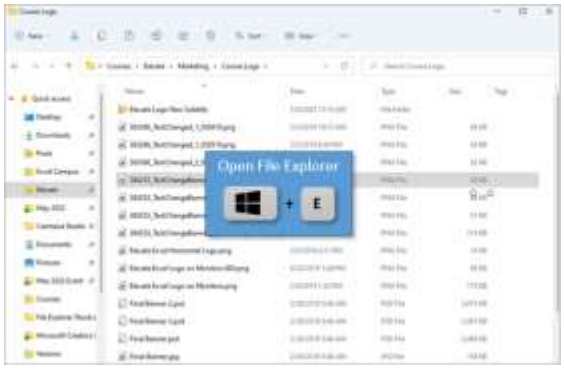
PENERAPAN ALGORITMA BFS DALAM IMPLEMENTASI *FOLDER CRAWLING* BERBASIS KATA KUNCI (*KEYWORD*)

Lola
Program Studi Teknik Informatika, FTI, Institut Teknologi Budi Utomo Jakarta,
lola.rezak@gmail.com

Abstrak
Folder Crawling merupakan proses penelusuran folder-folder yang ada di dalam direktori untuk mendapatkan direktori yang diinginkan. Folder Crawling umumnya dilakukan dengan memberikan input berupa nama file yang dituju. Namun tidak jarang, user lupa dengan nama file yang tersimpan didalam komputernya. Penambahan fitur seperti pencarian dengan kata kunci dapat dilakukan untuk menangani masalah tersebut. Folder Crawling berbasis isi konten seperti yang terdapat pada fitur searching pada File Explorer di Windows OS dilakukan dengan mencari hasil pengetikan pengguna dengan isi konten pada seluruh file yang berada di direktori awal pencarian. Pengaplikasian Folder Crawling berbasis isi konten ini dapat diterapakan dengan memanfaatkan algoritma pencarian dan pencocokan string, salah satunya algoritma BFS.
Kata kunci : Folder Crawling, Algoritma Pencarian BFS, Algoritma Pencocokan String.

1. PENDAHULUAN

Pada saat user ingin mencari file spesifik yang tersimpan pada komputer, seringkali task tersebut membutuhkan waktu yang lama apabila user melakukannya secara manual. Bukan saja harus membuka beberapa folder hingga dapat mencapai directory yang diinginkan, user bahkan dapat lupa di mana user meletakkan file tersebut. Sebagai akibatnya, user harus membuka berbagai folder secara satu persatu hingga user menemukan file yang diinginkan. Hal ini pastinya akan sangat memakan waktu dan energi. Apalagi bila user lupa nama file yang dicari, user harus membuka file yang diinginkan. Meskipun demikian, user tidak perlu cemas dalam menghadapi persoalan tersebut sekarang. Pasalnya, hampir seluruh sistem operasi sudah menyediakan fitur search yang dapat digunakan untuk mencari file yang kita inginkan. User cukup memasukkan query atau kata kunci pada kotak pencarian, dan komputer akan mencarikan seluruh file pada suatu starting directory (hingga seluruh children-nya) yang berkorespondensi terhadap query yang kita masukkan. Seperti pada Gambar 1 fitur searching pada File Explorer Windows.

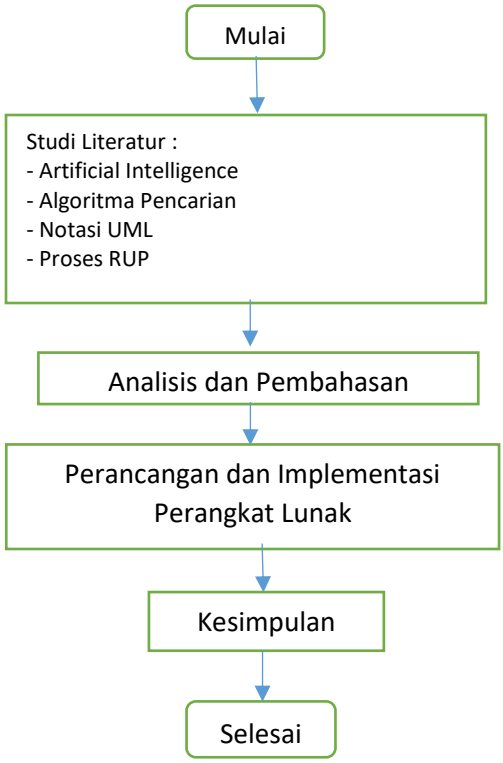


Gambar 1. Search File pada File Explorer Menggunakan Kata Kunci
Sumber : https://www.excelcampus.com/tips-shortcuts/file-explorer-shortcuts/

