

# **LAPORAN TUGAS BESAR 2**

Mata Kuliah IF2211 Strategi Algoritma

Dosen Pengampu: Rinaldi Munir, Nur Ulfa Maulidevi

Dwi Hendratmo Widyantoro, Rila Mandala



Disusun Oleh:

Alexander	13519090
Girvin Junod	13519096
Josep Marcello	13519164

**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**2021**

# **BAB I**

## **DESKRIPSI TUGAS**

Dalam tugas besar ini, diminta untuk membangun sebuah aplikasi GUI sederhana yang dapat memodelkan beberapa fitur dari *People You May Know* dalam jejaring sosial media (*Social Network*). Pengguna dapat menelusuri *social network* untuk mendapatkan rekomendasi teman (fitur *Friend Recommendation*) seperti pada fitur *People You May Know*. Selain itu, dengan memanfaatkan algoritma *breadth-first search* (BFS) dan *depth-first search* (DFS), terdapat juga fitur *Explore Friends* yang menunjukkan koneksi yang ada antara dua akun. Pada fitur *Friend Recommendation*, digunakan algoritma BFS atau DFS sehingga pengguna dapat mendapatkan rekomendasi teman untuk akun terpilih berdasarkan mutual friends yang ada antara akun terpilih dan akun-akun yang direkomendasikan itu. Pada fitur *Explore Friends*, pengguna dapat mengetahui seberapa jauh jarak antara dua akun yang dipilih dan jalur agar kedua akun tersebut dapat terhubung.

## BAB II

### LANDASAN TEORI

#### 2.1 *Graph Traversal*

Graf adalah suatu struktur yang melambangkan sebuah kumpulan objek dan keterhubungan antar objek tersebut. Suatu objek dalam graf digambarkan dengan sebuah node dan hubungan antara kedua node digambarkan dengan suatu sisi. Dalam bidang informatika, graf sering digunakan untuk memodelkan suatu *network* yang dalam kasus tugas ini berupa *social network*.

*Graph traversal* atau *graph search* adalah suatu proses mengunjungi setiap node dalam suatu graf. Yang dimaksud dengan mengunjungi di sini adalah aksi mengecek suatu node dan melakukan perubahan terhadap node tersebut sesuai kebutuhan. *Graph traversal* diklasifikasikan dari urutan pengunjungan nodenya. Algoritma yang berbeda akan menghasilkan urutan pengunjungan node yang berbeda. Oleh karena itu, klasifikasi *graph traversal* biasanya dilakukan berdasarkan algoritmanya. Pada tugas ini, digunakan algoritma BFS dan DFS untuk melakukan *graph traversal*.

#### 2.2 **BFS**

Algoritma *breadth-first search* (BFS) adalah suatu algoritma traversal untuk graf. Pada algoritma ini, traversal dimulai pada suatu node awal yang dikunjungi lalu dipilih, lalu mengunjungi semua node tetangga dari node terpilih yang memiliki depth yang sama, sebelum beralih ke node yang ada di depth level selanjutnya. Algoritma BFS pada umumnya dapat digambarkan dengan sebuah queue. Untuk tiap node yang dikunjungi dalam BFS node tersebut dicek apakah sudah pernah masuk ke queue dan jika belum maka dimasukkan ke queue. Node yang dipilih adalah node yang di-dequeue dari queue tersebut. Hal ini dilakukan terus sampai queue habis dan tidak ada lagi node yang masuk ke queue atau tercapai tujuan dari traversal.

#### 2.3 **DFS**

Algoritma *depth-first search* (DFS) adalah suatu algoritma traversal untuk graf. Pada algoritma ini, traversal dimulai pada suatu node awal yang dikunjungi lalu dipilih. Lalu, dikunjungi dan dipilih satu node tetangga dari node awal tersebut yang belum pernah dikunjungi. Dari node

tersebut dikunjungi dan dipilih lagi satu dari node tetangganya dan berulang terus sampai pada node yang dipilih tidak ada lagi node tetangga yang bisa dikunjungi. Setelah itu, dilakukan backtrack ke node-node sebelumnya sampai ada node tetangga yang masih dapat dikunjungi dan dipilih. Proses ini diulang terus sampai semua node telah dikunjungi atau tercapai tujuan dari traversal. Algoritma ini dapat dibuat secara iteratif atau rekursif. Untuk cara iteratif, digunakan stack untuk memodelkan urutan pengunjungan node. Tiap node yang dipilih adalah node yang di-pop dari stack. Untuk tiap node yang di-pop, dipush semua node tetangga dari node tersebut ke stack yang belum pernah di-pop. Untuk cara rekursif, dari node terpilih dilakukan pemanggilan rekursif dari fungsi DFS lagi untuk tiap node tetangga dari node terpilih yang belum pernah dipilih.

## 2.4 Pengembangan Aplikasi Desktop C#

Aplikasi Windows Penulis dikembangkan dengan bahasa C# menggunakan .NET sebagai kerangka kerja (*framework*)-nya dan menggunakan perangkat lunak Visual Studio serta Visual Studio Blend untuk menulis program dari aplikasinya serta untuk mendesain GUI-nya.

## BAB III

### ANALISIS PEMECAHAN MASALAH

#### 3.1 Langkah-Langkah Pemecahan Masalah

- Pembuatan Graph dari File eksternal
  - File Eksternal dibaca dan akan dilakukan parsing sehingga terbentuk suatu graf sesuai dengan struktur data.
- Pembuatan algoritma DFS dan BFS untuk pencarian di graph
- Pembuatan algoritma BFS atau DFS untuk fitur friend recommendations
  - Algoritma friend recommendation bekerja dengan mencari semua teman dari teman akun terpilih yang bukan teman dari akun terpilih, lalu diurutkan dari jumlah mutual friends terbanyak. Algoritma friend recommendation kelompok penulis merupakan variasi algoritma BFS.
- Implementasi DFS dan BFS pencarian untuk pembuatan fitur explore friends
  - DFS dan BFS digunakan untuk mencari jalur dari dua node yang dipilih di fitur explore friends.
- Mengimplementasikan hasil dari fungsi-fungsi ke GUI
  - Dihubungkannya GUI dengan algoritma yang ada, sehingga ketika fungsi fungsi tersebut dijalankan, hasilnya akan ditampilkan GUI dengan graf yang divisualisasikan dengan bantuan MSAGL.

#### 3.2 Proses Mapping persoalan menjadi elemen-elemen algoritma BFS dan DFS

- Setiap akun dibuat menjadi node di dalam graph
- Jika ada akun yang berteman dengan akun lain, maka ada sisi antara dua node yang merepresentasikan akun itu
- Akun yang dipilih dijadikan sebagai node mulai untuk BFS dan DFS
- Akun yang dipilih sebagai target fitur explore friends menjadi node target dari pencarian BFS dan DFS

#### 3.3 Ilustrasi Kasus Lain

Berikut merupakan ilustrasi dari kasus lain.

Berkas file eksternalnya. Tiap nama melambangkan sebuah node, tiap spasi diantara dua nama melambangkan sisi antara kedua node tersebut.

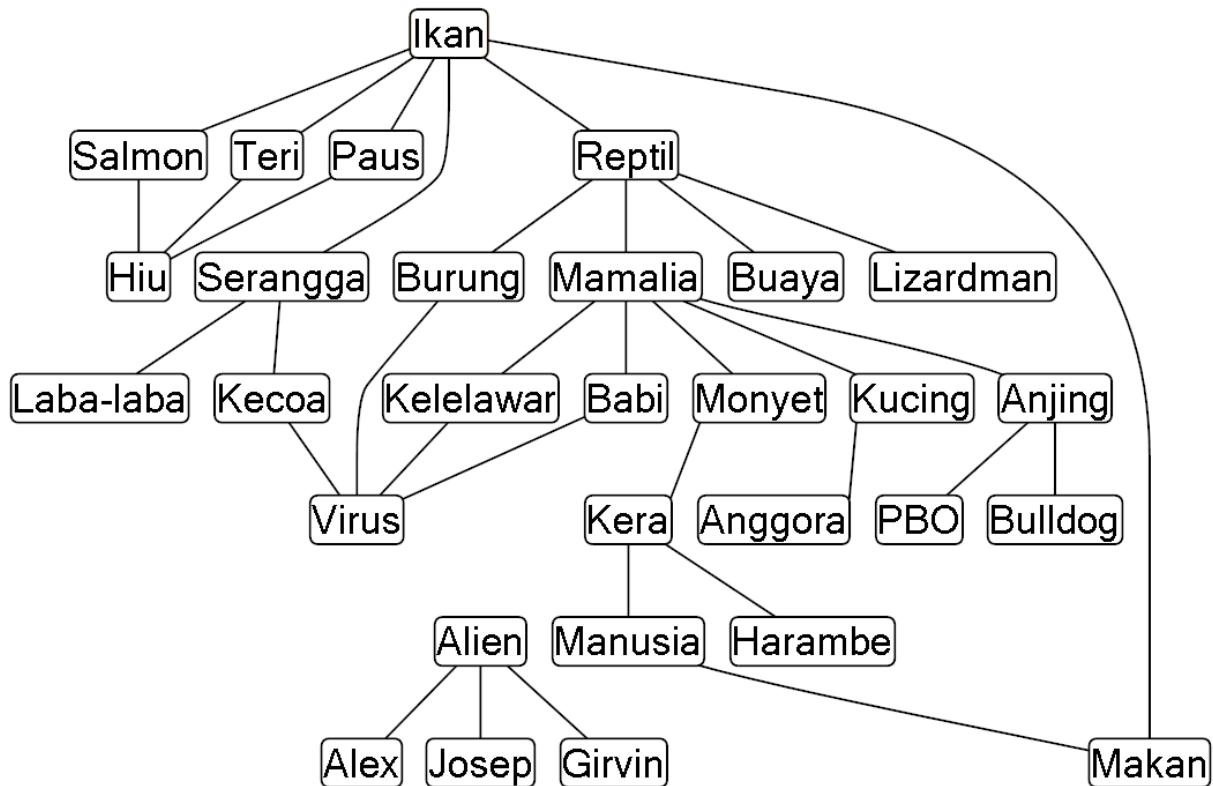
Ikan Reptil	Paus Hiu	Anjing Bulldog
Ikan Paus	Reptil Burung	Kucing Anggora
Ikan Teri	Reptil Mamalia	Reptil Buaya
Ikan Salmon	Mamalia Kucing	Mamalia Babi
Salmon Hiu	Mamalia Anjing	Ikan Serangga
Teri Hiu	Anjing PBO	Serangga Laba-laba

