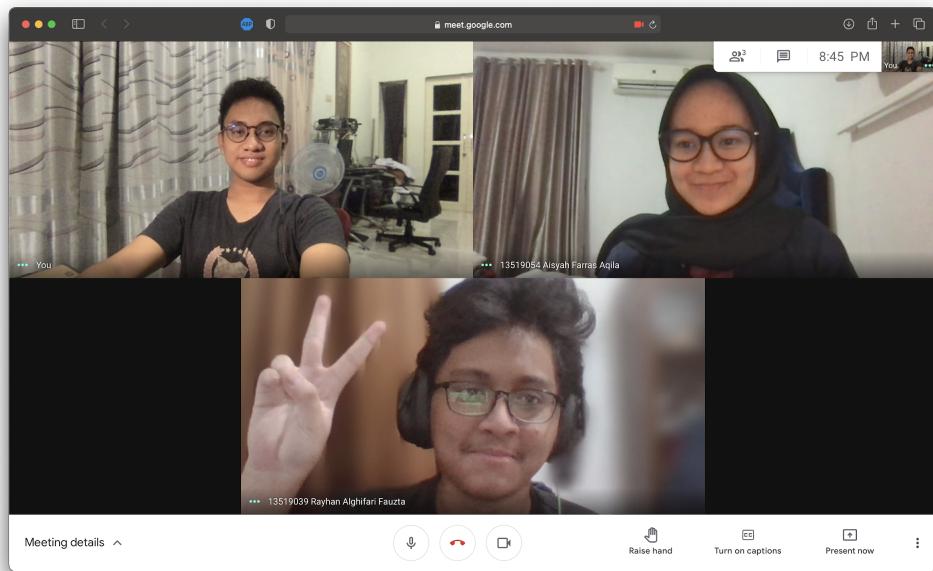


Tugas Besar 2

IF2211 Strategi Algoritma

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



Disusun Oleh :

Kelompok 51

13519039 Rayhan Alghifari Fauzta

13519054 Aisyah Farras Aqila

13519080 Daru Bagus Dananjaya

**PROGRAM STUDI SARJANA TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2021**

Daftar Isi

Daftar Isi	2
Deskripsi Tugas	3
Landasan Teori	6
Dasar Teori	6
Graph Traversal	6
Breadth First Search	7
Depth First Search	7
C# Desktop Application Development	7
Analisis Pemecahan Masalah	7
Langkah-Langkah Pemecahan Masalah	7
Mapping Persoalan menjadi Elemen-Elemen Algoritma BFS dan DFS	8
Contoh Ilustrasi Kasus	8
Explore Friends - DFS	9
Explore Friends - BFS	10
Friend Recommendation	10
Implementasi dan Pengujian	11
Implementasi Program	11
Struktur Data Program	13
Tata Cara Penggunaan Program	15
Hasil Pengujian	16
Analisis Desain Solusi	18
Kesimpulan, Saran, dan Refleksi	19
Kesimpulan	19
Saran	19
Refleksi	19
Referensi	20

1. 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.

