LAPORAN TUGAS BESAR STRATEGI ALGORITMA

Pengaplikasian Algoritma BFS dan DFS dalam Implementasi Folder Crawling

IF2211 STRATEGI ALGORITMA

Dosen Pengajar : Dr. Masayu Leylia Khodra, S.T., M.T.

Kelompok : Search Breathing

Disusun oleh :

Ilham Prasetyo Wibowo (13520013)

Muhammad Akmal Arifin (1320037)

Gregorius Moses Marevson (13520052)

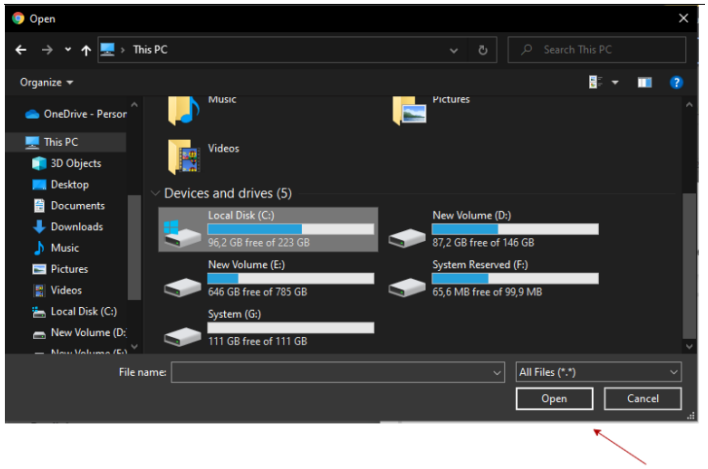
PROGRAM STUDI TEKNIK INFORMATIKA
SETKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2021

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.

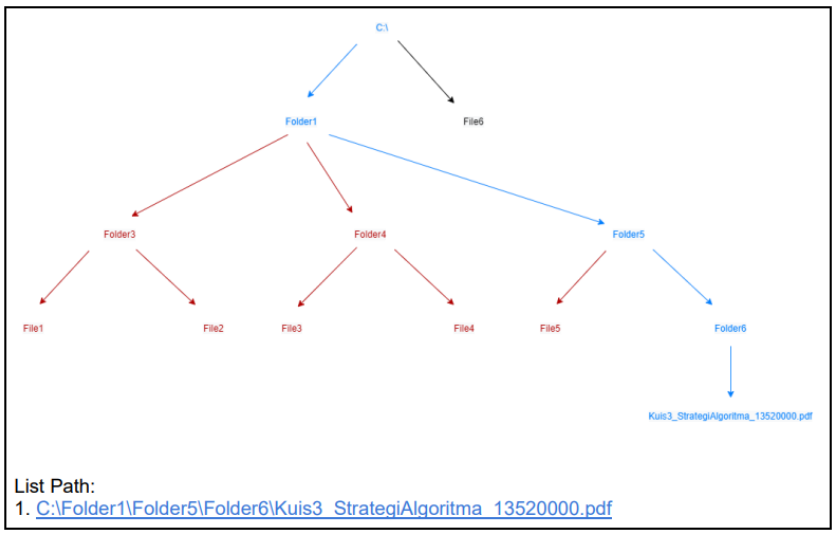
Selain pohon, Anda diminta juga menampilkan list path dari daun-daun yang bersesuaian dengan hasil pencarian. Path tersebut diharuskan memiliki hyperlink menuju folder parent dari file yang dicari, agar file langsung dapat diakses melalui browser atau file explorer. Contoh hal-hal yang dimaksud akan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



