LAPORAN TUGAS BESAR 2

IF2211 – STRATEGI ALGORITME

SEMESTER II - TAHUN 2020/2021

Pengaplikasian Algoritma BFS dan DFS dalam Fitur *People You May Know* Jejaring Sosial Facebook

1

2

3

Daftar Anggota Kelompok:

1. Leonardus Brandon Luwianto (13519102)
2. Nathaniel Jason (13519108)
3. Rizky Anggita Syarbaini Siregar (13519132)

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

2021

DAFTAR ISI

BAB I

DESKRIPSI TUGAS

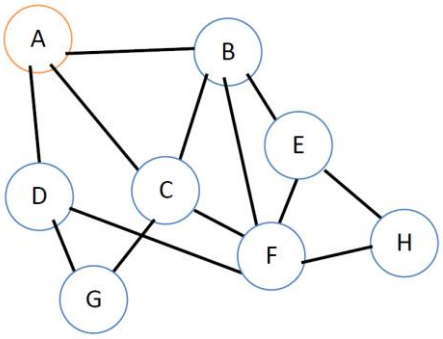
Dalam tugas besar ini, Anda akan diminta untuk membangun sebuah aplikasi GUI sederhana yang dapat memodelkan beberapa fitur dari *People You May Know* dalam jejaring sosial media (*Social Network*). Dengan memanfaatkan algoritma *Breadth First Search* (BFS) dan Depth First Search (DFS), Anda dapat menelusuri *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 Input dan Output Program

Contoh input berkas file eksternal:

```
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
```

Visualisasi graf pertemanan yang dihasilkan dari file eksternal:



Gambar 1. Contoh visualisasi graf pertemanan dari file eksternal

Untuk **fitur friend recommendation**, 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 2. Hasil output yang diharapkan untuk rekomendasi akun A

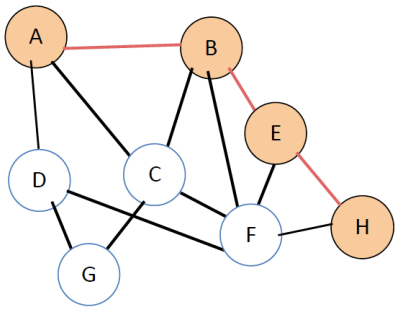
Untuk **fitur explore friends**, misalnya pengguna ingin mengetahui seberapa jauh jarak antara akun A dan H serta bagaimana jalur agar kedua akun bisa terhubung. Berikut output graf dengan penelusuran BFS yang dihasilkan. Berikut output yang diharapkan untuk penelusuran menggunakan BFS.

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

A → B → E → H
```

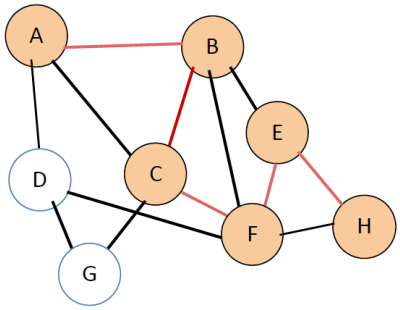
Gambar 3. Hasil output akun N^{th} *degree connection* akun A dan H menggunakan DFS

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.



Gambar 4. Hasil visualisasi jalur koneksi menggunakan BFS

Sedangkan untuk penggunaan algoritma DFS, diperoleh jalur lainnya, yaitu A-B-C-F-E-H yang digambarkan dalam bentuk graf sebagai berikut.



Gambar 5. Hasil visualisasi jalur koneksi menggunakan DFS

Pada fitur explore friends, 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.

Misalnya terdapat dua orang baru, yaitu J dan I yang hanya terhubung antara J-I. Maka jalur koneksi yang dibentuk dari A ke J adalah.

