaldi
fatur,buk fariska
yudi,pak infal
pak rinaldi,pak infal
pak rinaldi,pak munir
pak infal,pak ahmad

pak ahmad,pak rila

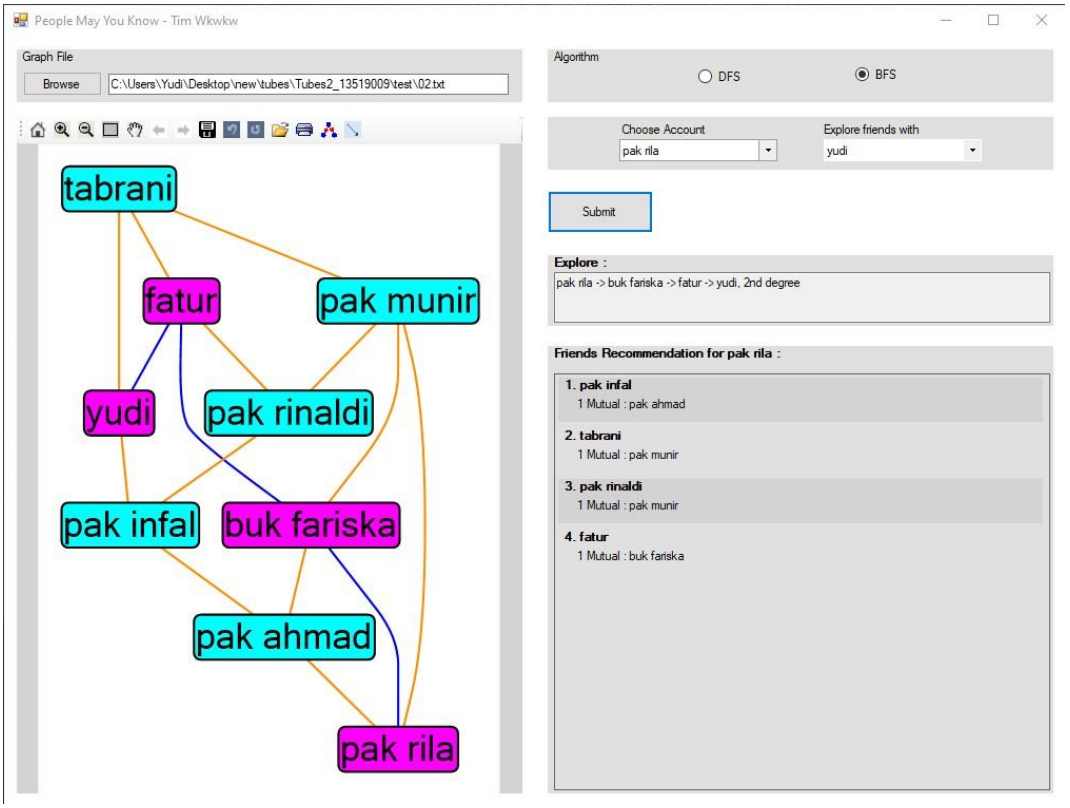
pak ahmad,buk fariska

pak rila,pak munir

pak rila,buk fariska

buk fariska,pak munir

Berikut ini hasil program yang muncul dengan masukan 02.txt.



Gambar 4.3 Hasil pengujian *file* 02.txt

Pada gambar di atas, dilakukan *explore friend* dari pak rila ke yudi dengan algoritma BFS menghasilkan pak rila -> buk fariska -> fathur -> yudi. *Friends recommendation* untuk pak rila juga ditampilkan di bawahnya.

