TUGAS BESAR II

Algoritma DFS dan BFS Pada Jejaring Sosial Media

IF2211 – Strategi Algoritma


Oleh

Kelompok 67 : gabisabahasaCsharp

Kahfi Soobhan Zulkifli 13519012

Ariya Adinatha 13519048

Rehagana Kevin Christian Sembiring 13519117

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

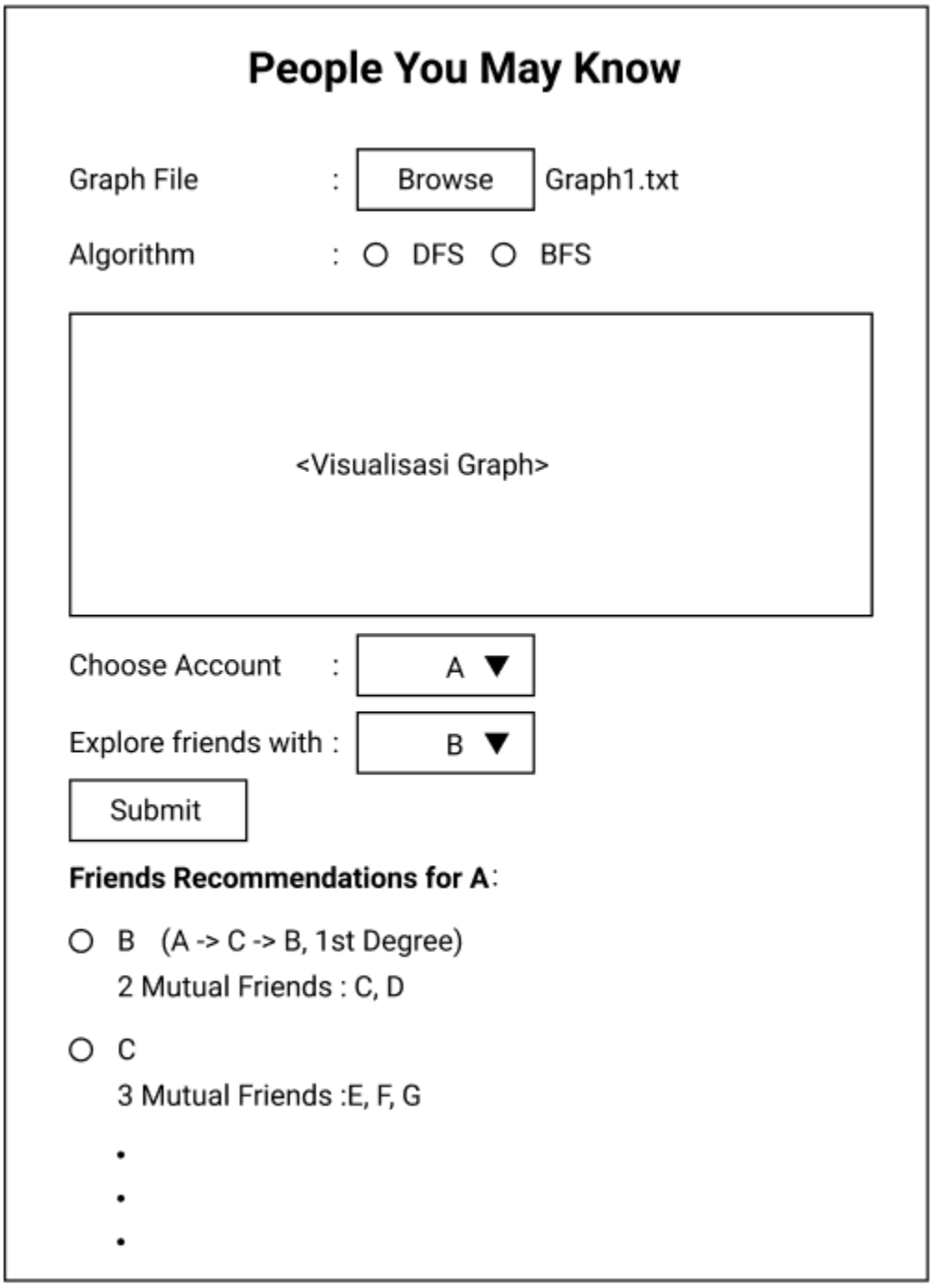
INSTITUT TEKNOLOGI BANDUNG

2021

BAB I
DESKRIPSI TUGAS

Spesifikasi Program:

Aplikasi yang akan dibangun dibuat berbasis GUI. Berikut ini adalah contoh tampilan dari aplikasi GUI yang akan 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
        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 antara dua 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 aku 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 Trasversal

Graph trasversal atau disebut juga dengan *graph search* merupakan proses dari mengunjungi dan mengecheck setiap vertex yang terdapat di dalam *graph*. Proses traversal yang dilakukan berdasarkan *vertices* yang akan dikunjungi secara sistematik. Dalam pencarian solusi graf merepresentasikan persoalan yang akan diselesaikan dan traversal graf merupakan proses pencarian solusi. Terdapat 2 algoritma pencarian solusi yaitu tanpa informasi atau dengan informasi. Pencarian tanpa informasi dilakukan tanpa informasi tambahan contoh algoritmanya adalah *Depth First Search, Breadth First Search, Depth limited Search, Iterative Deepening Search, Uniform Cost Search*. Kemudian pencarian dengan informasi dibasiskan dengan heuristik. Tanpa mengetahui goal stat diharapkan pencarian “lebih menjanjikan” daripada yang lain. Contoh dari pencarian dengan informasi adalah *Best First Search dan A**. Terdapat dua jenis graf dalam proses pencarian, yaitu graf statis dan graf dinamis. Dimana graf statis sudah terbentuk sebelum proses pencarian yang dilakukan, sedangkan graf dinamis terbentuk saat proses pencarian dilakukan.

2.2. Breadth First Search

Breadth First Search atau pencarian melebar merupakan pencarian traversal yang dimulai dari simpul secara melebar. Dimana pencarian ini akan mengunjungi semua simpul yang ada dan semua simpul yang bertetangga dengan simpul tersebut terlebih dahulu. Kemudian pencarian dilakukan dengan mengunjungi simpul yang belum dikunjungi dan bertetangga dengan simpul simpul yang tadi dikunjungi, demikian seterusnya sehingga ditemukan informasi yang diinginkan atau graf sudah habis dikunjungi.

2.3. Depth First Search

Depth First Search atau pencarian mendalam merupakan pencarian traversal yang dimulai dari simpul secara mendalam. Pencarian ini bekerja dengan mengunjungi yang

bertetangga dengan simpul awal dan kemudian mengunjungi simpul tetangganya lagi sehingga semua simpul yang bertetangga dengannya telah dikunjungi, kemudian pencarian runut-balik (*backtrack*) ke simpul terakhir yang dikunjungi sebelumnya dan mengunjungi simpul terakhir yang belum dikunjungi sebelumnya. Pencarian berakhir bila tidak ada lagi simpul yang belum dikunjungi yang dapat dicapai dari simpul yang telah dikunjungi.

2.4. C# Desktop Application Development

C# merupakan sebuah bahasa pemrograman yang berorientasi objek yang dikembangkan oleh Microsoft sebagai bagian dari inisiatif kerangka .NET Framework. Bahasa yang dibuat ini berbasi bahasa C++ yang telah mempengaruhi aspek aspek maupun fitur bahasa yang terdapat pada bahasa pemrograman lainnya. Bahasa pemrograman C# dibuat sebagai bahasa pemrograman yang bersifat *general-purpose*, berorientasi objek, modern, dan sederhana. Pada C# disediakan IDE Visual Studio yang mendukung proses *software development*. Dari Visual Studio disediakan banyak pilihan aplikasi yang dapat dibuat seperti *windows form application*, *console application, web application.*

BAB III
ANALISIS PEMECAHAN MASALAH

3.1. Langkah - Langkah Pemecahan Masalah

Terdapat 3 bagian masalah yaitu pencarian teman berdasarkan BFS, pencarian teman berdasarkan DFS, dan rekomendasi teman.

