LAPORAN TUGAS KECIL 1
MATA KULIAH STRATEGI ALGORITMA (IF2211)

Dibuat Oleh:

Juan Alfred Widjaya - 13522073

INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2024

DAFTAR ISI

I. PERMASALAHAN

Breach Protocol dalam permainan Cyberpunk 2077 adalah sebuah minigame yang mensimulasikan proses meretas jaringan lokal dari ICE (*Intrusion Countermeasures Electronics*) dalam dunia permainan tersebut. Dalam permainan ini, pemain akan dihadapkan dengan beberapa komponen utama seperti:

- Token, merupakan representasi dari karakter alfanumerik seperti E9, BD, dan 55. Token ini nantinya akan digunakan untuk menyusun urutan kode dalam permainan.
- Matriks, merupakan kumpulan token-token yang tersusun dalam sebuah *grid*. Pemain akan memilih token dari matriks ini untuk membentuk urutan kode
- Sekuens, merupakan rangkaian token (minimal dua token) yang harus dicocokkan oleh pemain dalam permainan. Setiap sekuens memiliki bobot hadiah yang berbeda-beda.
- Buffer, merupakan jumlah maksimal token yang dapat disusun secara sekuensial oleh pemain. Pemain akan berusaha untuk mengisi buffer ini dengan sekuens-sekuens yang sesuai.

Aturan permainan *Breach Protocol* antara lain mengharuskan pemain untuk bergerak dengan pola horizontal dan vertikal secara bergantian. Pemain akan memulai dengan memilih satu token dari baris paling atas matriks, lalu mencoba mencocokkan sekuens pada token-token yang berada di dalam buffer. Setiap token dalam buffer dapat digunakan untuk mencocokkan lebih dari satu sekuens. Proses pencocokan akan terus berlanjut sampai semua sekuens berhasil dicocokkan atau buffer telah penuh. Dengan mencocokkan sekuens-sekuens dengan benar, pemain akan memperoleh hadiah sesuai dengan bobot hadiah yang telah ditetapkan. Tujuan dari permainan ini adalah untuk mencari isi dari buffer paling optimal yang mendapatkan hadiah paling besar.

II. PEMECAHAN MASALAH

Untuk mencari solusi optimal dengan hadiah terbesar, digunakan algoritma *brute force* yang mengimplementasikan pendekatan konsep *depth-first search* (DFS). Algoritma ini melakukan pencarian secara rekursif dengan *backtracking* untuk mengeksplorasi seluruh kemungkinan rute di dalam matriks. Pencarian dimulai dari setiap titik di baris atas dan berlanjut dengan gerakan maju melalui matriks dengan aturan telah ditentukan.

Langkah pertama dimulai dengan gerakan vertikal ke bawah, diikuti dengan percobaan gerakan horizontal. Algoritma akan mencoba bergerak ke kiri terlebih dahulu, kemudian ke kanan jika tidak berhasil. Setelah itu, algoritma akan kembali bergerak secara vertikal, baik ke atas maupun ke bawah, dengan upaya pertama ke atas. Selama

proses pencarian, algoritma akan memeriksa apakah langkah selanjutnya akan mengunjungi sel yang sudah dikunjungi sebelumnya dan melacak jumlah langkah yang telah dilewati. Jika jumlah langkah mencapai batas maksimal yang ditetapkan (sama dengan jumlah buffer) atau sudah tidak ada sel yang dapat dikunjungi, algoritma akan melakukan *backtracking* untuk mencari rute lainnya.

Pada setiap langkah, algoritma akan menyimpan rute yang telah dilewati beserta koordinat sel yang telah dikunjungi. Algoritma juga akan mencari hadiah dari rute tersebut dengan memeriksa apakah sekuens berhadiah terdapat pada rute yang telah dilewati, serta menghitung total hadiah yang diperoleh dari rute tersebut. Jika total hadiah dari rute tersebut lebih besar dari hadiah terbesar yang telah tercapai sebelumnya (*default*-nya adalah 0), algoritma akan menyimpan rute tersebut dan meng-*update* data hadiah terbesar yang telah ditemukan.

