Pencarian Akun Instagram dengan Nama menggunakan Breadth First Search

Muhammad Fauzan Azhim - 13522153
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13522153@std.stei.itb.ac.id

***Abstrak*—Instagram adalah platform media social dimana pengguna dapat mengunggah foto atau video. Instagram juga seringkali menjadi tempat mencari teman baru. Fitur dari Instagram adalah dapat berteman dengan orang lain. Saat kita berteman dengan seseorang, kita dapat melihat dengan siapa orang tersebut berteman juga. Hal ini dapat membuat pohon pertemanan yang memungkinkan kita untuk memperluas jaringan pertemanan. Pencarian akun Instagram melalui teman juga memungkinkan kita untuk mencari orang yang punya hubungannya dengan kita, sehingga akan memudahkan menggunakan koneksi pertemanan yang ada untuk berkenalan. Teknik yang paling tepat untuk pencarian ini salah satunya adalah Breadth First Search.**

Keywords—Instagram, Breadth First Search, Pertemanan, Pencarian, Struktur data pohon, Sosial Media.

I. PENDAHULUAN

Instagram adalah platform media sosial terbesar ke-4 di dunia. Dimana 37.74% dari 5.3 milyar pengguna internet didunia memakai Instagram. Pengguna dari Instagram sendiri paling banyak berkisar 71% dari umur 18 sampai 24 tahun.

Di Indonesia, Instagram juga menjadi media sosial yang dimiliki oleh banyak remaja. Kebanyakan media sosial ini digunakan untuk menghabiskan waktu dengan menonton video-video yang ada. Ada juga yang berpenghasilan di Instagram sebagai selebgram yang melakukan promosi terhadap produk tertentu. Namun yang utama adalah penggunaan Instagram sebagai tempat menyimpan dan berbagi berbagai momen baik sendiri maupun Bersama.

Orang-orang yang bisa melihat foto-foto atau video yang dibagikan dibagi menjadi 3 tipe sesuai dengan tipe dari akun Instagram itu sendiri. Tipe yang pertama ada Public dimana semua orang bahkan yang tidak kita kenal dapat melihat foto-foto atau video yang kita unggah. Yang kedua ada private dimana hanya yang telah diterima sebagai pengikut dari akun tersebut yang dapat melihat media yang dibagikan. Yang terakhir adalah close friends dimana hanya orang-orang tertentu yang telah di berikan akses untuk dapat melihat beberapa media yang dibagikan.

Pencarian akun dalam media sosial Instagram sendiri sudah ada secara bawaan. Namun pencarian ini tidak selalu memperhitungkan kemungkinan kedekatan pengguna dengan kata kunci yang dicari. Terkadang orang yang kita cari tidak selalu mempunyai hubungan secara tidak langsung terhadap kita. Oleh karena itu pencarian akun Instagram melalui teman yang di follow adalah salah satu cara untuk mendapatkan akun yang relevan dengan status pertemanan kita saat ini.

Bentuk dari hubungan pertemanan antar akun ini sangat lah mirip dengan struktur data pohon. Dalam struktur data pohon terdapat simpul yang bertindak sebagai *parent* dari banyak simpul lainnya, dan anak dari simpul itu dapat memiliki anak simpul lagi. Pemetaan pertemanan pada pohon berarti adalah simpul sebagai akun Instagram target dan sisi sebagai hubungan pertemanan yang ada.

Beberapa Teknik pencarian data pada struktur data tree ada *Breadth First Search* dan *Depth First Search, Depth Limited Search, Iterative Deepening Search.* Untuk pencarian pertemanan tentunya kita ingin yang paling dekat dengan kita, oleh karena itu salah satu pencarian yang paling bagus adalah *Breadth First Search*, *Depth Limited Search,* dan *Iterative Deepening Search.*

Breadth First Search pastinya akan mengembalikan hasil pertemanan terdekat dan akan mencari keseluruhan dari teman yang di ikuti oleh teman kita. *Depth Limited Search,* dan *Iterative Deepening Search* mencari sampai ke limit kedalaman pertemanan yang kita inginkan.

