Pengaplikasian Algoritma BFS dan KMP dalam Implementasi Folder Crawling berbasis Isi Konten

Ghebyon Tohada Nainggolan - 13520079
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13520079@std.stei.itb.ac.id

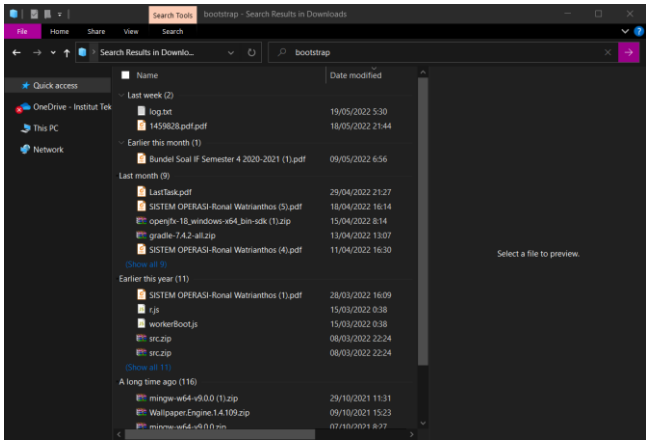
***Abstrak*—*Folder Crawling* merupakan proses penelusuran folder-folder yang ada di dalam direktori untuk mendapatkan direktori yang diinginkan. *Folder Crawling* umumnya dilakukan dengan memberikan input berupa nama file yang dituju. Namun tidak jarang, *user* lupa dengan nama file yang tersimpan didalam 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 dan *Knuth-Morris-Pratt*(KMP) .**

Keywords—Folder Crawling; algoritma pencarian; BFS; algoritma pencocokan string; *Knuth-Morris-Pratt*(KMP)

I. 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 1.1 fitur searching pada File Explorer Windows.



Gambar 1.1 Search file pada file explorer menggunakan kata kunci

Sumber : Dokumentasi penulis

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/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. Untuk algoritma pecarian string juga dapat dipilih bermacam-macam sesuai dengan kebutuhan seperti algoritma *Knuth-Morris Pratt* (KMP).

II. LANDASAN TEORI

A. *Algoritma Pencarian*

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.

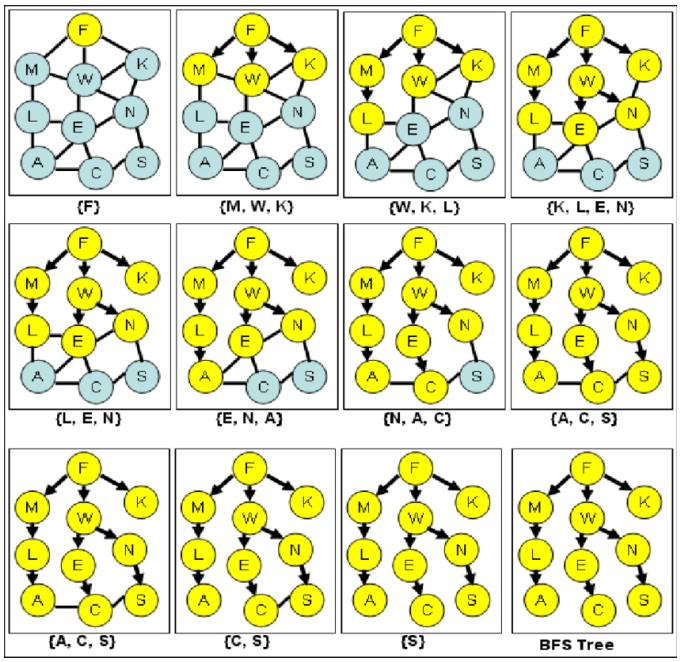
Berikut ini adalah beberapa algoritma dengan jenis *Informed search*

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

B. *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 2.1 Contoh Pengaksesan Node dengan algoritma BFS

Sumber : https://www.researchgate.net/figure/Pseudo-Code-for-Breadth-First-Search-BFS_fig11_266008323

C. *Pencocokan String*

Pencocokan string atau *string matching* merupakan proses pencarian *string pattern* P[0..n-1] pada teks T[0..m-1] dengan panjang string *pattern* n lebih kecil dari panjang string pada teks.