Proses pencarian akan terus berlanjut hingga seluruh kemungkinan jalur dari titik - titik awal pada baris pertama telah dieksplorasi sepenuhnya. Dengan menggunakan strategi rekursif dan *backtracking* yang optimal, algoritma mampu mengidentifikasi jalur yang memenuhi batasan gerakan yang ditetapkan dan mengoptimalkan hadiah yang diperoleh. Pada akhirnya, algoritma akan memberikan solusi terbaik dengan menghasilkan jalur optimal yang mencapai hadiah maksimum.

Optimalisasi algoritma *brute force* dapat dicapai dengan memperkenalkan kondisi penghentian prematur selama proses pencarian. Ini dilakukan dengan menghentikan algoritma segera setelah ditemukan sebuah rute yang mendapatkan hadiah maksimum yang mungkin. Pendekatan ini didasarkan pada pemahaman bahwa, jika sebuah rute sudah mencapai hadiah maksimal yang bisa diperoleh, maka tidak ada lagi kebutuhan untuk melanjutkan pengecekan terhadap rute-rute lainnya. Dengan demikian, efisiensi algoritma dapat ditingkatkan tanpa mengorbankan konsep dasar dari pendekatan *brute force* itu sendiri.

III. SOURCE CODE PROGRAM

Program ditulis dalam bahasa pemrograman Python. Program terdiri dari dua file, yakni main.py dan utils.py. File main.py berisi algoritma dan file utama jalannya program. File utils.py berisi fungsi - fungsi bantuan untuk program.

File: main.py

```
import time
from utils import read_file, user_input, save_file, print_welcome

def search_path(matrix, row, col, path, path_coordinate, n, visited, can_horizontal,
sequences , reward_info):
    if n < 0:
```

```
        return
    if visited[row][col]:
        return

    # # For optimized version (still bruteforce), stop when current path's reward is
the maximum reward (Best possible reward)
    # if reward_info['max_reward'] == reward_info['max'] :
    #     return

    visited[row][col] = True
    path.append(matrix[row][col])
    path_coordinate.append([col + 1, row + 1])

    current_path_str = " ".join(path)
    reward = sum(seq[1] for seq in sequences if seq[0] in current_path_str)

    if reward > reward_info['max_reward']:
        reward_info['max_reward'] = reward
        reward_info['max_reward_seq'] = current_path_str
        reward_info['seq_coordinate'] = path_coordinate.copy()

    if n > 0:
        if can_horizontal:
            new_col = col - 1
            while new_col >= 0 and not visited[row][new_col]:
                search_path(matrix, row, new_col, path, path_coordinate, n - 1,
visited, False, sequences, reward_info)
                new_col -= 1

            new_col = col + 1
            while new_col < len(matrix[0]) and not visited[row][new_col]:
                search_path(matrix, row, new_col, path, path_coordinate, n - 1,
visited, False, sequences, reward_info)
                new_col += 1
        else:
            new_row = row - 1
            while new_row >= 0 and not visited[new_row][col]:
                search_path(matrix, new_row, col, path, path_coordinate, n - 1,
visited, True, sequences, reward_info)
                new_row -= 1

            new_row = row + 1
            while new_row < len(matrix) and not visited[new_row][col]:
                search_path(matrix, new_row, col, path, path_coordinate, n - 1,
```

```
visited, True, sequences, reward_info)
                new_row += 1

    visited[row][col] = False
    path.pop()
    path_coordinate.pop()

def find_sequences(matrix, n, sequences):
    visited = [[False for _ in range(len(matrix[0]))] for _ in range(len(matrix))]
    max_r = sum(seq[1] for seq in sequences if seq[1] ⩾ 0)
    reward_info = {'max': max_r,'max_reward': 0, 'max_reward_seq': matrix[0][0],
'seq_coordinate': [[1, 1]]}

    for col in range(len(matrix[0])):
        search_path(matrix, 0, col, [], [], n, visited, False, sequences, reward_info)

    return reward_info

print_welcome()
while True :
    choice = input("Input from file? (y/n): ").lower()
    if choice ≠ 'y' and choice ≠ 'n' :
        print("\nInvalid choice.\n")
    else :
        if choice == 'y' :
            buffer_length, matrix_size, matrix, num_sequence, sequences = read_file()
        else :
            buffer_length, matrix_size, matrix, num_sequence, sequences = user_input()
        break

print(f"\nMatrix Size: {matrix_size[0]}x{matrix_size[1]}")
print("Matrix:")
for row in matrix:
    print(" ".join(row))
print("\nBuffer Length:", buffer_length)
print("\nNumber of Sequences:", num_sequence)
x = 1
for row in sequences :
    print(f"Sequence {x} : {row[0]} (Reward: {row[1]})")
    x += 1

start = time.time()
result = find_sequences(matrix, buffer_length - 1, sequences)
duration = round((time.time() - start) * 1000)
```

```
print("\nMax Reward Solution ")
print(f"Reward: {result['max_reward']}")
print(f"Solution: {result['max_reward_seq']}")
print(f"Solution's Token Coordinate: ")
for cdn in result['seq_coordinate']:
    print(f"{cdn[0]}, {cdn[1]}")
print(f"\n{duration} ms\n")

while True :
    choice = input("Do you want to save solution (y/n): ").lower()
    if choice =/= 'y' and choice =/= 'n' :
        print("\nInvalid choice.\n")
    else :
        if choice == 'y' :
            save_file(result)
        else :
            print("\nProgram has ended.")
        break
```

