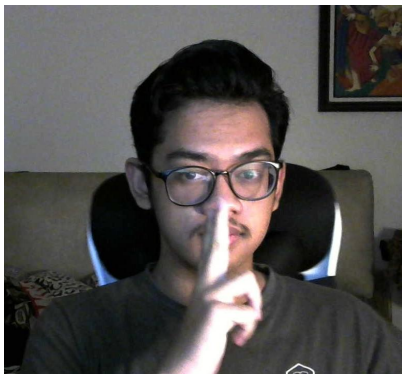
Tugas Besar IF2211 Strategi Algoritma

Pemanfaatan Breadth First Search dan Depth First Search dalam File dan Folder Searching

Semester I Tahun 2021/2022





Disusun oleh:

Jaya Mangalo Soegeng Rahardjo 13520015

Muhammad Naufal Satriandana 13520064

Vincent Christian Siregar 13520136

Institut Teknologi Bandung

Sekolah Teknik Elektro dan Informatika

DAFTAR ISI

Bab I

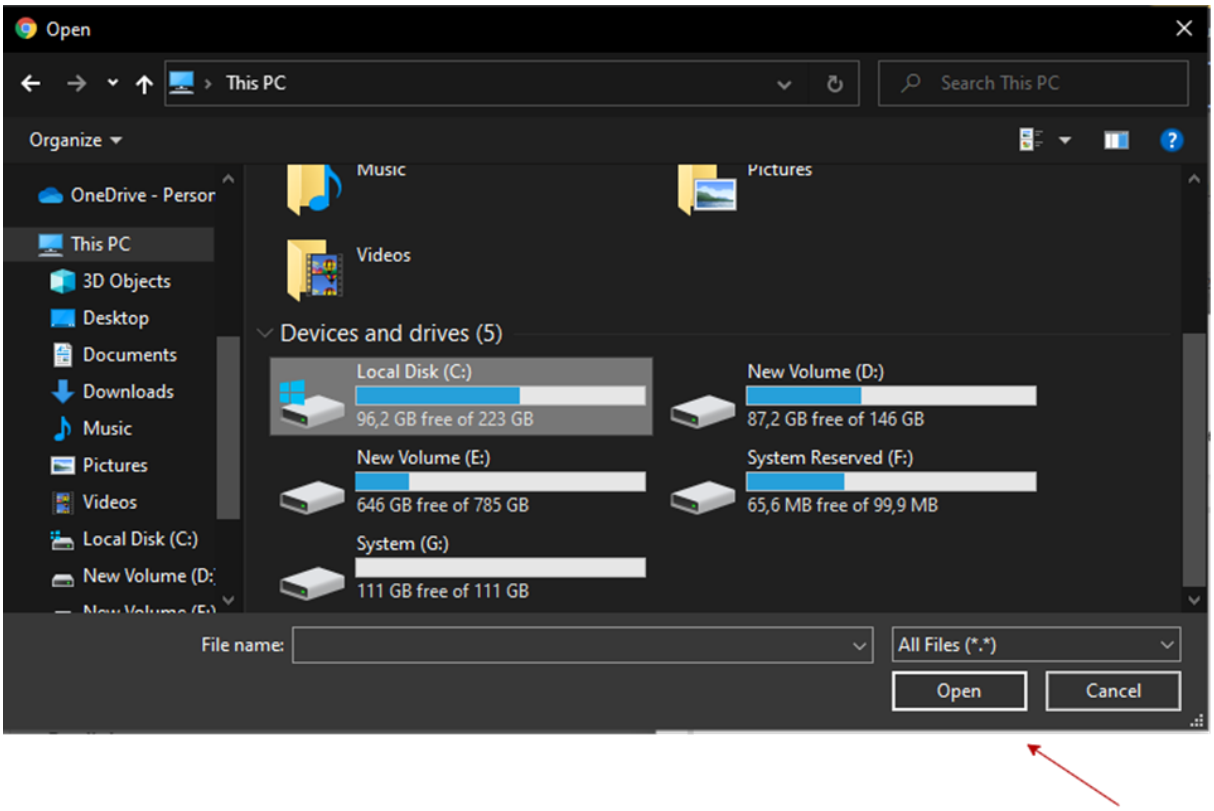
Deskripsi Tugas

Dalam tugas besar ini, Anda akan diminta untuk membangun sebuah aplikasi GUI sederhana yang dapat memodelkan fitur dari *file explorer* pada sistem operasi, yang pada tugas ini disebut dengan *Folder Crawling*. Dengan memanfaatkan algoritma *Breadth First Search* (BFS) dan *Depth First Search* (DFS), Anda dapat menelusuri folder-folder yang ada pada direktori untuk mendapatkan direktori yang Anda inginkan. Anda juga diminta untuk memvisualisasikan hasil dari pencarian *folder* tersebut dalam bentuk pohon.

Selain pohon, Anda diminta juga menampilkan list *path* dari daun-daun yang bersesuaian dengan hasil pencarian. *Path* tersebut diharuskan memiliki *hyperlink* menuju folder *parent* dari file yang dicari, agar file langsung dapat diakses melalui *browser* atau *file explorer*. Contoh hal-hal yang dimaksud akan dijelaskan di bawah ini.

