

Tugas Besar I IF2211 Strategi Algoritma

Semester II Tahun 2020/2021

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

Oleh

13519085 Nizamixavier Rafif Lutvie

13519112 Pratama Andiko

13519211 Muhammad Alfandavi Aryo Utomo



SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2020/2021

Daftar Isi

BAB 1	3
BAB 2	7
BAB 3	8
BAB 4	12
BAB 5	19
Daftar Pustaka	20

BAB 1

Deskripsi Tugas

Dalam tugas besar ini, akan dibangun 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), aplikasi ini dapat menelusuri social network pada akun facebook untuk mendapatkan rekomendasi teman seperti pada fitur People You May Know. Selain untuk mendapatkan rekomendasi teman, aplikasi juga diminta untuk mengembangkan fitur lain agar dua akun yang belum berteman dan tidak memiliki mutual friends sama sekali bisa berkenalan melalui jalur tertentu.

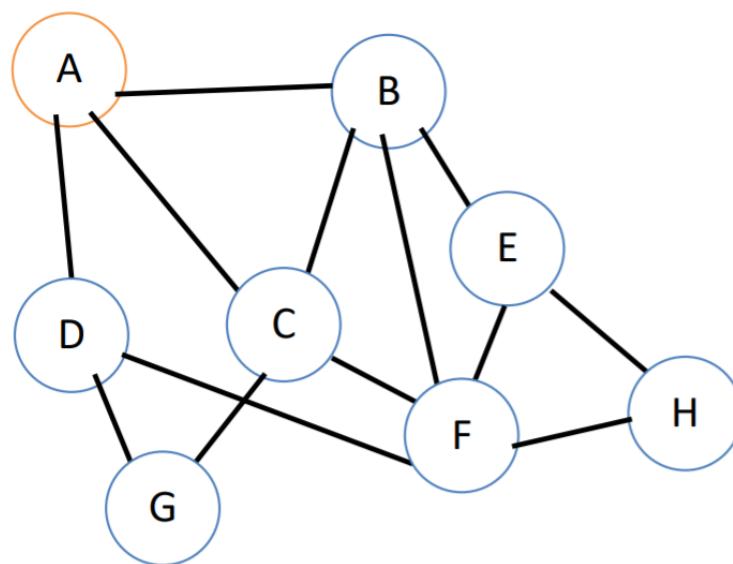
Contoh Input dan Output Program

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.

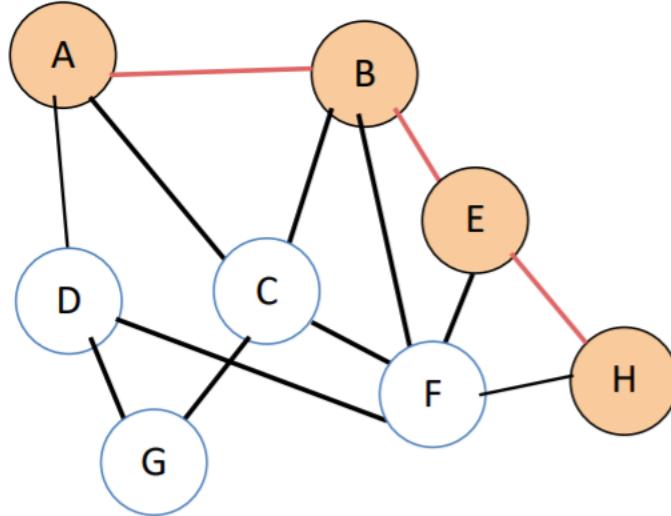
Nama akun: A dan H

2nd-degree connection

A → B → E → H

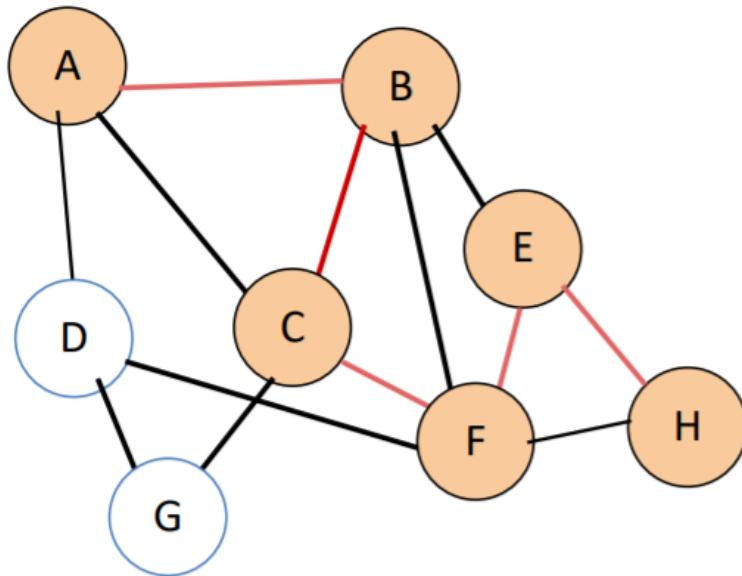
Gambar 4. Hasil output akun Nth degree connection akun A dan H menggunakan BFS