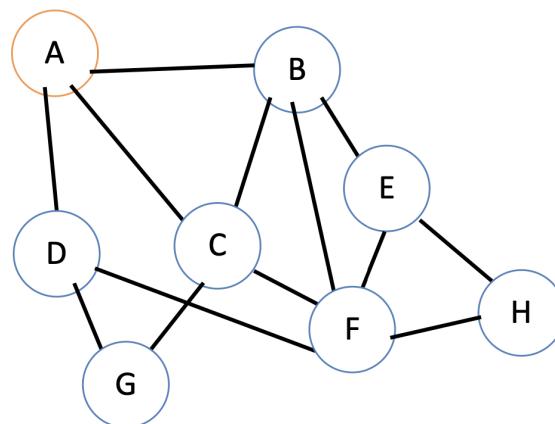
Contoh Input dan Output Program

Contoh 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
```

Gambar 1. Contoh input berkas file eksternal

Visualisasi graf pertemanan yang dihasilkan dari file eksternal:



Gambar 2. 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 3. Hasil output yang diharapkan untuk rekomendasi akun A

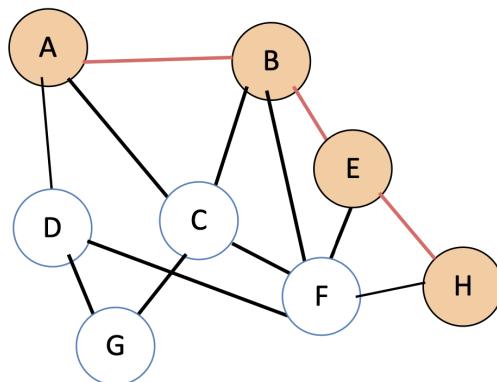
Untuk **fitur explore friends**, misalnya pengguna ingin mengetahui seberapa jauh jarak antara akun A dan H serta bagaimana jalur agar kedua akun bisa terhubung. Berikut output graf dengan penelusuran BFS yang dihasilkan.

Berikut output yang diharapkan untuk penelusuran menggunakan BFS.

Nama akun: A dan H
 2nd-degree connection
 $A \rightarrow B \rightarrow E \rightarrow H$

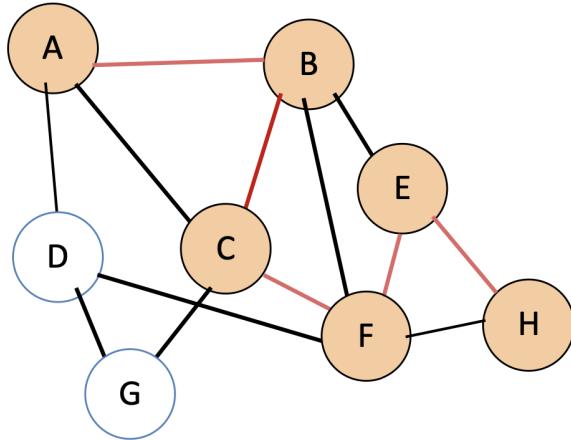
Gambar 4. Hasil output akun Nth degree connection akun A dan H menggunakan BFS

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 5. 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 6. 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 7. Hasil output tidak ada jalur koneksi antara A dan J

2. Landasan Teori

2.1. Dasar Teori

2.1.1. Graph Traversal

Traversal pada suatu *graph* dilakukan dengan menelusuri seluruh simpul dan sisi secara sistematis. Algoritma traversal ini ada dua jenis, yaitu Breadth First Search dan Depth First Search. Kedua algoritma ini digunakan pada berbagai aplikasi yang melibatkan struktur *graph* seperti *artificial intelligence* dan *operations research*. Selain itu BFS dan DFS juga sangat dibutuhkan dalam pemeriksaan efisiensi dari *property* sebuah *graph* seperti keterhubungan dan keberadaan siklus.

2.1.2. Breadth First Search

Breadth First Search dilakukan dengan melakukan traversal secara konsentris. Hal ini dilakukan dengan menelusuri semua simpul yang bertetanggaan dengan simpul awal, lalu semua simpul yang belum dikunjungi yang berjarak dua sisi dari simpul awal, dan seterusnya hingga semua simpul yang dapat terhubung dengan simpul awal dikunjungi.

Breadth First Search lebih mudah dilakukan dengan menggunakan *queue*. *Queue* diinisialisasi dengan simpul awal yang ditandai sudah dikunjungi. Pada setiap iterasi, algoritma mengidentifikasi semua simpul yang belum dikunjungi dan bertetanggaan dengan simpul pada antrian pertama pada *queue*, mengidentifikasi simpul-simpul tersebut dengan status sudah dikunjungi, dan melakukan *enqueue*. Lalu simpul pada antrian pertama di-*dequeue*.

2.1.3. Depth First Search

Depth First Search dilakukan dengan melakukan traversal yang bermula di simpul awal. Pada setiap iterasi, algoritma mengunjungi sebuah simpul yang bertetanggaan dengan simpul awal (jika terdapat beberapa simpul, pemilihan simpul disesuaikan dengan struktur data dari *graph*). Lalu penelusuran dilanjutkan pada simpul yang bertetanggaan dengan simpul tersebut hingga mencapai *dead end*. Pada saat tersebut, algoritma melakukan *backtracking* sebanyak satu *edge* ke simpul sebelumnya dan menelusuri kembali simpul yang belum dikunjungi.

2.2. C# Desktop Application Development