Input Starting Directory

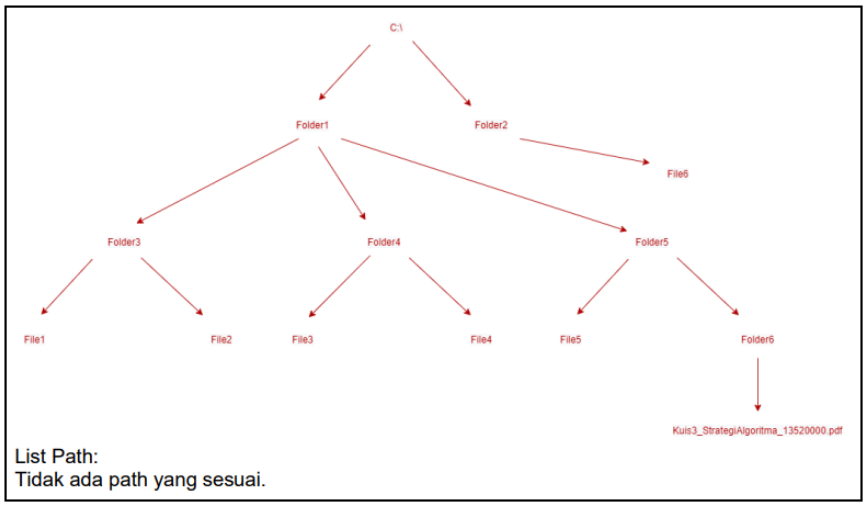
Input Nama File

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 antrean tapi belum diperiksa diberi warna hitam. Anda bebas menentukan warnanya asalkan dibedakan antara ketiga hal tersebut.



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

A. Dasar Teori

1. Traversal Graf

Algoritma traversal graf adalah algoritma mengunjungi simpul secara sistematik. Algoritma traversal graf merupakan salah satu algoritma pencarian solusi untuk representasi persoalan menggunakan graf.

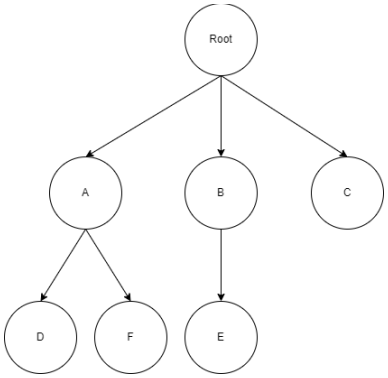
Algortima Pencarian Solusi

2. Tanpa Informasi
   Contoh dari pencarian solusi tanpa informasi adalah Depth First Search(DFS), Breadth First Search(BFS), Depth Limited Search, Iterative Deepening Search, Uniform Cost Search.
3. Dengan informasi
   Pencarian solusi berbasis heuristik, contohnya adalah Best First Search.

Breadth First Search

Algoritma pencarian melebar (Breadth First Search) menggunakan antrean dalam pencariannya. Pencarian dimulai dari akar graf, kemudian akan dicari setiap simpul yang bersisian dengan akar graf. Setiap simpul tersebut akan masuk ke dalam antrean. Proses akan diulang untuk setiap elemen pada antrean. Proses pencarian berhenti ketika elemen sudah ditemukan atau antrean kosong (semua simpul sudah dikunjungi).

Ilustrasi :

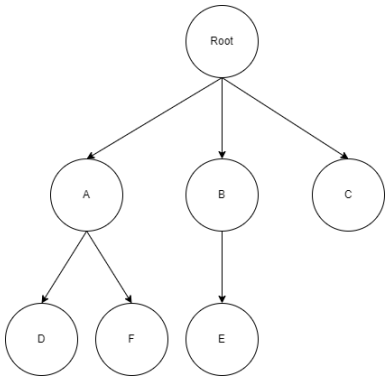


Pada pencarian di atas, algoritma BFS akan memulai pada simpul Root, kemudian memasukkan A, B, C ke dalam antrean. Kemudian, simpul A akan diproses dan simpul D, F akan dimasukkan ke dalam antrean. Jadi rute pencarian menggunakan BFS adalah Root – A – B – C – D – F – E.

Depth First Search

Berbeda dengan BFS yang melebar, DFS melakukan pencarian secara mendalam. Artinya BFS akan mencari simpul sampai simpul tersebut adalah simpul daun. Jika simpul tersebut adalah elemen yang akan dicari maka pencarian dihentikan. Jika belum ditemukan maka algoritma akan melakukan runut balik (backtrack) pada

simpul sebelumnya dan mengulang langkah pencarian. Pencarian DFS ini bisa diimplementasikan menggunakan rekursif atau menggunakan struktur data tumpukan (stack).
Ilustrasi :



Rute pencarian : Root – A – D – A – F – A – Root – B – E – B – Root - C

B. Pengembangan Desktop App dengan C#

Desktop merupakan sebuah teknologi komputer pribadi yang dapat menjalankan berbagai macam program yang terinstall di dalamnya. Terdapat 3 jenis Operating System pada Desktop yang umum saat ini, yaitu Macintosh, Windows, dan Linux. Pada tugas besar ini, operating system yang digunakan pada desktop adalah Windows

Pengembangan aplikasi tugas besar ini menggunakan platform Visual Studio. Visual Studio merupakan sebuah perangkat lunak lengkap yang dapat digunakan untuk melakukan pengembangan aplikasi, baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasinya, dalam bentuk aplikasi console, aplikasi Windows, ataupun aplikasi Web. Penggunaan Visual Studio dalam pengembangan aplikasi ini memudahkan pembuatan GUI aplikasi dikarenakan sudah tersedia fitur untuk membuat GUI pada Visual Studio. GUI pada aplikasi ini menggunakan Windows Form. Sedangkan untuk pemilihan bahasa pada pembuatan aplikasi ini menggunakan C#.

