Collabits Journal
Vol 1 No. 1 January 2024 : 13-20
E-ISSN : 3046-6709
https://publikasi.mercubuana.ac.id/index.php/collabits

File Search on an FTP Server Using the Fuzzywuzzy Method and the DFS Algorithm

Naufal Saidhus Syuhur[1], Bambang Jokonowo[2], Agil Dwiki Wijaya[2]

[1,2,3] Fakultas Ilmu Komputer, Universitas Mercu Buana, Indonesia

*Coressponden Author: 41522110068@student.mercubuana.ac.id

Abstract - The Depth-First Search (DFS) algorithm can be used to search for files on an FTP server. The fuzzywuzzy method can be applied to perform fuzzy string matching for the file names. To implement this, one can establish a connection to the FTP server using a programming language that supports FTP operations, retrieve the file listings using DFS, apply fuzzy matching to the file names, and retrieve the matching files. The fuzzywuzzy method is a string matching library that uses the Levenshtein Distance to calculate the differences between sequences. While there are no specific examples of using DFS with fuzzywuzzy on an FTP server, the general approach can be implemented using ftplib in Python for FTP operations and the fuzzywuzzy library for fuzzy string matching.

Keywords :
Depth-First Search (DFS);
Fuzzywuzzy;
FTP;

Article History:
Received: 25-10-2023
Revised: 08-12-2023
Accepted: 23-01-2024

Article DOI : 10.22441/collabits.v1i1.25561

1. INTRODUCTION

Pencarian file pada FTP Server menggunakan algoritma Depth-First Search (DFS) merupakan salah satu metode yang berguna dalam menavigasi struktur direktori pada server yang menyimpan berbagai file dan folder. FTP (File Transfer Protocol) adalah protokol yang digunakan untuk mentransfer file antar komputer melalui jaringan, dan DFS adalah algoritma pencarian yang melakukan penjelajahan secara mendalam pada suatu struktur data.

Penerapan DFS dalam pencarian file pada FTP Server memungkinkan pengguna untuk lebih efektif dalam menemukan file yang diinginkan tanpa harus mengetahui struktur folder secara rinci. Dengan menggunakan algoritma ini, pengguna dapat mengelola dan mengakses file-file yang ada pada server FTP dengan lebih efisienmerek pasta gigi untuk mendapatkan hasil akurasi yang komprehensif.

Kemampuan pencarian yang lebih terarah dan efisien ini memungkinkan peningkatan productivities dalam manajemen file dan akses terhadap berbagai informasi yang disimpan pada server FTP. Penggunaan algoritma DFS pada pencarian file FTP memiliki dampak yang signifikan dalam membantu pengguna untuk mengelola file dan informasi yang tersimpan pada server FTP dengan lebih efektif dan efisien.

2. METODE PENELITIAN

Metode penelitian yang dapat dilakukan terkait Pencarian File Menggunakan Algoritma DFS Pada FTP Server dengan menggunakan metode fuzzywuzzy adalah sebagai berikut:

Pengumpulan Data

Pengumpulan data yang dapat dilakukan terkait Pencarian File Menggunakan Algoritma DFS Pada FTP Server dengan menggunakan metode fuzzywuzzy adalah sebagai berikut:

1. Data mengenai server FTP yang akan digunakan, termasuk struktur direktori dan daftar file yang tersedia.
2. Data mengenai algoritma DFS dan implementasinya pada operasi FTP.
3. Data mengenai metode fuzzywuzzy dan pustaka FuzzyWuzzy yang digunakan untuk pencocokan string secara fuzzy.
4. Data mengenai analisis perbandingan antara algoritma DFS dengan metode fuzzywuzzy dan algoritma BFS dengan metode pencarian lainnya untuk menentukan kelebihan dan kekurangan masing-masing metode.

Data-data tersebut dapat dikumpulkan melalui penelitian literatur, pengujian pada server FTP yang digunakan, serta analisis keamanan dan perbandingan antara metode pencarian yang berbeda.

