Tugas Besar II
IF2211 Strategi Algoritma

Kelompok “Sharpbook”

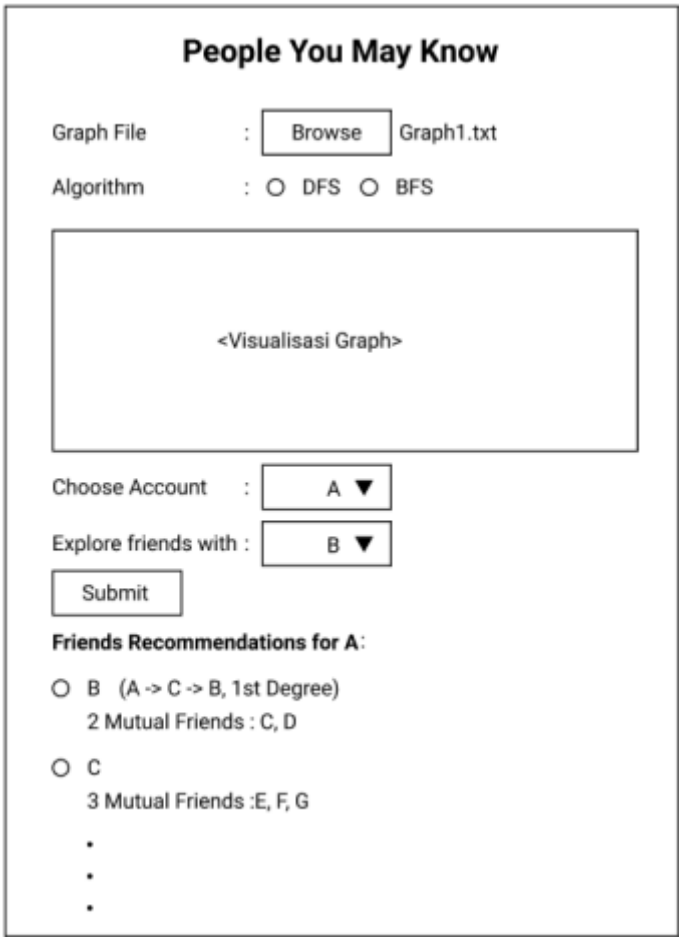
Andrew	13519036
Daffa Ananda Pratama Resyaly	13519107
Made Kharisma Jagaddhita	13519176

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

Tahun Ajaran 2020/2021

Bab 1
Deskripsi Tugas

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



Gambar 9. Tampilan layout dari aplikasi desktop yang dibangun

Spesifikasi GUI:

1. Program dapat menerima input berkas file eksternal dan menampilkan visualisasi graph.
2. Program dapat memilih algoritma yang digunakan.
3. Program dapat memilih akun pertama dan menampilkan friends recommendation untuk akun tersebut.
4. Program dapat memilih akun kedua dan menampilkan jalur koneksi kedua akun dalam bentuk visualisasi graf dan teks bertuliskan jalur koneksi kedua akun.
5. GUI dapat dibuat sekreatif mungkin asalkan memuat 4 spesifikasi di atas.

Program yang dibuat harus memenuhi spesifikasi wajib sebagai berikut:

1) Buatlah program dalam bahasa C# untuk melakukan penelusuran social network facebook sehingga diperoleh daftar rekomendasi teman yang sebaiknya di-add. Penelusuran harus memanfaatkan algoritma BFS dan DFS.
2) Awalnya program menerima sebuah berkas file eksternal yang berisi informasi pertemanan di facebook. Baris pertama merupakan sebuah integer N yang adalah banyaknya pertemanan antar akun di facebook. Sebanyak N baris berikutnya berisi dua buah string IF2211 Strategi Algoritma - Tugas Besar 2 7 (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.
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 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 2
Landasan Teori

2.1. Dasar teori *graph traversal*

Graph traversal adalah proses mengunjungi setiap simpul dalam suatu graf secara sistematik. Dalam pencarian sebuah solusi dengan representasi graf, terdapat dua pendekatan yaitu: graf statis dan graf dinamis. Graf statis yaitu graf sudah terbentuk terlebih dahulu sebelum pencarian dilakukan. Sedangkan graf dinamis yaitu graf akan terbentuk saat proses pencarian dilakukan. Terdapat dua algoritma *graph traversal* dalam pencarian solusi pendekatan graf statis yaitu: BFS (Breadth-First Search) dan DFS (Depth-First Search).

2.1.1. Breadth-First Search

Breadth-First Search adalah algoritma traversal graf secara melebar. Misalkan traversal dimulai dari sebuah simpul v. Maka langkah-langkah dari algoritma ini adalah:

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

Depth-First Search adalah algoritma traversal graf secara mendalam. Misalkan traversal dimulai dari sebuah simpul v. Maka langkah-langkah dari algoritma ini adalah:

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 yang 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.

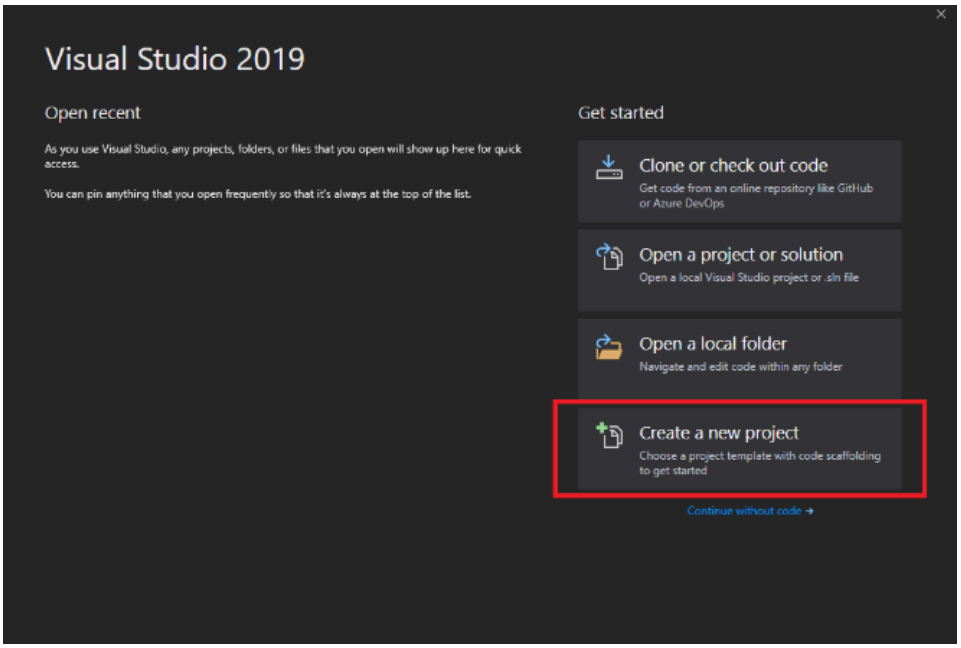
2.2. C# *desktop application development*

