Laporan Tugas Besar II
IF2211 Strategi Algoritma
Pengaplikasian Algoritma BFS dan DFS dalam Implementasi Folder Crawling

Oleh:

Kelompok 55 - 13520011

Muhammad Akyas David Al Aleey	13520011
Fadil Fauzani	13520032
Vincent Prasetiya Atmadja	13520099

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
SEMESTER II 2021/2022

Daftar Isi

BAB I

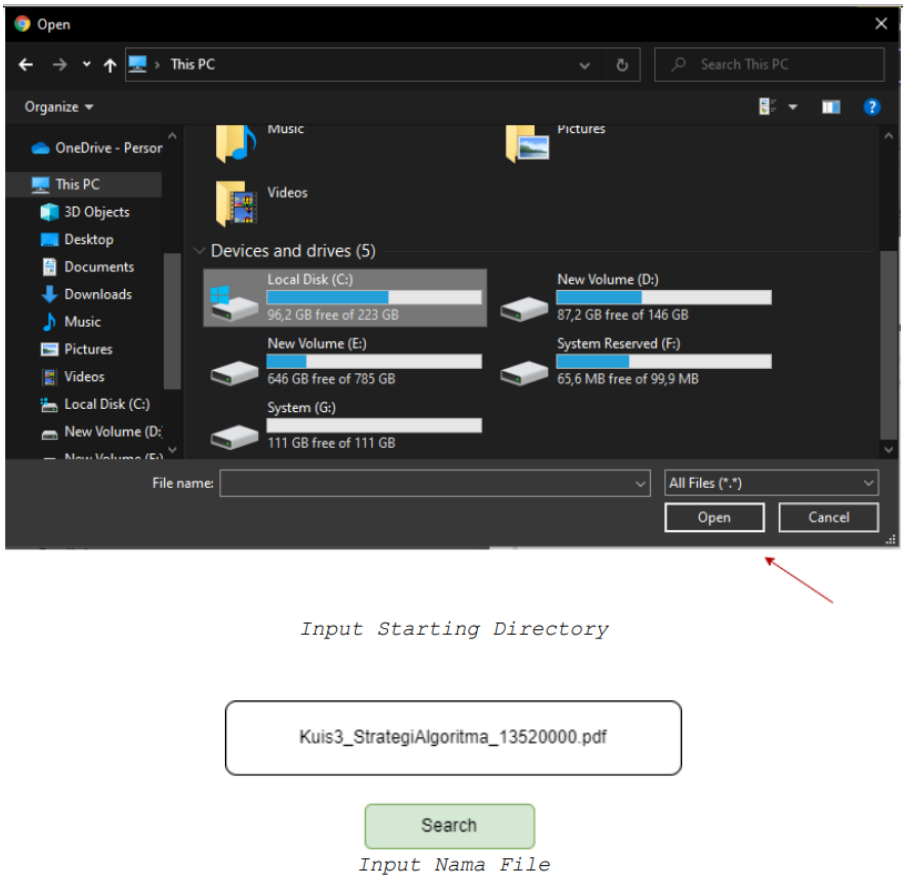
Deskripsi Tugas

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.

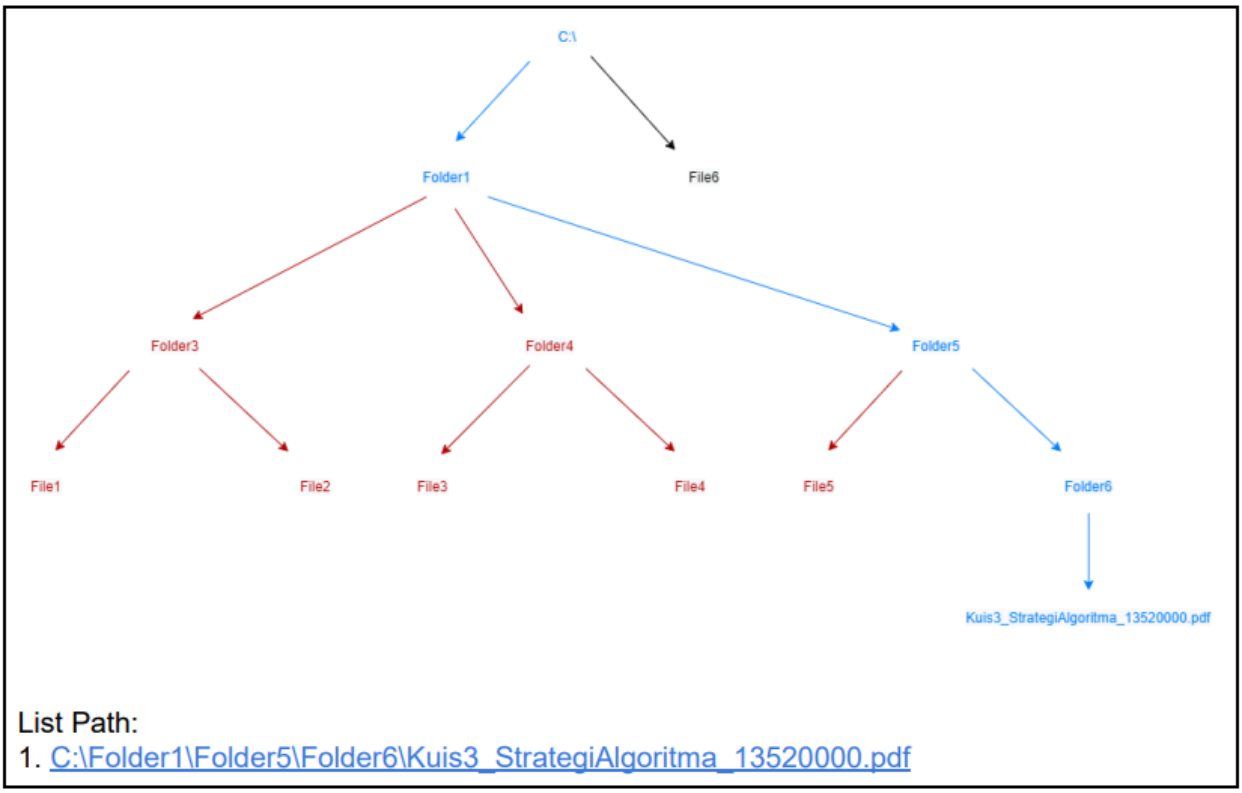
Contoh Input dan Output Program

Contoh masukan aplikasi:



