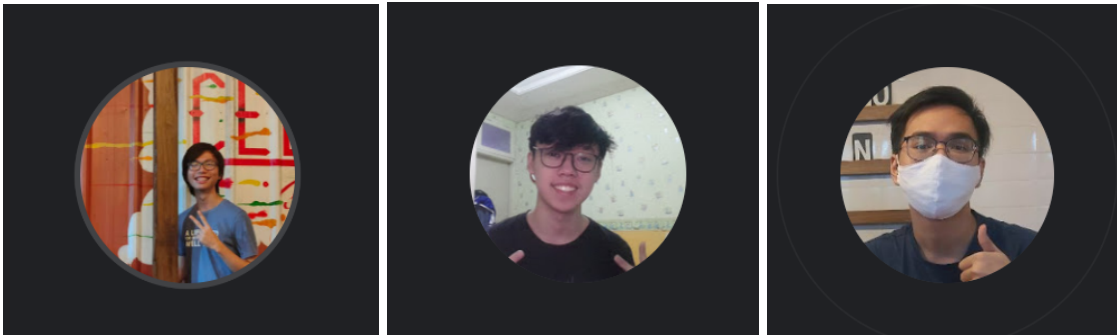


# **PENGAPLIKASIAN ALGORITMA BFS DAN DFS DALAM FITUR *PEOPLE YOU MAY KNOW* JEJARING SOSIAL FACEBOOK**

Laporan Tugas Besar 2 IF2211 Strategi Algoritma  
Semester II Tahun Akademik 2020/2021



Oleh:

Kelompok 29 - Grandma

Dionisius Darryl Hermansyah	13519058
James Chandra	13519078
Jordan Daniel Joshua	13519098

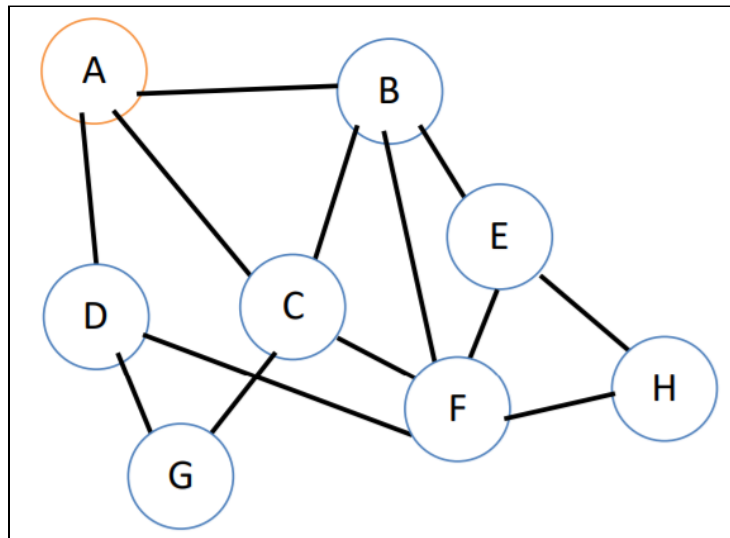
**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG  
2021**

# BAB I

## DESKRIPSI TUGAS

### 1.1. Deskripsi tugas

Dalam tugas besar ini, kami 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), Kita dapat menelusuri *social network* pada akun facebook untuk mendapatkan rekomendasi teman seperti pada fitur *People You May Know*. Selain untuk mendapatkan rekomendasi teman, kami juga diminta untuk mengembangkan fitur lain agar dua akun yang belum berteman dan tidak memiliki mutual friends sama sekali bisa berkenalan melalui jalur tertentu. Visualisasi graf pertemanan yang dihasilkan dari file eksternal:



Gambar 1.1. 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

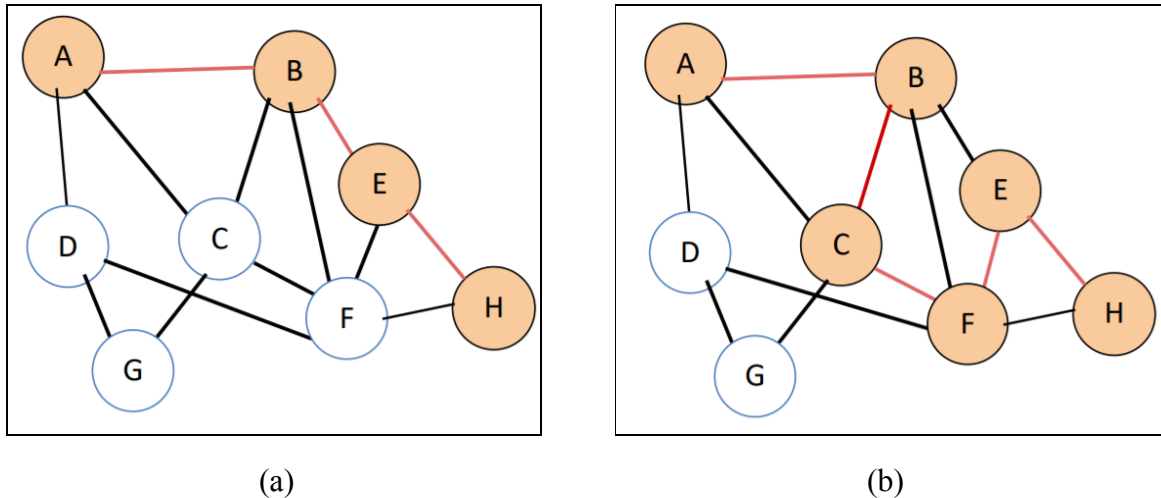
Untuk fitur explore friends, misalnya pengguna ingin mengetahui seberapa jauh jarak antara akun A dan H serta bagaimana jalur agar kedua akun bisa terhubung. Berikut output graf dengan penelusuran BFS yang dihasilkan. Berikut output yang diharapkan untuk penelusuran menggunakan BFS:

Nama akun: A dan H

2nd-degree connection

$A \rightarrow B \rightarrow E \rightarrow H$

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.



Gambar 1.2. Hasil visualisasi jalur koneksi menggunakan (a) BFS dan (b) 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.

## 1.2. Spesifikasi Program

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.

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 kaskas yang tersedia. Sebagai referensi, salah satu kaskas yang tersedia untuk melakukan visualisasi adalah MSAGL.
4. Terdapat dua fitur utama, yaitu:
  - a. *Fitur friend recommendation*
    - i. 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.

ii. 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*

i. 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).

ii. Program menerima pilihan dua akun yang belum berteman.

iii. Program akan menampilkan nilai N-th degree connection antar kedua akun dan memberikan jalur melalui akun mana saja sampai kedua akun bisa terhubung.

iv. 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.

v. Jika tidak ditemukan jalur koneksi sama sekali antar kedua akun karena graf not fully connected, maka tampilkan informasi bahwa kedua akun tidak dapat terhubung.

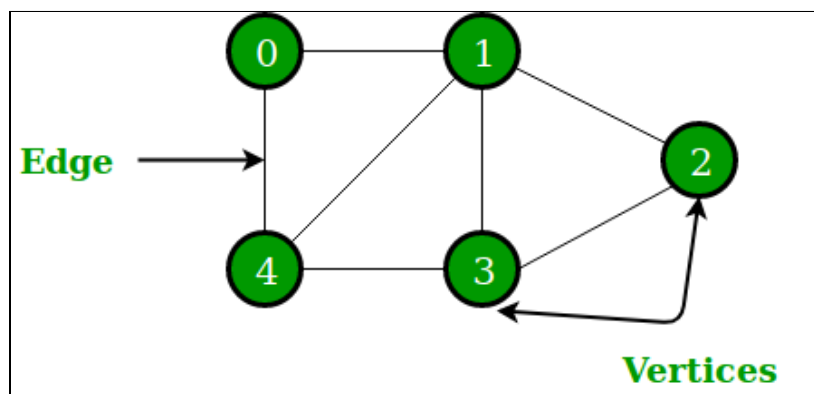
5. Mahasiswa tidak diperkenankan untuk melihat atau menyalin library lain yang mungkin tersedia bebas terkait dengan pemanfaatan BFS dan DFS.

