

Laporan Tugas Besar 2

Pengaplikasian Algoritma BFS dan DFS dalam Implementasi *Folder Crawling*

Mata Kuliah IF2211 - Strategi Algoritma



Disusun Oleh :

Kelompok 4 (FileExplorer)

Gede Sumerta Yoga (13520021)

Averrous Saloom (13520100)

Jundan Haris (13520155)

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2021/2022

DAFTAR ISI

DAFTAR ISI.....	1
BAB I DESKRIPSI TUGAS	2
BAB II LANDASAN TEORI.....	5
Traversal Graph.....	5
Breadth First Search (BFS)	5
Depth First Search (DFS).....	5
C# Desktop Application Development	6
MSAGL.....	6
BAB III ANALISIS PEMECAHAN MASALAH	7
Langkah-Langkah.....	7
Proses Mapping Persoalan	7
Graf yang dibentuk	7
Penyelesaian dengan BFS	7
Penyelesaian dengan DFS.....	8
Contoh Ilustrasi Kasus Lain	9
BAB IV IMPLEMENTASI DAN PENGUJIAN.....	10
Implementasi	10
Modularitas Program	10
Struktur Data.....	10
Pseudocode	11
Tata Cara Penggunaan Program.....	13
Pengujian.....	15
Analisis dari Desain Solusi Algoritma BFS dan DFS.....	17
BAB V KESIMPULAN DAN SARAN	19
Kesimpulan.....	19
Saran.....	19
REFERENSI	20
LAMPIRAN.....	21

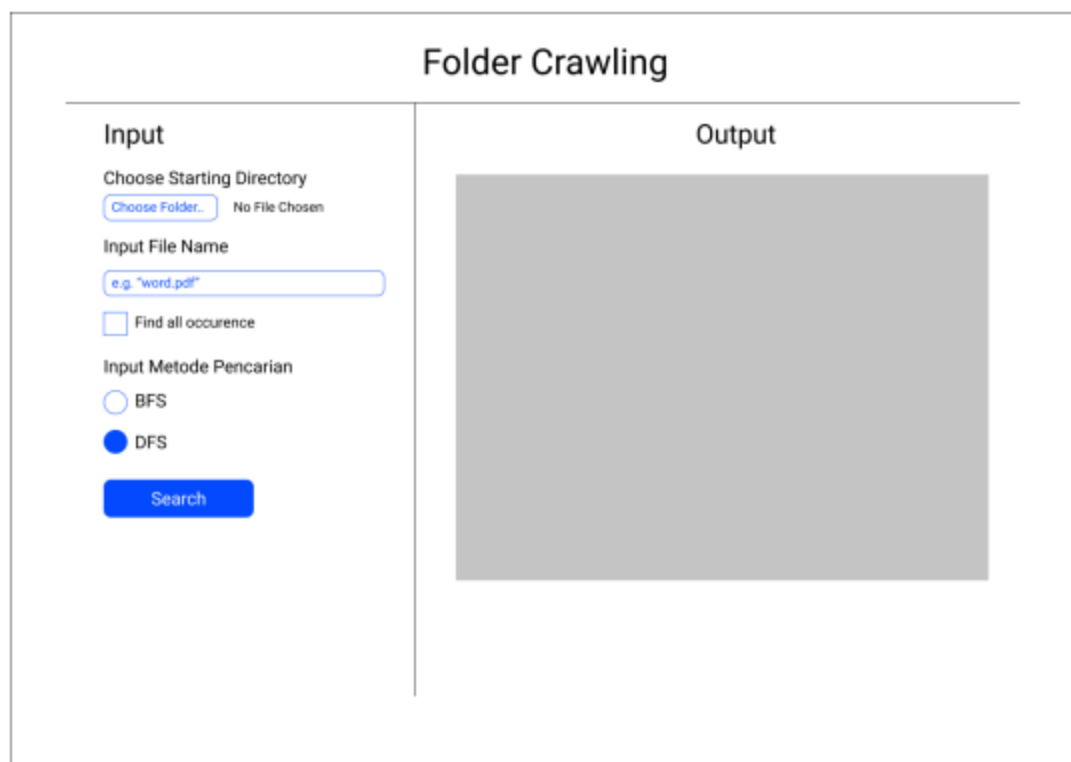
BAB I DESKRIPSI TUGAS

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.

Spesifikasi Program:

Aplikasi yang akan dibangun dibuat berbasis GUI. Berikut ini adalah contoh tampilan dari aplikasi GUI yang akan dibangun.



The screenshot shows a web-based GUI application titled "Folder Crawling". It is divided into two main sections: "Input" and "Output".

Input Section:

- Choose Starting Directory:** A button labeled "Choose Folder..." and a status text "No File Chosen".
- Input File Name:** A text input field with a placeholder "e.g. 'word.pdf'".
- Find all occurrence:** A checkbox that is currently unchecked.
- Input Metode Pencarian:** Two radio buttons: "BFS" (unselected) and "DFS" (selected).
- Search:** A blue button.

Output Section:

A large, empty gray rectangular area intended for displaying the search results, such as a directory tree or a list of paths.

