

**LAPORAN TUGAS BESAR
IF2211 Strategi Algoritma
Semester II 2020/2021**

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



Disusun oleh:
Kelompok Socialink

Anggota:

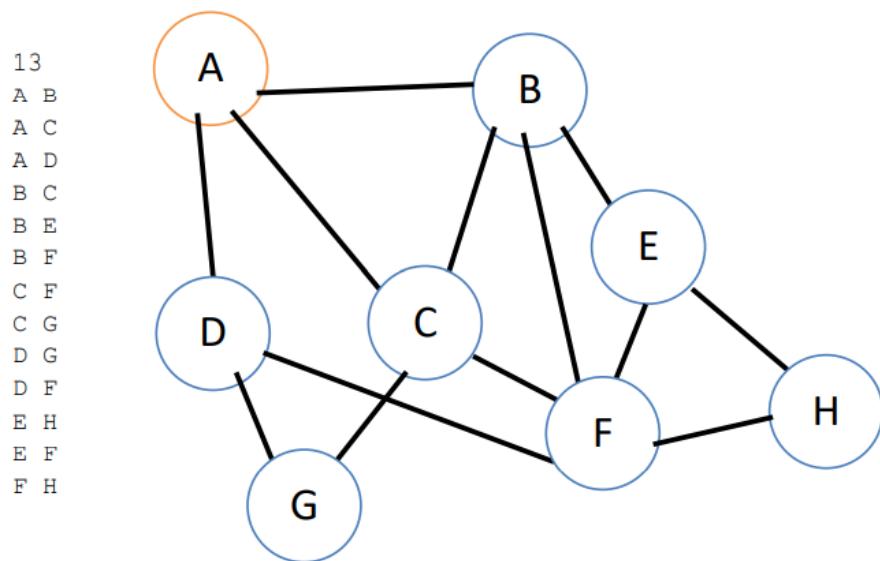
13519063	Melita
13519070	Mhd. Hiro Agayeff Muslion
13519171	Fauzan Yubairi Indrayadi

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

Bab 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.

Berikut contoh visualisasi dan graf 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

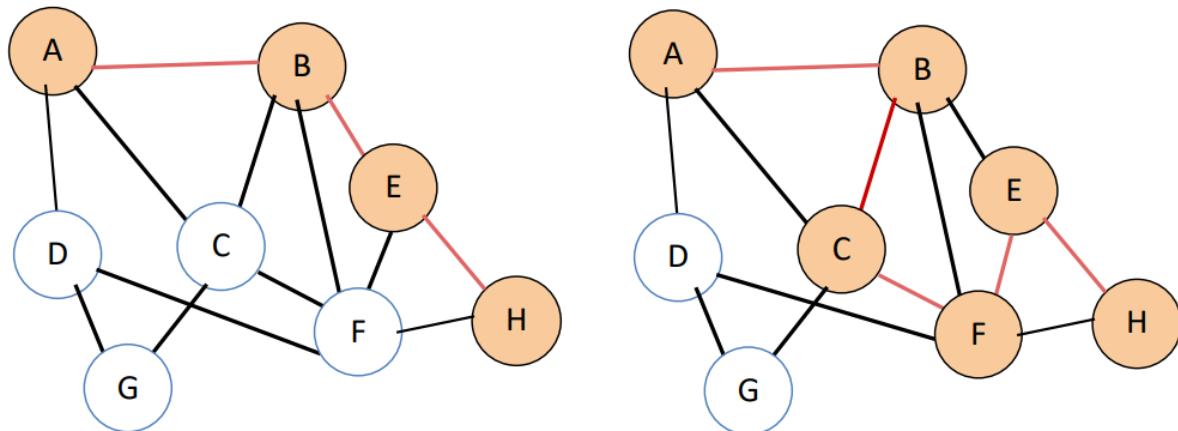
1 mutual friend:

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.

Nama akun: A dan H
2nd-degree connection
A → B → E → 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. Sedangkan untuk penggunaan algoritma DFS, diperoleh jalur lainnya, yaitu A-B-C-F-E-H.



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.

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.

Bab 2 Landasan Teori

2.1 Dasar Teori

Algoritma traversal graf merupakan algoritma pencarian solusi untuk mengunjungi simpul dari graf terhubung dengan cara yang sistematik. Dalam proses pencarian solusi, terdapat dua pendekatan, yaitu graf statis yang sudah terbentuk sebelum proses pencarian dilakukan dan graf dinamis yang terbentuk saat proses pencarian dilakukan. Terdapat pula dua jenis algoritma pencarian solusi, yaitu tanpa informasi (*uninformed/blind search*) dan dengan informasi (*informed search*). Beberapa contoh dari algoritma tanpa informasi adalah pencarian melebar (*breadth first search/BFS*) dan pencarian mendalam (*depth first search/DFS*).

Pencarian melebar (BFS) merupakan pencarian dengan algoritma sebagai berikut. Pertama, kunjungi simpul awal v . Lalu, kunjungi semua simpul yang bertetangga dengan simpul v terlebih dahulu. Kemudian, kunjungi simpul yang belum dikunjungi dan bertetangga dengan simpul-simpul tetangga v yang dikunjungi sebelumnya sampai nilai yang dicari ditemukan. Pencarian juga akan berakhir jika semua simpul telah dikunjungi.

Sementara itu, pencarian mendalam (DFS) merupakan pencarian dengan algoritma sebagai berikut. Pertama, kunjungi simpul awal v , lalu kunjungi satu simpul w yang merupakan tetangga simpul v . Setelah itu, akan dikunjungi simpul tetangga dari w yang belum dikunjungi, dan proses akan diulangi sampai ditemukan nilai yang dicari. Jika pencarian mencapai suatu simpul yang semua tetangganya sudah dikunjungi, akan dilakukan *backtracking* ke simpul sebelumnya yang mempunyai tetangga yang belum dikunjungi. Pencarian akan berakhir jika semua simpul telah dikunjungi.

2.2 C# Desktop Application Development

Tugas besar ini dibuat dengan bahasa C# sebagai *desktop application*. *Desktop application* adalah aplikasi yang berjalan langsung pada sistem operasi komputer, berbeda dengan *web application* yang berjalan pada *browser*. Terdapat beberapa IDE yang dapat digunakan untuk membuat *desktop application*, salah satunya adalah Visual Studio.