```
Nama akun: A dan J
Tidak ada jalur koneksi yang tersedia

Anda harus memulai koneksi baru itu sendiri.
```

Gambar 6. Hasil output tidak ada jalur koneksi antara A dan J

BAB II
LANDASAN TEORI

2.1 Traversal Graf, BFS, DFS

Graf merupakan representasi suatu persoalan sedangkan traversal graf merupakan pencarian solusi persoalan tersebut. Algoritme traversal graf yaitu mengunjungi simpul dengan cara yang sistematik. Algoritme traversal graf terbagi menjadi dua jenis:

2.1.1 *Breadth First Search* (BFS)

Breadth First Search adalah pencarian yang dilakukan secara melebar. Misalkan traversal dimulai dari simpul v, maka algoritme BFS sebagai berikut:

1) Kunjungi simpul v.
2) Kunjungi semua simpul yang bertetangga dengan simpul v terlebih dahulu.
3) Kunjungi simpul yang belum dikunjungi dan bertetangga dengan simpul-simpul yang tadi dikunjungi, demikian seterusnya.

2.1.2 *Depth First Search* (DFS)

Depth First Search adalah pencarian yang dilakukan secara mendalam. Misalkan traversal dimulai dari simpul v, maka algoritme DFS sebagai berikut:

1) Kunjungi simpul v.
2) Kunjungi simpul w yang bertetangga dengan simpul v.
3) Ulangi DFS mulai dari simpul w.
4) Ketika mencapai simpul u sedemikian sehingga semua simpul bertetangga dengannya telah dikunjungi, pencarian dirunut-balik (*backtrack*) ke simpul terakhir yang dikunjungi sebelumnya dan mempunyai simpul w yang belum dikunjungi.
5) Pencarian berakhir bila tidak ada lagi simpul yang belum dikunjungi yang dapat dicapai dari simpul yang telah dikunjungi.

2.2 *C# Desktop Application Development*

Program dalam tugas besar ini dibuat dengan menggunakan kakas Visual Studio .NET. .NET Framework merupakan *software framework* yang dikembangkan oleh Microsoft pada tahun 2002. .NET Framework terdiri atas aturan kerja, aturan pemrograman, dan banyak *class library* (*Framework Class Library*) untuk membangun bermacam aplikasi, salah satunya aplikasi desktop. Pada tahun 2016, Microsoft mengembangkan .NET Core, yaitu .NET Framework yang bersifat *open-source* dan multiplatform, artinya .NET Core dapat dijalankan di platform

Windows, Linux, dan Mac OS. Bahasa pemrograman yang didukung .NET Core antara lain C#, Visual Basic, dan F#. Pada tugas besar ini, program ditulis menggunakan bahasa C#.

C# adalah bahasa pemrograman yang sederhana, modern, *object-oriented*, dan *type safe*. Bahasa ini berakar dari bahasa pemrograman C sehingga struktur program dan tata cara penulisannya mirip dengan bahasa pemrograman yang berakar dari C lainnya seperti Java, C++, dan Javascript.

Pada program aplikasi desktop, program utamanya disimpan pada file Program.cs. Contoh *source code* Program.cs untuk aplikasi desktop dengan template Windows Forms sebagai berikut.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace bacefook
{
    static class Program
    {
        /// <summary>
        ///  The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.SetHighDpiMode(HighDpiMode.SystemAware);
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

Blok namespace

Blok program

Blok pertama adalah blok namespace, yang berisi kode untuk memanggil namespace lain yang digunakan pada program yang kita tulis. Untuk memanggil namespace lain, digunakan kata kunci `using` diikuti nama namespace seperti contoh di atas. Namespace berisi kelas yang di dalamnya berisi fungsi-fungsi yang dapat digunakan oleh program.

Blok kedua berisi program yang kita tulis. Nama namespace biasanya sesuai dengan nama *project* atau nama folder tempat file ini disimpan. Pada contoh di atas, nama *project* yang kelompok kami buat adalah "bacefook". Kemudian di dalam blok `namespace bacefook` terdapat kelas (*class*). Di dalam sebuah namespace dapat berisi beberapa kelas. Namun, pada umumnya sebuah file hanya berisi sebuah kelas saja. Pada aplikasi desktop, nama kelas utama yang harus ada adalah "Program". Di dalam sebuah kelas dapat berisi banyak method. Namun, method yang harus ada untuk aplikasi desktop adalah method `Main()`. Method Main ini akan dipanggil pertama kali jika program dijalankan.

BAB III

ANALISIS PEMECAHAN MASALAH

3.1 Langkah-Langkah Pemecahan Masalah

Awalnya data dari file txt akan diolah menjadi sebuah graph. Pada program yang kami buat, digunakan sebuah fungsi TxtToGraph, dengan parameter nama file dan graf, akan terbentuk graf dengan node dan edges sesuai dengan txt tersebut. Kelas Graph yang kami implementasikan sendiri, memiliki dua atribut dan beberapa method. Atribut tersebut adalah sebuah ArrayList yang mencatak semua node pada graph, dan Dictionary atau map bernama adj dengan Key adalah string node dan value adalah Arraylist.