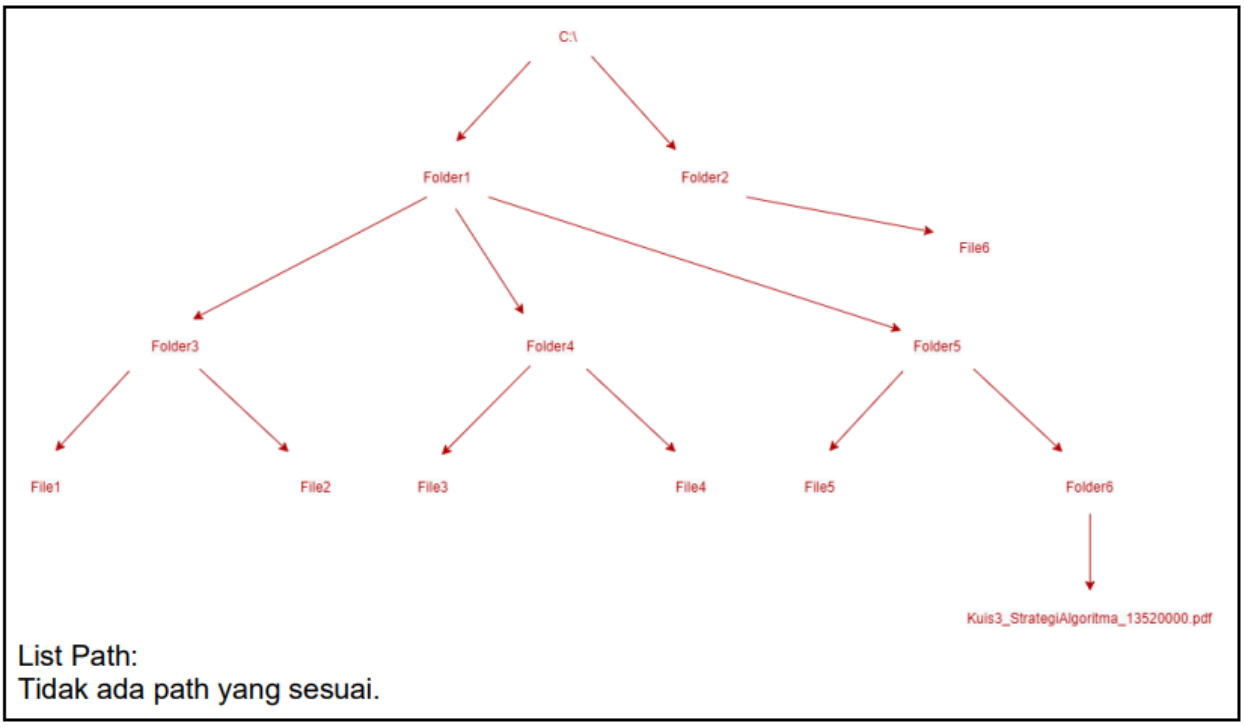
Gambar 1. Contoh Input Program

Contoh output aplikasi:



Gambar 2. Contoh Output Program

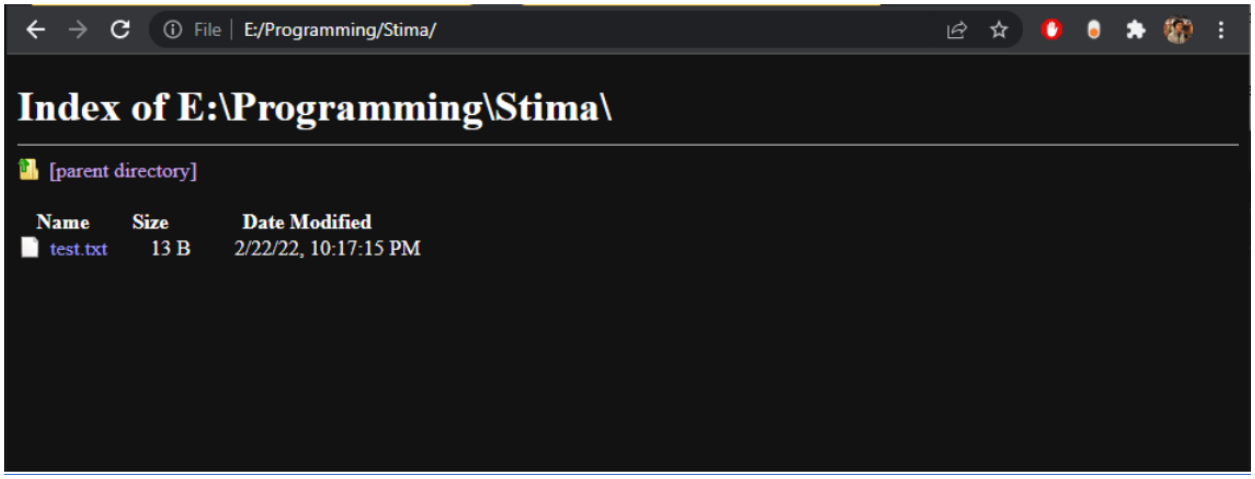
Misalnya pengguna ingin mengetahui langkah folder crawling untuk menemukan file Kuis3_StrategiAlgoritma_13520000.pdf. Maka, path pencarian DFS adalah sebagai berikut. C:\ → Folder1 → Folder3 → File1 → Folder3 → File2 → Folder3 → Folder1 → Folder4 → File3 → Folder4 → File4 → Folder4 → Folder1 → Folder5 → 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.



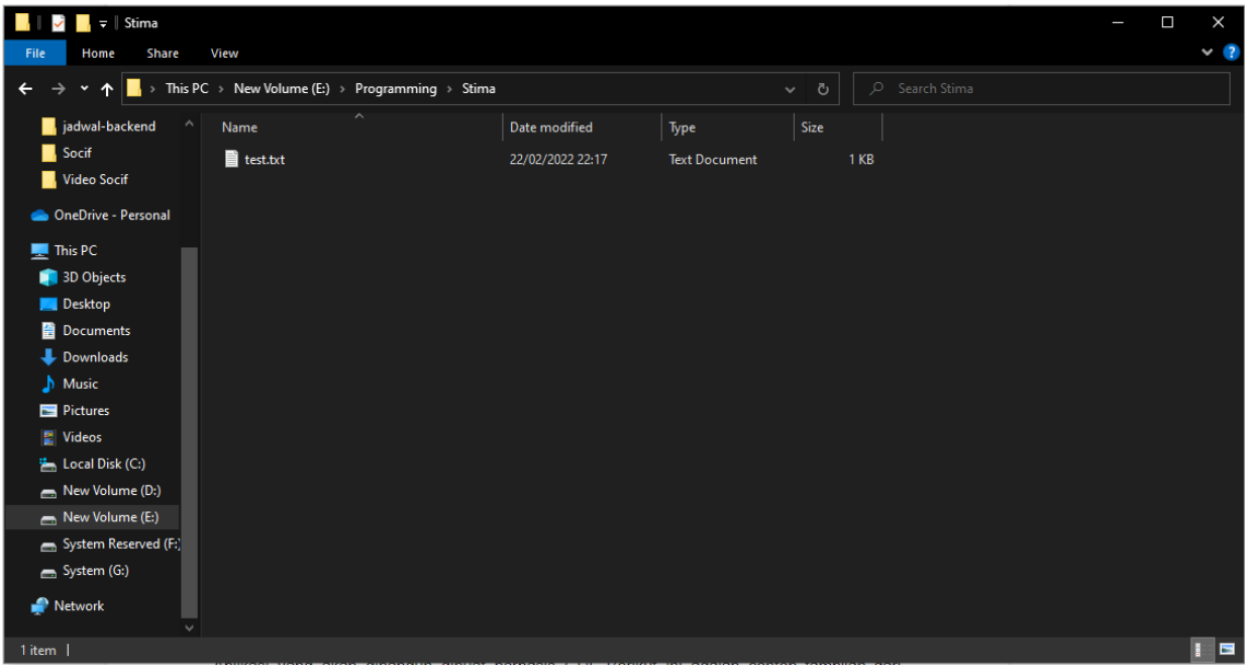
Gambar 3. Contoh Output Program Jika File Tidak Ditemukan

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.