Fitur ini diimplementasikan dengan teknik folder crawling, di mana mesin komputer akan mulai mencari file yang sesuai dengan query mulai dari starting directory hingga seluruh children dari starting directory tersebut sampai satu file pertama atau seluruh file ditemukan atau tidak ada file yang ditemukan. Algoritma yang dapat dipilih untuk melakukan crawling tersebut pun dapat bermacam-macam dan setiap algoritma akan memiliki teknik dan konsekuensinya sendiri.

2. METODOLOGI

Metodologi penelitian digambarkan dalam bentuk diagram alir sebagai berikut :



Gambar 2. Diagram Alir Metodologi Penelitian
Sumber : https://www.researchgate.net/publication/338235695_Metode-Metode_Penelitian_Dalam_Penulisan_Jurnal_Ilmiah_Elektronik

3. HASIL DAN PEMBAHASAN

3.1 HASIL

Algortima Pencarian merupakan salah satu teknik dalam pencarian solusi dengan membentuk ruang status pencarian berupa pohon. Secara umum, algoritma pencarian string dapat dibagi menjadi dua jenis algoritma pencarian, yaitu:

1. Uniformed Search

Uniformed Search merupakan jenis pencarian yang tidak memiliki informasi tambahan tentang keadaan atau ruang pencarian selain cara mengakses tree, sehingga implementasinya mirip dengan brute force.

Berikut ini adalah beberapa algoritma dengan jenis Uninformed search :

a. *Breadth-first Search*
b. *Depth-first Search*
c. *Depth-limited Search*
d. *Iterative Deepning Dept-First Search*
e. *Uniform Cost Search*
f. *Bidirectional Search*

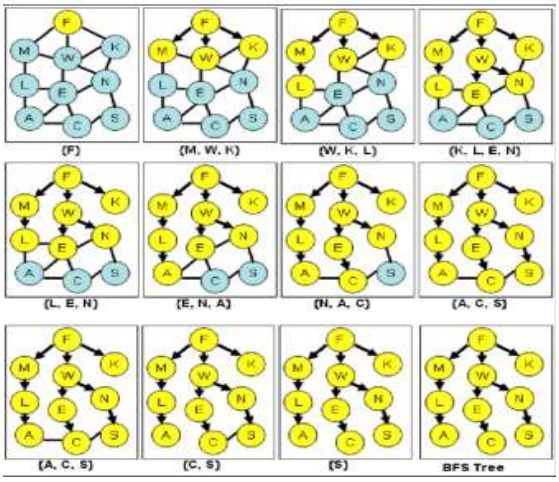
2. *Informed Search*

Informed Search merupakan jenis pencarian yang memiliki informasi tambahan tentang keadaan atau ruang pencarian, sehingga implementasinya akan lebih efisien. Berikut ini adalah beberapa algoritma dengan jenis *Informed search* :

a. Berbasis heuristik
b. Greedy Best First Search
c. A*

Algoritma Breadth First Search (BFS) atau dikenal sebagai pencarian melebar merupakan salah satu algoritma pencarian data pada tree yang dilakukan dengan cara mengunjungi semua simpul yang bertetangga dengan simpul yang sedang diproses itu. Algoritma tersebut adalah sebagai berikut:

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.