C# adalah bahasa pemrograman multiparadigma buatan Microsoft yang dijalankan pada framework .NET. Aplikasi *desktop*, *web app*, aplikasi *mobile*, maupun *game* adalah contoh perangkat lunak yang dapat dibuat dengan C#. Pembuatan suatu program C# umumnya menggunakan *integrated development environment* (IDE) Visual Studio pada sistem operasi yang sudah terinstall framework .NET seperti .NET Core. Seperti yang telah disebutkan sebelumnya, C# dapat digunakan untuk membuat aplikasi desktop dengan *graphical user interface* (GUI) seperti pada tugas besar kali ini. Pembuatan aplikasi GUI dilakukan dengan pustaka Windows Forms sebagai bagian dari framework .NET. Salah satu kekurangan Windows Forms ini adalah hanya dapat berjalan di sistem operasi Microsoft Windows, sehingga di sistem operasi UNIX-like seperti macOS membutuhkan *workaround* yang sulit seperti menggunakan framework Mono. Pustaka lain yang digunakan adalah Microsoft Automatic Graph Layout (MSAGL) untuk membuat dan menampilkan graph pada aplikasi.

3. Analisis Pemecahan Masalah

3.1. Langkah-Langkah Pemecahan Masalah

Pemecahan masalah dimulai dengan mengabstraksi deskripsi masalah di spesifikasi menjadi bagian yang lebih spesifik. Abstraksi masalah yang didapat adalah dibutuhkan suatu program berbasis GUI yang menerima input graf pertemanan dan kemudian menampilkan rekomendasi teman atau jalur pertemanan. Masalah ini dapat diselesaikan dengan algoritma BFS atau DFS. Langkah-langkah penyelesaiannya secara garis besar dimulai dengan merepresentasikan input pengguna dalam bentuk graf lengkap dengan node, edge, dan komponen lainnya. Kemudian kasus dibedakan menjadi dua yaitu *explore friend* dan *friend recommendation*. Kasus friend recommendation tidak harus dilakukan penerapan algoritma BFS atau DFS karena hanya meminta

satu akun yang akan dicari rekomendasi teman berdasarkan temannya saat ini. Algoritma BFS dan DFS baru akan diterapkan di kasus explore friends. Kedua penyelesaian ini juga akan dilengkapi dengan visualisasi graf pertemanan untuk memudahkan pemahaman solusi.

3.2. Mapping Persoalan menjadi Elemen-Elemen Algoritma BFS dan DFS

Pada aplikasi media sosial, setiap akun dapat memiliki koneksi dengan sesama akun yang dapat direpresentasikan dalam sebuah *graph*. Dalam kasus ini, jika akun A berteman dengan akun B, maka B juga berteman dengan A. Berdasarkan kasus tersebut, representasi yang sesuai untuk jaringan pertemanan pada aplikasi media sosial ini adalah *undirected* dan *unweighted graph*. Masing-masing akun akan direpresentasikan oleh sebuah *node* dan jaringan pertemanan yang dimiliki antara dua buah akun direpresentasikan oleh sebuah *edge*. Dengan demikian, proses pencarian teman dan rekomendasi teman dapat dilakukan dengan menggunakan algoritma DFS maupun BFS.

Dalam Algoritma DFS, iterasi yang dilakukan akan membangkitkan semua node yang terhubung langsung dengan node yang sedang dievaluasi menggunakan prinsip *First-In-Last-Out*. Sedangkan dalam algoritma BFS, setiap iterasi yang dilakukan akan membangkitkan semua node yang terhubung langsung dengan node yang sedang dievaluasi menggunakan prinsip *First-In-First-Out*. Oleh karena itu, dalam implementasi algoritma BFS, digunakan *queue*.

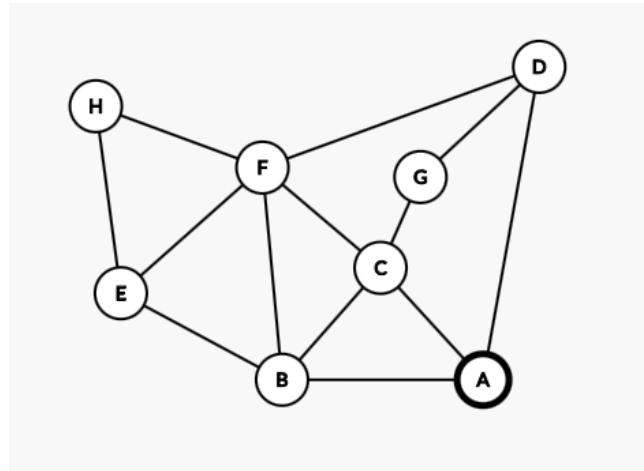