Bab 3 Analisis Pemecahan Masalah

3.1 Langkah-Langkah Pemecahan Masalah

Pertama, program membaca input dari *file .txt*. Lalu, program menghitung jumlah simpul unik dalam *file*, misal n , lalu membuat sebuah matriks ketetanggan berukuran $n \times n$. Setiap indeks matriks melambangkan satu simpul dan semua simpul disusun sesuai urutan alfabetis pada indeks matriks. Matriks elemen i,j diisi dengan nilai *true* jika terdapat koneksi antara simpul i dan j . Dibuat juga sebuah daftar nama simpul yang menyimpan nama simpul untuk semua indeks matriks.

Setelah itu, program akan membuat sebuah graf yang menggambarkan hubungan pertemanan dari *file*. Program juga akan menampilkan pilihan akun untuk ditelusuri rekomendasi temannya atau dicari hubungannya dengan satu akun lain. Program akan memberi rekomendasi teman atau melakukan *explore* saat tombol terkait ditekan. Pada algoritma DFS, program mencari jalur dari simpul awal ke simpul akun tujuan sesuai dengan algoritma DFS yang sudah dijelaskan, begitu juga dengan algoritma BFS.

3.2 Mapping Persoalan

Persoalan rekomendasi teman menggunakan jumlah *mutual friends* dan pencarian jalur dari akun A ke akun B pada Facebook dapat diselesaikan dengan metode pencarian graf BFS atau DFS. Dalam hal ini, simpul graf adalah akun pengguna Facebook, sedangkan sisi graf adalah hubungan pertemanan antara kedua akun. Dengan demikian, persoalan ini dapat digambarkan dengan graf sederhana. Akun pertama menjadi simpul awal pencarian, sedangkan akun kedua menjadi simpul tujuan. Pencarian akan dihentikan jika simpul tujuan ditemukan atau jika semua simpul sudah diperiksa.

3.3 Contoh Ilustrasi Kasus Lain

Contoh kasus pertama adalah jika kedua akun yang dipilih untuk fitur *explore friends* sama. Dalam kasus tersebut, akan dimunculkan sebuah pesan *error* agar *user* memilih dua akun yang berbeda. Contoh kasus kedua adalah jika akun tidak memiliki rekomendasi teman. Dalam kasus tersebut, akan dimunculkan tulisan tidak ada rekomendasi teman.

Bab 4 Implementasi dan Pengujian

4.1 Pseudocode Program Utama

```
procedure recFriend (input matriks:matriks of boolean, input daftarHuruf:list of string, input akun:integer)
Deklarasi :
    jumlahNode : integer
    friend : list of integer
    recom : array of integer
Algoritma :
    for i ← 0 to jumlahNode do
        if matriks[akun, i] then
            friend.Add(i)
        endif
    endfor
    for i ← 0 to jumlahNode do
        recom[i] ← 0
    endfor
    for node in friend do
        for i ← 0 to jumlahNode do
            if matriks[node, i] and not friend.Contains(i) and i /= akun then
                recom[i] ← recom[i] + 1
            endif
        endfor
    endfor
    { print recommended friend }

procedure expDFS (input matriks:matriks of boolean, input daftarHuruf:list of string, input a, b : integer)
Deklarasi :
    Stack, printStack : Stack
    jumlahNode, i, j : integer
    visited : array of boolean
Algoritma :
    for k ← 0 to jumlahNode do
        visited[k] ← false
    endfor
    stack.Push(a)
    visited[a] ← true
    selesai ← false
    while not visited[b] do
        if j = jumlahNode then
            j = stack.Pop()
            if stack.Count() = 0 then
```

```

                selesai ← true
            else
                i ← stack.Peek()
                j ← j + 1
            endif
        else if matriks[i, j] and not visited[j] then
            stack.Push(j)
            visited[j] ← true
            i ← j
            j ← 0
        else
            j ← j + 1
        endif
    endwhile
    { print hasil explore dengan DFS }

procedure    expBFS(input      matriks:matriks      of      boolean,      input
daftarHuruf:list of string, input a, b : integer)
Deklarasi :
    q : Queue
    jumlahNode : integer
    temu : boolean
    visited : array of boolean
    prevroute : array of integer
Algoritma :
    q.Enqueue(a)
    for i ← 0 to jumlahNode do
        prevroute[i] ← -1
    endfor
    visited[a] ← true
    temu ← false
    while q.Count /= 0 and not temu do
        current ← q.Dequeue()
        if current /= b then
            for i ← 0 to jumlahNode do
                if not visited[i] and matriks[i, current]
                    visited[i] ← true
                    q.Enqueue(i)
                    prevroute[i] ← current
            endif
        endfor
    else
        temu ← true
    endif
    endwhile
    { print hasil explore dengan BFS }

```

```

procedure button3_Click(input...)
Deklarasi :
    matriks : matriks of boolean
    daftarHuruf : list of string
    dari, ke : integer
    procedure expDFS (input matriks:matriks of boolean, input
daftarHuruf:list of string, input a, b : integer)
    procedure expBFS(input matriks:matriks of boolean, input
daftarHuruf:list of string, input a, b : integer)
Algoritma :
    if dari = ke then
        output("Harap pilih 2 akun yang berbeda")
    else
        if DFSbutton.Checked then
            expDFS(matriks, daftarHuruf, dari, ke)
        else if BFSbutton.Checked then
            expBFS(matriks, daftarHuruf, dari, ke)
        endif
    endif
endif

```

4.2 Struktur Data

Struktur data yang digunakan untuk menggambarkan hubungan graf adalah matriks ketetanggan. Matriks berukuran $n \times n$, dengan n sebagai jumlah simpul unik dari file input. Matriks ini berupa array dua dimensi dari boolean yang elemen $[i,j]$ -nya akan bernilai *true* jika terdapat koneksi pertemanan antara simpul i dan j. Indeks elemen matriks memiliki urutan yang sama dengan urutan alfabetis dari semua simpul unik pada file. Daftar nama simpul yang telah diurutkan ini disimpan dalam sebuah *list of string*.