Folder Crawling

Input

Choose Starting Directory
 C:/

Input File Name

☐ Find all occurrence

Input Metode Pencarian
☐ BFS
☒ DFS

Output

```

graph TD
    C["C:/"] --> F1["Folder1"]
    C --> F1_1["File1"]
    F1 --> F2["Folder2"]
    F1 --> F3["Folder3"]
    F1 --> F5["Folder5"]
    F2 --> F2_1["File1"]
    F2 --> F2_2["File2"]
    F3 --> F3_1["File1"]
    F3 --> F3_2["File2"]
    F5 --> F5_1["File1"]
    F5 --> F6["Folder6"]
    F6 --> F6_1["Kuis3_StrategiAlgoritma_13520000.pdf"]
          
```

Path File :
 • [C:/Folder1/Folder5/Folder6/Kuis3_StrategiAlgoritma_13520000.pdf](#)

Time spent: 20.02s

Spesifikasi GUI:

1. Program dapat menerima input folder dan query nama file.
2. Program dapat memilih untuk menampilkan satu hasil saja atau menemukan semua file yang memiliki nama file sama persis dengan input query
3. Program dapat memilih algoritma yang digunakan.
4. Program dapat menampilkan pohon hasil pencarian file tersebut dengan memberikan keterangan folder/file yang sudah diperiksa, folder/file yang sudah masuk antrian tapi belum diperiksa, dan rute folder serta file yang merupakan rute hasil pertemuan.
5. **(Bonus)** Program dapat menampilkan progress pembentukan pohon dengan menambahkan node/simpul sesuai dengan pemeriksaan folder/file yang sedang berlangsung.
6. Program dapat menampilkan hasil pencarian berupa rute/path (bisa lebih dari satu jika memilih menemukan semua file) serta durasi waktu algoritma.
7. GUI dapat dibuat sekreatif mungkin asalkan memuat 5(6 jika mengerjakan bonus) spesifikasi di atas.

Program yang dibuat harus memenuhi **spesifikasi wajib** sebagai berikut:

1) Buatlah program dalam bahasa **C#** untuk melakukan penelusuran Folder Crawling sehingga diperoleh hasil pencarian file yang diinginkan. Penelusuran harus memanfaatkan algoritma **BFS dan DFS**.

2) Awalnya program menerima sebuah input folder pada direktori yang ada dan nama file yang akan dicari oleh program.

3) Terdapat dua pilihan pencarian, yaitu:

a. Mencari 1 file saja Program akan memberhentikan pencarian ketika sudah menemukan file yang memiliki nama sama persis dengan input nama file.

b. Mencari semua kemunculan file pada folder root Program akan berhenti ketika sudah memeriksa semua file yang terdapat pada folder root dan program akan menampilkan daftar semua rute file yang memiliki nama sama persis dengan input nama file

4) Program kemudian dapat menampilkan **visualisasi pohon pencarian file** berdasarkan informasi direktori dari folder yang di-input. Pohon hasil pencarian file ini memiliki root adalah folder yang di-input dan setiap daunnya adalah file yang ada di folder root tersebut. Setiap folder/file direpresentasikan sebagai sebuah node atau simpul pada pohon. Cabang pada pohon menggambarkan folder/file yang terdapat di folder *parent*-nya.

Visualisasi pohon juga harus disertai dengan **keterangan** node yang sudah diperiksa, node yang sudah masuk antrian tapi belum diperiksa, dan node yang bagian dari rute hasil penemuan. Proses visualisasi ini boleh memanfaatkan pustaka atau kakas yang tersedia. Sebagai referensi, salah satu kakas yang tersedia untuk melakukan visualisasi adalah **MSAGL** (<https://github.com/microsoft/automatic-graph-layout>)

5) Program juga dapat menyediakan hyperlink pada setiap hasil rute yang ditemukan. Hyperlink ini akan membuka folder parent dari file yang ditemukan. Folder hasil hyperlink dapat dibuka dengan browser atau file explorer.

6) Mahasiswa **tidak diperkenankan** untuk melihat atau menyalin library lain yang mungkin tersedia bebas terkait dengan pemanfaatan BFS dan DFS. Tapi untuk algoritma lainnya seperti string matching dan akses directory, diperbolehkan menggunakan library jika ada.

BAB II

LANDASAN TEORI

Traversal Graph

Dalam computer sains, traversal graf adalah proses dari mengunjungi setiap simpul yang ada pada graph secara sistematis. Ada dua metode dari Algoritma traversal graf, yaitu:

- Pencarian melebar (Breadth First Search/BFS)
- Pencarian mendalam (Depth First Search/DFS)

Breadth First Search (BFS)

Pada metode BFS, semua simpul pada level n akan dikunjungi semua sebelum mengunjungi simpul-simpul pada level selanjutnya atau level $n+1$. Pencarian akan dimulai dari simpul awal kemudian akan mengecek simpul anaknya satu persatu begitu seterusnya.

Algoritma BFS:

- Kunjungi simpul awal, misal simpul v
- Kunjungi semua simpul yang bertetangga dengan simpul v terlebih dahulu
- Kunjungi simpul yang belum dikunjungi dan bertetangga dengan simpul-simpul yang tadi dikunjungi, demikian seterusnya.

Depth First Search (DFS)

Pada metode DFS, proses pencarian akan dilaksanakan pada semua anaknya sebelum mengecek ke simpul-simpul dalam level yang sama. Pencarian dimulai dari simpul akar ke simpul yang lebih tinggi. Proses ini diulangi hingga solusi ditemukan.

Algoritma DFS:

- Kunjungi simpul awal, misal simpul v
- Kunjungi simpul yang bertetangga dengan simpul v , misal simpul w
- Ulangi DFS mulai dari simpul w
- Ketika mencapai simpul yang semua simpul tetangganya telah dikunjungi maka backtrack ke simpul terakhir yang dikunjungi sebelumnya
- Pencarian berakhir jika tidak ada lagi simpul yang belum dikunjungi yang dapat dicapai dari simpul yang telah dikunjungi.

C# Desktop Application Development

Pada pengerjaan tugas besar ini, program ditulis dengan menggunakan bahasa C# (C Sharp) dengan menggunakan aplikasi visual studio. Pada aplikasi ini terdapat Windows Form yang dapat digunakan untuk membuat GUI. Windows Form merupakan graphical class library yang tergabung dalam Microsoft .NET, .NET Framework yang dapat memudahkan developer untuk membuat aplikasi desktop. Terdapat banyak fitur yang disediakan seperti label, button, radio button, checkbox, picturebox, dan lain-lain yang dapat digunakan untuk membuat UI/UX dari program yang dibuat.