## BAB II

### LANDASAN TEORI

#### 2.1. *Graph traversal*

*Graph traversal* adalah sebuah proses mengunjungi semua simpul yang terdapat di sebuah graf terhubung secara sistematis. Adapun graf yang dimaksud adalah representasi dari sebuah persoalan yang dapat dicari solusinya.



Gambar 2.1. Sebuah graf tidak berarah (Sumber: Google Images)

Secara umum *graph traversal* dibagi menjadi 2 jenis yaitu pencarian melebar (Breadth First Search/BFS) dan pencarian mendalam (Depth First Search/DFS). Terdapat 2 pendekatan yang ada untuk proses pencarian yaitu graf statis (graf yang sudah dibentuk sebelum proses pencarian) dan graf dinamis (graf yang terbentuk saat proses pencarian dilakukan). Contoh beberapa permasalahan yang bisa diselesaikan dengan *graph traversal* adalah *Citation Map*, penjelajahan web dengan *Web Spider*, pencarian dokumen di dalam direktori, pencarian rute terpendek, dan pencarian solusi beberapa permainan papan (tic-tac-toe, catur, sudoku, dll) pada *artificial intelligence*.

#### 2.2. *Breadth first search (BFS)*

*Breadth first search* adalah salah satu algoritma traversal graf yang mencari solusi pada sebuah graf dengan mengunjungi semua simpul dari akarnya terlebih dahulu. Algoritma dari BFS ini dimulai dengan pertama-tama mengunjungi simpul akar kemudian mengunjungi semua

simpul anak dari simpul akar tersebut. Setelah itu akan dikunjungi semua simpul anak dari simpul anak yang barusan dikunjungi dengan syarat bahwa simpul yang akan dikunjungi belum pernah dikunjungi sebelumnya. Adapun kompleksitas waktu dan ruang dari BFS adalah  $b^d$  ( $b$  = jumlah maksimum simpul dari sebuah akar,  $d$  = kedalaman graf). Pencarian dari BFS juga menjamin kelengkapan dan keoptimalan dari jawaban yang ada.

### **2.3. Depth first search (DFS)**

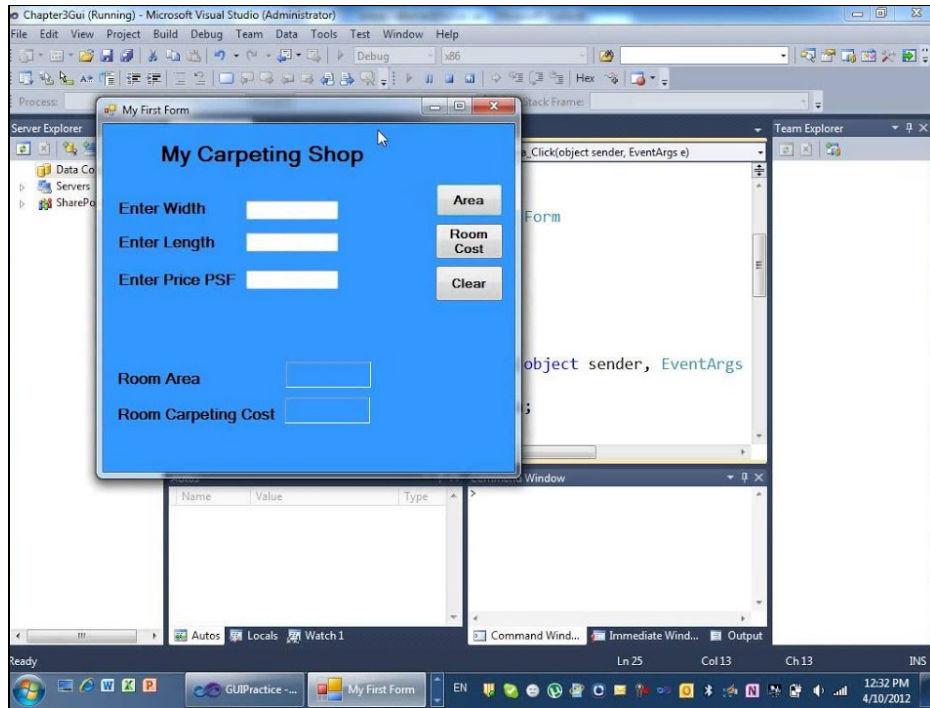
*Depth first search* adalah salah satu algoritma traversal graf yang digunakan untuk mencari solusi permasalahan pada suatu graf terhubung secara mendalam. Algoritma pencarian solusi ini tergolong ke dalam kategori algoritma pencarian solusi tanpa informasi tambahan atau *uninformed/blind search*, yang berarti tidak berbasiskan heuristik dan hanya memiliki informasi berupa persoalan itu sendiri dalam representasi graf.

Cara kerja algoritma *depth first search* dilakukan secara mendalam, yang berarti pada suatu graf dengan traversal yang dimulai dari suatu simpul tertentu (katakanlah  $v$ ), maka kita akan mengunjungi simpul tersebut, kemudian mengunjungi simpul yang bertetangga dengan simpul tersebut (katakanlah  $w$ ), lalu mengulangi DFS dengan simpul  $w$ , saat mencapai suatu simpul, katakanlah  $u$ , dimana semua simpul yang bertetangga dengannya sudah dikunjungi, maka dilakukan runut-balik hingga ditemukan simpul sebelumnya yang memiliki simpul  $w$  yang belum dikunjungi. Pencarian akan selesai saat tidak ada lagi simpul yang belum dikunjungi yang dapat dicapai.

### **2.4. C# desktop application development**

C# adalah sebuah bahasa pemrograman berorientasi objek yang bersifat *general purpose* (dapat digunakan untuk berbagai *use case* yang berbeda tanpa perlunya rombakan signifikan). Bahasa C# dikembangkan oleh Microsoft pada tahun 2000 serta dirancang oleh Anders Hejlsberg, seorang rekayasawan perangkat lunak yang juga merupakan pencipta Turbo Pascal.





Gambar 2.4.1. Visual Studio yang menjadi IDE bagi C# (Sumber: Google Images)

Bahasa C# merupakan bahasa yang *strongly-typed* dan relatif mudah dimengerti, serta menyediakan beragam library yang dapat digunakan pengembang. Seperti yang telah disebutkan sebelumnya, bahasa C# merupakan bahasa berorientasi objek, sehingga menerapkan konsep dasar pemrograman berorientasi objek seperti enkapsulasi, polimorfisme, dan *inheritance*. Pengembangan serta debugging untuk bahasa ini biasa dilakukan dengan editor Visual Studio (yang digunakan pula untuk bahasa-bahasa framework .NET lainnya seperti Visual Basic).

Framework .NET adalah framework pengembangan perangkat lunak pada suatu runtime yang paling predominan digunakan dalam platform Windows. Selain memiliki akses ke library kelas-kelas yang sangat ekstensif (FCL), framework ini juga menawarkan *common language runtime* yang cara kerjanya adalah menyediakan *virtual machine* sebagai lapisan perlindungan keamanan lebih dan berperan juga dalam exception handling program dan lainnya.

