LAPORAN TUGAS KECIL 1

IF2211 STRATEGI ALGORITMA

SEMESTER II TAHUN 2023/2024

“Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force”

Disusun oleh:

Ivan Hendrawan Tan

13522111

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA (STEI)

INSTITUT TEKNOLOGI BANDUNG

DAFTAR ISI

I. DESKRIPSI ALGORITMA BRUTE FORCE

Algoritma brute force adalah sebuah cara untuk mencari sebuah solusi secara langsung dan umumnya tidak kompleks. Algoritma ini mencoba seluruh kemungkinan solusi hingga menemukan solusi yang paling optimal atau hingga seluruh kemungkinan telah dicoba. Kekurangan yang biasanya dialami oleh algoritma brute force adalah algoritma ini tidak efisien terutama jika diberi persoalan yang berukuran besar karena akan sangat memakan waktu untuk memeriksa seluruh kemungkinan yang ada, tetapi solusi yang diberikan akan terjamin karena semua kemungkinan diperiksa.

Algoritma brute force digunakan untuk menemukan solusi dari permainan kecil dari game Cyberpunk 2077 Breach Protocol, solusi yang diinginkan merupakan yang paling optimal dari setiap kombinasi matriks, sekuens, dan ukuran buffer. Solusi yang paling optimal berupa poin maksimal yang didapat dari setiap sekuens yang terpenuhi dengan penggunaan buffer seminim mungkin.

Pada program ini, pencarian solusi dilakukan dengan memanggil fungsi *find_optimal_path* yang menghasilkan sebuah jalan yang paling optimal. Di dalam fungsi *find_optimal_path* terdapat fungsi *check_sequences_in_path* dan fungsi *path_generator*. Fungsi *check_sequences_in_path* berfungsi untuk memeriksa apakah ada sebuah jalan yang menghasilkan reward yang lebih besar dari sebelumnya atau ada sebuah jalan yang lebih pendek dengan reward yang sama besarnya. Fungsi *path_generator* digunakan untuk menghasilkan sebuah jalan yang baru dan mengecek apakah jalan yang akan dilalui sudah pernah dilewati atau belum, fungsi *path_generator* ini akan digunakan secara rekursif agar dapat mengecek setiap jalan yang memungkinkan.