Misal terdapat 4 node pada graf dan merupakan graf lengkap, maka pada kelas graph Arraylist nodes dan Dictionary<string,arrayList> adj sebagai berikut:

ArrayList nodes -> [“A”,”B”,”C”,”D”]
Dictionary<string, ArrayList> adj = [(“A”, [“B”,”C”,”D”]) , (“B”, [“A”, “C”, “D”]), (“C”, [“A”,”B”, “D”]) , (“D” [“A”, “B”, “C”])]

Kemudian kelas graph memiliki method standar sebuah graph ditambah dengan method algoritma BFS, algoritma DFS, dan method lain untuk membantu fitur Explore Friend dan Friend Recommendation. Selanjutnya, objek graf tinggal diberikan message(method) yang sesuai untuk menyelesaikan permasalahan yang ada.

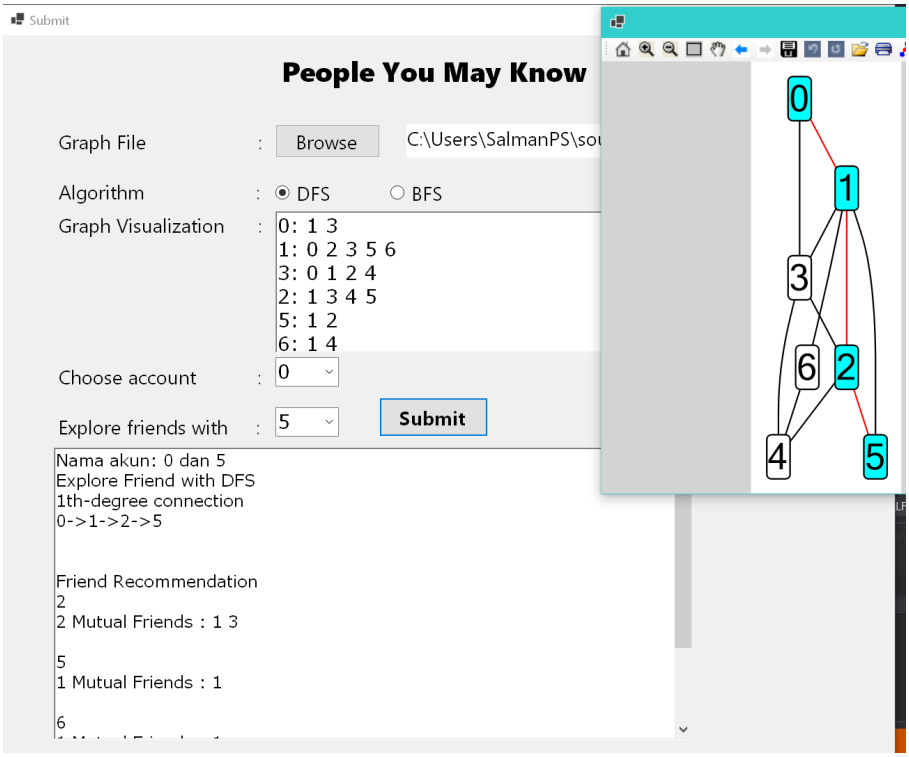
3.2 Proses Mapping Persoalan Menjadi Elemen-Elemen Algoritma BFS dan DFS

Pada method yang kami buat, algoritma BFS dan DFS memiliki beberapa parameter yang dibutuhkan. Untuk algoritma BFS yang kami buat, kami menggunakan 4 parameter, yaitu string startNode, string finalNode, dan Dictionary<string, int> distance, Dictionary<string, string> predPath. String startNode dan finalNode didapat dari pilihan user di dalam aplikasi, yang merupakan node pada graf. Dictionary distance akan mencatat panjang lintasan dari startNode menuju finalNode, jika finalNode dapat ditemukan. Dictionary predPath akan mencatat node predecessor dari sebuah node. Algoritma BFS menggunakan Queue<string> livingNode dan Dictionary<string,bool> visited. Queue livingNode untuk mencatat simpul hidup yang akan dikunjungi, dan Dictionary visited akan menandai simpul mana yang sudah dikunjungi.

Pada algoritma DFS, kami mengimplementasikan DFS dengan metode iteratif. Memiliki 3 paramater, yaitu string startNode, string finalNode, List<string> pathRet. String startNode dan

finalNode didapat dari pilihan user di dalam aplikasi, yang merupakan node pada graf. Dan List pathret akan mencatat lintasan yang dilalui dari startNode ke finalNode. Pada algoritma DFS ini digunakan Stack untuk mencatat node mana saja yang akan dikunjungi.

3.3 Contoh Ilustrasi Kasus



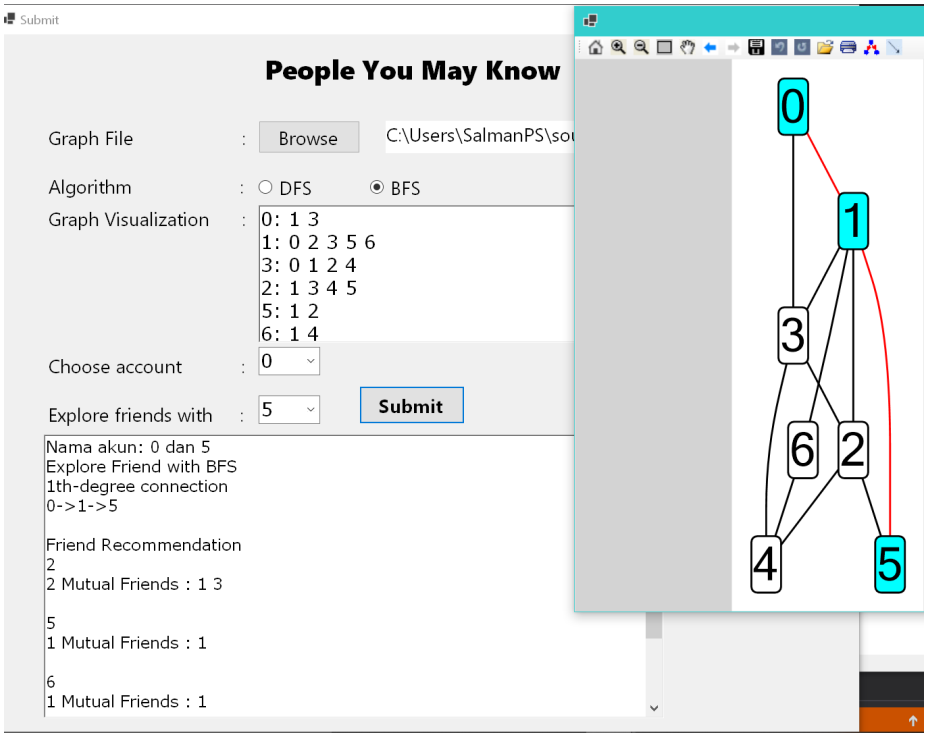
Gambar 7. Tampilan aplikasi GUI dengan pilihan algoritme DFS

Contoh kasus di atas, terdapat 6 buah node string, (0,1,2,3,4,5,6) dan 11 buah edge. Ketika akun 0 dipilih dan Explore Friend dengan akun 5, dengan algoritma DFS lintasan yang dilalui dari node 0 ke node 5 adalah 0->1 -> 2 ->, sesuai dengan gambar di atas. Jika ditelusuri dari awal, lintasan semula adalah 0 -> 1 -> 2 -> 4 -> 3 dan berhenti. Masih ada node yang belum dikunjungi dan belum menemukan final node, maka algoritma akan mencari node lain, yaitu lintasan menjadi 0 -> 1 -> 2 -> 4 -> 6 dan berhenti juga. Masih ada node yang belum dikunjungi dan belum menemukan final node, maka lanjut. Sehingga lintasan solusi menjadi 0 ->1 -> 2 ->5.

Antara akun 0 dan 5 terpisah sebuah akun lain, yaitu akun 1 (0 -> 1 -> 5), maka akun 0 dan 5 merupakan 1st-degree connection. Pada implementasi yang kami buat, kami mencatat jarak minimum sebuah node dari node sebelumnya, sehingga didapatkan 1st degree connection, bukan 2th degree connection jika mengikuti solusi lintasan DFS (0 ->1 -> 2 -> 5)

Untuk fitur Friend Recommendation, dengan algoritma DFS, akan dicari rekomendasi teman bagi akun 0 dengan diurutkan berdasarkan jumlah mutual friend terbanyak. Algoritmanya adalah mencari akun yang tidak berteman dengan akun 0, yaitu node yang tidak bertetangga dengan akun 0. Pada graf tersebut, akun 0 berteman dengan 2 buah akun lain, yaitu akun 1 dan 3. Akun 0 tidak berteman dengan 4 akun lainnya, yaitu akun 2, 4, 6, 5.

Untuk setiap akun yang bukan teman akun 0, maka akan dicek apakah akun tersebut berteman dengan akun temannya 0. Contoh akun 2, akan dicek dengan akun 1 dan 3. Didapat bahwa akun 2 berteman dengan akun 1 dan 3, sehingga akun 0 dan akun 2 memiliki 2 mutual friend (akun 1 dan 3). Untuk akun 4, berteman dengan akun 3, maka akun 0 dan 4 memiliki 1 mutual friend. Begitu juga dengan akun 6 dan 5. Akan ditampilkan semua rekomendasi akun dengan mutual friend lebih dari 0. Akun dengan mutual friend 0 tidak akan direkomendasikan (tidak ditampilkan).



Gambar 8. Tampilan aplikasi GUI dengan pilihan algoritme BFS

Dengan algoritma BFS, tidak jauh berbeda dengan algoritma DFS sebelumnya terutama di bagian Friend Recommendation. Untuk Explore Friend antara akun 0 dan 5, maka akan dilakukan pencarian secara BFS dan didapat solusi lintasan 0 -> 1 -> 5. Urutan node yang dikunjungi sebenarnya adalah 0 -> 1 -> 3 -> 2 -> 5. Tetapi didapat lintasan solusi adalah 0 -> 1 -> 5. Antara 0 dan 5 terpisah sebuah akun yaitu akun 1, maka antara akun 0 dan 5 merupakan 1st degree connection.

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Program (*Pseudocode*)

