LAPORAN TUGAS BESAR STRATEGI ALGORITMA

"File Searcher Menggunakan Metode DFS dan BFS"

IF2211 STRATEGI ALGORITMA

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

Kelompok : cariGunung

Disusun oleh :

Ken Kalang (13520010)

Gibran Darmawan (13520061)

Steven Siaahan (13520145)

PROGRAM STUDI TEKNIK INFORMATIKA

SETKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2021

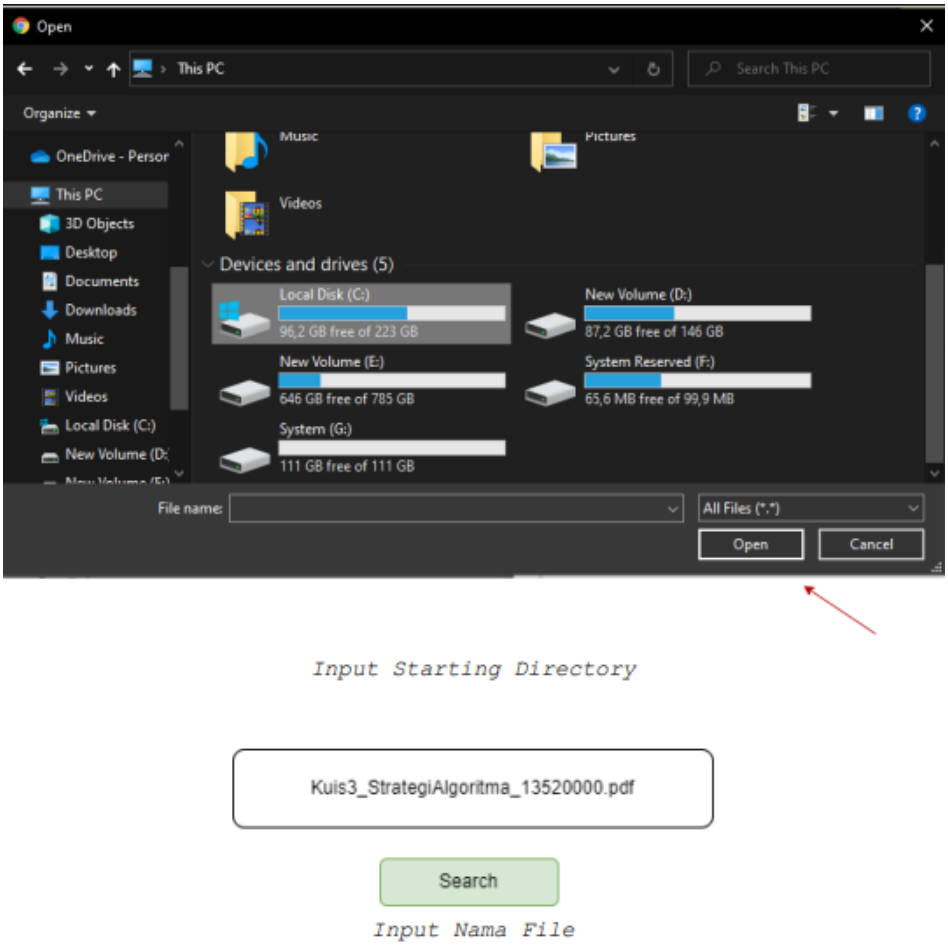
BAB I

DESKRIPSI TUGAS

Dalam tugas besar ini, mahasiswa 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. Mahasiswa juga diminta untuk memvisualisasikan hasil dari pencarian folder tersebut dalam bentuk pohon. Selain pohon, mahasiswa 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.