Pada penerapannya, masing-masing algoritma pencarian membutuhkan atribut tambahan yang digunakan untuk menyimpan status kunjungan *node* serta parent dari *node*. Untuk algoritma DFS, dibutuhkan tiga status kunjungan, yaitu belum dikunjungi, sedang dievaluasi, dan sudah dievaluasi. Sedangkan untuk algoritma BFS, hanya dibutuhkan dua buah status kunjungan, yaitu sudah dikunjungi atau belum dikunjungi karena algoritma BFS tidak membutuhkan *backtracking*. atribut *parent node* pada kedua algoritma berfungsi sebagai bantuan untuk mendapatkan path dari suatu starting *node* menuju *node* tujuan.

Fitur *Explore Friends* pada program ini dapat diimplementasikan menggunakan algoritma DFS maupun BFS. Hasil akhir dari fitur ini adalah sebuah path yang menghubungkan antara dua buah akun.

Fitur *friend recommendation* pada program ini akan diselesaikan menggunakan algoritma BFS untuk mencari jarak terpendek antara akun yang akan dicari dengan akun lainnya, kemudian untuk setiap koneksi yang dimiliki oleh akun, dievaluasi menggunakan bantuan DLS dengan kedalaman 2 untuk mengetahui apakah akun tersebut pantas untuk direkomendasikan kepada akun awal.

3.3. Contoh Ilustrasi Kasus

Pada contoh ilustrasi kasus, akan digunakan graph tidak berarah dan tidak berbobot seperti pada gambar di bawah untuk diuji.

**Gambar 8. Ilustrasi Graf****3.3.1. Explore Friends - DFS**

Pada kasus ini, akan dilakukan proses pencarian jalur pertemanan dari akun A ke akun H menggunakan algoritma DFS, maka proses traversal grafnya adalah

Simpul Eksplan	Simpul Hidup
A	B[A], C[A], D[A]
B[A]	C[A], D[A], C[AB], E[AB], F[AB]
C[AB]	D[A], E[AB], F[AB], F[ABC], G[ABC]
F[ABC]	C[A], D[A], E[AB], F[AB], G[ABC], D[ABCF], E[ABCF], H[ABCF]
D[ABCF]	C[A], D[A], E[AB], F[AB], G[ABC], E[ABCF], H[ABCF], G[ABCFD]
G[ABCFD]	C[A], D[A], E[AB], F[AB], G[ABC], E[ABCF], H[ABCF]
E[ABCF]	C[A], D[A], E[AB], F[AB], G[ABC], H[ABCF], H[ABCFE]
H[ABCFE]	Solusi ketemu
Path hasil: A → B → C → F → E → H	

3.3.2. Explore Friends - BFS

Pada kasus ini, akan dilakukan proses pencarian jalur pertemanan dari akun A ke akun H menggunakan algoritma BFS, maka proses traversal grafnya adalah

Simpul Ekspan	Simpul Hidup
A	B[A], C[A], D[A]
B	C[A], D[A], C[AB], E[AB], F[AB]
C	D[A], C[AB], E[AB], F[AB], F[AC], G[AC]
D	C[AB], E[AB], F[AB], F[AC], G[AC], F[AD], G[AD]
C	E[AB], F[AB], F[AC], G[AC], F[AD], G[AD], F[ABC], G[ABC]
E	F[AB], F[AC], G[AC], F[AD], G[AD], F[ABC], G[ABC], F[ABE], H[ABE]
F	F[AB], F[AC], G[AC], F[AD], G[AD], F[ABC], G[ABC], H[ABE]
H	Solusi ketemu
Path hasil : A → B → E → H	

Berdasarkan tabel di atas, dapat dilihat bahwa setiap simpul yang dibangkitkan pada masing-masing iterasi akan diposisikan pada posisi terakhir simpul hidup, hal tersebut menunjukkan penggunaan prinsip FIFO pada BFS.

3.3.3. Friend Recommendation

Friend recommendation dapat digunakan untuk mengetahui akun mana saja yang dapat direkomendasikan kepada akun A, pada fitur ini, dilakukan kombinasi antara algoritma BFS dan DLS. Pada tahap pertama, algoritma BFS akan melakukan traversal ke seluruh *node* dalam graf dan menyimpan levelnya. Kemudian, untuk setiap *node* yang bertetangga langsung dengan *node* A, akan dilakukan pencarian secara DLS dengan batasan level 2 untuk menyimpan daftar teman yang dapat direkomendasikan ke A. Jika pada saat DLS ditemukan akun yang dapat direkomendasikan ke akun A, maka akun tersebut akan disimpan sebagai *mutual friend*, sampai akhirnya ditampilkan terurut berdasarkan *mutual friend* terbanyak.