Serangga Kecoa  
Mamalia Monyet  
Monyet Kera  
Kera Manusia  
Kera Harambe  
Reptil Lizardman

Mamalia Kelelawar  
Kelelawar Virus  
Babi Virus  
Burung Virus  
Kecoa Virus  
Alien Josep

Alien Alex  
Alien Girvin  
Ikan Makan  
Manusia Makan

Dari input file ini, didapat graf berupa:



Untuk fitur Friend Recommendation, misal dipilih node Ikan dari graf, maka akan didapatkan hasil seperti:

Friend Recommendation(s):

=> Hiu

3 Mutual Friend(s):

Paus, Salmon, Teri

=> Laba-laba

1 Mutual Friend(s):

Serangga

=> Kecoa

1 Mutual Friend(s):

Serangga

=> Mamalia

1 Mutual Friend(s):

Reptil

=> Lizardman

1 Mutual Friend(s):

Reptil

=> Burung

1 Mutual Friend(s):

Reptil

=> Buaya

1 Mutual Friend(s):

Reptil

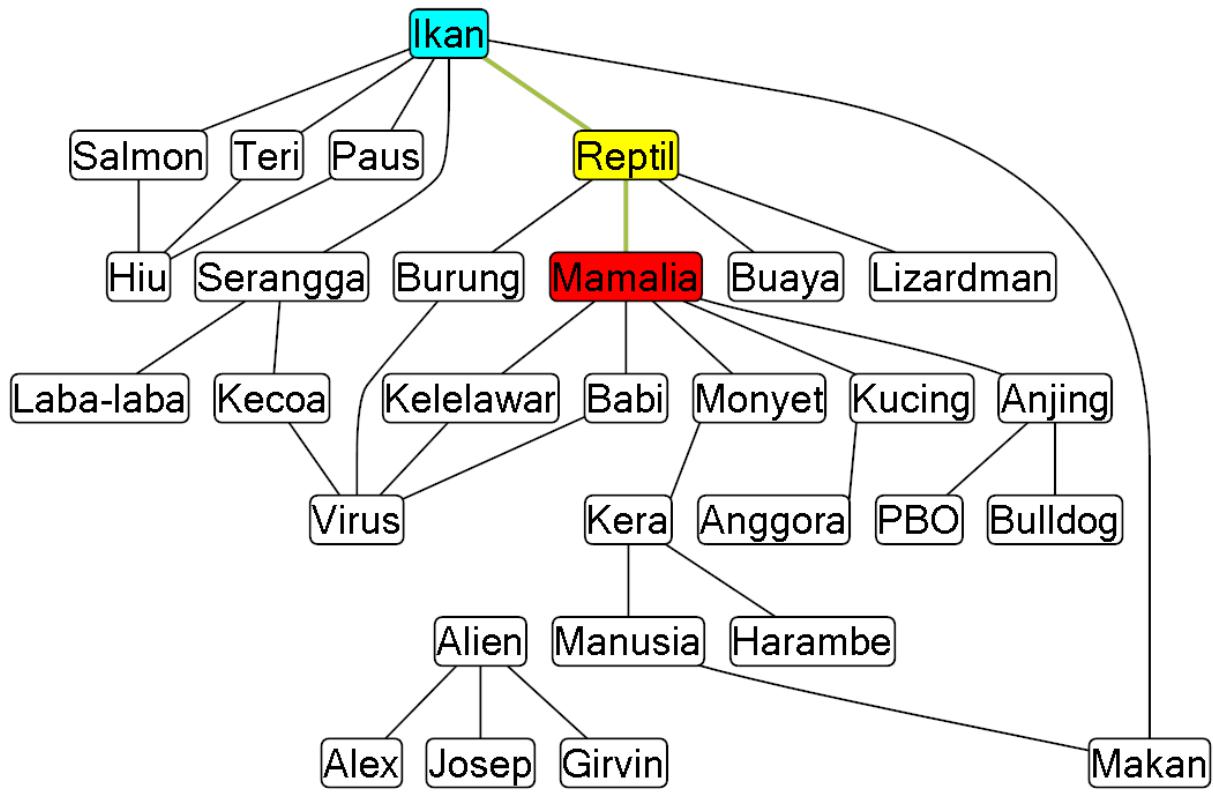
=> Manusia

1 Mutual Friend(s):

Makan

Hasil ini didapat dari melihat node apa saja yang memiliki mutual friends dengan node Ikan namun bukan merupakan teman dari node Ikan. Dalam kata lain, fitur friend recommendations mencari node yang merupakan tetangga dari tetangganya node terpilih dan bukan merupakan tetangga dari node terpilih. Node-node ini akan diurutkan dari jumlah mutual friends atau banyaknya node tetangga akun terpilih yang memiliki sisi dengannya. Dalam kasus ini ada 8 friend recommendations, dimulai dengan Hiu yang ada 3 mutual friends yaitu Paus, Salmon dan Teri, lalu dilanjutkan oleh Laba-laba dan Kecoa yang sama-sama memiliki 1 mutual friend yaitu Serangga, lalu dilanjutkan oleh Mamalia, Lizardman, Burung, Buaya yang sama-sama memiliki 1 mutual friend yaitu Reptil, dan diakhiri dengan Manusia yang memiliki 1 mutual Friend yaitu Makan.

Untuk fitur Explore Friends, dilakukan pencarian melalui algoritma BFS dan DFS, berikut adalah hasil untuk pencarian dari node Ikan ke node Mamalia melalui BFS.



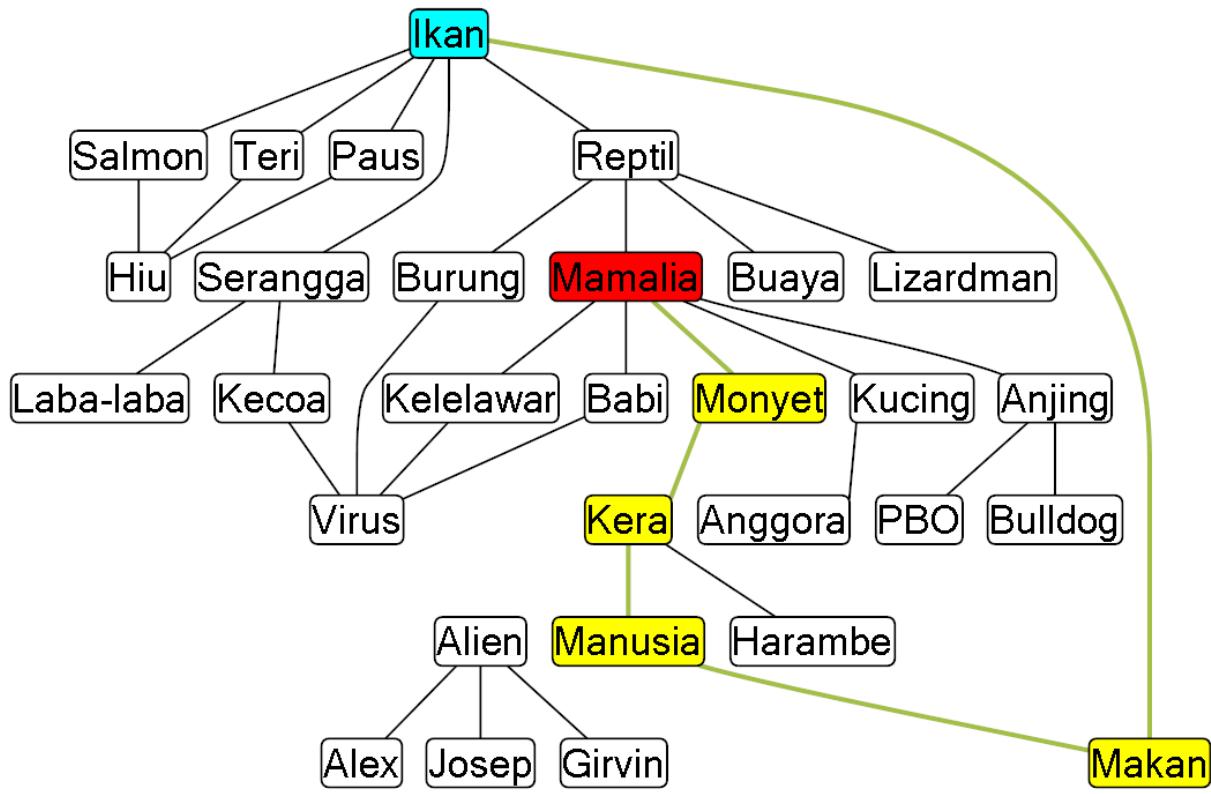
Output BFS:

Ikan dan Mamalia

Ikan → Reptil → Mamalia [1st degree]

Dari node Ikan, dimasukkan ke queue pengecekan node Makan, Paus, Reptil, Salmon dan Teri terurut dari abjad karena mereka merupakan node tetangga dari Ikan. Lalu di-dequeue node dari queue itu dan didapatkan node Makan, yang dari itu memasukkan node Manusia ke queue. Lalu di-dequeue lagi dan didapatkan node Paus yang memasukkan node Hiu ke queue. Setelah itu di-dequeue lagi dan didapatkan node Reptil dari queue yang memasukkan node Mamalia ke queue. Proses dequeue berlangsung sampai yang di-dequeue adalah node Mamalia yang tadi dimasukkan ke queue oleh node Reptil. Dari itu didapat jalurnya dari siapa yang memasukkan node ke queue sehingga didapat jalur Ikan-Reptil-Mamalia karena Mamalia dimasukkan oleh Reptil dan Reptil dimasukkan oleh Ikan.

Berikut adalah hasil untuk pencarian dari Node Ikan ke Mamalia melalui DFS.



Output DFS:

Ikan dan Mamalia

Ikan → Makan → Manusia → Kera → Monyet → Mamalia [4th degree]

Pencarian DFS dimulai dari node Ikan. Dari Ikan dipilih node tetangga yang memiliki abjad terkecil. Node tersebut adalah node Makan. Lalu dilakukan hal yang sama lagi di node Makan dan seterusnya sehingga memilih node Manusia, Kera, Monyet, dan pada akhirnya memilih node tujuan yaitu Mamalia. Dalam kasus ini, tidak dibutuhkan untuk melakukan backtrack karena node target ketemu tanpa mencapai jalan buntu.

## BAB IV

### IMPLEMENTASI DAN PENGUJIAN

#### 4.1 Implementasi Program (Pseudocode)

```
KAMUS
file:           File {struktur data/class yang menyimpan info File}
graf:           Graph {struktur data/class Graph}
terpilih, target, n: Node {struktur data/class sudut pada class graph}
friendsRecs, m:   Node[]
Pilihan:        string

{Fungsi untuk mencari secara BFS semua node yang tidak bertetanggaan dengan
node terpilih tapi memiliki tetangga yang bertetanggaan dengan node
terpilih}
func Graph::FriendRecommendations(Node terpilih) -> Array of Node

{Mencari node yang bertetanggaan dengan node terpilih dan node n}
func Graph::MutualFriends(Node n, Node terpilih) -> Array of Node

{Algoritma searching di graf dengan metode BFS dimulai dari terpilih sampai
ke entry}
func Graph::BFS(Node terpilih, Node target) -> Array of Node

{Algoritma searching di graf dengan metode DFS dimulai dari terpilih sampai
ke entry}
func Graph::DFS(Node terpilih, Node target) -> Array of Node

{Menghitung banyak elemen di array of node dikurangi 2, alias menghitung
banyak node yang harus dilalui agar sampai ke node akhir dari node awal}
func Graf::NDegreeConnection(Node[] jalur)-> string
```

```
ALGORITMA
input(file)
graf <- parse(file)

terpilih <- input(node) {input pilihan node dari graf}
target <- input(node) {input node target dari graf}

{Untuk explore friends}
input(pilihan) {BFS atau DFS}
if pilihan = "BFS" then
    jalur <- graf.BFS(terpilih, target)
else
    Jalur <- graf.DFS(terpilih, target)

output(jalur)
d <- Graph.NDegreeConnection(jalur) {Mencari n-degree-connection jalur}
```

```

output(d)

{Mencari semua node yang memiliki tetangga yang juga tetangganya node
terpilih}
friendRecs <- graf.FriendRecommendations(terpilih)

foreach n in friendRecs:
    output(n) {node friend yang recommended}

{Mencari mutualfriends dari dua node}
m <- graf.MutualFriends(n, terpilih)
output(m) {mutual antara node terpilih dan yang recommended}

```

## 4.2 Penjelasan Struktur Data

Struktur data yang digunakan untuk penyelesaian masalah *people you may know* pada media sosial menggunakan graf tidak berarah (*undirected-graph*). Sebuah sudut atau *vertex* pada graf merepresentasikan sebuah akun di media sosial kami. Sisi atau *edge* antara dua orang menandakan kedua orang itu saling berteman.

Struktur data graf direpresentasikan sebagai *adjacency list* (senarai ketetanggaan) dengan menggunakan koleksi data *dictionary* (semacam *hashmap*) dengan setiap sudut sebagai *key* (kunci) dan setiap tetangganya direpresentasikan dengan *linked list* (senarai terhubung) dan merupakan *value* (nilai) di *dictionary*.

Penggunaan representasi graf sebagai *adjacency list* adalah karena representasi ini memakan memori yang lebih kecil daripada representasi *adjacency matrix* dan waktu pemrosesan lebih cepat untuk graf dengan tetangga yang sedikit. Penggunaan *dictionary* agar pengambilan senarai tetangga dari suatu sudut lebih cepat dan penggunaan *linked list* untuk senarai tetangga suatu sudut dipilih agar penambahan tetangga bisa dilakukan lebih cepat karena penambahan tetangga dilakukan terurut naik berdasar *lexicon*.

## 4.3 Penjelasan tata cara penggunaan program

- Jalankan program dengan menjalankan HanyaKipas.exe.
- Klik choose file untuk memilih file input graph.
- Pastikan file input graph mengikuti format penulisan file seperti di readme.
- Pilih radio button DFS atau BFS untuk preferensi pencarian.
- Pilih akun dari node yang ada di graph
- Pilih akun target dari node yang ada di graph
- Klik tombol search.
- Output akan terlihat di bagian kanan interface.
- Pilihan-pilihan akun, file, dan metode pencarian dapat diubah dan dapat dilakukan pencarian kembali dengan mengklik tombol search lagi.