Selagi Algoritma *Depth First Search* tidaklah efisien karena semakin dalam pertemanan yang kita cari akan semakin sulit untuk berhubungan dengan orang tersebut.

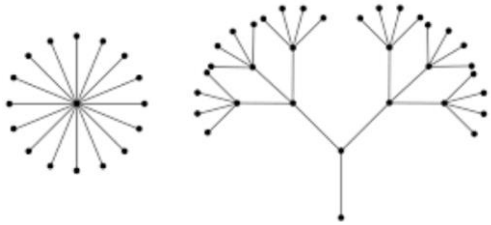
Pada Makalah kali ini Penulis akan menggunakan algoritma Breadth First Search dalam pencarian akun Instagram menggunakan fitur Following yang ada pada media sosial Instagram. Diharapkan dengan adanya hal ini dapat membantu orang-orang yang ingin memperluas lingkaran pertemanannya, ingin mencari orang yang mungkin dikenal namun adalah teman dari teman seseorang, dan mungkin mencari hubungan antara seseorang dan orang lainnya.

II. LANDASAN TEORI

A. Struktur data pohon

Pohon adalah graf tak-berarah terhubung yang tidak mengandung sirkuit. Syarat suatu graf adalah pohon ada 3:

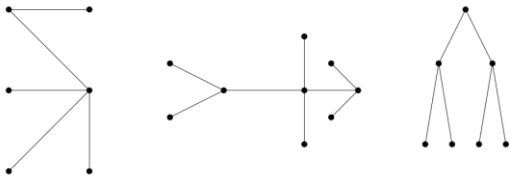
1. graf tidak berarah,
2. harus terhubung,
3. dan tidak memiliki sirkuit (siklus).



Gambar 1. Contoh Pohon
(sumber : Website Rinaldi Munir) diakses pada 8 Juni 2024

Hutan (forest) adalah:

- kumpulan pohon yang saling lepas, atau
- graf tidak terhubung yang tidak mengandung sirkuit. Setiap komponen di dalam graf terhubung tersebut adalah pohon.



Gambar 2. Contoh Hutan yang terdiri dari 3 Pohon
(sumber : Website Rinaldi Munir) diakses pada 8 Juni 2024

Sifat-sifat (properti) pohon

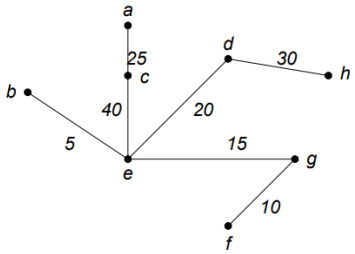
Teorema. Misalkan G = (V, E) adalah graf tak-berarah sederhana dan jumlah simpulnya n. Maka, semua pernyataan di bawah ini adalah ekivalen:

1. G adalah pohon.
2. Setiap pasang simpul di dalam G terhubung dengan lintasan tunggal.
3. G terhubung dan memiliki m = n – 1 buah sisi.
4. G tidak mengandung sirkuit dan memiliki m = n – 1 buah sisi.
5. G tidak mengandung sirkuit dan penambahan satu sisi pada graf akan membuat hanya satu sirkuit.
6. G terhubung dan semua sisinya adalah jembatan.

Teorema di atas dapat dikatakan sebagai definisi lain dari pohon.

Pohon Merentang Minimum

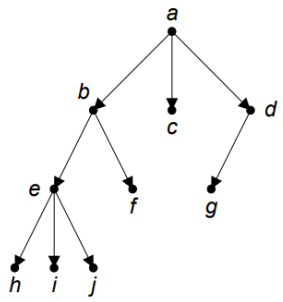
Graf terhubung-berbobot mungkin mempunyai lebih dari 1 pohon merentang. Pohon merentang yang berbobot minimum –dinamakan pohon merentang minimum (minimum spanning tree).



Gambar 3. Contoh Pohon Merentang minimum
(sumber : Website Rinaldi Munir) diakses pada 8 Juni 2024

Pohon berakar (rooted tree)

Pohon yang satu buah simpulnya diperlakukan sebagai akar dan sisi-sisinya diberi arah sehingga menjadi graf berarah dinamakan pohon berakar (rooted tree).