Perhatikan busur antara akun A dan H, terbentuk salah satu jalur koneksi sebagai berikut: A-BE-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

BAB 2

Landasan Teori

2.1 Dasar teori

Graf traversal mengacu pada proses mengunjungi tiap simpul dalam graf, ada beberapa macam cara pengunjungan namun kali ini akan dibahas strategi BFS dan DFS. BFS atau yang kita sebut dengan Breadth-First Search adalah suatu strategi yang mengunjungi semua simpul tetangga suatu simpul sebelum mengunjungi anak simpul tersebut, proses ini cenderung lebih lama dibandingkan DFS namun dijamin menemukan solusinya. Berkebalikan dengan BFS, DFS atau Depth-First Search adalah suatu strategi algoritma yang mengunjungi semua anak suatu simpul sebelum mengunjungi simpul tetangga, proses ini memang memiliki waktu lebih cepat namun tidak dijamin menemukan solusi.

2.2 Penjelasan singkat C# desktop application development

Desktop app ini ditulis dengan bahasa C# (C-Sharp) di dalam aplikasi visual studio, dengan menggunakan windows Form. Windows Form atau sering disebut dengan WinForm adalah Class library buatan microsoft yang bersifat GUI atau Graphical User Interface. Windows Form tergabung dalam Framework .Net. Tujuan diciptakan Win Form adalah untuk mempermudah developer dalam membuat suatu aplikasi berbasis desktop. Bayak sekali fitur yang disediakan oleh winform, seperti label, panel dan beberapa fitur lainnya yang menunjang developer dalam UI/UX aplikasi.

BAB 3

Analisis Pemecahan Masalah.

3.1 Langkah-langkah

Langkah-langkah pemecahan masalah yang kami gunakan yaitu, membuat graf 3 yang berisi data secara reversibel dari file eksternal sehingga setiap titik pada graf memiliki atribut next dan prev yang akan digunakan pada persoalan 1 dan 2. Persoalan 1 diselesaikan dengan mencari anak-node yang memiliki parent chooseAccount dan memiliki anak-node exploreWith, sehingga didapatkan hasilnya. Persoalan 2 diselesaikan dengan menggunakan prinsip BFS dan DFS secara biasa, algoritma BFS digunakan dengan mencari node yang belum dikunjungi secara melebar dan memasukkannya ke dalam antrian, algoritma DFS digunakan secara iteratif secara mendalam node-node yang belum dikunjungi sepanjang nilai tingkat yang telah didefinisikan pada parameter fungsi. Proses diawali dengan menyimpan data didalam array basis sebagai acuan enkripsi/dekripsi kita dan membuat array baru panduan2 sebagai array yang benar-benar diproses, misalkan di array basis berisi {b,c,d,e,z} maka panduan2 berisi {a,b,c,d,e} sesuai dengan urutan alfabet sehingga mudah dioperasikan dan tidak bermasalah dengan fungsi-fungsi seperti convert ke int dan converter ke char.

3.2 Proses mapping persoalan

Pada fitur explore friends, dapat digunakan 2 metode yaitu menggunakan algoritma BFS atau DFS.

Untuk algoritma BFS, berikut adalah langkah-langkah cara kerja algoritma:

1. Setiap pengguna dianggap sebagai simpul dan setiap hubungan antara 2 orang dianggap sebagai sisi.
2. Ditentukan simpul mana yang menjadi simpul mulai dan simpul tujuan
3. Tandai simpul mulai sebagai sudah dikunjungi
4. Masukkan simpul mulai ke dalam suatu queue sebagai penyimpan simpul untuk diproses
5. Keluarkan simpul tertua pada queue dan jadikan simpul tersebut sebagai simpul mulai
6. Dicek apakah simpul mulai adalah simpul tujuan. Jika iya, hentikan pemrosesan queue. Jika tidak, cek semua simpul yang terhubung ke simpul mulai. Untuk setiap simpul yang belum dikunjungi, tandailah simpul tersebut sebagai sudah dikunjungi dan catatlah simpul mulai sebagai simpul yang dikunjungi sebelumnya.
7. Masukkan simpul-simpul tersebut ke dalam queue.
8. Ulangi langkah 5 - 8 sampai simpul tidak memiliki percabangan yang bisa dikunjungi atau simpul tujuan telah ditemukan
9. Jika simpul tujuan tidak ditemukan, maka tidak ada solusi yang tepat untuk permasalahan
10. Jika simpul tujuan ditemukan, masukkan simpul tersebut ke dalam sebuah stack
11. Jika simpul yang baru dimasukkan ke dalam stack memiliki sebuah simpul yang dikunjungi sebelumnya (lihat langkah 6), masukkan simpul tersebut ke dalam stack
12. Ulangi langkah 11 sampai simpul yang baru dimasukkan tidak memiliki simpul yang dikunjungi sebelumnya