MSAGL

Pada pengerjaan tugas besar ini, visualisasi graph pencarian baik DFS maupun BFS menggunakan bantuan MSAGL. MSAGL adalah sebuah tool .Net untuk membuat dan menampilkan graph. MSAGL ini dikembangkan di Microsoft oleh Lev Nachmanson, Sergey Pupyrev, Tim Dwyer and Ted Hart. Dalam package ini terdapat:

- Layout engine (Microsoft.MSAGL.dll)
- Drawing module (Microsoft.MSAGL.Drawing.dll)
- Viewer control (Microsoft.MSAGL.GraphViewerGDIGraph.dll)

BAB III

ANALISIS PEMECAHAN MASALAH

Langkah-Langkah

Permasalahan yang diberikan yaitu pencarian sebuah file/folder pada penyimpanan komputer dengan cara menelusuri folder-folder yang dapat diakses dari direktori awal yang sudah dipilih. Permasalahan ini dapat dimodelkan dengan menggunakan graf atau pohon. Secara umum, langkah-langkah penyelesaiannya yaitu :

1. Pilih direktori awal
2. Masukkan nama file/folder yang ingin dicari
3. Pilih jenis pencarian (satu solusi atau semua solusi)
4. Pilih metode pencarian (BFS atau DFS)
5. Pencarian dilakukan
6. Hasil pencarian ditampilkan

Pada saat pencarian, dilakukan permodelan menggunakan graf atau lebih tepatnya pohon. Direktori awal akan menjadi akar, direktori dan file yang dapat diakses dari direktori awal akan menjadi simpul-simpulnya, dan direktori atau file yang dicari akan menjadi daunnya. Pemecahan masalah dibagi menjadi 2, yaitu dengan menggunakan *Breadth-First Search* dan *Depth-First Search*. Ketika persoalan sudah dimodelkan menjadi graf/pohon, maka algoritma BFS dan DFS dapat digunakan untuk memecahkan persoalan tersebut.

Proses Mapping Persoalan

Graf yang dibentuk

Sistem direktori dapat dimodelkan menjadi graf. Dalam kondisi tidak ada file shortcut, kita dapat memodelkan menjadi pohon. Berikut uraian elemen graf:

- Simpul : folder dan file
- Sisi : hubungan antara folder dan hubungan antar file dan folder

Graf sistem direktori tidak dapat dibangkitkan diawal. Oleh karena itu, penyelesaian algoritma pencarian menggunakan pendekatan graf dinamis.

Penyelesaian dengan BFS

Untuk mencapai permintaan spesifikasi, penguraian elemen dari algoritma BFS dengan graf dinamis perlu dibagi dua, yakni saat satu solusi cukup serta saat seluruh solusi harus ditemukan.

Elemen	Satu solusi cukup	Seluruh solusi ditemukan
Operator	Peroleh seluruh anak dari simpul	Peroleh seluruh anak dari simpul
Akar (initial state)	Direktori awal	Direktori awal
Simpul (problem state)	Direktori atau file yang dapat diakses setelah direktori awal	Direktori atau file yang dapat diakses setelah direktori awal
Daun (goal state)	Direktori atau file yang bernama sama dengan tujuan	Direktori atau file yang bernama sama dengan tujuan
Ruang status	Himpunan yang berisi simpul yang diakses secara melebar (<i>Breadth</i>) yang berada sebelum dan pada ditemukannya solusi pertama	Himpunan yang berisi seluruh simpul yang berada di dalam direktori awal
Ruang solusi	Himpunan yang berisi simpul solusi pertama yang ditemukan	Himpunan yang berisi seluruh solusi yang ditemukan

Penyelesaian dengan DFS

Untuk menyelesaikan persoalan yang diminta, algoritma DFS yang dibuat diuraikan dalam beberapa elemen. Penguraian elemen graf dinamis yang dibuat dibagi menjadi dua, yaitu ketika hanya mencari solusi dan ketika mencari semua solusi yang ada.

Elemen	Satu solusi cukup	Seluruh solusi ditemukan
Operator	Peroleh seluruh anak dari simpul	Peroleh seluruh anak dari simpul
Akar (initial state)	Direktori awal	Direktori awal
Simpul (problem state)	Direktori atau file yang dapat diakses setelah direktori awal	Direktori atau file yang dapat diakses setelah direktori awal
Daun (goal state)	Direktori atau file yang bernama sama dengan tujuan	Direktori atau file yang bernama sama dengan tujuan
Ruang status	Himpunan yang berisi simpul yang diakses secara mendalam (<i>Depth</i>)	Himpunan yang berisi seluruh simpul yang berada di dalam direktori awal

	yang berada sebelum dan pada ditemukannya solusi pertama	
Ruang solusi	Himpunan yang berisi simpul solusi pertama yang ditemukan	Himpunan yang berisi seluruh solusi yang ditemukan

Contoh Ilustrasi Kasus Lain

Kasus lain yang mungkin terjadi pada permasalahan ini dan usaha kami untuk menanganinya:

Kasus	Penanganan
Pengguna memasukkan nama folder sebagai tujuan dan bukan nama file	Program dapat mencari kedua jenis direktori
Pengguna memasukkan nama file dan bukan folder sebagai direktori awal	Program membatasi direktori awal hanya nama folder saja
Direktori awal memiliki direktori turunan yang sangat banyak sehingga program menjadi lama	Viewer untuk graf dikosongkan ketika proses pencarian berlangsung
Direktori yang menjadi destinasi tidak ditemukan	Diberikan pesan bahwa direktori tidak ditemukan