a. Pemecahan Masalah dengan Algoritma BFS
   i. Menerima masukan berupa simpul awal dan tujuan
   ii. Mengecek apakah simpul awal dan tujuan terdapat pada graph
   iii. Jika iya masukkan simpul awal ke list
   iv. Masukkan list ke antrian
   v. Ambil elemen pertama antrian
   vi. Cek semua simpul tetangga dengan elemen terakhir pada elemen pertama antrian
   vii. Masukkan semuanya ke dalam antrian
   viii. Bila tetangga sama dengan simpul tujuan kembalikan list

b. Pemecahan Masalah dengan Algoritma DFS
   i. Menerima masukan berupa simpul awal dan tujuan
   ii. Mengecek apakah simpul awal dan tujuan terdapat pada graph
   iii. Jika iya masukkan simpul ke stack
   iv. Lalu cek simpul tetangga pada elemen paling atas stack
   v. Jika tidak ada lagi tetangga pop
   vi. Ambil simpul tetangga yang belum dikunjungi
   vii. Apabila ditemukan simpul tujuan salin ke list

viii. Kembalikan list

c. Rekomendasi teman
   i. Menerima masukan berupa akun yang ingin dicari ke dalam bentuk string
   ii. Mencari semua node berdasarkan banyak sisi
   iii. Memasukkan node tersebut ke dalam list
   iv. Mencari node yang bersisian dengan node yang ingin dicari *mutual friends*-nya
   v. Menambahkan sisi node tersebut ke dalam list
   vi. Mencetak isi dari sisi node tersebut

3.2. Proses Mapping Persoalan

Elemen - elemen yang digunakan dalam algoritma BFS dan DFS.

1. Akun pertama sebagai simpul awal
2. Akun kedua sebagai simpul tujuan
3. Teman sebagai sisi dari graf

3.3. Contoh Ilustrasi Kasus Lain

Selain untuk mencari relasi antar akun, algoritma BFS dan DFS juga dapat digunakan untuk melakukan penelusuran direktori, direktori direktori digambarkan sebagai graf dan ditelusuri dengan menggunakan algoritma BFS ataupun DFS hingga menemukan *file* atau *folder* yang diinginkan. Proses *spidering* pada pencarian *query website* juga menggunakan algoritma BFS dan DFS. *Website* digambarkan sebagai simpul dan ditelusuri setiap simpulnya sehingga habis dan menampilkan hasil yang didapatkan.

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1. Implementasi Program

```
function ExploreFriendsBFS(input a,b : string, output
test:listofstring)
Deklarasi
Antrian    : Queueof(ListofString)
Size       : int
Found      : bool
Dikunjungi : Arrayofbool[Size]
Tmp        : listofstring
Last       : string
I          : int
Node1      : string
Node2      : string
Tmp1       : listofstring
```