```
Procedure ProgramUtama()
{Program utama akan menerima input dari user, lalu akan melakukan penelusuran graph dan
visualisasi dengan algoritma BFS atau DFS}

Deklarasi
BFSChecked, DFSChecked: boolean
accountName, targetName, result, algorithm: string
MessageBox, RichTextBox: {komponen pada desktop app}
SocialNetworkGraph: Graph

Algoritma
SocialNetworkGraph <- {menerima input dari file}
BFSChecked, DFSChecked <- {menerima input dari pengguna berupa radio button click}
if !BFSChecked && !DFSChecked then
MessageBox.show("Algoritma belum dipilih", "Warning")
else
    if BFSChecked then algorithm <- "BFS" else algorithm <- "DFS" endif
accountName, targetName <- {menerima input dari pengguna berupa pilihan pada dropdown
list}
    if accountName = "" || targetName = "" then
            MessageBox.show("Choose account atau Explore Friend with belum dipilih",
"Warning")
    else
            result <- ExploreFriends(accountName, targetName, SocialNetworkGraph, algorithm)
            result <- result + FriendRecommendation(SocialNetworkGraph, accountName,
algorithm)
            RichTextBox.Text <- result
            SocialNetworkGraph.visualizeGraph(accountName, targetName, algorithm)
    endif
endif
```

4.2 Struktur Data yang Digunakan dan Spesifikasi Program

Struktur data utama yang digunakan pada kali ini adalah graph. Graph direpresentasikan dengan kelas. Kelas tersebut memiliki 2 atribut, *nodes* dan *adj*. Nodes bertipe data ArrayList, pada implementasinya, ArrayList ini memiliki elemen bertipe data *string*. Nodes digunakan untuk untuk menyimpan semua nama dari node yang tersedia pada graph. Adj digunakan untuk menyimpan semua node beserta dengan nama node yang bersebelahan dengannya. Adj direpresentasikan dengan tipe data Dictionarynya dengan key bertipe string dan valuenya bertipe ArrayList. Value pada dictionary itu pada implementasinya adalah array of string. Seperti class pada umumnya, class Graph memiliki default constructor dan juga destructor. Constructor digunakan untuk membuat sebuah objek bertipe Graph, dan destructor berfungsi sebaliknya,

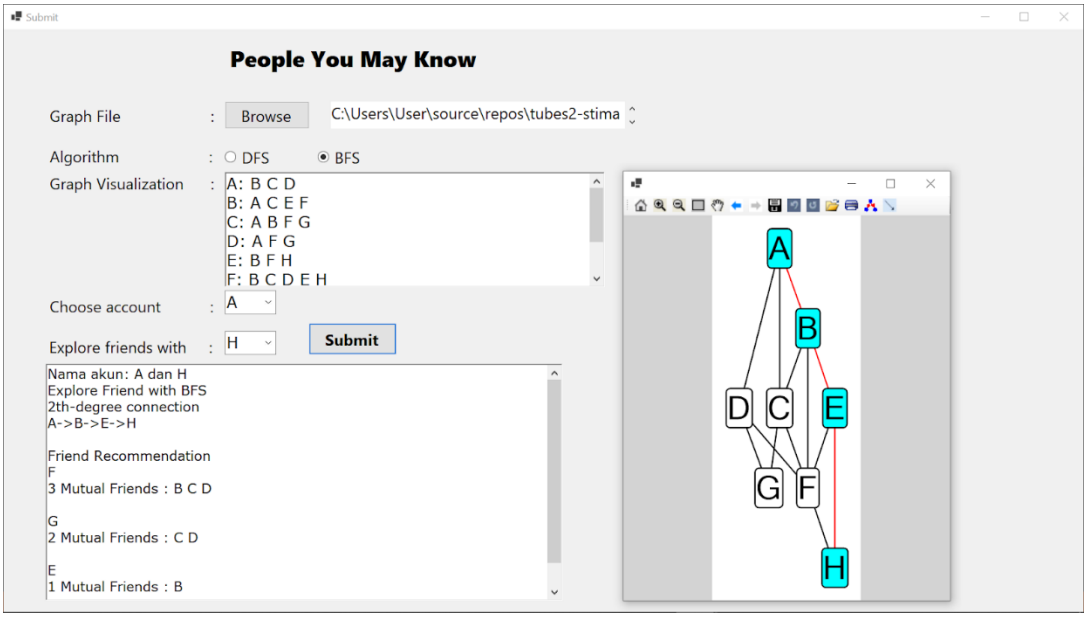