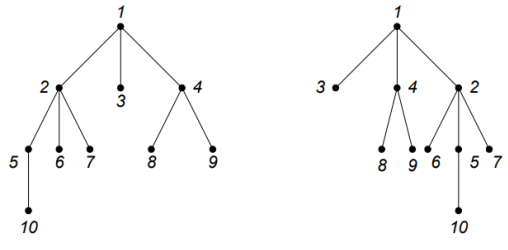
Gambar 4. Contoh Pohon Berakar
(sumber : Website Rinaldi Munir) diakses pada 8 Juni 2024

Terminologi pada Pohon Berakar

1. Anak (child atau children) dan Orangtua (parent)
2. Lintasan (path)
3. Saudara kandung (sibling)
4. Upapohon (subtree)
5. Derajat (degree)
6. Daun (leaf)
7. Simpul Dalam (internal nodes)
8. Aras (level) atau Tingkat
9. Tinggi (height) atau Kedalaman (depth)

Pohon Terurut (ordered tree)

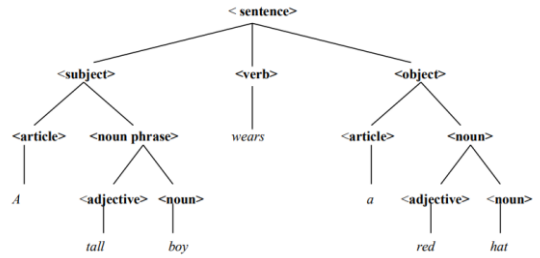
Pohon berakar yang urutan anak-anaknya penting disebut pohon terurut (ordered tree).



Gambar 5. Pohon A dan Pohon B adalah 2 pohon yang berbeda
(sumber : Website Rinaldi Munir) diakses pada 8 Juni 2024

Pohon n-ary

Pohon berakar yang setiap simpul cabangnya mempunyai paling banyak n buah anak disebut pohon n-ary.

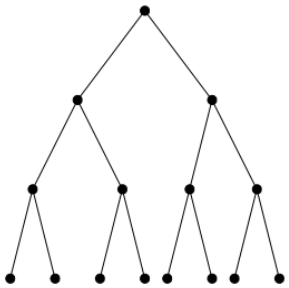


Gambar Pohon parsing dari kalimat *A tall boy wears a red hat*

Gambar 6. Pohon parsing dari kalimat A tall boy wears a red hat
(sumber : Website Rinaldi Munir) diakses pada 8 Juni 2024

Pohon Biner (binary tree)

Adalah pohon n-ary dengan n = 2. Pohon yang paling penting karena banyak aplikasinya. Setiap simpul di dalam pohon biner mempunyai paling banyak 2 buah anak. Dibedakan antara anak kiri (left child) dan anak kanan (right child). Karena ada perbedaan urutan anak, maka pohon biner adalah pohon terurut.



Gambar 7. Pohon Binary Penuh
(sumber : Website Rinaldi Munir) diakses pada 8 Juni 2024

B. Algoritma Breadth First Search

BFS atau Breadth-First Search adalah salah satu algoritma yang digunakan dalam penelusuran atau pencarian pada struktur data berbentuk graf yaitu struktur data yang memiliki simpul-simpul atau node dan simpul-simpul tersebut terhubung oleh sisi atau edge.

Algoritma Breadth First Search melakukan pencarian dengan cara traversal dari simpul utama. Pencarian dilakukan secara bertahap level ke level. Pencarian akan menelusuri seluruh simpul pada level saat ini sebelum melanjut ke level selanjutnya.

Struktur data lainnya yang dibutuhkan untuk membuat Breadth First Search adalah Queue, Map dan Set. Queue digunakan untuk mengatur antrian simpul yang akan dicari. Map digunakan untuk menyimpan hubungan antara simpul. Set digunakan untuk menyimpan simpul mana saja yang sudah di kunjungi.

Proses penelusuran tree menggunakan algoritma Breadth First Search sebagai berikut.