3. *File* 03.txt

8

MA1101,MA1201

MA1101,IF2130

MA1101,IF2123

MA1101,IF2110

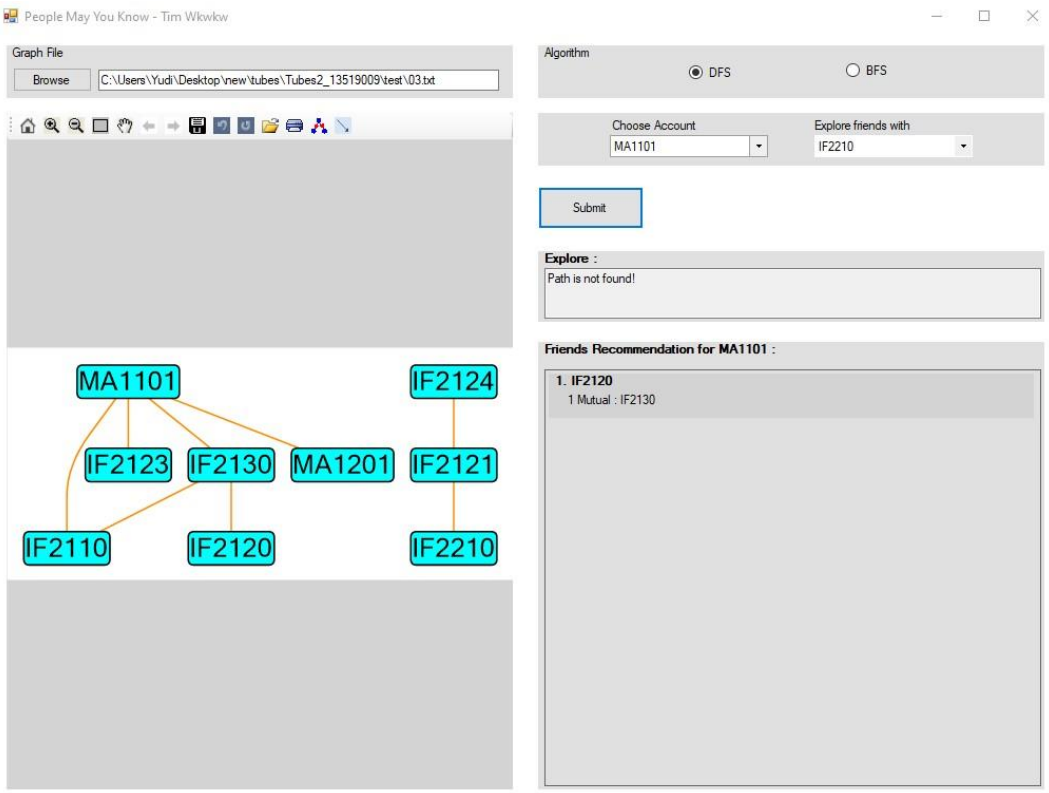
IF2130,IF2110

IF2130,IF2120

IF2124,IF2121

IF2121,IF2210

Berikut ini hasil program yang muncul dengan masukan 03.txt.



Gambar 4.4 Hasil pengujian *file* 03.txt

Pada gambar di atas, dilakukan *explore friend* dari MA1101 ke IF2210 dengan algoritma DFS tidak mendapatkan jalur apapun. *Friends recommendation* untuk MA1101 juga ditampilkan di bawahnya.