Contoh Hyperlink Pada Path:



Gambar 4. Contoh Hyperlink Dibuka Melalui Browser



Gambar 5. Contoh Hyperlink Dibuka Melalui Penyimpanan Lokal

BAB II

Landasan Teori

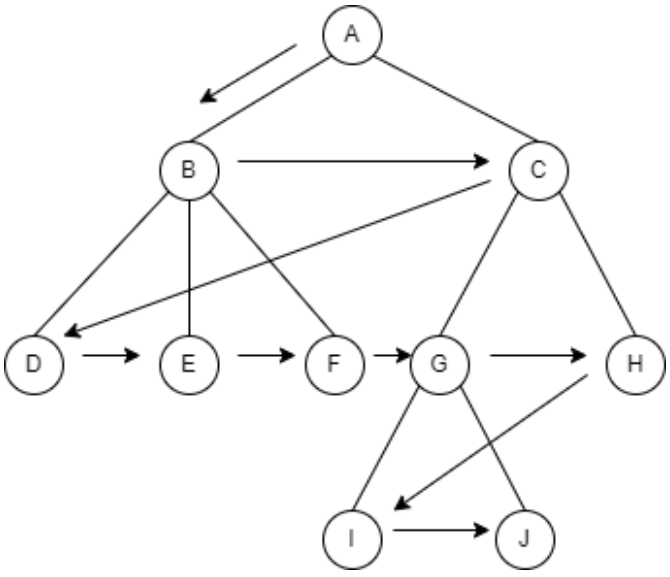
2.1. Definisi Graph Traversal, BFS dan DFS

2.1.1. Graph Traversal

Algoritma traversal di dalam graf merupakan algoritma yang mengiterasi atau mengunjungi setiap simpul-simpul pada graf dengan cara yang sistematik atau berdasarkan urutan tertentu. Algoritma ini terbagi menjadi dua, yaitu Algoritma Pencarian Melebar (*Breadth-First Search* atau BFS) dan Algoritma Pencarian Mendalam (*Depth-First Search* atau DFS). Baik BFS maupun DFS dapat digunakan untuk menyelesaikan pencarian dengan cukup baik. Namun, kriteria setiap algoritma tersebut berbeda sehingga menyebabkan perbedaan performansi di antara keduanya.

2.1.2. *Breadh-First Search* (BFS)

Algoritma Pencarian Melebar (*Breadth First Search* atau BFS) merupakan salah satu algoritma pencarian pada graf yang dimulai dengan mengunjungi sebuah simpul pada graf (V), kemudian mengunjungi semua simpul yang bertetangga dengan simpul V terlebih dahulu. Setelah semua tetangga telah dikunjungi, algoritma tersebut berlanjut dengan mengunjungi simpul yang belum dikunjungi dan bertetangga dengan simpul-simpul yang tadi dikunjungi. Langkah-langkah tersebut diterapkan seterusnya hingga ditemukan solusi yang diinginkan. Struktur data yang dimanfaatkan pada algoritma BFS ini yaitu struktur data Queue untuk menyimpan daftar simpul yang akan dikunjungi.

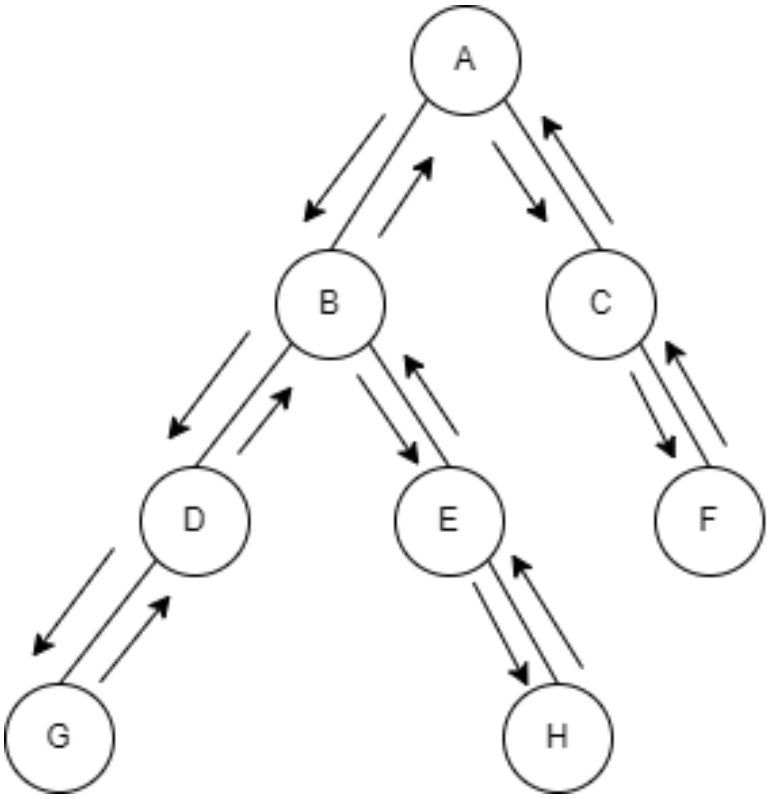


Gambar 6. Ilustrasi Alur Pencarian pada Algoritma BFS

Algoritma pencarian BFS memiliki kompleksitas waktu sebesar O(|V| + |E|) karena setiap vertex dan ujung (*edge*) akan dijelajahi dalam *worst case.*

2.1.3. DFS

DFS atau Depth First Search adalah salah satu algoritma pencarian pada struktur data Graph. Karakteristik algoritma ini adalah melakukan pencarian pada node anak terlebih dahulu. Pada DFS jika sudah di daun graph maka akan melakukan backtracking dan mencoba jalur lain yang masih bisa dilalui.



Gambar 7. Contoh Alur Pencarian pada Algoritma DFS

Algoritma pencarian DFS memiliki kompleksitas O(|V| + |E|), dengan |V| adalah banyaknya node pada graph dan |E| adalah banyaknya edge pada graph.

2.2. Penjelasan Singkat C# Desktop Application Development

Desktop Application merupakan sebuah perangkat lunak yang didesain agar dapat dijalankan secara lokal di desktop komputer yang dapat berupa aplikasi console atau *Graphical User Interface* (GUI). C# *Desktop Application Development* merupakan suatu pengembangan aplikasi yang berjalan di desktop dengan C# sebagai bahasa utama yang digunakan. Bahasa C# tergolong dalam salah satu bahasa pemrograman berorientasi objek diantara bahasa lainnya. Aplikasi desktop ini dikembangkan melalui sebuah IDE bernama Visual Studio yang dapat digunakan untuk melakukan pengembangan aplikasi, baik dalam bentuk aplikasi console,

aplikasi Windows, ataupun aplikasi Web. Pengembangan aplikasi ini dilakukan dengan memanfaatkan framework .NET yang salah satunya berisi WinForm atau Windows Form. WinForm adalah Class Library untuk mempermudah developer dalam membuat suatu aplikasi berbasis desktop karena terdapat banyak fitur yang menunjang developer seperti label, panel dan beberapa fitur lainya.

BAB III

Analisis Pemecahan Masalah

3.1. Langkah-Langkah Pemecahan Masalah

Untuk menyelesaikan permasalahan yang telah disebutkan di deskripsi persoalan, dilakukan beberapa langkah sebagai berikut:

1. Memahami permasalahan yang diberikan.
2. Memahami konsep algoritma BFS dan DFS dalam traversal graf.
3. Mempelajari bahasa C#.
4. Memetakan persoalan menjadi elemen-elemen BFS dan DFS. Kami membuat array of string berisi nama-nama folder yang berada di root, sehingga dapat membantu dalam penggunaan algoritma BFS dan DFS. Kami juga membuat array of string berisi nama-nama file dan mengecek tiap file di array tersebut apakah namanya sama dengan nama file yang dicari atau tidak.
5. Mengimplementasikan algoritma BFS dan DFS dalam bahasa C#.
6. Membuat GUI.
7. Menghubungkan GUI dengan program utama dan membuat visualisasi graf.