C# atau yang dibaca C sharp adalah bahasa pemrograman sederhana yang digunakan untuk tujuan mum, dalam artian bahasa pemrograman ini dapat digunakan untuk berbagai fungsi misalnya untuk pemrograman server-side pada website, membangun aplikasi desktop ataupun mobile, pemrograman game dan sebagainya. Selain itu C# juga bahasa pemrograman yang berorientasi objek, sehingga C# mengusung konsep objek seperti *inheritance*, *class*, *polymorphism*, dan *ecapsulation*. Akan tetapi dalam prakteknya C# sangat bergantung dengan framework yang disebut .NET Framework, *framework* inilah yang nanti digunakan untuk men-*compile* dan menjalankan kode C#

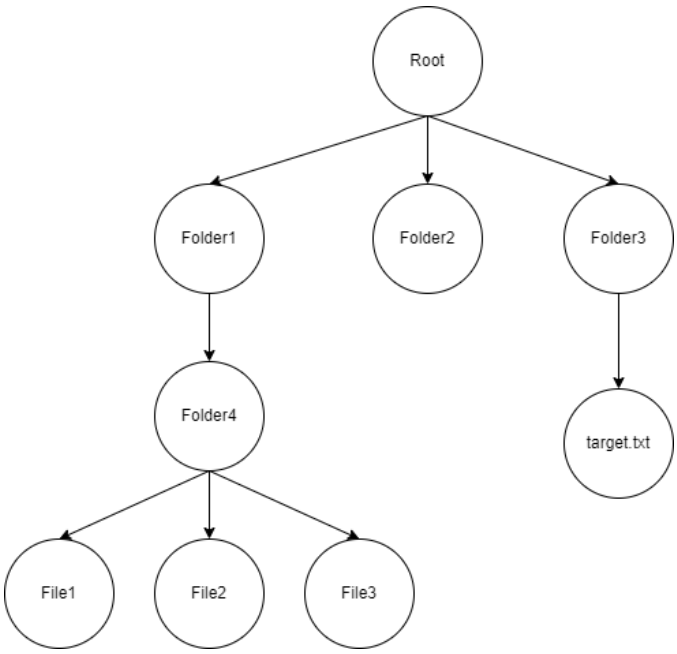
BAB III

PEMECAHAN MASALAH

1. Langkah – langkah Pemecahan Masalah
   Permasalahan akan dibagi menjadi 3 bagian yaitu GUI, algoritma DFS, dan algoritma BFS. Program akan ditulis menggunakan bahasa C#. Pengembangan aplikasi akan menggunakan Microsoft Visual Studio.

2. Mapping Persoalan
   a. GUI
      1. Graf
         Graf digunakan untuk visualisasi dari pencarian menggunakan algoritma BFS maupun DFS.
      2. Forms
         Untuk media input/output sistem dengan pengguna.
   b. Pencarian Solusi
      Setiap folder dan file akan menjadi sebuah simpul yang nanti akan dikunjungi oleh algoritma pencarian. Subdirektori dari sebuah folder adalah child dari folder tersebut. Dengan demikian akan terbentuk sebuah pohon yang akan merepresentasikan pencarian solusi dalam folder.
      Pencarian solusi juga akan menggunakan berbagai struktur data terutama antrean. Antrean digunakan dalam algoritma BFS.

3. Ilustrasi Kasus Lain
   Ilustrasi : Misalkan struktur dari folder dan file dari root adalah sebagai berikut.



Misalkan file yang akan dicari adalah target.txt. Jika pencarian solusi menggunakan algoritma BFS, maka pencarian akan melebar terlebih dahulu sampai bertemu dengan target.txt. Pencarian dimulai dari folder root yang akan menghasilkan Folder1, Folder2, dan Folder3 kemudian dimasukkan ke dalam antrean. Proses akan terus diulang (mencari

subdirektori dari folder dan dimasukkan ke dalam antrean) sampai antrean kosong atau file sudah ditemukan. Untuk kasus gambar di atas, path file yang terbentuk adalah :