```

Daftar rekomendasi teman untuk akun A :
Nama Akun : F
3 mutual friend(s) :
B
C
D

Nama Akun : G
2 mutual friend(s) :
C
D

Nama Akun : E
1 mutual friend(s) :
B

```

Gambar 9. Hasil visualisasi friend recommendation

4. Implementasi dan Pengujian

4.1. Implementasi Program

Program Main

KAMUS

```

Graph: Microsoft MSAGL Graph
startAccount: string
destAccount: string
fileContent: array of string
networkGraph: Graph
bfs: BreadthFirstSearch
dfs: DepthFirstSearch
friendRec: MutualFriends
people: List of string
bfsPath: List of node
dfsPath: List of node
openFileDialog: OpenFileDialog

```

ALGORITMA

```

InitializeComponent() {inisialisasi komponen di forms}
buttonBrowse.Click()
openFileDialog <- openFileDialog.ShowDialog() {pilih file txt}
if (openFileDialog = OK) then
    filePath <- chosen file
    fileContent <- ReadAllLines(filePath)
    fillComboBox(fileContent) {isi combo box choose account}
    drawGraph(fileContent) {buat graph MSAGL}
else
    catch(IOException)

startAccount < ""
destAccount < ""
networkGraph <- Graph(startAccount)
if (destAccount <> Null) then
    if (radioButtonDFS.Checked) then
        DFS()
    else if (radioButtonBFS.Checked) then
        BFS()

```

MutualFriends()

Procedure DFS(input startNode : Node)

KAMUS LOKAL

TrailNode : SuccessorNode

ALGORITMA

```

visited[startNode] ← 1 {startNode visited, set the visited value to 1}
Interval[startNode][0] ← t {time when the startNode visited for the
first time}

result.Add(startNode) {add startNode to result list}
t ← t + 1 {increment the time}

TrailNode ← startNode.Tail {set TrailNode with startNode's
successorNode}
while (TrailNode ≠ null) do
    if (visited[TrailNode.Succ] = 0) then
        parentNode[TrailNode.Succ] ← startNode
        DFS(TrailNode.Succ)
    {end if}
    TrailNode ← TrailNode.Next
{end while}

visited[startNode] ← 2      {Backtrack}
interval[startNode][1] ← t  {time when the startNode visited for the
second time}
t ← t +1

```

Procedure BFS(input startNode : Node)

KAMUS LOKAL

V : Node

Child : SuccessorNode

ALGORITMA

```

level[startNode] ← 0          {set the level of startNode with 0}
visited[startNode] ← True    {set the startNode visited value with
true}
q.Enqueue(startNode)

while (not q.IsEmpty()) do
    V ← q.Dequeue()
    result.Add(V)

    Child ← V.Tail
    while (Child ≠ null) do      {traversal all of V's neighbors}
        if (not visited[Child.Succ]) then
            visited[Child.Succ] ← True
            parentNode[Child.Succ] ← V
            level[Child.Succ] ← level[V] + 1
            q.Enqueue(Child.Succ)
        {end if}
        Child ← Child.Next
{end while}

```

Procedure FindMutualFriend**KAMUS LOKAL**

```

mutual          : List<Node>
V, N           : Node
NChild, TrailNode : SuccessorNode

```

ALGORITMA