## BAB III

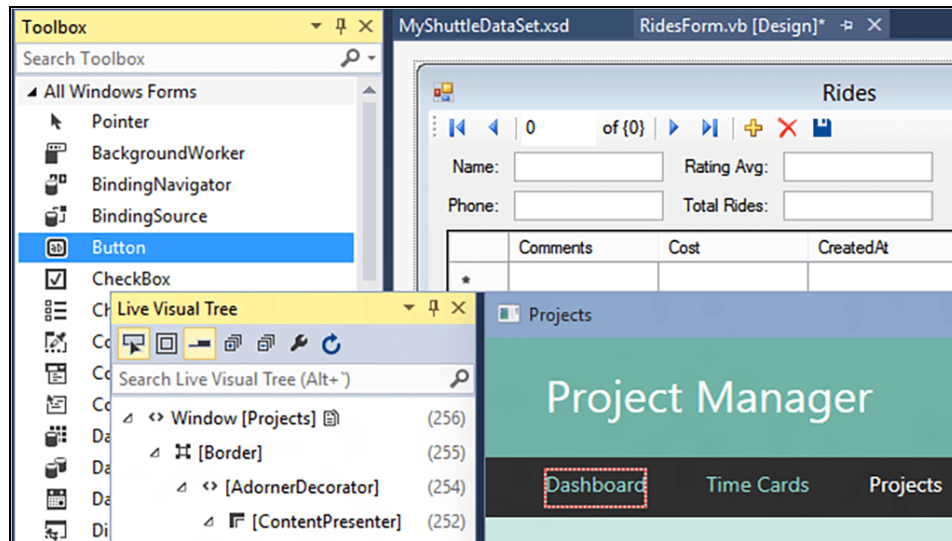
### ANALISIS PEMECAHAN MASALAH

#### 3.1. Langkah pemecahan masalah

Ada 2 permasalahan utama yang harus diselesaikan pada tugas besar 2 ini. Permasalahan pertama adalah pengimplementasian fitur *People You May Know* dan yang kedua adalah *Explore Friends*. Kedua fitur ini memiliki perbedaan pada strategi algoritma yang digunakan, namun secara garis besar, struktur data dan jenis algoritma yang digunakannya serupa yaitu graf tak berarah serta algoritma BFS dan DFS.

Pertama-tama harus ditemukan solusi untuk merepresentasikan jejaring sosial terkait dan komponen-komponennya. Komponen yang ada adalah subjek berupa orang atau pengguna jejaring sosial serta hubungan antara subjek berupa pertemanan antara satu orang dengan orang lain. Permasalahan ini dapat dipecahkan dengan menggunakan struktur data graf tak berarah. *Node* (simpul) pada tiap graf akan merepresentasikan satu orang, sedangkan hubungan pertemanan akan digambarkan dengan *vertex* (busur) antar simpul. Selain struktur data kelas graf yang harus direpresentasikan, harus pula diimplementasikan fungsi untuk membaca file eksternal dan mengkonversinya ke dalam sebuah graf.

Untuk kedua permasalahan di atas, yaitu *People You May Know* dan *Explore Friends*, dibutuhkan sebuah cara untuk menelusuri setiap jaringan pertemanan yang ada antar orang. Hal ini dapat dipecahkan dengan menggunakan algoritma untuk melakukan *traversal* graf. Terdapat dua jenis algoritma yang bisa digunakan yaitu *breadth first search* (BFS) dan *depth first search* (DFS).



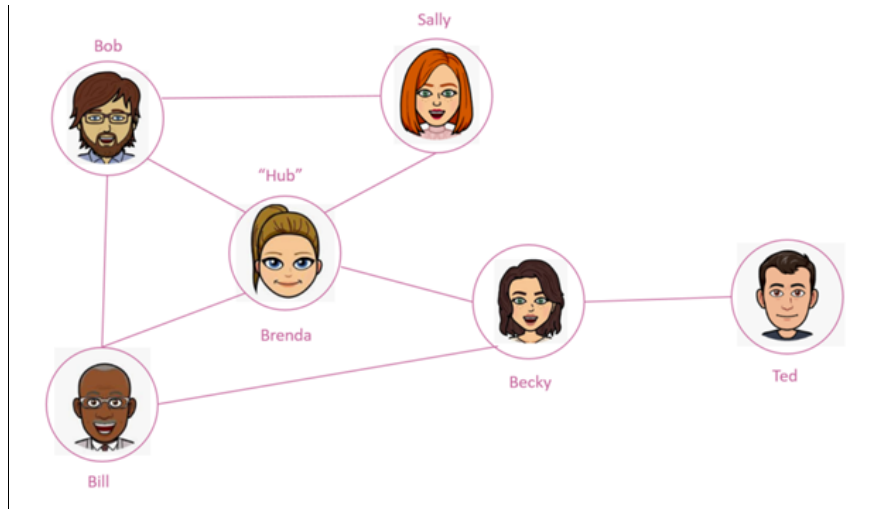
Gambar 3.1. Kakas pengembangan aplikasi desktop C# dengan .NET untuk pembuatan GUI (Sumber: Microsoft)

Adapun permasalahan umum lain yang dihadapi adalah spesifikasi *graphical user interface* (GUI) yang perlu didesain dan dibuat sedemikian rupa. Untuk memecahkan masalah ini, dapat digunakan kakas bahasa C# yang merupakan sebuah kakas untuk mengembangkan *desktop application* (aplikasi desktop) yang dilengkapi dengan *.NET framework*.

### 3.2. Mapping persoalan

Berdasarkan analisis pemecahan masalah yang telah dilakukan, maka berikut merupakan *mapping* persoalan ke dalam algoritma BFS dan DFS serta dalam implementasi program nantinya:

- Jaringan *social network* : Graf tidak berarah
- Pengguna *social network* : *Node* (simpul) pada graf
- Hubungan pertemanan : *Edge* (sisi) pada graf



Gambar 3.2. Ilustrasi jaringan pertemanan antara 6 orang yang dapat direpresentasikan dengan graf tak berarah (Sumber: Medium)

Setelah melakukan *mapping* persoalan tersebut, berikut merupakan rancangan singkat dari algoritma BFS dan DFS yang akan diimplementasikan untuk masing-masing fitur yang akan dijelaskan lebih lanjut pada bab selanjutnya:

- Untuk fitur *friend recommendation*, algoritma yang digunakan merupakan algoritma BFS, dengan pembatasan level pencarian. Hal ini karena, seorang *recommended friend* dapat didefinisikan sebagai sebuah simpul level 2 dari sebuah akar graf, dimana akar tersebut merupakan simpul yang dipilih.
- Untuk fitur *friend explore*, terdapat 2 algoritma yang dapat dipilih yaitu BFS dan DFS. Sesuai dengan cara kerja algoritma masing-masing, algoritma yang dipilih akan menjadi sebuah *path* atau jalur dari simpul yang dipilih ke simpul tujuan dan meng-*output* hasil yang sesuai. Jika tidak ditemukan maka di-*output* pesan tidak ada koneksi yang mungkin.

### 3.3. Ilustrasi kasus lain

Selain kasus yang terdapat pada spesifikasi, akan dilakukan eksplorasi pula terhadap *edge case* yang dapat dikarenakan anomali atau ketidaksesuaian *input* maupun konfigurasi pemrosesan pada program, akan dijelaskan pula masing-masing kasus tersebut serta bagaimana ekspektasi keluarannya pada aplikasi perangkat desktop nantinya pada saat pengembangan.