Contoh Input dan Output Program

Contoh masukan aplikasi:



Input Starting Directory

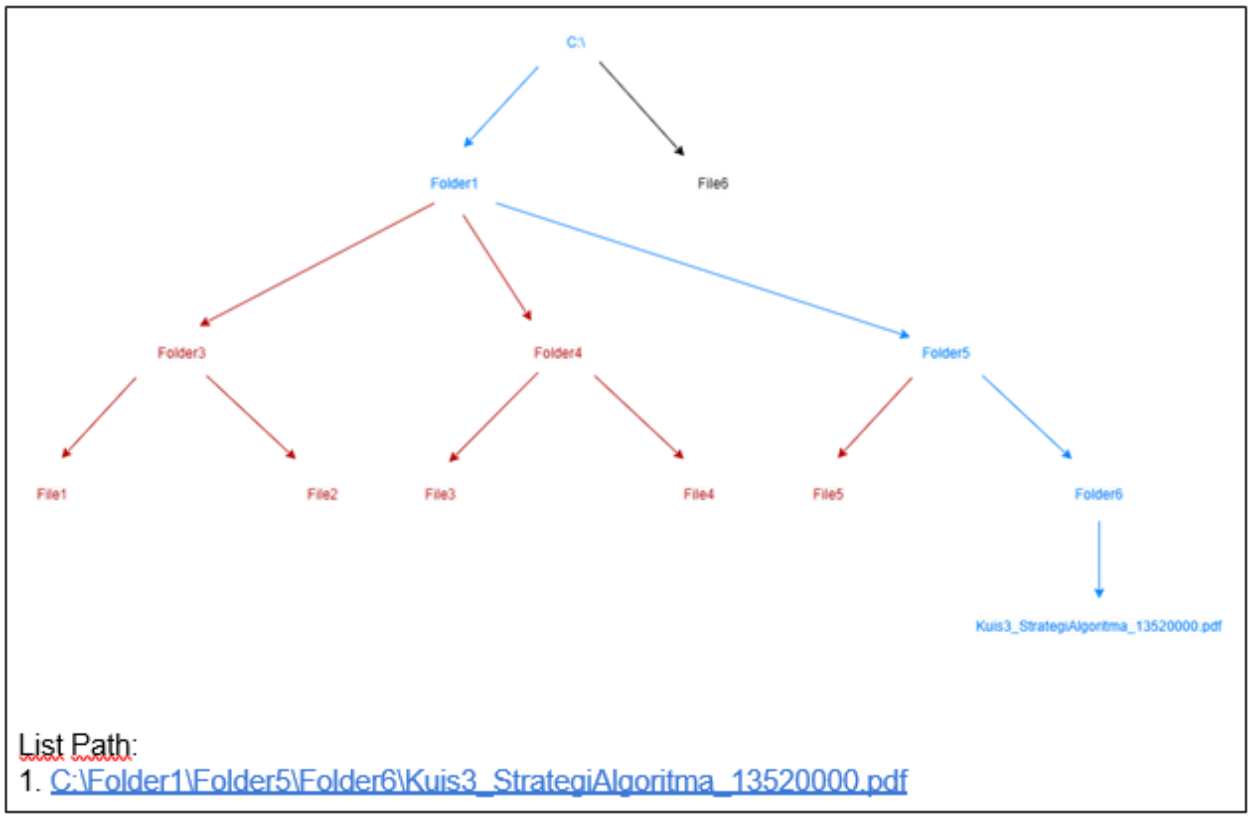
Kuis3_StrategiAlgoritma_13520000.pdf

Search

Input Nama File

Gambar 1. Contoh input program

Contoh output aplikasi:

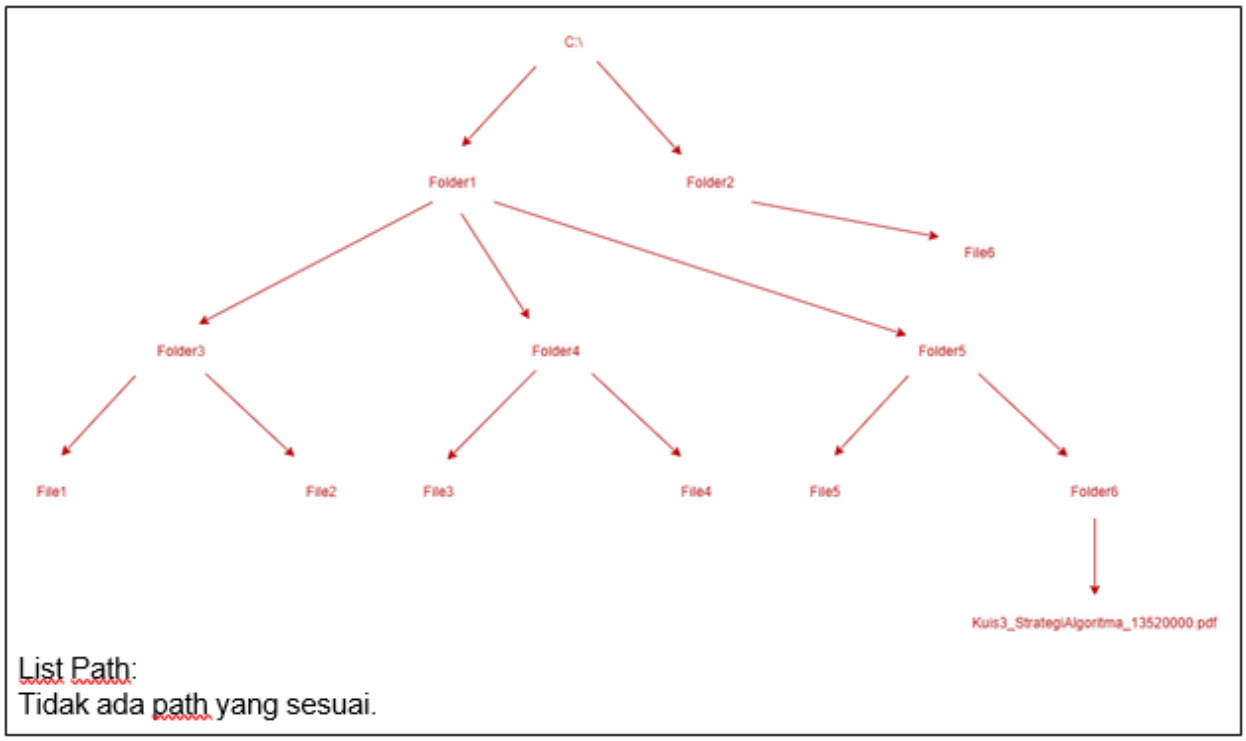


Gambar 2. Contoh output program

Misalnya pengguna ingin mengetahui langkah *folder crawling* untuk menemukan file Kuis3_StrategiAlgoritma_13520000.pdf.