Root > Folder1 > Folder2 > Folder3 > Folder4 > target.txt

Jika file tidak ditemukan (tujuan file bukan target txt) maka rute pencarian yang terbentuk adalah :

Root > Folder1 > Folder2 > Folder3 > Folder4 > target.txt > File1 > File2 > File3

Berbeda dengan pencarian solusi menggunakan algortima BFS, algoritma DFS akan mencari simpul terdalam terlebih dahulu. Proses ini bisa menggunakan algoritma rekursif atau memanfaatkan struktur data tumpukan (stack). Setiap simpul yang dikunjungi akan menghasilkan simpul-simpul lainnya terlebih dahulu (dalam hal ini subdirektori) sampai ditemukan daun dari tree (simpul yang tidak memiliki child lagi). Kemudian akan dilakukakn pengecekan, jika file tidak sesuai maka akan dilakukan runut balik (backtracking) dan mencari simpul lainnya. Untuk gambar di atas, rute pencarian file target.txt adalah :

Root > Folder1 > Folder4 > File1 > Folder4 > File2 > Folder4 > File3 > Folder4 > Folder1 > Root > Folder2 > Root > Folder3 > target.txt

Dalam kasus ini, jika pencarian tidak ditemukan maka rute pencarian file adalah sama.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

1. Implementasi Program

Pseudocode algoritma BFS
function bfs(dir:string, filename : string, findall:Boolean)

KAMUS
currentdir:string
stopsearch:boolean
q:Queue
dupes:Queue

ALGORITMA
q.Enqueue(dir);
stopsearch = false;

```
while (q.Count > 0 and not stopsearch) do
        currentdir = q.Dequeue();
        dp = dupes.Dequeue();
        //enqueue folders
        for (di in subDirectories(currentdir))
            q.Enqueue(di);
            dupes.Enqueue(countDuplicates(graph, di))
            Tambah simpul folder kedalam graf, warna merah.

        //search filename
        for (f in subFiles(currentdir))
          if (Path.GetFileName(f) == filename) //solusi ditemukan
             this.solution.Add(f) // masukkan solusi ke dalam list solusi
            Tambah simpul kedalam graf, warna biru, kemudian warnai simpul
            yang ditemukan sampai root.
             if (!findall)
                    colorQueue(q,ref this.graph) //masukkan sisa queue kedalam
                    graf, warnai queue yang masih tersisa, warna hitam
                    stopsearch = true;

          else

            Masukkan nama file kedalam graf, warna merah.
If (solution = empty) then

        Warnai root node merah.
```

Pseudocode Algortima DFS

```
Function TraverseTree(root: string, filename: string, graph: Graph,
findAll: bool) -> List

KAMUS
Dirs : Stack
Result: List
Founded: bool
CurrentDir : string
SubDir, files: array of string

ALGORITMA
Founded <- false
Push root to dirs
While (count of dirs > 0) do
    Pop root to currentDir
    SubDirs <- getDirectory of root
    Files = getFiles of root
    Foreach (file in files) do
        if (file = filename) then
            add file to result
            Paint tree to root with blue color
            Set found to true if not findAll
        If (not founded) then
            Add tree with red color
        Else
            Add tree with black color

    Foreach (str in subDirs) do
        Add tree with black color
        Push str to dirs
    If (not founded and subDirs = 0) do
        Paint to the root with red color
    Else if (founded) do
        Paint to the root with blur color
        Break
    End while

If (result > 0) do
    Paint to the root of all result with blue color
```

2. Struktur Data
   Struktur data yang digunakan pada program ini adalah
   a. List

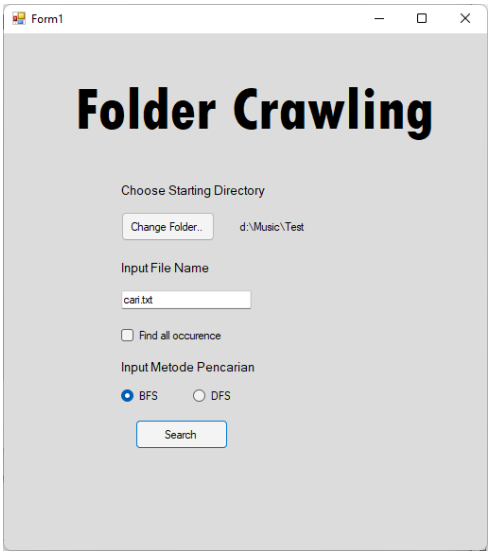
Untuk menampung hasil path dari pencarian.