II. SOURCE CODE PROGRAM (PYTHON)

File: main.py

```
import time
import random

def file_input(lines):
    def is_valid_matrix(rows, cols):
        row_count = 0
        for line in lines[2:2 + rows]:
            element = line.strip().split()
            if (len(element) != cols) or (any(len(element) != 2
for element in element)):
                return False
            row_count += 1

        if (row_count != rows):
            return False

        return True

    sequences = {}
    sequences_reward = []

    file_lines_count = len(lines)
    buffer_size = int(lines[0].strip())
    split = lines[1].strip().split()
    matrix_width = int(split[0])
    matrix_height = int(split[1])

    if (is_valid_matrix(matrix_height, matrix_width)):
        matrix = [line.strip().split() for line in lines[2:2 +
matrix_height]]
    else:
        return 0, 0, 0, 0, 0

    for j in range(4 + matrix_height, file_lines_count, 2):
        temp = lines[j].strip().split()
        sequences_reward.append(temp)

    counter = 0
    for i in range(3 + matrix_height, file_lines_count, 2):
        temp = lines[i].strip().split()
        seq_tup = tuple(temp)
        reward = int(sequences_reward[counter][0])
        sequences[seq_tup] = reward
        counter += 1
    return buffer_size, matrix_width, matrix_height, matrix,
sequences
```

```
def cli_input():
    sequences = {}
    token_count = int(input("Masukkan jumlah token: "))
    token = input("Masukkan token: ")
    token_split = token.split()

    flag = True
    while (flag):
        if len(token_split) > token_count:
            print("Token yang diinput terlalu banyak")
        elif len(token_split) < token_count:
            print("Token yang diinput terlalu sedikit")
        elif not all(len(pair) == 2 for pair in token_split):
            print("Ada token yang panjangnya bukan 2")
        else:
            flag = False
        if (flag):
            token = input("Masukkan token: ")
            token_split = token.split()

    buffer_size = int(input("Masukkan ukuran buffer: "))

    matrix_size = input("Masukkan ukuran matriks: ")
    matrix_size_split = matrix_size.strip().split()
    while (len(matrix_size_split) != 2):
        print("Matriks harus berdimensi 2")
        matrix_size = input("Masukkan ukuran matriks: ")
        matrix_size_split = matrix_size.strip().split()

    matrix_width = int(matrix_size_split[0])
    matrix_height = int(matrix_size_split[1])

    sequences_count = int(input("Masukkan jumlah sekuens: "))
    sequences_max_size = int(input("Masukkan ukuran maksimal dari
sekuens: "))

    matrix = []
    for _ in range(matrix_height):
        row = []
        for _ in range(matrix_width):
            randomizer = random.randint(0, token_count-1)
            random_token = token_split[randomizer]
            row.append(random_token)
        matrix.append(row)

    for _ in range(sequences_count):
        temp = []
        seq_size = random.randint(2, sequences_max_size)
```

```
        for _ in range(seq_size):
            seq_idx = random.randint(0, len(token_split) - 1)
            temp.append(token_split[seq_idx])
        seq_tup = tuple(temp)
        reward = random.randint(10, 50)
        sequences[seq_tup] = reward
    print("\nMatriks: ")
    for list in matrix:
        print(' '.join(list))
    for x, y in sequences.items():
        print(f"Sekuens dan hadiahnya:\n{x}\n{y}")
    return buffer_size, matrix_width, matrix_height, matrix,
sequences


def find_optimal_path(matrix, matrix_width, matrix_height,
sequences, buffer_size):
    max_reward = 0
    optimal_path = []
    optimal_coords = []

    def check_sequences_in_path(path):
        total_reward = 0
        path_str = ' '.join(path)
        for seq, reward in sequences.items():
            seq_str = ' '.join(seq)
            if seq_str in path_str:
                total_reward += reward
        return total_reward

    def path_generator(row, col, path, coords, visited, step,
direction):
        nonlocal max_reward, optimal_path, optimal_coords
        if step >= buffer_size or visited[row][col]:
            return

        visited[row][col] = True
        new_path = path + [matrix[row][col]]
        new_coords = coords + [(row, col)]

        reward = check_sequences_in_path(new_path)

        if (reward > max_reward) or (reward == max_reward and
len(new_path) < len(optimal_path)):
            max_reward = reward
            optimal_path = new_path
            optimal_coords = new_coords

        if (direction == 'horizontal'):
```

```
            for next_col in range(matrix_width):
                if not(visited[row][next_col]) and (next_col !=
col):
                    path_generator(row, next_col, new_path,
new_coords, visited, step + 1, 'vertical')
        else:
            for next_row in range(matrix_height):
                if not(visited[next_row][col]) and (next_row !=
row):
                    path_generator(next_row, col, new_path,
new_coords, visited, step + 1, 'horizontal')

        visited[row][col] = False

    for col in range(matrix_width):
        visited = [[False for _ in range(matrix_width)] for _ in
range(matrix_height)]
        path_generator(0, col, [], [], visited, 0, 'vertical')

    return optimal_path, optimal_coords, max_reward

while (True):
    pilihan = int(input("Masukkan 1 untuk membaca input dari file
dan 2 untuk membaca input dari CLI: "))
    if (pilihan == 1):
        while (True):
            file_name = input("Masukkan nama file beserta .txt: ")
            updated_file_name = "./test/" + file_name
            try:
                with open(updated_file_name, 'r') as fp:
                    lines = fp.readlines()
                break
            except FileNotFoundError:
                print("File tidak ditemukan.")
        buffer_size, matrix_width, matrix_height, matrix,
sequences = file_input(lines)
        break
    elif (pilihan == 2):
        buffer_size, matrix_width, matrix_height, matrix,
sequences = cli_input()
        break
    else:
        print("Masukkan hanya '1' atau '2'")

start_time = time.time()
optimal_path, optimal_coords, max_reward =
find_optimal_path(matrix, matrix_width, matrix_height, sequences,
buffer_size)
end_time = time.time()
```

```
print(f"\n{max_reward}")
for item in optimal_path:
    print(item, end=' ')
print('\0')
update_optimal_coord = [(y + 1, x + 1) for x, y in optimal_coords]
for x, y in update_optimal_coord:
    print(f"{x}, {y}")
proc_time = round(end_time-start_time, 2) * 1000
print(f"\n{int(proc_time)} ms\n")

flag = True
while (True):
    yes_no = str(input("Apakah ingin menyimpan solusi? (y/n) "))
    if (yes_no == 'y'):
        while (flag):
            save_file_name = str(input("Masukkan nama file beserta
.txt: "))
            updated_save_file_name = "./test/" + save_file_name
            if not(updated_save_file_name.endswith('.txt')):
                print("File harus berakhiran .txt")
            else:
                with open(updated_save_file_name, 'w') as file:
                    file.write(f"{str(max_reward)}\n")
                    file.write(' '.join(optimal_path))
                    file.write("\n")
                    for x, y in update_optimal_coord:
                        file.write(f"{x}, {y}\n")
                    file.write(f"\n{str(int(proc_time))} ms")
                flag = False
        break
    elif (yes_no == 'n'):
        break
    else:
        print("Masukkan hanya 'y' atau 'n'")
```