Maka, path pencarian DFS adalah sebagai berikut. C:\ → Folder1 → Folder3 → File1 → Folder3 → File2 → Folder3 → Folder1 → Folder4 → File3 → Folder4 → File4 → Folder4 → Folder1 → Folder5 → File5 → Folder5 → Folder6 → Kuis3_StrategiAlgoritma_13520000.pdf.

Pada gambar di atas, rute yang dilewati pada pencarian DFS diwarnai dengan warna merah. Sedangkan, rute untuk menuju tempat file berada diberi warna biru. Rute yang masuk antrian tapi belum diperiksa diberi warna hitam. Anda bebas menentukan warnanya asalkan dibedakan antara ketiga hal tersebut.



Gambar 4. Contoh output program jika file tidak ditemukan

Jika file yang ingin dicari pengguna tidak ada pada direktori file, misalnya saat pengguna mencari Kuis3Probstat.pdf, maka path pencarian DFS adalah sebagai berikut: C:\ → Folder1 → Folder3 → File1 → Folder3 → File2 → Folder3 → Folder1 → Folder4 → File3 → Folder4 → File4 → Folder4 → Folder1 → Folder5 → File5 → Folder5 → Folder6 → Kuis3_StrategiAlgoritma_13520000.pdf → Folder6 → Folder5 → Folder1 → C:\ → Folder2 → File6.

Pada gambar di atas, semua simpul dan cabang berwarna merah yang menandakan seluruh direktori sudah selesai diperiksa semua namun tidak ada yang mengarah ke tempat file berada.

Bab II

Landasan Teori

2.1 Dasar Teori

Traversal Graf

Algoritma Traversal Graf adalah mengunjungi simpul dengan cara yang sistematik. Terdapat 2 jenis traversal graf yaitu pencarian melebar (breadth first search/BFS) dan pencarian mendalam (depth first search/DFS). Asumsi yang digunakan adalah graf merupakan graf terhubung. Traversal Graf digunakan untuk pencarian solusi pada graf.

Terdapat dua jenis algoritma pencarian solusi yaitu tanpa informasi (*uninformed/blind search*) dan dengan informasi (*informed Search*). Pencarian solusi tanpa informasi artinya tidak memiliki informasi tambahan. Pencarian solusi dengan informasi artinya pencarian menggunakan heuristik.

Algoritma BFS dan DFS yang digunakan pada folder crawling merupakan algoritma pencarian solusi tanpa informasi.

BFS

BFS atau *Breadth First Search* adalah pencarian graf secara melebar. Pada graf pohon dengan kedalaman n, pencarian dilakukan dari level 0 sampai level n secara berurutan.

Algoritma BFS secara umum:

Misalkan traversal dimulai dari simpul v

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.

DFS

DFS atau *Depth First Search* adalah pencarian graf secara mendalam.

Algoritma DFS secara umum:

Misalkan traversal dimulai dari simpul v

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

Bahasa c#

C# merupakan sebuah bahasa pemrograman yang berorientasi objek yang dikembangkan oleh Microsoft sebagai bagian dari inisiatif kerangka .NET Framework. Program c# berbasi c++ yang dipengaruhi oleh aspek-aspek ataupun fitur bahasa yang terdapat pada bahasa-bahasa pemrograman lain seperti java, Delphi, Visual Basic, dan lain-lain.

Windows Forms

Salah satu fitur yang dapat digunakan dalam membuat dekstop aplication berbasis bahasa c# adalah dengan menggunakan Windows Forms. Windows Forms adalah framework UI untuk membentuk windows desktop app. Windows Forms menyediakan kakas yang dapat digunakan pemrogram untuk memberikan visualisasi terhadap program. Windows Forms mendukung fitur drag-n-drop dalam menambahkan kontrol visual pada layar.

Windows Forms menggunakan platform .NET 6.0 dalam pengoprasiannya.

MSAGL

Microsoft Automatic Graph Layout (MSAGL) adalah library yang digunakan untuk memberikan visualisasi graph ke layar. MSAGL terintegrasi dengan Windows Forms sehingga visualisasi MSAGL dapat digunakan di dalam Windows Forms.

MSAGL terdiri memiliki 4 buah module yaitu:

1. The Core Layout engine (AutomaticGraphLayout.dll) digunakan untuk membuat layout dari graf.
2. The Drawing module (AutomaticGraphLayout.Drawing.dll) digunakan untuk membuat objek graf yang nantinya dapat digunakan untuk layout.
3. WPF control (Microsoft.Msagl.WpfGraphControl.dll) digunakan untuk memberikan control pada visualisasi graf khusus untuk kakas WPF.
4. Windows Forms Viewer control (Microsoft.Msagl.GraphViewerGdi.dll) digunakan untuk memberikan control pada visualisasi graf.

Bab III

Analisis Pemecahan Masalah

3.1 Langkah Breakdown

Masalah dalam Tugas Besar 2 ini adalah mengenai pencarian/searching dalam folder dan files menggunakan algoritma Breadth First Search dan Depth First Search. Untuk langkah secara detail, lebih dijelaskan pada bagian 4.1 Pseudocode, tetapi untuk garis besar cara kerja algoritm adalah sebagai berikut:

Initialization

Membaca input yang diberikan oleh user melalui *Graphical User Interface* (GUI). Input yang diberikan berupa:

- Folder yang ingin di-*search* isinya.
- Nama file yang ingin dicari dalam folder (Nama file harus sama persis dan menggunakan extensionnya, Ex: note.txt)
- Pilihan metode searching (BFS/DFS)
- Optional Pilihan untuk mencari semua *occurrence* atau hanya yang ditemukan pertama. (Find All Occurrences)

Searching

Setelah Initialization, Algoritma akan melakukan Searching berdasarkan opsi pilihan yang dipilih pada step Initialization.