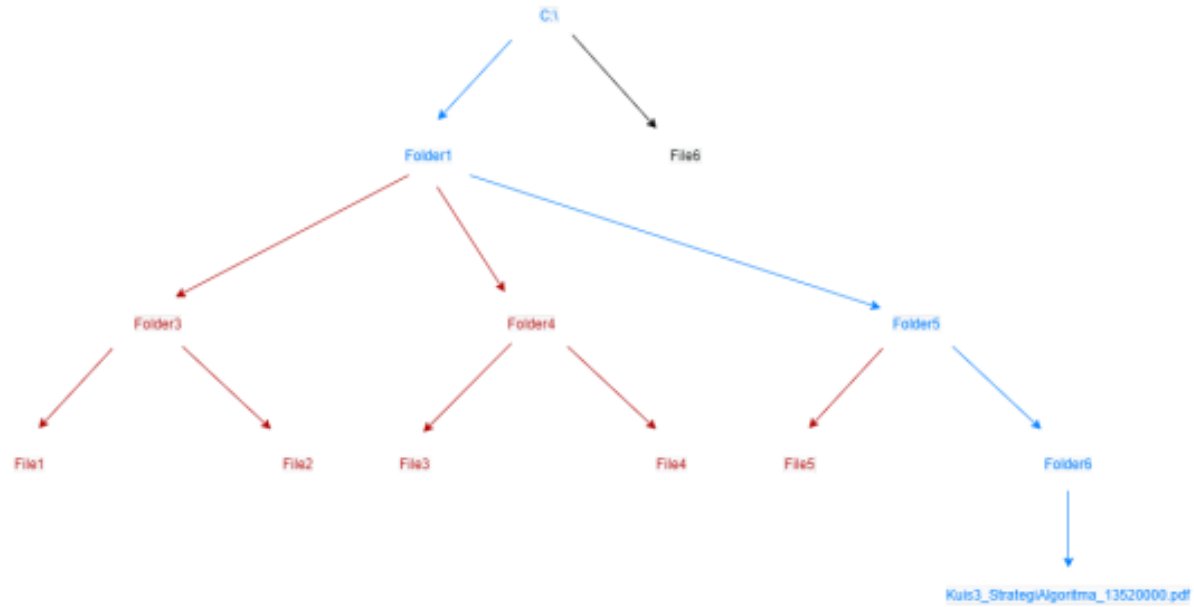
Contoh input dan output program:

Input program:



Gambar 1. Contoh input

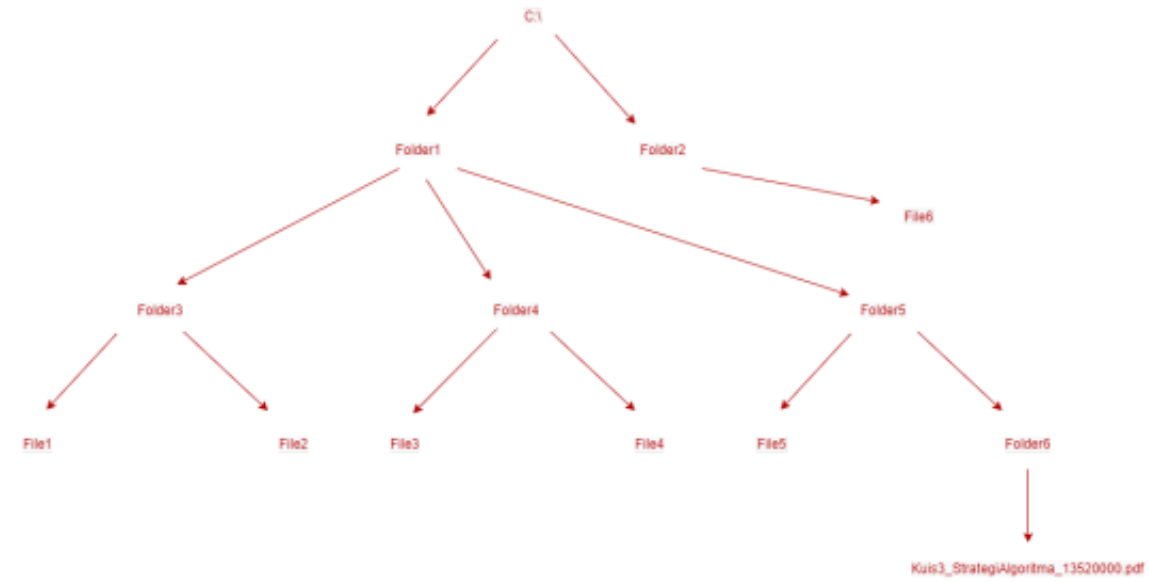
Output program:



List Path:
1. C:\Folder1\Folder5\Folder6\Kuis3_StrategiAlgoritma_13520000.pdf

Gambar 2. Contoh output

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. Mahasiswa bebas menentukan warnanya asalkan dibedakan antara ketiga hal tersebut.