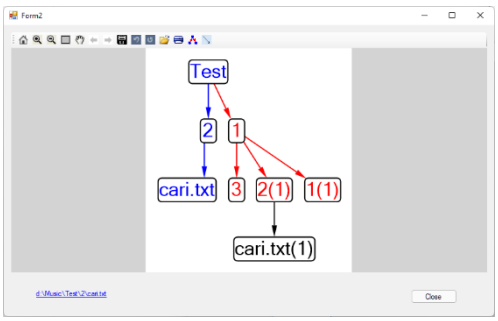
b. Queue
   Queue digunakan dalam algoritma BFS.
c. Stack
   Stack digunakan dalam algoritma DFS.
d. Graph
   Graph digunakan untuk melakukan visualisasi dari struktur folder yang telah dicari.

3. Tata Cara Penggunaan Program

Langkah-langkah untuk menggunakan program ini adalah

a. Buka executable file yang ada pada folder bin
b. Pilih folder untuk memulai pencarian dengan cara menekan tombol yang bertuliskan “Choose Folder”
c. Kemudian pilih folder yang diinginkan, lalu tekan tombol Ok
d. Masukkan nama file yang ingin dicari pada text box yang terdapat di bawah tulisan “Input File Name”
e. Centang check box Find all occurrence apabila ingin mencari semua file yang bisa ditemukan.
f. Terakhir pilih metode pencarian BFS atau DFS
g. Setelah memastikan semua masukkan benar, tekan tombol Search untuk memulai pencarian
h. Pada jendela yang baru muncul akan terlihat hasil graf pencarian
i. Untuk membuka folder file yang dicari, klik *hyperlink* yang ada pada bagian kiri bawah jendela.
j. Untuk menutup jendela yang baru, tekan tombol Close