BFS

Searching dengan algoritma Breadth First Search mengutilisasi Queue dengan beberapa step berikut:

1. Dequeue: Me-*dequeue* folder *top* yang terletak di queue dan melakukan Step 2.
2. Folder Searching: Mencari semua file dan folder yang terdapat dalam folder yang di-search.
3. Checking: Semua file yang ditemukan dalam folder diperiksa jika merupakan file yang dicari.
4. Inserting to Queue: Semua folder yang ditemukan dalam folder dimasukkan ke dalam Queue.
5. Display: Folder dan File yang di search di-display ke GUI secara Real-Time.
6. Repeat.

End Case:
Find All Occurrences - Program akan berhenti jika Queue sudah empty (semua folder sudah di search)
Find First Occurrences - Program akan berhenti kita file ditemukan pada Step 3.

DFS

Implementasi depth first search yang diimplementasikan hanya menggunakan struktur data tree. Langkah implementasinya adalah sebagai berikut:

1. Untuk setiap file yang ada pada directory saat ini, traversal hingga ketemu file yang diinginkan.
2. Jika belum ketemu, untuk setiap folder yang ada pada directory saat ini, lakukan DFS (kembali ke langkah 1). Hal ini akan otomatis menyebabkan pencarian dilakukan secara DFS karena sifat dari rekurensnya.
3. Jika sudah ketemu, terminasi fungsi dengan sebuah flag yang menyatakan bahwa file sudah ditemukan dan rekurens bisa diterminasi. Jika tipe pencarian adalah find all occurences, lanjutkan DFS hingga pada directory saat ini tidak ada folder lagi.
4. Jika tidak ditemukan file yang diinginkan walaupun sudah mencari keseluruhan directory, terminasi.

End Case:
Find All Occurrences - Program akan berhenti jika tidak ada folder lagi pada setiap instance rekurensnya (sudah semua direktori dijelajahi)
Find First Occurrences - Program akan berhenti kita file ditemukan atau sudah semua direktori dijelajahi.

3.2 Mapping

Dalam mapping persoalan searching file menjadi elemen-elemen yang bisa digunakan oleh algoritma BFS dan DFS ada sebagai berikut:

Root - Folder yang dipilih melalui GUI bekerja sebagai start atau root dari program BFS/DFS.
Search
Nodes - Noda yang harus dicek/traverse adalah semua file dan folder yang terdapat di folder parent-nya.
End - Untuk Search all, program akan berakhir jika semua folder dan file sudah di cek. Sementara untuk not search all, program akan berakhir untuk *instance* pertama yang ditemukan atau ketika semua folder dan file sudah di cek apabila ternyata tidak terdapat file yang dicari.

3.3 Corner Cases

Jika terdapat sebuah test case yang sangat besar, atau ketika sebuah program sedang searching, tidak ada safety gate untuk memberhentikan user untuk melakukan searching lagi.

Jika ini terjadi, maka akan terdapat dua searching terjadi bersamaan dan display akan direbut oleh kedua program tersebut.

Bab IV

Implementasi dan Pengujian

4.1 Implementasi program

BFS

```
Enqueue root into queue
While (queue not empty) {
	Current = dequeue from queue

	If ( current name == target) {
		Color Current as found // blue
		Color Current parent as found // blue
		Append Current to pathout
		return //if search all occurrences disabled. if enabled, remove this line.
	}
	Else{
		Color Current as not found //red
	}

	Foreach file from  files {
		Enqueue file into Queue
	}
	Foreach folder from  folders {
		Enqueue folder into Queue
	}
}
```

DFS

```
Create array of available file and folders in the current directory
If(current node is a folder) then
	If(file exists in current directory) then
		Foreach (file in current directory)
		//if search all occurrences is disabled, check first if already found
			Do dfs to them
		Endforeach
	Endif
	If(folder exists in current directory) then
```

```
            Foreach (folder in current directory)
            //if search all occurrences is disabled, check first if already found
                Do dfs to them
            Endforeach
        Endif
Else
        If(this file has the same name as wanted) then
            Mark this node as found by coloring its path from its parent
```

4.2 Penjelasan Struktur Data

Program ditulis dalam bahasa C#, dikembangkan dengan IDE Visual Studio, Windows Forms, MSAGL, dan WindowsAPICodePack,. Struktur data utama yang digunakan untuk melakukan Searching adalah struktur data Graph dan Queue.

Kelas Graph dengan nama Tree
Tree memiliki atribut:

- Counter (static int)
- Id (int)
- FileName (String)
- Path (String)
- warna (Enum {Merah, HItam, Biru})
- parent (Tree)
- Children (List<Tree>)