1. Inisialisasi untuk simpul awal.
2. Masukkan simpul awal ke dalam antrian.
3. Tandai simpul awal sebagai simpul yang sudah dikunjungi.
4. Lakukan perulangan sampai antrian kosong.
    a. Ambil simpul terdepan dari antrian
    b. Kunjungi semua anak simpul tersebut
    c. Masukkan semua anak simpul tersebut ke antrian

Keuntungan dari BFS adalah algoritma ini akan menemukan jalur terpendek yang pertama kali ditemukan. Selain itu, BFS juga dapat digunakan untuk menemukan semua simpul yang dapat dijangkau dari simpul awal, tanpa melewatkan satu pun. Oleh karena hal tersebut BFS sangat cocok digunakan untuk menyelesaikan permasalahan mencari rute terpendek dari suatu graf.

C. Sosial Media

Media sosial adalah teknologi digital yang memungkinkan pengguna untuk saling berinteraksi melalui jaringan dan komunitas secara online yang memungkinkan manusia untuk saling berinteraksi tanpa dibatasi ruang dan waktu. Konten yang disebarkan dalam sosial media beragam. Mulai dari foto, video, text, musik, dll.

Media sosial juga dapat digunakan untuk menyebarkan pendapat. Isi dari pendapat ini dapat disebarkan lagi oleh orang lain. Hubungan ini terus berlanjut hingga tidak ada yang menyebarkan lagi. Hal ini membuat media sosial menjadi tempat untuk mengemukakan pendapat dan mencari dukungan maupun opini.

III. IMPLEMENTASI DAN UJI COBA

Untuk mengimplementasikan pencarian akun instagram menggunakan metode Breath First Search penulis akan menggunakan bahasa Python.

Alasan penggunaan bahasa python karena dibutuhkannya library scraping untuk melakukan penelusuran, mempunyai library opencv yang memudahkan pembuatan image recognition, mempunyai kemudahan dalam membuat threading. Program akan dibuat dengan input nama untuk prototype testing terlebih dahulu sebelum dilanjutkan dengan input foto.

A. Desain Program

Program akan terdiri dari beberapa kelas dalam python.

1. BFS (Breadth First Search)

    Kelas BFS berisikan fungsi-fungsi untuk menjalankan pencarian secara BFS terhadap akun yang ingin dicari. Implementasi dari kelas ini telah ditambahkan multiprocessing agar pencarian dapat lebih cepat. Karena kelas BFS sendiri mempunyai fungsi yang panjang maka penulis hanya akan mencantumkan pseudocode.

```
Class BFS:
    Initialize with queue, visited, target_username,
target_photo, using_name, using_photo
        Set instance variables

    Function fetch_photo(url):
        Create async session
        Fetch photo from URL
        Return photo data

    Function compare_photos_async(profiles):
        Initialize tasks list
        For each profile in profiles:
            If profile is private, continue
            Add fetch_photo task to tasks list
        Fetch all photos asynchronously
        Initialize results list
        For each photo in photos:
            Decode photo to image
            Compare image with target_photo
            Add comparison result to results list
        Return results

    Function compare_names(profiles):
        Create ThreadPoolExecutor
        Compare profile names with target_username
using multiprocessing
        Return results

    Function bfs():
        Set batch_size
        Copy queue to local variable
        While queue is not empty:
            Get batch of profiles from queue
            Add batch to visited set
            Initialize name_results and photo_results
            If using_name, compare names in batch
            If using_photo, compare photos in batch
asynchronously
            For each profile in batch:
                If name or photo comparison is
successful:
                    Set found to True
                    Add profile to result list
        Return found

    Function get_queue():
        Return queue

    Function get_found():
        Return found

    Function get_result():
        Return result
```