13. Keluarkan simpul satu per satu dari stack menggunakan metode pop untuk mendapatkan jalur koneksi dari simpul mulai menuju simpul tujuan

Untuk algoritma DFS, berikut adalah langkah-langkah cara kerja algoritma:

1. Setiap pengguna dianggap sebagai simpul dan setiap hubungan antara 2 orang dianggap sebagai sisi.
2. Ditentukan simpul mana yang menjadi simpul mulai dan simpul tujuan
3. Tandai simpul mulai sebagai sudah dikunjungi
4. Dicek apakah simpul mulai adalah simpul tujuan. Jika iya, hentikan proses pencarian. Jika tidak, lakukan lakukan pemrosesan rekursif dari langkah 3 - 4 untuk setiap simpul cabang yang belum dikunjungi. Untuk setiap langkah rekursif, catatlah urutan simpul yang dilewati
5. Proses rekursif berhenti jika tidak ada simpul cabang yang dapat diproses lagi
6. Jika terdapat satu di antara proses rekursif tersebut yang menemukan simpul tujuan, tuliskan hasil rute simpul berdasarkan urutan simpul yang dilewati oleh proses rekursif tersebut

3.3 Contoh ilustrasi

Contoh penerapan lain dari algoritma BFS/DFS adalah pada game 8 puzzle. Game 8 puzzle berbentuk matriks yang berordo 3 X 3. Di awal permainan tersebut disediakan angka 1-8 yang ditata secara acak dan hasil akhir dari permainan tersebut akan terbenut angka 1-8 yang urut mengelilingi matriks tersebut. Ilustrasi game tersebut dicontohkan pada gambar berikut.

2	1	6
4		8
7	5	3

1	2	3
8		4
7	6	5

(a) Susunan awal
(initial state) (b) Susunan akhir
(goal state)

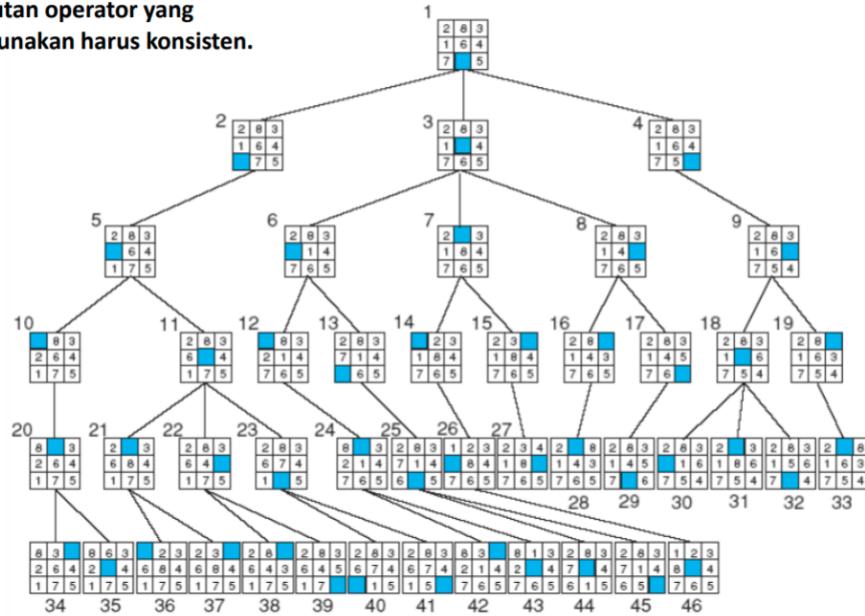
Sumber: file powerpoint BFS/DFS Rinaldi Munir

Game tersebut dapat diselesaikan dengan menggunakan algoritma BFS/DFS dengan menggunakan pohon status seperti gambar dibawah ini.

BFS untuk 8-Puzzle

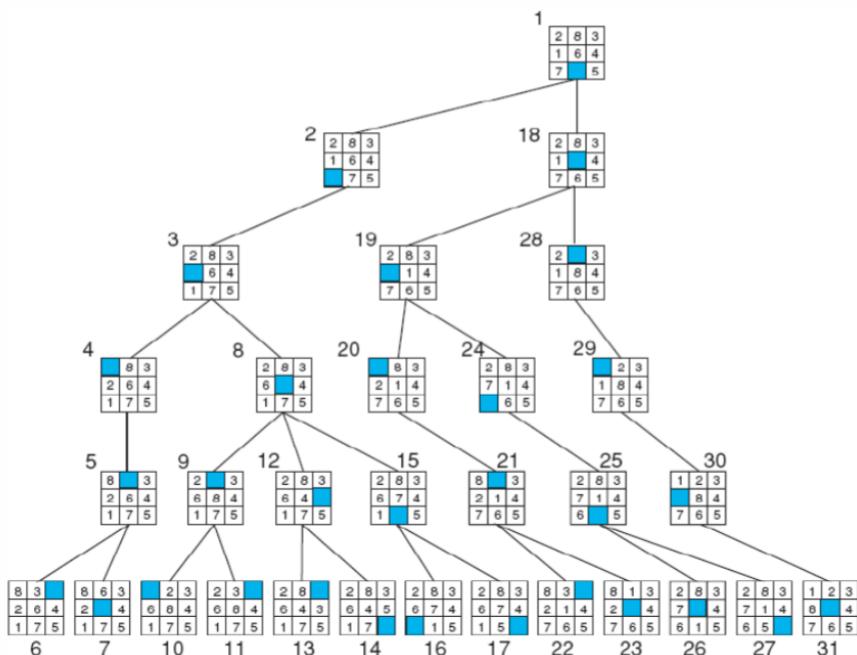
Catatan:

Urutan operator yang digunakan harus konsisten.



Sumber: file powerpoint BFS/DFS Rinaldi Munir

DFS untuk 8-Puzzle



Sumber: file powerpoint BFS/DFS Rinaldi Munir

Kedua algoritma dapat menemukan penyelesaian dari game tersebut, namun yang berbeda adalah tingkat kompleksitasnya. Dalam kasus game puzzle ini DFS akan lebih baik dalam kompleksitas ruang, namun kurang baik dalam kompleksitas waktu, karena pencarinya dilakukan secara mendalam ke daun terbawah. Sebaliknya, algoritma BFS lebih baik dalam kompleksitas waktu, namun kurang baik dalam kompleksitas ruang karena pencarinya dilakukan secara menyebar.

BAB 4

Implementasi dan Pengujian

4.1 Implementasi program

Pseudocode di bawah ini merupakan pseudocode untuk algoritma program utama kami, yang menjadi acuan dari baik fungsi/metode-metode lainnya

```
function main(){
    input(file)
    filepath = getPath(file)
    filename= getName(file)
    i traversal (1,len(file))
        arrayofChar1 = file[i][0]
        temp2 = string(arrayofChar1)
        arrayofChar2 = file[i][2]
        temp4 = string(arrayofChar2)
        insertTo(Basis,i)
    Basis.sort
    foreach a in Basis
        insertToCombobox(a,1)
        insertToCombobox(a,2)
    i traversal (len(Basis))
        insertTo(Panduan2,ConvertToChar(i+1))
    input(algoritma)
    input(akun)
    input(explorewith)
    countgraf = int(file[0][0])
    g3 = new graf(27)
    z traversal (1,countgraf)
        arrayofChar1 = file[z][0]
        temp2 = string(arrayofChar1)
        arrayofChar2 = file[z][2]
        temp4 = string(arrayofChar2)
        g3.addSimpul(temp2,temp4)
        g3.addSimpul(temp4,temp2)
    src = ConvertToInt(encrypt(akun,Basis,Panduan2))
    string1 = encrypt(explorewith,Basis,Panduan2)
    cond = false
    if string1.inPanduan2 then
        cond = true
    int2 = ConvertToInt(string1)
    if (cond) then
```

```

        string2 = ConverterToChar(int2)
        output(NamaAkun)
        Soal1(g3, src, pelakor2, countgraf)
    else
        output(akunNotFound)
    dest = ConverterToInt(string1)
    temp = Soal1TanpaOutput(g3, src, pelakor2, countgraf)
    initTuple(data)
    i traversal(len(panduan2))
        dest2 = ConverterToInt(panduan2[i])
        temp2 = Soal1TanpaOutput(g3, src, pelakor2, countgraf)
        data.add(initTupleV2(i+1,temp2))
    data.sort(desc)
    i traversal (len(data))
        if (data[i].Item1 /= int2 && data[i].Item1 /= src) then
            output(NamaAkunYangSudahDiDecrypt)
            Soal1(g3, src, data[i].Item1, countgraf)
    if (algoritma = DFS) then
        awal = ConverterKeInt(encrpyt(akun,basis,panduan2))
        akhir = ConverterKeInt(encrpyt(explorewith,basis,panduan2))
        initarray(dikunjungi)
        initarraydikunjungi(res)
        res[0] = awal
        hasil = Soal2DFS(g3, awal, akhir, dikunjungi, result, 0, basis, panduan2)
        if (hasil = false) then
            output(NoConnection)
    else // ini BFS
        awal3 = ConverterKeInt(encrpyt(akun,basis,panduan2))
        akhir4 = ConverterKeInt(encrpyt(explorewith,basis,panduan2))
        Soal2BFS(g3, awal3, akhir4, 27, basis, panduan2)
    inithashSet
    outputEach(hashSet)
    z traversal countgraf
        c1 = file[z][0]
        s1 = string(c1)
        ss1 = decrypt(s1,basis,panduan)
        c2 = file[z][2]
        s2 = string(c2)
        ss2 = decrypt(s2,basis,panduan)
        graf.addEdge(ss1,ss2)
    z traversal countgraf
        c1 = file[z][0]
        s1 = string(c1)