4. *File* 04.txt

10

1,3

1,2

3,6

3,7

2,4

2,5

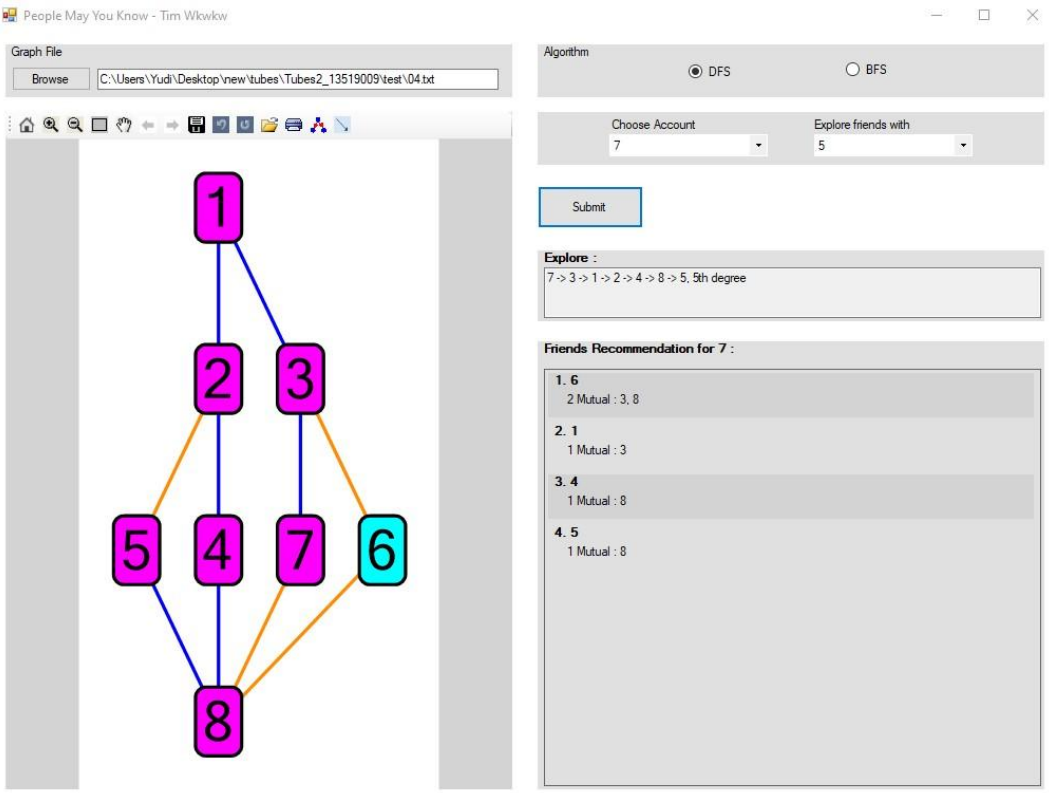
8,4

8,5

8,6

8,7

Berikut ini hasil program yang muncul dengan masukan 04.txt.



Gambar 4.5 Hasil pengujian *file* 04.txt

Pada gambar di atas, dilakukan *explore friend* dari 7 ke 5 dengan algoritma DFS menghasilkan 7 -> 3 -> 1 -> 2 -> 4 -> 8 -> 5. *Friends recommendation* untuk 7 juga ditampilkan di bawahnya.

5. *File* 05.txt

16

0,2

0,4

0,5

1,4

1,5

2,3

2,4

4,5

6,7

6,5

3,8

9,10

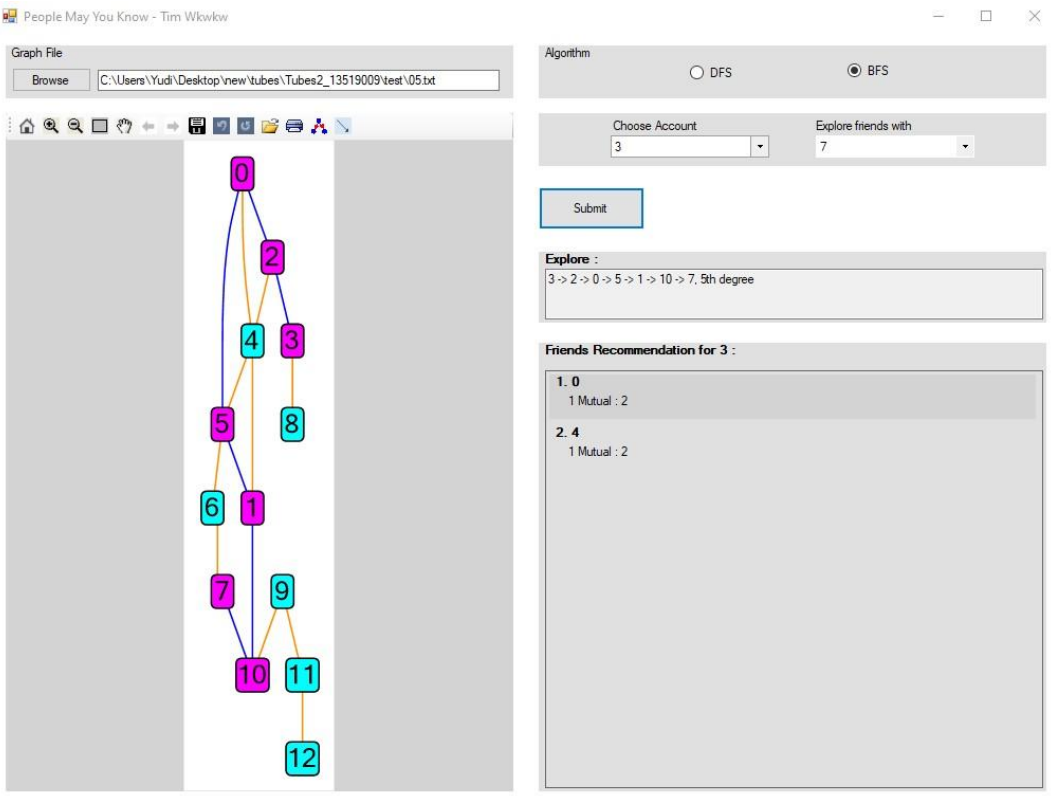
11,9

11,12

10,1

7,10

Berikut ini hasil program yang muncul dengan masukan 05.txt.