Dalam mengembangkan aplikasi desktop dengan menggunakan bahasa C#, kakas yang sering digunakan adalah Windows Forms. Windows Forms adalah sebuah framework yang dikembangkan pada Windows *graphics device interface* (GDI) *engine*. Windows Forms bergantung pada Visual Studio Designer sebagai cara utama dalam membuat *User Interface.*

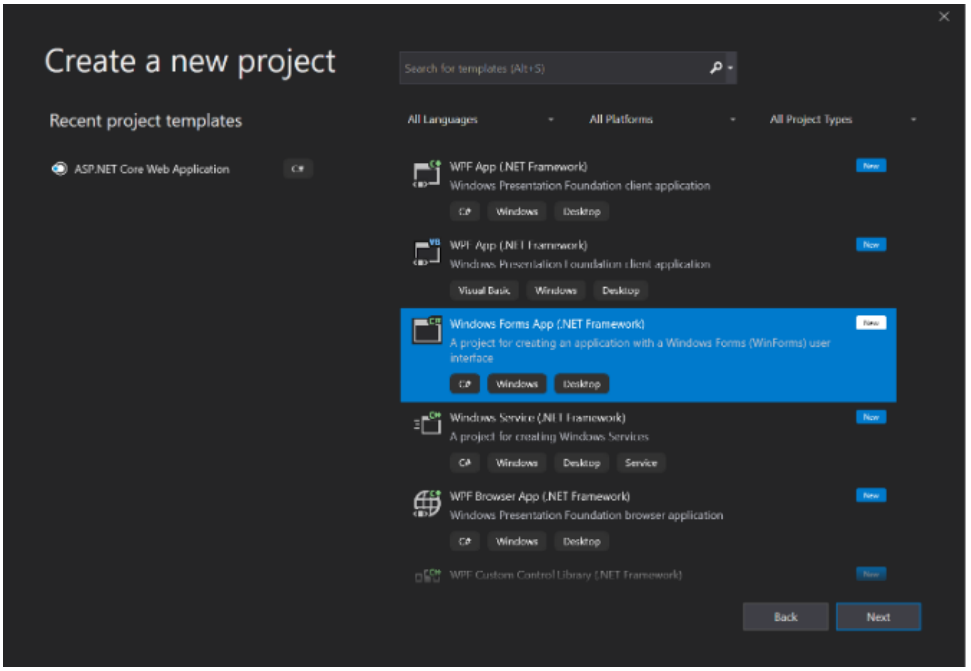
Berikut merupakan cara untuk membuat project aplikasi C#:

1. Buka Visual Studio 2019

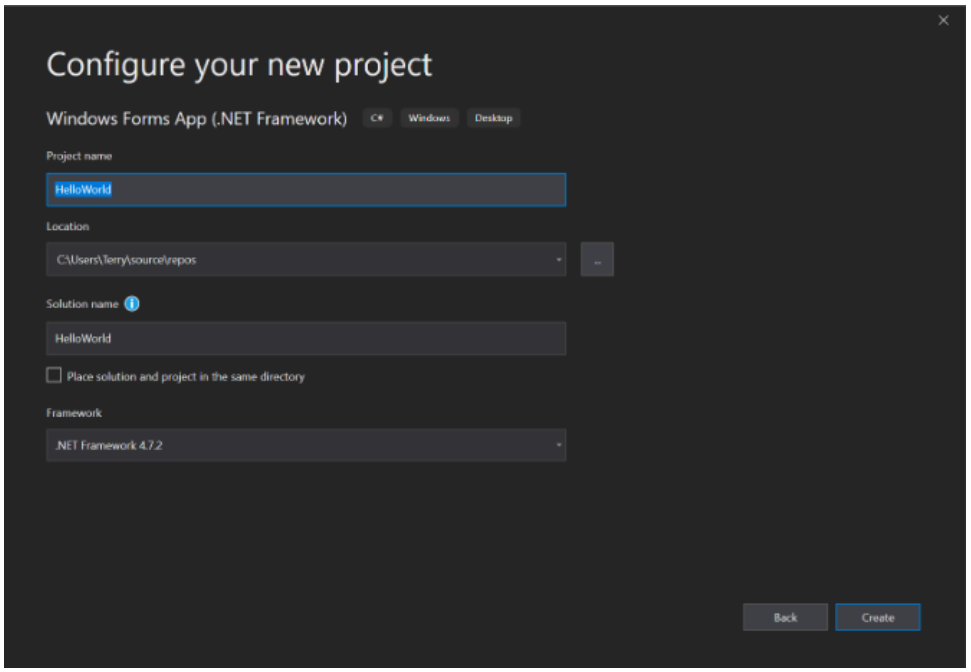
2. Pada window start, pilih “Create a new project”



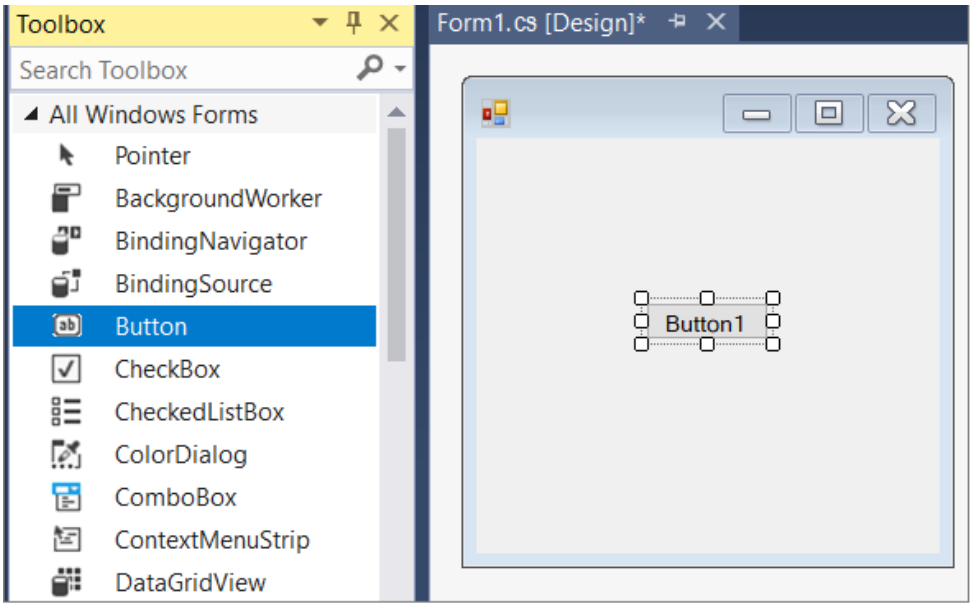
3. Pilih Windows Forms App (.NET Framework)



4. Konfigurasi project yang ingin dibuat.



5. Untuk membuat UI, dapat menggunakan toolbox pada sisi kiri dan melakukan *drag and drop* pada objek yang dibutuhkan. Setelah itu, konfigurasi aksi-aksi yang dilakukan dari setiap objek sesuai dengan kebutuhan.