```

```

ss1 = decrypt(s1,basis,panduan)
c2 = file[z][2]
s2 = string(c2)
ss2 = decrypt(s2,basis,panduan)
if (hashSetcontainss1Danss2) then
    graf.addEdge(ss1,ss2).kasiWarnaHijau
grafKasiWarnaNode(chooseAccount).kasiWarnaHijau
grafKasiWarnaNode(exploreWith).kasiWarnaHijau dan Bentuk Diamond
tampilkanGraf
}

```

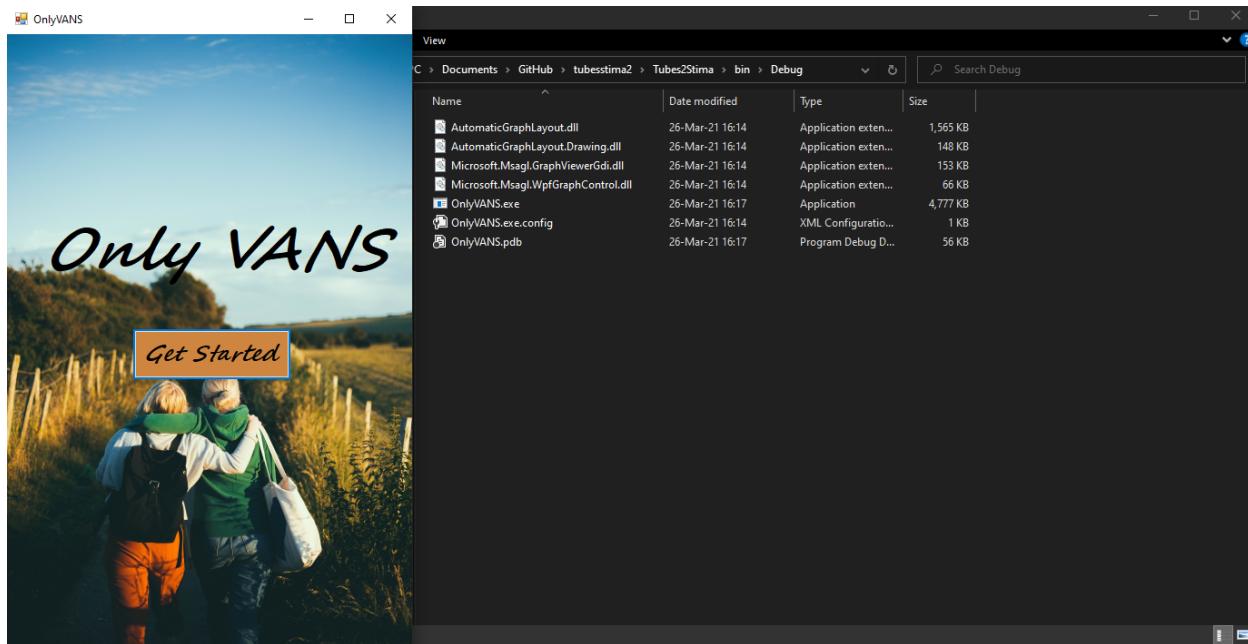
4.2 Penjelasan struktur data

Pada program ini kami, tidak menggunakan berbagai variasi struktur data yang bermacam-macam. Struktur data yang kami gunakan ada dua jenis, yaitu:

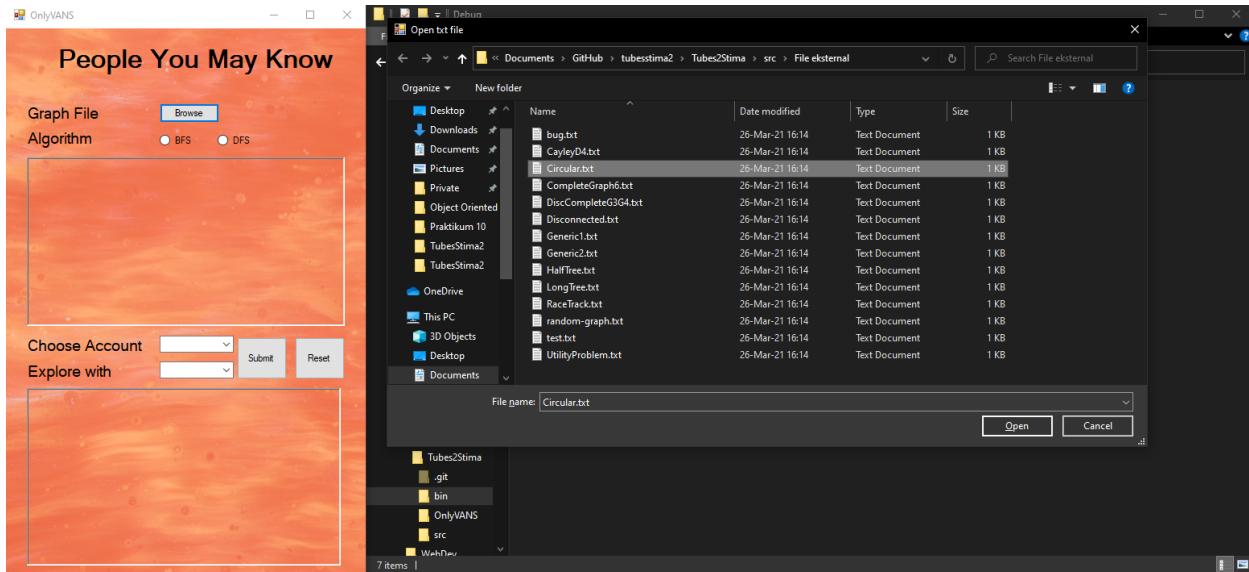
1. List: digunakan untuk menampung data dari file external dan acuan sebagai perubahan nantinya, menampung path yang ditemukan oleh algoritma,
2. Graf: digunakan untuk proses data utama dan visualisasi
3. Array: digunakan pada setiap proses khususnya iterasi yang diperlukan
4. Queue: digunakan pada algoritma BFS
5. Stack: digunakan pada algoritma BFS
6. Set: Berupa hash set yang digunakan untuk menyimpan path/jalur algoritma