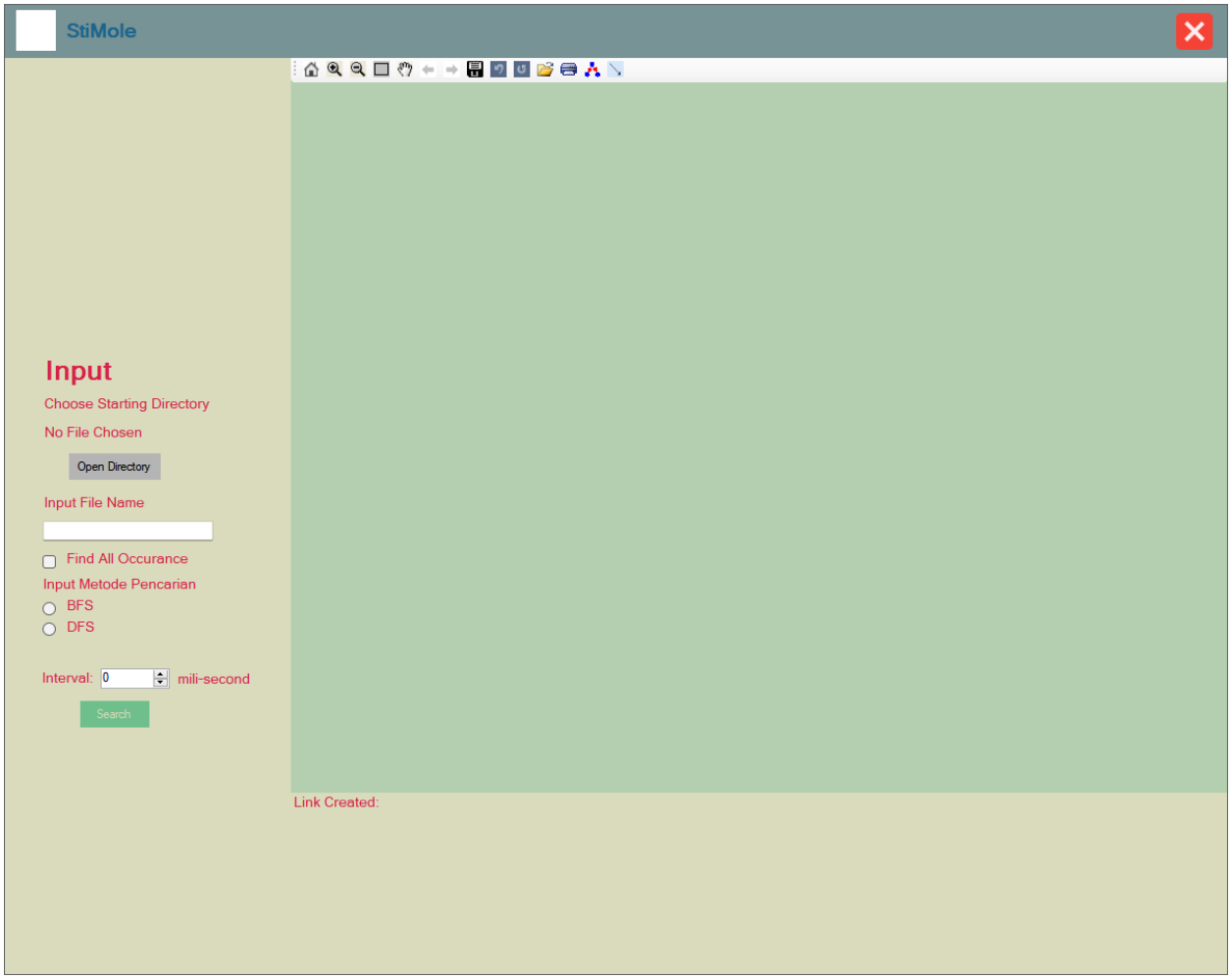
Tree digunakan untuk membantu melakukan pencarian baik untuk metode BFS dan DFS.

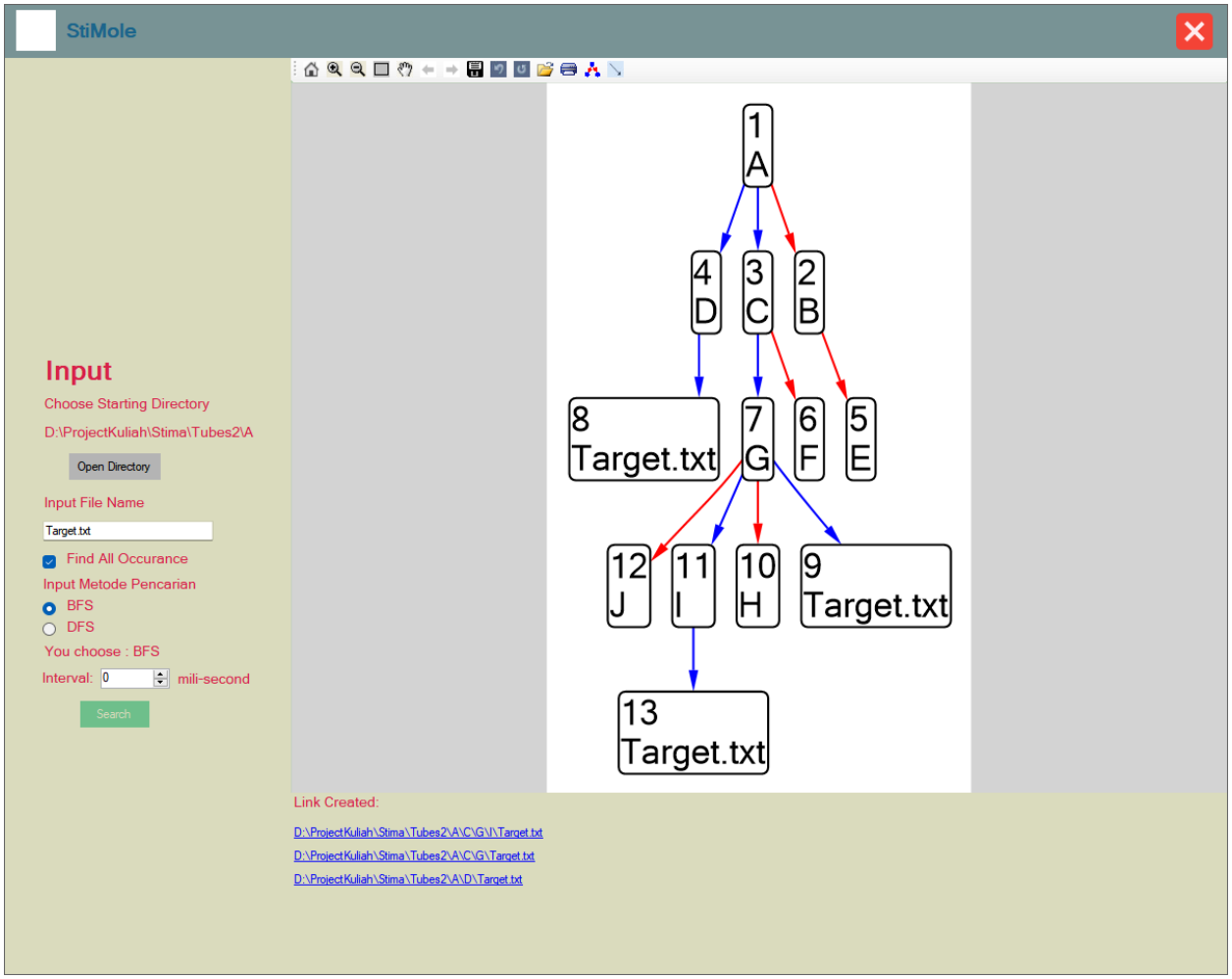
Kelas Queue yang digunakan dalam algoritma BFS menyimpan node Tree.