2. ImageCompare

Kelas ImageCompare berisikan implementasi dari pencocokan input foto dengan foto akun yang diperiksa. Kelas ini menggunakan library face_recognition yang ada pada python.

```
import face_recognition


class ImageCompare:
    """Compares two images to check if the faces
match."""

    def __init__(self, img1, img2):
        self.img1 = img1
        self.img2 = img2
```

```
    def is_face_match(self):
        """Checks if the face in the input image
matches the face in the target image."""
        input_face_encodings =
self.get_face_encodings(self.img1)
        target_face_encodings =
self.get_face_encodings(self.img2)

        if not input_face_encodings or not
target_face_encodings:
            return False  # No faces found in one of
the images

        for input_face_encoding in
input_face_encodings:
            if
self.compare_faces(target_face_encodings,
input_face_encoding):
                return True

        return False

    def get_face_encodings(self, image):
        """Returns the face encodings for the faces
in the image."""
        face_locations =
face_recognition.face_locations(image)
        return
face_recognition.face_encodings(image,
face_locations)

    def compare_faces(self, known_face_encodings,
face_encoding_to_check):
        """Compares a list of face encodings against
a candidate encoding to see if they match."""
        matches =
face_recognition.compare_faces(known_face_encodings,
face_encoding_to_check)
```

3. StringCompare

Kelas StringCompare berisikan implementasi dari pencocokan string yang di input dengan string yang diperiksa. Untuk pencocokan patternya penulis sendiri tidak mengimplementasikan algoritma yang sudah diajarkan seperti KMP dan Boyer-Moore. Penulis mengunakan fungsi bawaan python yang mampu mengetahui apakah suatu string adalah subset dari string target.

```
class StringCompare:
    """
    Compare two strings
    """
    def __init__(self, s1, s2):
        self.s1 = s1
        self.s2 = s2

    def compare(self):
        """
        Compare two strings
        :return: bool, True if the strings are
equal, False otherwise
        """
        return self.s1 == self.s2

    def substring(self):
        """
        Is s2 a substring of s1
        :return: bool, True if s2 is a substring
of s1, False otherwise
        """
        is_substring = self.s2 in self.s1
        return is_substring
```

4. Instagram

Kelas Instagram berisikan implementasi dari scraping media sosial instagram. Implementasi dari kelas ini menggunakan library *Instaloader*. Di kelas ini program akan melakukan login akun instagram, lalu mengambil data dari following suatu akun.

```
import instaloader


class Instagram:
    """Instagram class to interact with Instagram API using
Instaloader library"""

    def __init__(self, username, password):
        self.username = username
        self.password = password
        self.L = instaloader.Instaloader()

    def login(self) -> tuple[bool, str]:
        """
        Login to Instagram account using provided
credentials
        :return: tuple of login status and message
        """
        try:
            self.L.login(self.username, self.password)
            print("Login successful")
            return True, "Login successful"

        except
instaloader.exceptions.BadCredentialsException:
            print("Login failed")
            return False, f"Login failed for
{self.username}, bad credentials"

        except instaloader.exceptions.ConnectionException:
            print("Login failed")
            return False, f"Login failed for
{self.username}, connection error"

        except Exception as e:
            print("Login failed")
            return False, f"Login failed for
{self.username}, {e}"

    def get_profile(self, target_username) ->
instaloader.Profile | None:
        """
        Get profile metadata of a target username
        :param target_username: str, username of the target
profile
        :return: instaloader.Profile object or None
        """
        try:
            profile =
instaloader.Profile.from_username(self.L.context,
target_username)
            return profile

        except
instaloader.exceptions.ProfileNotExistsException:
            print(f"Profile {target_username} does not
exist")
            return None

        except instaloader.exceptions.ConnectionException:
            print("Connection error")
            return None

        except Exception as e:
            print(f"Error: {e}")
            return None
```

```
def get_followers(self, target_username) ->
list[instaloader.Profile] | None:
        """
        Get followers of a target profile
        :param target_username: str, username of the
target profile
        :return: list of instaloader.Profile objects
or None
        """
        try:
            profile =
self.get_profile(target_username)
            if profile:
                followers = [follower for follower in
profile.get_followers()]
                return followers

            return None

        except Exception as e:
            print(f"Error: {e}")
            return None

    def get_following(self, target_username) ->
list[instaloader.Profile] | None:
        """
        Get following of a target profile
        :param target_username: str, username of the
target profile
        :return: list of instaloader.Profile objects
or None
        """
        try:
            profile =
self.get_profile(target_username)
            if profile:
                following = [following for following
in profile.get_followees()]
                return following

            return None
```