4.3 Penjelasan tata cara penggunaan program

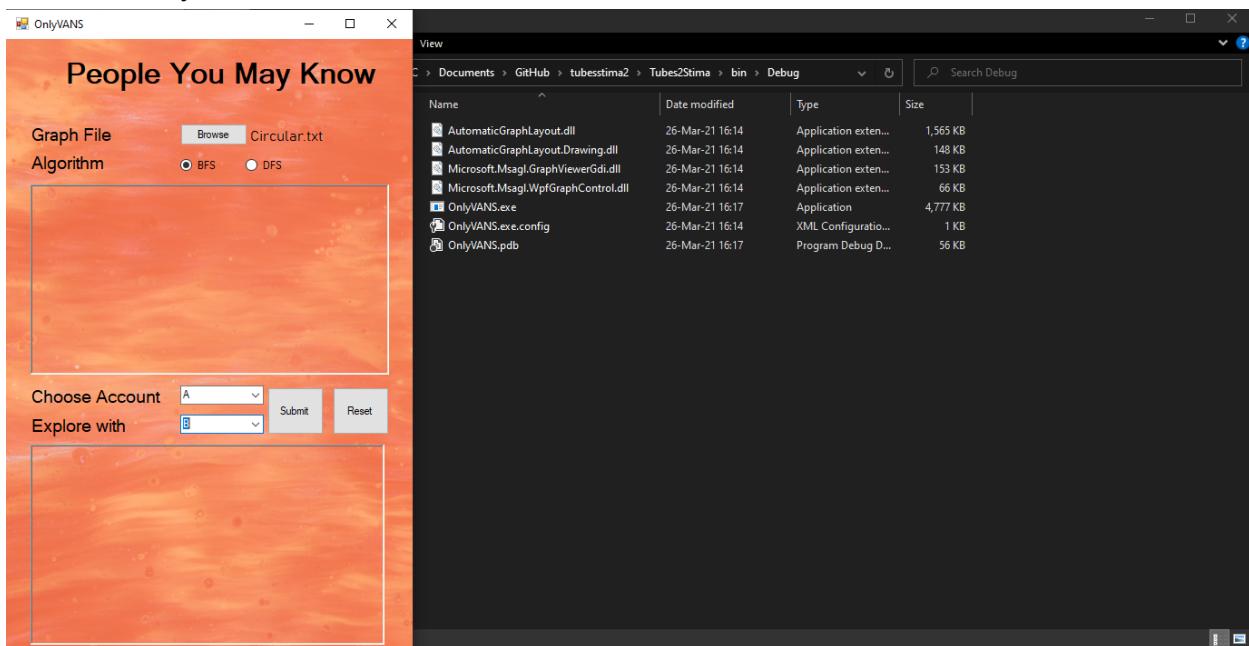
1. Ubah direktori folder ke bin/debug dan run OnlyVANS.exe