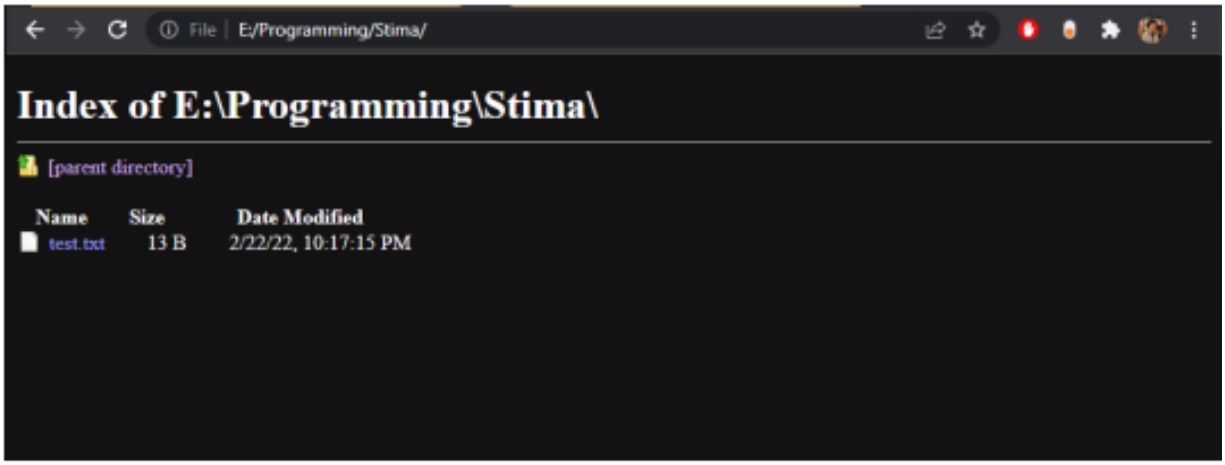


List Path:
Tidak ada path yang sesuai.

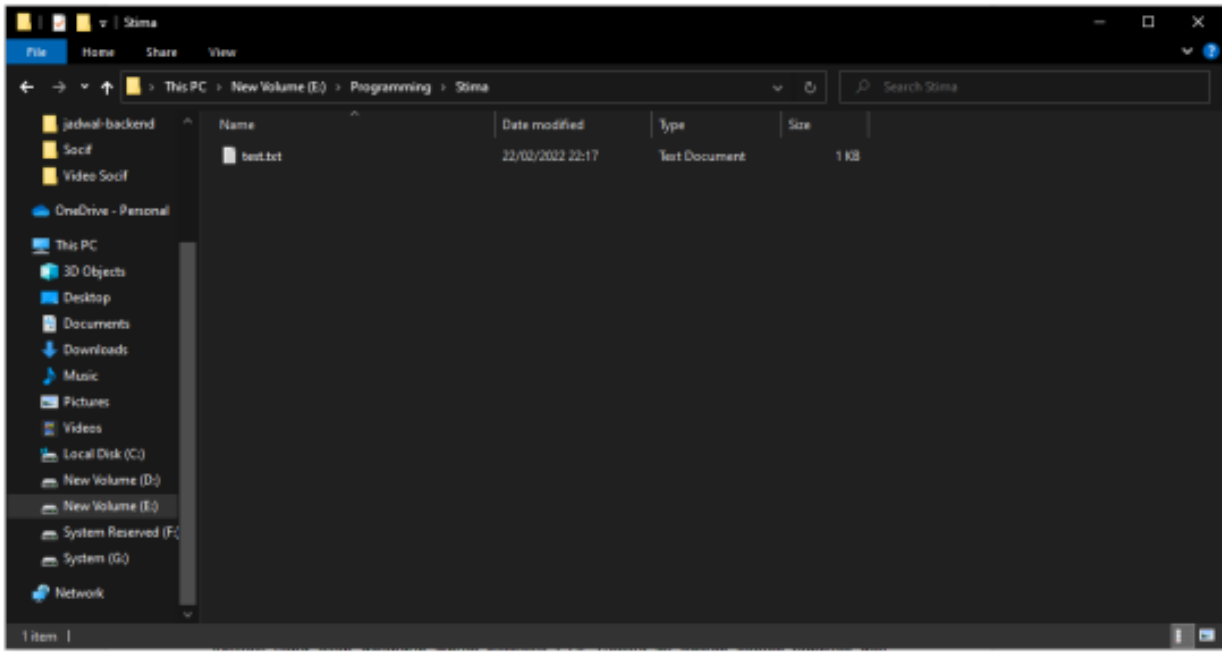
Gambar 3. Contoh output file tidak ada

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 :