Untuk melakukan visualisasi terhadap graf, dibutuhkan sebuah kakas tambahan yaitu MSAGL (*Microsoft Automatic Graph Layout)*. Untuk menggunakannya langkah-langkah yang harus dilakukan adalah menginstall NuGet Package untuk MSAGL dengan Package Manager, lalu menambahkan library MSAGL pada project, dengan menginstal modules MSAGL yang ada dari NuGet Package Manager

Bab 3
Analisis Pemecahan Masalah

3.1 Langkah Pemecahan Masalah

Friend Recommendation:

Pertama-tama, program akan mencari simpul-simpul yang bertetangga dengan simpul yang ingin dicari rekomendasi temannya (root). Lalu, untuk setiap simpul yang bertetangga dari simpul root, kunjungi simpul-simpul yang bertetangga. Jika simpul tersebut bukan tetangga dari simpul root dan bukan simpul root itu sendiri, maka mutual friend dari simpul tersebut akan bertambah satu. Ulangi langkah ini sampai tidak ada lagi simpul tetangga dari simpul root yang harus dikunjungi tetangganya.

BFS:

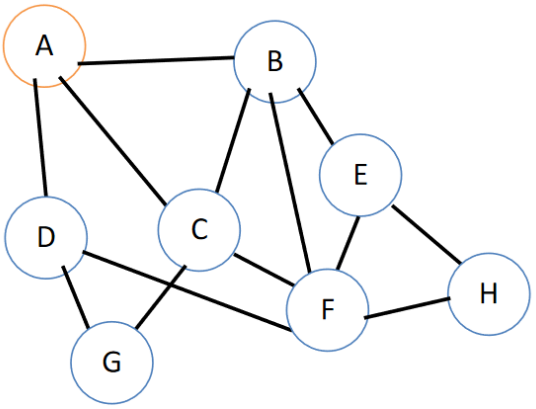
Pertama-tama, program akan memasukkan simpul yang akan diproses ke dalam queue. Program akan melakukan dequeue dan menandai simpul yang di-dequeue menjadi sudah dikunjungi. Kemudian, program akan melakukan iterasi setiap simpul yang bersisian dengan simpul yang telah di-dequeue. Jika simpul tersebut belum dikunjungi, maka program akan memasukkan simpul tersebut ke dalam queue. Ulangi langkah ini sampai queue kosong.

DFS:

Pertama-tama, program akan memproses simpul. Proses pada DFS terdiri atas tiga kasus. Kasus pertama, jika simpul yang diproses adalah simpul tujuan, maka proses akan mengembalikan lintasan dari simpul awal sampai simpul tujuan. Kasus kedua, jika simpul yang diproses bukan simpul tujuan, maka program akan melakukan proses untuk setiap simpul yang bersisian dengan simpul yang diproses. Kasus ketiga, jika lintasan simpul memasuki jalan buntu, maka program akan mengembalikan lintasan kosong. Proses-proses ini dilakukan secara rekursif.

3.2 Proses Mapping Persoalan

Diberikan graf jaringan pertemanan seperti berikut.



Untuk fitur explore friends dengan BFS, jika user ingin melakukan explore dari simpul A ke simpul H, maka urutan iterasi simpulnya adalah sebagai berikut.

A > B > C > D > E > F > G > H

Untuk fitur explore friends dengan DFS, jika user ingin melakukan explore dari simpul A ke simpul H, maka urutan iterasi simpulnya adalah sebagai berikut.

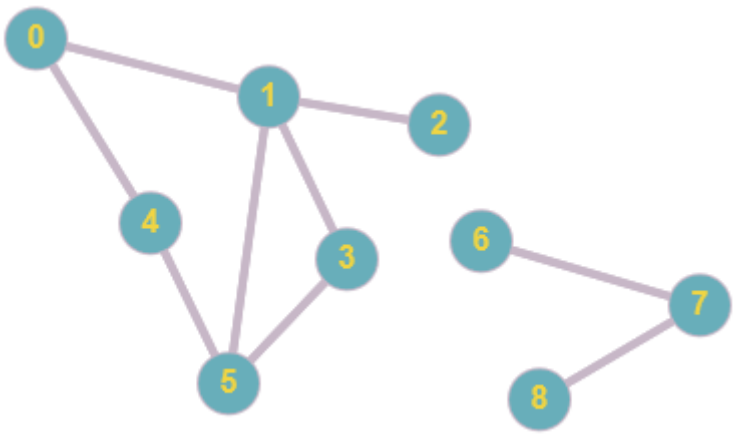
A > B > C > F > D > G > E > H

Didapatkan:

1. Graf pertemanan: ruang status
2. Simpul: status-status dari persoalan
3. Cabang: prioritas langkah yang diambil dalam pencarian (dalam program ini prioritas berdasarkan abjad)
4. Simpul akar: state awal dari pencarian
5. Simpul goal: state goal yang harus dicapai
6. Ruang status: himpunan semua simpul pada graf pertemanan
7. Ruang solusi: himpunan semua status solusi.

3.3 Ilustrasi Kasus Lain

Diberikan graf jaringan pertemanan seperti berikut.



Perhatikan bahwa graf tersebut memiliki dua komponen, yaitu komponen pertama dengan simpul 0, 1, 2, 3, 4, dan 5, dan komponen kedua dengan simpul 6, 7, dan 8. Dengan demikian, tidak dapat dilakukan explore friends antara simpul-simpul di dalam komponen pertama dan simpul-simpul di dalam komponen kedua.