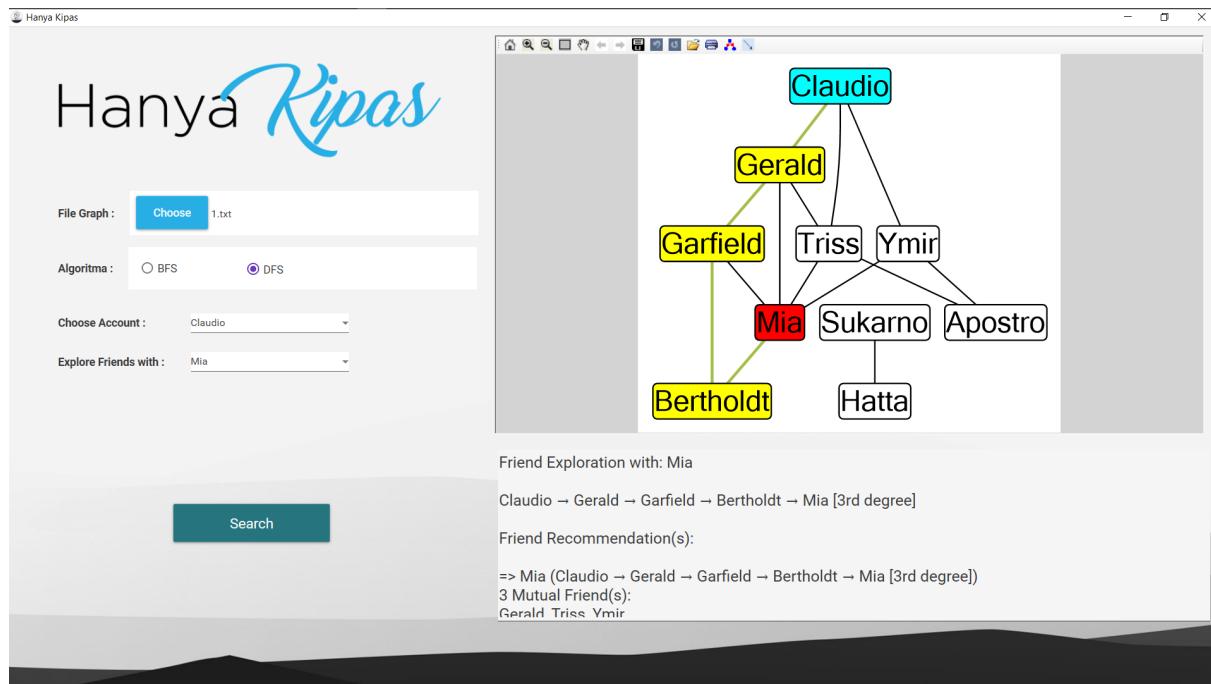
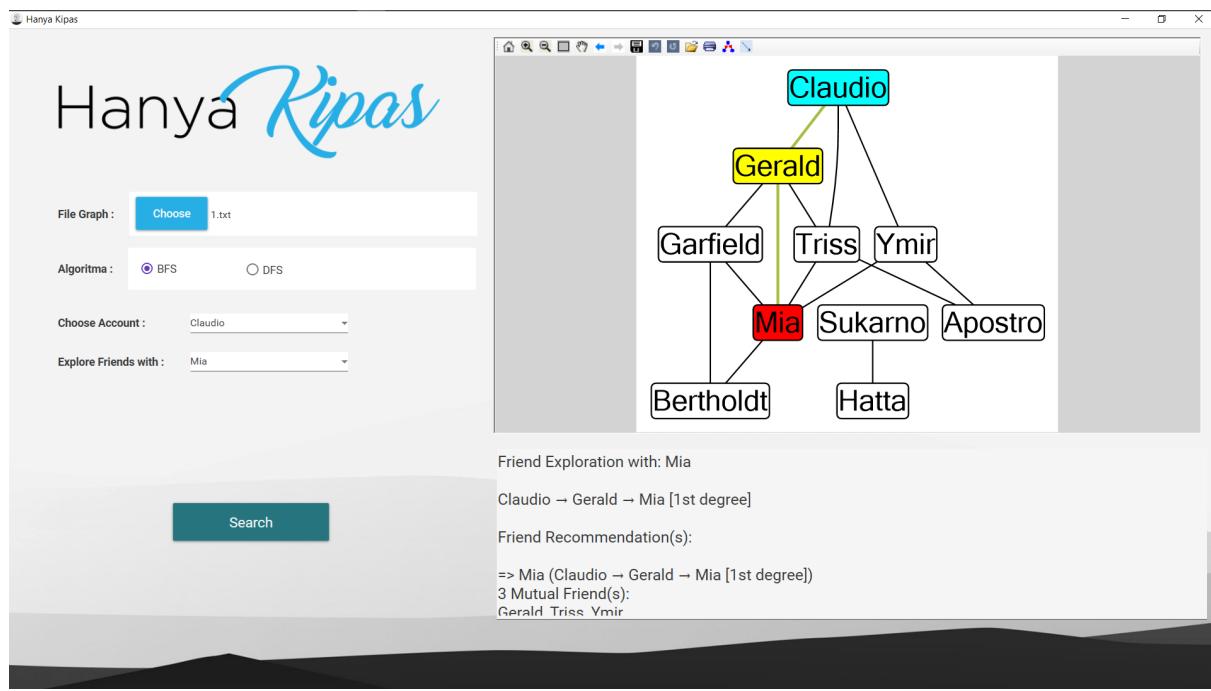
## 4.4 Hasil Pengujian

### 1. Kasus uji pertama (1.txt)

Claudio Gerald  
Claudio Triss  
Claudio Ymir  
Gerald Triss  
Gerald Garfield

Gerald Mia  
Triss Mia  
Triss Apostro  
Ymir Apostro  
Ymir Mia

Garfield Bertholdt  
Garfield Mia  
Mia Bertholdt  
Sukarno Hatta



## 2. Kasus uji kedua (2.txt)

Josep  
 Felicia  
 Alex  
 Suggoi  
 Josep

Girvin  
 Alex  
 Daniel  
 Josep  
 James  
 Irene  
 Girvin

Cynthia  
 James  
 Alex  
 Cynthia  
 Kevin  
 Josep

Hanya Kipas

File Graph : Choose 2.txt

Algoritma :  BFS  DFS

Choose Account : Josep

Explore Friends with : Irene

**Search**

Friend Exploration with: Irene

Josep → Felicia → Alex → Cynthia → James → Irene [4th degree]

Friend Recommendation(s):

=> Alex (Josep → Felicia → Alex [1st degree])  
2 Mutual Friend(s):  
Felicia Suggoi

Hanya Kipas

File Graph : Choose 2.txt

Algoritma :  BFS  DFS

Choose Account : Josep

Explore Friends with : Irene

**Search**

Friend Exploration with: Irene

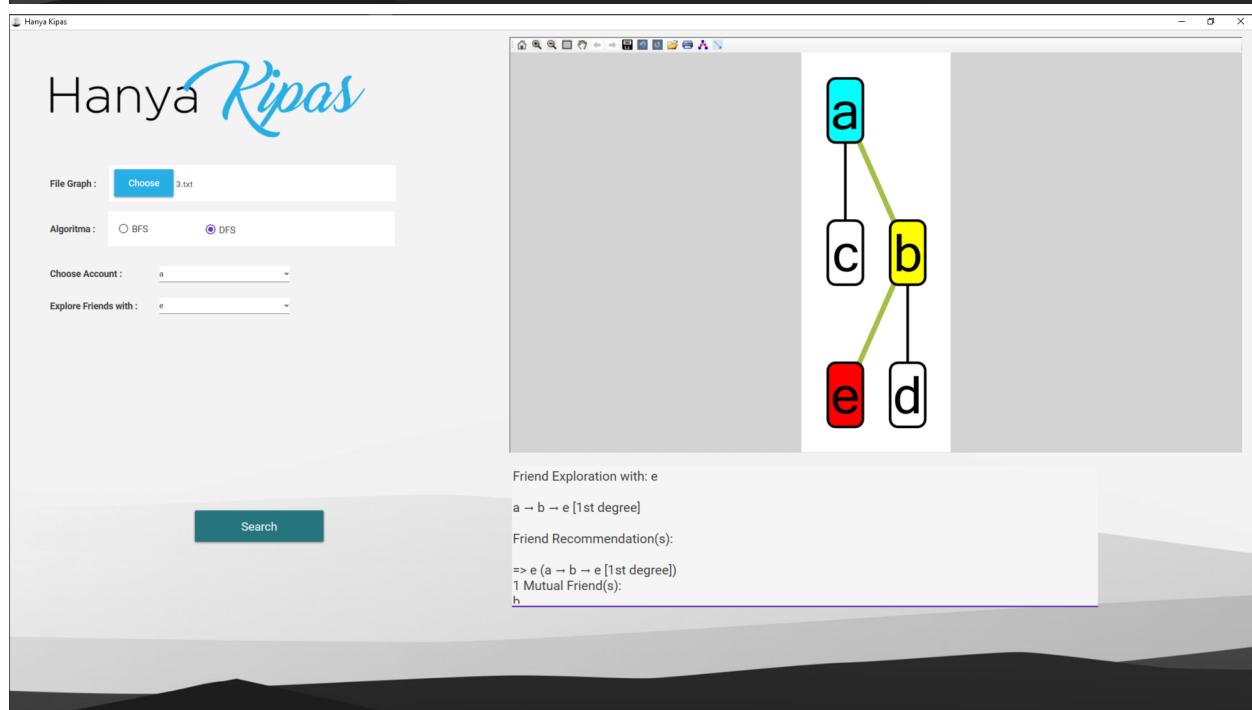
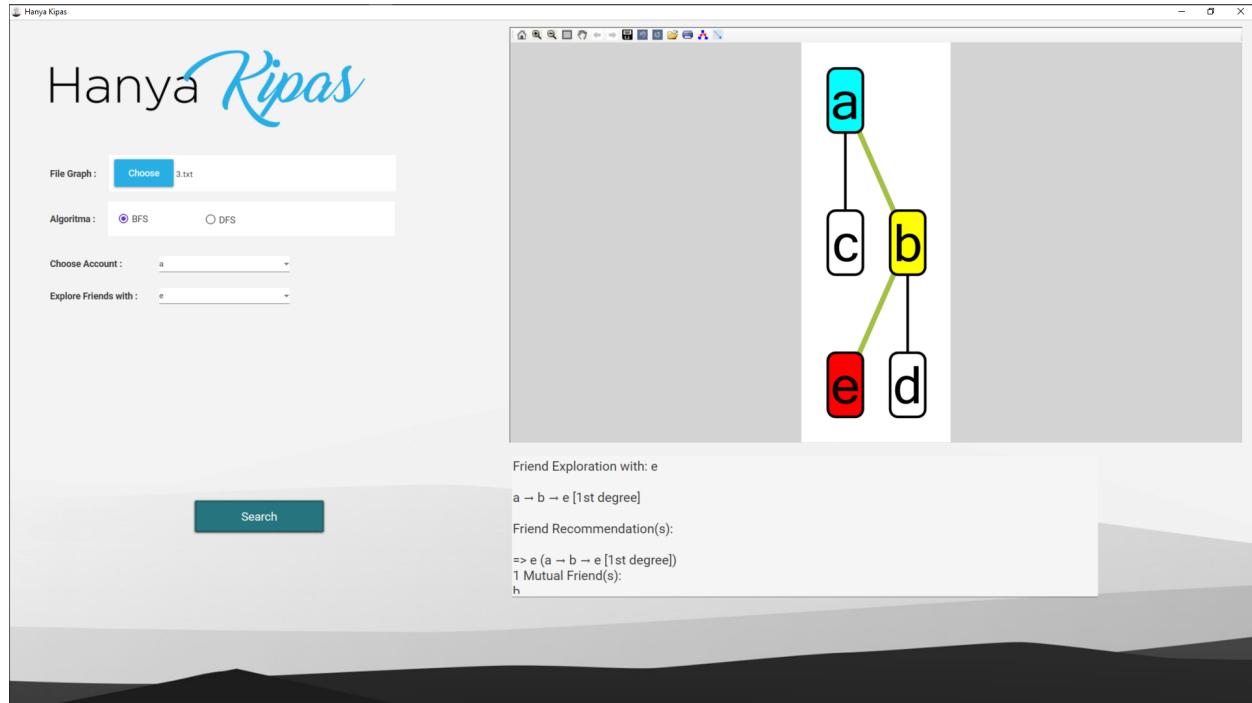
Josep → Felicia → Alex → Cynthia → James → Irene [4th degree]