Contoh Hyperlink Dibuka Melalui Browser



Contoh Hyperlink Dibuka Melalui Browser

Gambar 4. Contoh impelmentasi hyperlink

Spesifikasi program:

a. 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.

b. Spesifikasi wajib
   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 mencari semua file yang muncul pada root atau file yang pertama kali sesuai dengan query pencarian.
   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**.
   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

1. Algoritma DFS

Dengan algoritma DFS pencarian file dilakukan dengan *tracking* satu persatu-jalur yang ada hingga ditemukan suatu jalur yang memiliki solusi. Persoalan yang dipecahkan dengan metode ini adalah jalur yang ditempuh akan lebih cepat dalam kondisi solusi berada pada jalur pertama.

2. Algoritma BFS

Dengan algoritma BFS pencarian file dilakukan dengan *tracking* semua jalur yang ada secara bersamaan hingga ditemukan suatu jalur yang memiliki solusi. Persoalan yang dipecahkan dengan metode ini adalah waktu yang dibutuhkan lebih sedikit karena jika path banyak maka akan lebih cepat jika semua jalur dicek secara bersamaan.

3. Bahasa C#

C# merupakan sebuah bahasa pemrograman yang berorientasi objek yang dikembangkan oleh Microsoft sebagai bagian dari inisiatif kerangka .NET Framework. Bahasa pemrograman ini dibuat berbasiskan bahasa C++ yang telah dipengaruhi oleh aspek-aspek ataupun fitur bahasa yang terdapat pada bahasa-bahasa pemrograman lainnya seperti Java, Delphi, Visual Basic, dan lain-lain) dengan beberapa penyederhanaan. Menurut standar *ECMA-334 C# Language Specification*, nama C# terdiri atas sebuah huruf Latin **C** yang diikuti oleh tanda pagar yang menandakan angka **#** . Tanda pagar # yang digunakan memang bukan tanda kres dalam seni musik, dan tanda pagar # tersebut digunakan karena karakter kres dalam seni musik tidak terdapat di dalam papan tombol standar.

Salah satu fungsi C# adalah C# desktop application development. Developer dapat membuat sebuah aplikasi windows dengan mengutilisasikan fungsi ini. Dengan menggunakan .NET framework yang disediakan oleh Microsoft, developer dapat membuat aplikasi dengan mudah.

BAB III

ANALISIS PEMECAHAN MASALAH

1. Mapping Persoalan BFS dan DFS

Pemecahan masalah dengan menggunakan metode BFS :

Gunakan queue sebagai antrian dari folder dan file yang akan dicek. Enqueue folder dan file (sebagai children dari path awal) pada queue untuk proses pengecekan selanjutnya. Pada proses pengecekan lakukan pengecekan apakah children merupakan folder atau bukan. Jika children merupakan folder maka enqueue setiap children yang dimiliki pada queue utama. Jika children merupakan file maka cek apakah file sama dengan file yang dicari. Jika tidak maka ulangi proses pengecekan awal sampai ditemukan file yang dicari. Jika sudah ditemukan maka proses akan berhenti mencari.

2. Contoh Ilustrasi Kasus Lain

Pemecehan masalah dengan menggunakan metode DFS :

Gunakan queue sebagai penyimpanan folder tracking. Pada path utama lakukan pengecekan setiap file childrennya. Jika children tidak sama dengan file yang dicari maka cek children file selanjutnya sampai ditemukan. Jika file pada parent utama sudah dicek, lakukan rekursif proses DFS pada setiap folder children pada parent utama hingga ditemukan solusi.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

Implementasi dengan metode BFS :

```
public void BFS(string path)
{
   //create the graph content
```

```
FileAttributes attr = File.GetAttributes(path);

//detect whether its a directory or file
if ((attr & FileAttributes.Directory) == FileAttributes.Directory)
{
   string[] children = Directory.GetDirectories(path);
   foreach (var child in children)
   {
      visited.Enqueue(new folderfile(path, child));
      penyimpanan.Enqueue(new folderfile(path, child));
   }
   string[] childrenfile = Directory.GetFiles(path);
   foreach (var child in childrenfile)
   {
      visited.Enqueue(new folderfile(path, child));
      penyimpanan.Enqueue(new folderfile(path, child));
   }
   Console.WriteLine(path);
}

else
{
   if (Path.GetFileName(path) == namafile)
   {
      if (kemungkinan == "N" && found == false)
      {
         Console.WriteLine(Path.GetFullPath(path); // File ketemu
         found = true;

      }
```

```
            else
            {
                Console.WriteLine(Path.GetFullPath(path); // File ketemu
            }
            hasilpath.Enqueue(Path.GetFullPath(path));
            cari_bapak(path);

        }
        else
        {
            Console.WriteLine(path);
        }
    }
    while (visited.Any() && found == false)
    {
        folderfile baru = visited.Dequeue();
        string foldernext = baru.anakfolder;
        BFS(foldernext);
    }

}
```