Bab 4
Implementasi dan Pengujian

4.1. Implementasi Program

```
Function FriendRecommendation(input namaSimpul:string,  output
sortedRecommendation{string:ListofString}:Dictionary)
{
    Dictionary recommendation{string:ListofString}
    Dictionary sortedRecommendation{string:ListofString}
    List rootFriends[string]
    List friendFromFriends[string]
    Simpul root

    root <- GetSimpulByName(namaSimpul)
    rootFriends <- GetSimpulYangBersisian(root)

    for friend in rootFriends
       friendFromFriends <- GetSimpulYangBersisian(friend)
       for friendFromFriend in friendFromFriends
          if (friendFromFriend not in rootFriends and friendFromFriend ≠  namaSimpul)
             if (recommendation contains key friendFromFriend)
                 Add friend to recommendation with key friendFromFriend;
                 else
                  List mutualFriends[string]
                  Add friend to mutual friends
                  Add friendFromFriend as new recommendation key
                  Add mutualFriends as new recommendation value
             endif
          endif
       endfor
    endfor
    sortedRecommendation <- orderDescendingValueOf(recommendation)
    -> sortedRecommendation
}

Function BFS(input s:string, e:string, output lintasan[string]:List)
{
   Queue queue
   Dictionary visited{string:bool}
   Dictionary previous{string:string}

   for i = 0 to SimpulN
      Add NamaSimpul as visited key
      Add false as visited value
      Add NamaSimpul as visited key
      Add blank as visited value
   endfor
   Enqueue s to queue;
   Visited[s] <- true
   while (queue.Any())
      String dequeued <- Dequeue from queue
      Simpul simp <- GetSimpulByName from dequeued
      for calon in simp.GetBersisian()
         if (!visited[calon])
            Enqueue calon ke queue
            visited[calon] <- true
            previous[calon] <- dequeued
         endif
      endfor
   endwhile

   List lintasan[string]
```

```
   string x
   for x = e, x ≠ "", x = previous[x]
      Add x to lintasan
   endfor
   Reverse lintasan
   if (lintasan[0] = s)
      for string i in lintasan)
          Output to Console i
      endfor
      -> lintasan
      else
          Output to Console "Tidak ditemukan"
          -> null
   endif
}

Function DFS(input s:string, e:string, output DFSHandler[string]:List)
{
   Dictionary visited{string:bool}
   List path[string]
   Int i

   for i from 0 to SimpulN
      Add NamaSimpul as visited key
      Add false as visited value
   endfor
   -> DFSHandler(visited, path, s, e)
}

Function DFSHandler(input visited{string,bool}:Dictionary, path[string]:List,
at:string, finish:string, output path[string]:List, temp[string]:List)
{
   List temp[string]
   String astg
   String calon
   List tempPath[string]

   Visited[at] <- true
   if (at = finish)
      Add at to path
      -> path
   endif
   for calon in SimpulYangBersisianDengan(at)
      if visited[calon] = false
         foreach astg in path
            Add astg to tempPath
         endfor
         Add at to tempPath
         temp[] <- DFSHandler(visited, tempPath, calon, finish);
         if (temp ≠ null)
            -> temp
         endif
      endif
   endfor
   -> null
}
```

4.2. Struktur data yang digunakan

Pada program ini, struktur data utama yang digunakan adalah Graf. Struktur tersebut diimplementasikan dengan menggunakan objek Graf dan Simpul. Representasi graf yang digunakan adalah representasi *adjacency list*.

Berikut merupakan struktur kelas Graf:

```
Class Graf
{
   private simpuls: array of Simpul
   private nSimpuls: integer

   public ShowGraf()
   public AddSimpul()
   public DelSimpul(): Simpul
   public GetSimpulByName(string simpul): Simpul
   public GetSimpuls(): array of Simpul
   public GetNSimpul(): integer
   public  FriendRecommendation(string  namaSimpul):  Dictionary<string,
   array of string>
   public BFS(string s, string e): array of string
   public DFS(string s, string e): array of string
   public  DFSHandler(Dictionary<string,  bool>  visited,  List<string>
   path, string at, string finish): array of string
}

Class Simpul
{
   private nama: string
   private nBersisian: integer
   private bersisian: array of string

   public GetNama(): string
   public AddBersisian(string _namaB)
   public sortBersisian()
   public GetBersisian(): array of string
   public ShowSimpul()
}
```

4.3. Tata cara Penggunaan Program

1. Program menerima input file eksternal berisi hubungan-hubungan simpul dalam suatu graf dan menyimpan input tersebut ke dalam sebuah kelas graf.
2. User memilih fitur yang ingin digunakan dari dua pilihan, yaitu “Explore Friend” dan “Friend Recommendation” dimana fitur default program adalah “Explore Friend”.
3. Jika user memilih fitur “Explore Friend”, program akan menampilkan tiga *Combo Box*.
4. User harus memilih sebuah akun ‘dari’ yang berada di dalam *Combo Box* dengan label “From:”, sebuah akun tujuan yang berada di dalam *Combo Box* dengan label “To:”, dan algoritma pencarian yang ingin dipakai pada *Combo Box* dengan label “Algoritma:”
5. Setelah user memilih, user harus menekan tombol “Submit”
6. Program kemudian akan menampilkan nama akun pilihan user, derajat koneksi, dan jalur koneksi antara akun ‘dari’ dan akun tujuan.
7. Jika user memilih fitur “Friend Recommendation”, program akan menampilkan sebuah *Combo Box* yang merupakan akun yang ingin dicari rekomendasi temannya.
8. User harus memilih akun agar kemudian dapat menekan tombol “Submit”.