yaitu untuk menghancurkan objek bertipe Graph. Graph memiliki metode AddNode yang digunakan untuk menambahkan node ke graph, adapula metode AddEdge untuk menambahkan edge ke graph. CountDegree digunakan untuk mehgitung berapa jumlah edge yang ada pada suatu node. Selain metode tersebut ada beberapa metode penting lainnya, metode BFS digunakan untuk menelusuri graph dengan menggunakan konsep Breadth First Search. Metode DFSIterative digunakan untuk menelusuri graph secara Depth First Search dan juga secara iteratif, sebenarnya ada cara lain yaitu secara rekursif.

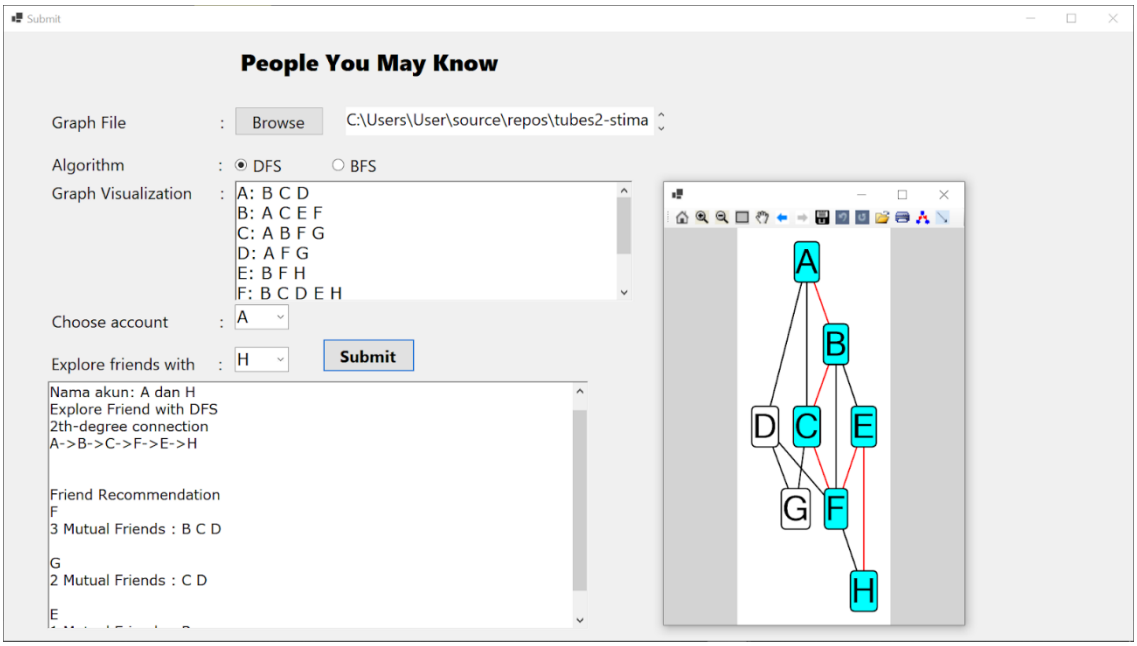
4.3 Tata Cara Penggunaan Program

Program dapat menerima input dengan cara membaca file melalui filesystem. Format file yang dibaca adalah "txt". Isi file yang dibaca memiliki format tertentu. Barisan paling atas adalah jumlah baris yang akan dibaca. Sejumlah baris setelahnya adalah nama node yang dibatasi dengan spasi, hal ini menunjukan bahwa kedua note tersebut bersebelahan, artinya ada edge yang menghubungkan dua node tersebut. Setelah membaca file input, kita akan memilih algoritma yang akan digunakan untuk komputasi. Kita dapat memilih antara BFS atau DFS. Jika kita klik submit sebelum melakukan pemilihan algoritma, maka akan keluar pesan kesalahan. Setelah kita memilih algoritma yang akan digunakan untuk komputasi, kita akan memilih account. Account dapat kita anggap sebagai orang utama, yang nanti akan kita berikan rekomendasi teman dari fitur *friend recommendation*. Kita juga akan memasukan input untuk siapa yang akan kita *explore friends with*. Kedua input tersebut adalah berupa dropdown yang dapat kita pilih. Jika kita tidak melakukan pemilihan tersebut, maka akan diberikan pesan kesalahan juga. Setelah semua input yang perlu diisi telah diisi, kita dapat klik submit untuk melakukan penelusuran social network. Setelah berhasil submit, kita akan ditunjukan visualisasi graph sesuai dengan input yang kita berikan. Visualisasi graph yang diberikan adalah lintasan yang perlu dilalui untuk mencapai node dengan nama yang kita input pada *explore friends with* dimulai dari node dengan nama yang diinput di *account*. Akan ditampilkan juga berapa derajat antara kedua node yang kita sudah input. Kita juga akan ditampilkan lintasan yang harus diambil tetapi kali ini berupa tulisan, bukan visualisasi. Terakhir, akan ditampilkan semua *friend recommendation* untuk node pada account.

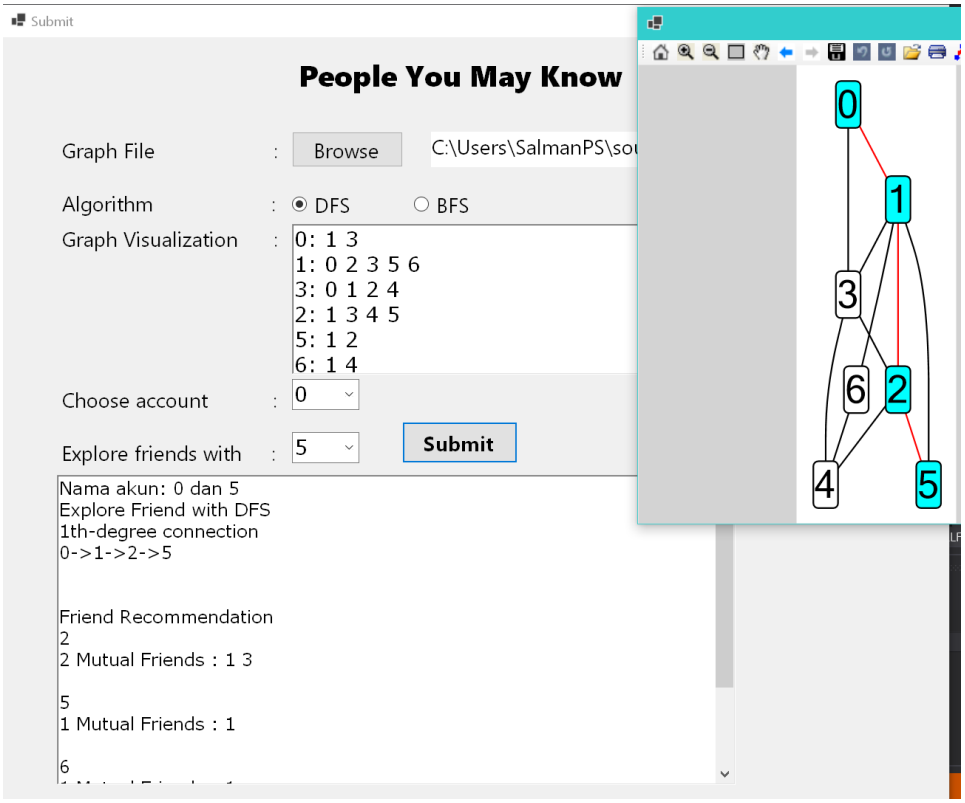