Friend Recommendation(s):

=> Alex (Josep → Felicia → Alex [1st degree])  
2 Mutual Friend(s):  
Felicia Suggoi

### 3. Kasus uji ketiga (3.txt)

a b  
a c  
b d  
b e

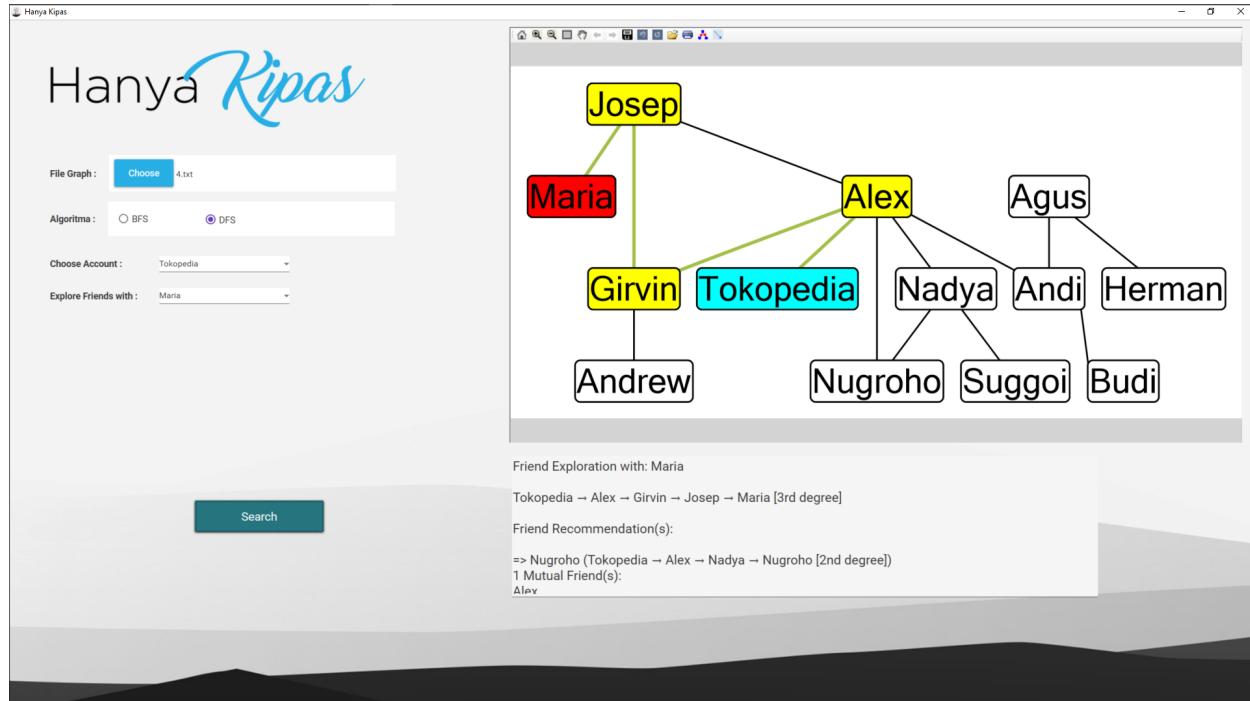
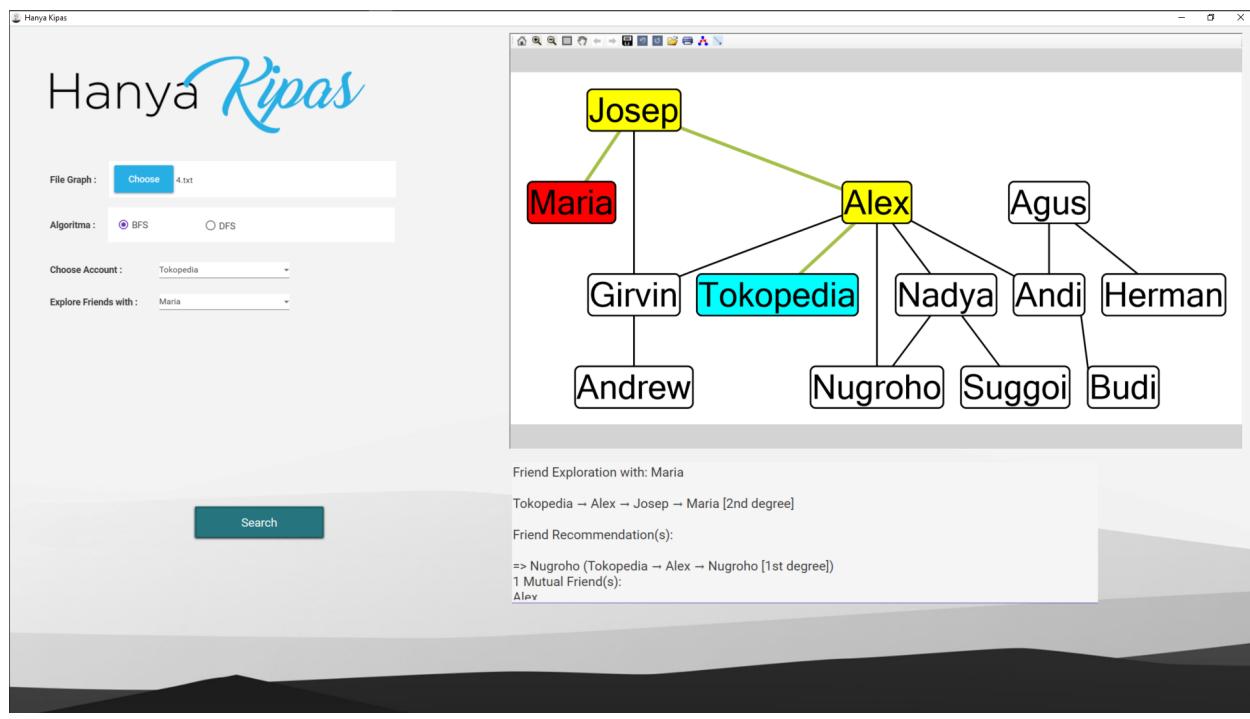