File: utils.py

```
import random

def print_welcome():
    print("""
   _____  ____  _ __       ________ _____
  / ____|/ __ \| |\ \     / /  ____|  __ \
 | (___ | |  | | | \ \   / /| |__  | |__) |
  \___ \| |  | | |  \ \/ / |  __| |  _  /
  ____) | |__| | |___\  /  | |____| | \ \
 |_____/ \____/|______\/   |______|_|  \_\ """)
    print("Welcome to Cyberpunk 2077 Breach Protocol Solver!")
    print("Happy Solving!\n")

def is_alphanumeric(token):
    alfanum = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9',
               'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M',
               'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']
    if len(token) =/= 2 :
        return False
    if token[0] in alfanum and token[1] in alfanum :
        return True
```

```
    return False

def input_valid_int(msg, min):
    while True :
        try :
            integer = int(input(f"\n{msg}"))
            if integer < min :
                print(f"\nInvalid, input must be integer greater than {min - 1}.")
            else :
                return integer
        except :
            print(f"\nInvalid, input must be integer greater than {min - 1}.")

def input_valid_an(msg):
    while True :
        try :
            token = input(f"\n{msg}").upper()
            if not is_alphanumeric(token):
                print("\nInvalid, input must be alphanumeric.")
            else :
                return token
        except :
            print("\nInvalid, input must be alphanumeric.")

def generate_matrix(matrix_size, tokens):
    matrix = [[0 for _ in range(matrix_size[0])] for _ in range(matrix_size[1])]
    for i in range(matrix_size[1]):
        for j in range(matrix_size[0]):
            matrix[i][j] = random.choice(tokens)
    return matrix

def generate_sequences(num, max_length, tokens):
    sequences = []
    list_of_seq = []
    for _ in range(num):
        length = random.randint(2, max_length)
        while True:
            seq = " ".join([random.choice(tokens) for _ in range(length)])
            if seq not in list_of_seq :
                list_of_seq.append(seq)
                break
        reward = random.randint(0, 50)
        sequences.append([seq, reward])
    return sequences
```