Secara umum, algoritma pencocokan string dibagi menjadi dua jenis algoritma, yaitu :

1. *Exact String Matching Algortihms*

Exact String Matching Algortihms merupakan jenis pencocokan string untuk menemukan, satu, beberapa, maupun semua kemunculan string *pattern* pada teks yang setiap pencocokannya sempurna atau semua abjad pada string *pattern* harus dicocokkan dengan urutan yang sesuai. Jenis pencocokan string ini dibagi lagi menjadi empat kategori :

a. *Algorithms based on character comparison*
Contoh dari algoritma ini, diantaranya :
- Naïve Algorithm
- Knuth Morris Pratt Algorithm
- Boyere Moore Algorithm
- Using the Trie data structur

b. Metode *Deterministic Finite Automaton* (DFA)
Salah satu contoh penerapan kategori ini adalah *Automaton Matcher Algorithm*
c. *Algorithms based on Bit*
Salah satu contoh penerapan kategori ini adalah *Aho-Corasick Algorithm*
d. *Hashing-string matching algorithms*
Salah satu contoh penerapan kategori ini adalah *Rabi Karp Algorithm*

2. *Approximate String Matching Algorithms*

Approximate String Matching Algorithms merupakan jenis pencocokan string untuk menemukan semua kemunculan pola dalam teks yang *edit distance*-nya mendekati suatu nilai k. Untuk penentuan nilai *edit distance-nya* dapat digunakan dengan beberapa cara, contohnya dengan menggunakan *Levenshtein edit distance* atau *Hamming edit distance*. Jenis pencocokan string ini dapat dibagi lagi menjadi tiga kategori, yaitu:

a. Naïve Approach
b. Sellers Algorithm (Dynamic Programming)
c. Shift or Alghorithm (Bitmap Algorithm)

D. *Algoritma Knuth-Morris-Pratt*

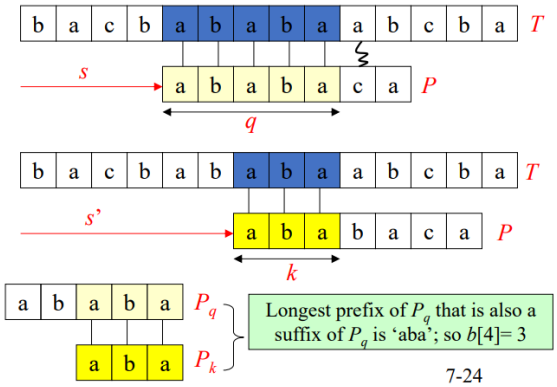
Algoritma *Knuth-Morris Pratt* merupakan salah satu algoritma pencocokan string yang dikembangkan oleh Donald Ervin Knuth (1967) dan James H. Morris bersama Vaughan R. Pratt(1966) secara terpisah, namun dipubilkasikan secara bersamaan pada tahun 1977.

Ide utama dari algoritma *Knuth-Morris-Pratt* melakukan pencocokan string *pattern* dari kiri ke kanan seperti pada *Brute Force* namun, pergeseran dilakukan sebanyak panjang *prefix* yang juga merupakan *suffix.*

Berikut adalah langkah-langkah pencocokan string dengan menggunakan algoritma KMP :

1. Tentukan string *pattern* dan teks
2. Hitung nilai fungsi pinggiran untuk setiap karakter pada string *pattern.*
3. Dari kiri ke kanan, lakukan pencocokan string. Apbaila terjadi ketidakcocokan, lakukan pergeseran pattern terhadap teks dengan banyak pergeseran senilai dengan fungsi pinggiran b(k) dengan k = j -1 dan j merupakan posisi terjadi *mismatch.*
4. Lakukan pencocokan ulang seperti pada langkah ke 2, sehingga didapatkan string yang cocok atau tidak didapatkan string yang cocok pada teks.