Implementasi dengan metode DFS :

```
public void DFS(string path)
{
    string[] isi = Directory.GetFiles(path, "*.*");
    foreach (string file in isi)
    {
        penyimpanan.Enqueue(new folderfile(path, file));
```

```
            if (Path.GetFileName(file) == namafile)
            {
                if (kemungkinan == "N" && found == false)
                {
                    Console.WriteLine(Path.GetFullPath(file) //  File ketemu
                    found = true;
                    hasilpath.Enqueue(Path.GetFullPath(file));
                    cari_bapak(file);
                }
                else
                {
                    Console.WriteLine(Path.GetFullPath(file); // File ketemu
hasilpath.Enqueue(Path.GetFullPath(file));
                    cari_bapak(file);
                    continue;
                }
            }
            else
            {
                Console.WriteLine(file);
            }
        }

        string[] folder = Directory.GetDirectories(path);

        foreach (string subfolder in folder)
        {
            if (found == false)
            {
```

```
            Console.WriteLine(subfolder);


            penyimpanan.Enqueue(new folderfile(path, subfolder));
            DFS(subfolder);
        }
    }
}
```

Pembuatan pohon DFS dan BFS

```
public void show_graph_DFS()
{

    bool ada = false;
    bool jabingan;
    foreach (folderfile cek in penyimpanan)
    {
        jabingan = false;


        FileAttributes attr = File.GetAttributes(cek.anakfolder);
        if ((attr & FileAttributes.Directory) == FileAttributes.Directory)
        {
            foreach (folderfile gajetot in ketemuhasil)
            {
                if (Path.GetFullPath(gajetot.parent) == cek.anakfolder ||
Path.GetFullPath(gajetot.anakfolder) == cek.anakfolder)
                {
                    graph.AddEdge(Path.GetFullPath(cek.parent),
Path.GetFullPath(cek.anakfolder)).Attr.Color = Microsoft.Msagl.Drawing.Color.Green;
                    jabingan = true;
                    break;
                }
```

```
                }
                if (jabingan == false)
                {
                    graph.AddEdge(Path.GetFullPath(cek.parent),
Path.GetFullPath(cek.anakfolder)).Attr.Color = Microsoft.Msagl.Drawing.Color.Red;
                    graph.FindNode(cek.anakfolder).Attr.FillColor =
Microsoft.Msagl.Drawing.Color.Red;
                }
            }
            else
            {
                if (Path.GetFileName(cek.anakfolder) == namafile)
                {
                    if (kemungkinan == "N" && ada == false)
                    {
                        graph.AddEdge(Path.GetFullPath(cek.parent),
Path.GetFullPath(cek.anakfolder)).Attr.Color = Microsoft.Msagl.Drawing.Color.Green;
                        graph.FindNode(cek.anakfolder).Attr.FillColor =
Microsoft.Msagl.Drawing.Color.PaleGreen;
                        graph.FindNode(cek.anakfolder).Attr.Shape =
Microsoft.Msagl.Drawing.Shape.Diamond;
                        ada = true;
                        break;


                    }
                    else
                    {
                        graph.AddEdge(cek.parent, cek.anakfolder).Attr.Color =
Microsoft.Msagl.Drawing.Color.Green;
                        graph.FindNode(cek.anakfolder).Attr.FillColor =
Microsoft.Msagl.Drawing.Color.PaleGreen;
                        graph.FindNode(cek.anakfolder).Attr.Shape =
Microsoft.Msagl.Drawing.Shape.Diamond;
```

```
                    graph.FindNode(cek.anakfolder).Label.Text = new
DirectoryInfo(cek.anakfolder).Name;
                    continue;
                }
            }
            else
            {
                graph.AddEdge(Path.GetFullPath(cek.parent),
Path.GetFullPath(cek.anakfolder)).Attr.Color = Microsoft.Msagl.Drawing.Color.Red;
                graph.FindNode(cek.anakfolder).Attr.FillColor =
Microsoft.Msagl.Drawing.Color.Red;
            }
        }

        graph.FindNode(cek.parent).Label.Text = new DirectoryInfo(cek.parent).Name;
        graph.FindNode(cek.anakfolder).Label.Text = new
DirectoryInfo(cek.anakfolder).Name;
    }

    folderfile warnajalur;
    while (ketemuhasil.Count > 0)
    {
        warnajalur = new folderfile(ketemuhasil.Dequeue());
        graph.FindNode(warnajalur.parent).Attr.FillColor =
Microsoft.Msagl.Drawing.Color.Green;
        graph.FindNode(warnajalur.anakfolder).Attr.FillColor =
Microsoft.Msagl.Drawing.Color.Green;
        graph.FindNode(warnajalur.parent).Label.Text = new
DirectoryInfo(warnajalur.parent).Name;
        graph.FindNode(warnajalur.anakfolder).Label.Text = new
DirectoryInfo(warnajalur.anakfolder).Name;
    }
    viewer.Graph = graph;
```