3.2. Proses Pemetaan Persoalan

Dalam pemetaan persoalan ke dalam algoritma BFS dan DFS, langkah-langkah yang digunakan berbeda pada tiap algoritma.

Berikut merupakan langkah-langkah cara kerja algoritma BFS pada persoalan ini.

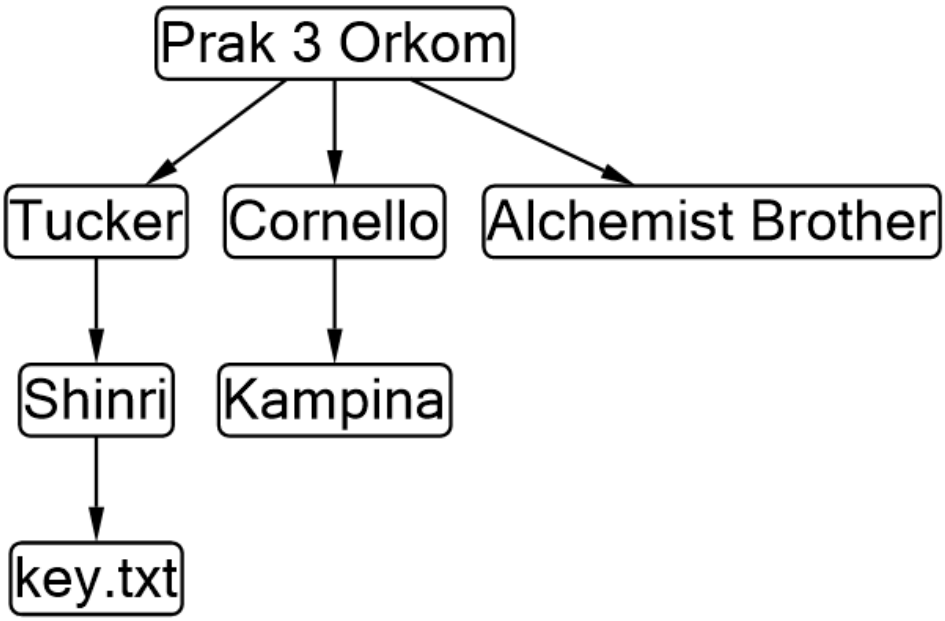
1. Menyimpan seluruh informasi direktori atau simpul dalam sebuah list of tuple dengan 3 parameter dengan parameter pertama bernilai 0 jika simpul tersebut berupa folder dan bernilai 1 jika simpul tersebut berupa file. Parameter kedua merupakan nama direktori, dan parameter ketiga merupakan *path* dari lokasi direktori. Kemudian menyimpan seluruh tetangga dari setiap direktori atau simpul.
2. Membuat array of boolean dengan ukuran sejumlah simpul untuk menandai apakah simpul tersebut sudah dikunjungi atau belum.
3. Membuat antrian atau queue untuk menyimpan direktory yang ingin dikunjungi.
4. Kunjungi elemen pertama dari queue. Jika elemen tersebut berupa folder, maka masukkan semua tetangga elemen ke dalam antrian. Jika berupa file, maka akan dicek apakah file tersebut sama dengan file yang ingin dicari.
5. Jika sama dan pengguna hanya mencari satu file saja, maka proses pencarian akan dihentikan.

6. Jika sama dan pengguna ingin mencari seluruh file yang bersusuaian, maka program akan tetap dilanjutkan (Elemen yang telah dikunjungi akan dikeluarkan dari antrian dan ulangi langkah 4).
7. Kumpulan solusi *path* direktori akan disimpan dalam sebuah variabel string.

Berikut merupakan langkah-langkah cara kerja algoritma DFS pada persoalan ini.

1. Ambil tiap nama file yang ada di dalam folder yang sedang dibuka.
2. Jika ada file yang namanya sama dengan nama file yang ingin dicari maka file sudah ketemu.
3. Jika tidak, maka ambil tiap nama folder yang ada di dalam folder yang sedang dibuka.
4. Pada tiap folder yang ada lakukan pencarian dfs (ulangi dari langkah 1).

3.3. Contoh Ilustrasi Kasus



Gambar 8. Kasus Folder Praktikum Orkom

Untuk Algoritma BFS, maka pencarian akan dimulai dari folder “Prak 3 Orkom”, di folder tersebut terdapat 3 folder yaitu Tucker, Cornello, dan Alchemist Brother, masukkan ketiga folder tersebut ke dalam Queue dengan melihat urunan abjadnya. Maka pertama pilih isi Queue terdepan (Alchemist Brother), setelah di Alchemist Brother tidak terdapat folder lagi dan tidak ditemukan file key.txt, maka lanjut ke isi Queue setelah Alchemist Brother, yaitu Cornello. Di Cornello terdapat Folder Kampina maka masukkan Folder Kampina ke Queue karena Cornello tidak terdapat file key.txt, maka lanjut ke isi Queue selanjutnya yaitu Tucker, di dalam Tucker terdapat folder Shinri maka masukkan folder tersebut ke Queue, karena di folder tucker tidak ada file key.txt lanjut ke isi Queue selanjutnya, yaitu Kampina. Di Kampina juga tidak ada folder maupun file lain, maka lanjut ke Queue selanjutnya (Shinri). Di Shinri tidak ada folder lain namun terdapat file key.txt maka BFS berhasil menemukan solusi, karena Queue

sudah kosong maka BFS selesai. Maka pada BFS urutan penulisannya adalah “Prak 3 Orkom” > Alchemist Brother > Cornello > Tucker > Kampina > Shinri

Untuk Algoritma DFS, pencarian juga dimulai dari folder “Prak 3 Orkom”. Terdapat 3 folder yaitu Tucker, Cornello, dan Alchemist Brother. DFS akan memilih folder dengan Abjad paling kecil terlebih dahulu yaitu Alchemist Brother, didalam Alchemist tidak ditemukan folder atau file apapun maka akan dilakukan backtracking dan dilanjutkan ke folder Cornello. Di Cornello terdapat file Kampina, maka masuk ke folder Kampina, di folder kampina tidak ditemukan folder atau file apapun, maka backtracking lagi ke folder Cornello, di folder Cornello tidak ada folder lagi yang dapat di telusuri maka backtracking lagi, dan melanjutkan penulusuran ke Folder Tucker, di Folder Tucker ditemukan folder Shinri maka masuk ke folder Shinri, di Folder Shinri tidak ditemukan folder lain namun terdapat file key.txt, maka DFS menemukan solusi. Backtracking ke folder Tucker, karena folder tucker tidak memiliki folder lain yang dapat ditelusuri maka backtracking lagi ke folder Prak 3 Orkom, karena semua folder di Prak 3 Orkom telah di telurusi maka DFS selesai, maka urutan penelusurannya adalah “Prak 3 Orkom” > Alchemist Brother > Cornello > Kampina > Tucker > Shinri

BAB IV

Implementasi dan Pengujian

4.1. Implementasi Program

```
main(){
  input(path)
  input(filename)
  input(algoritma)
  input(findAll)
  if(algoritma = “BFS”) then
    Bfs <- new BFS(filename, path, findAll)
    Bfs.run()
  else if (algoritma = “DFS”) then
    Dfs <- new DFS(0, filename, path, findAll)
    Dfs.run()
}
```