Berikut adalah contoh penerapan algoritma KMP dalam pencocokan pattern “ababaca” pada teks “bacbababaabcba”



Gambar 2.2 Contoh Penyelesaian pencocokan *pattern* dengan teks menggunakan algoritma KMP
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

Dapat dilihat, pada saat ditemukan beberapa komponen berurutan yang sesuai dengan pattern P “ababaca” yaitu “ababa”, hendak dilakukan pergeseran sebanyak 3 yaitu *prefix* “aba” yang juga merupakan *sufix*. Setelah dilakukan pergeseran, dilakukan lagi pencocokan hingga didapatkan atau tidak didapatkan string *pattern* yang sesuai pada teks.

III. Implementasi Prototipe Folder Crawling

Prototipe *Folder Crawling* dibuat dengan menggunakan bahas 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 3.1 Source Code import library dan Kamus Global
Sumber : Dokumentasi Penulis

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

A. *Implementasi Algoritma Breadth First Search*

Algoritma *Breadth-First Search* diimplementasikan dengan memodifika 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 3.2 Source Code Algoritma BFS
Sumber : Dokumentasi penulis

B. Implementasi Algortima Knuth-Morris-Pratt

```
def KMPSearch(pattern, text):
    m = len(pattern)
    n = len(text)
    countFound = 0
    lps = [0]*m
    computeLPSArray(pattern, m, lps)

    i = 0
    j = 0
    while i < n:
        if pattern[j].lower() == text[i].lower():
            i += 1
            j += 1

        if j == m:
            print ("--Pattern ditemukan pada idx ", str(i-j)+ "--")
            countFound += 1
            j = lps[j-1]
        elif i < n and pattern[j].lower() != text[i].lower():
            if j != 0:
                j = lps[j-1]
            else:
                i += 1
    return countFound

def computeLPSArray(pattern, m, lps):
    length = 0
    lps[0]
    i = 1
    while i < m:
        if pattern[i].lower() == pattern[length].lower():
            length += 1
            lps[i] = length
            i += 1
        else:
            if length != 0:
                length = lps[length-1]
            else:
                lps[i] = 0
                i += 1
```

Gambar 3.3 Source Code Algoritma KMP
Sumber : Dokumentasi Penulis

Fungsi KMPSearch akan mengembalikan hasil berupa variabel countFound yang nilainya adalah jumlah string pada teks yang sesuai dengan pattern.

C. Main Program

```
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 3.3 Source Code Main Program
Sumber : Dokumentasi Penulis

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.

D. Hasil Pengujian

```
D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah>python program.py
Cari : pariatur
Yang dicari      :  pariatur
List yang dicari :  ['pariatur']


Current path : D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root
root_test1.txt
root_test2.txt
root_test3.txt

Current path : D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder1
folder1_test1.txt
folder1_test2.txt
--Pattern ditemukan pada idx  328--
folder1_test3.txt

Current path : D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder2
folder2_test1.txt
folder2_test2.txt
folder2_test3.txt

Current path : D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder3
folder3_test1.txt
folder3_test2.txt
folder3_test3.txt
--Pattern ditemukan pada idx  467--


HASIL
D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder1\folder1_test2.txt
D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder3\folder3_test3.txt
```

Gambar 3.4 Hasil Pengujian 1
Sumber : Dokumentasi Penulis

```
D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah>python program.py
Cari : Beautiful ugly
Yang dicari      :  Beautiful ugly
List yang dicari :  ['Beautiful', 'ugly']


Current path : D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root
root_test1.txt
--Pattern ditemukan pada idx  0--
--Pattern ditemukan pada idx  25--
root_test2.txt
root_test3.txt