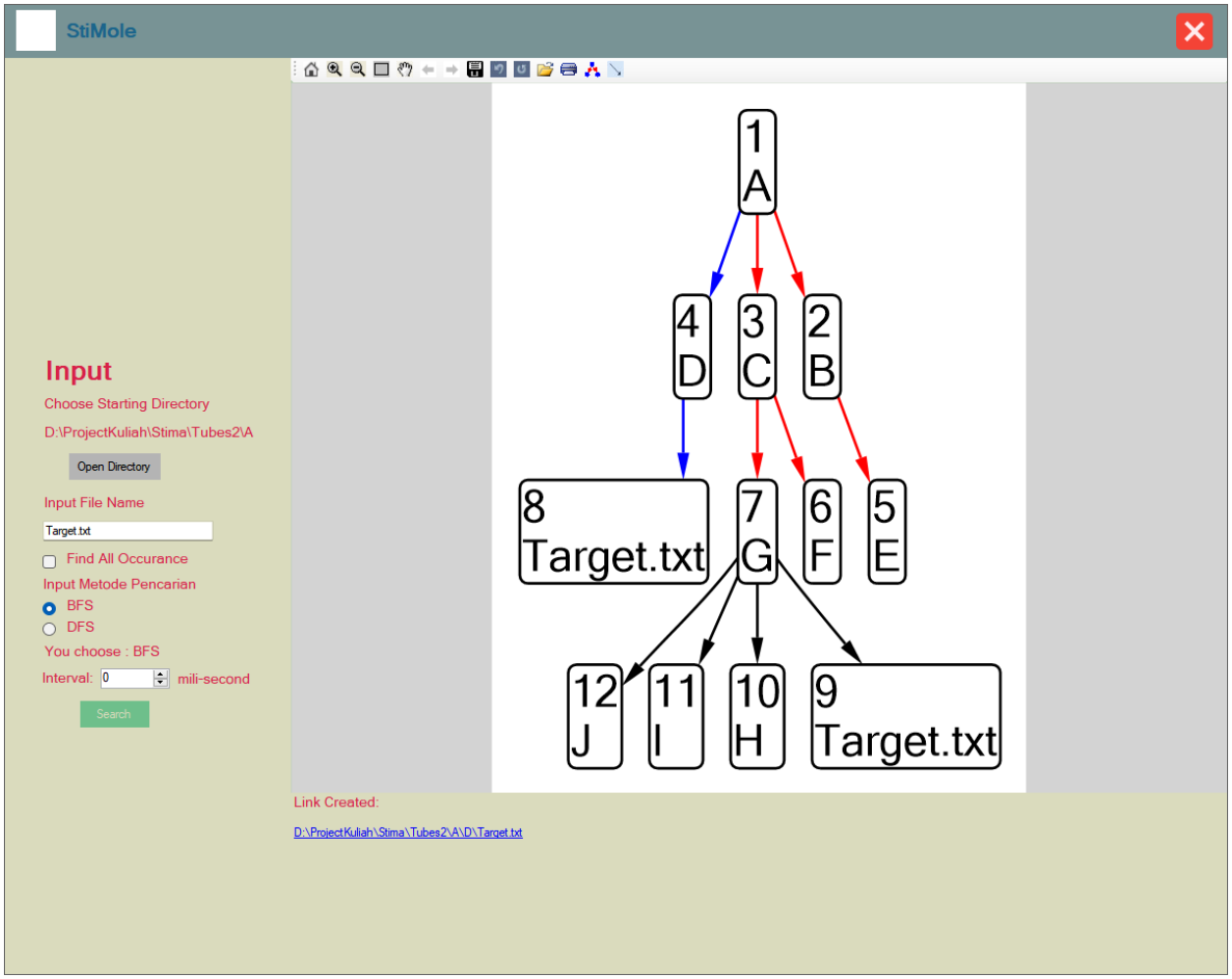
4.3 Penjelasan Tata Cara Penggunaan Program