Implementasi Algoritma DFS

Implementasikan algoritma DFS untuk menavigasi struktur direktori pada server FTP dan mengambil daftar file yang tersedia. Algoritma Depth-First Search (DFS) merupakan algoritma pencarian yang digunakan untuk mengeksplorasi atau menelusuri semua node pada graf atau struktur data seperti pohon (tree) atau grafik (graph). DFS melakukan penelusuran sejauh mungkin pada sebuah cabang (branch) sebelum beralih ke cabang lainnya. Prinsipnya, DFS mencapai node yang paling dalam dari suatu cabang terlebih dahulu sebelum mundur dan mengeksplorasi cabang lainnya.

Penerapan Metode Fuzzywuzzy

Terapkan metode fuzzywuzzy untuk melakukan pencocokan string secara fuzzy terhadap nama file yang ada pada daftar file yang telah diambil. FuzzyWuzzy adalah suatu package di python, yang dapat digunakan untuk mencari kemiripan (*matching*) dari suatu string terhadapat suatu kumpulan string.

Evaluasi Hasil

Evaluasi hasil pencarian dengan membandingkan file yang ditemukan dengan kriteria pencarian yang telah ditentukan.

Implementasi DFS dan Fuzzywuzzy

Implementasikan algoritma DFS dan metode fuzzywuzzy pada server FTP yang digunakan dan evaluasi kinerjanya.

Analisis Perbandingan

Lakukan analisis perbandingan antara algoritma DFS dengan metode fuzzywuzzy dan algoritma BFS dengan metode pencarian lainnya untuk menentukan kelebihan dan kekurangan masing-masing metode.
Metode penelitian di atas dapat dilakukan dengan menggunakan bahasa pemrograman Python dan modul ftplib untuk operasi FTP, serta pustaka fuzzywuzzy untuk pencocokan string secara fuzzy.

3. HASIL DAN PEMBAHASAN

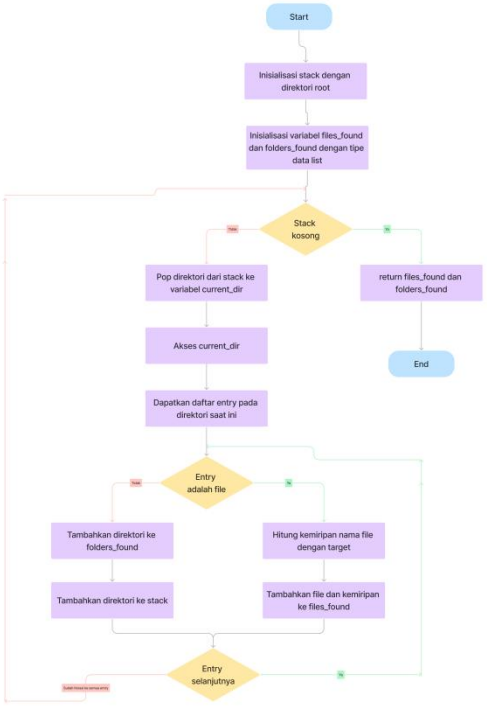
Main Flowchart

Berikut adalah main flowchart dari program yang akan dibuat.

Flowchart diatas menggambarkan proses pencarian informasi pada server FTP. Proses dimulai dengan memasukkan kredensial yang diperlukan untuk mengakses server FTP. Jika autentikasi berhasil, proses dilanjutkan dengan memeriksa apakah informasi yang dicari tersedia di server. Jika iya, hasil pencarian akan ditampilkan; jika tidak, proses berakhir.

Flowchart Pencarian DFS

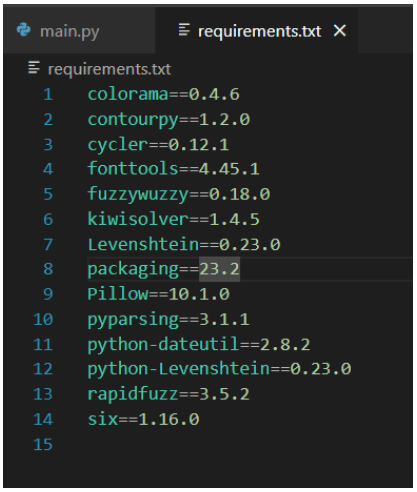
Berikut adalah flowchat Pencarian DFS menggunakan metode fuzzywuzzy.