Gambar 4.6 Hasil pengujian *file* 05.txt

Pada gambar di atas, dilakukan *explore friend* dari 3 ke 7 dengan algoritma BFS menghasilkan 3 -> 2 -> 0 -> 5 -> 1 -> 10 -> 7. *Friends recommendation* untuk 3 juga ditampilkan di bawahnya.

6. *File* 06.txt

101

0,1

0,3

0,5

1,7

1,8

1,9
1,10
3,11
3,12
5,13
5,14
5,15
5,16
7,17
8,20
8,21
9,28
9,29
9,30
9,31
10,40
10,41
10,42
11,50
11,51
11,52
12,53
12,54
13,70
13,71
14,72
14,73
14,74
15,75
15,76
15,77
15,78
16,79
16,80
17,18
17,19
20,22
20,23
20,24

21,25
21,26
21,27
28,32
29,33
29,34
29,35
30,36
30,37
31,38
31,39
40,43
40,44
41,45
41,46
41,47
42,48
42,49
50,55
50,56
51,57
51,58
51,59
52,60
52,61
52,62
53,63
53,64
53,65
54,66
54,67
54,68
54,69
70,81
70,82
70,83
71,84
71,85
71,86

72,87

72,88

72,89

73,91

74,90

74,92

75,93

75,94

75,95

76,96

77,99

78,97

78,98

79,100

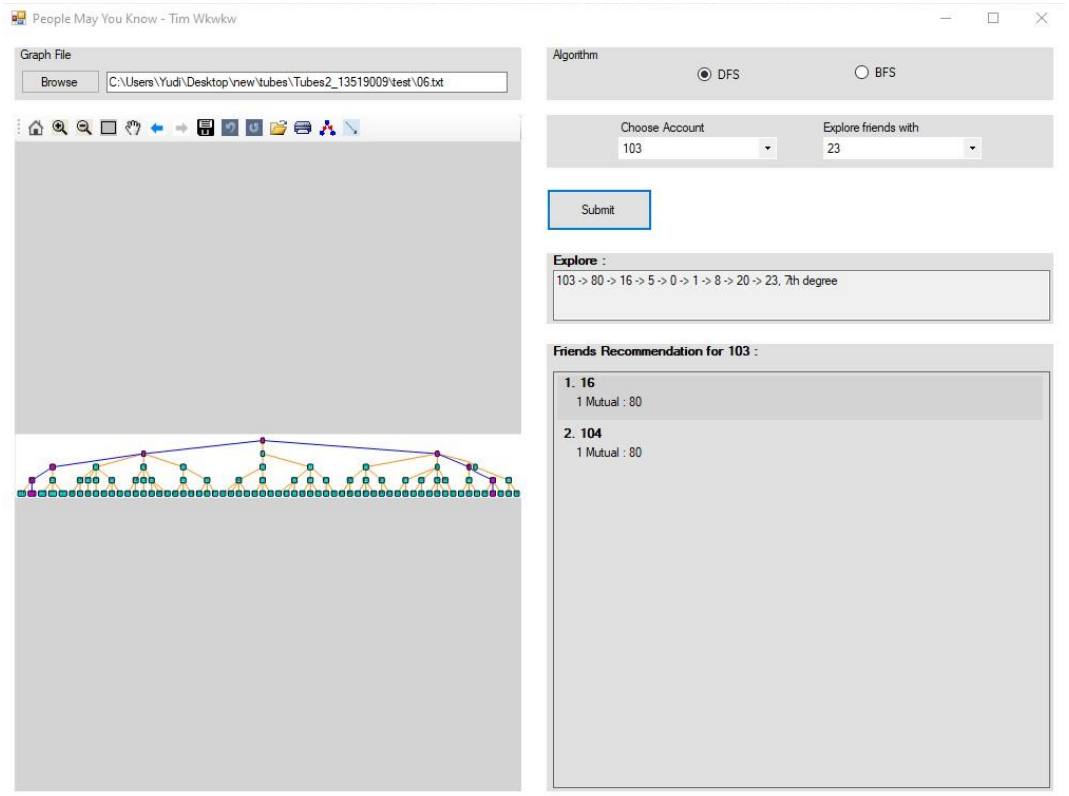
79,101

79,102

80,103

80,104

Berikut ini hasil program yang muncul dengan masukan 06.txt.