Algoritma

```
Size ←  banyakNodeGraph
{inisiasi semua nilai false pada list dikunjungi}
For (i = 0 until size)
     Dikunjungi[i] ← false

test.Add(a) {masukkan nilai a ke dalam test}
Antrian.Enqueue(a) {memasukkan list test ke dalam antrian}
Dikunjungi[Posisi a pada list of vertices] ← true

If (a not in simpul graph or b not in simpul graph)
     → null
Else {a dan b di dalam simpul}
     While (antrian.count > 0 and found = false)
          Tmp = antrian.dequeue
          Last = Tmp[element terakhir]
          I ← 0
          If (Last = b)
               Test ← Tmp
               Found ← true
          Else
               While (i < sizeofArrayEdges)
                    Node1 = getNode1 dari Edge ke-i
                    Node2 = getNode2 dari Edge ke-i
                    {copy tmp ke tmp1}
                    For (y = 0 until sizeofTemp)
                         tmp1.Add(tmp[y])
                    If (Node1 = Last and not dikunjungi[posisi
                    node2 pada vertice])
                         tmp1.Add(Node2)
                         dikunjungi[posisi node2 pada vertice])
                         ← true
                         antrian.Enqueue(tmp1)
                    I ← I+1
     {jalur friend ditemukan}
     If (Found) then
          → Test
     {jalur friend tidak ditemukan}
     Else
          → Return null
```

```
function ExploreFriendsDFS(input a,b: string, output result :
listofstring)
Deklarasi
Size       : int
Dikunjungi : Arrayofbool[Size]
Test       : StackofString
Found      : Boolean
I          : int
Found1     : Boolean
Count      : int

Algoritma
Size ←  banyakNodeGraph
{inisiasi semua nilai false pada list dikunjungi}
For (i = 0 until size)
     Dikunjungi[i] ← false

Test.Push(a)
Dikunjungi[Posisi a pada list of vertices] ← true

If (a not in simpul graph or b not in simpul graph)
     → result
Else
     While (sizeofTest > 0 and Found = false)
          I ← 0
          Found1 ← false
          While (i < sizeofEdgeList and Found1 = false)
               Node1 = getNode1 dari Edge ke-i
               Node2 = getNode2 dari Edge ke-i
               If (Node1 = HeadofTest and not dikunjungi[posisi
               node2 pada vertice])
                    Test.Push(Node2)
                    dikunjungi[posisi node2 pada vertice] ←
                    true
                    Found1 ← true
               Else
                    I ← I+1
          If (HeadofTest = b)
               Found ← true
          Else
               Test.Pop()
     If(Found = false)
          → result
```

```
      Else {menyalin stack ke list dari depan}
            Count ← 0
            While (sizeofTest > 0)
                  Result.Insert(0, HeadofTest)
                  test.Pop()
                  Count ← Count + 1
            → result
```

```
procedure MutualFriends(input a,b: string)
Deklarasi
I           : int
Tmp         : ListofString
ThisEdge    : ListofEdges
Res         : ListofString
K           : int
J           : int

Algoritma
I ← 0
While (i < sizeofThisEdge)
      Node1 = getNode1 dari Edge ke-i
      Node2 = getNode2 dari Edge ke-i
      If (Node1 = a)
            If (sizeofTmp = 0 or Node2 not in Tmp)
                  Tmp.Add(Node2)
      I ← I+1
K <- 0
While (K < sizeofThisEdge)
      Node1 = getNode1 dari Edge ke-i
      Node2 = getNode2 dari Edge ke-i
      If (Node1 = b)
            If (Node2 not in Tmp)
                  Res.Add(Node2)
      K ← K+1
If (sizeOfRes > 0)
      Output("Nama Akun :" +b)
      Output("Mutual Friends :" +sizeOfRes)
      For (j = 0 until sizeofRes)
            If (j = sizeofRes - 1 )
                  Output(Res[j])
            Else
                  Output(Res[j] + ",")
```

```
Output("\n")
```

4.2. Struktur Data

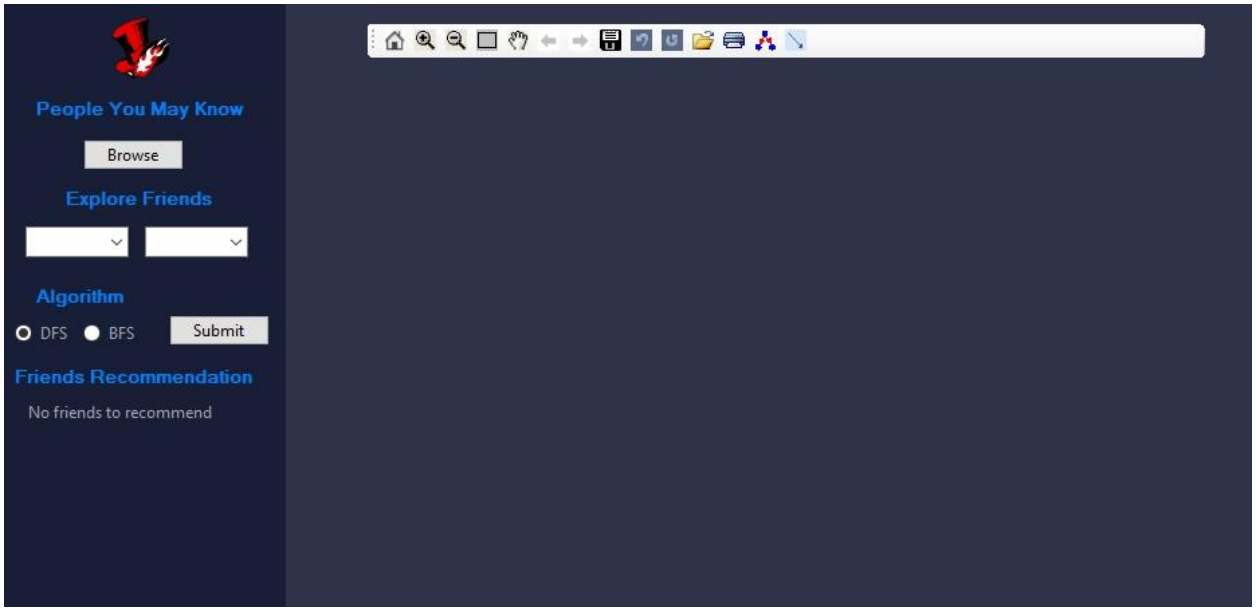
Pada pengerjaan aplikasi ini terdapat 2 struktur data, yaitu Graph dan Edges.