4.4 Hasil Pengujian

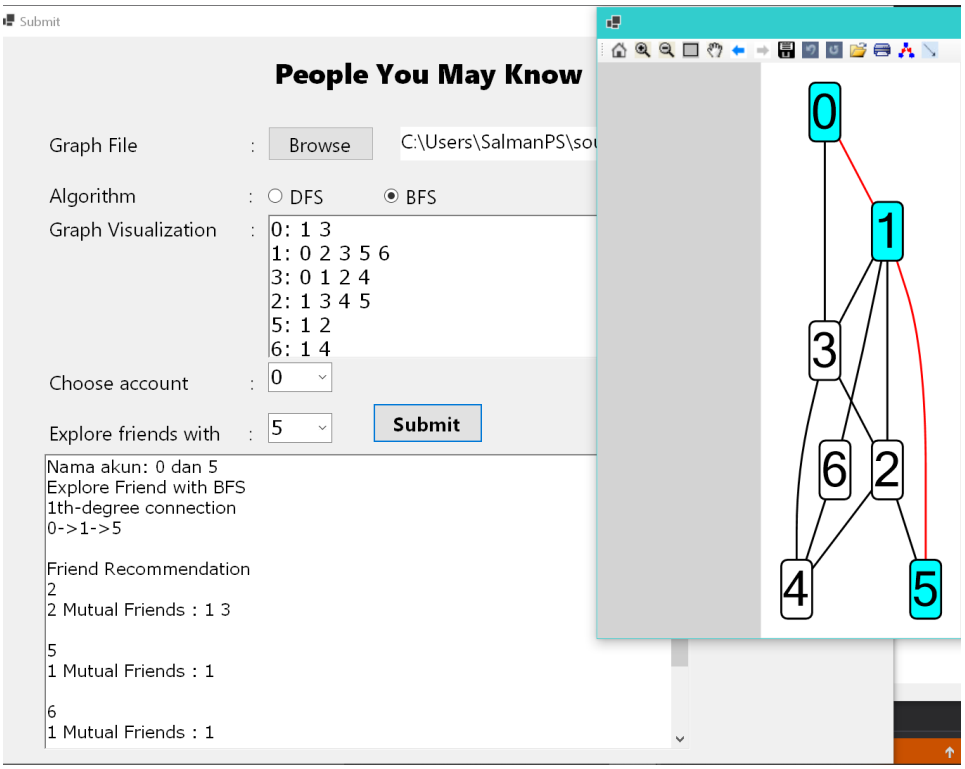
4.4.1 Input1.txt



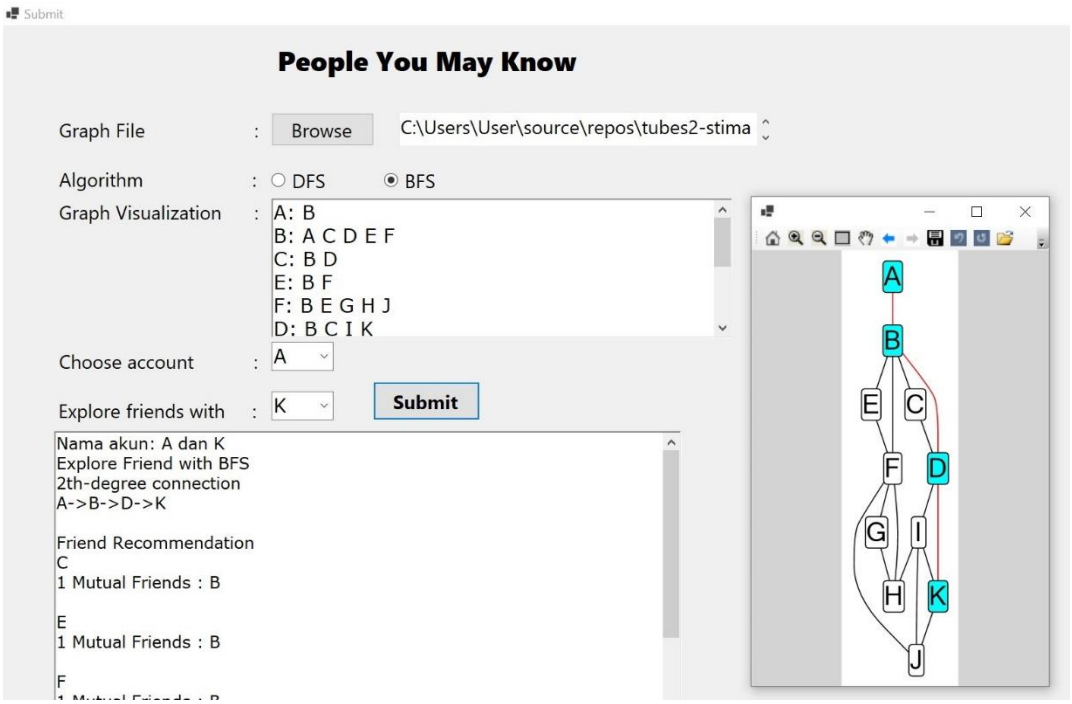


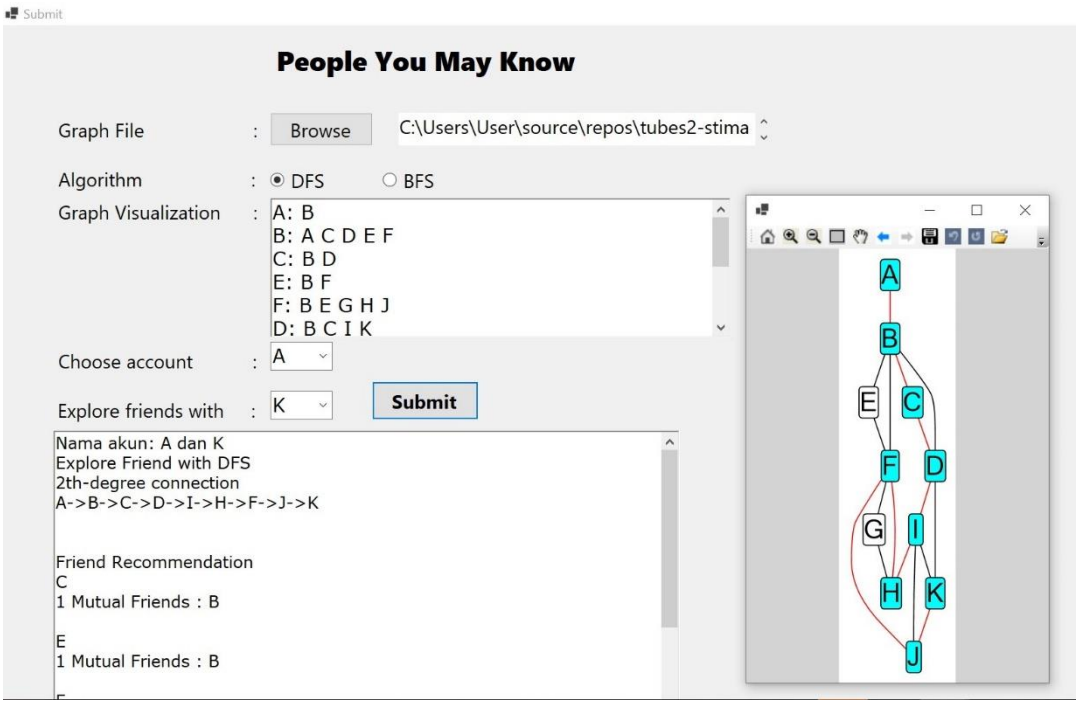
4.4.2 Input2.txt


Submit
People You May Know
Graph File
Browse
C:\Users\SalmanPS\sou
Algorithm
DFS
BFS
Graph Visualization
0: 1 3
1: 0 2 3 5 6
3: 0 1 2 4
2: 1 3 4 5
5: 1 2
6: 1 4
Choose account
0
Explore friends with
5
Submit
Nama akun: 0 dan 5
Explore Friend with DFS
1th-degree connection
0->1->2->5
Friend Recommendation
2
2 Mutual Friends : 1 3
5
1 Mutual Friends : 1
6
0
1
3
6
2
4
5

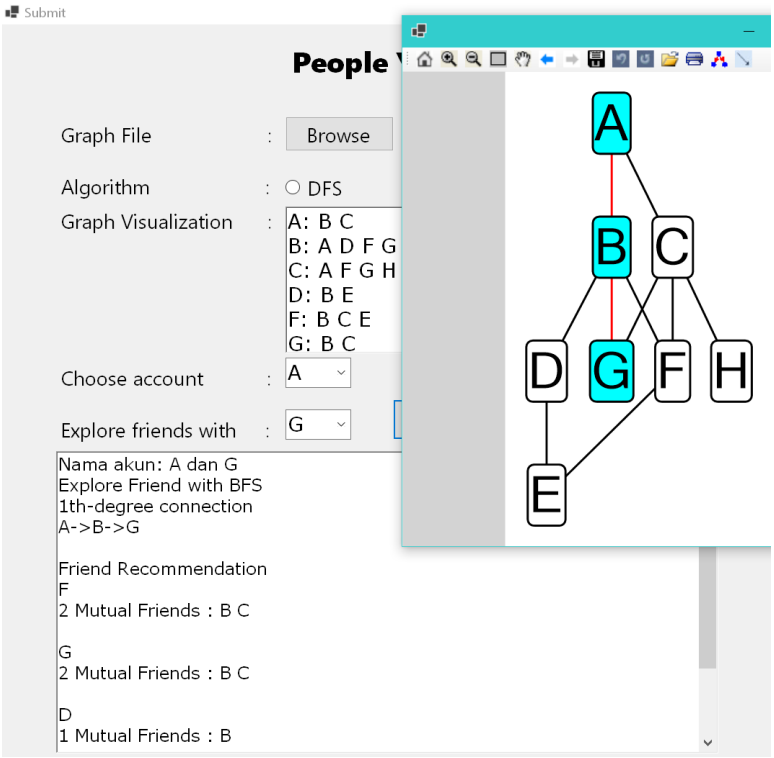

Submit
People You May Know
Graph File
Browse
C:\Users\SalmanPS\sou
Algorithm
DFS
BFS
Graph Visualization
0: 1 3
1: 0 2 3 5 6
3: 0 1 2 4
2: 1 3 4 5
5: 1 2
6: 1 4
Choose account
0
Explore friends with
5
Submit
Nama akun: 0 dan 5
Explore Friend with BFS
1th-degree connection
0->1->5
Friend Recommendation
2
2 Mutual Friends : 1 3
5
1 Mutual Friends : 1
6
1 Mutual Friends : 1
0
1
3
6
2
4
5

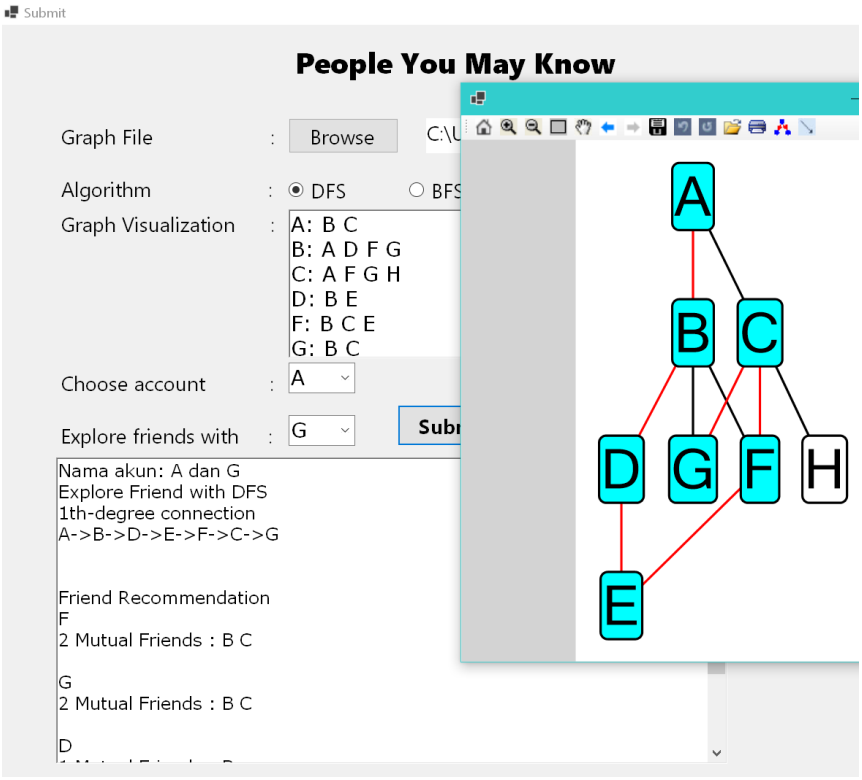
4.4.3 Input3.txt



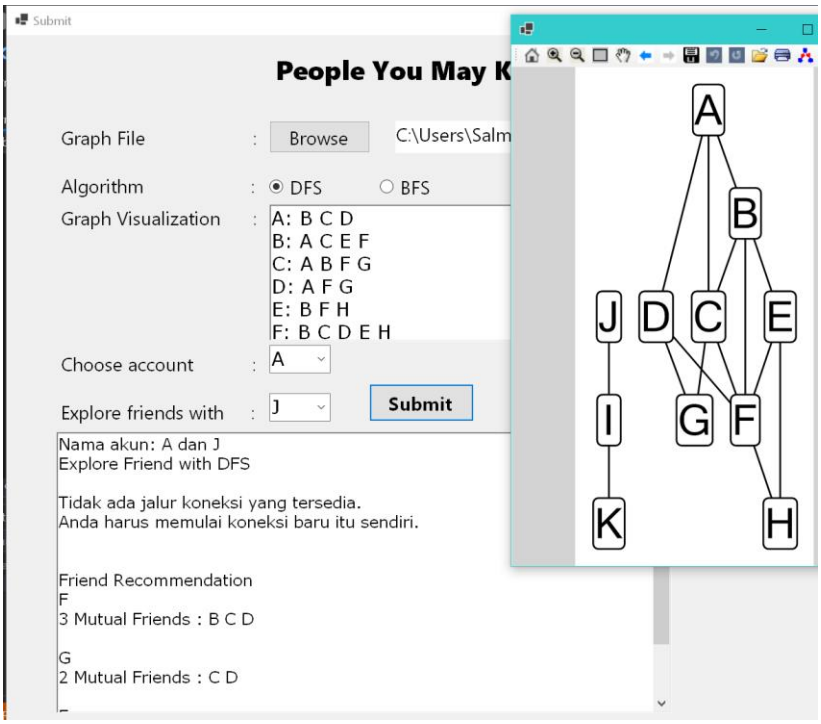


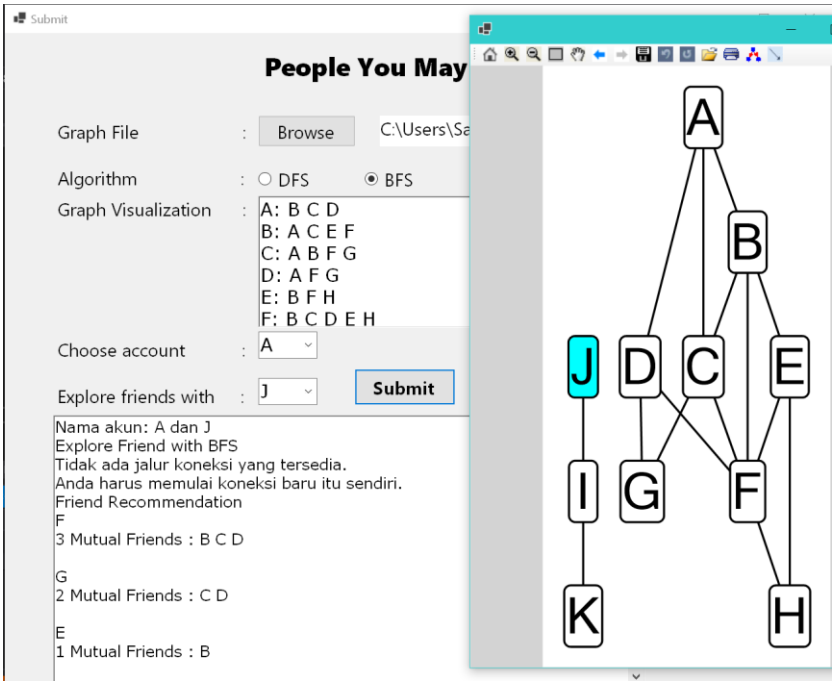
4.4.4 Input4.txt





4.4.5 Input5.txt


Submit
People You May K
Graph File : Browse C:\Users\Salm
Algorithm : DFS BFS
Graph Visualization :
A: B C D
B: A C E F
C: A B F G
D: A F G
E: B F H
F: B C D E H
Choose account : A
Explore friends with : J
Submit
Nama akun: A dan J
Explore Friend with DFS
Tidak ada jalur koneksi yang tersedia.
Anda harus memulai koneksi baru itu sendiri.
Friend Recommendation
F
3 Mutual Friends : B C D
G
2 Mutual Friends : C D
A
B
J
D
C
E
I
G
F
K
H


Submit
People You May
Graph File : Browse C:\Users\Sa
Algorithm : DFS BFS
Graph Visualization :
A: B C D
B: A C E F
C: A B F G
D: A F G
E: B F H
F: B C D E H
Choose account : A
Explore friends with : J
Submit
Nama akun: A dan J
Explore Friend with BFS
Tidak ada jalur koneksi yang tersedia.
Anda harus memulai koneksi baru itu sendiri.
Friend Recommendation
F
3 Mutual Friends : B C D
G
2 Mutual Friends : C D
E
1 Mutual Friends : B
A
B
J
D
C
E
I
G
F
K
H

4.5 Analisis dari Desain Solusi Algoritma BFS Dan DFS yang Diimplementasikan