Pertama, pada *input*, apabila kita memasukkan jumlah *edge* yang lebih atau kurang atau tidak sesuai dengan banyaknya pasangan *node* yang diketik, maka harapannya kasus yang terjadi adalah, akan ada pasangan *node* yang tidak terbaca atau akan ada *input edge* yang menjadi tidak valid. Kasus ini dijamin tidak terjadi karena diasumsikan bahwa masukan sudah 100% valid.

Kemudian terdapat pula kasus dimana *input* akan menghasilkan graf yang tidak terhubung, apabila kita ingin mencari koneksi di antara *node* yang tidak ada hubungannya/terputus, maka harapan *output* adalah tidak bisa mencari *n-th degree connection* dan tidak akan memunculkan pula *node* itu pada bagian *friend recommendation* karena sudah dijamin tidak ada *mutual friends*. Kasus ini sudah dibuat sedemikian rupa agar memiliki *output* yang wajar demi praktik *good code*, walau sudah dijamin spesifikasi tugas besar, tidak akan terjadi.

Sebagai penjas tambahan, output yang dikeluarkan akan menjelaskan pula bahwa *node* tersebut tidak terhubung, sehingga visualisasi jalan graf juga tidak dilakukan.

Ada Pula ilustrasi kasus samping yang dapat terjadi pada beberapa kondisi pemrosesan, seperti pencarian derajat koneksi, katakan node *A* ke node *B* saat node *A* tidak memiliki teman mutual dengan *B*, maka walaupun pada keadaan normal, node *B* tidak akan dicetak, karena ingin dicari derajat koneksinya, maka node *B* akan tercetak namun diberikan output pula tidak memiliki *mutual friends*. Perhatikan apabila ternyata node *B* punya mutual friends dengan node *A* namun sudah langsung berteman dengan node *A*, sama halnya seperti deskripsi kasus sebelumnya, harapannya, node *B* dengan derajat koneksi akan dicetak namun ditambah dengan teman mutualnya, walaupun awalnya tidak dicetak karena *A* dan *B* sudah berteman langsung.

## BAB IV

### IMPLEMENTASI DAN PENGUJIAN

#### 4.1. Implementasi program

Berikut ini merupakan implementasi program utama dan algoritma *Grandma Social Network* dalam bahasa C# yang digunakan dalam bentuk *pseudocode*:

```
function getAdjNode(input n:Node, g:Graph) -> Node
ALGORITMA
    foreach (tetangga(n) in g)
        if tetangga(n).isVisited = false
            return tetangga(n)
```

```
procedure resetIsVisited(input,output : Graph)
KAMUS
    n : Node
ALGORITMA
    foreach (n in Graph)
        n.isVisited <- false
```

```
function cari_mutual(input n,e:Node, g:Graph) -> Array[Node]
KAMUS LOKAL
    q : Queue<Node>
    curr_level, arr_index : integer
    hasil : Array[Node]
    levelOne : List<Node>
    curr_node : Node
ALGORITMA
    {Inisialisasi}
    resetisVisited(g)
    n.isVisited <- true
    q.Enqueue(n)

    curr_level <- 1
    arr_index <- 0

    while (q.Count!=0 and curr_level<=levelOne.Count+1) do
        curr_node <- q.Dequeue()
        foreach (tetangga(curr_node)) do
            if (tetangga(curr_node)=e and levelOne.Contains(curr_node)) then
                hasil[arr_index] <- curr_node
                arr_index <- arr_index+1
            if (tetangga(curr_node).isVisited = false) then
                if curr_level == 1
                    levelOne.add(tetangga(curr_node))
                q.Enqueue(tetangga(curr_node))
                tetangga(current_node).isVisited <- true
            cur_level <- cur_level +1
        -> hasil
```

```

procedure fr(input s:Node, g:Graph; output hasil:Queue<Node>)
KAMUS
    allNodeMutual : List<Node,int>
    temp : Tuple<Node,int>
    node : Node
    n,i,j,max_idx : int
ALGORITMA
    {Inisialisasi}
    resetIsVisited(g)
    foreach (node in g) do
        allNodeMutual.add(node,cari_mutual(s,node).Count)

    {Sort array dari jumlah mutual terbanyak ke kecil}
    n = nodes.Length
    i <- 0
    while (i < n - 1)
        max_idx = i;
        j <- i+1
        while (j < n) do
            if (allNodeMutual[j].Item2 > allNodeMutual[max_idx].Item2) then
                max_idx <- j
            j <- j+1

        temp <- allNodeMutual[max_idx];
        allNodeMutual[max_idx] <- allNodeMutual[i];
        allNodeMutual[i] <- temp;
        i <- i+1

    {Memasukkan elemen ke-1 dari allNodeMutual ke hasil}
    i <- 0
    while (i < n) do
        hasil[i] = nodes[allNodeMutual[i].Item1];
        i <- i+1

```

```

procedure fe_bfs(input s,e:Node, g:Graph; output hasil:Queue<Node>)
KAMUS
    prev,q : Queue<Node>
    n,at : Node
    curr_node : Node
ALGORITMA
    {Inisialisasi}
    resetIsVisited(g)
    found <- false
    s.isVisited <- true
    q.push(s)

    {Operasi pencarian node parent dari masing-masing node}
    while (q.count!=0) do
        n <- q.Dequeue
        foreach (tetangga(n) in g) do
            if tetangga.isVisited = false then
                q.Enqueue(tetangga(n))
                tetangga(n).isVisited = true
                prev[tetangga(n)] = n

    for (at=e; at!=null; at=prev[at]) do
        hasil.Enqueue(at)
    hasil.Reverse()
    if (hasil[0]!=s) then
        hasil.Clear()

```

```

    procedure fe_dfs(input s,e:Node, g:Graph; output hasil:Queue<Node>)
    KAMUS
        found : boolean
        curr_node : Node
    ALGORITMA
        {Inisialisasi}
        resetIsVisited(g)
        found <- false
        s.isVisited <- true
        hasil.push(s)

        {Operasi pencarian jalur}
        while (hasil.count!=0 and not found) do
            curr_node <- getAdjNode(hasil[0])
            if curr_node = -1 then
                hasil.Pop()
            else
                if curr_node = e then
                    found <- true
                    curr_node.isVisited <- true
                    hasil.Push(curr_node)
        hasil.Reverse()

```

## 4.2. Struktur data

Berdasarkan program dan algoritma yang telah diimplementasikan, dalam praktiknya digunakan beberapa struktur data dalam bentuk kelas dan objek. Berikut merupakan penjelasan singkat dari masing-masing struktur data yang ada:

Tabel 4.2.1. Struktur data dalam program

Struktur Data	Atribut dan Method	Keterangan
Node	name	Atribut nama node
	isVisited	Atribut apakah node sudah dikunjungi
	Node(name)	Konstruktor Node dengan inisiasi nama dan isVisited=false
Graph	n_max	Jumlah node maksimum



n_node	Jumlah node efektif
nodes	Array of node
m	<i>Adjacency matrix</i> untuk representasi graf
q	Queue untuk algoritma BFS
Graph(max)	Konstruktor Graph
addNode(a)	Menambahkan node baru ke dalam graph
addEdge(a, b)	Menambahkan sisi baru ke dalam graph
getAdjNode(node_pos)	Mengambil node adjacent yang belum di kunjungi
printNodeX(X)	Cetak node pada posisi X
printAllNodes()	Cetak seluruh node
getAllNodeInString()	Mereturn nodes yang ada dalam bentuk string
getAllNodesInArray()	Mereturn nodes yang ada dalam bentuk array
getAdjMatrix()	Mereturn m ( <i>adjacency matrix</i> )
getNNode()	Mereturn n_node
findIdxNode(s)	Mereturn index node bernama s
initNodes(input)	Inisialisasi node pada sebuah graf
initEdges(input)	Inisialisasi sisi antar node pada sebuah graf
sortNode()	Melakukan sorting pada node
initArray(arr)	Inisialisasi sebuah array kosong
cari_mutual(n, end)	Mencari <i>mutual friend</i>
fr(s)	Mencari <i>friend recommendation s</i>
getResult_fr(s, N)	Me-return string result dari <i>friend recommendation</i>
getResult_fe(Q)	Me-return string result dari <i>friend explore</i>
fe_dfs(s, e)	Algoritma <i>friend explore</i> dengan DFS
fe_bfs(s, e)	Algoritma <i>friend explore</i> dengan BFS

	getNeighbour(e)	Mencari tetangga dari node e
	solve(s)	Fungsi untuk menyelesaikan <i>friend explore</i>
	reconstructPath(s, e, prev)	Membuat path untuk <i>friend explore</i>
	resetIsVisited()	Me- <i>reset</i> seluruh isVisited node pada graf menjadi false
Form1	G	Graph program utama
	Form1()	Konstruktor form yang merupakan GUI utama
	handleGraphFile()	Membaca graf dari file eksternal
	buttonXClick()	Menangani <i>event</i> saat sebuah button diklik, ada beberapa implementasi yang berbeda sesuai dengan nomor button X.
Form3	Form3()	Konstruktor Form3, merupakan form yang digunakan untuk animasi <i>loading screen</i>

#### 4.3. Tata cara penggunaan program

Tata cara penggunaan program, baik untuk menjalankan program atau mengkompilasi program, adalah sebagai berikut:

- Untuk menjalankan program, dapat di-*run* secara langsung melalui file Grandma.exe yang terdapat pada direktori ./bin.
- Jika ingin meng-*compile* program, dapat dibuka file *solution* Grandma.sln pada ./src/Grandma/Grandma.sln dalam Visual Studio dan melakukan kompilasi dengan F5 atau CTRL + F5.

Pastikan struktur folder sama dan tidak ada yang berubah terutama file data test yang digunakan dapat ditaruh dalam folder ./test/ atau menggunakan data test yang sudah ada.

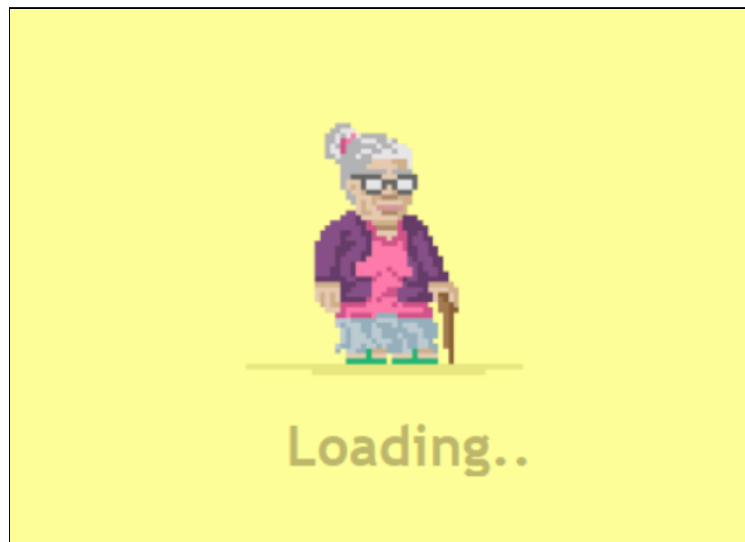
“Grandma Social Network” memiliki beberapa fitur seperti halnya sebuah jejaring sosial media. Berikut merupakan penjelasan singkat dari fitur-fitur yang dimiliki oleh program:

- Fitur utama *friend recommendation*: Sebuah fitur yang terinspirasi dari fitur *People You May Know* dalam jejaring sosial media seperti Facebook. Dengan memanfaatkan algoritma *Breadth First Search* (BFS), pengguna dapat menelusuri *social network* dengan akun lainnya untuk mendapatkan rekomendasi teman.

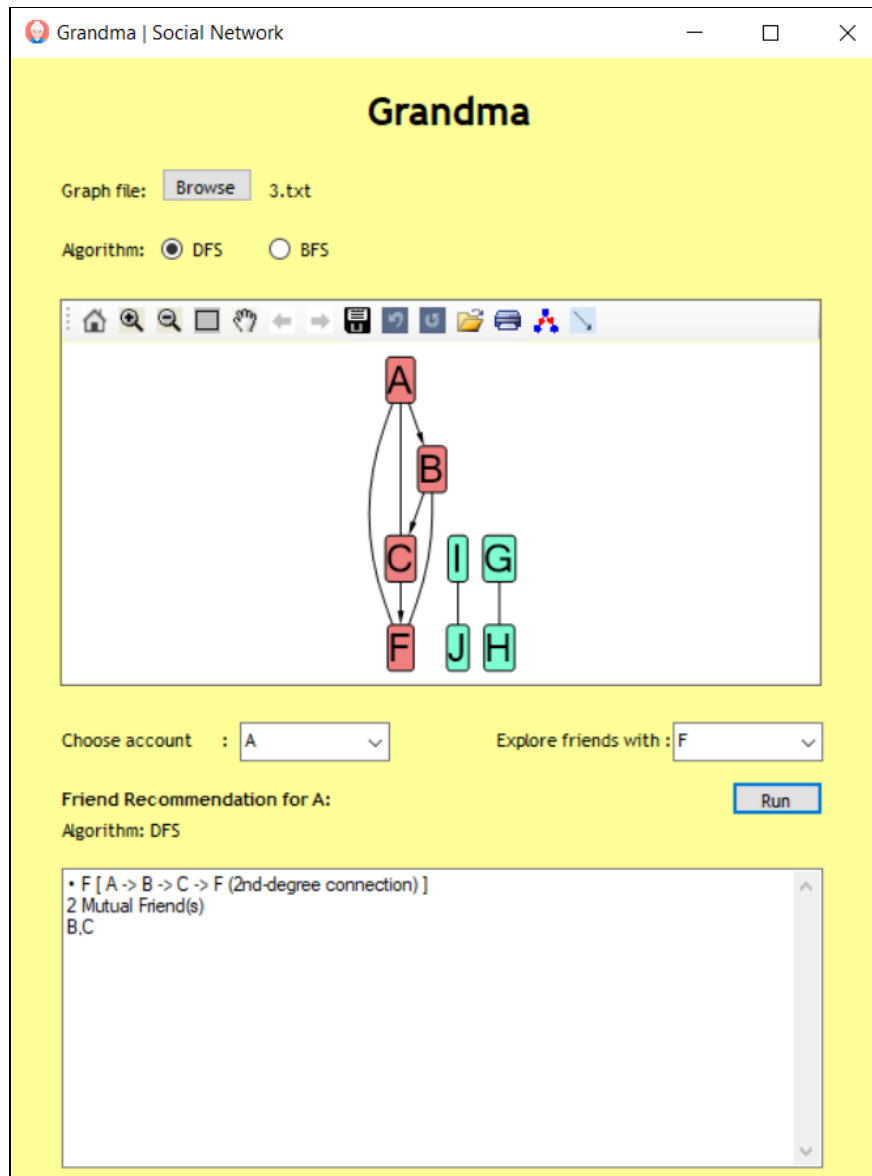
- b. Fitur utama *friend explore*: Sebuah fitur yang menampilkan jarak *degree* dari pengguna ke pengguna lain yang dipilih oleh pengguna tersebut. Fitur ini juga dapat menampilkan jalur pertemanan yang dapat ditempuh secara langsung oleh pengguna dengan menggunakan algoritma pilihan pengguna baik *Breadth First Search* (BFS) maupun *Breadth First Search* (DFS).
- c. Fitur utama *graph visualization*: Sebuah fitur yang dapat memvisualisasikan graf secara interaktif menggunakan Microsoft Automatic Graph Layout (MSAGL). Pengguna juga dapat berinteraksi dengan graf seperti memindahkan node ke satu tempat ke tempat lain secara *real-time*.
- d. Fitur konstruksi *custom graph*: Sebuah fitur yang dapat menerima file eksternal seperti file .txt yang berisi sebuah susunan sisi dari graf. Program dapat membaca file tersebut dan mengkonstruksi graf yang dapat langsung dipakai dalam program.