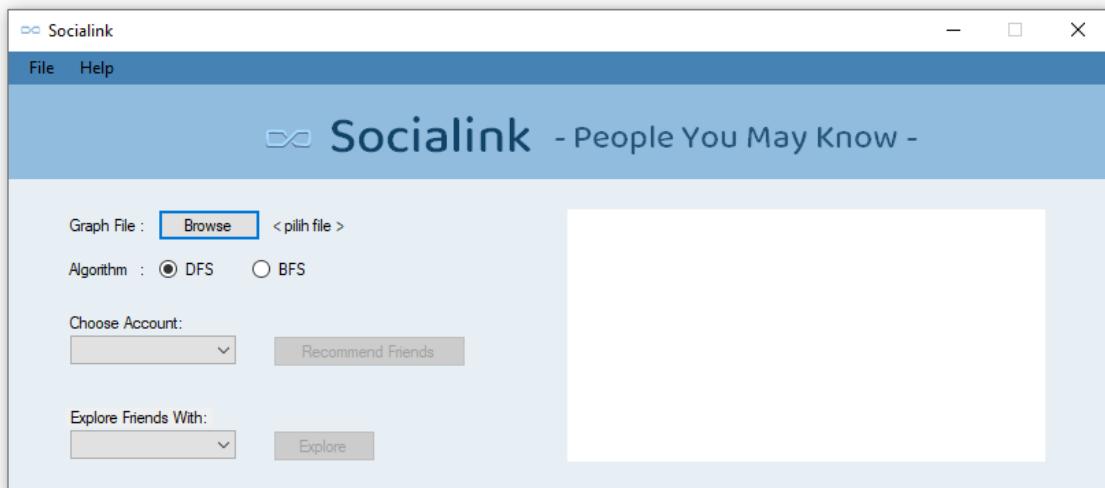
Selain itu, digunakan juga struktur *stack* untuk menyimpan jalur yang dilalui pada graf agar simpul awal bisa mencapai simpul yang dicari. Pada algoritma DFS, *stack* digunakan untuk menyimpan simpul yang diperiksa, dan *backtracking* dilakukan dengan melakukan *pop* pada *stack* dan melanjutkan memeriksa tetangga berikutnya dari elemen *top of stack* setelah melakukan *pop*. Pada saat pencarian, digunakan juga sebuah *array* yang menyimpan nilai *boolean* untuk menandai apakah sebuah simpul sudah dikunjungi. Proses pencarian dilakukan sampai *stack* kosong.

Untuk BFS, *stack* digunakan untuk membuat susunan jalur dari simpul awal ke simpul akhir. Digunakan juga *array* untuk menyimpan informasi apakah sebuah simpul sudah dikunjungi. Pada algoritma ini, digunakan sebuah *queue* untuk menyimpan urutan simpul yang akan diperiksa. Pada saat memeriksa simpul, semua tetangganya akan di-*enqueue* ke dalam *queue* tersebut. Setelah itu, simpul berikutnya yang akan diperiksa didapat dari hasil *dequeue*. Digunakan juga sebuah *array of integer* untuk menyimpan simpul yang digunakan untuk mencapai simpul tertentu. Proses pencarian dilanjutkan sampai *queue* kosong.

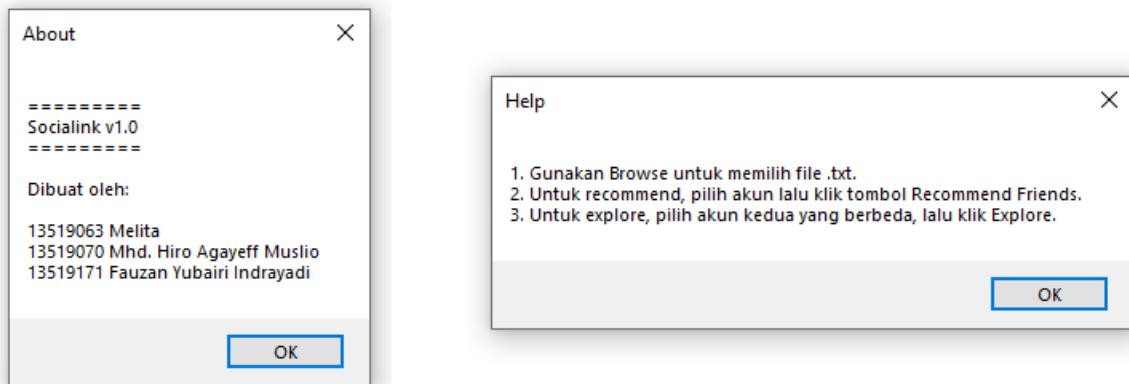
Dalam fitur *recommendation*, semua teman akun yang akan direkomendasikan teman dimasukkan ke dalam sebuah *list of integer*. Sebuah *array of integer* digunakan untuk menghitung jumlah *mutual friend* tiap akun yang belum berteman dengan akun tersebut.

4.3 Tata Cara Penggunaan Program

Tampilan dari program Socialink dapat dilihat pada gambar di bawah ini.

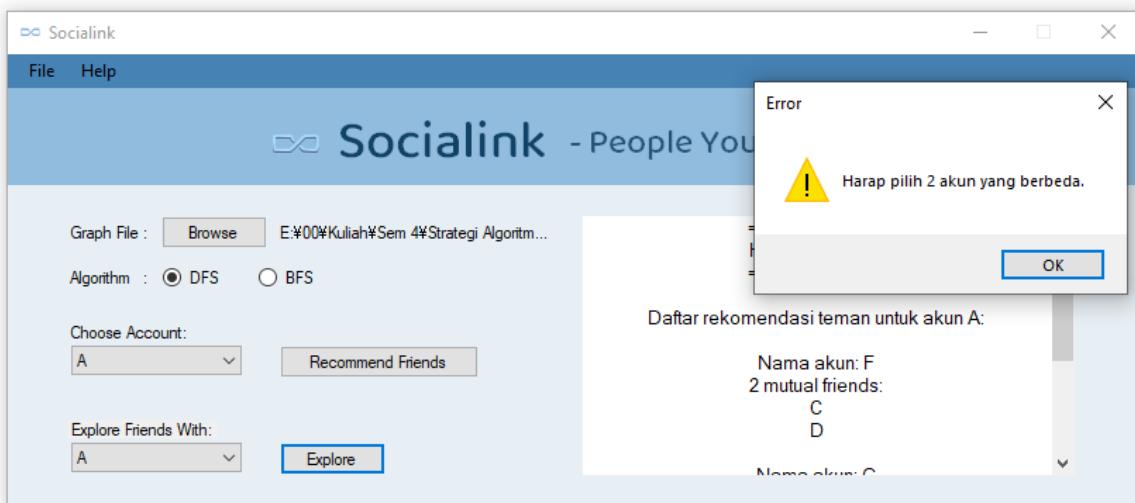


Pada menu, terdapat pilihan **File** dan **Help**. Pada File terdapat pilihan **Exit** yang dapat digunakan sebagai alternatif keluar dari program selain tombol *close* di bagian kanan atas *window*. Pada Help, terdapat pilihan **Guide** dan **About**. Guide memberikan panduan singkat cara menggunakan program dalam sebuah *message box*. Sementara itu, About memberikan keterangan nama dan versi program serta nama pembuat program.