Terdapat perbedaan hasil dari penggunaan algoritma BFS dan DFS pada fitur Explore Friend. Ada beberapa kondisi atau kasus dimana menggunakan BFS adalah pilihan yang tepat, adapun kondisi yang sebaliknya, ada kondisi dimana menggunakan algoritma DFS akan memberikan hasil yang lebih baik. Pada kasus pencarian lintasan dimana dua simpul tersebut bersebelahan, penggunaan algoritma BFS akan menghasilkan lintasan yang lebih pendek dan juga waktu pencarian yang lebih sedikit, dikarenakan sifatnya yang mencari secara melebar terlebih dahulu. Sebaliknya, penggunaan algoritma pada kasus tersebut tidak akan terlalu memberikan hasil yang optimal, lintasan yang diberikan belum tentu yang paling dekat, dan juga waktu pencarian yang dibutuhkan lebih lama dibandingkan algoritma BFS. Ada juga kondisi dimana penggunaan algoritma DFS lebih baik daripada menggunakan algoritma BFS, yaitu pada saat kita ingin mencari lintasan antara dua simpul dimana dua simpul tersebut terpisah jarak yang cukup jauh. Waktu pencarian yang dibutuhkan dengan algoritma DFS akan lebih cepat, hal ini dikarenakan sifatnya dari DFS atau Depth First Search yang akan mencari secara mendalam, oleh karena itu, simpul simpul yang berjarak jauh akan lebih cepat dicapai. Sebaliknya, penggunaan BFS pada persoalan seperti ini akan membutuhkan waktu pencarian yang lebih lama, hal ini dikarenakan sifat dari BFS yang mencari secara melebar, yang kemudian menyebabkan simpul simpul yang berjarak jauh akan lebih lama untuk dijangkau.

BAB V
PENUTUP

5.1 Kesimpulan

Algoritma DFS dan BFS adalah algoritma yang dapat kita gunakan untuk pencarian pada graf. Graf sendiri adalah representasi suatu persoalan dimana terdapat simpul dan sisi. Salah satu cara untuk melakukan pencarian pada graf adalah dengan cara traversal. Algoritma traversal graf dapat kita bagi menjadi dua, BFS dan DFS. BFS atau Breadth First Search adalah pencarian yang dilakukan secara melebar, sedangkan DFS atau Depth First Search adalah pencarian yang dilakukan secaara mendalam.

Algoritma DFS dan BFS dapat kita gunakan untuk mengimplementasikan fitur *people you may know* seperti yang ada pada beberapa media sosial. Adapun kelebihan dan kekurangan dari kedua algoritma tersebut pada kasus kasus tertentu. Algoritma DFS dan BFS