Current path : D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder1
folder1_test1.txt
--Pattern ditemukan pada idx  0--
--Pattern ditemukan pada idx  96--
--Pattern ditemukan pada idx  25--
--Pattern ditemukan pada idx  121--
folder1_test2.txt
folder1_test3.txt

Current path : D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder2
folder2_test1.txt
--Pattern ditemukan pada idx  0--
--Pattern ditemukan pada idx  96--
--Pattern ditemukan pada idx  192--
--Pattern ditemukan pada idx  25--
--Pattern ditemukan pada idx  121--
--Pattern ditemukan pada idx  217--
folder2_test2.txt
folder2_test3.txt

Current path : D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder3
folder3_test1.txt
--Pattern ditemukan pada idx  0--
--Pattern ditemukan pada idx  96--
--Pattern ditemukan pada idx  192--
--Pattern ditemukan pada idx  288--
--Pattern ditemukan pada idx  25--
--Pattern ditemukan pada idx  121--
--Pattern ditemukan pada idx  217--
--Pattern ditemukan pada idx  313--
folder3_test2.txt
folder3_test3.txt


HASIL
D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder3\folder3_test1.txt
D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder2\folder2_test1.txt
D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder1\folder1_test1.txt
D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\root_test1.txt
```

Gambar 3.5 Hasil Pengujian 2
Sumber : Dokumentasi Penulis

```
D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah>python program.py
Cari : Lorem
Yang dicari      :  Lorem
List yang dicari :  ['Lorem']


Current path : D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root
root_test1.txt
root_test2.txt
root_test3.txt
--Pattern ditemukan pada idx  764--

Current path : D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder1
folder1_test1.txt
folder1_test2.txt
--Pattern ditemukan pada idx  0--
folder1_test3.txt
--Pattern ditemukan pada idx  0--
--Pattern ditemukan pada idx  568--

Current path : D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder2
folder2_test1.txt
folder2_test2.txt
folder2_test3.txt

Current path : D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder3
folder3_test1.txt
folder3_test2.txt
--Pattern ditemukan pada idx  0--
--Pattern ditemukan pada idx  732--
--Pattern ditemukan pada idx  806--
--Pattern ditemukan pada idx  942--
--Pattern ditemukan pada idx  1243--
folder3_test3.txt
--Pattern ditemukan pada idx  0--


HASIL
D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder3\folder3_test2.txt
D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder1\folder1_test3.txt
D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\root_test3.txt
D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder1\folder1_test2.txt
D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder3\folder3_test3.txt
```

Gambar 3.6 Hasil Pengujian 3
Sumber : Dokumentasi Penulis

```
D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah>python program.py
Cari : laborum distinctio
Yang dicari      :  laborum distinctio
List yang dicari :  ['laborum', 'distinctio']


Current path : D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root
root_test1.txt
root_test2.txt
root_test3.txt
--Pattern ditemukan pada idx  508--

Current path : D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder1
folder1_test1.txt
folder1_test2.txt
--Pattern ditemukan pada idx  442--
folder1_test3.txt
--Pattern ditemukan pada idx  222--

Current path : D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder2
folder2_test1.txt
folder2_test2.txt
--Pattern ditemukan pada idx  274--
--Pattern ditemukan pada idx  346--
folder2_test3.txt
--Pattern ditemukan pada idx  21--
--Pattern ditemukan pada idx  46--
--Pattern ditemukan pada idx  256--

Current path : D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder3
folder3_test1.txt
folder3_test2.txt
folder3_test3.txt


HASIL
D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder2\folder2_test3.txt
D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder2\folder2_test2.txt
D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\root_test3.txt
D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder1\folder1_test2.txt
D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder1\folder1_test3.txt
```

Gambar 3.7 Hasil Pengujian 4
Sumber : Dokumentasi Penulis

```
D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah>python program.py
Cari : nihil
Yang dicari      :  nihil
List yang dicari :  ['nihil']