Flowchart diatas menggambarkan proses pencarian file dan folder dalam struktur direktori. Proses dimulai dengan menginisialisasi stack dengan direktori root dan variabel untuk file dan folder yang ditemukan. Stack kemudian diperiksa apakah kosong; jika tidak, direktori saat ini dipop dari stack, dan entri-entri di dalamnya diproses untuk memeriksa apakah mereka cocok dengan kriteria file atau folder. Entri yang cocok ditambahkan ke daftar yang ditemukan, dan direktori ditambahkan kembali ke stack untuk diproses lebih lanjut. Jika stack kosong, itu mengembalikan file dan folder yang ditemukan.

Paket Pendukung

Buatlah sebuah file yang bernama requirements.txt yang nantinya akan berperan sebagai file yang berisi daftar paket atau pustaka yang diperlukan untuk mengerjakan suatu proyek yang semuanya dapat diinstal dengan file tersebut.



Berikut adalah beberapa penjelasan terkait paket yang ada pada file tersebut

1) Colorama==0.4.6
   Library untuk menyediakan penyesuaian tampilan teks (termasuk warna) di terminal Python.
2) Contourpy==1.2.0:
   Library untuk membuat visualisasi contour plot atau grafik kontur dari data dalam Python.
3) Cycler==0.12.1
   Library untuk memfasilitasi penciptaan sirkular dalam Python, umumnya digunakan dalam pengaturan properti siklus dalam matplotlib.
4) Fonttools==4.45.1:
   Library untuk memanipulasi font, terutama dalam format TrueType (TTF) dan OpenType (OTF).
5) Fuzzywuzzy==0.18.0
   Library yang menyediakan algoritma string matching untuk mencari kesamaan (matching) dan mencocokkan string dengan menggunakan pendekatan fuzzy.
6) Kiwisolver==1.4.5
   Library untuk menyelesaikan masalah pencarian nilai dalam masalah optimasi matematika, sering digunakan dalam pemecahan masalah layout pada matplotlib.
7) Levenshtein==0.23.0:
   Library yang menyediakan implementasi algoritma jarak Levenshtein untuk menghitung jarak antara dua string.
8) Packaging==23.2
   Library yang menyediakan alat-alat untuk bekerja dengan paket Python, pengelolaan dependensi, distribusi, dan lainnya.
9) Pillow==10.1.0:
   Library untuk pemrosesan gambar dalam Python, umumnya digunakan untuk membuka, memanipulasi, dan menyimpan berbagai format gambar.
10) Pyparsing==3.1.1
    Library untuk mempermudah parsing teks atau string dalam Python.
11) Python-dateutil==2.8.2
    Library untuk bekerja dengan tanggal dan waktu dalam Python dengan lebih nyaman, menyediakan fungsi-fungsi tambahan dari modul datetime bawaan Python.
12) Python-Levenshtein==0.23.0
    Perpanjangan dari library Levenshtein yang memberikan implementasi jarak Levenshtein dalam Python dengan performa yang lebih baik.
13) Rapidfuzz==3.5.2:
    Library lain untuk pencocokan string yang cepat, menawarkan algoritma pencocokan string dengan kinerja tinggi.
14) Six==1.16.0
    Library untuk memberikan kompatibilitas antara Python versi 2 dan 3.