#### 4. Kasus uji keempat (4.txt)

Josep Alex  
 Josep Girvin  
 Girvin Alex  
 Alex Nadya  
 Nadya Nugroho

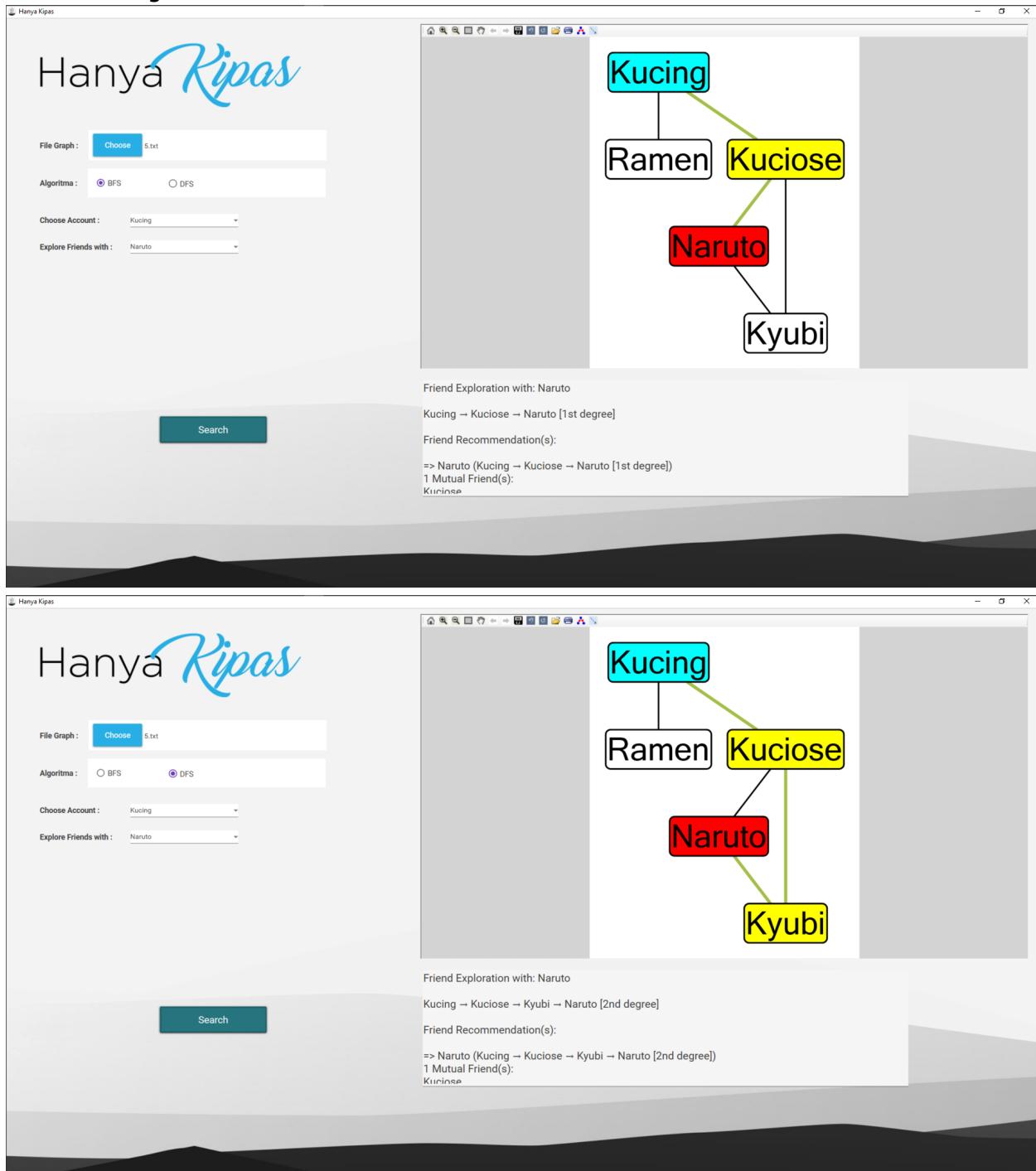
Suggoi Nadya  
 Nugroho Alex  
 Alex Tokopedia  
 Andrew Girvin  
 Agus Andi

Andi Alex  
 Budi Andi  
 Agus Herman  
 Josep Maria



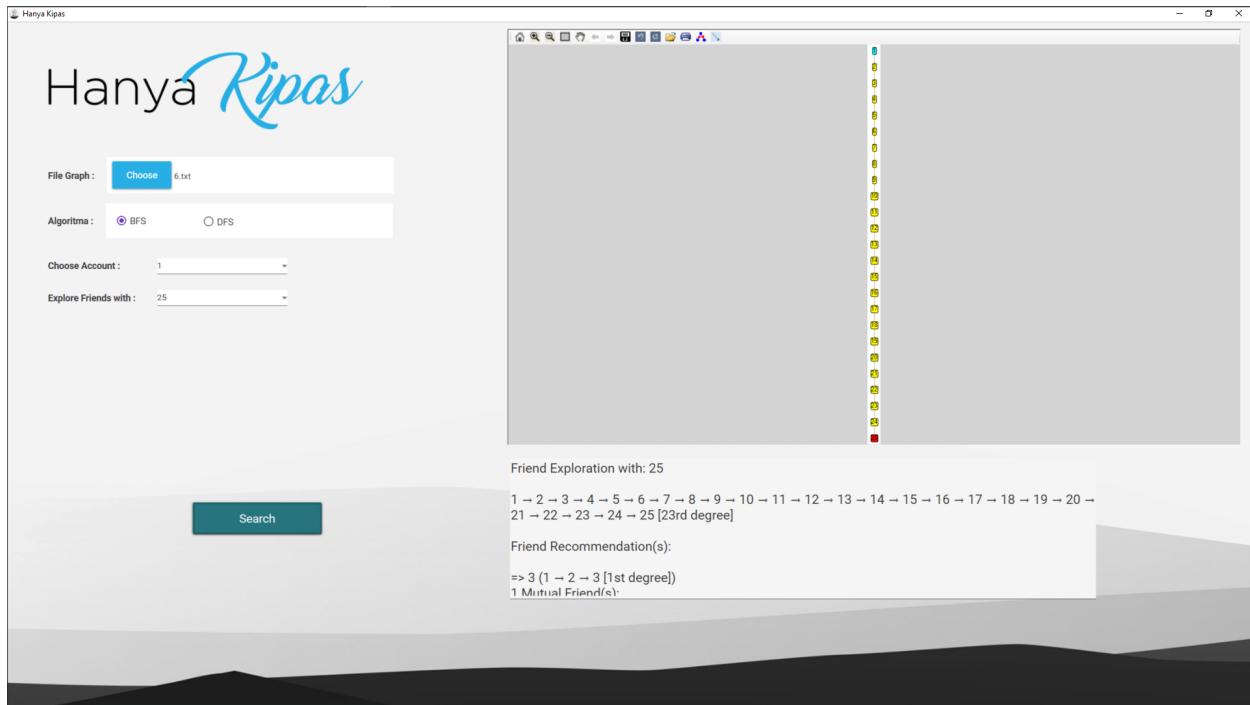
## 5. Kasus uji kelima (5.txt)

Kucing Kuciose  
 Kuciose Naruto  
 Naruto Kyubi  
 Kyubi Kuciose  
 Ramen Kucing



## 6. Kasus uji keenam (6.txt)

1 2	9 10	17 18
2 3	10 11	18 19
3 4	11 12	19 20
4 5	12 13	20 21
5 6	13 14	21 22
6 7	14 15	22 23
7 8	15 16	23 24
8 9	16 17	24 25