- Graph berisi:
  1. vertice : List<string>
  2. edge : List<Edges>
- Edges berisi:
  1. node1 : string
  2. node2 : string
- Form berisi
  1. fileBrowsed : bool
  2. namaFile : string

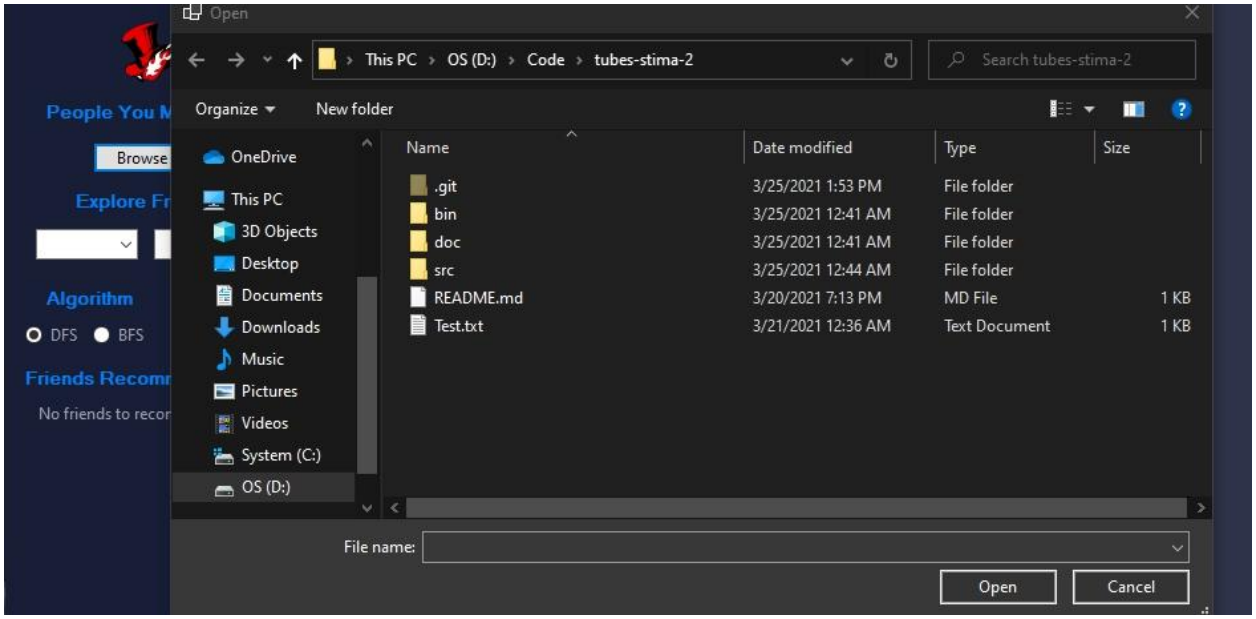
4.3. Tata Cara Penggunaan Program

Jalankan program kemudian masukkan dengan meng-*click* tombol *browse* lalu pilih *file* yang akan dimasukkan, *file* yang dimasukkan berisi hubungan antar graf tanpa menyertakan banyak graf yang berhubungan. Setelah memasukkan *file* akan muncul graf awal representasi dari *file* yang baru saja dimasukkan dan juga pilihan orang orang yang terdapat pada *file* tersebut, pilih orang yang ingin ditelusuri dalam *dropdown section*, kemudian pilih algoritma yang diinginkan (BFS / DFS) dan tekan tombol *submit*. Program akan menampilkan pencarian graf dengan algoritma yang telah ditentukan dengan mewarnai lintasan graf dengan warna merah dan memberikan panah terkait jalur pencarian. Selain itu program juga akan menampilkan rekomendasi teman untuk orang yang dimasukkan pada *dropdown section* yang pertama.