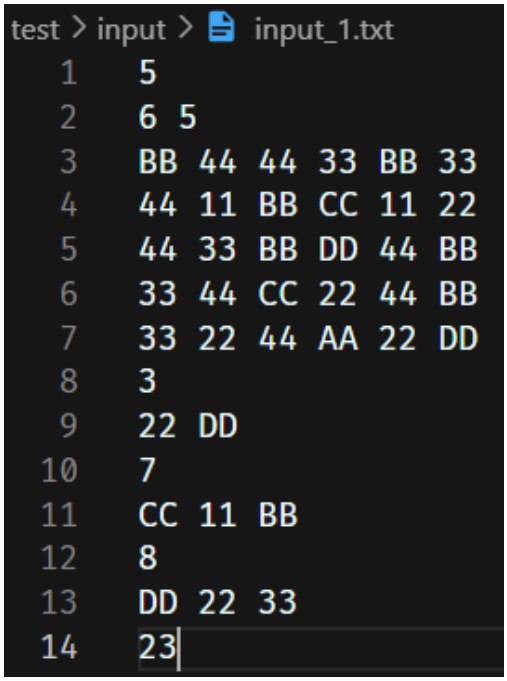
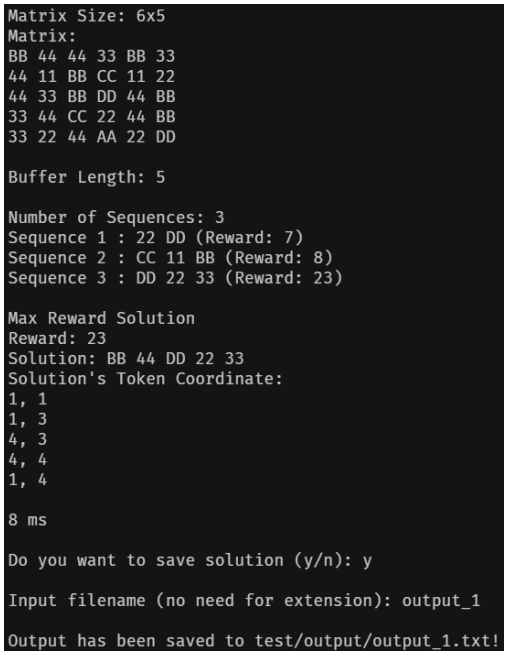
```
def user_input():
    n_unique_token = input_valid_int("Number of unique tokens: ", 1)
    tokens = []
    for i in range(n_unique_token) :
        token = input_valid_an(f"Input alphanumeric token no.{i + 1} (A1, 3E, etc.): ")
        tokens.append(token)
    buffer_length = input_valid_int("Input buffer length: ", 1)
    matrix_col = input_valid_int("Input matrix column size: ", 1)
    matrix_row = input_valid_int("Input matrix row size: ", 2)
    matrix_size = (matrix_col, matrix_row)
    matrix = generate_matrix(matrix_size, tokens)
    num_sequence = input_valid_int("Input number of sequences: ", 1)
    max_sequence_length = input_valid_int("Input maximum sequence's length: ", 2)
    sequences = generate_sequences(num_sequence, max_sequence_length, tokens)

    return buffer_length, matrix_size, matrix, num_sequence, sequences

def read_file():
    while True :
        filename = input("\nInput .txt filename (no need for extension): ")
        file_path = f"../test/input/{filename}.txt"
        try :
            with open(file_path, 'r') as file:
                lines = file.readlines()

                buffer_length = int(lines[0].strip())
                matrix_size = list(map(int, lines[1].strip().split()))
                if len(matrix_size) ≠ 2 :
                    raise Exception

                matrix = []
                for i in range(2, 2 + matrix_size[1]):
                    row = lines[i].strip().split()
                    if len(row) ≠ matrix_size[0] :
                        raise Exception
                    if not all(is_alphanumeric(x) for x in row) :
                        raise Exception
                    matrix.append(row)

                num_sequence = int(lines[2 + matrix_size[1]])
                start = 3 + matrix_size[1]
                sequences = []
```

```
                temp = []
                isSequence = True
                for i in range(start, start + (num_sequence * 2)):
                    if isSequence :
                        line = lines[i].strip().split()
                        if not all(is_alphanumeric(x) for x in line) :
                            raise Exception
                        temp.append(" ".join(line))
                        isSequence = False
                    else :
                        temp.append(int(lines[i].strip()))
                        sequences.append(temp)
                        temp = []
                        isSequence = True

                return buffer_length, matrix_size, matrix, num_sequence, sequences
        except :
            print("\nFile not found or wrong file input format.")

def save_file(reward_info):
    filename = input("\nInput filename (no need for extension): ")
    file_path = f"../test/output/{filename}.txt"

    output_str = ""
    output_str += str(reward_info['max_reward']) + '\n'
    output_str += reward_info['max_reward_seq'] + '\n'
    for cdn in reward_info['seq_coordinate']:
        output_str += f"{cdn[0]}, {cdn[1]}\n"
    output_str = output_str[:-1]

    with open(file_path, 'w') as file:
        file.write(output_str)

    print(f"\nOutput has been saved to test/output/{filename}.txt!")
```