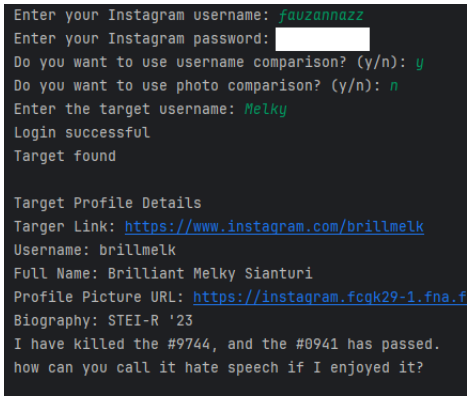
5. Main

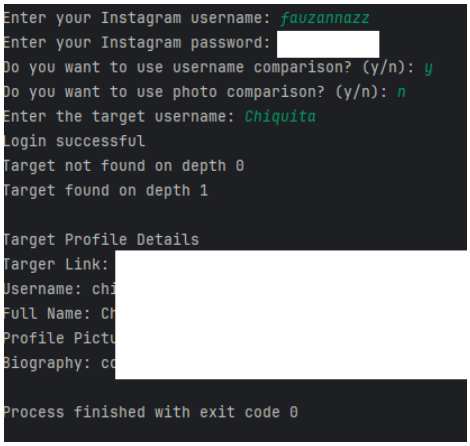
Kelas Main adalah kelas utama yang menyatukan keseluruh kelas yang ada dan memuat alur dari program sekaligus sebagai entrypoint dari program. Program akan menanyakan input yang diinginkan apa dan meminta input sesuai yang sebelumnya ditanyakan. Setelah itu program akan langsung melakukan pencarian.

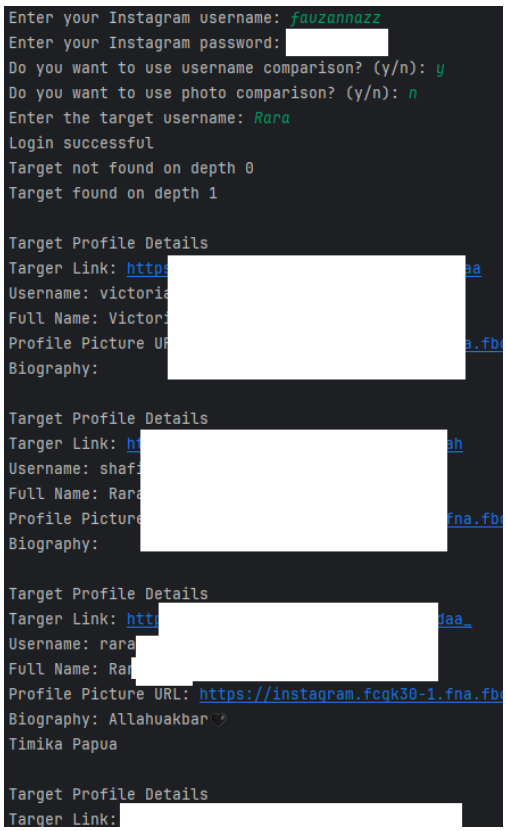
```
import Instagram
import BFS
import cv2
import time


def main():
    try:
        # Get user input
        username = input('Enter your Instagram username:
')
        password = input('Enter your Instagram password:
')

        while True:  # Loop until valid input is
provided
            try:
                using_name = input('Do you want to use
username comparison? (y/n): ')
                using_photo = input('Do you want to use
photo comparison? (y/n): ')

                # Convert input to boolean
                if using_name.lower() == 'y':
                    using_name = True
                else:
                    using_name = False

                # Convert input to boolean
                if using_photo.lower() == 'y':
                    using_photo = True
                else:
                    using_photo = False

                break
            except Exception as e:
                print(f"Error: {e}")
```

```
        # Comparator Check
        if using_name:
            target_username = input('Enter the
target username: ')
        else:
            target_username = None

        if using_photo:
            target_profile_path = input('Enter
the target profile photo path: ')
            target_profile_img =
cv2.imread(target_profile_path)
        else:
            target_profile_img = None

        if not using_name and not using_photo:
            print("Please select at least one
comparison method")
            return

        # Login to Instagram
        insta = Instagram.Instagram(username,
password)
        login_status, login_message =
insta.login()

        if not login_status:  # If login failed
            print(login_message)
            return

        followers = insta.get_followers(username)
        if not followers:  # If no followers
found
            print(f"No followers found for
{username}")

        # Initialize BFS first depth
        visited = set()
        bfs = BFS.BFS(
            queue=followers,
            visited=visited,
            target_username=target_username,
            target_photo=target_profile_img,
            using_name=using_name,
            using_photo=using_photo
        )
        found = bfs.bfs()
        while not found:
            print("Target not found")
            # Continue BFS until target is found
using the followers of last depth queue
            next_queue = []
            for last_queue in bfs.get_queue():
                next_followers =
insta.get_followers(last_queue.username)
                if next_followers:

next_queue.extend(next_followers)
                    time.sleep(0.5)  # Sleep for
0.5 seconds to avoid rate limit

            bfs.queue = next_queue
            found = bfs.bfs()

        print("Target found")

        # Show the target profile details
        for profile in bfs.get_result():
            print(f"\nTarget Profile Details")
            print(f"Targer Link:
https://www.instagram.com/{profile.username}")
            print(f"Username:
{profile.username}")
            print(f"Full Name:
{profile.full_name}")
            print(f"Profile Picture URL:
{profile.profile_pic_url}")
            print(f"Biography:
{profile.biography}")

        return

    except Exception as e:
        print(f"Error: {e}")
        return


if __name__ == '__main__':
    main()
```