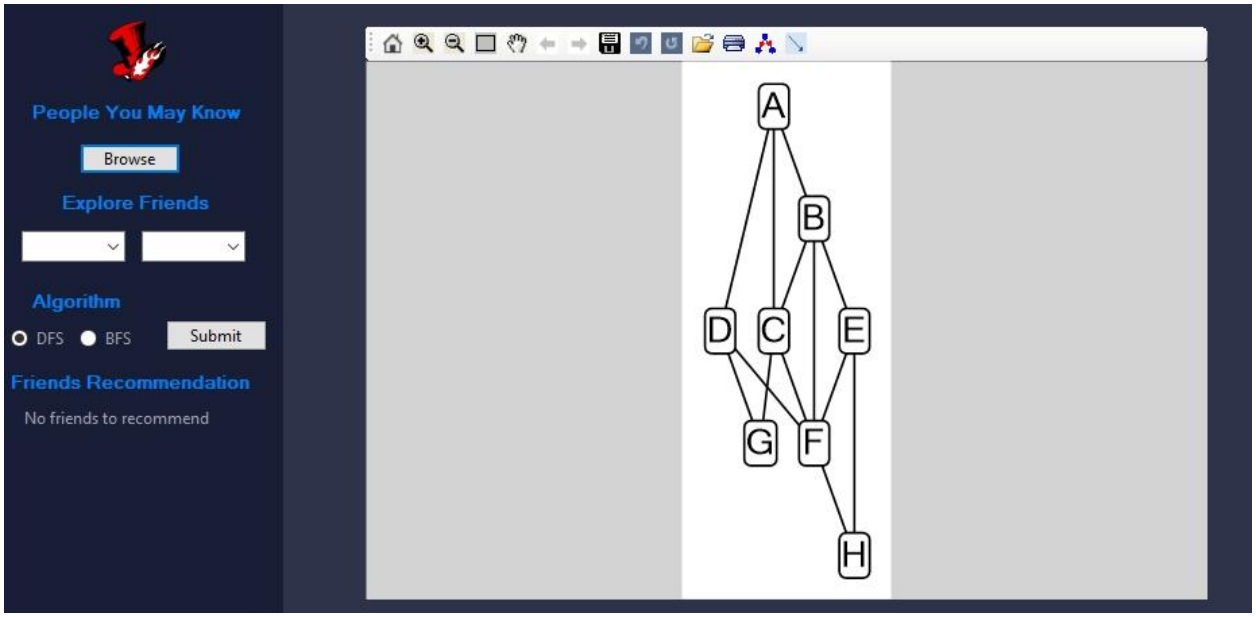
4.4. Hasil Pengujian



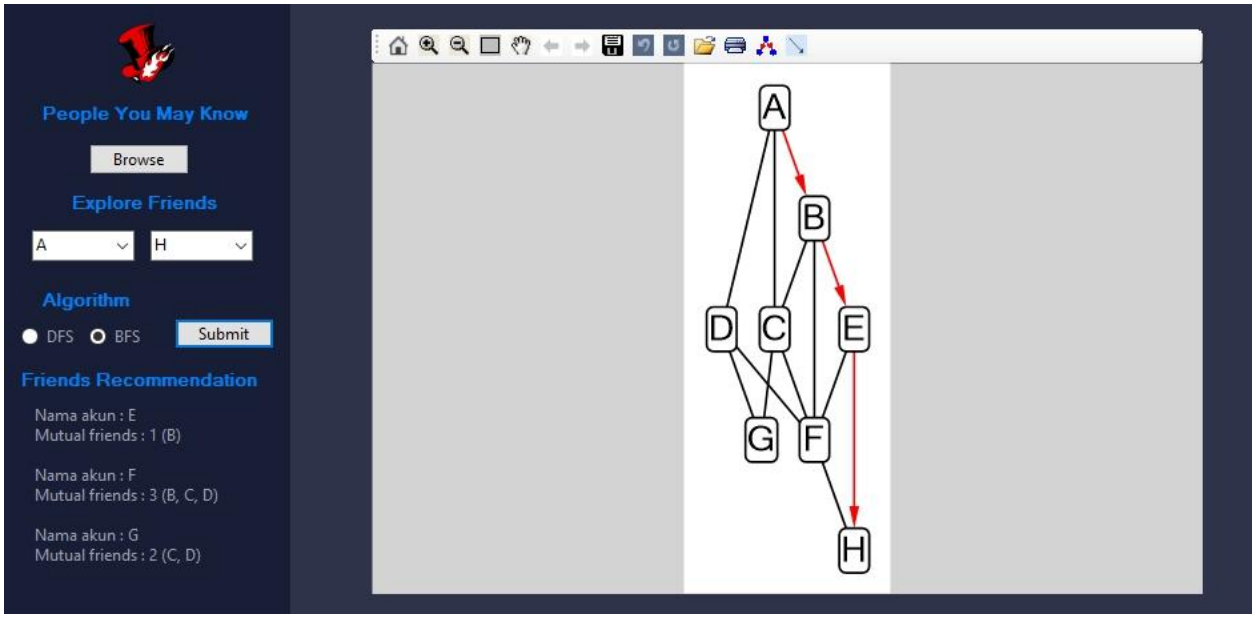
Tampilan awal program



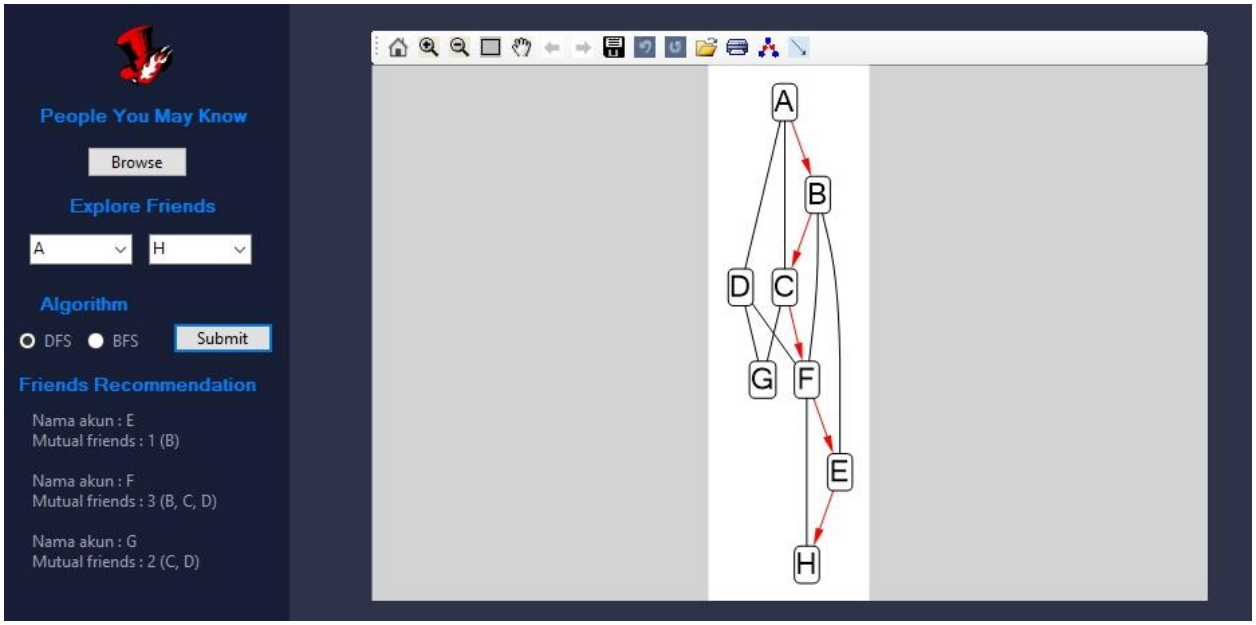
Melakukan *browse file*



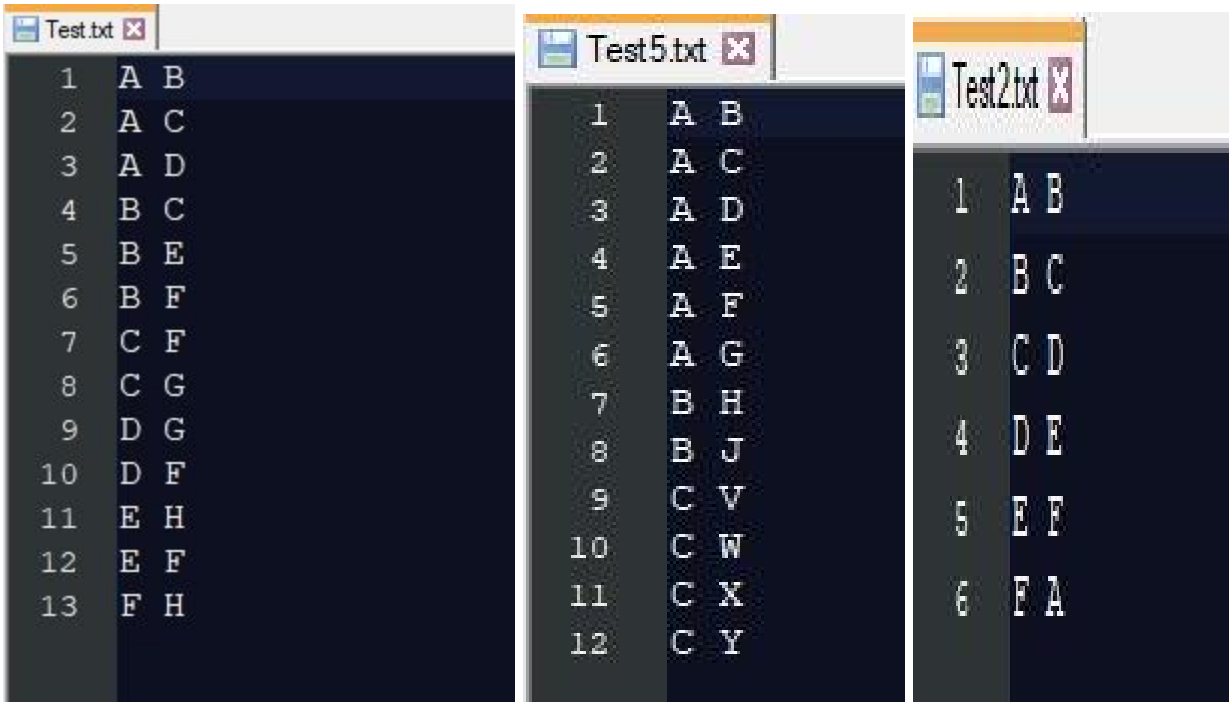
Graf akan ditampilkan setelah memasukkan file



Memilih algoritma BFS dan menekan tombol *submit*



Memilih algoritma DFS dan menekan tombol *submit*



Isi dari *test file*

4.5. Analisis Desain Solusi

Dari permasalahan yang diberikan, pencarian mutual friend lebih baik jika dilakukan dengan menggunakan *breadth first search* karena pencarian yang dilakukan "dekat" dengan teman yang dimiliki oleh akun pertama. Hal ini juga berlaku pada rekomendasi teman, dimana *breadth first search* akan mencari simpul simpul tetangga. Tidak seperti *depth first search* yang akan mendalami suatu simpul dan "menjauh" dari teman teman yang ada. Oleh karena itu algoritma *breadth first search* lebih baik untuk digunakan pada pencarian jejaring sosial media.

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Proses rekomendasi teman pada jejaring sosial media dapat dilakukan dengan menggunakan struktur data graf dengan melakukan pencarian kesamaan teman maupun keterhubungan teman dengan menggunakan algoritma BFS dan DFS. Dimana algoritma BFS akan mencari graf secara melebar terlebih dahulu sedangkan algoritma DFS akan mencari graf secara mendalam. Implementasi GUI juga membantu dalam memvisualisasikan algoritma yang telah dibuat sehingga mempermudah dalam melihat jalur explorasi antar akun.

5.2. Saran

1. Mempelajari C# terlebih dahulu

2. Membaca dokumentasinya dari windows application form
3. Menanyakan pertanyaan yang kurang jelas kepada asisten atau membaca FAQ
4. Mempelajari algoritma BFS dan DFS
5. Membuat rancangan algoritma sebelum mengimplementasikannya

DAFTAR PUSTAKA

1. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag1.pdf (Diakses pada 24 Maret 2021; 2:00)
2. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag2.pdf (Diakses pada 24 Maret 2021; 2:12)
3. https://docs.microsoft.com/en-us/dotnet/csharp/ (Diakses pada 24 Maret 2021; 2:31)