Kita dapat menggunakan kakas Visual Studio .Net untuk mengembangkan aplikasi dengan GUI atau Graphical User Interface pada platform desktop. Bahasa yang digunakan untuk hal ini adalah C#. Karakteristik dari bahasa tersebut adalah object-oriented dan type safe. Maka dari itu, penulisan dari pada program ini dilakukan dengan pendekatan object oriented.

5.2 Saran

Penulisan kode dengan menggunakan *best practices* yang sudah ada dapat meningkatkan kecepatan dari pada proses *development*. Seharusnya kita dapat menuliskan kode dengan lebih rapi atau biasa disebut dengan menerapkan *clean code*. Beberapa contohnya adalah menggunakan konvensi penamaan yang sama, menggunakan nama variabel yang baik. Dengan menerapkan konsep - konsep tersebut, akan lebih memudahkan untuk mengerjakan secara kolaboratif, dikarenakan tingkat keterbacaan yang tinggi.

Saat mengembangkan suatu aplikasi ataupun program, ada baiknya kita memikirkan kenyamanan dari orang yang akan menggunakan program kita. Hal ini dapat kita terapkan dari bagaimana kita membangun UI atau User Interface dan juga UX atau User Experience. Hal - hal seperti ini mungkin terlihat minor, tetapi ketika kita mengembangkan aplikasi yang berbasis pengguna pada khalayak umum, maka hal ini akan menjadi sangat penting sekali.

5.3 Refleksi dan Komentar terhadap Tugas

Melalui tugas besar ini, kita dapat lebih memahami bagaimana penggunaan algoritma - algoritma yang sudah ada seperti BFS dan DFS untuk menyelesaikan persoalan. Dengan tugas ini, kita dapat menerapkan secara langsung algoritma BFS dan DFS. Melalui hal tersebut kita akan lebih mengerti serta dapat membuka ruang untuk analisa yang lebih luas. Tugas ini juga dibungkus dengan menarik, yaitu dengan menggunakan *desktop application* untuk sarana input output dan juga tampilan untuk ke pengguna. Hal ini membuat kita sebagai mahasiswa memiliki kemampuan untuk belajar secara mandiri juga.

DAFTAR PUSTAKA

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Tugas-Besar-2-IF2211-Strategi-Algoritma-2021.pdf diakses pada tanggal 3 Maret 2021.

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag1.pdf diakses pada tanggal 23 Maret 2021.

[3] https://github.com/microsoft/automatic-graph-layout diakses pada tanggal 23 Maret 2021.

[4] Faisal, M. Reza dan Erick Kurniawan. 2020. *Seri Belajar Windows Forms: Membangun Aplikasi Desktop berbasis .NET Core 3.1 dengan Visual Studio 2019*. Banjarmasin: Indonesia .NET Developer Community.