```

traversal (NodeLevel in bfs_result.Level)
    if (NodeLevel.Value = 2) then
        V <- bfs_result.startNode
        TrailNode <- V.Trail

        while (TrailNode ≠ null) do
            N <- TrailNode.Succ
            NChild <- N.Trail
            status <- false

            while (not status and NChild ≠ Null) do
                if (NodeLevel.Key.Name = NChild.Succ.Name) then
                    mutual.Add(N)
                    status <- True
                {end if}
                N <- N.Next
            {end while}
            TrailNode <- TrailNode.Next
        {end while}
        result.Add(NodeLevel.Key, mutual)
    {end if}
{end traversal}

```

4.2. Struktur Data Program**NODE**

```

class Node :
< numPred   : integer
  name      : string
  next      : Node
  trail     : SuccessorNode >

```

SUCCESSOR NODE

```

class SuccessorNode :
< succ      : Node
  next      : SuccessorNode >

```

GRAPH

```

class Graph :
< head : Node >

```

QUEUE

```
class EQueue : {Element Queue}
< info      : Node
  Next      : Equeue >
```

```
class Queue :
< head      : Equeue
  tail      : Equeue >
```

MUTUAL FRIENDS

```
class MutualFriend :
< g          : Graph
  startNode : string
  Result    : Dictionary<Node, List<Node>> >
```

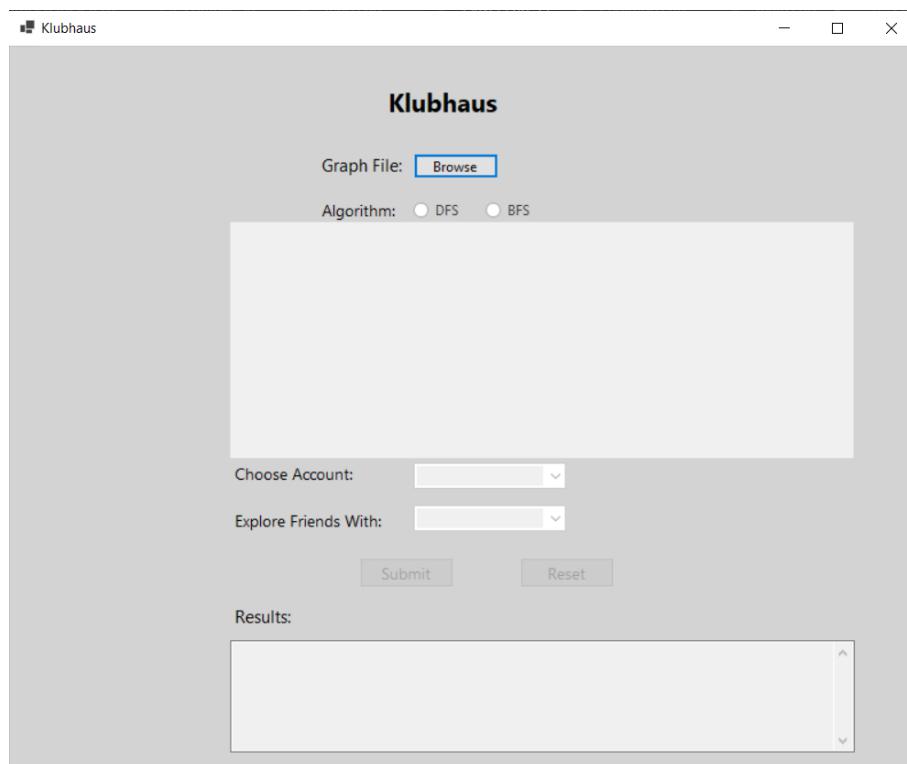
BFS

```
class BFS :
< startNode : Node
  g          : Graph
  q          : Queue
  result    : List<Node>
  visited   : Dictionary<Node, boolean>
  level     : Dictionary<Node, integer>
  parentNode: Dictionary<Node, Node> >
```

DFS