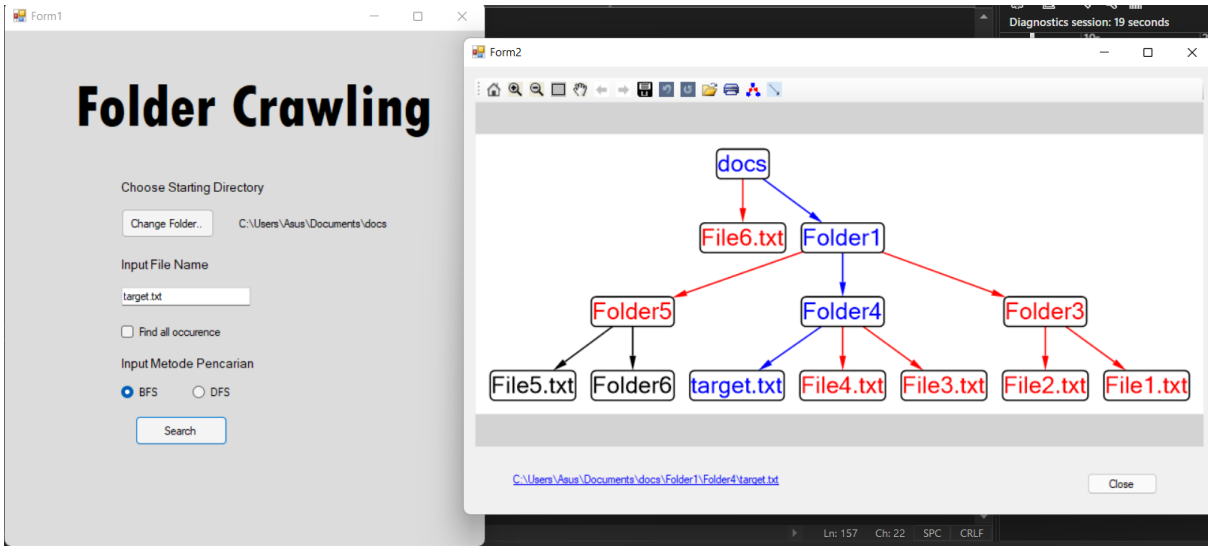


4. Hasil Pengujian
   Algoritma BFS
   Pengujian dilakukan dengan menggunakan struktur folder sebagai berikut.

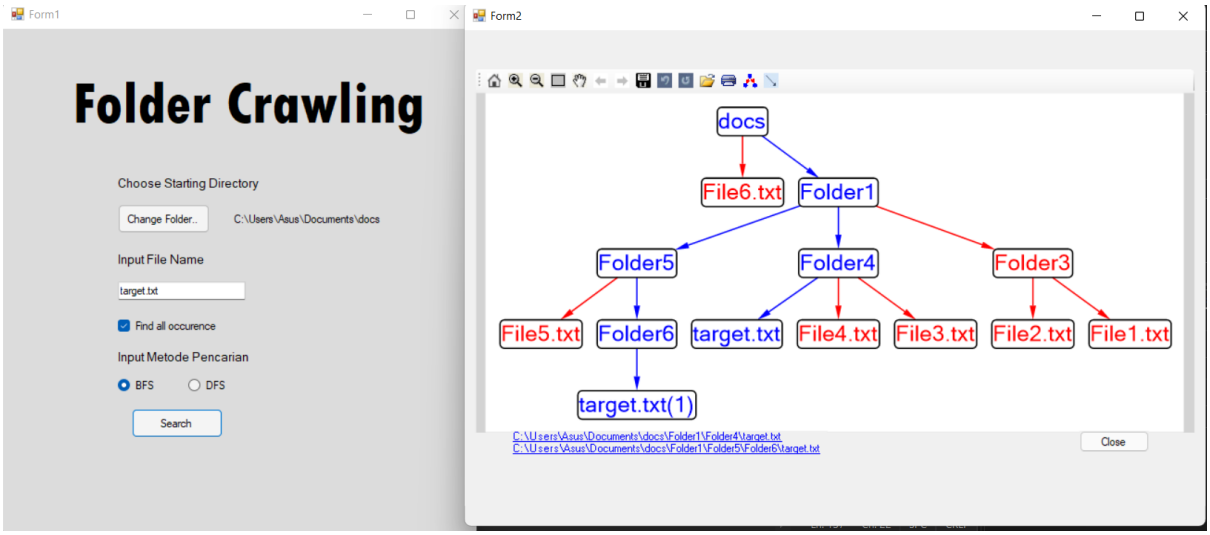
```
docs
  Folder1
      Folder3
          File1.txt
          File2.txt
      Folder4
          File3.txt
          File4.txt
          target.txt
      Folder5
          Folder6
              target.txt
          File5.txt
  File1.txt
```

Hasil pengujian :

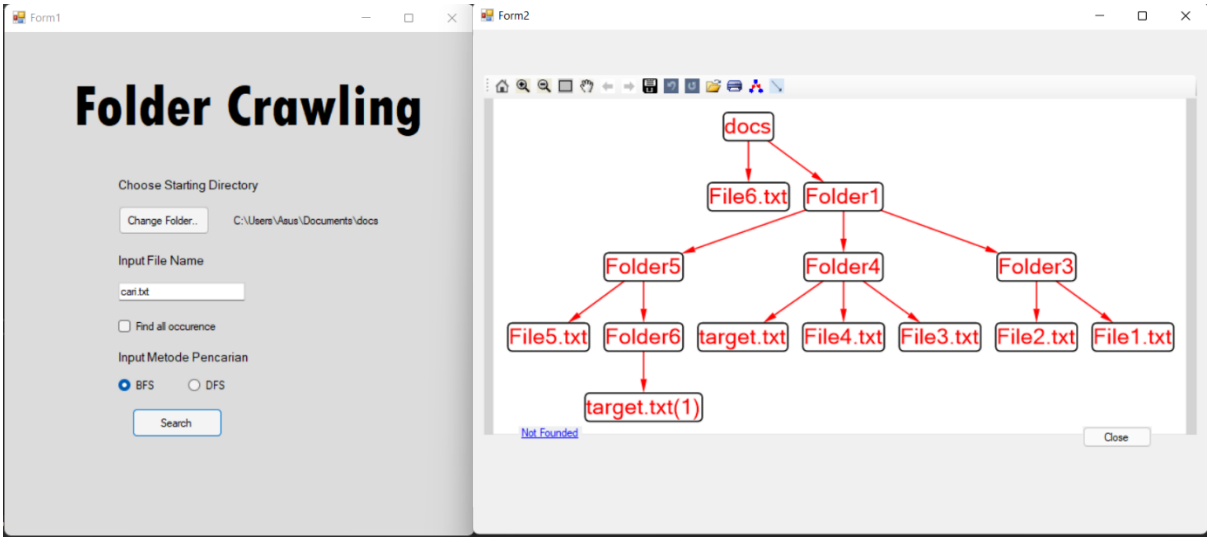
a. Tidak mencari semua kemunculan file.



b. Mencari semua kemunculan file.