Untuk menggunakan program, pilihlah *file* eksternal yang sudah dibuat sesuai ketentuan menggunakan **Browse**. Setelah itu, akan muncul sebuah *window* baru yang berisi visualisasi graf pertemanan. Menu *dropdown* untuk akun pertama dan kedua juga akan terisi dengan daftar nama semua akun yang ada.

Untuk mendapatkan rekomendasi teman, pilih akun yang diinginkan di bawah tulisan *choose account* lalu tekan tombol **Recommend Friends**. Hasil rekomendasi akan ditampilkan dalam *textbox* di bagian kanan program. Untuk mencari jalur antara dua akun, pilih algoritma yang akan digunakan dengan *radio button* DFS atau BFS, lalu pilih akun kedua di bawah tulisan *explore friends with*. Jika kedua akun sama, akan muncul *error message* seperti pada gambar berikut.



Jika dua akun berbeda, hasil pencarian akan muncul pada *textbox* bagian kanan. Jika tidak terdapat jalur antara kedua akun tersebut, akan muncul pesan “Tidak ada jalur koneksi yang tersedia.” Sementara itu, jika ditemukan jalur antara kedua akun tersebut, akan ditampilkan *connection degree* dan jalur yang diambil dalam bentuk tulisan pada *textbox*, dan akan ditampilkan juga graf baru dengan jalur yang dilalui berupa garis tebal berwarna merah.

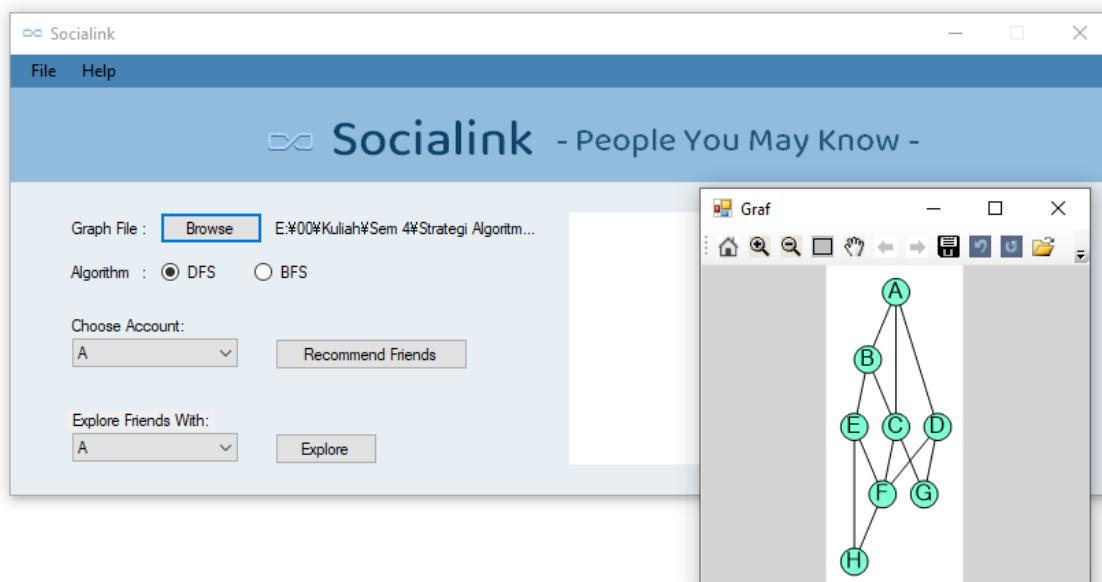
4.4 Hasil Pengujian

Berikut adalah dua *file* yang digunakan sebagai bahan pengujian program.

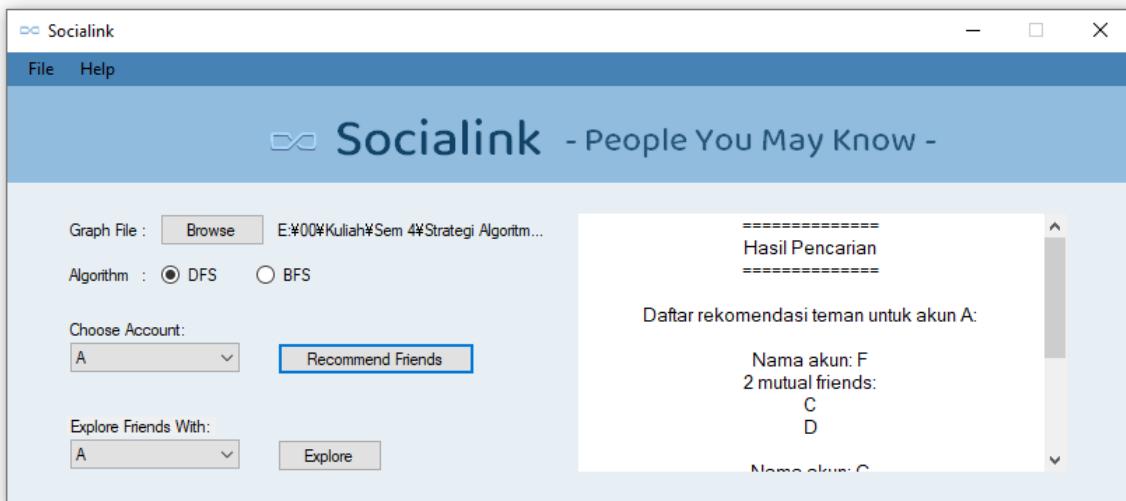
The image shows two separate windows of the Windows Notepad application. The left window, titled '01.txt - Notepad', contains the following text:
13
A B
A C
A D
B C
B E
C F
C G
D G
D F
E H
E F
F H

The right window, titled '02.txt - Notepad', contains the following text:
6
Budi Siti
Doni Eko
Siti Eko
Feli Budi
Budi Ali
Gio Heri