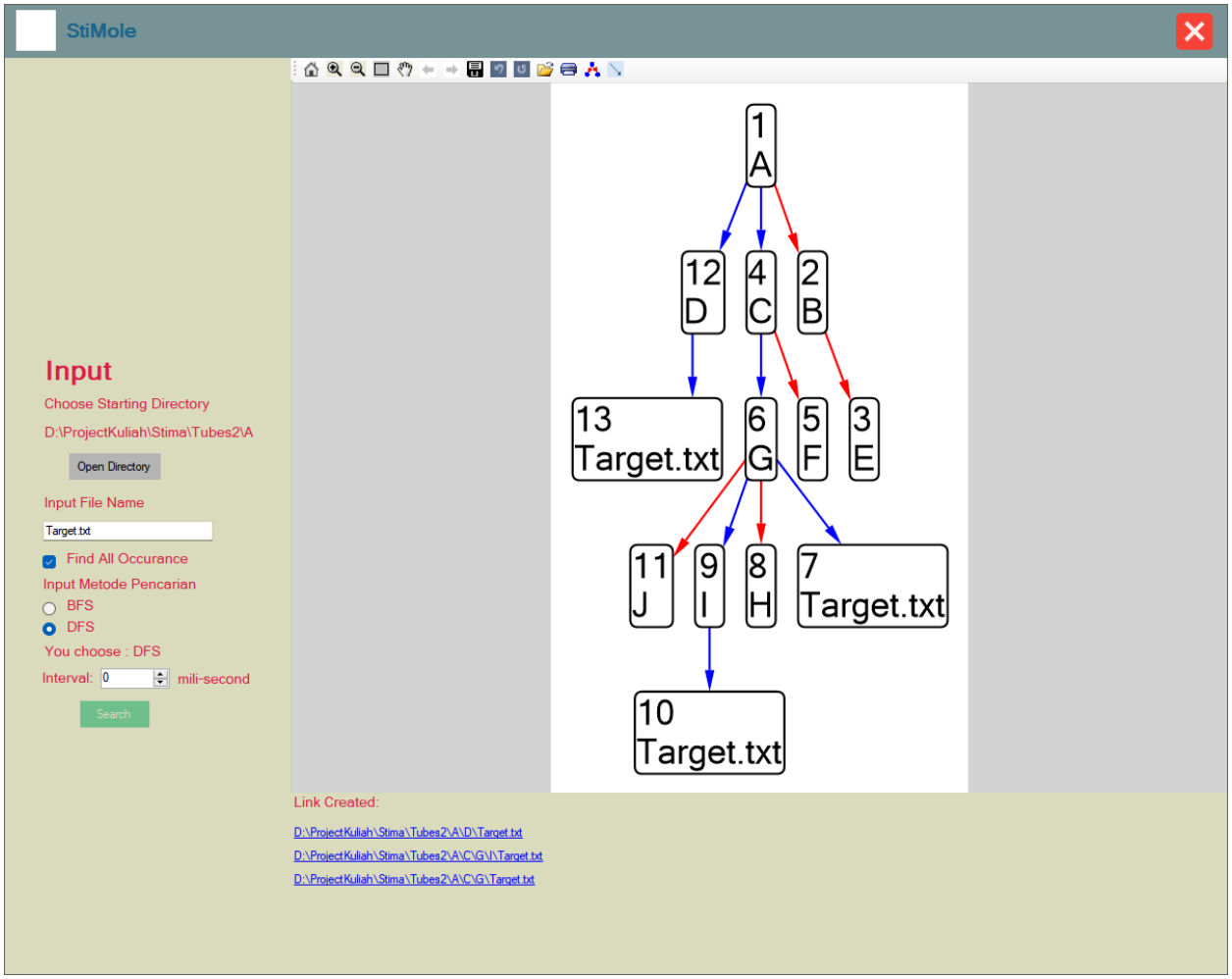
1. Start Program
2. Enter Folder yang ingin dicari file-file dan foldernya.
3. Enter filename
4. Pilih Metode Searching: BFS/DFS
5. Extra Option: Checkbox “Find All Occurrences” untuk mencari semua *instance* file.
6. Extra Option: Display Interval untuk memberikan interval real-time ke display.
7. Press Submit.
8. Gunakan Hyperlink untuk membuka file.