III. INPUT DAN OUTPUT

1.

```
PS C:\Users\tfpri\Documents\ITB\Sem 4\Stima\Tucil1-13522111> python -u "c:\Use
rs\tfpri\Documents\ITB\Sem 4\Stima\Tucil1-13522111\src\main.py"
Masukkan 1 untuk membaca input dari file dan 2 untuk membaca input dari CLI: 1

Masukkan nama file beserta .txt: tc1.txt

50
7A BD 7A BD 1C BD 55
1, 1
1, 4
3, 4
3, 5
6, 5
6, 3
1, 3

90 ms

Apakah ingin menyimpan solusi? (y/n) y
Masukkan nama file beserta .txt: ans1.txt
```

2.

```
PS C:\Users\tfpri\Documents\ITB\Sem 4\Stima\Tucil1-13522111> python -u
"c:\Users\tfpri\Documents\ITB\Sem 4\Stima\Tucil1-13522111\src\main.py"
Masukkan 1 untuk membaca input dari file dan 2 untuk membaca input dari
 CLI: 1
Masukkan nama file beserta .txt: tc2.txt

0


0 ms

Apakah ingin menyimpan solusi? (y/n) y
Masukkan nama file beserta .txt: ans2.txt
```

3.

```
PS C:\Users\tfpri\Documents\ITB\Sem 4\Stima\Tucil1-13522111> python -u "c:\Use
rs\tfpri\Documents\ITB\Sem 4\Stima\Tucil1-13522111\src\main.py"
Masukkan 1 untuk membaca input dari file dan 2 untuk membaca input dari CLI: 1

Masukkan nama file beserta .txt: tc3.txt

60
7A BD BD 55 7A BD
1, 1
1, 5
3, 5
3, 6
6, 6
6, 3

20 ms

Apakah ingin menyimpan solusi? (y/n) y
Masukkan nama file beserta .txt: ans3.txt
```

4.

```
PS C:\Users\tfpri\Documents\ITB\Sem 4\Stima\Tucil1-13522111> python -u "c:\Use
rs\tfpri\Documents\ITB\Sem 4\Stima\Tucil1-13522111\src\main.py"
Masukkan 1 untuk membaca input dari file dan 2 untuk membaca input dari CLI: 1

Masukkan nama file beserta .txt: tc4.txt

75
1C 55 7A 1C E9 E9 BD
5, 1
5, 4
3, 4
3, 2
5, 2
5, 3
6, 3

440 ms

Apakah ingin menyimpan solusi? (y/n) y
Masukkan nama file beserta .txt: ans4.txt
```