Gambar 3. Contoh Pengaksesan Node dengan Algoritma BFS
Sumber : Meghanathan Natarajan, 2021, https://www.researchgate.net/figure/Pseudo-Code-for-Breadth-First-Search-BFS_fig11_266008323

Prototipe Folder Crawling dibuat dengan menggunakan bahasa Python dan berbasis Command Line Interface (CLI). Dengan masukan berupa kata kunci yang merupakan bagian dari isi konten dari file yang dicari. Pencarian akan dilakukan pada folder Root yang merupakan sampel direktori awal pencarian. Dan pencarian dilakukan pada file bertipe .txt. Pada program digunakan 3 variabel yang dijadikan vairable global, untuk memudahkan modifikasi nilai pada variabel tersebut.

```
import os
import operator

# KAMUS GLOBAL:
    #listToSearch : list of string konten dicari
    #result : dictionary hasil pencarian
    #queueFolder : list of string direktori
    #             (berperilaku sebagai queue)
```

Gambar 4. Source Code Import Library dan Kamus Global
Sumber : Munir, R, 2021.

Hasil yang diharapkan adalah ditampilkan seluruh path directory file yang memiliki isi konten yang cocok dengan input pengguan.

Algoritma Breadth-First Search diimplementasikan dengan memodifikasi variabel global listToSearch, result dan queueFolder hingga didapatkan. Pada setiap node yang dikunjungi akan dijalankan fungsi KMP untuk melakukan pencocokan string pada tiap kata kunci pada listToSearch.

```
def startBFS(startingDirectory):
    global listToSearch
    global result
    global queueFolder
    print("\nCurrent path :",startingDirectory)
    folders_files = [f for f in os.listdir(startingDirectory)]
    for folder_file in folders_files:
        if os.path.isfile(os.path.join(startingDirectory,folder_file)):
            print(folder_file)
            file = os.path.join(startingDirectory,folder_file)
            temp = 0
            with open(file) as input_file:
                content = input_file.read()
                content = content.replace('\n', ' ')
                for string in listToSearch:
                    temp += KMPSearch(string, content)
            if temp != 0:
                result[file] = temp
        if os.path.isdir(os.path.join(startingDirectory,folder_file)):
            queueFolder.append(os.path.join(startingDirectory,folder_file))
    nextStepBFS()

def nextStepBFS():
    global listToSearch
    global result
    global queueFolder
    currDir = queueFolder.pop(0)
    print("\nCurrent path :",currDir)
    folders_files = [f for f in os.listdir(currDir)]
    for folder_file in folders_files:
        if os.path.isfile(os.path.join(currDir,folder_file)):
            print(folder_file)
            file = os.path.join(currDir,folder_file)
            temp = 0
            with open(file) as input_file:
                content = input_file.read()
                content = content.replace('\n', ' ')
                for string in listToSearch:
                    temp += KMPSearch(string, content)
            if temp != 0:
                result[file] = temp
        if os.path.isdir(os.path.join(currDir,folder_file)):
            queueFolder.append(os.path.join(currDir,folder_file))

    if(len(queueFolder) != 0):
        nextStepBFS()
```

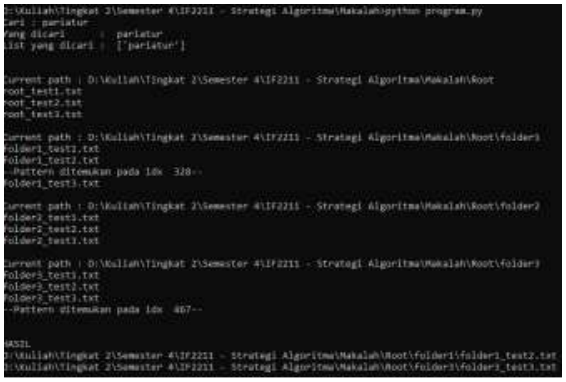
Gambar 5. Source Code Algoritma BFS
Sumber : Munir, R, 2021.

```
result = {}
textToSearch = input("Cari : ")
listToSearch = textToSearch.split()
queueFolder = []

print("Yang dicari      : ", textToSearch)
print("List yang dicari : ", listToSearch)
print()
startingDirectory =  os.path.join(os.getcwd(),'Root')
startBFS(startingDirectory)

if(len(result) == 0):
    print("\nHasil tidak ditemukan")
else:
    print("\n\nHASIL")
    sortedResult = dict(sorted(result.items(),
                               key=operator.itemgetter(1)
                               reverse=True))
    for res in (sortedResult):
        print(res)
```