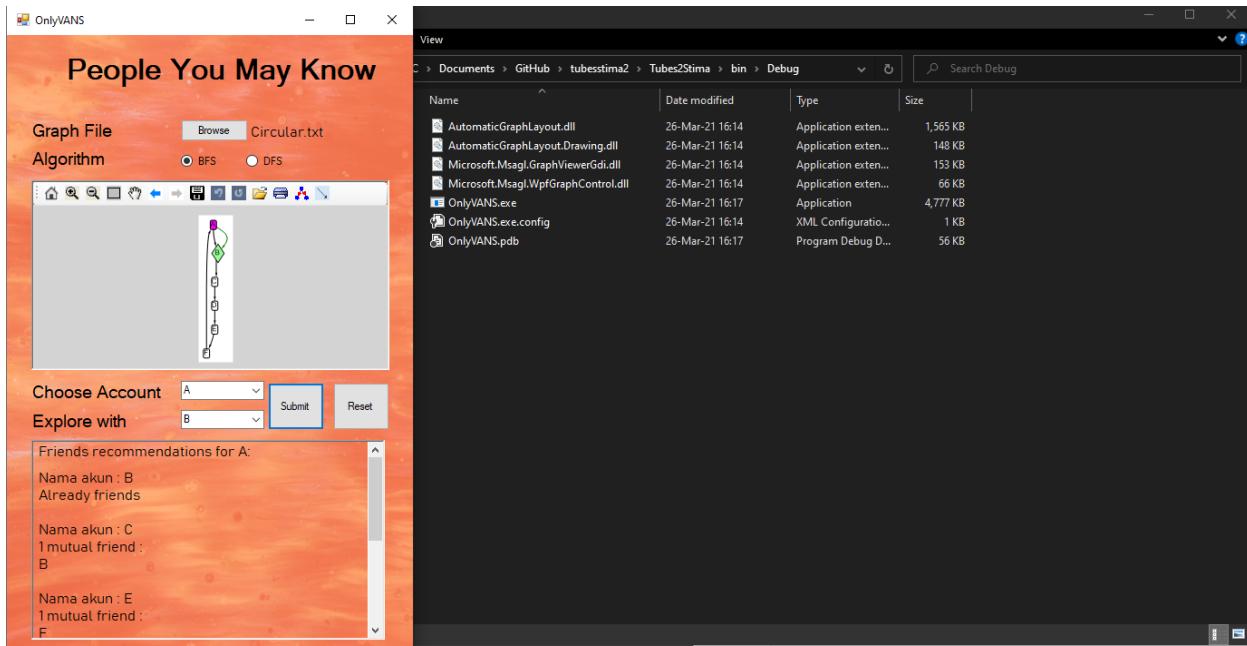
2. Pilih file eksternal berupa txt berisi data graf yang akan digunakan



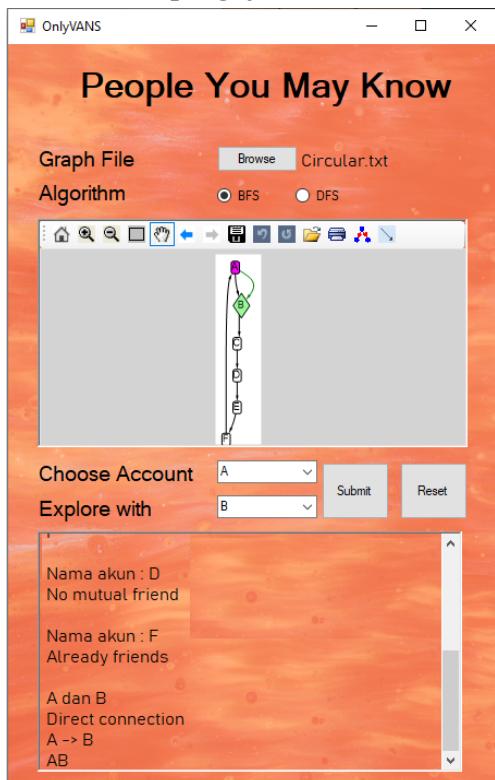
- Pilih algoritma yang ingin digunakan (DFS atau BFS) dan akun yang ingin ditelusuri beserta acuannya dan klik submit

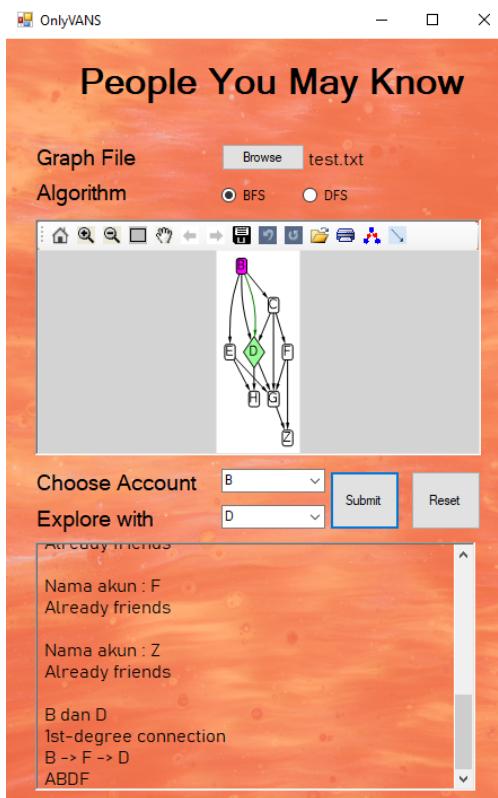
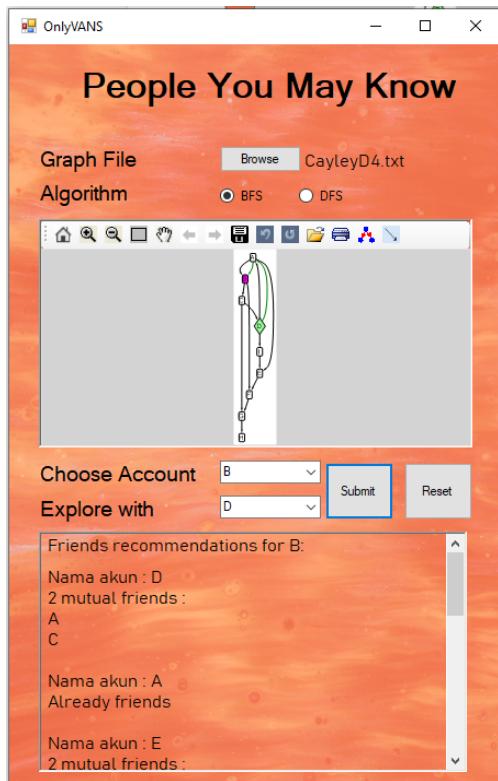