4.2. Penjelasan Struktur Data yang Digunakan

Kelas BFS
Deskripsi Kelas: - Digunakan untuk metode BFS.
Atribut Kelas: 1. filename : berisikan nama file yang dicari. 2. startingFolder : bersisikan nama folder awal pada pencarian. 3. findAll : penanda apakah pencarian mencari semua keberadaan file atau tidak. 4. found : penanda keberhasilan BFS dalam mencari file. 5. solution : berisi solusi BFS 6. AdjacentVertices : Menyimpan seluruh direktori yang menjadi tetangga suatu simpul. 7. ListDirectories : Menyimpan seluruh informasi setiap simpul.
Method Kelas 1. BFS : Konstruktor 2. MakeGraph : Mengiterasi seluruh direktori dimulai dari startingFolder kemudian mengisi AdjacentVertices serta ListDirectories 3. DoBFS : melakukan BFS pada input yang diberikan 4. Run : fungsi pembantu untuk digunakan pada main

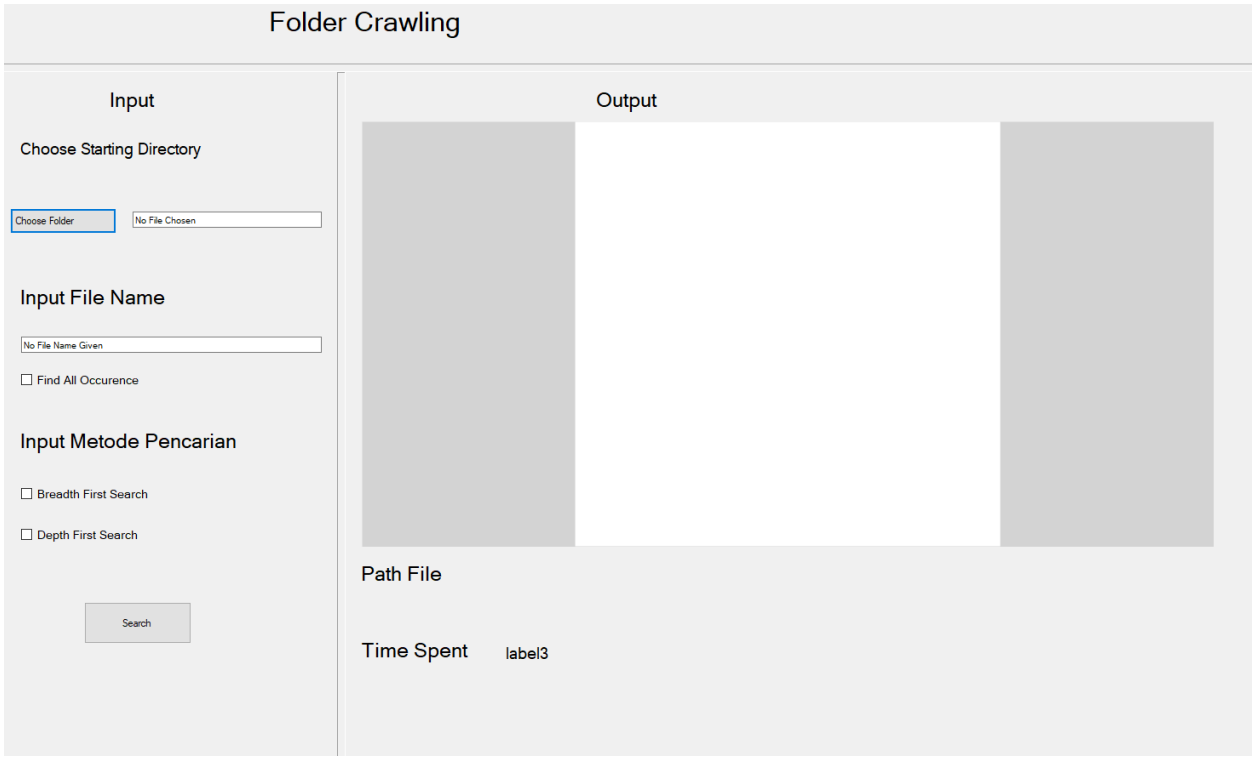
Kelas DFS
Deskripsi Kelas: - Digunakan untuk metode DFS.
Atribut Kelas: 1. filename : berisikan nama file yang dicari. 2. startingFolder : berisikan nama folder awal pada pencarian. 3. Level : berisikan level pada graf folder DFS berlangsung. 4. findAll : penanda apakah pencarian mencari semua keberadaan file atau tidak. 5. solution : berisi solusi DFS. 6. found : penanda keberhasilan DFS dalam mencari file.
Method Kelas 1. DFS : Konstruktor 2. doDFS : melakukan DFS pada input yang diberikan 3. Run : fungsi pembantu untuk digunakan pada main

Kelas Utils
Deskripsi Kelas: Berisikan method pembantu dalam file management
Atribut Kelas -
Method Kelas 1. getFolders : mengembalikan semua nama folder yang ada di folder input 2. getFiles : mengembalikan semua nama file yang ada di folder input

4.3. Penjelasan Tata Cara Penggunaan Program

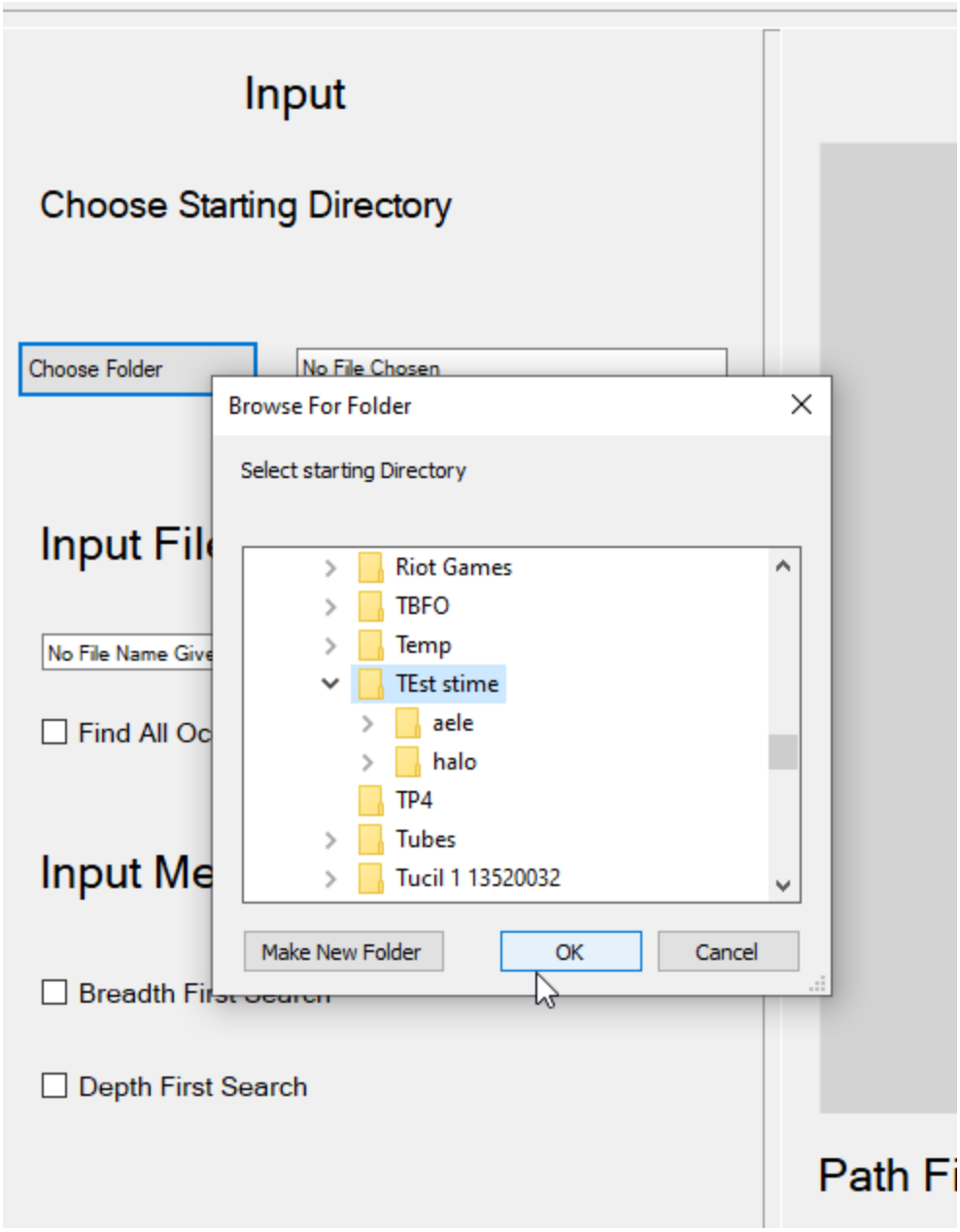
Berikut langkah-langkah penggunaan program:

1. Buka Aplikasi



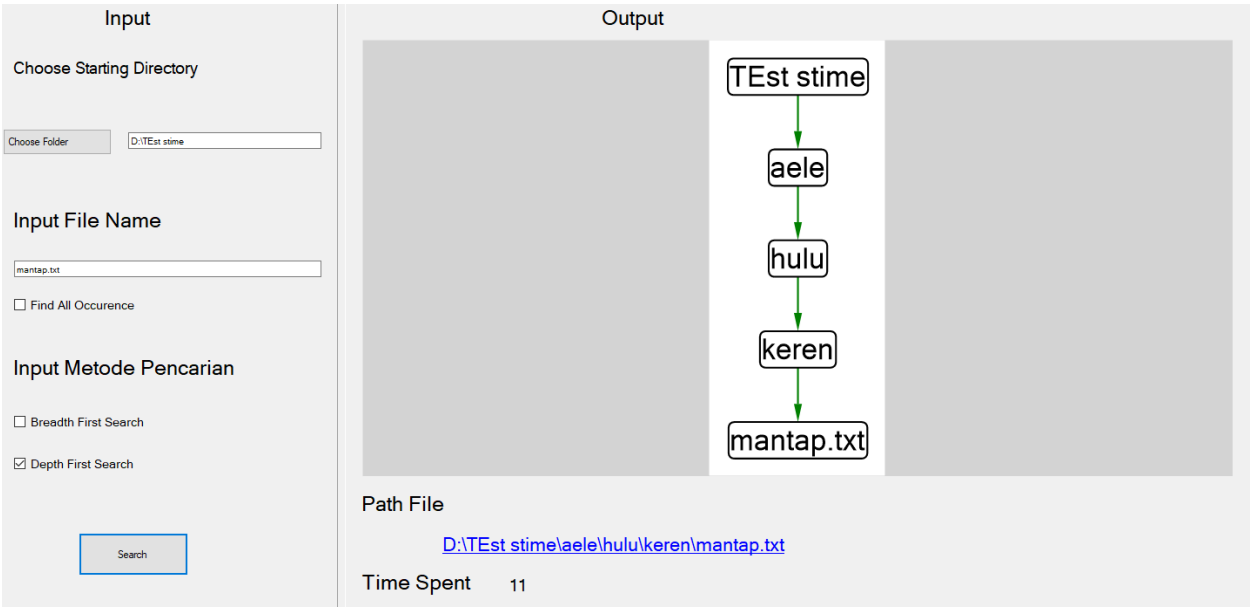
Gambar 9. Tampilan Aplikasi

2. Tekan Tombol “Choose Folder”, lalu pilih folder awal yang ingin dipakai, lalu tekan OK.



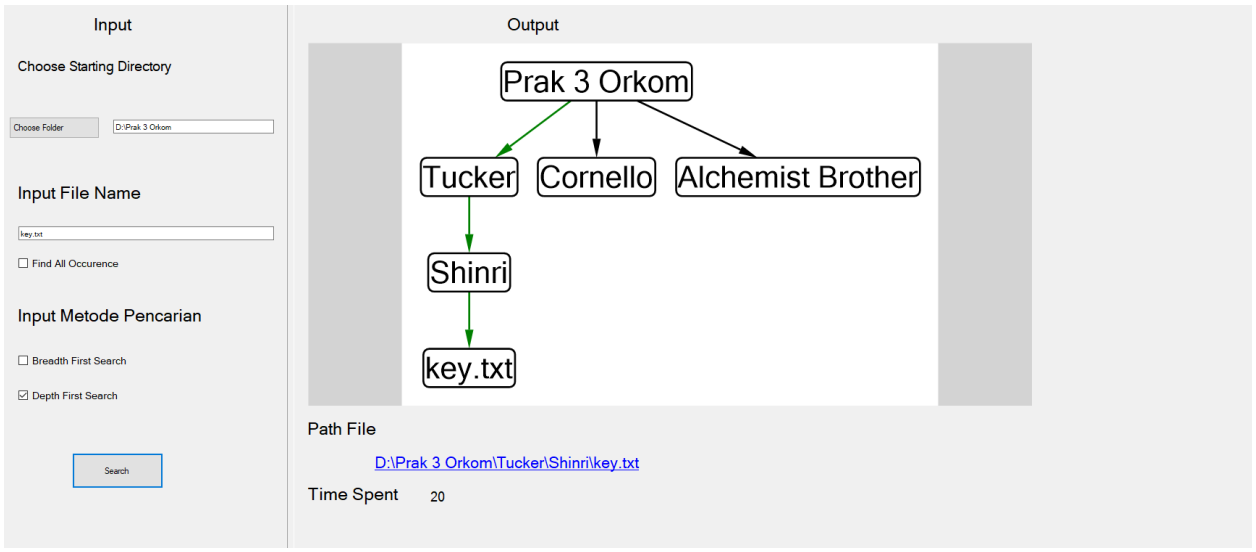
Gambar 10. Tampilan ketika memilih folder awal

3. Setelah memilih folder, isi box input file dengan nama file yang ingin dicari, beri check pada check-box find-all jika ingin mencari semua keberadaan file tersebut.
4. Lalu, pilih metode pencarian yang dilakukan dengan membebri check pada check-box BFS atau DFS.
5. Tekan tombol Search.
6. Hasilnya akan tampil di layar samping.