3.1 Program

Berikut adalah program python untuk mencari file menggunakan algoritma DFS pada sebuh FTP Server.

```
import ftplib
import os
import time
from colorama import Fore, Style
from fuzzywuzzy import fuzz


def dir_name_parser(current_dir, sub_dir):
    if current_dir == "/":
        return f"{current_dir}{sub_dir}"
    else:
        return f"{current_dir}/{sub_dir}"

def get_top_matches(files_found):
    """
    Filter and sort the files based on their match scores.

    Args:
        files_found (list): A list of files with their
     corresponding match scores.

    Returns:
        list: The filtered and sorted list of files based on
     their match scores.
    """
    files_found = [file for file in files_found if file[1] >
     60]
    return sorted(files_found, key=lambda x: x[1],
     reverse=True)

def get_file_name(path):
    return os.path.basename(path)

def contains_name(path, name):
    return name in path
```

```
def dfs_search(directory, target):
    """
    Perform a depth-first search on the FTP server starting
     from the given directory,
    searching for files and folders that match the target.

    Args:
        directory (str): The starting directory for the search.
        target (str): The target file or folder to search for.

    Returns:
        tuple: A tuple containing two lists. The first list
     contains the found files
        along with their similarity scores, and the second
     list contains the found folders.
    """
    stack = [directory]
    files_found = []
    folders_found = []

    while stack:
        current_dir = stack.pop()
        try:
            for entry in ftp_server.mlsd(current_dir):
                if entry[1]["type"] == "file":
                    similarity = fuzz.ratio(entry[0], target)
                    files_found.append((dir_name_parser(curren
    t_dir, entry[0]), similarity))
                elif entry[1]["type"] == "dir":
                    folders_found.append(entry[0])
                    stack.append(dir_name_parser(current_dir,
    entry[0]))
        except PermissionError:
            continue
        except ftplib.error_perm:
            continue

    return files_found, folders_found

# file to find
file_to_find = input("Masukkan nama file yang ingin
    dicari: ")

# FTP server details
hostname = input("Masukkan Hostname FTP Server: ")
username = input("Masukkan Username: ")
password = input("Masukkan Password: ")

ftp_server = ftplib.FTP(hostname, username, password)
```

```
root_dir = ftp_server.pwd()
# print(f"Current directory: {root_dir}")
print("Mencari file...")

start_time = time.time()
result, folders = dfs_search(root_dir, file_to_find)
end_time = time.time()

search_time = end_time - start_time

top_matches = get_top_matches(result)

if not top_matches:
    print(f"File
     '{Fore.YELLOW}{file_to_find}{Style.RESET_ALL
     }' tidak ditemukan di server FTP.")
# print("Berikut file yang ditemukan berdasarkan
    kesamaan nama:")
else:
    print(f"{Fore.CYAN}Berikut file yang ditemukan
     berdasarkan kesamaan nama:{Style.RESET_ALL}")
    for path, similarity in top_matches:
        print(f"{Fore.YELLOW}{get_file_name(path)}{St
     yle.RESET_ALL} (Kesamaan:
     {Fore.GREEN}{similarity}%{Style.RESET_ALL})"
     )
        print(path)

    print("\n======================\n")

    # print("Berikut file yang namanya mengandung kata
     yang dicari:")
    print(f"{Fore.CYAN}Berikut file yang namanya
     mengandung kata yang
     dicari:{Style.RESET_ALL}")
    for path, similarity in result:
        if contains_name(path, file_to_find):
            print(f"{Fore.YELLOW}{get_file_name(path)}{
     Style.RESET_ALL}")
            print(path)
    print(f"\nTotal file:
     {Fore.YELLOW}{len(result)}{Style.RESET_ALL}"
     )
    print(f"Total folder:
     {Fore.YELLOW}{len(folders)}{Style.RESET_ALL
     }")
    print(f"Waktu pencarian:
     {Fore.YELLOW}{search_time}{Style.RESET_ALL
```

```
    } detik")
```