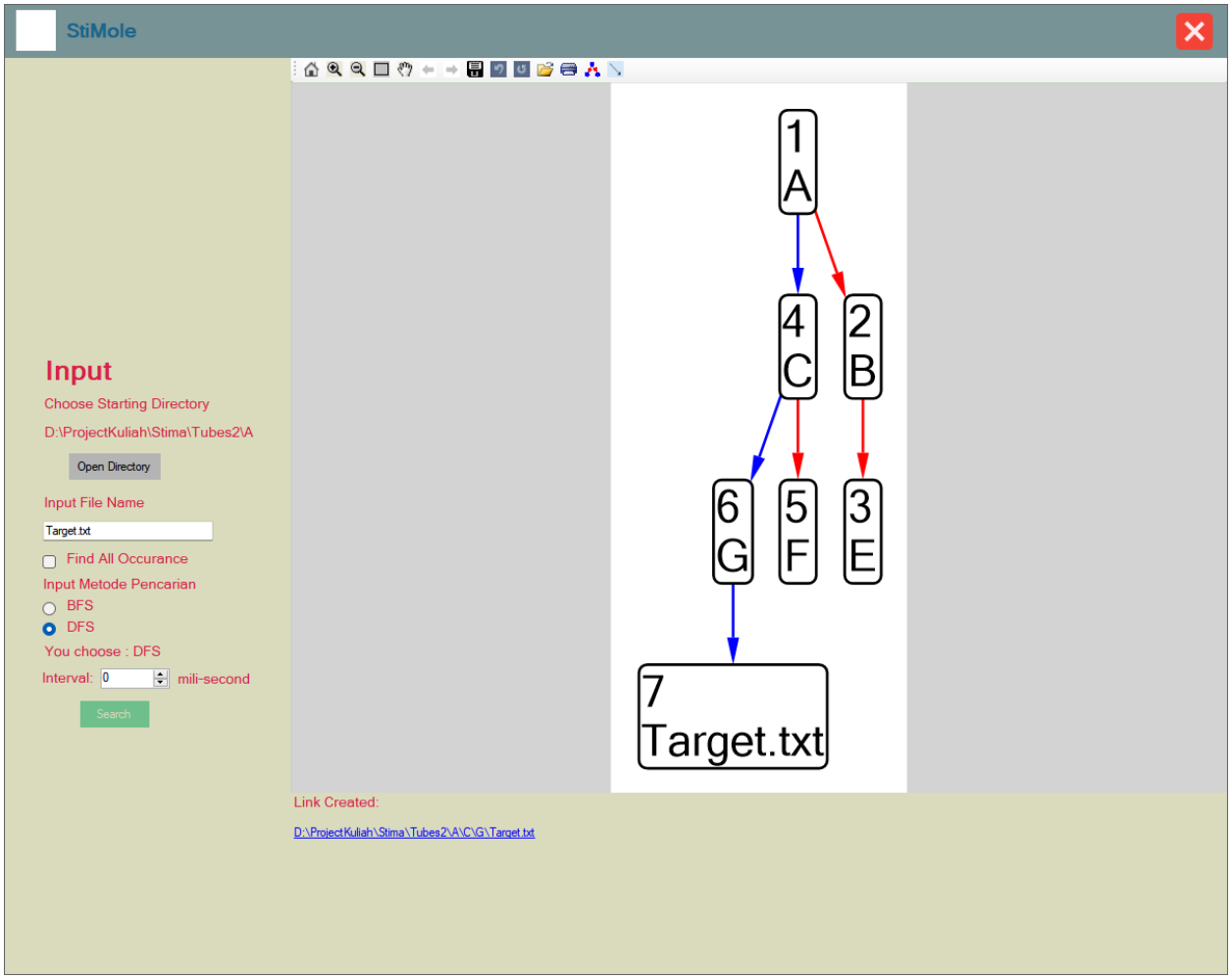
4.4 Hasil Pengujian


StiMole
Input
Choose Starting Directory
No File Chosen
Open Directory
Input File Name
Find All Occurance
Input Metode Pencarian
BFS
DFS
Interval:
0
mili-second
Search
Link Created:


StiMole
Input
Choose Starting Directory
D:\ProjectKuliah\Stima\Tubes2\A
Open Directory
Input File Name
Target.txt
Find All Occurance
Input Metode Pencarian
BFS
DFS
You choose : BFS
Interval:
0
mili-second
Search
1
A
4
D
3
C
2
B
8
Target.txt
7
G
6
F
5
E
12
J
11
I
10
H
9
Target.txt
13
Target.txt
Link Created:
D:\ProjectKuliah\Stima\Tubes2\A\C\G\I\Target.txt
D:\ProjectKuliah\Stima\Tubes2\A\C\G\Target.txt
D:\ProjectKuliah\Stima\Tubes2\A\D\Target.txt


StiMole
Input
Choose Starting Directory
D:\ProjectKuliah\Stima\Tubes2\A
Open Directory
Input File Name
Target.txt
Find All Occurance
Input Metode Pencarian
BFS
DFS
You choose : BFS
Interval:
0
mili-second
Search
1
A
4
D
3
C
2
B
8
Target.txt
7
G
6
F
5
E
12
J
11
I
10
H
9
Target.txt
Link Created:
D:\ProjectKuliah\Stima\Tubes2\A\D\Target.txt


StiMole
Input
Choose Starting Directory
D:\ProjectKuliah\Stima\Tubes2\A
Open Directory
Input File Name
Target.txt
Find All Occurance
Input Metode Pencarian
BFS
DFS
You choose : DFS
Interval:
0
mili-second
Search
1
A
12
D
4
C
2
B
13
Target.txt
6
G
5
F
3
E
11
J
9
I
8
H
7
Target.txt
10
Target.txt
Link Created:
D:\ProjectKuliah\Stima\Tubes2\A\D\Target.txt
D:\ProjectKuliah\Stima\Tubes2\A\C\G\I\Target.txt
D:\ProjectKuliah\Stima\Tubes2\A\C\G\Target.txt



4.5 Analisis Desain Solusi Algoritma BFS dan DFS

Pada contoh kasus di atas, baik DFS maupun BFS tampak memiliki efisiensi yang serupa. Akan tetapi perbedaan keduanya akan terlihat untuk directory yang besar dan letak file target yang spesifik. DFS akan bagus jika file yang dicari adalah file yang sangat dalam dan file tersebut terletak pada folder yang dicari dahulu setiap rekursi, tetapi DFS akan kurang bagus bila sebenarnya file target terletak di direktori yang dangkal namun tidak dicari terlebih dahulu oleh program. Sebaliknya, BFS akan lebih bagus bila file tersebut terletak pada direktori yang dangkal, tetapi akan kurang bagus bila file tersebut terletak di directory yang sangat jauh dari root. Kedua hal tersebut terjadi karena DFS mengutamakan kedalaman, sehingga akan lebih cepat mencari sesuatu yang dalam, sedangkan BFS mengutamakan pencarian menyamping sehingga akan bagus jika file target terletak pada directory yang dangkal.

Bab V

Kesimpulan dan Saran

Kelompok kami berhasil membuat program sesuai dengan spesifikasi tugas serta spesifikasi bonus tambahan. Jika dilihat lebih dalam, Algoritma BFS dan DFS kami bekerja dengan baik, tetapi kami tidak begitu fokus terhadap GUI dan safety checks sehingga GUI kami basic dan sebuah user dapat merusak program jika tidak melakukan dengan sesuai.

Saran yang dapat didapat adalah pada dasarnya Algoritma BFS dan DFS lumayan mudah dan sudah diberikan banyak *resource* untuk belajar membuat BFS dan DFS, sehingga waktu sisa dapat dialokasikan terhadap pembuatan GUI yang lengkap dan bersih, serta dalam fitur-fitur tambahan lainnya.

Daftar Pustaka

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag1.pdf
diakses 3/23/2022

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

diakses 3/23/2022

https://id.wikipedia.org/wiki/C_Sharp_(bahasa_pemrograman)

diakses 3/23/2022

Links

Repo Github : https://github.com/Vincent136/Tubes2_13520015.git

Demo Video : https://youtu.be/iMvCFH1h9eM