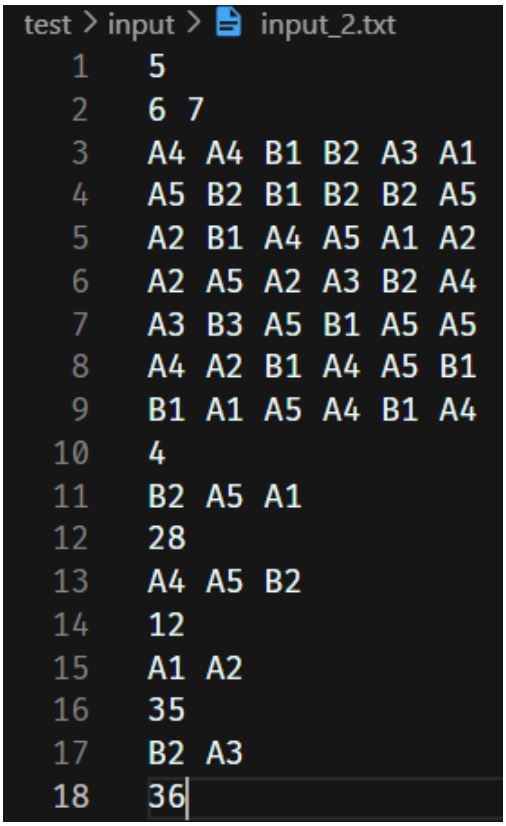
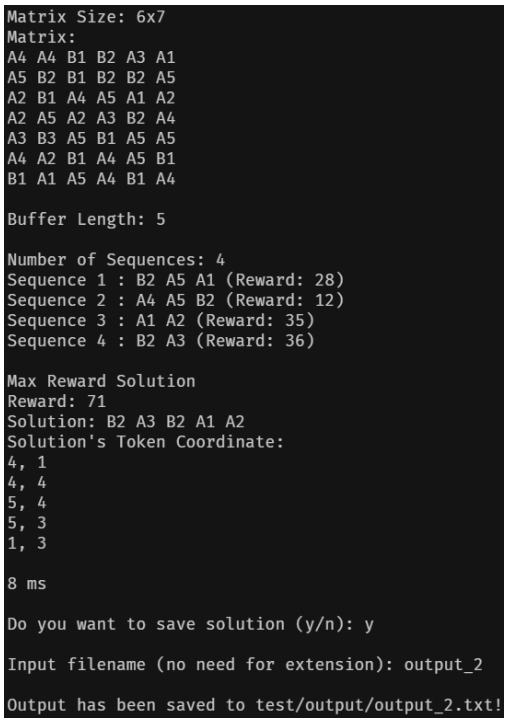
IV. UJI KASUS

Seluruh data untuk uji kasus terdapat pada folder test. Pada folder test, terdapat folder input yang berisi file-file txt yang digunakan untuk pengujian kasus dan folder output yang berisi file-file txt dari hasil pengujian.