9. Program kemudian akan menampilkan daftar rekomendasi teman untuk akun pilihan user.

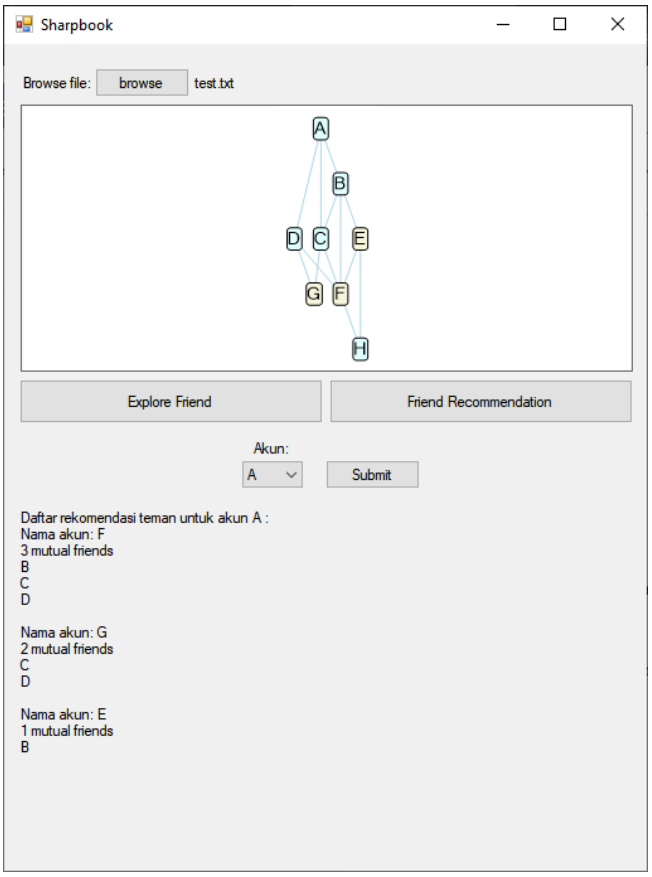
4.4. Hasil Pengujian

1. Test Case Pertama
   Input:

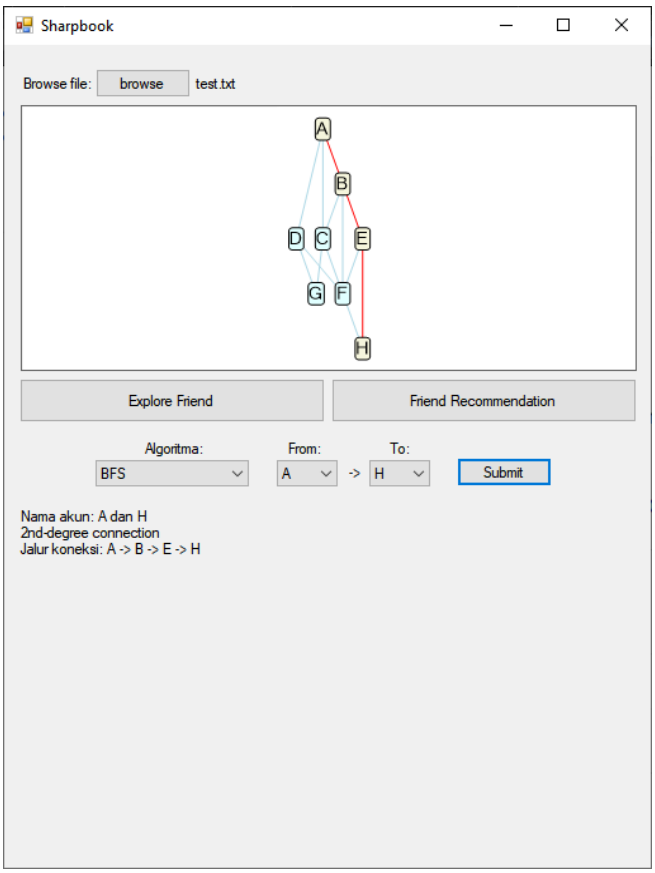
```
13
A B
A C
A D
B C
B E
B F
C F
C G
D G
D F
E H
E F
F H
```

   Output:

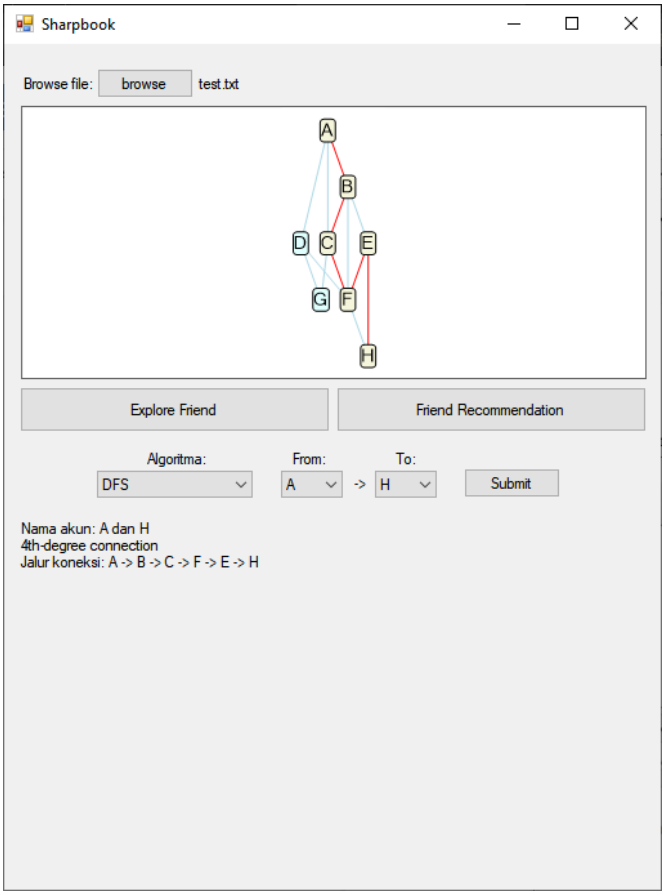
   a. Friend Recommendation dari A



   b. Explore Friend menggunakan BFS dari A ke H



c. Explore Friend menggunakan DFS dari A ke H



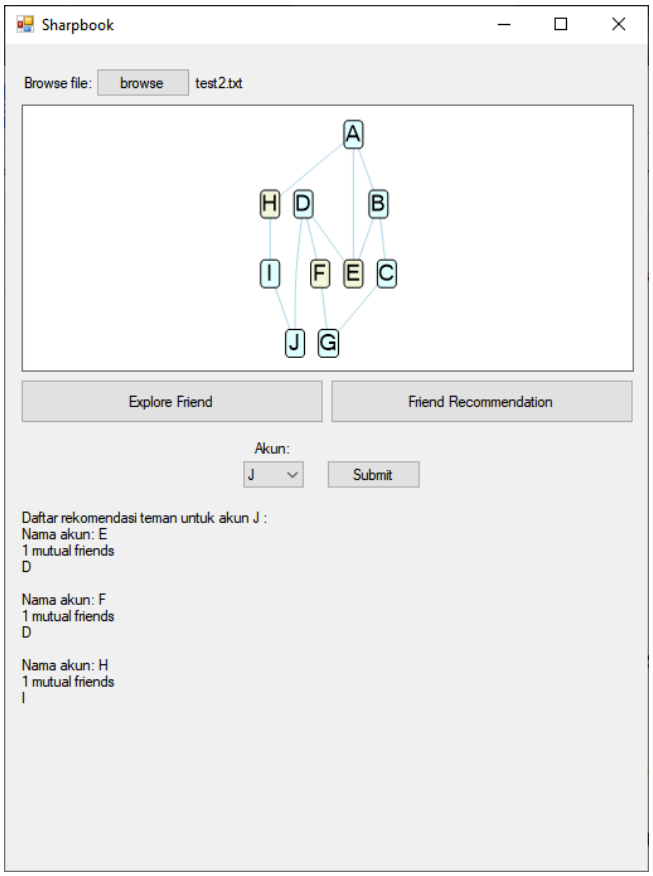
2. Test Case Kedua

Input:

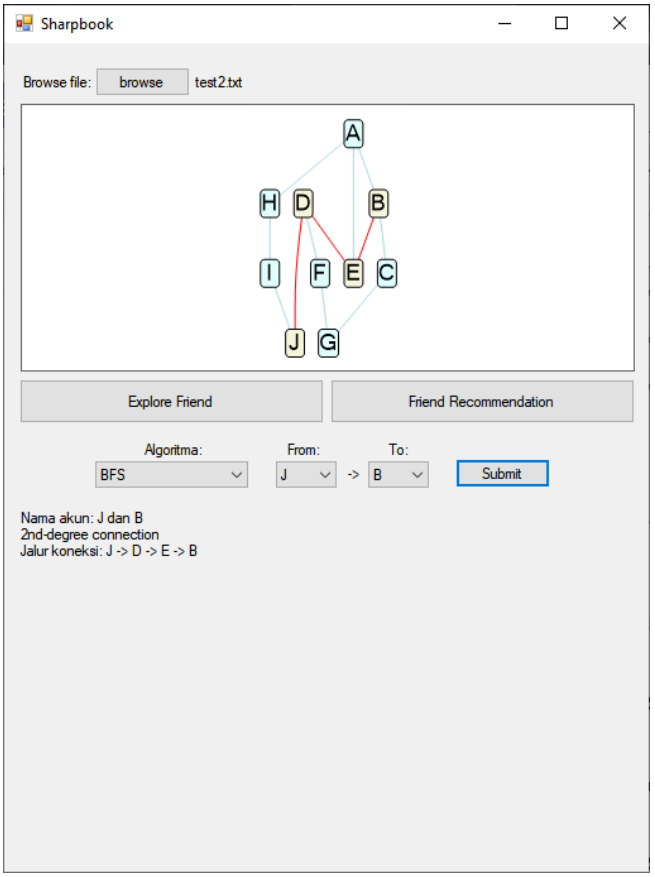
```
12
A B
A E
A H
B C
B E
C G
D E
D F
D J
F G
H I
I J
```