Program di atas melakukan pencarian file dan folder pada sebuah server FTP berdasarkan input nama file yang diberikan pengguna. Ini menggunakan modul **ftplib** untuk berinteraksi dengan server FTP, **os** untuk manipulasi path file, **time** untuk menghitung waktu pencarian, **colorama** untuk output berwarna di terminal, dan **fuzzywuzzy** untuk menghitung kesamaan antara nama file yang dicari dan yang ada di server. Berikut juga adalah program Tkinter untuk membuat antarmuka grafis pengguna (GUI).

```
import tkinter as tk
from tkinter import  scrolledtext
from ftp_search import FTPSearcher

class FTPSearchTool:
   def __init__(self, root):
      self.root = root
      root.title("FTP Search Tool")

      # Creating the labels and entries for each field
      self.create_label_and_entry("Alamat FTP Server",
  row=0)
      self.ftp_address = self.entry

      self.create_label_and_entry("Username", row=1)
      self.username = self.entry

      self.create_label_and_entry("Password", row=2,
  hide_text=True)
      self.password = self.entry

      self.create_label_and_entry("Nama file yang ingin
  dicari", row=3)
      self.filename = self.entry

      # Submit Button
      submit_button = tk.Button(root, text="Submit",
  command=self.submit_action)
      submit_button.grid(row=4, column=1, pady=10)

      # Results Display
      self.results_display =
  scrolledtext.ScrolledText(root, width=60, height=15)
      self.results_display.tag_config('info',
  foreground='blue')
      self.results_display.tag_config('warning',
  foreground='red')
      self.results_display.grid(row=5, column=0,
  columnspan=2, padx=10, pady=10)
```

```
   def create_label_and_entry(self, text, row,
  hide_text=False):
      label = tk.Label(self.root, text=text)
      label.grid(row=row, column=0, pady=5, padx=10,
  sticky="e")
      self.entry = tk.Entry(self.root, width=50, show="*"
  if hide_text else "")
      self.entry.grid(row=row, column=1, padx=10)

   def submit_action(self):
      # Clear previous results
      self.results_display.delete('1.0', tk.END)

      # Retrieving data from the entries
      ftp_address = self.ftp_address.get()
      username = self.username.get()
      password = self.password.get()
      filename = self.filename.get()

      # Searching for the file
      ftp_searcher = FTPSearcher(ftp_address, username,
  password)
      result = ftp_searcher.search_and_display(filename)

      # Display results
      top_matches = result["top_matches"]
      name_matches = result["name_matches"]
      search_time = result["search_time"]

      # Update the results display
      # self.results_display.insert(tk.END, f"Top
  Matches:\n{top_matches}\n\n")
      self.results_display.insert(tk.END, f"Top
  Matches:\n","info")
      if(len(top_matches) == 0):
  self.results_display.insert(tk.END, "Tidak ada file yang
  ditemukan\n","warning")
      for match in top_matches:
         self.results_display.insert(tk.END, f"{match[0]}
  (Kemiripan: {match[1]}%)\n")
      self.results_display.insert(tk.END, "\n")

      self.results_display.insert(tk.END, f"Name
  Matches:\n","info")
      if(len(name_matches) == 0):
  self.results_display.insert(tk.END, "Tidak ada file yang
  ditemukan\n","warning")
      for match in name_matches:
```

```
        self.results_display.insert(tk.END,
 f"{match[0]}\n")
    self.results_display.insert(tk.END, "\n")

    self.results_display.insert(tk.END, f"Search Time:
 {search_time} seconds\n")

# Creating the main window and running the application
root = tk.Tk()
app = FTPSearchTool(root)
root.mainloop()
```

Berikut adalah penjelasan singkat tentang setiap bagian dari program:

1. **Import Library dan Module:**
   - Mengimpor modul **tkinter** untuk membuat GUI.
   - Mengimpor kelas **scrolledtext** dari modul **tkinter** untuk membuat area teks dengan kemampuan pengguliran.
   - Mengimpor kelas **FTPSearcher** dari modul **ftp_search** untuk melakukan pencarian file di server FTP.
2. **Kelas FTPSearchTool:**
   - Konstruktor kelas ini digunakan untuk menginisialisasi antarmuka pengguna dan elemen-elemen GUI.
3. **Metode create_label_and_entry:**
   - Metode ini digunakan untuk membuat label dan elemen input entry untuk setiap parameter (Alamat FTP Server, Username, Password, Nama file yang ingin dicari).
4. **Metode submit_action:**
   - Metode ini dipanggil ketika tombol "Submit" ditekan.
   - Mengambil nilai dari entri (input) yang dimasukkan oleh pengguna.
   - Membuat objek **FTPSearcher** dan melakukan pencarian file menggunakan metode **search_and_display**.
   - Menampilkan hasil pencarian pada area teks yang dapat digulir.
5. **Tampilan GUI:**
   - Label dan entri untuk setiap parameter (Alamat FTP Server, Username, Password, Nama file yang ingin dicari).
   - Tombol "Submit" untuk memulai pencarian.
   - Area teks
   - yang dapat digulir untuk menampilkan hasil pencarian.
6. **Main Window dan Loop:**
   - Membuat instance dari kelas **Tk** sebagai jendela utama.
   - Membuat instance dari kelas **FTPSearchTool** dengan menggunakan jendela utama sebagai parameter.
   - Menjalankan loop utama dengan **root.mainloop()** untuk menjaga jendela agar tetap terbuka.

3.2 Alur Kerja Program

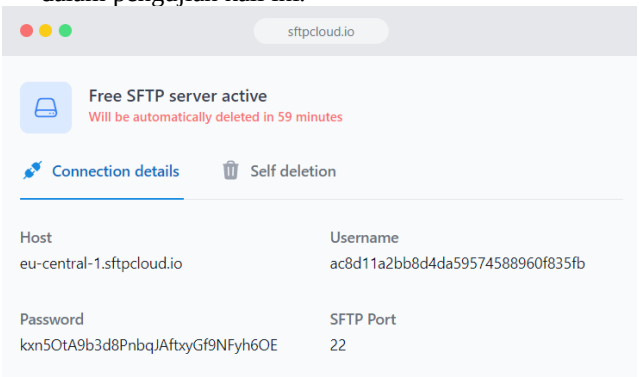
Berikut adalah alur kerja utama dari program ini:

1. Pengguna diminta untuk memasukkan nama file yang ingin dicari di server FTP, Hostname FTP Server yang dituju, username yang akan digunakan dan password yang akan digunakan.
2. Skrip akan terhubung ke server FTP yang ditentukan dengan kredensial yang di input.
3. Program akan melakukan pencarian file dan folder di server FTP menggunakan algoritma DFS (Depth-First Search) dimulai dari direktori awal.
4. Pencarian akan membandingkan setiap file/folder yang ditemukan dengan nama file yang dicari menggunakan perhitungan kesamaan string dengan bantuan dari fungsi **fuzz.ratio()** dari **fuzzywuzzy**.
5. Setelah pencarian selesai, program akan menampilkan hasil berupa file yang memiliki kesamaan nama tertinggi dengan file yang dicari, file yang namanya mengandung kata yang dicari, serta jumlah total file dan folder yang ditemukan beserta waktu yang diperlukan untuk pencarian.

Program ini memanfaatkan teknik fuzzy string matching (**fuzzywuzzy**) untuk menemukan file yang memiliki kesamaan nama dengan file yang dicari.

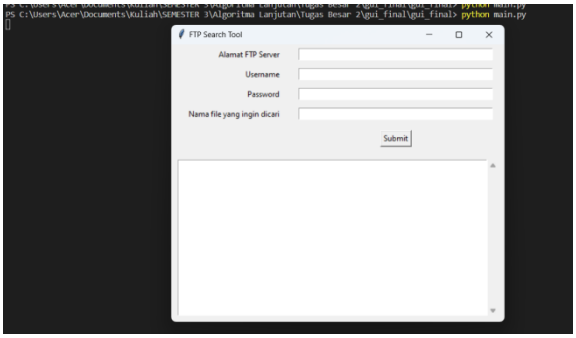
Hasil Program

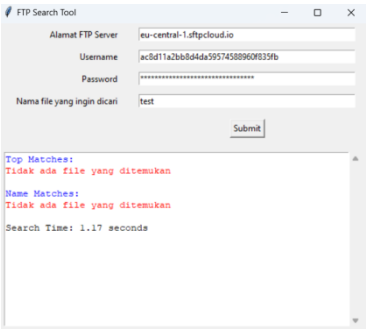
1. Berikut adalah Server FTP yang akan digunakan dalam pengujian kali ini.