Current path : D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root
root_test1.txt
root_test2.txt
root_test3.txt
--Pattern ditemukan pada idx  68--
--Pattern ditemukan pada idx  285--
--Pattern ditemukan pada idx  1034--

Current path : D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder1
folder1_test1.txt
folder1_test2.txt
folder1_test3.txt
--Pattern ditemukan pada idx  64--

Current path : D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder2
folder2_test1.txt
folder2_test2.txt
--Pattern ditemukan pada idx  422--
folder2_test3.txt
--Pattern ditemukan pada idx  238--
--Pattern ditemukan pada idx  391--

Current path : D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder3
folder3_test1.txt
folder3_test2.txt
--Pattern ditemukan pada idx  59--
--Pattern ditemukan pada idx  1093--
folder3_test3.txt
--Pattern ditemukan pada idx  77--
--Pattern ditemukan pada idx  255--


HASIL
D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\root_test3.txt
D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder2\folder2_test3.txt
D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder3\folder3_test2.txt
D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder3\folder3_test3.txt
D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder1\folder1_test3.txt
D:\Kuliah\Tingkat 2\Semester 4\IF2211 - Strategi Algoritma\Makalah\Root\folder2\folder2_test2.txt
```

Gambar 3.8 Hasil Pengujian 5
Sumber : Dokumentasi Penulis

E. Pembahasan

Pada hasil pengujian 1 sampai 5, pencarian file secara BFS dan pencocokan string dengan algortima KMP sudah memberikan hasil yang diinginkan. Serta untuk hasil akhir, akan ditampilkan hasil yang sesuai dengan pengurutan terhadap jumlah queri ditemukan pada file.

IV. Kesimpulan

Algoritma *Breadth-First Search* dan KMP dapat diterapkan dalam *Folder Crawling* berbasisi isi konten. BFS digunakan sebagai algortima untuk pengaksesan file dan KMP sebagai algortima pencocokan input pengguna dengan isi konten.

Melakukan *Folder Crawling* dengan input berupa kata kunci/isi konten, akan lama apabila jumlah file yang diperikas banyak dan isi konten yang kompleks

V. Ucapan Terima Kasih

Penulis mengucapkan puji syukur kepada Tuhan Yang Maha Esa atas berkat dan rahmatnya, penulis bisa menyelesaikan tugas makalah ini. Penulis juga mengucapkan terimakasih kepada Dr. Masayu Leylia Khodra, S.T., M.T. selaku dosen mata kuliah Strategi Algoritma, yang selama ini membimbing pembelajaran Strategi Algoritma dan Bapak Rinaldi Munir, yang selama satu semester ini menyediakan website yang dapat dengan mudah diakses berisi materi-materi kuliah, Latihan-latihan soal untuk kuis dan ujian, dan semua dokumen pembelajaran, soal dan lainnya yang tentunya berguna dalam proses pembelajaran Strategi Algoritma

Video Link Youtube

https://youtu.be/FWhXMQ92jPw

References

[1] Munir, R. 2021. Bahan Kuliah IF2211 Strategi Algoritma: Pencocokan String (String/Pattern Matching). https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf. Diakses tanggal 20 Mei 2022.

[2] GeeksforGeeks.2020. Applications of String Matching Algorithms. https://www.geeksforgeeks.org/applications-of-string-matching-algorithms/ . Diakses tanggal 20 Mei 2022

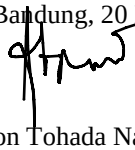
[3] Meghanathan Natarajan. 2011. Pseudo Code for Breadth First Search (BFS). https://www.researchgate.net/figure/Pseudo-Code-for-Breadth-First-Search-BFS_fig11_266008323 . Diakses tanggal 21 Mei 2022

[4] Munir, R. 2021. Bahan Kuliah IF2211 Strategi Algoritma:Breadth/Depth First Search(BFS/DFS). https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag2.pdf

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 20 Mei 2022



Ghebyon Tohada Nainggolan
13520079