5.

```
PS C:\Users\tfpri\Documents\ITB\Sem 4\Stima\Tucil1-13522111> python -u "c:\Use
rs\tfpri\Documents\ITB\Sem 4\Stima\Tucil1-13522111\src\main.py"
Masukkan 1 untuk membaca input dari file dan 2 untuk membaca input dari CLI: 2

Masukkan jumlah token: 5
Masukkan token: BD 1C 7A 55 E9
Masukkan ukuran buffer: 7
Masukkan ukuran matriks: 6 6
Masukkan jumlah sekuens: 3
Masukkan ukuran maksimal dari sekuens: 4

Matriks:
1C 1C 1C 55 E9 BD
BD 1C 1C 1C E9 E9
55 55 7A BD E9 1C
E9 1C 7A BD 1C 7A
55 55 BD 1C 7A 7A
E9 55 1C E9 1C 55
Sekuens dan hadiahnya:
('BD', '1C', 'E9', '1C')
38
Sekuens dan hadiahnya:
('BD', 'E9', 'E9')
44
Sekuens dan hadiahnya:
('1C', '7A')
31

82
BD 1C E9 1C BD E9 E9
6, 1
6, 3
5, 3
5, 4
4, 4
4, 6
1, 6

90 ms

Apakah ingin menyimpan solusi? (y/n) y
Masukkan nama file beserta .txt: ans5.txt
```

6.

```
PS C:\Users\tfpri\Documents\ITB\Sem 4\Stima\Tucil1-13522111> python -u "c:\User
s\tfpri\Documents\ITB\Sem 4\Stima\Tucil1-13522111\src\main.py"
Masukkan 1 untuk membaca input dari file dan 2 untuk membaca input dari CLI: 2
Masukkan jumlah token: 2
Masukkan token: AB CD EF
Token yang diinput terlalu banyak
Masukkan token:
```

7.

```
PS C:\Users\tfpri\Documents\ITB\Sem 4\Stima\Tucil1-13522111> python -u "c:\User
s\tfpri\Documents\ITB\Sem 4\Stima\Tucil1-13522111\src\main.py"
Masukkan 1 untuk membaca input dari file dan 2 untuk membaca input dari CLI: 2
Masukkan jumlah token: 3
Masukkan token: AB CD
Token yang diinput terlalu sedikit
Masukkan token:
```

8.

```
PS C:\Users\tfpri\Documents\ITB\Sem 4\Stima\Tucil1-13522111> python -u "c:\User
s\tfpri\Documents\ITB\Sem 4\Stima\Tucil1-13522111\src\main.py"
Masukkan 1 untuk membaca input dari file dan 2 untuk membaca input dari CLI: 2
Masukkan jumlah token: 3
Masukkan token: AB CDE FG
Ada token yang panjangnya bukan 2
Masukkan token:
```

9.

```
PS C:\Users\tfpri\Documents\ITB\Sem 4\Stima\Tucil1-13522111> python -u "c:\User
s\tfpri\Documents\ITB\Sem 4\Stima\Tucil1-13522111\src\main.py"
Masukkan 1 untuk membaca input dari file dan 2 untuk membaca input dari CLI: 2
Masukkan jumlah token: 3
Masukkan token: AB CD EF
Masukkan ukuran buffer: 5
Masukkan ukuran matriks: 2
Matriks harus berdimensi 2
Masukkan ukuran matriks:
```

IV. CHECKLIST

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan	✓	
2. Program berhasil dijalankan	✓	
3. Program dapat membaca masukan berkas .txt	✓	
4. Program dapat menghasilkan masukan secara acak	✓	
5. Solusi yang diberikan program optimal	✓	
6. Program dapat menyimpan solusi dalam berkas .txt	✓	
7. Program memiliki GUI		✓

V. REPOSITORY

Link repository GitHub: https://github.com/Bodleh/Tucil1-13522111.git