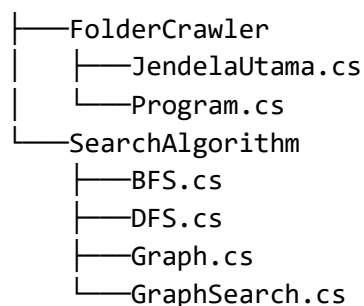
BAB IV IMPLEMENTASI DAN PENGUJIAN

Implementasi

Implementasi dilakukan menggunakan pendekatan *object oriented programming*. Hal ini dilakukan dengan membuat sebuah kelas abstrak `GraphSearch` dan menurunkannya pada `DFS` dan `BFS`. Visualisasi menggunakan kelas `Graph` dengan

Untuk melakukan sebuah pencarian, perlu dilakukan instantiasi kelas `BFS` atau `DFS` bergantung pada metode yang dipilih. Setelah melakukan instantiasi, dilakukan pemanggilan metode `crawl(Mode, bool)`, metode ini adalah metode yang membangun graf (atribut kelas), mencari nilai bergantung mode, yakni antara *First* atau *All*. Mode *First* akan mengisi graf sampai simpul solusi pertama, atau sampai seluruh turunan dari akar tercapai tergantung pada nilai boolean `ShowAll`, serta mencatat waktu hanya sampai program menemukan simpul solusi pertama. Mode *All* akan mengisi graf sampai seluruh turunan dari akar tercapai, serta mencatat waktu sampai proses pencarian pada seluruh ruang status selesai.

Modularitas Program

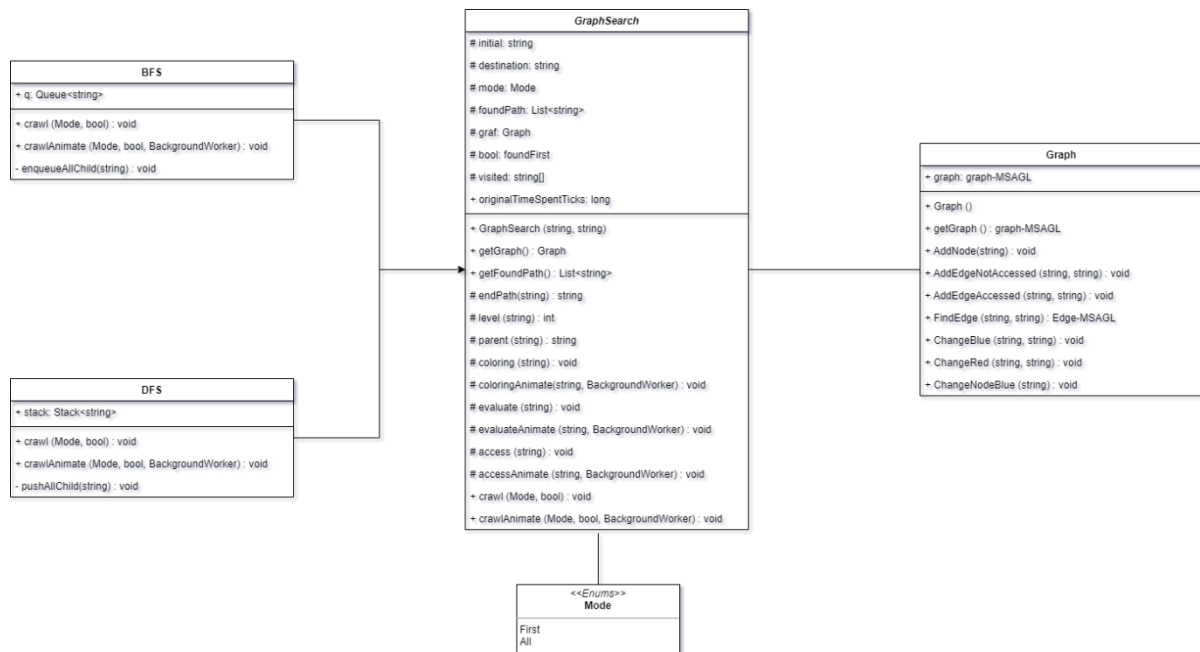


Gambar 1 Modularitas Program

Modularitas program menggunakan pendekatan dua buah *package*, yakni `FolderCrawler` dan `SearchAlgorithm`. `FolderCrawler` digunakan sebagai program utama yang mengandung antarmuka. Sementara `SearchAlgorithm` mengandung algoritma pencarian yang dilakukan serta kelas `Graph` yang mengenkapsulasi implementasi dari pustaka visualisasi graf `MSAGL`.

Struktur Data

Secara keseluruhan program, struktur data yang digunakan menggunakan struktur data berbasis objek. Diagram kelas dipilih untuk menjelaskan struktur data, karena sudah merepresentasikan seluruh deskripsi yang dibutuhkan,



Gambar 2 Diagram Kelas

Pseudocode

Fungsi yang menjadi pokok dari program adalah prosedur crawl yang diimplementasikan pada kelas BFS serta DFS.