B. *Uji Coba*

- 

- 

- 

C. *Hasil Uji Coba*

Program telah bekerja dengan baik sesuai dengan yang diinginkan. Program menemukan banyak akun instagram yang sebenarnya berhubungan dekat dengan pengguna sesuai dengan parameter nama yang diinginkan. Beberapa akun yang ditemukan tidak bisa didapatkan dari fungsi pencarian bawaan yang dimiliki instagram.

Ada sebuah masalah dalam prototype program saat ini. Masalah itu adalah adanya rate limit yang diterapkan pada API Instagram. Hal ini membuat perlambatan karena harus memenuhi tuntutat Instagram dalah scraping akun. Hal ini dapat diatasi dengan menerapkan *caching*. Namun menggunakan caching akan membuat hasil pencarian kadang tidak terlalu akurat jika tidak sering di update hasil dari pencarian sebelumnya.

Saat ini program hanya mengimplementasikan nama sebagai perbandingan input. Program dapat dikembangkan berdasarkan beberapa input lainnya seperti informasi yang bisa kita dapatkan dari bio akun.

IV. KESIMPULAN

Pada pencarian bawaan instagram ada juga yang direkomendasikan sama seperti pada program karena mengikuti teman yang sama. Namun kelemahan pada pencarian instagram ini dia tidak dapat menemukan teman yang ada pada kedalaman 2 atau teman dari temannya teman pengguna. Walau mulai dari kedalaman 2, opsi pembangkitan simpul yang akan ditelusuri semakin kecil karena akan terbagi menjadi dua tipe akun. Akun *private* tidak akan bisa ditelusuri lagi karena tidak berteman dengan akun utama, namun akun *public* tetap bisa ditelusuri.

REFERENCES

[1] Munir, Rinaldi. “Breadth/Depth First Search (BFS/DFS) Bagian 1.” *Homepage Rinaldi Munir Sekolah Teknik Elektro dan Informatika (STEI) ITB*, Rinaldi Munir, 2021, https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/BFS-DFS-2021-Bag1-2024.pdf. Accessed 12 06 2024.

[2] Munir, Rinaldi. “Breadth/Depth First Search (BFS/DFS) Bagian 2.” *Homepage Rinaldi Munir Sekolah Teknik Elektro dan Informatika (STEI) ITB*, Rinaldi Munir, 2021, informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag2.pdf. Accessed 12 06 2024.

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, 12 Juni 2024

Muhammad Fauzan Azhim, 13522153