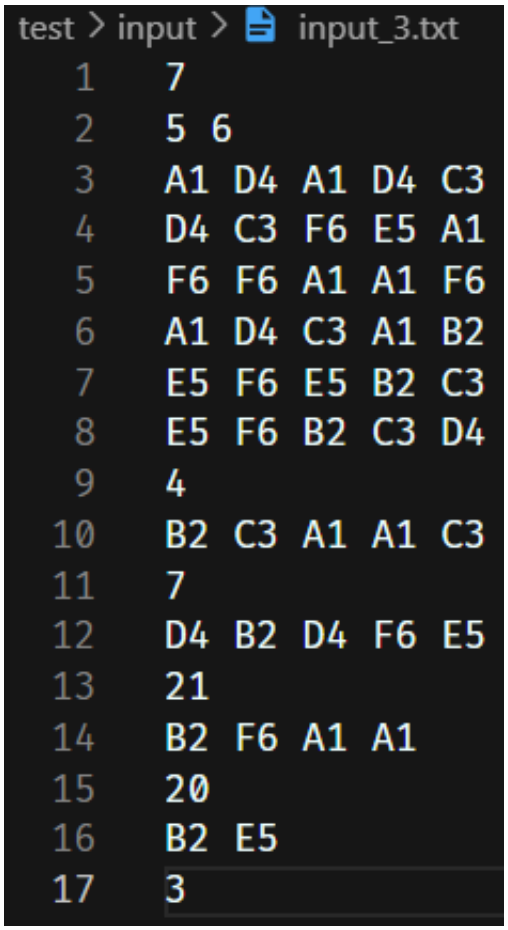
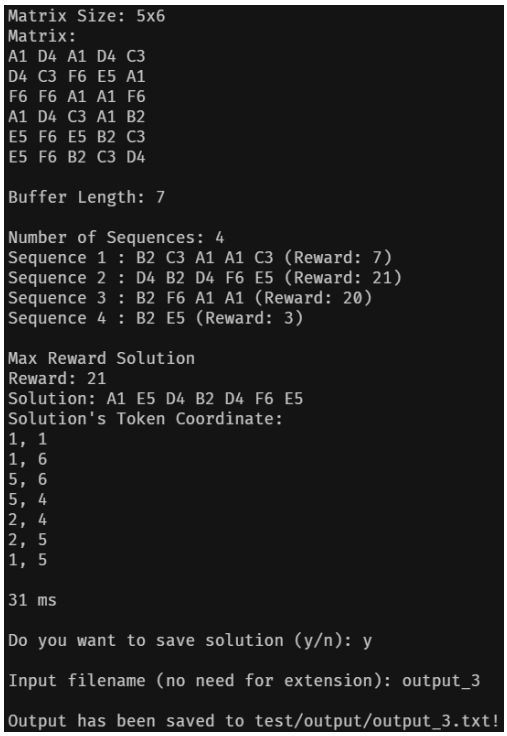
a. Kasus 1

Input: input_1.txt	Output: output_1.txt
	

b. Kasus 2

Input: input_2.txt	Output: output_2.txt
	

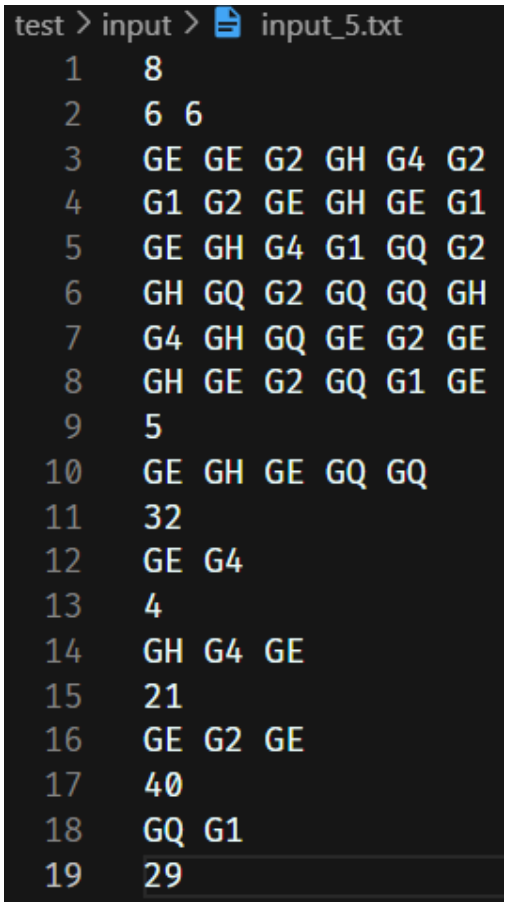
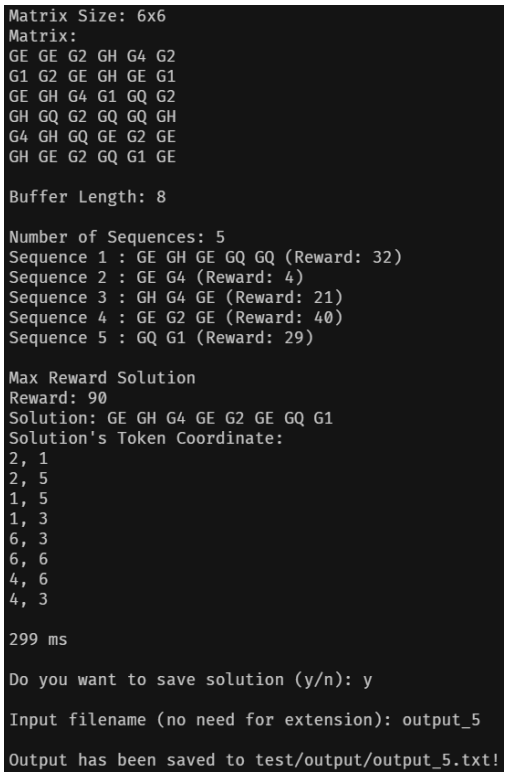
c. Kasus 3