Gambar 4.7 Hasil pengujian *file* 06.txt

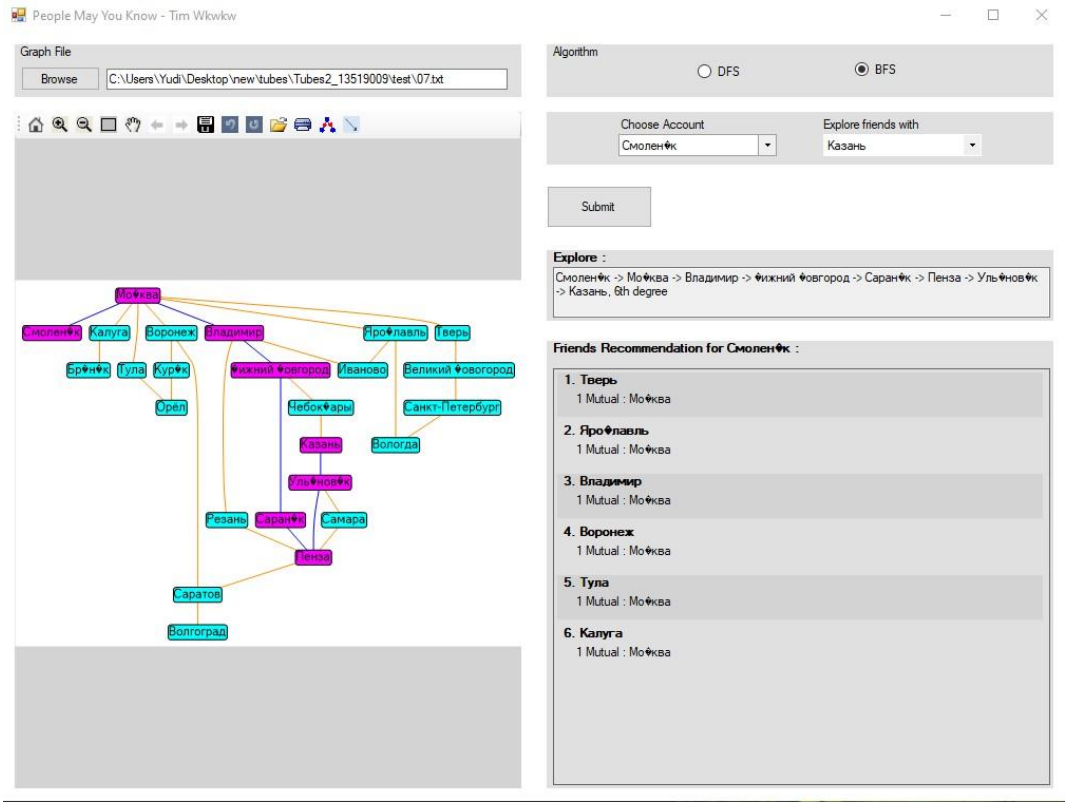
Pada gambar di atas, dilakukan *explore friend* dari 103 ke 23 dengan algoritma DFS menghasilkan 103 -> 80 -> 16 -> 5 -> 0 -> 1 -> 8 -> 20 -> 23. *Friends recommendation* untuk 103 juga ditampilkan di bawahnya.

7. *File* 07.txt

31

Мо�ква,Тверь
Мо�ква,Яро�лавль
Мо�ква,Владимир
Мо�ква,Воронеж
Мо�ква,Тула
Мо�ква,Калуга
Мо�ква,Смолен�к
Тверь,Великий �овогород
Великий �овогород,Санкт-Петербург
Санкт-Петербург,Вологда
Яро�лавль,Вологда
Яро�лавль,Иваново
Владимир,Иваново
Владимир,�ижний �овгород
Владимир,Резань
�ижний �овгород,Чебок�ары
�ижний �овгород,Саран�к
Чебок�ары,Казань
Казань,Уль�нов�к
Уль�нов�к,Самара
Уль�нов�к,Пенза
Самара,Пенза
Пенза,Саран�к
Пенза,Саратов
Пенза,Резань
Саратов,Волгоград
Саратов,Воронеж
Воронеж,Кур�к
Кур�к,Орёл
Орёл,Тула
Калуга,Бр�н�к

Berikut ini hasil program yang muncul dengan masukan 07.txt.