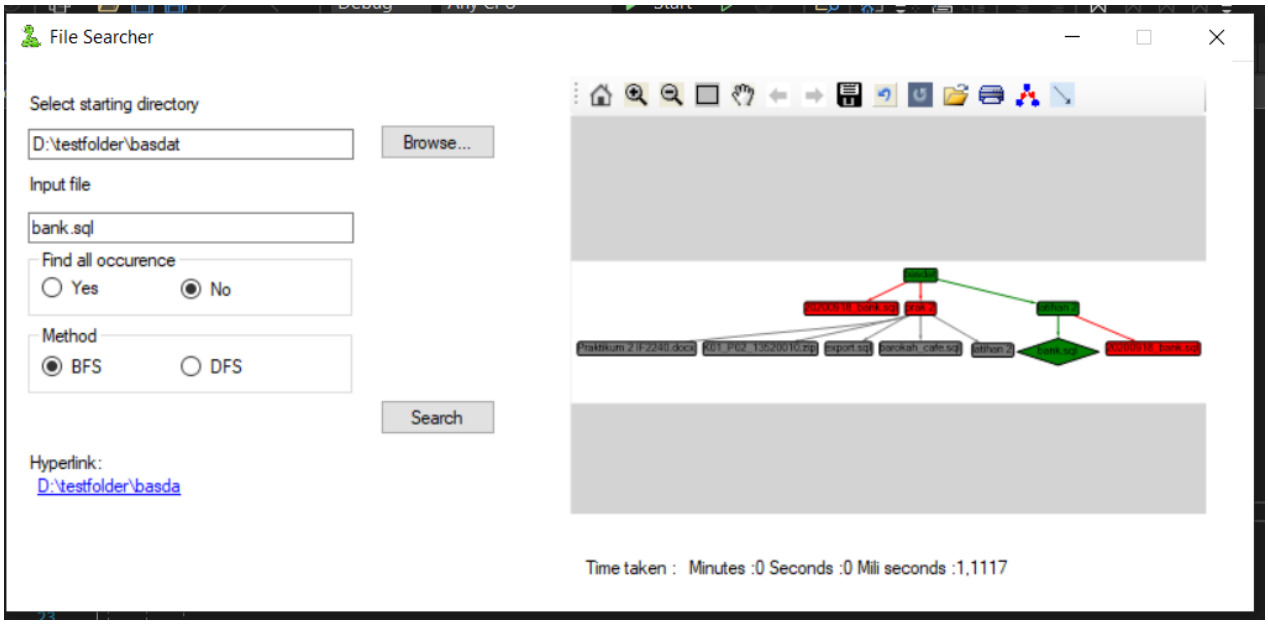
```
            //associate the viewer with the form

        }


        public void show_graf_BFS()
        {
            bool ada = false;
            bool jabingan;
            foreach (folderfile cek in penyimpanan)
            {
                jabingan = false;

                FileAttributes attr = File.GetAttributes(cek.anakfolder);
                if ((attr & FileAttributes.Directory) == FileAttributes.Directory)
                {
                    foreach (folderfile gajetot in ketemuhasil)
                    {
                        if (Path.GetFullPath(gajetot.parent) == cek.anakfolder ||
Path.GetFullPath(gajetot.anakfolder) == cek.anakfolder)
                        {
                            graph.AddEdge(Path.GetFullPath(cek.parent),
Path.GetFullPath(cek.anakfolder)).Attr.Color = Microsoft.Msagl.Drawing.Color.Green;
                            jabingan = true;
                            break;
                        }
                    }
                    if (jabingan == false)
                    {
                        graph.AddEdge(Path.GetFullPath(cek.parent),
Path.GetFullPath(cek.anakfolder)).Attr.Color = Microsoft.Msagl.Drawing.Color.Red;
```

```
                    graph.FindNode(cek.anakfolder).Attr.FillColor =
Microsoft.Msagl.Drawing.Color.Red;
                }
            }
            else
            {
                if (Path.GetFileName(cek.anakfolder) == namafile)
                {
                    if (kemungkinan == "N" && ada == false)
                    {
                        graph.AddEdge(Path.GetFullPath(cek.parent),
Path.GetFullPath(cek.anakfolder)).Attr.Color = Microsoft.Msagl.Drawing.Color.Green;
                        graph.FindNode(cek.anakfolder).Attr.FillColor =
Microsoft.Msagl.Drawing.Color.PaleGreen;
                        graph.FindNode(cek.anakfolder).Attr.Shape =
Microsoft.Msagl.Drawing.Shape.Diamond;
                        ada = true;
                        break;
                    }
                    else
                    {
                        graph.AddEdge(cek.parent, cek.anakfolder).Attr.Color =
Microsoft.Msagl.Drawing.Color.Green;
                        graph.FindNode(cek.anakfolder).Attr.FillColor =
Microsoft.Msagl.Drawing.Color.PaleGreen;
                        graph.FindNode(cek.anakfolder).Attr.Shape =
Microsoft.Msagl.Drawing.Shape.Diamond;
                        graph.FindNode(cek.anakfolder).Label.Text = new
DirectoryInfo(cek.anakfolder).Name;
                        continue;
                    }
                }
                else
```

```
                {
                    graph.AddEdge(Path.GetFullPath(cek.parent),
Path.GetFullPath(cek.anakfolder)).Attr.Color = Microsoft.Msagl.Drawing.Color.Red;
                    graph.FindNode(cek.anakfolder).Attr.FillColor =
Microsoft.Msagl.Drawing.Color.Red;
                }
            }

            graph.FindNode(cek.parent).Label.Text = new DirectoryInfo(cek.parent).Name;
            graph.FindNode(cek.anakfolder).Label.Text = new
DirectoryInfo(cek.anakfolder).Name;

        }
        folderfile tempo;
        while (visited.Count > 0)
        {
            tempo = new folderfile(visited.Dequeue());

            graph.AddEdge(Path.GetFullPath(tempo.parent),
Path.GetFullPath(tempo.anakfolder)).Attr.Color = Microsoft.Msagl.Drawing.Color.Gray;
            graph.FindNode(tempo.anakfolder).Attr.FillColor =
Microsoft.Msagl.Drawing.Color.Gray;
            graph.FindNode(tempo.parent).Label.Text = new
DirectoryInfo(tempo.parent).Name;
            graph.FindNode(tempo.anakfolder).Label.Text = new
DirectoryInfo(tempo.anakfolder).Name;

        }

        folderfile warnajalur;
        while (ketemuhasil.Count > 0)
        {
            warnajalur = new folderfile(ketemuhasil.Dequeue());
```