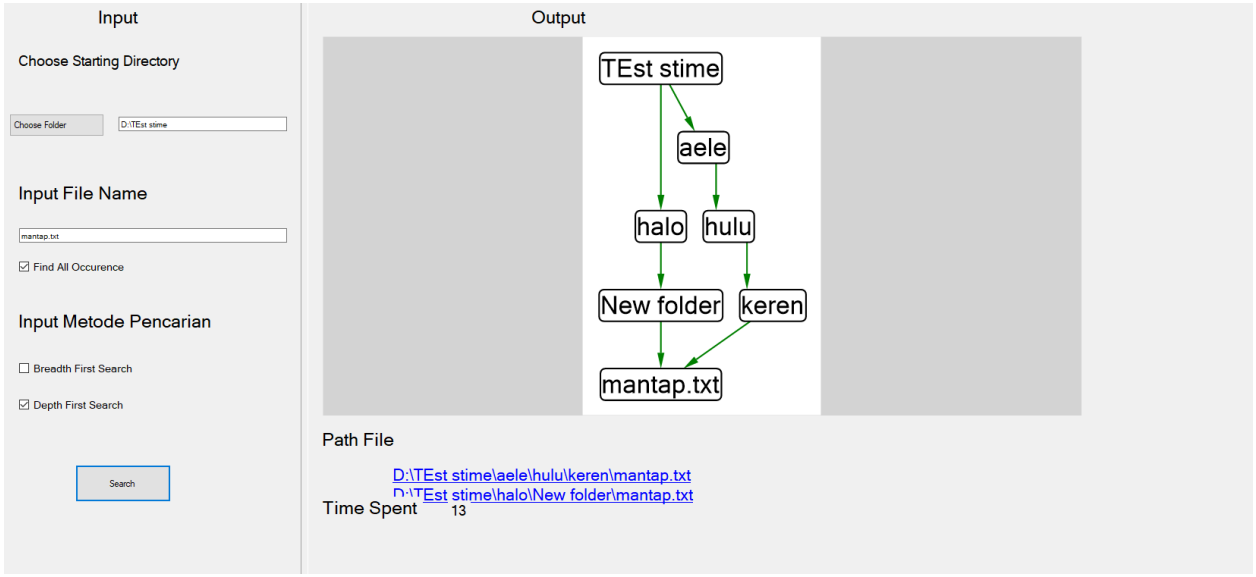


Gambar 11. Tampilan Hasil Proses Pencarian File

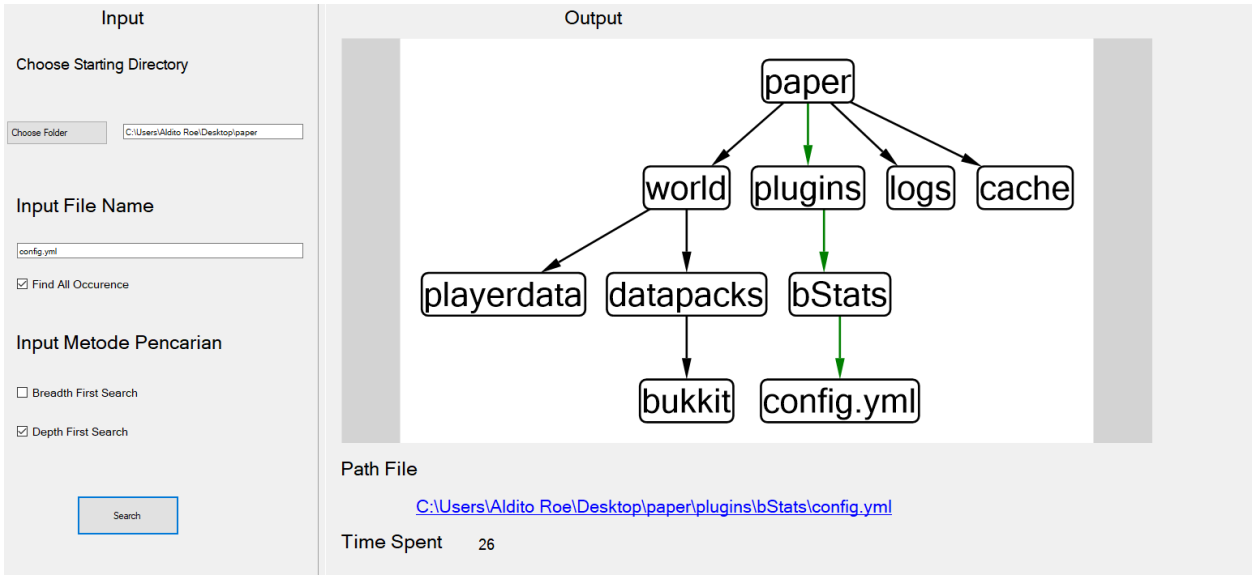
4.4. Hasil Pengujian



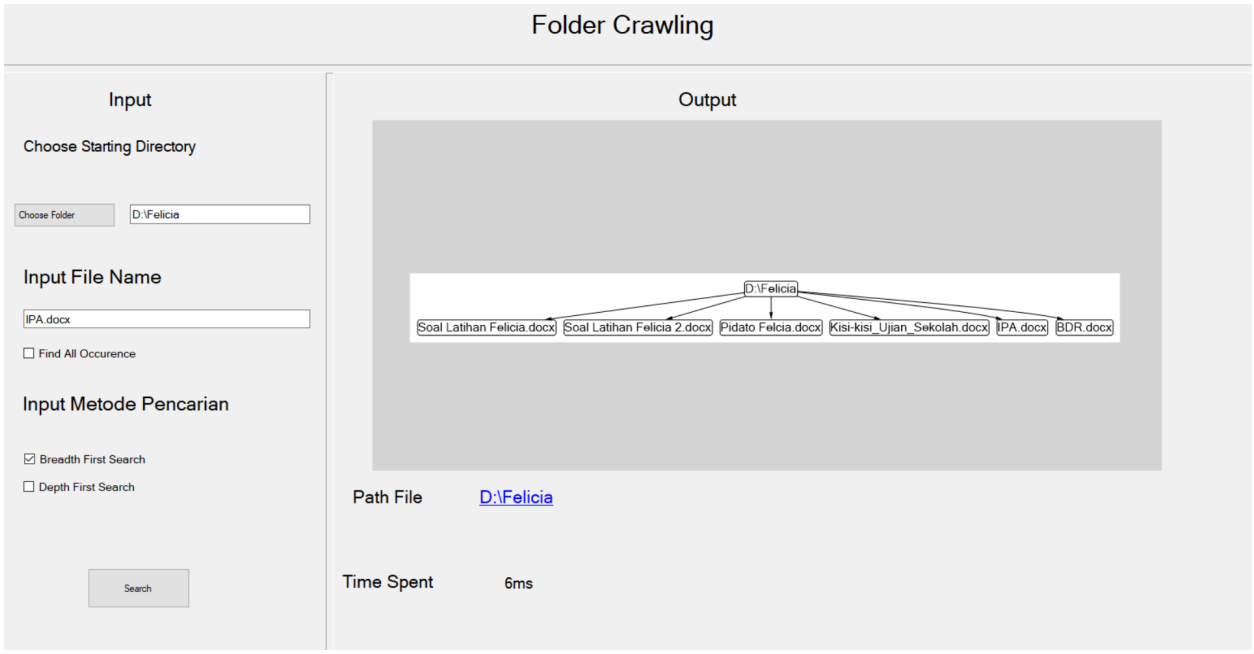
Gambar 12. Hasil DFS 1



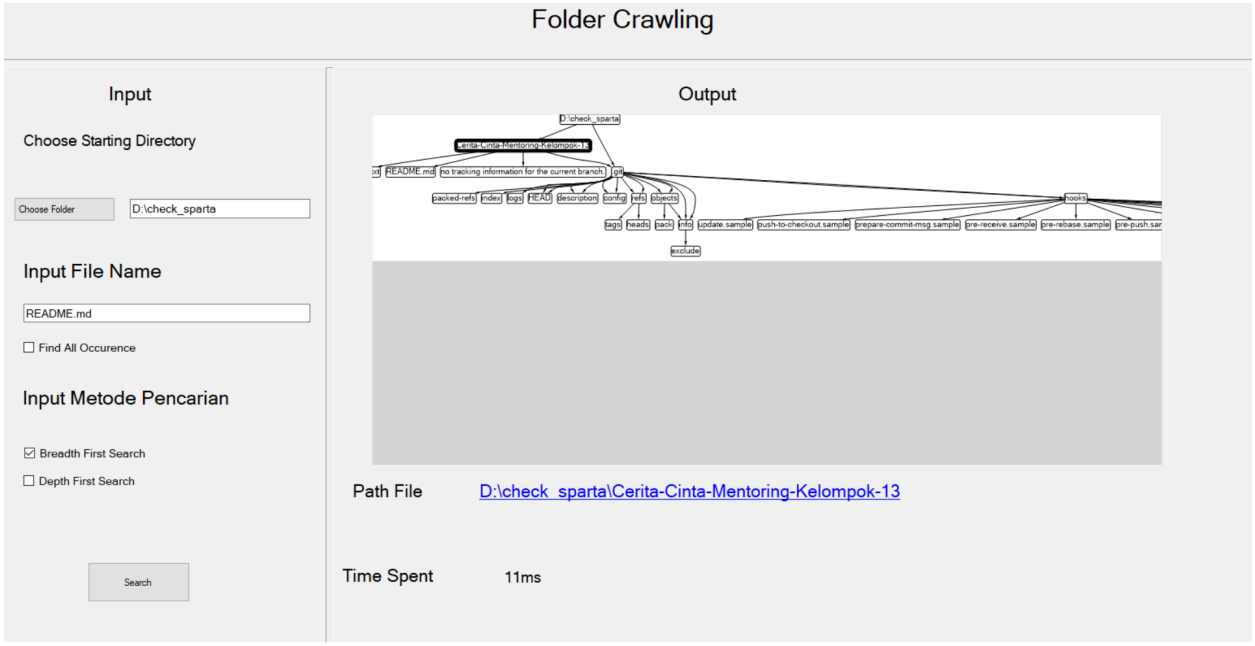
Gambar 13. Hasil DFS 2



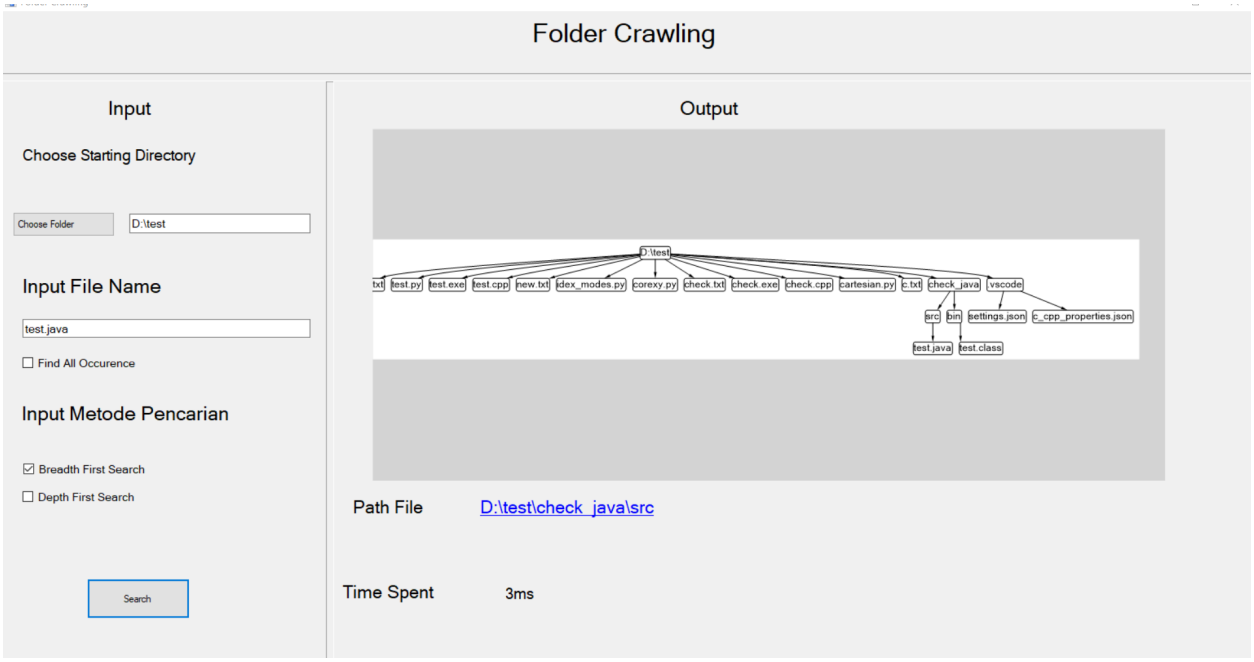
Gambar 14. Hasil DFS 3



Gambar 15. Hasil BFS 1



Gambar 16. Hasil BFS 2



Gambar 17. Hasil BFS 3

4.5. Analisis Desain Solusi

Dari berbagai percobaan yang telah dilakukan, selalu ditemukan solusi dari permasalahan baik dengan menggunakan algoritma DFS maupun DFS. Dengan menganalisis hasil percobaan yang telah dilakukan, didapat bahwa kedua algoritma sejatinya tidak dapat dibandingkan dari segi kecepatan karena waktu untuk menemukan solusinya rata-rata hanya berbeda sepersekian detik. Akan tetapi, kompleksitas algoritma dari kedua algoritma tersebut cenderung berbeda untuk setiap kasus. Jika pada suatu percobaan, simpul tujuan pada representasi graf memiliki kedalaman yang relatif tinggi terhadap simpul awal, maka kompleksitas dari algoritma BFS akan cenderung lebih besar dibandingkan dengan kompleksitas algoritma DFS. Sebaliknya, jika pada suatu percobaan, simpul tujuan pada representasi graf memiliki jarak yang relatif pendek terhadap simpul awal, maka kompleksitas dari algoritma DFS akan cendering lebih besar dibandingkan dengan kompleksitas dari algoritma BFS. Oleh karena itu, kedua algoritma tersebut memiliki kelebihan dan kekurangan tersendiri yang dipengaruhi oleh posisi ketercapaian simpul tujuan terhadap simpul awal.