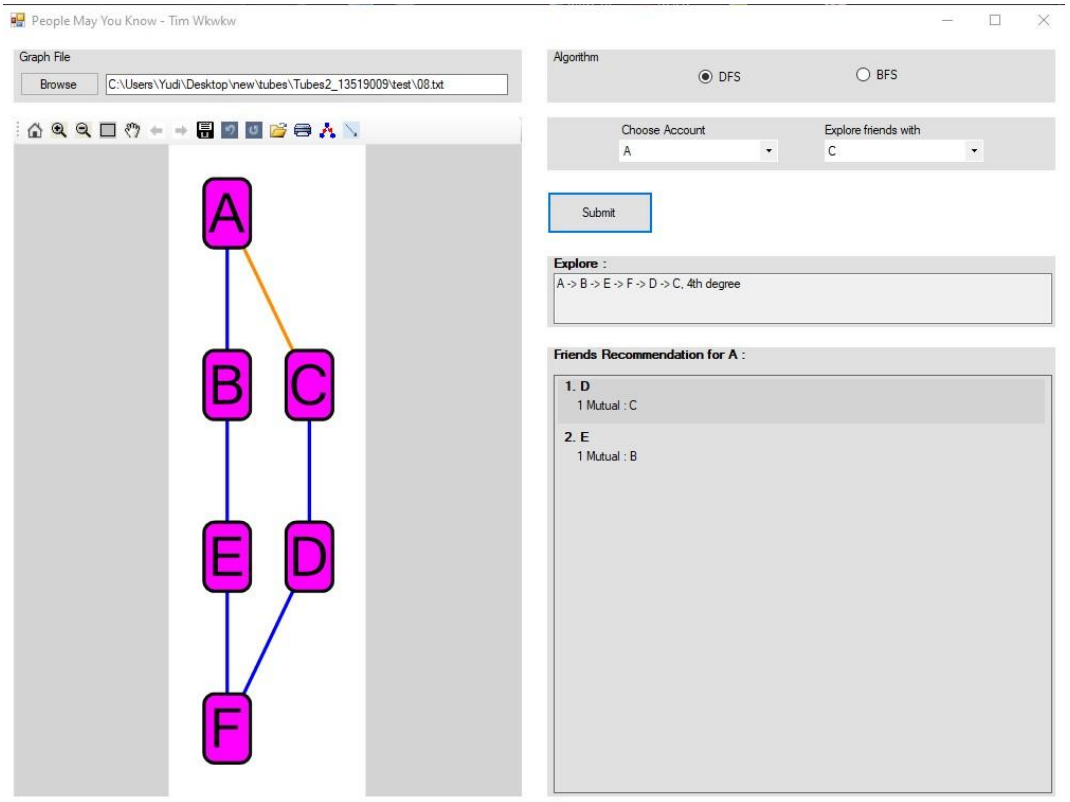
Gambar 4.8 Hasil pengujian *file* 07.txt

Pada gambar di atas, dilakukan *explore friend* dengan algoritma BFS. *Friends recommendation* juga ditampilkan di bawahnya.

8. *File* 08.txt

```
6
A,C
A,B
B,E
C,D
D,F
E,F
```

Berikut ini hasil program yang muncul dengan masukan 08.txt.



Gambar 4.9 Hasil pengujian *file* 08.txt

Pada gambar di atas, dilakukan *explore friend* dari A ke C dengan algoritma DFS menghasilkan A -> B -> E -> F -> D -> C. *Friends recommendation* untuk A juga ditampilkan di bawahnya.