BFS.cs/crawl

```

procedure crawl(Mode m, boolean showAll)
    bool foundFirst = false
    Graph graf
    string[] visited
    string[] foundPath
    Queue q

    INISIALISASI()
    MULAI_STOPWATCH()
    q.Enqueue(this.initial)
    string path;
    while (((showAll and q.Count > 0) or (!showAll and q.Count > 0 and
        !foundFirst))
    do
        path = q.Dequeue()
        access(path)
        ENQUEUE_ALL_CHILD(path)

        if (m = Mode.First and foundFirst = true and
            this.originalTimeSpentTicks = 0)
        then
            HENTIKAN_STOPWATCH()
  
```

```

if (foundFirst = false) then
    HENTIKAN_STOPWATCH()

PEMBENTUKAN_GRAPH_TERSISA_PADA_QUEUE()

```

DFS.cs/crawl

```

procedure crawl(Mode m, boolean showAll)
    bool foundFirst = false
    Graph graf
    string[] visited
    string[] foundPath
    Stack q

    INISIALISASI()
    MULAI_STOPWATCH()
    stack.Push(this.initial)
    string path;
    while (((showAll and stack.Count > 0) or (!showAll and stack.Count > 0
and !foundFirst))
        do
            path = stack.Pop()
            access(path)
            PUSH_ALL_CHILD(path)

            if (m = Mode.First and foundFirst = true and
                this.originalTimeSpentTicks = 0)
            then
                HENTIKAN_STOPWATCH()

            if (foundFirst = false) then
                HENTIKAN_STOPWATCH()

    PEMBENTUKAN_GRAPH_TERSISA_PADA_QUEUE()

```

Namun, pada program antarmuka pada *JendelaUtama*, prosedur yang dipanggil untuk memproses nilai adalah prosedur *crawlAnimate* yang juga diimplementasikan pada kelas *BFS* dan *DFS*. Prosedur tersebut adalah prosedur yang memungkinkan animasi dijalankan pada antar muka. Prosedur ini mirip dengan prosedur *crawl* dengan modifikasi, karena animasi yang ditunjukkan adalah animasi proses *crawling* yang terjadi.

BFS.cs/crawlAnimate dan DFS.cs/crawlAnimate

```

procedure crawlAnimate(Mode m, boolean showAll, Worker w)
    crawl(m, showAll)
    CRAWL_DENGAN_MODIFIKASI()

```

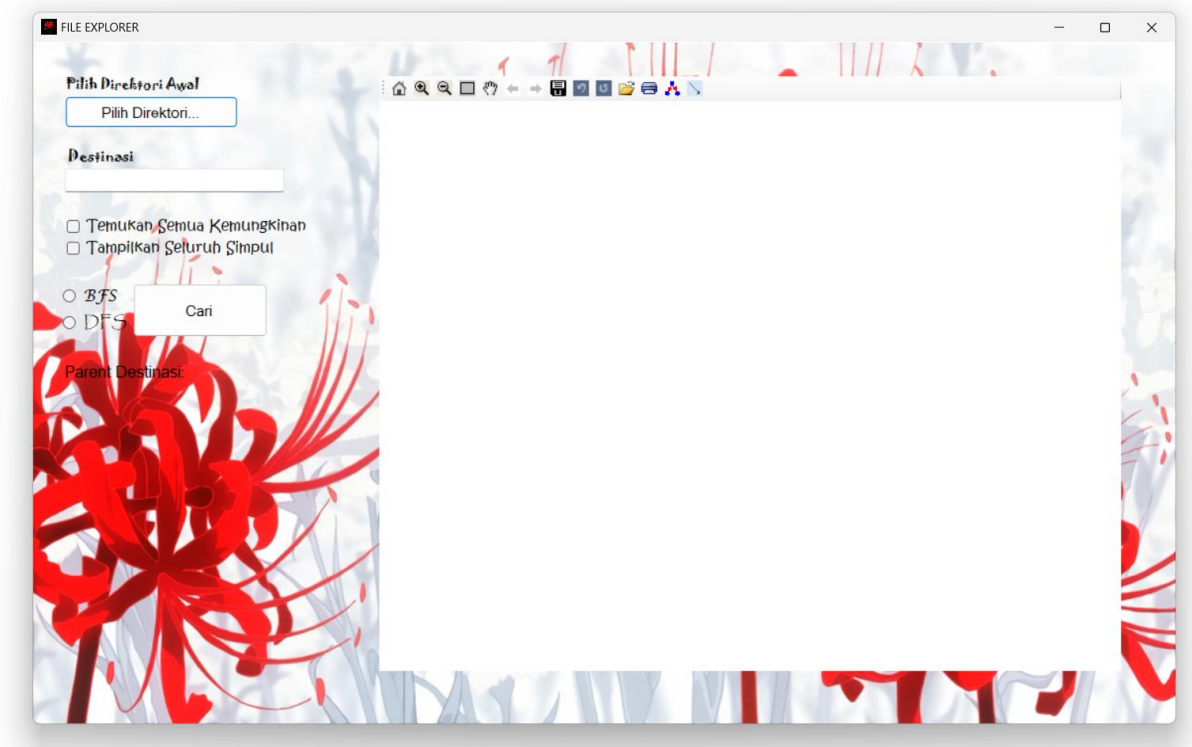
Pemanggilan program antarmuka yang ditampilkan, prosedur yang dipanggil bersifat *event based*. Event ini muncul jika tombol “Cari” pada antarmuka ditekan. Penanganan event tersebut akan memanggil prosedur work yang akan memanggil worker yang akan mengerjakan BFS atau DFS. Pada worker inilah prosedur crawlAnimate dipanggil.

JendelaUtama.cs/<BFSWorker/DFSWorker>__DoWork

```
procedure <BFSWorker/DFSWorker>__DoWork()  
  INSTANSIASI_OBJEK_ALGORITMA(a)  
  if (ALL_OCCURENCE = true)  
    a.crawlAnimate(Mode.All, ShowAllVertices.Checked, worker);  
  else if (FIRST_OCCURENCE = true)  
    a.crawlAnimate(Mode.First, ShowAllVertices.Checked, worker);
```