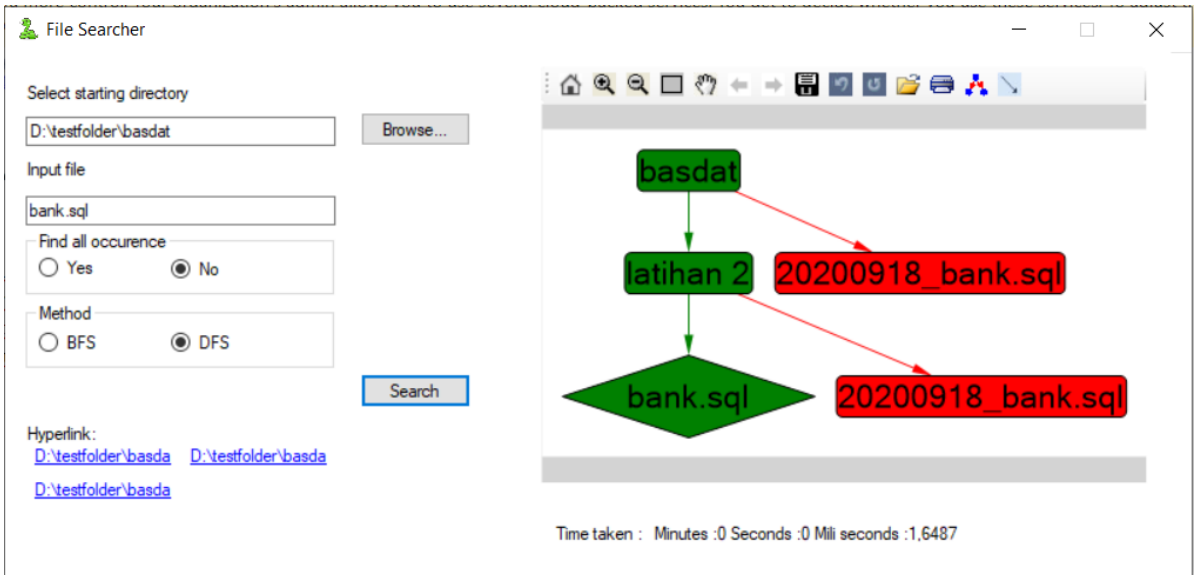
```
            graph.FindNode(warnajalur.parent).Attr.FillColor =
Microsoft.Msagl.Drawing.Color.Green;

            graph.FindNode(warnajalur.anakfolder).Attr.FillColor =
Microsoft.Msagl.Drawing.Color.Green;

            graph.FindNode(warnajalur.parent).Label.Text = new
DirectoryInfo(warnajalur.parent).Name;

            graph.FindNode(warnajalur.anakfolder).Label.Text = new
DirectoryInfo(warnajalur.anakfolder).Name;

        }

        viewer.Graph = graph;


    }

}
```