Input: input_3.txt	Output: output_3.txt
	

d. Kasus 4

Input: input_4.txt	Output: output_4.txt
test > input > input_4.txt 7 7 7 AR AR AR 75 AR H5 H5 9O 9O 9O AR G2 H5 H5 H5 G2 75 H5 AR AR H5 9O AR 9O AR AR 75 H5 G2 G2 75 H5 G2 H5 9O H5 AR AR AR 75 AR 9O H5 9O AR AR 75 AR G2 5 AR H5 21 H5 75 H5 16 9O G2 AR 75 G2 34 AR 9O 43 9O G2 AR 46	Matrix Size: 7x7 Matrix: AR AR AR 75 AR H5 H5 9O 9O 9O AR G2 H5 H5 H5 G2 75 H5 AR AR H5 9O AR 9O AR AR 75 H5 G2 G2 75 H5 G2 H5 9O H5 AR AR AR 75 AR 9O H5 9O AR AR 75 AR G2 Buffer Length: 7 Number of Sequences: 5 Sequence 1 : AR H5 (Reward: 21) Sequence 2 : H5 75 H5 (Reward: 16) Sequence 3 : 9O G2 AR 75 G2 (Reward: 34) Sequence 4 : AR 9O (Reward: 43) Sequence 5 : 9O G2 AR (Reward: 46) Max Reward Solution Reward: 126 Solution: AR 9O G2 AR H5 75 H5 Solution's Token Coordinate: 1, 1 1, 2 5, 2 5, 1 6, 1 6, 4 7, 4 286 ms Do you want to save solution (y/n): y Input filename (no need for extension): output_4 Output has been saved to test/output/output_4.txt!

e. Kasus 5

Input: input_5.txt	Output: output_5.txt
	

f. Kasus 6

Input: input_6.txt	Output: output_6.txt
test > input > input_6.txt 8 8 8 N5 HW 8F FA FA N5 HW BT F8 KG HW A1 HW HW P2 B0 B0 FA 8F P2 P2 N5 HW HW B0 HW FA FA 8F A1 N5 KG BT A1 N5 KG 8F 8F A1 BT P2 BT HW B0 P2 B0 A1 N5 KG KG N5 BT HW A1 8F BT BT A1 A1 BT KG FA KG B0 5 B0 F8 B0 BT 8F 49 F8 FA A1 28 A1 F8 KG B0 BT 24 BT A1 8F A1 P2 9 F8 KG HW P2 45	Matrix Size: 8x8 Matrix: N5 HW 8F FA FA N5 HW BT F8 KG HW A1 HW HW P2 B0 B0 FA 8F P2 P2 N5 HW HW B0 HW FA FA 8F A1 N5 KG BT A1 N5 KG 8F 8F A1 BT P2 BT HW B0 P2 B0 A1 N5 KG KG N5 BT HW A1 8F BT BT A1 A1 BT KG FA KG B0 Buffer Length: 8 Number of Sequences: 5 Sequence 1 : B0 F8 B0 BT 8F (Reward: 49) Sequence 2 : F8 FA A1 (Reward: 28) Sequence 3 : A1 F8 KG B0 BT (Reward: 24) Sequence 4 : BT A1 8F A1 P2 (Reward: 9) Sequence 5 : F8 KG HW P2 (Reward: 45) Max Reward Solution Reward: 49 Solution: HW FA B0 F8 B0 BT 8F Solution's Token Coordinate: 2, 1 2, 3 1, 3 1, 2 8, 2 8, 1 3, 1 5275 ms Do you want to save solution (y/n): y Input filename (no need for extension): output_6 Output has been saved to test/output/output_6.txt!

V. REPOSITORY GITHUB

Seluruh kode program, uji kasus beserta hasilnya, dan laporan ini terlampir pada *repository* Github (https://github.com/juanaw6/Tucil1_13522073).

LAMPIRAN

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		✔