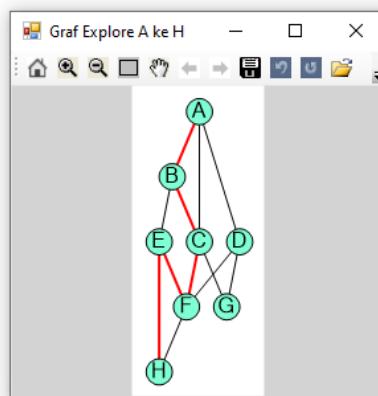
Hasil graf yang dibuat untuk 01.txt:



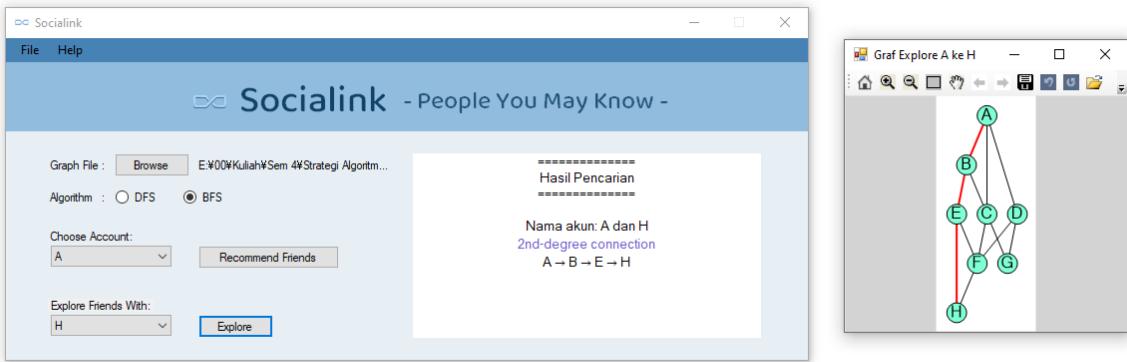
Contoh penggunaan rekomendasi teman untuk akun A:



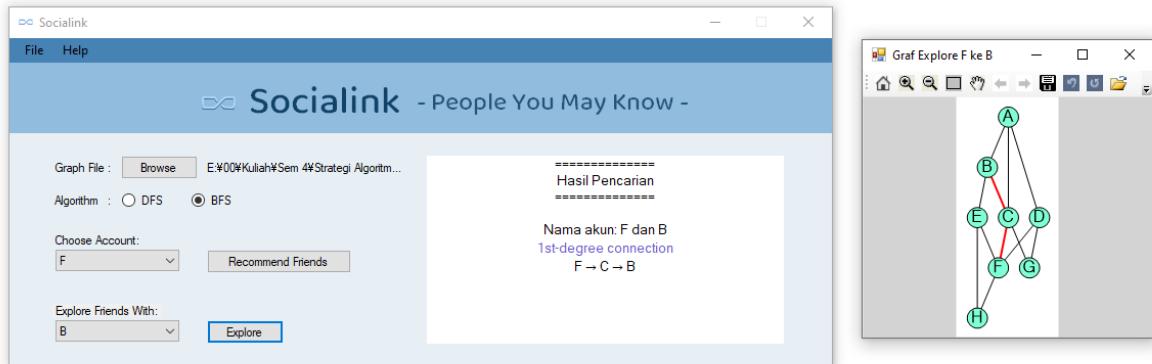
Contoh penggunaan *explore* antara akun A dengan H menggunakan DFS:



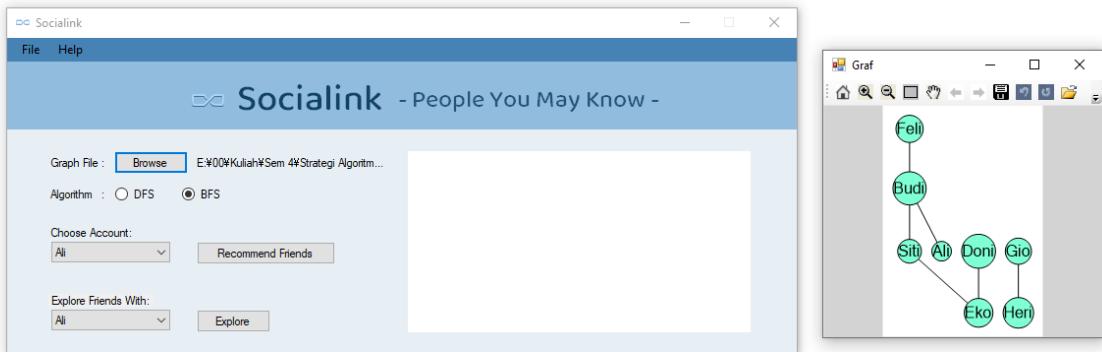
Contoh penggunaan *explore* antara akun A dan H menggunakan BFS:



Contoh penggunaan *explore* antara akun F dan D menggunakan BFS:



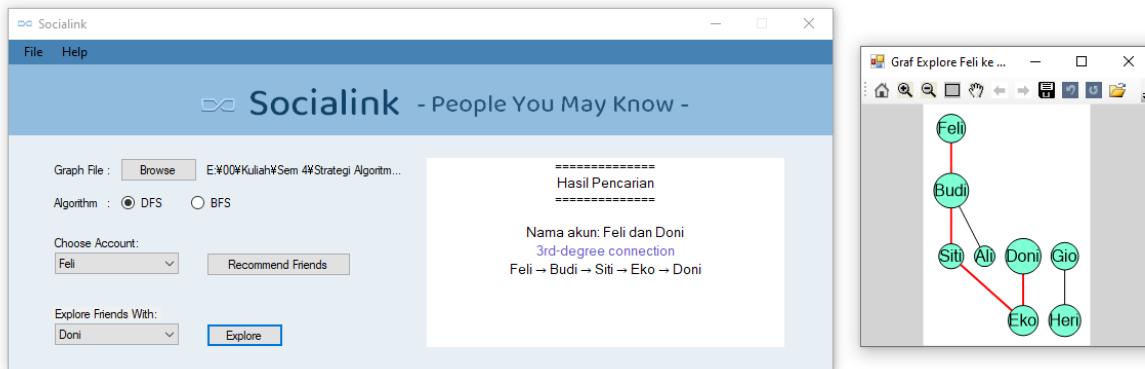
Hasil graf yang dibuat untuk 02.txt:



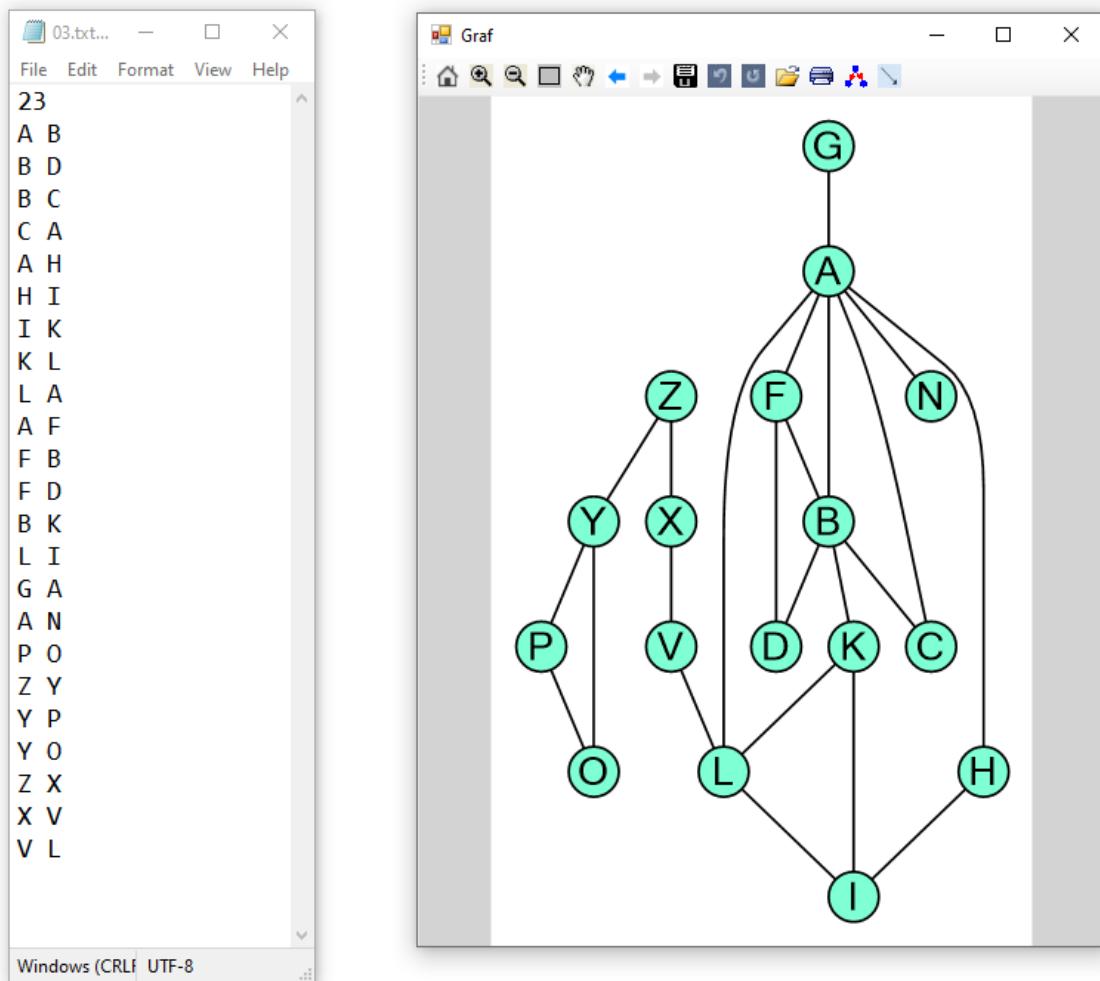
Contoh penggunaan *explore* antara Ali dengan Heri menggunakan BFS:



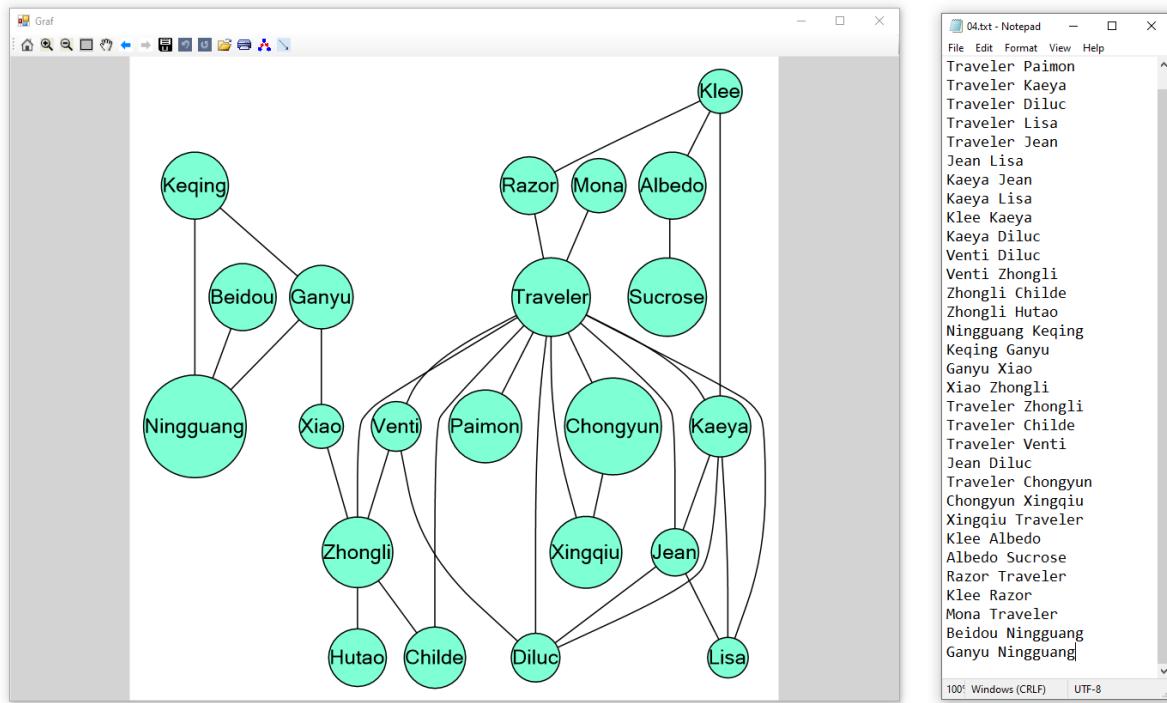
Contoh penggunaan *explore* antara Feli dengan Doni menggunakan DFS:



Contoh graf 03.txt:



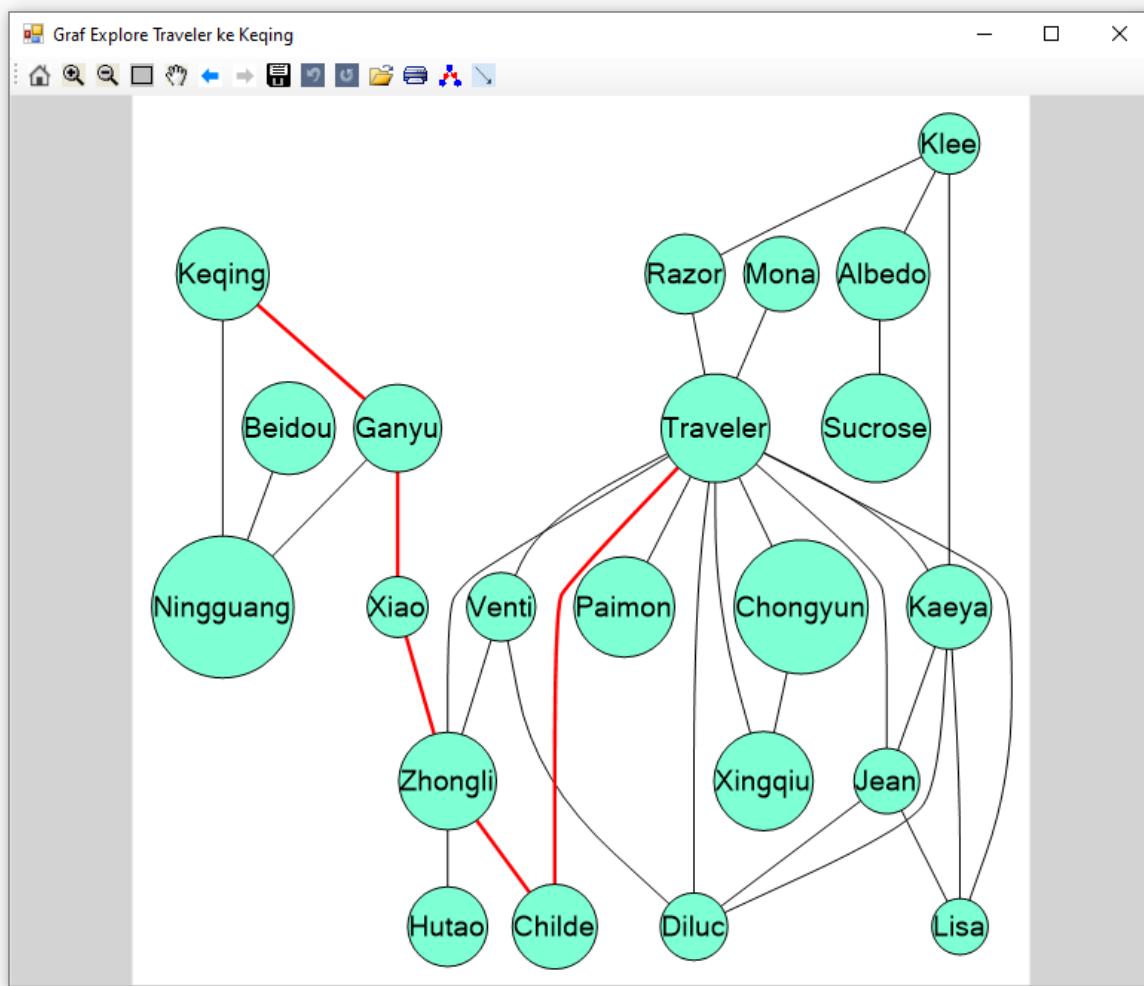
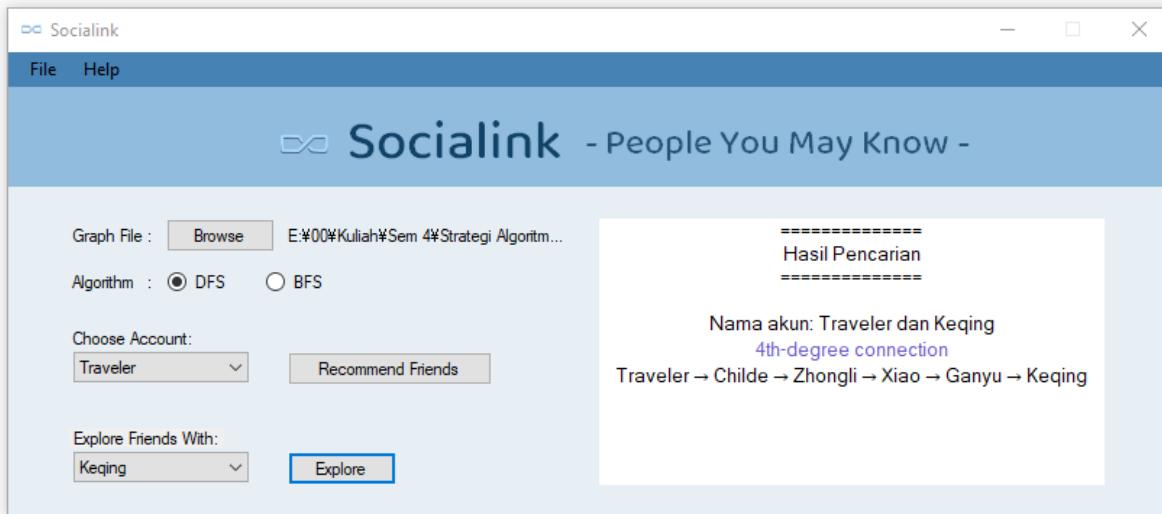
Contoh graf 04.txt:



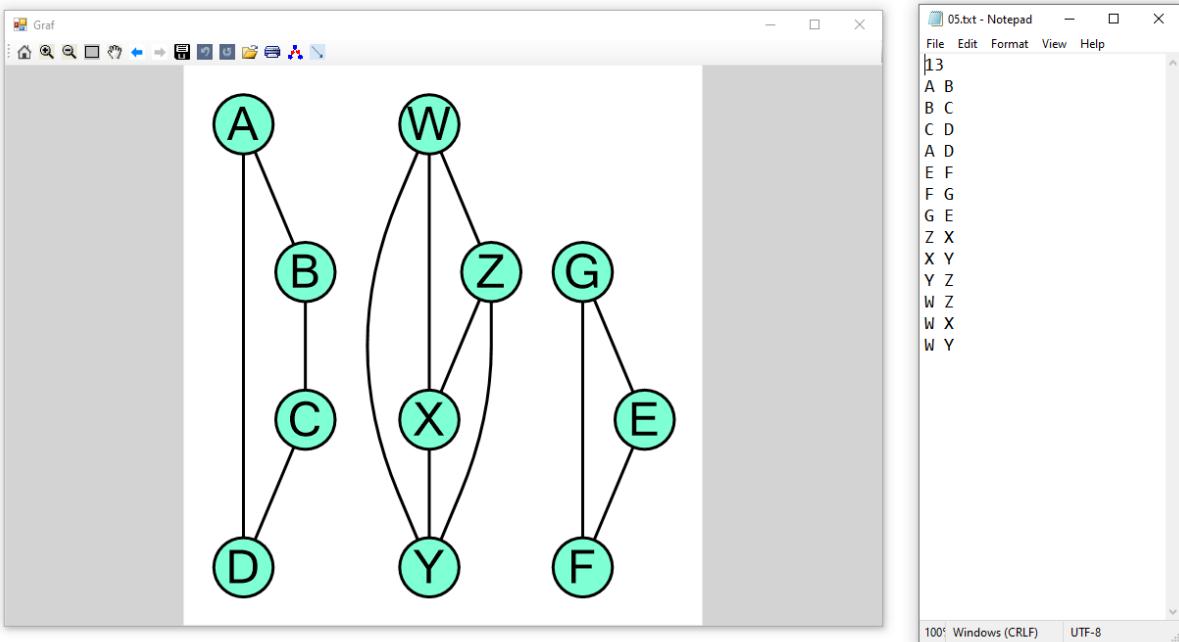
Contoh penggunaan rekomendasi teman untuk Zhongli:



Contoh *explore* Traveler dengan Keqing menggunakan DFS:



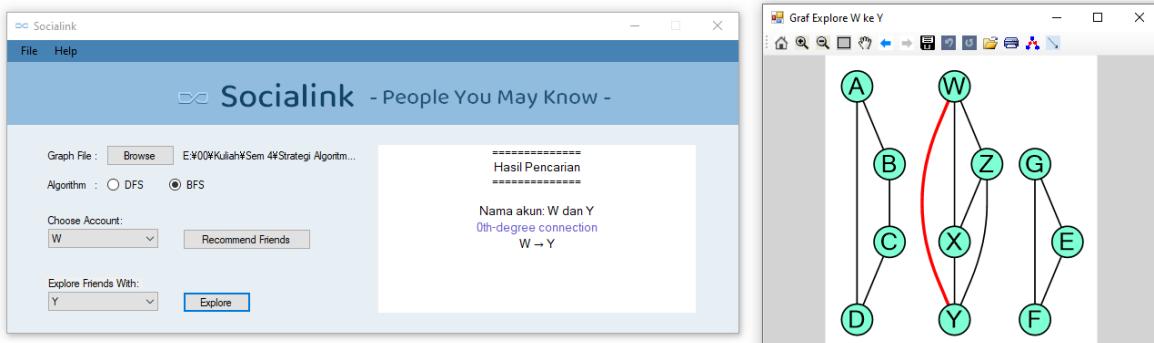
Contoh graf 05.txt:



Contoh *explore* antara E dengan A menggunakan DFS:



Contoh *explore* antara W dengan Y menggunakan BFS:



4.5 Analisis Desain Solusi

Algoritma BFS pasti menghasilkan jalur terpendek antara dua akun pada fitur *explore*, sedangkan algoritma DFS tidak. Hal ini dapat terjadi jika simpul-simpul yang dilalui memiliki lebih dari 2 teman, dan akan lebih terlihat jika setiap simpul memiliki banyak teman dan terhubung satu sama lain sehingga ada banyak jalur dari simpul awal ke simpul tujuan. Jika jalur yang tersedia antara simpul awal dan tujuan hanya satu, DFS dapat menghasilkan jalur minimum seperti BFS. Algoritma DFS dapat menemukan solusi lebih cepat daripada BFS jika simpul yang bukan merupakan jalur memiliki banyak tetangga dan jalur ke simpul tujuan hanya ada satu, karena dalam BFS program akan memeriksa semua tetangga terlebih dahulu.

Bab 5 Kesimpulan dan Saran

5.1 Kesimpulan

Pertemanan dalam media sosial *facebook* dapat digambarkan menggunakan graf. Simpul melambangkan akun, sedangkan sisi melambangkan adanya status pertemanan antara kedua akun. Ada beberapa algoritma pencarian dalam graf, di antaranya adalah DFS dan BFS. BFS pasti menghasilkan jalur minimal dari simpul awal ke tujuan, tetapi membutuhkan ruang memori yang lebih banyak daripada DFS.

5.2 Saran

Perilisan tugas besar sebaiknya dijadwalkan sesudah UTS agar mahasiswa tidak kaget dengan perilisan tugas besar yang bertubi-tubi (mata kuliah lain juga merilis tugas besar atau *milestone*). Perilisan tugas besar beriringan dengan suasana sebelum UTS. Hal tersebut disarankan oleh penulis agar mahasiswa dapat lebih fokus dengan UTS dan tidak membuang-buang tenaga dengan memikirkan tugas besar yang datang.

5.3 Refleksi dan Komentar

Refleksi:

- Tugas menambah pengetahuan dalam materi algoritma BFS dan DFS.
- Tugas menambah pengetahuan dalam pengimplementasian algoritma BFS dan DFS pada level pemrograman.
- Tugas menambah pengetahuan pemrograman menggunakan kakas Visual Studio .NET.
- Tugas menambah pengetahuan pemrograman dalam bahasa C#.

Komentar:

- Penjelasan deskripsi dan spesifikasi tugas pada spesifikasi sudah baik.
- Ilustrasi *interface* aplikasi pada spesifikasi sudah tergambar dengan jelas.

Daftar Pustaka

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Tugas-Besar-2-IF2211-Strategi-Algoritma-2021.pdf>

<http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag1.pdf>