9. *File* 09.txt

17
0,1
0,2
1,2
1,3
1,4
2,3
2,6
2,7
3,5
3,6
4,5

4,8

5,8

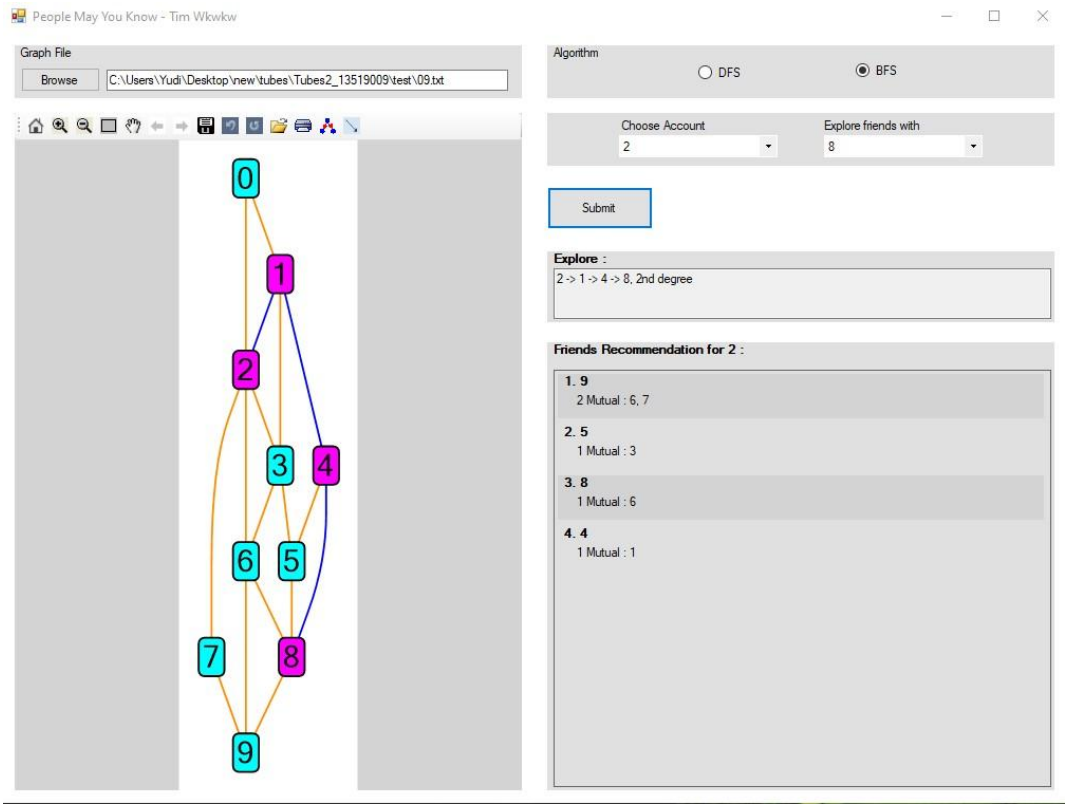
6,8

6,9

7,9

8,9

Berikut ini hasil program yang muncul dengan masukan 09.txt.



Gambar 4.10 Hasil pengujian *file* 09.txt

Pada gambar di atas, dilakukan *explore friend* dari 2 ke 8 dengan algoritma BFS menghasilkan 2 -> 1 -> 4 -> 8. *Friends recommendation* untuk 2 juga ditampilkan di bawahnya.