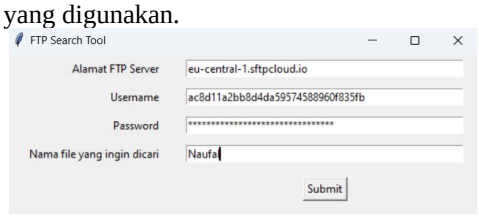
2. Berikut adalah beberapa file yang terdapa pada server FTP tersebut.

3. Jalankan program python.



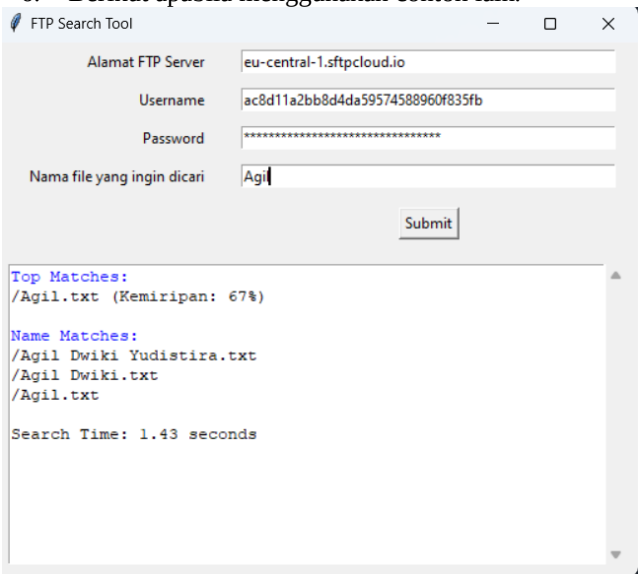


4. Masukkan nama file yang ingin di cari, hostname ftp server, username yang digunakan, lalu password yang digunakan.



5. Apabila sudah, maka program akan mencari file yang mengandung kesamaan dengan kata yang kita input, berikut adalah hasilnya.

6. Berikut apabila menggunakan contoh lain.



7. Dan berikut apabila program tidak menemukan file yang sesuai.

4. KESIMPULAN

Pencarian File dengan menggunakan Algoritma DFS (Depth-First Search) pada FTP Server dan menerapkan metode fuzzywuzzy memiliki beberapa kesimpulan penting:

1. **Efisiensi Pencarian:** Algoritma DFS digunakan untuk mencari file pada FTP Server. Pendekatan ini mengizinkan pencarian yang komprehensif dengan melintasi struktur direktori secara menyeluruh, meskipun dapat menjadi kurang efisien dalam kasus struktur direktori yang sangat besar.
2. **Akurasi Pencocokan:** Metode fuzzywuzzy memungkinkan pencocokan yang lebih fleksibel. Dalam konteks pencarian file, ini memperbolehkan penyesuaian terhadap kesalahan ketik, perbedaan minor dalam nama file, atau variasi dalam penulisan sehingga dapat menemukan hasil yang relevan meskipun ada perbedaan dalam penulisan atau ejaan.
3. **Ketepatan Hasil:** Kombinasi antara DFS dan fuzzywuzzy meningkatkan ketepatan hasil pencarian dengan mempertimbangkan kemiripan antara string yang dicari dengan nama file yang ada di server. Hal ini membantu dalam menemukan file yang diinginkan, bahkan jika ada perbedaan kecil dalam nama file atau kesalahan dalam kueri pencarian.
4. **Keterbatasan Kinerja:** Meskipun metode ini membantu meningkatkan ketepatan pencarian, penggunaan metode fuzzywuzzy dapat menambah beban komputasi pada server, terutama saat melakukan pencarian pada dataset yang besar. Ini dapat mempengaruhi kinerja server jika tidak dikelola dengan baik.
5. **Kemampuan Adaptasi:** Metode ini memungkinkan adaptasi yang baik terhadap perubahan atau variasi dalam nama file yang umumnya terjadi di lingkungan FTP Server. Kemampuannya untuk mengatasi kesalahan atau variasi dalam penamaan file membuatnya menjadi solusi yang lebih dinamis.