Tata Cara Penggunaan Program

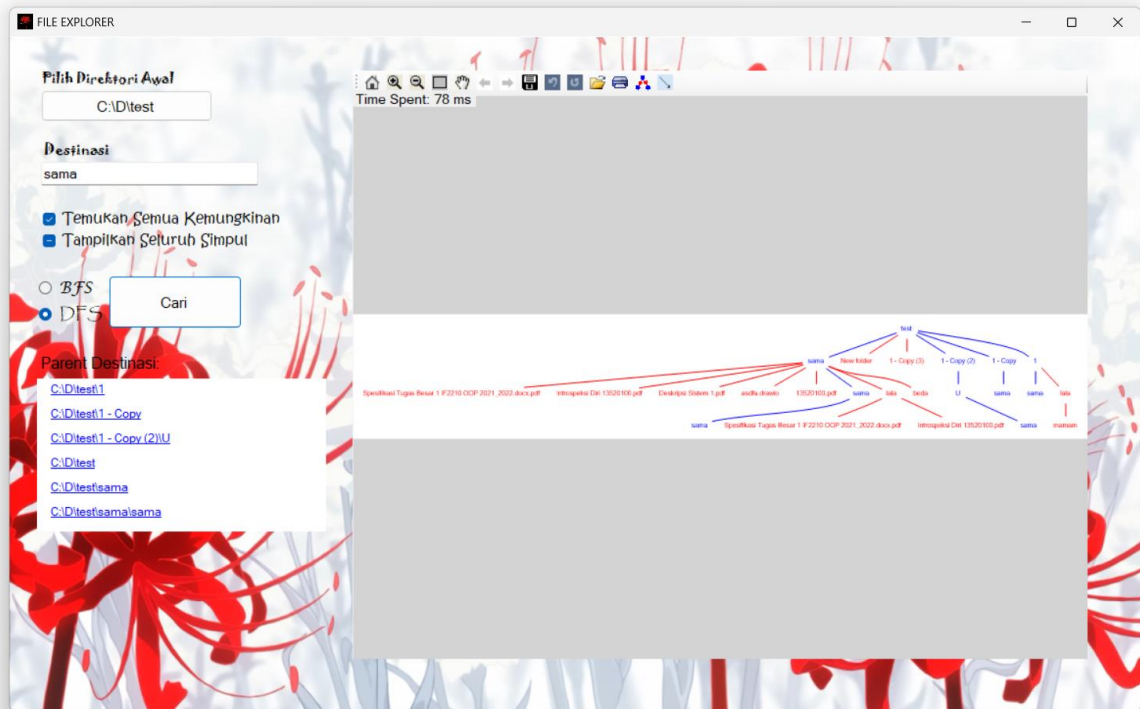
Untuk menggunakan program, diperlukan file binary dari program yang juga disertakan dependensi yang berada di satu folder dengan file binary. Jalankan file binary tersebut, lalu akan muncul tampilan seperti ini



Gambar 3 Tampilan Awal Program

Berikut tahap serta penjelasan fitur yang dapat dilakukan untuk mencari suatu destinasi file atau folder:

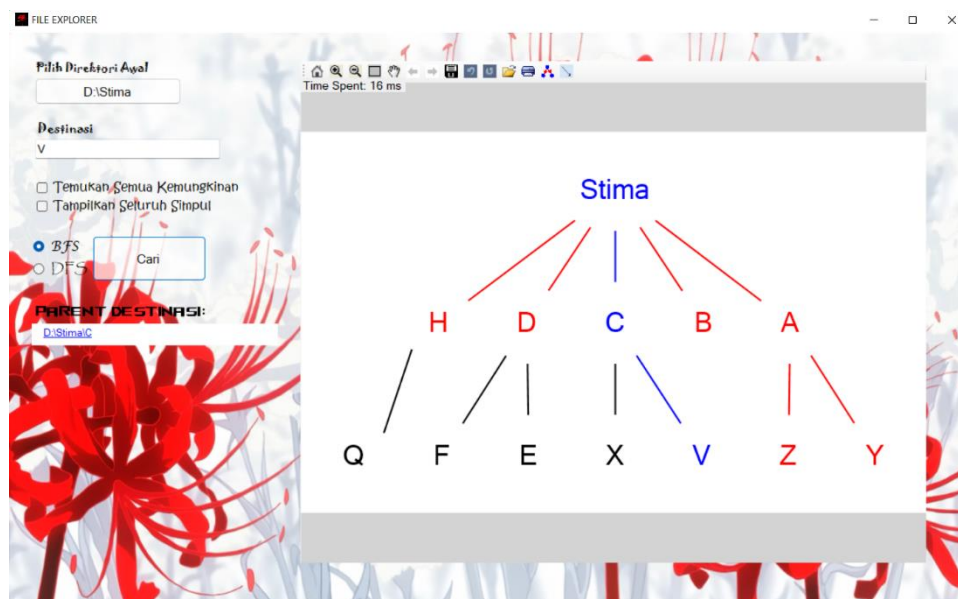
1. Klik tombol “Pilih Direktori” untuk memilih direktori awal dari proses pencarian
2. Isi destinasi yang ingin dicari, untuk mencari destinasi bertipe file, pastikan ekstensi file dicantumkan
3. Jika ingin mencari semua solusi, centangkan kotak centang “Tampilkan Semua Kemungkinan”
4. Jika ingin mencari sampai solusi pertama saja, biarkan kotak centang tersebut
5. Jika mencari semua solusi, kotak centang “Tampilkan Seluruh Simpul” akan secara otomatis terkunci
6. Jika mencari sampai solusi pertama saja, centang kotak centang “Tampilkan Seluruh Simpul” jika ingin menampilkan seluruh simpul. Namun, waktu tetap dihitung hanya sampai solusi pertama ditemukan
7. Jika kotak centang “Tampilkan Seluruh Simpul” tidak tercentang, maka yang ditampilkan hanya proses pencarian sampai solusi pertama saja.
8. Klik tombol “Cari”
9. Hasil akan muncul ke Graph Viewer di kanan dengan animasi
10. Waktu akan dimunculkan di bagian “Time Spent”, sesuai dengan mode yang digunakan.
11. List tautan dari orang tua seluruh destinasi yang ditemukan akan muncul di list Parent Destinasi