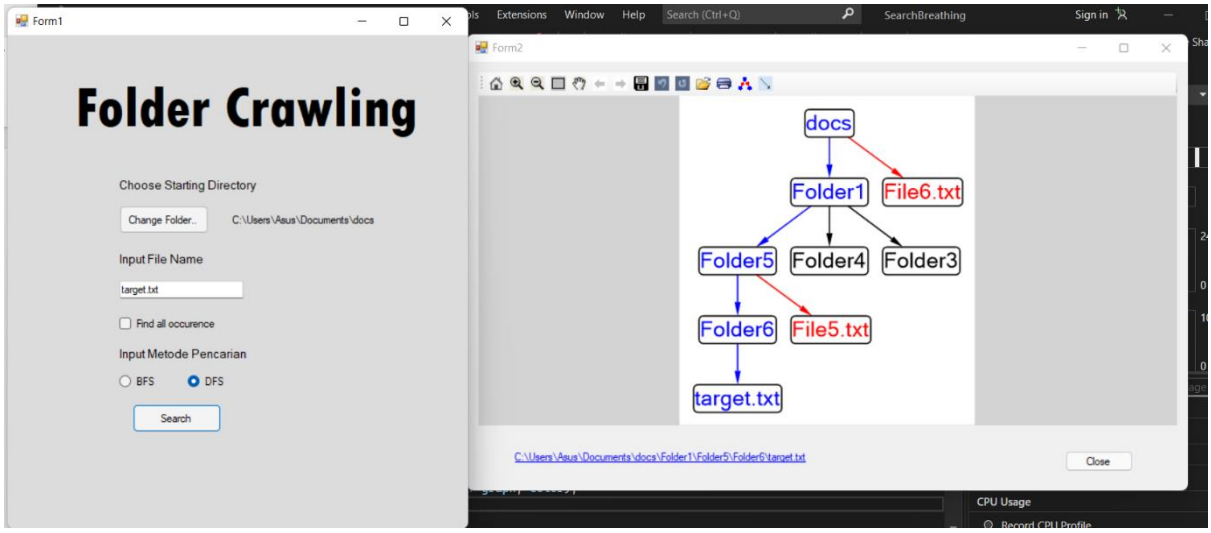
c. File tidak ditemukan.



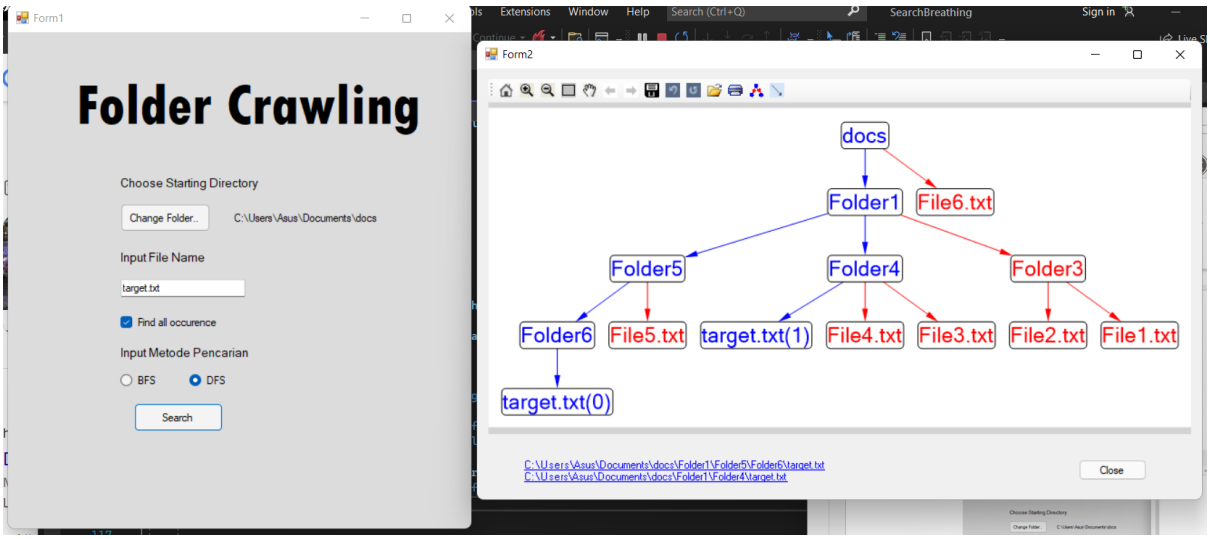
Algoritma DFS

Pengujian algoritma DFS menggunakan struktur folder yang sama dengan pada pengujian BFS agar bisa diperoleh perbandingan.