```
class DFS :
< startNode : Node
  g          : Graph
  t          : integer
  result    : List<Node>
  visited   : Dictionary<Node, integer>
  interval  : Dictionary<Node, array of integer>
  parentNode: Dictionary<Node, Node> >
```

4.3. Tata Cara Penggunaan Program



Gambar 10. Tampilan layar awal program

Tata cara penggunaan program adalah sebagai berikut:

1. Pertama, pengguna menekan tombol *browse* untuk menginput file .txt berisi akun-akun dan koneksinya dengan format file sesuai spesifikasi tugas besar. Pengguna tidak dapat melakukan hal lain di program sebelum menginput file .txt ini.
2. Selanjutnya program akan menampilkan *graph* awal di panel bagian tengah dan pengguna akan memilih akun yang ingin ditelusuri.
3. Jika pengguna hanya memilih satu akun (*choose account*), maka program otomatis akan menampilkan *friend recommendation* saja. Jika pengguna juga memilih akun yang akan di *explore friends*-nya serta algoritma yang digunakan, maka program akan menelusuri jalur pertemanan sesuai dengan algoritma terpilih.
4. Setelah memilih akun, pengguna menekan tombol *submit*.
5. Program menampilkan hasil penelusuran di text box.
6. Pengguna dapat menekan tombol *submit* untuk memulai pencarian baru.

4.4. Hasil Pengujian

No	Test Case	Friend Recommendation	BFS	DFS
1	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	<p>Daftar rekomendasi teman untuk akun G Nama Akun : A 2 mutual friend(s) : C D</p> <p>Nama Akun : F 2 mutual friend(s) : C D</p> <p>Nama Akun : B 1 mutual friend(s) : C</p>	Nama akun: A dan H 3th degree connection A > B > E > H	Nama akun: A dan H 9th degree connection A > B > C > F > E > H
2	15 A D A E A F B C B E B G C D C J D H E I F G G H H I I J J F	<p>Daftar rekomendasi teman untuk akun C Nama Akun : A 1 mutual friend(s) : D</p> <p>Nama Akun : E 1 mutual friend(s) : B</p> <p>Nama Akun : F 1 mutual friend(s) : J</p> <p>Nama Akun : G 1 mutual friend(s) : B</p> <p>Nama Akun : H 1 mutual friend(s) : D</p> <p>Nama Akun : I 1 mutual friend(s) : J</p>	Nama akun: B dan H 2th degree connection B > G > H	Nama akun: B dan H 6th degree connection B > C > D > A > E > I > H

3	15 A B A C A D B E B H D E C F F D F G G A H I H F B C C D I F	Daftar rekomendasi teman untuk akun A Nama Akun : F 3 mutual friend(s) : C D G Nama Akun : E 2 mutual friend(s) : B D Nama Akun : H 1 mutual friend(s) : B	Nama akun: F dan E 4th degree connection F > C > A > B > E	Nama akun: F dan E 2th degree connection F > D > E
4	14 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 J I	Daftar rekomendasi teman untuk akun E Nama Akun : C 2 mutual friend(s) : B F Nama Akun : A 1 mutual friend(s) : B Nama Akun : D 1 mutual friend(s) : F	Nama akun: A dan I Tidak ada jalur koneksi yang tersedia Anda harus memulai koneksi baru itu sendiri.	Nama akun: A dan J Tidak ada jalur koneksi yang tersedia Anda harus memulai koneksi baru itu sendiri.

5 19 A B A C A D A E B F B G B H D I D J G K G L L M M N N O N P O Q O R P S S T	Daftar rekomendasi teman untuk akun G Nama Akun : A 1 mutual friend(s) : B Nama Akun : F 1 mutual friend(s) : B Nama Akun : H 1 mutual friend(s) : B Nama Akun : M 1 mutual friend(s) : L	Nama akun: A dan T 8th degree connection A > B > G > L > M > N > P > S > T	Nama akun: A dan T 18th degree connection A > B > G > L > M > N > P > S > T
---	---	--	---

4.5. Analisis Desain Solusi

Berdasarkan hasil uji coba atas program yang dibuat, persoalan sosial *networking* dapat direpresentasikan dengan baik menggunakan sebuah graf tak berarah dan tak berbobot karena dengan mempresentasikannya sebagai graf, fitur-fitur seperti friend recommendation dan *explore friends* dapat diimplementasikan dengan baik pula dengan memanfaatkan algoritma traversal graf, yaitu BFS dan DFS.