#### 4.4. Hasil pengujian

Program “Grandma Social Network” memiliki tampilan utama seperti yang ditunjukkan oleh gambar 4.4.1 dan gambar 4.4.2.



Gambar 4.4.1. *Loading Screen* “Grandma Social Network”

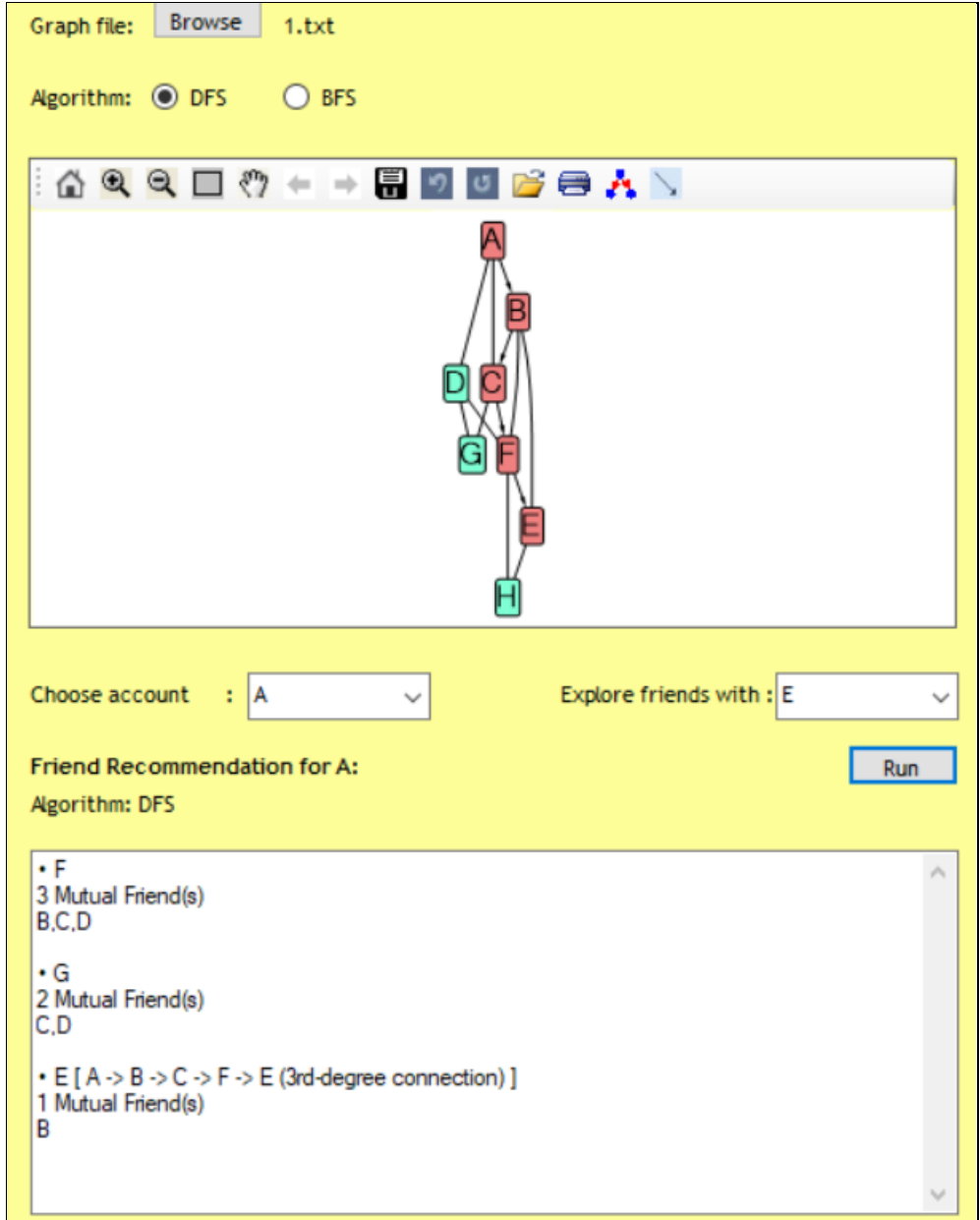


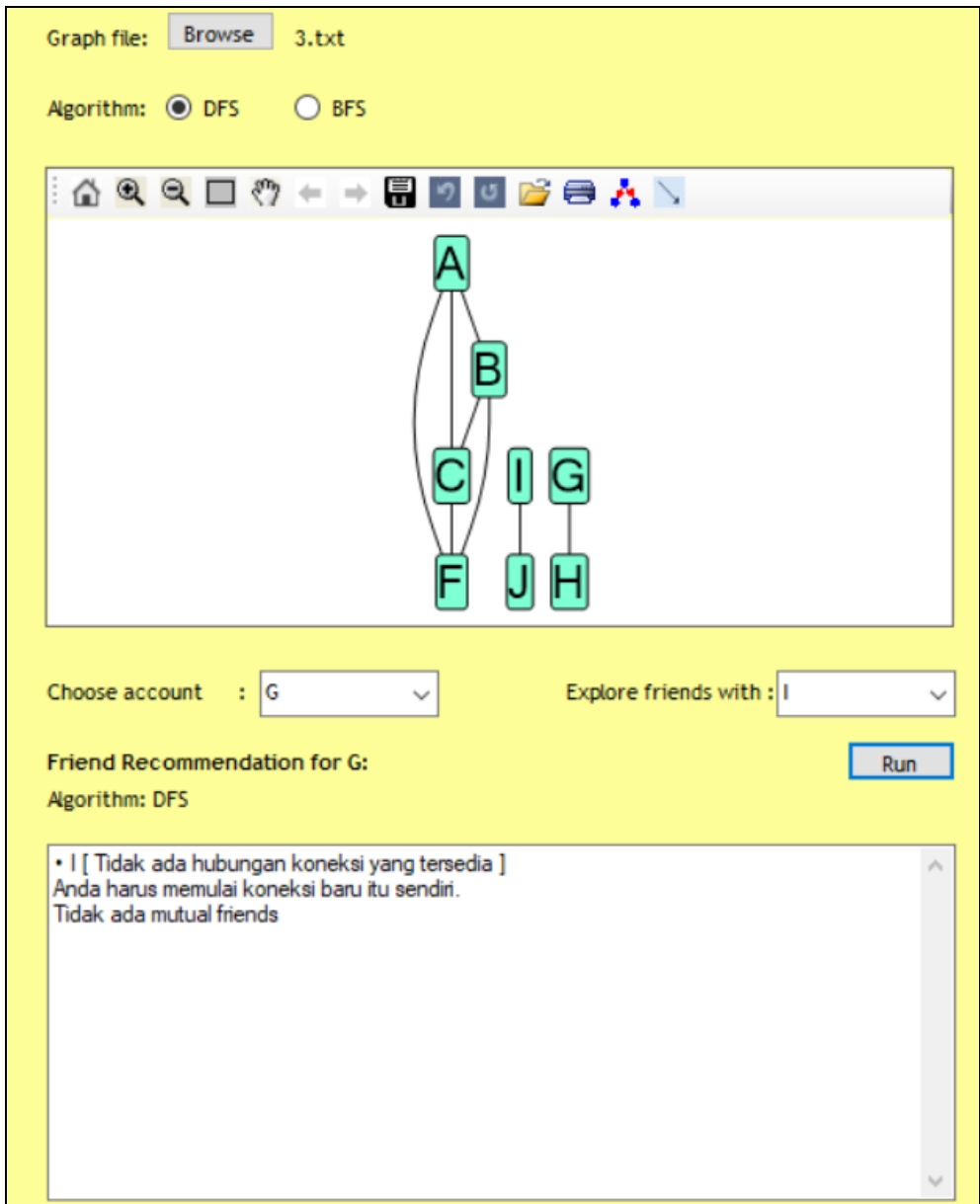
Gambar 4.4.1. Tampilan utama program GUI “Grandma Social Network”