Gambar 4 Tampilan Akhir Program

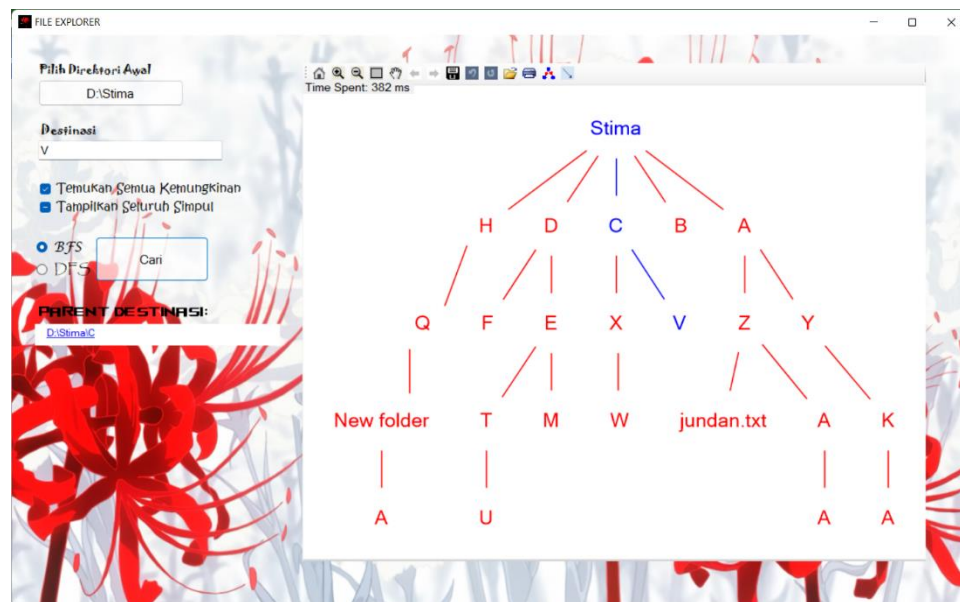
Pengujian

- Pengujian BFS mode First



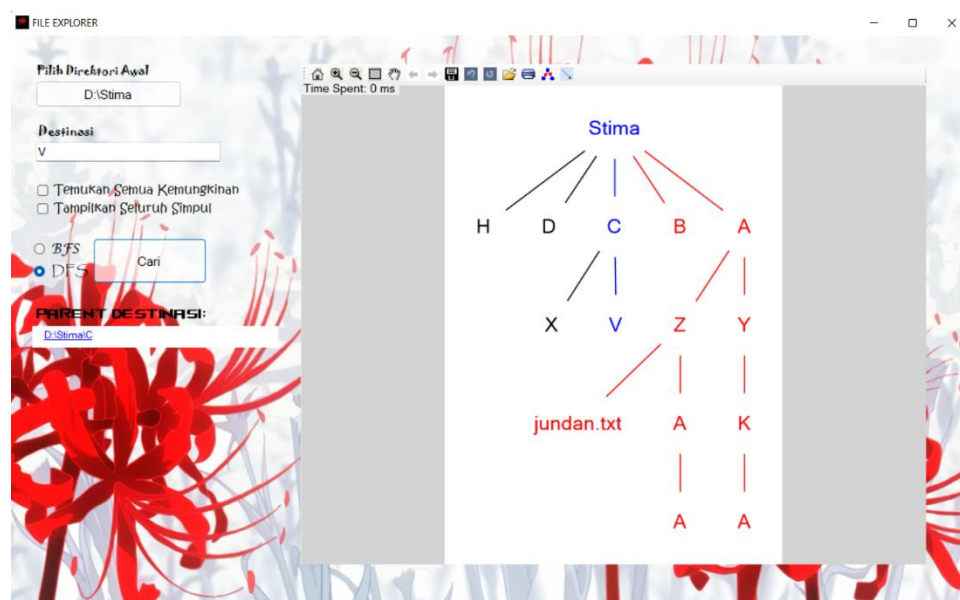
Gambar 5 Pengujian BFS Mode First

- Pengujian BFS mode All



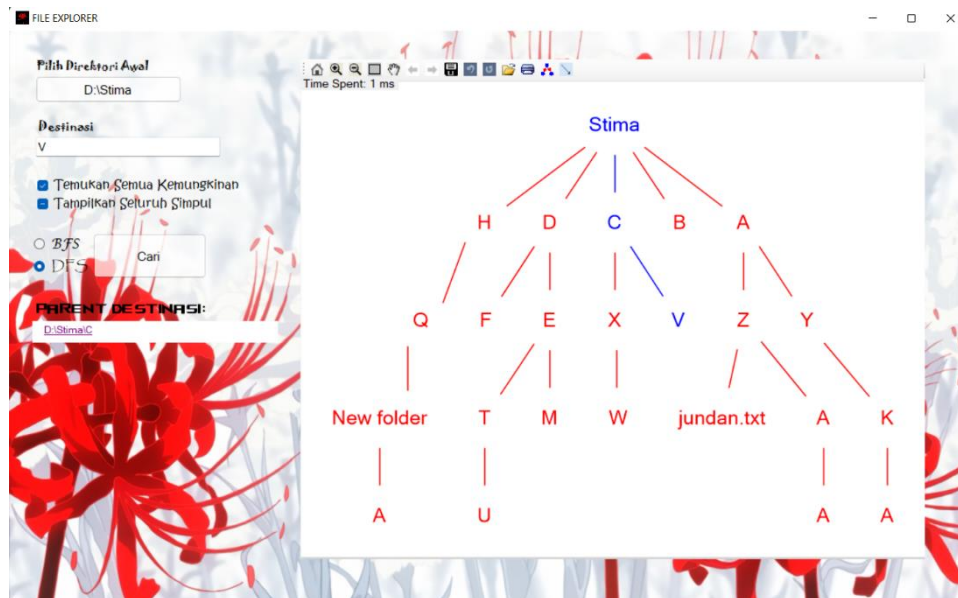
Gambar 6 Pengujian BFS Mode All

- Pengujian DFS mode First



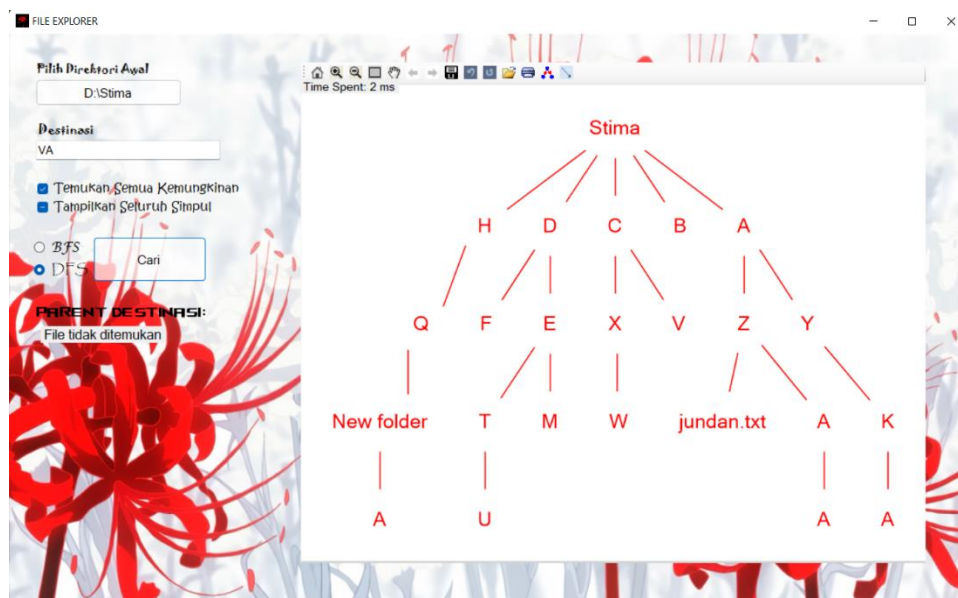
Gambar 7 Pengujian DFS Mode First

- Pengujian DFS mode All



Gambar 8 Pengujian DFS Mode All

- Pengujian file tidak ditemukan



Gambar 9 Pengujian file tidak ditemukan

Analisis dari Desain Solusi Algoritma BFS dan DFS

Analisis kompleksitas dari kedua algoritma berbeda bergantung mode.

	Mode solusi pertama	Mode seluruh solusi
BFS	Rerata: $T(n/2)$	Rerata: $T(n)$

	Big-O: $O(n)$ Berperforma tinggi jika destinasi berada pada level tidak dalam	Big-O: $O(n)$
DFS	Rerata: $T(n/2)$ Big-O: $O(n)$ Berperforma tinggi jika destinasi berada pada level dalam dan berada pada direktori yang berada pada urutan atas alfabet	Rerata: $T(n/2)$ Big-O: $O(n)$

Desain solusi algoritma BFS dan DFS memiliki pendekatan yang berbeda mengenai pengaksesan mendalam atau melebar terlebih dahulu. Perbedaan pendekatan ini menghasilkan performa yang berbeda antar kedua algoritma pada kasus-kasus khusus. Kami meringkas kasus-kasus tersebut pada tabel berikut:

Kasus	Lebih Baik		Alasan
	BFS	DFS	
Level kedalaman yang tidak terlalu dalam	√		Karena BFS akan menelusuri setiap node atau simpul dari setiap level terlebih dahulu baru berpindah ke level selanjutnya.