Output:

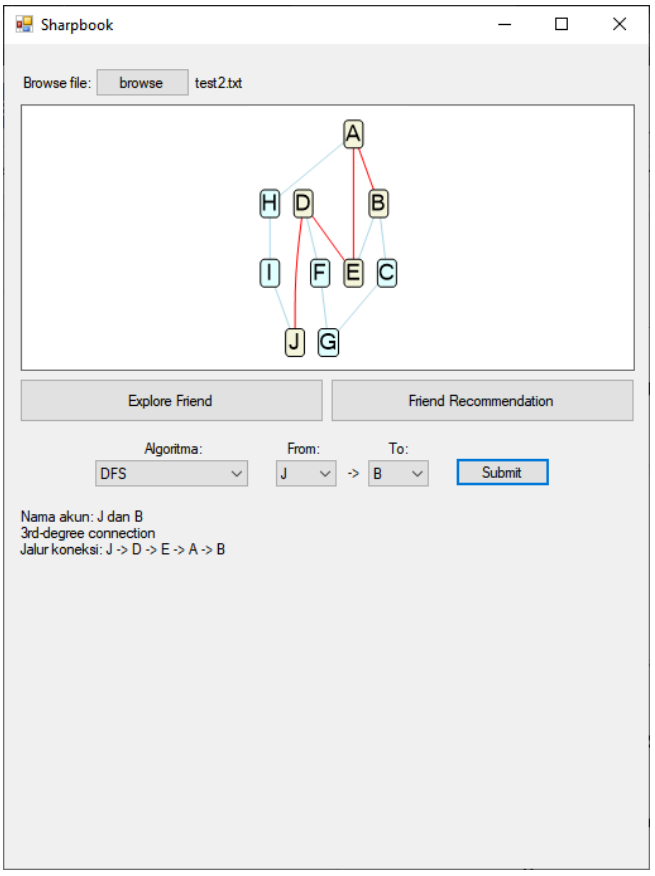
a. Friend Recommendation dari J



b. Explore Friend menggunakan BFS dari J ke B



c. Explore Friend menggunakan DFS dari J ke B



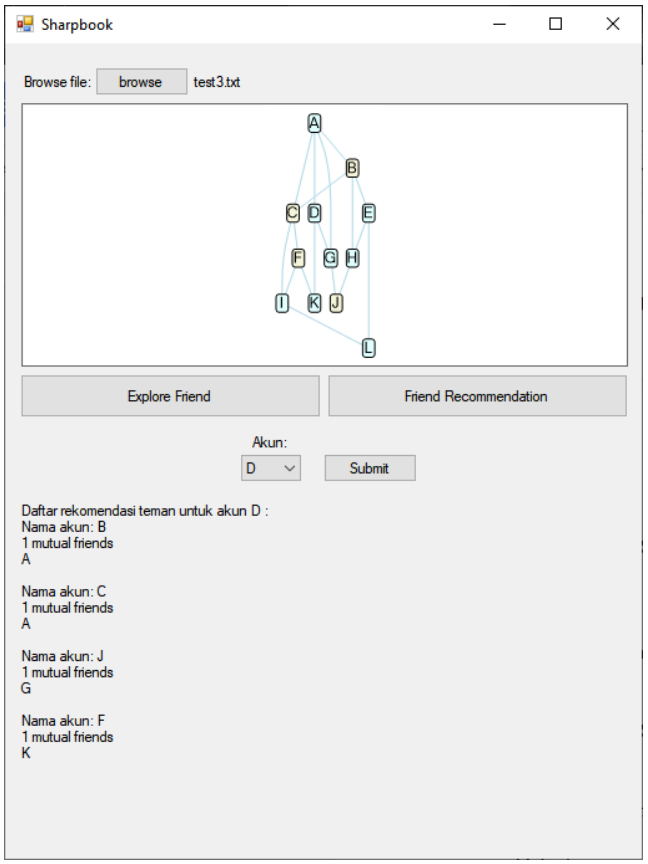
3. Test Case Ketiga

Input:

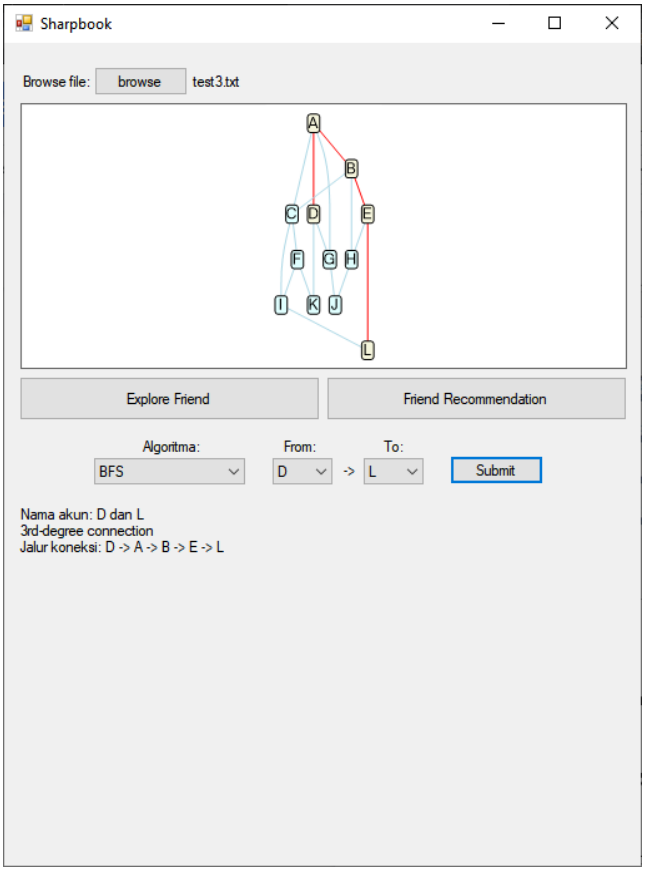
```
18
A B
A C
A D
A G
B C
B E
B H
C F
C I
D G
D K
E H
E L
F I
F K
G J
H J
I L
```