10. File 10.txt

15

A,B

A,C

A,D

A,E

A,F

B,C

B,D

B,E

B,F

C,D

C,E

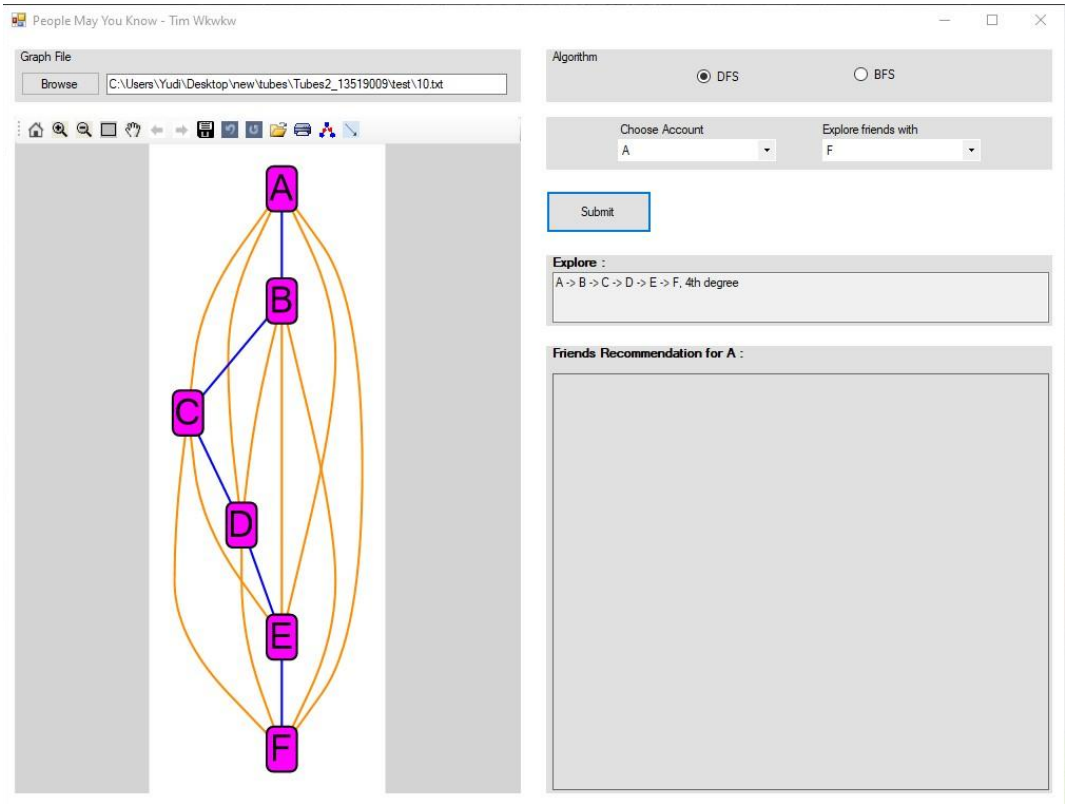
C,F

D,E

D,F

E,F

Berikut ini hasil program yang muncul dengan masukan 10.txt.



Gambar 4.11 Hasil pengujian *file 1*0.txt

Pada gambar di atas, dilakukan *explore friend* dari A ke F dengan algoritma DFS menghasilkan A -> B -> C -> D -> E -> F. *Friends recommendation* untuk A tidak ditampilkan karena A tidak memiliki *friends recommendation*.

BAB V
KESIMPULAN, SARAN, REFLEKSI, DAN KOMENTAR

A. Kesimpulan

Setelah menyelesaikan tugas besar 2 ini, didapatkan bahwa algoritma *breadth first search* dan *depth first search* dapat bisa diterapkan untuk menyelesaikan permasalahan dalam representasi graf. Di beberapa kasus BFS melakukan pencarian jalur dengan lebih sedikit simpul yang dikunjungi dibandingkan DFS. Namun, DFS lebih mudah di-*tracking* secara langsung.

B. Saran

Pada pengerjaan tugas besar ini, masih banyak kekurangannya. Tugas besar 2 ini diharapkan dapat menjadi referensi mahasiswa dalam pengerjaan tugas atau saat sudah terjun dalam pengembangan aplikasi *desktop*

C. Refleksi

Pada proses pengerjaan tugas ini, kelompok mengalami beberapa kendala teknis dan beberapa kendala non-teknis. Namun kendala sudah dapat terselesaikan. Kendala teknis berupa, mulanya program masih sering *error*. Sedangkan kendala non-teknis berupa *time management* kelompok yang kurang baik, sehingga menimbulkan ke-*chaos*-an pada saat pengerjaan yang ditambah dengan menumpuknya tugas lain. Kelompok kami berharap, kami mampu memperbaiki *skill time management.*

D. Komentar

Menurut Tim wkwk tugas besar 2 yang diberikan ini sangat menarik. Tugas besar ini dapat menambah wawasan mahasiswa dalam pengembangan aplikasi berbasis *desktop*. Namun, beberapa mahasiswa yang tidak menggunakan sistem operasi Windows atau MacOS kesulitan dalam melakukan *debugging* terhadap aplikasi, sehingga cukup merepotkan saat pengembangan aplikasi. Secara keseluruhan tugas besar ini sangat menantang dan meningkat pola pikir serta kreativitas.

DAFTAR PUSTAKA

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Tugas-Besar-2-IF2211-Strategi-Algoritma-2021.pdf (diakses pada 25 Maret 2021).

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag1.pdf (diakses pada 25 Maret 2021).

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag2.pdf (diakses pada 25 Maret 2021).

https://docs.microsoft.com/en-us/dotnet/csharp/ (dikases pada 26 Maret 2021).