Level kedalaman yang sangat dalam	√		Karena DFS akan menelusuri setiap node hingga kedalaman paling tinggi. Ketika kedalamannya sangat tinggi atau bahkan tak terhingga, bisa saja solusi tidak ditemukan.

BAB V

KESIMPULAN DAN SARAN

Kesimpulan

Program implementasi *Folder Crawling* dengan mengaplikasikan algoritma BFS dan DFS dapat berjalan dengan baik. Penggunaan algoritma BFS dan DFS sangat cocok untuk memecahkan persoalan yang diberikan. Masing-masing memiliki kelebihan dan kekurangannya sendiri. Program yang dibuat berbentuk desktop app. Program dapat mencari file ataupun folder dengan memasukkan kata kunci dan direktori awal. Spesifikasi bonus, penampilan pembentukan pohon selama pencarian, juga dapat diimplementasikan dengan baik. Laporan yang dibuat pun sudah selesai dengan baik. Namun, tampilan dari program masih bisa dibuat lebih menarik.

Saran

Apabila di kemudian hari program ini dibuat ulang, tampilannya bisa dibuat lebih bagus lagi dan mungkin saja bisa menambahkan fitur-fitur lain. Pemberian tugas besar yang bersamaan dengan mata kuliah lain sedikit membuat kami kesulitan untuk mengatur waktu. Penambahan waktu pengerjaan akan membantu kami untuk *explore* lebih dalam mengenai materi ini.

REFERENSI

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/stima21-22.htm>

<https://www.geeksforgeeks.org/iterative-depth-first-traversal/>

<https://stackoverflow.com/questions/5278580/non-recursive-depth-first-search-algorithm>

<https://docs.microsoft.com/en-us/dotnet/api/system.io.directory?view=net-6.0>

LAMPIRAN

Link Video Youtube: <https://youtu.be/yFt4oDx5poA>

Link Repositori Github: https://github.com/sumertayoga/Tubes2_13520021.git