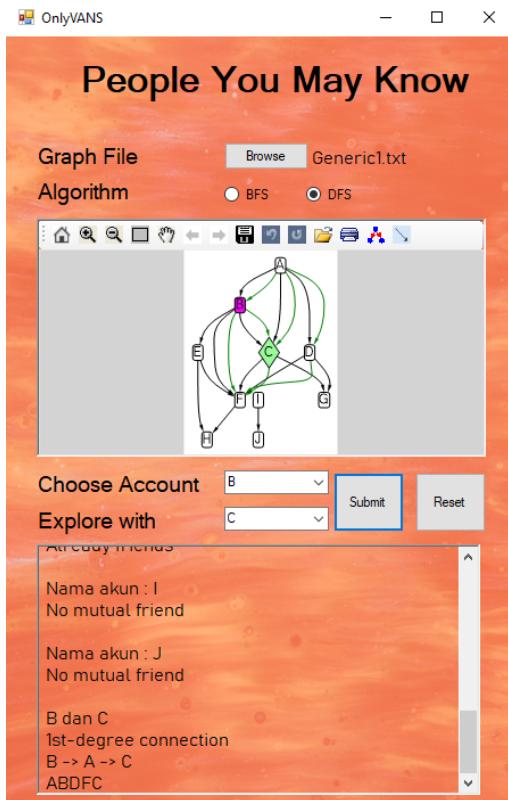
- Hasil akan keluar dan apabila ingin mereset tinggal tekan tombol reset



4.4. Hasil pengujian







4.5. Analisis Desain

Dari hasil percobaan berbagai test case yang dilakukan, selalu ditemukan solusi dari permasalahan baik menggunakan algoritma DFS maupun BFS. Kedua algoritma tidak dapat dibandingkan dari segi kecepatan karena waktu untuk menemukan solusinya rata-rata hanya sepersekian detik. Akan tetapi dengan melihat kompleksitas algoritma yang dimiliki oleh kedua algoritma tersebut, terdapat perbedaan kompleksitas tergantung dari test case yang diberikan. Jika pada test case yang diberikan, titik tujuan pada representasi graf memiliki kedalaman yang tinggi relatif terhadap titik mulai, kompleksitas dari algoritma BFS cenderung lebih besar daripada kompleksitas algoritma DFS. Sebaliknya, jika pada test case yang diberikan, titik tujuan pada representasi graf memiliki jarak yang pendek relatif terhadap titik mulai, kompleksitas dari algoritma DFS cenderung lebih besar daripada kompleksitas algoritma BFS. Hal ini berarti algoritma BFS lebih efektif apabila titik tujuan yang dicari persoalan memiliki kedalaman yang tinggi relatif terhadap titik mulai. Sedangkan algoritma BFS lebih efektif apabila titik tujuan yang dicari persoalan berjarak dekat secara tingkat terhadap titik mulai.

BAB 5

Kesimpulan, Saran, Refleksi, dan Komentar

Algoritma DFS dan BFS dapat digunakan untuk menyelesaikan permasalahan berbasis graf statis. Meskipun kedua algoritma tidak selalu menghasilkan solusi tercepat atau terpendek, keduanya akan selalu memberikan solusi terhadap permasalahan. Algoritma DFS dan BFS memiliki keunggulan masing-masing tergantung dari posisi titik awal dan titik akhir. Jika graf digambarkan secara bertingkat dimulai dari titik awal, algoritma DFS lebih efektif jika titik yang dicari berjarak jauh dengan titik awal, sedangkan algoritma BFS lebih efektif jika titik yang dicari berjarak dekat dengan titik awal.

Penulis menyadari bahwa masih terdapat banyak kekurangan dalam pembuatan laporan ini. Penulis berharap bahwa persoalan dapat dijelajahi lebih lanjut agar dapat ditemukan solusi yang lebih baik dan efisien untuk menyelesaikan permasalahan soal.

Daftar Pustaka

Munir, Rinaldi. Algoritma Greedy (Bagian 1).

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-backtracking-2021-Bagian1.pdf>

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

Diakses pada 25 Maret 2021.