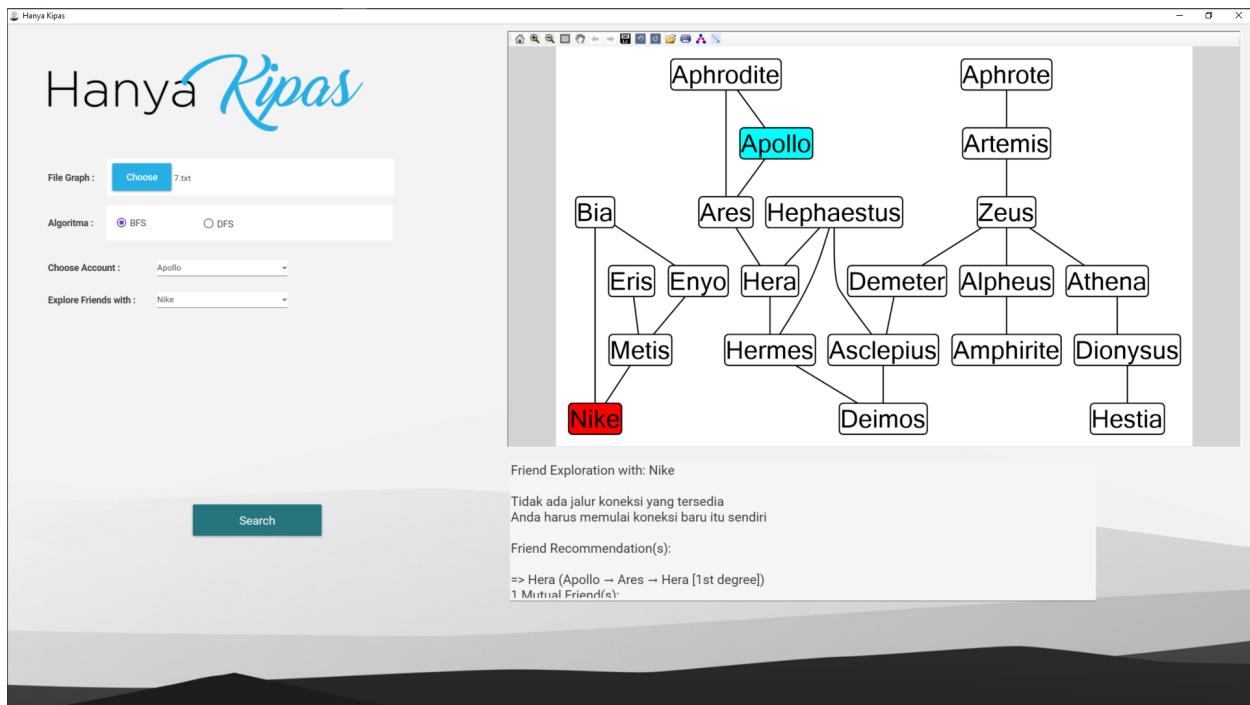


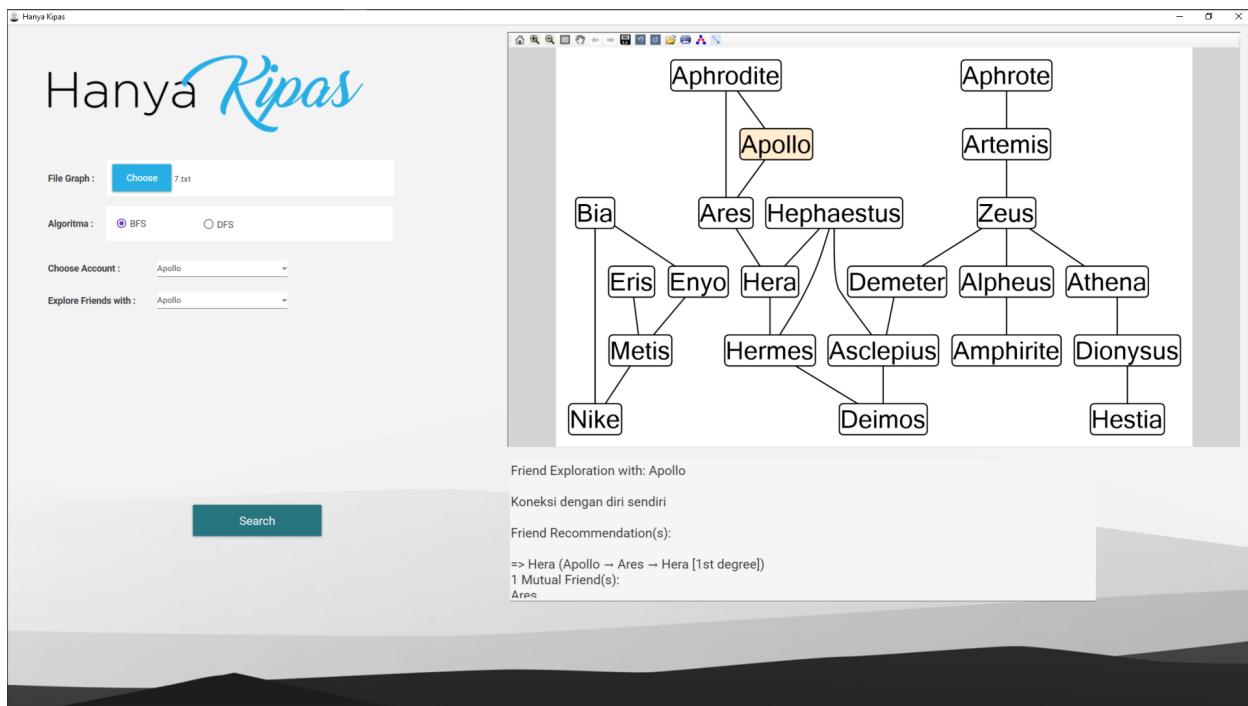
## 7. Kasus uji ketujuh (7.txt)

Aphrodite Apollo  
Apollo Ares  
Ares Aphrodite  
Aphrodite Artemis  
Artemis Zeus  
Zeus Demeter  
Zeus Athena  
Athena Dionysus

Hephaestus Hera  
Hera Ares  
Hephaestus Hermes  
Hera Hermes  
Hestia Dionysus  
Zeus Alpheus  
Alpheus Amphirite  
Demeter Asclepius

Asclepius Hephaestus  
Bia Enyo  
Asclepius Deimos  
Deimos Hermes  
Eris Metis  
Metis Nike  
Nike Bia  
Enyo Metis



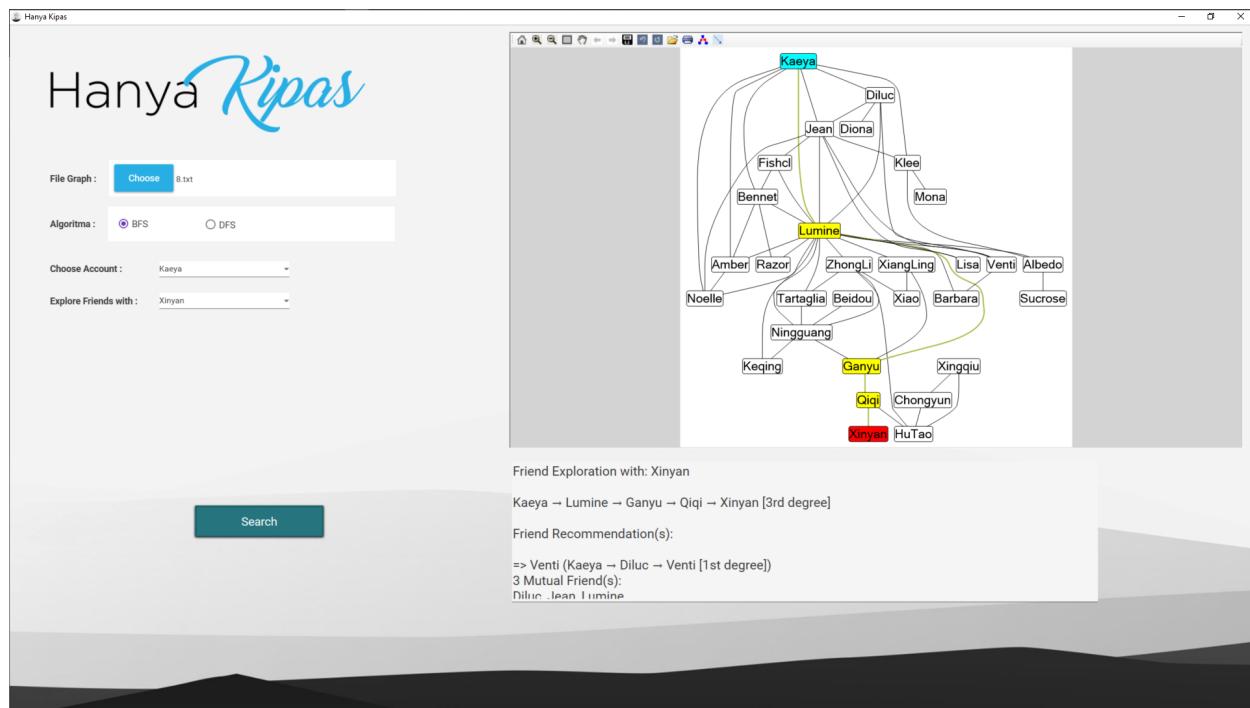


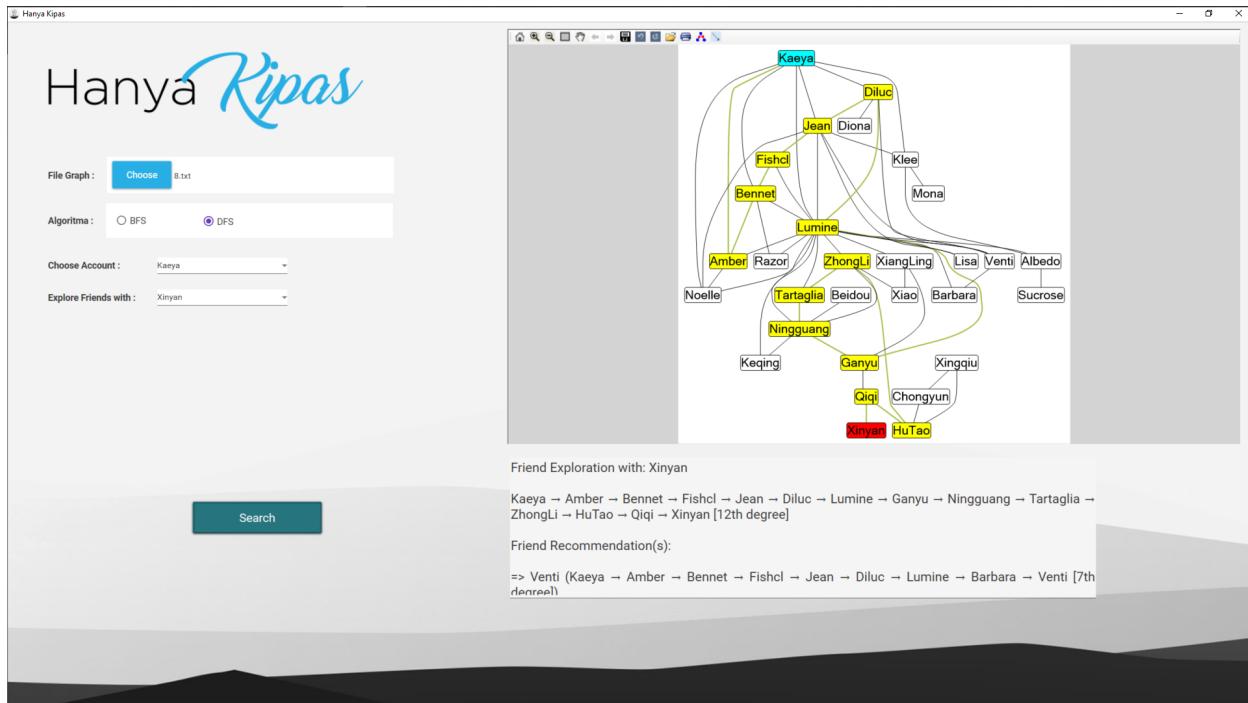
## 8. Kasus uji kedelapan (8.txt)