Kesimpulannya, penggunaan Algoritma DFS dengan metode fuzzywuzzy pada FTP Server memberikan keunggulan dalam menemukan file dengan lebih baik, meskipun perlu mempertimbangkan keseimbangan antara ketepatan pencarian dan kinerja sistem.

Daftar Pustaka

[1] Alabi, T. H. (2023, September 11). *Python Requirements.txt – How to Create and Pip Install Requirements.txt in Python.* Retrieved from

www.freecodecamp.org: https://www.freecodecamp.org/news/python-requirementstxt-explained/

[2] Bachmann, M. (2023, October 11). *pypi.org*. Retrieved from python-Levenshtein 0.23.0: https://pypi.org/project/python-Levenshtein/

[3] Bachmann, M. (2023, November 3). *pypi.org*. Retrieved from rapidfuzz 3.5.2: https://pypi.org/project/rapidfuzz/

[4] Caswell, T. A. (2023, October 7). *pypi.org*. Retrieved from cycler 0.12.1: https://pypi.org/project/cycler/

[5] Clark, J. A. (2023, October 15). *pypi.org*. Retrieved from Pillow 10.1.0: https://pypi.org/project/Pillow/

[6] Cohen, A. (2020, February 14). *pypi.org*. Retrieved from fuzzywuzzy 0.18.0: https://pypi.org/project/fuzzywuzzy/

[7] Cynthia Kustanto, R. M. (n.d.). Penerapan Algoritma Breadth-first Search dan Depth-first Search. *Laboratorium Ilmu dan Rekayasa Komputasi*.

[8] Hartley, J. (2022, October 25). *pypi.org*. Retrieved from colorama 0.4.6: https://pypi.org/project/colorama/ianthomas23. (2023, November 3). *pypi.org*. Retrieved from contourpy 1.2.0: https://pypi.org/project/contourpy/

[9] J. Postel, J. R. (1985). RFC 959: File Transfer Protocol. *IETF (Internet Engineering Task Force)*.

[10] McGuire, P. (2023, July 30). *pypi.org*. Retrieved from pyparsing 3.1.1: https://pypi.org/project/pyparsing/

[11] Niemeyer, G. (2021, July 14). *python.org*. Retrieved from python-dateutil 2.8.2: https://pypi.org/project/python-dateutil/

[12] Peterson, B. (2021, May 5). *pypi.org*. Retrieved from six 1.16.0: https://pypi.org/project/six/

[13] Pratama, L. G. (2020, December 4). *medium.com*. Retrieved from Contoh implementasi "FuzzyWuzzy" pada pandas python: https://medium.com/belajar-nulis/contoh-implementasi-fuzzywuzzy-pada-pandas-python-62de6898a4d0#:~:text=FuzzyWuzzy%20adalah%20suatu%20package%20di,menggunakan%20pendekatan%20%E2%80%9CLevenshtein%20Distance%E2%80%9D.

[14] Rossum, J. v. (2023, December 19). *pypi.org*. Retrieved from fonttools 4.47.0: https://pypi.org/project/fonttools/

[15] Sedgewick, R. &. (2011). Algorithms (4th ed.). In Addison-Wesley, *Algorithms (4th ed.)*.

[16] Stufft, D. (2023, October 1). *pypi.org*. Retrieved from packaging 23.2: https://pypi.org/project/packaging/

[17] The Nucleic Development Team. (2023, August 24). *pypi.org*. Retrieved from kiwisolver 1.4.5: https://pypi.org/project/kiwisolver/

[18] van Rossum, G. D. (2009). *The Python Language Reference Manual.* Python Software Foundation.