Pada dasarnya, *explore friends* yang dilakukan pada program ini memiliki cara kerja yang sama seperti pada path-finding problem sebuah graf yang diketahui titik awal dan titik akhirnya. Selama titik awal dan titik akhir terhubung dalam sebuah graf, maka jalur dari titik awal dan titik akhir akan selalu dapat ditemukan, baik menggunakan algoritma DFS maupun BFS. Namun, pada program kami ini, kami menemukan bahwa jalur yang dihasilkan masing-masing algoritma akan berbeda. Pada proses implementasi program ini, kami menemukan bahwa untuk persoalan pathfinding pada graf tak berarah dan tak berbobot, algoritma BFS akan selalu menghasilkan path yang optimal, namun tidak dengan algoritma DFS.

Pada fitur friend recommendation, program melakukan list seluruh node dalam graf yang memiliki jarak dengan node asal yaitu dua, oleh karena itu, fitur ini akan menghasilkan yang sama tepatnya ketika diimplementasikan menggunakan algoritma DFS maupun algoritma BFS.

Dalam kasus representasi graf menggunakan adjacency list, algoritma BFS dan DFS memiliki kompleksitas waktu yang sama yaitu $O(V + E)$, dengan V adalah jumlah simpul dan E adalah jumlah sisi. Namun masing-masing algoritma memiliki keunggulannya masing-masing dalam menangani masalah traversal graf. Algoritma BFS akan memiliki performa yang lebih baik ketika jarak antara simpul awal dan simpul akhir relatif dekat, sedangkan algoritma DFS akan memiliki performa yang lebih baik ketika sebuah graf memiliki kedalaman yang besar. Karena persoalan yang dihadapi dalam permasalahan media sosial ini tidak terlalu besar, maka dapat disimpulkan bahwa penggunaan algoritma BFS akan lebih optimal, baik dari segi waktu maupun memory.

5. Kesimpulan, Saran, dan Refleksi

5.1. Kesimpulan

Berdasarkan proses pembuatan aplikasi, penulis dapat menarik kesimpulan, yaitu :

1. Pembuatan aplikasi windows dengan *tools* Visual Studio .NET hanya dapat dilakukan pada sistem operasi Windows.
2. Pada penelusuran graf tidak berbobot dan tidak berarah, algoritma BFS selalu dapat memberikan solusi optimal, sedangkan DFS tidak.
3. Algoritma DFS dan BFS dapat diimplementasikan ke dalam banyak kasus di kehidupan sehari-hari, terutama yang dapat direpresentasikan melalui graf, salah satunya adalah aplikasi jaringan pertemanan.
4. Algoritma BFS akan memiliki performa lebih baik ketika graf memiliki kedalaman yang relatif rendah.
5. Algoritma DFS akan memiliki performa yang lebih baik ketika graf memiliki kedalaman yang besar.

5.2. Saran

Program dapat disempurnakan lagi dengan memperbaiki algoritma beserta cara-cara menampilkan hasilnya. Dari sisi antarmuka, program juga dapat diberi sentuhan lebih agar terlihat lebih menarik dan interaktif seperti pemilihan warna atau modularisasi tampilan. Satu hal yang juga patut disorot adalah *versatility* program karena pembuatannya yang menggunakan Windows Forms. Aplikasi Windows Forms hanya dapat berjalan di Windows secara native. Oleh karena itu, sebaiknya dicari alternatif pustaka lainnya yang *cross-platform* untuk memudahkan programmer dan pengguna.

5.3. Refleksi

Berdasarkan proses pembuatan aplikasi, ada beberapa hal yang penulis temukan dan diharapkan dapat diperbaiki di kemudian hari, yaitu:

1. Seluruh anggota kelompok sebaiknya memiliki sistem operasi atau perangkat yang kompatibel dengan spek tugas. Hal ini guna memudahkan proses testing program secara paralel dan tidak dibebankan ke satu atau dua orang saja.
2. Karena tidak semua anggota kelompok memiliki perangkat yang kompatibel dengan spek tugas, maka proses penggerjaan program ini menjadi sedikit terhambat.

6. Referensi

<https://web.stanford.edu/class/archive/cs/cs161/cs161.1166/lectures/lecture10.pdf>

<https://www.geeksforgeeks.org/bfs-using-vectors-queue-per-algorithm-clrs/>

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

https://www2.seas.gwu.edu/~bell/csci212/graph_search.pdf