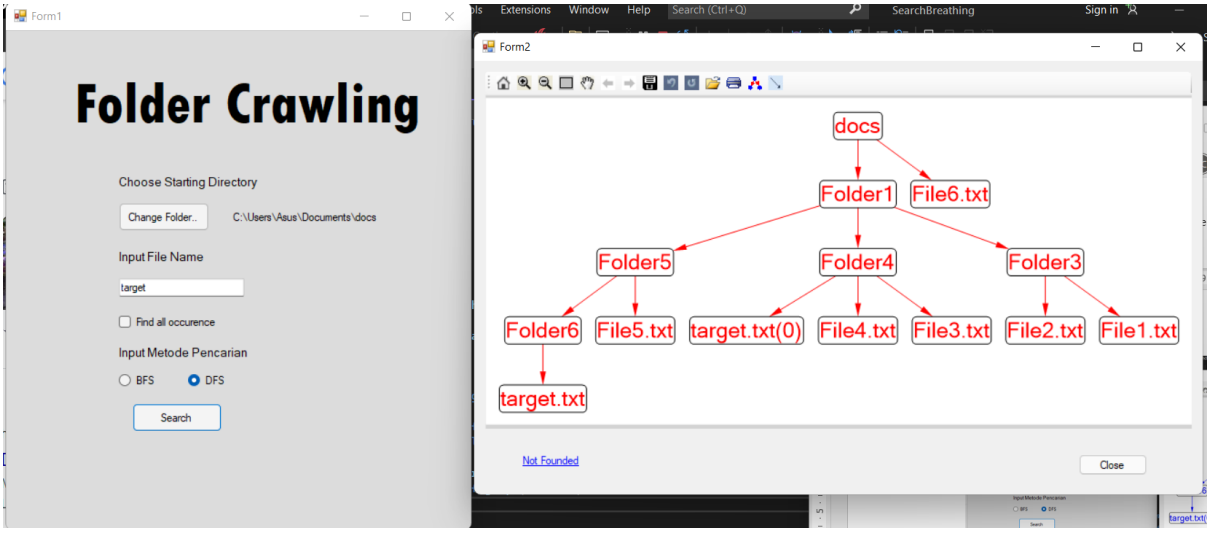
a. Mencari satu file.



b. Mencari semua kemunculan file



c. File tidak ditemukan.



5. Analisis Desain Solusi

Algoritma DFS dan BFS memilki kekurangan serta kelebihannya masing-masing. Kelebihan yang dimiliki DFS antara lain

- Pemakaian memori hanya sedikit, berbeda jauh dengan BFS yang harus menyimpan semua node yang pernah dibangkitkan
- Jika solusi yang dicari berada pada level yang dalam dan paling kiri, maka DFS akan dapat menemukannya dengan cepat

Sedangkan, kelemahan DFS adalah:

- Jika pohon yang dibangkitkan mempunyai level yang dalam (tak terhingga, maka tidak ada jaminan untuk menemukan solusi (*incomplete*)
- Jika terdapat lebih dari satu solusi yang sama tetapi berada pada level yang berbeda, maka pada DFS tidak ada jaminan untuk menemukan solusi yang paling baik.

BAB V

KESIMPULAN DAN SARAN

A. Kesimpulan

Pada tugas besar ini, kami dapat menyelesaikan sebuah program yang dapat melakukan pencarian sebuah file dengan menggunakan metode pencarian Breadth First Search dan Depth First Search. Kami juga berhasil membuat GUI sebagai platform untuk menampilkan hasil pencarian file tersebut.

B. Saran

Aplikasi yang kami buat masih memiliki potensi yang besar untuk dikembangkan. Beberapa diantaranya adalah pada penampilan. Penampilan pada aplikasi dapat dirancang dengan baik dan indah. Selain itu juga bisa dikembangkan dalam proses pencarian, ditampilkan bagaimana proses pencarian tersebut berjalan, sehingga dapat diketahui dengan baik perbedaan antara BFS dan DFS

DAFTAR PUSTAKA

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

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

https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms?view=windowsdesktop-6.0

LAMPIRAN

Tautan menuju repository kode :

https://github.com/ilhamwibowo/Tugas-Besar-2-STIMA

Tautan menuju demo tugas besar 2 :

https://youtu.be/98LZlEl4IqQ