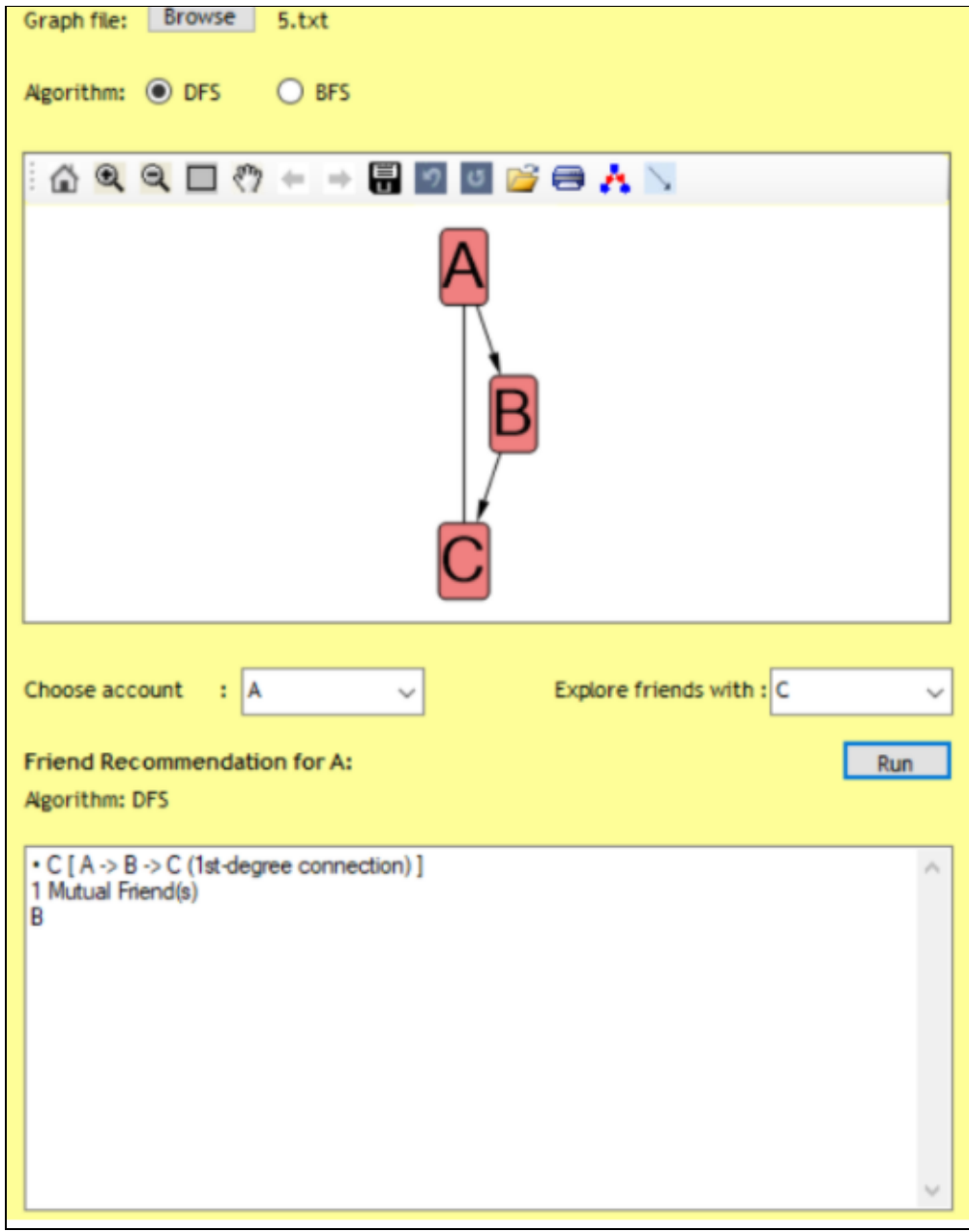
Tabel 4.4.1. Menunjukkan hasil *running* program pada beberapa data uji yang diberikan dalam bentuk file eksternal .txt.