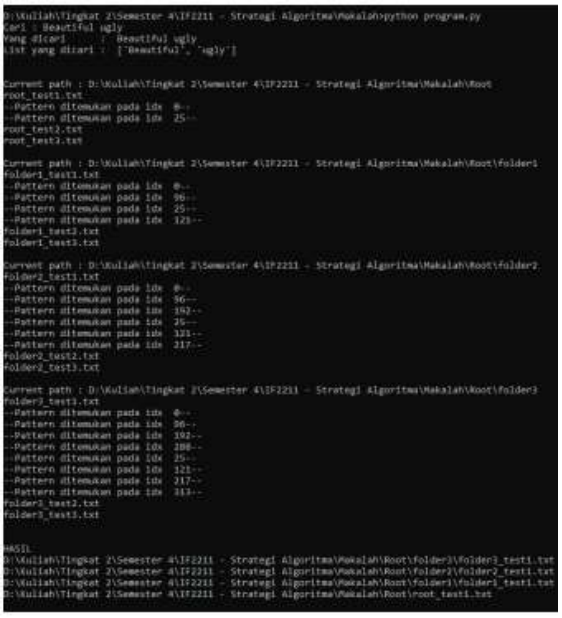
Gambar 6. Source Code Main Program
Sumber : Munir, R, 2021.

User diminta memasukan input berupa kata kunci atau isi konten yang dicari. Kata kunci akan di-split dengan separator spasi dan hasil split dimasukkan ke dalam list listToSearch.

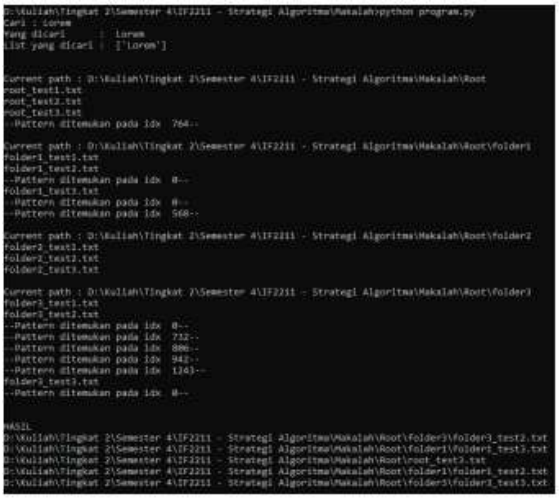
Setelah itu dilakukan pencarian dengan algoritma BFS, kemudian program akan menampilkan hasil secara berurutan sesuai dengan banyaknya kata kunci ditemukan didalam file.



Gambar 7. Hasil Pengujian 1
Sumber : GeeksforGeeks.2020



Gambar 8. Hasil Pengujian 2
Sumber : GeeksforGeeks.2020



Gambar 9. Hasil Pengujian 3
Sumber : GeeksforGeeks.2020

3.2 PEMBAHASAN

Pada hasil pengujian 1 sampai 3, pencarian file secara BFS sudah memberikan hasil yang diinginkan. Serta untuk hasil akhir, akan ditampilkan hasil yang sesuai dengan pengurutan terhadap jumlah queri ditemukan pada file

4. KESIMPULAN

Algoritma Breadth-First Search (BFS) dapat diterapkan dalam Folder Crawling berbasis isi konten. BFS digunakan sebagai algortima untuk pengaksesan file dengan isi konten. Melakukan Folder Crawling dengan input berupa kata kunci/isi konten, akan lama apabila jumlah file yang diperiksa banyak dan isi konten yang kompleks.

DAFTAR PUSTAKA

1. GeeksforGeeks.2020. Applications of String Matching Algorithms.
2. Meghanathan Natarajan, (2021). Pseudo Code for Breadth First Search (BFS). https://www.researchgate.net/figure/Pseudo-Code-for-Breadth First-Search-BFS_fig11_266008323.
3. Munir, R, (2021). Bahan Kuliah IF2211 Strategi Algoritma:Breadth/Depth First Search (BFS/DFS).