BAB V

Kesimpulan dan Saran

5.1. Kesimpulan

Dari Tugas Besar IF2211 Strategi Algoritma semester 2 2021/2022 berjudul "Pengaplikasian Algoritma BFS dan DFS dalam Implementasi Folder Crawling", kami berhasil mengembangkan sebuah program yang dapat mencari file yang ditentukan pada folder yang dipilih dengan memanfaatkan algoritma BFS dan DFS seperti yang sudah diajarkan pada kelas. Kami juga berhasil merepresentasikan pencarian tersebut dalam suatu graph yang terlihat pada tampilan aplikasi program kami. Secara garis besar, tugas ini telah mengajarkan kami untuk berpikir kreatif dalam membuat algoritma BFS, DFS, serta membuat tampilan aplikasi yang mudah dipakai.

5.2. Saran

Saran yang dapat kami berikan untuk tugas besar IF 2211 Strategi Algoritma Semester 2 2021/2022 meliputi:

1. Membuat representasi graf menjadi lebih baik dan lebih rapi lagi
2. Membuat tampilan aplikasi menjadi lebih bagus.

Daftar Pustaka

Munir, Rinaldi, and Nur Ulfa Maulidevi. “Breadth/Depth First Search (BFS/DFS).” *Informatika*, 2022, Diakses online dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag1.pdf. pada 25 Maret 2022.

Munir, Rinaldi, and Nur Ulfa Maulidevi. “Breadth/Depth First Search (BFS/DFS).” *Informatika*, 2022, Diakses online dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag2.pdf. pada 25 Maret 2022.