Kaeya Diluc  
 Diluc Jean  
 Jean Kaeya  
 Jean Klee  
 Klee Kaeya  
 Fishcl Bennet  
 Fishcl Jean  
 Lumine Jean  
 Bennet Lumine  
 Bennet Kaeya  
 Mona Klee  
 ZhongLi Lumine  
 Tartaglia ZhongLi  
 Beidou Ningguang  
 Ningguang ZhongLi  
 Ningguang Tartaglia  
 Diona Diluc  
 Venti Jean  
 Venti Diluc

Barbara Venti  
 XiangLing Ganyu  
 Ganyu Ningguang  
 Ningguang Keqing  
 Amber Lumine  
 Lisa Jean  
 Lisa Lumine  
 Lumine Ningguang  
 Lumine Tartaglia  
 Lumine Venti  
 Lumine Barbara  
 Lumine XiangLing  
 Lumine Ganyu  
 Lumine Keqing  
 Lumine Diluc  
 Lumine Kaeya  
 Amber Bennet  
 Lumine Fishcl  
 Lumine Bennet

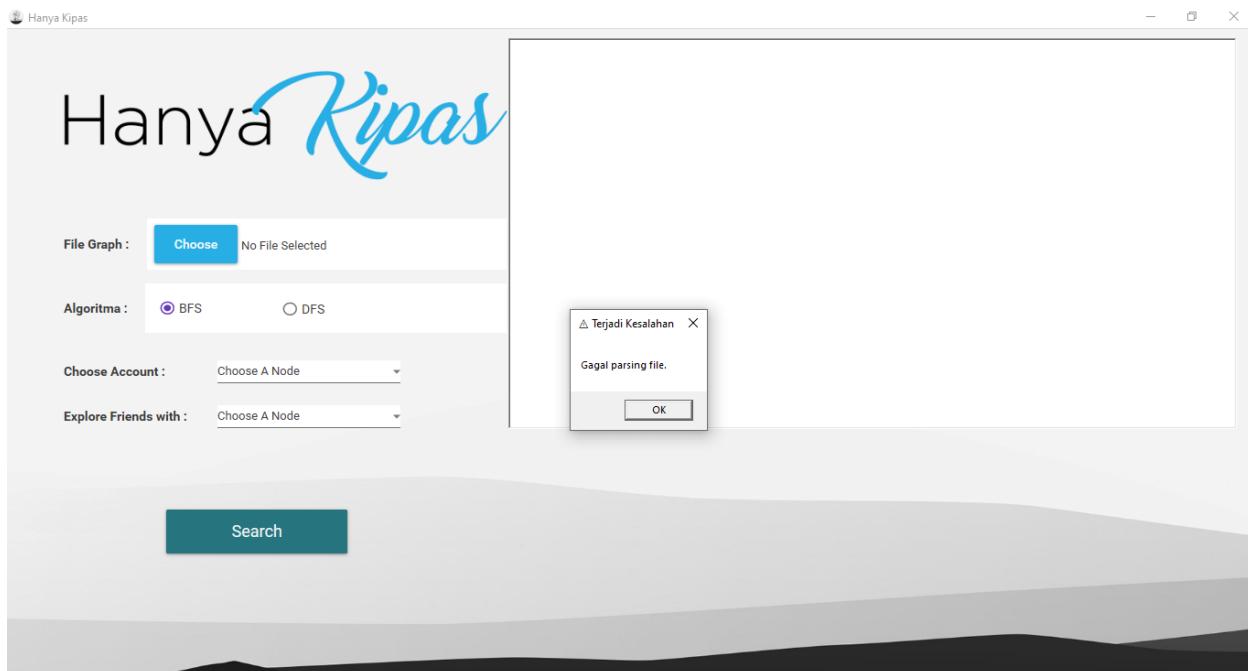
Razor Bennet  
 Razor Lumine  
 Noelle Jean  
 Noelle Kaeya  
 Noelle Lumine  
 Amber Noelle  
 Amber Kaeya  
 Albedo Lumine  
 Albedo Sucrose  
 Albedo Klee  
 Xiao ZhongLi  
 Xiao XiangLing  
 Xingqiu Chongyun  
 Qiqi HuTao  
 ZhongLi HuTao  
 Xingqiu HuTao  
 Chongyun HuTao  
 Xinyan Qiqi  
 Ganyu Qiqi





## 9. Kasus uji kesembilan (9.txt)

a  
b



Gagal parsing file karena format penulisan file 9.txt salah

#### 4.5 Analisis Desain solusi BFS dan DFS

Untuk masalah eksplorasi pertemanan, lebih baik menggunakan BFS. BFS dipilih karena akan menghasilkan jalur antara sudut asal ke sudut tujuan dengan jarak terpendek meskipun menggunakan memori yang lebih besar. Karena jarak yang dihasilkan pasti terpendek, maka derajat hubungan (*degree connection*) yang dihasilkan dijamin minimal.

Penggunaan DFS tidak optimal karena sifat DFS yang akan terus mendatangi tetangga dari tetangga sampai ke target. Sedangkan BFS bisa optimal karena BFS akan mendatangi tetangga-tetangganya dulu baru mendatangi tetangga dari tetangga.

BFS juga dipakai untuk fitur *friend recommendation* dibanding dengan DFS karena lebih cocok untuk pencarian *mutual friends*. Karena pencarian *mutual friends* itu hanya mencari tetangga dari tetangga node terpilih, maka lebih cocok menggunakan BFS dibanding DFS karena pencarinya lebih cocok dengan pencarian melebar dibanding pencarian dalam. Hubungan dengan *mutual friends* pasti *first-degree-connection* sehingga tidak berguna untuk pencarian dalam dan jauh lebih efektif dengan pencarian BFS.

Secara keseluruhan desain solusi BFS dan DFS untuk fitur *explore friends* berhasil untuk mencapai tujuan dari fitur. Desain solusi BFS untuk fitur *friend recommendations* juga berhasil untuk mencapai tujuan dari fitur.

## BAB V

### KESIMPULAN DAN SARAN

#### 5.1 Kesimpulan

Algoritma DFS dan BFS berhasil diimplementasikan dan digunakan dalam fitur *Explore Friends* dan pencarian *nth-degree-connection*. Algoritma untuk *Friend Recommendation* juga berhasil untuk diimplementasikan dengan baik. Semua kode ini mampu dihubungkan dengan baik dengan GUI sehingga berhasil tercipta suatu aplikasi yang memenuhi segala spesifikasi tugas yang diberikan.

#### 5.2 Saran

Saran yang dapat kami berikan setelah mengerjakan tugas besar ini adalah

- Mulai mengerjakan tugas besar dari jauh hari, sehingga tidak tertekan oleh deadline tubes lainnya
- IDE Visual Studio berat, sehingga harus dipersiapkan komputer/laptop yang memadai dan memiliki sistem operasi Windows. Koneksi internet, kuota internet, sisa *space* di media penyimpanan juga harus memadai untuk mengunduh Visual Studio.

#### 5.3 Refleksi

Pembagian tugas yang lebih dinamis (tidak ditetapkan seseorang harus mengerjakan apa) membuat keberlangsungan penggerjaan tugas lebih nyaman dan lebih mudah. Selain itu, bahasa C# ternyata tidak sulit untuk dipelajari dan memiliki *syntax* yang lebih mudah dibaca dibandingkan Java serta memiliki banyak library yang memudahkan penggerjaan. Program yang dibuat mungkin dapat dibuat lebih *object-oriented* lagi dan dapat dibagi menjadi lebih banyak modul lagi agar lebih modular.

#### 5.4 Komentar

Tugas ini menurut Penulis sangat seru dan sangat menantang. Penulis untuk mempelajari hal baru terkait pengembangan *front-end* pada perangkat lunak desktop. Penulis juga berterima kasih karena pembagian kelompok tidak diacak sehingga tidak mendapatkan anggota kelompok yang seakan-akan ditelan *black hole*. Penulis juga senang karena tubesnya tidak begitu berat sehingga tetap bisa semangat selama badai tubes. Tubes ini menjadi titik cerah di tubes-tubes semester 4 yang gelap-gelap :D. Sayangnya tubes ini membuat laptop Penulis menangis dan teriak minta tolong karena beratnya Microsoft Visual Studio.

## **DAFTAR PUSTAKA**

1. “BFS-DFS-2021-Bag1”. Rinaldi Munir  
[informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag1.pdf](http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag1.pdf)
2. “BFS-DFS-2021-Bag2”. Rinaldi Munir  
[informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag2.pdf](http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag2.pdf)