Pengujian



Gambar 4.1 Uji pencarian file metode BFS dengan all occurrence no

Pada pengujian di atas, digunakan metode BFS dengan all occurrence no. Dapat dilihat pada gambar bahwa jalur berwarna hijau merupakan jalur ditemukan file, jalur berwarna merah merupakan jalur yang sudah dicek namun tidak ditemukan solusi, sedangkan jalur berwarna abu-abu merupakan jalur yang sudah dimasukkan ke dalam antrian namun belum dicek karena pencarian sudah menemukan file sehingga proses berhenti.



Gambar 4.2 Uji pencarian file metode DFS dengan all occurrence no

Pada pengujian di atas, digunakan metode DFS dengan all occurrence no. Dapat dilihat pada gambar bahwa jalur berwarna hijau merupakan jalur ditemukan file, sedangkan jalur berwarna merah merupakan jalur yang sudah dicek namun tidak ditemukan solusi.

BAB V

KESIMPULAN DAN SARAN

DFS dan BFS dapat digunakan untuk memeceahkan banyak kasus. Dalam tugas besar ini, kami dapat memahami konsep DFS dan BFS dengan mengimplementasikannya dalam program File Crawler. Sebuah GUI dibuat untuk membantu user menggunakan program ini. Walaupun GUI yang dibuat sangatlah sederhana, GUI ini tetap bisa mengimplementasikan algoritma backend sesuai fitur yang diminta oleh spesifikasi tugas.

DAFTAR PUSTAKA

1. Munir, R. (n.d.). *Breadth/depth First Search (BFS/DFS) (bagian 1)*. Website Rinaldi Munir. Retrieved March 25, 2022, from https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag1.pdf
2. Munir, R. (n.d.). *Breadth/depth First Search (BFS/DFS) (bagian 2)*. Website Rinaldi Munir. Retrieved March 25, 2022, from https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag2.pdfh
3. Kathleen Dollard Principal Program Manager, Dollard, K., 8, R. L. N., Lander, R., 8, S. K. N., Kalaskar, S., Drabek, M., Sommer, T., Pilichowski, M., Gooding, T., Jeske, D., Torgersen, M., Selormey, P., Kempf, W., Mortazavi, S., Leibovits, A., Kellner, P.,

Muryshkin, E., Larini, P., … Whited, M. (2021, November 9). *Welcome to C# 10*. .NET Blog. Retrieved March 25, 2022, from https://devblogs.microsoft.com/dotnet/welcome-to-csharp-10/

4. *Applications of breadth first traversal*. GeeksforGeeks. (2019, December 10). Retrieved March 25, 2022, from https://www.geeksforgeeks.org/applications-of-breadth-first-traversal/

Link Penting

Tautan GitHub source code

Video demonstrasi program