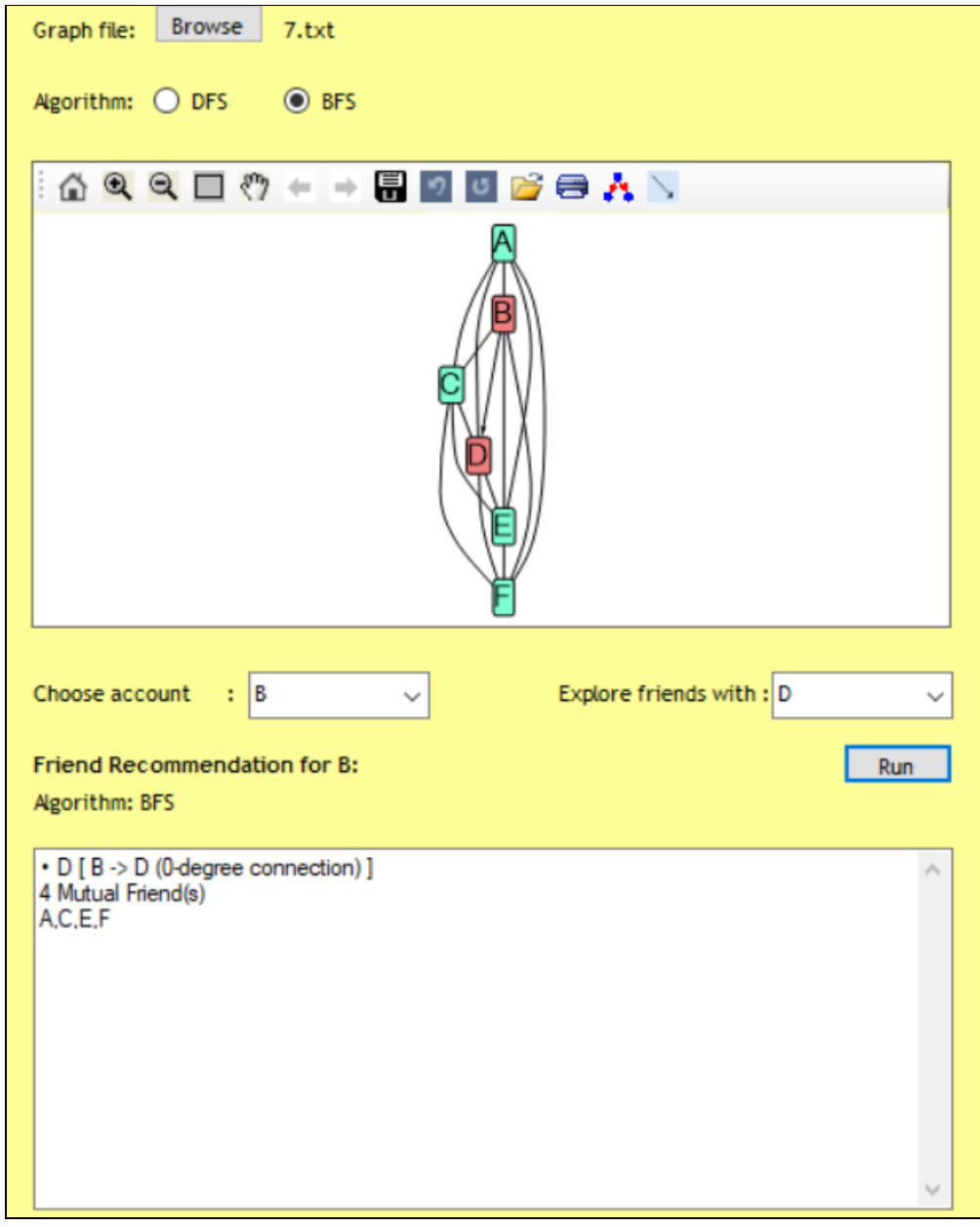
Tabel 4.4.1. Hasil pengujian program

Data Uji	Output program
13 A B	<i>Explore friends</i> antara A dan E dengan DFS

<p> AC  AD  BC  BE  BF  CF  CG  DG  DF  EH  EF  FH </p>	
<p> 8  AB  AC  BC  FB  FC  FA  GH  IJ </p>	<p>Explore friends antara G dan I dengan DFS</p>

	 <p>Graph file: <input type="button" value="Browse"/> 3.txt</p> <p>Algorithm: <input checked="" type="radio"/> DFS <input type="radio"/> BFS</p> <p>Choose account : <input type="text" value="G"/> Explore friends with : <input type="text" value="I"/></p> <p>Friend Recommendation for G: <input type="button" value="Run"/></p> <p>Algorithm: DFS</p> <p>• I [ Tidak ada hubungan koneksi yang tersedia ] Anda harus memulai koneksi baru itu sendiri. Tidak ada mutual friends</p>
<p>3 A B A C B C</p>	<p><i>Explore friends</i> antara A dan B dengan DFS</p>

	
15 AB AC AD AE AF BC BD BE	<p><i>Explore friends</i> antara B dan D dengan BFS</p>

<p> B F  C D  C E  C F  D E  D F  E F </p>	
<p> 20  A B  A C  A D  B C  B E  B F  C F  C G  D G  D F </p>	<p>Explore friends antara F dan A dengan BFS</p>



E H  
 E F  
 F H  
 I J  
 I H  
 I K  
 L M  
 Z Y  
 P F  
 Q D

Graph file:  6.txt

Algorithm: ☐ DFS ☒ BFS

Choose account :  Explore friends with :

Friend Recommendation for F:

Algorithm: BFS

- A [ F -> B -> A (1st-degree connection) ]  
3 Mutual Friend(s)  
B,C,D
- G  
2 Mutual Friend(s)  
C,D
- I  
1 Mutual Friend(s)  
H
- Q

#### 4.5. Analisis solusi algoritma

Tabel 4.5.1. menunjukkan hasil analisis solusi algoritma baik untuk fitur *friend recommendation* maupun *friend explore* yang telah diimplementasikan.

Tabel 4.5.1. Analisis solusi algoritma

Algoritma	Analisis
<i>Friend recommendation</i> dengan BFS	Pencarian rekomendasi teman menggunakan BFS sangatlah efektif. Hal ini terbukti dari hasil yang dihasilkan merupakan solusi yang optimal. Adapun kompleksitas waktu dari algoritma ini sama seperti kompleksitas BFS pada umumnya yaitu $O(E+V)$ .
<i>Friend explore</i> dengan BFS	<i>Friend explore</i> menggunakan BFS menghasilkan solusi yang optimal. Hal ini dikarenakan pencarian jalur teman yang terpendek dilakukan dengan membangkitkan semua simpul anak yang ada terlebih dahulu sehingga pencarian teman yang ada menghasilkan jalur dengan kedalaman yang terendah. Untuk kompleksitas waktu dari algoritma ini sama seperti kompleksitas BFS pada umumnya yaitu $O(E+V)$
<i>Friend explore</i> dengan DFS	<i>Friend explore</i> menggunakan DFS tidak selalu menghasilkan solusi yang optimal. Hal ini karena pencarian dilakukan dengan membangkitkan anak dari simpul sehingga dalam mencari sebuah simpul tidak selalu dihasilkan jalur dengan kedalaman yang minimum. Kompleksitas waktu dari algoritma ini sama seperti algoritma-algoritma sebelumnya yaitu $O(E+V)$ .

## **BAB V**

### **KESIMPULAN, SARAN, REFLEKSI, DAN KOMENTAR**

#### **5.1. Kesimpulan**

Kesimpulan dari tugas besar ini adalah fitur jejaring sosial Facebook yaitu *people you may know* dapat dimodelkan dan diaplikasikan menggunakan algoritma *breadth first search* (BFS) dan *depth first search* (DFS) dalam bentuk fitur *friend explore* dan *friend recommendation* serta dapat dibuat GUI-nya (*graphical user interface*) menggunakan kakas C# .NET.

#### **5.2. Saran**

Dalam pengerjaan proyek yang bertopik serupa kedepannya, penulis memberikan beberapa saran yaitu:

- a. Mengembangkan aplikasi ini menjadi sebuah *web application* yang dapat diakses melalui jaringan internet.
- b. Meningkatkan *user interface* (UI) yang ada dari GUI menjadi UI yang lebih modern.
- c. Menambahkan fitur-fitur baru pada jejaring sosial media yang telah dibuat.

#### **5.3. Refleksi dan komentar**

Penulis berterima kasih kepada dosen dan asisten IF2211 Strategi Algoritma karena telah merancang tugas besar 2 ini secara detail dan aplikatif. Melalui pengerjaan tugas besar 2 IF2211 Strategi Algoritma ini, penulis memperoleh banyak hal baik dari segi akademik maupun non-akademik. Penulis dapat memahami materi tentang algoritma BFS dan DFS serta aplikasinya secara lebih mendalam dan jelas. Melalui implementasi algoritma-algoritma secara nyata dalam program, hal ini membantu penulis dalam memahami materi yang ada. Selain itu, penulis juga belajar untuk bekerja sama dalam tim terutama dalam mengerjakan sebuah proyek serta dapat melatih skill manajemen waktu. Seluruh hasil yang telah direfleksikan ini, diharapkan dapat membantu penulis untuk berkembang ke arah yang lebih baik lagi.

## DAFTAR PUSTAKA

Microsoft. N.D. *C# Documentation*. Dilansir dari [www.docs.microsoft.com/en-us/dotnet/csharp/](http://www.docs.microsoft.com/en-us/dotnet/csharp/) pada 18 Maret 2021.

N.N. 2021. *Pengaplikasian Algoritma BFS dan DFS dalam Fitur People You May Know Jejaring Sosial Facebook*. Bandung: Institut Teknologi Bandung.

Munir, R. 2021. *BFS - DFS (2021)*. Bandung: Institut Teknologi Bandung.