Output:

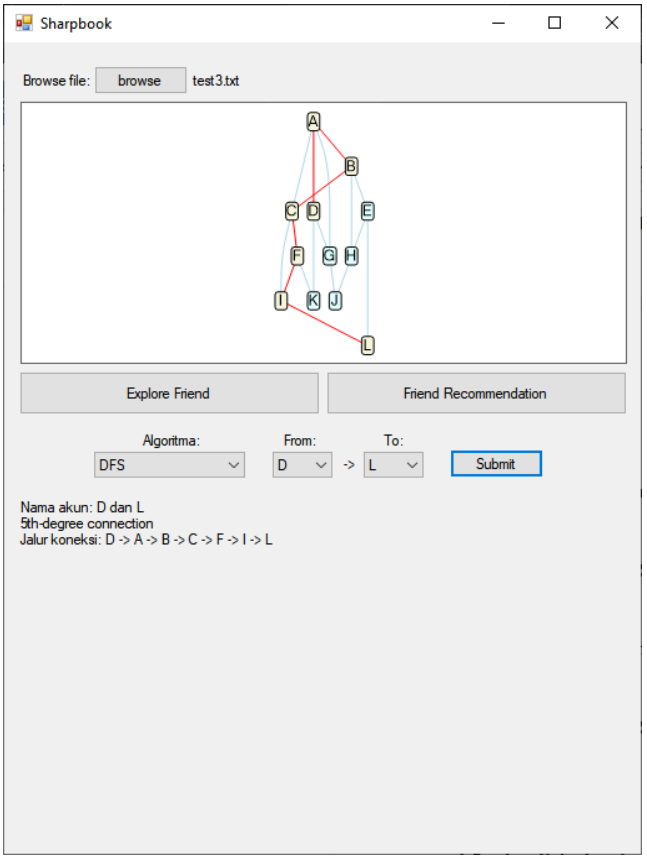
a. Friend Recommendation dari D



b. Explore Friend menggunakan BFS dari D ke L



c. Explore Friend menggunakan DFS dari D ke L



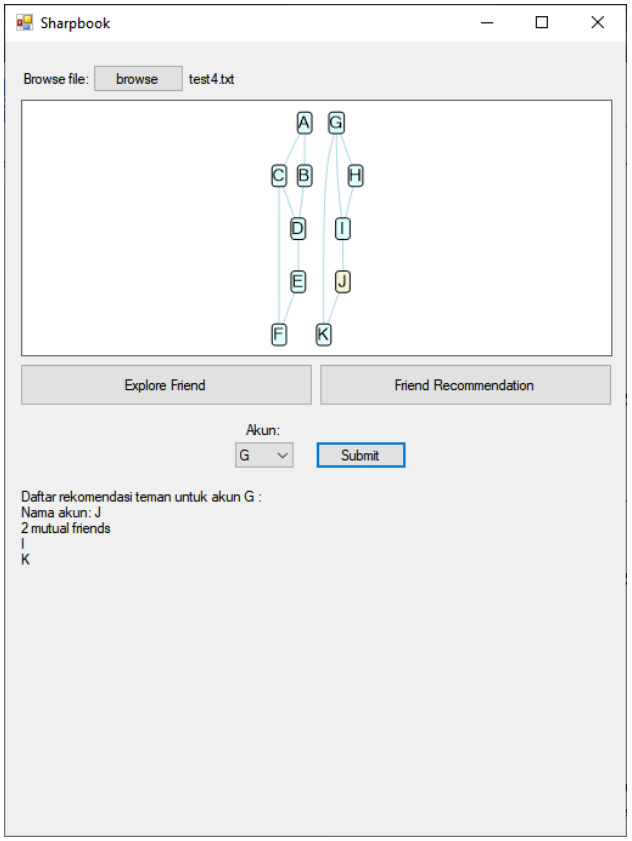
4. Test Case Keempat

Input:

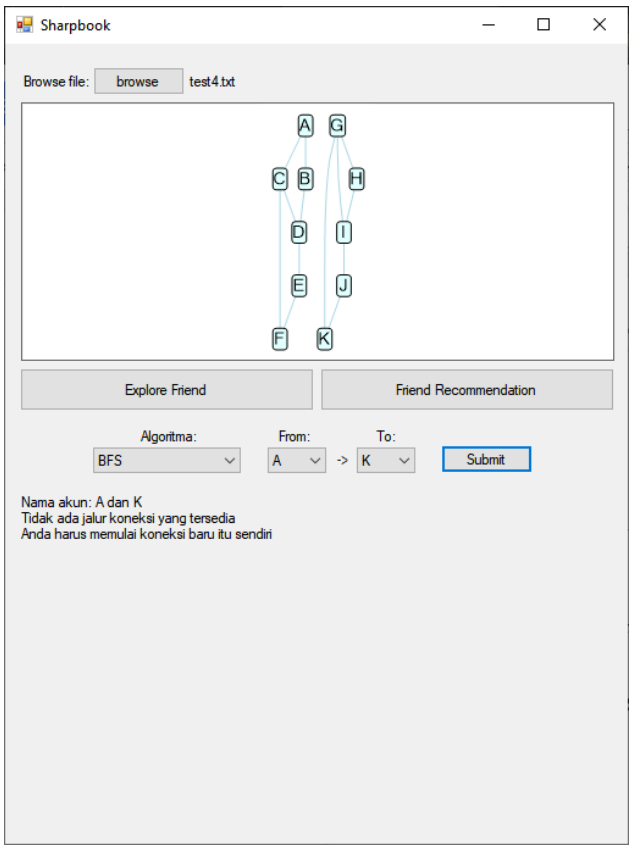
```
13
A B
A C
B D
C D
C F
D E
E F
G H
G I
G K
H I
I J
J K
```

Output:

a. Friend Recommendation dari G



b. Explore Friend dari A ke K



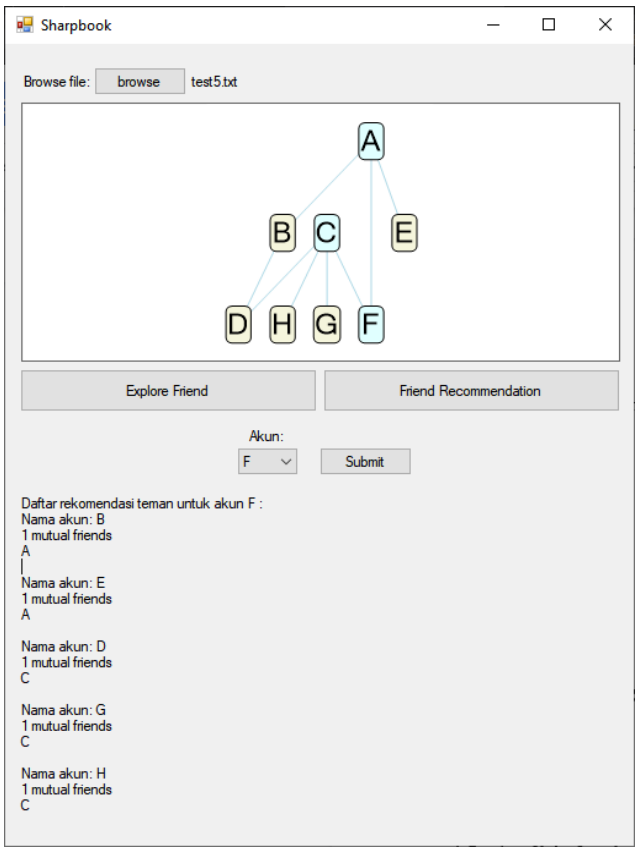
5. Test Case Kelima

   Input:

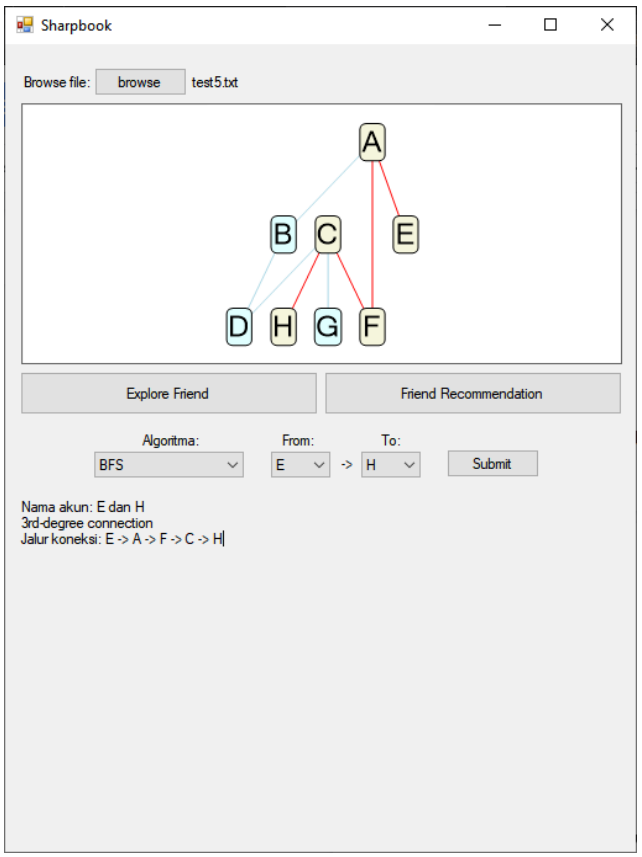
```
8
A B
A E
A F
B D
C D
C F
C G
C H
```

   Output:

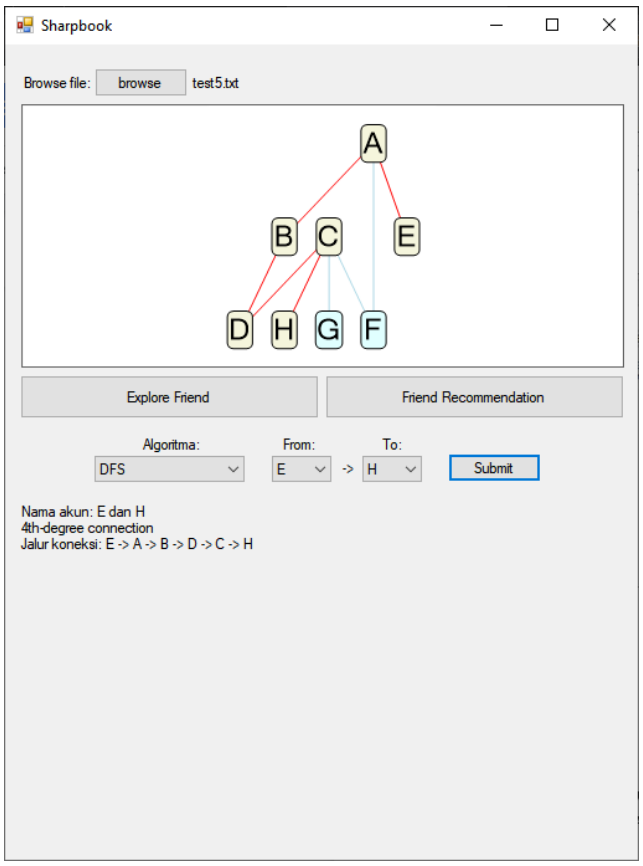
   a. Friend Recommendation dari F



b. Explore Friend menggunakan BFS dari E ke H



c. Explore Friend menggunakan BFS dari E ke H



4.5. Analisis desain solusi

Dari desain solusi yang dibuat, setiap algoritma memiliki kelebihan dan kekurangan masing-masing. Pada algoritma DFS, memori yang digunakan lebih sedikit dibandingkan dengan BFS dan terdapat kemungkinan pencarian yang dilakukan lebih lama jika simpul yang ada sangat banyak. Algoritma DFS juga tidak menjamin mendapatkan jalan terdekat untuk mencapai simpul goal. Sementara itu algoritma BFS menjamin mendapatkan jalan terdekat untuk mencapai simpul goal. Tetapi akibatnya memori yang digunakan sangat boros.

Bab 5
Kesimpulan, Saran, dan Refleksi

5.1. Kesimpulan

Algoritma traversal graf memiliki banyak kegunaan. Salah satunya adalah fitur "People You May Know" yang terdapat pada aplikasi Facebook. Untuk mencari seseorang yang mungkin dikenal, dapat dilakukan pencarian dengan memakai algoritma BFS (Breadth-First Search) maupun DFS (Depth-First Search).

5.2. Saran

Diberlakukan sistem milestone karena sangat banyak mahasiswa yang senang mengerjakan tugas dekat dengan deadline.

5.3. Refleksi

Kerjakan tugas dengan semangat dan pantang menyerah.

5.4. Komentar

Tugas yang diberikan sangat asyik dan membantu dalam memahami algoritma traversal graf. Selain itu, spesifikasi yang mengharuskan membuat GUI sangat mendorong kami untuk bereksplorasi.

Daftar Pustaka

- Munir, Rinaldi. Breadth First Search (BFS) dan Depth First Search (DFS) Bagian 1 dan 2, Program Studi Teknik Informatika ITB. 2021.
- docs.microsoft.com, “Create a Windows Forms app in Visual Studio with C#”, 26 September 2019. <https://docs.microsoft.com/en-us/visualstudio/ide/create-csharp-winform-visual-studio?view=vs-2019> diakses pada 25 Maret 2021.
- github.com, “Automatic Graph Layout”, 18 Maret 2021. <https://github.com/microsoft/automatic-graph-layout> diakses pada 19 Maret 2021.