Tugas Kecil 1 IF2211 Strategi Algoritma

Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force

Disusun oleh :

Erdianti Wiga Putri Andini (13522053)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2024

DAFTAR ISI

BAB I

DESKRIPSI MASALAH DAN ALGORITMA

1.1. Algoritma Brute Force

Algoritma brute force adalah metode yang langsung digunakan untuk menyelesaikan suatu masalah dengan cara yang sederhana, tanpa memperkenalkan kompleksitas yang tidak diperlukan. Pendekatan ini umumnya mengacu pada pernyataan dalam permasalahan dan konsep yang terlibat. Proses penyelesaian dilakukan secara langsung dan jelas, sesuai dengan prinsip "*Just do it*!" atau "*Just Solve it!"*.[1] Solusi brute force adalah dengan menghitung total jarak untuk setiap rute yang mungkin, kemudian memilih yang terpendek. Ini tidak terlalu efisien karena ada kemungkinan untuk mengeliminasi banyak rute melalui algoritma yang cerdas. Kompleksitas waktu dari brute force adalah O(mn), yang kadang-kadang ditulis sebagai O(n*m). Jadi, jika kita mencari string "n" karakter dalam sebuah string "m" karakter menggunakan brute force, dibutuhkan n * m percobaan. [2]

Pendekatan brute force memiliki sejumlah kelebihan dan kekurangan. Dari segi kelebihan, metode ini menjamin menemukan solusi yang benar dengan mencantumkan semua solusi kandidat yang mungkin untuk masalah yang dihadapi. Selain itu, pendekatan ini bersifat generik dan tidak terbatas pada domain masalah tertentu, sehingga dapat diterapkan secara luas. Metode brute force juga cocok untuk menyelesaikan masalah-masalah kecil dan sederhana serta dikenal karena kesederhanaannya, sehingga sering digunakan sebagai benchmark perbandingan. Namun, di sisi lain, pendekatan brute force cenderung tidak efisien. Untuk masalah real-time, analisis algoritma sering kali melebihi urutan pertumbuhan O(N!). Metode ini juga lebih mengandalkan penggunaan daya sistem komputer untuk memecahkan masalah daripada pada desain algoritma yang baik. Brute force algorithms juga cenderung lambat dan kurang konstruktif atau kreatif dibandingkan dengan algoritma yang dibangun menggunakan paradigma desain lainnya.[3]

1.2. Cyberpunk 2077 Breach Protocol

Breach Protocol dalam Cyberpunk 2077 adalah sebuah minigame meretas yang mensimulasikan proses peretasan jaringan lokal dari ICE (Intrusion Countermeasures Electronics) dalam dunia permainan tersebut. Dalam minigame ini, pemain dihadapkan pada beberapa komponen kunci yaitu token, matriks, sekuens, dan buffer. Token merupakan karakter alfanumerik berpasangan seperti E9, BD, dan 55. Matriks adalah susunan token yang akan dipilih untuk membentuk urutan kode. Sekuens adalah rangkaian token yang harus dipasangkan. Sedangkan buffer menentukan jumlah maksimal token yang dapat disusun secara berurutan. Aturan permainan Breach Protocol mengharuskan pemain untuk bergerak secara bergantian secara horizontal dan vertikal hingga semua sekuens berhasil dipasangkan atau buffer mencapai kapasitas maksimal. Pemain memulai dengan memilih satu token dari baris teratas dalam matriks, lalu mencocokkan sekuens pada token yang ada di buffer. Token yang ada di buffer dapat digunakan untuk lebih dari satu sekuens. Setiap sekuens memiliki bobot hadiah yang berbeda-beda dan minimal terdiri dari dua token.[4]

1.3. Algoritma Penyelesaian Cyberpunk 2077 Breach Protocol dengan Pendekatan Brute Force

Penyelesaian permasalahan permainan Cyberpunk 2077 Breach Protocol dapat dilakukan menggunakan pendekatan *brute force*. Untuk lebih detailnya, langkah-langkah penyelesaian tertera sebagai berikut.

1. Program meminta pengguna untuk memilih jenis input. Pengguna dapat menginput dari pembacaan file (.txt) maupun melalui CLI dan mendapat matriks serta sekuens secara acak. Input melalui file akan memanfaatkan fungsi read_file untuk mengekstrak elemen-elemen di dalam file tersebut. Sedangkan input CLI membutuhkan beberapa input dari pengguna seperti jumlah token unik, nama token, ukuran buffer, ukuran matriks (baris dan kolom), jumlah sekuens, ukuran maksimal sekuens, dan poin tiap sekuens. Pembacaan input ini akan memanfaatkan fungsi manual_input dan nantinya akan dihasilkan matriks dan sekuens yang disusun secara acak.

2. Setelah berhasil membaca seluruh input, algoritma brute force ini mengecek semua kemungkinan path (kombinasi dari beberapa sekuens) yang dimulai dari setiap elemen pada baris pertama matriks. Hal ini akan diproses melalui fungsi collect_possible_path. Fungsi ini bersifat rekursif untuk mencari kombinasi sekuens yang mungkin terbentuk dari pergerakan secara vertikal maupun horizontal dengan detail pergerakan vertikal – horizontal – vertikal – horizontal – dst sampai buffer penuh. Fungsi ini memanfaatkan algoritma brute force karena bersifat iteratif hingga semua kemungkinan dicari kemudian disimpan. Setiap path hasil kombinasi beberapa sekuens ini akan disimpan di suatu array (menjadi array of array). Hasil yang disimpan dari fungsi ini tidak hanya urutan token saja, namun urutan koordinat tiap token juga.
3. Setelah ditemukan path sebagai array kombinasi sekuens, dilanjutkan dengan pencarian poin setiap path. Caranya adalah dengan memanfaatkan fungsi is_subsequence dan count_points, dimana fungsi ini mengecek apakah ada sekuens yang tertera pada path tersebut. Bila ada, poin dari sekuens yang tertera tadi akan ditambahkan. Poin-poin tiap sekuens ini nantinya disimpan dalam array melalui fungsi collect_possible_point.
4. Lalu pada program main, setelah fungsi-fungsi tersebut sudah dijalankan, akan dilanjutkan dengan pencarian poin maksimum pada array poin yang sudah dihasilkan tadi. Dari langkah ini akan mendapat index dimana poin maksimum ini berada sehingga dapat memilih elemen pada array yang mengandung poin maksimum, baik itu pada array of coordinate ataupun array of token.
5. Bila tidak terdapat poin maksimum, artinya tidak ada path yang memenuhi sekuens pada matriks tersebut .
6. Pada akhir program, pengguna dapat memilih apakah ingin menyimpan solusi ke dalam file atau tidak. Bila pengguna memilih untuk menyimpan file, maka akan dijalankan fungsi save_file.

BAB II

IMPLEMENTASI ALGORITMA DALAM BAHASA PYTHON

2.1. File input.py

File ini berisi fungsi-fungsi yang digunakan untuk membaca input sesuai dengan pilihan pengguna.

Nama Fungsi	Deskripsi
read_file	Fungsi untuk membaca input berupa file. Akan mengembalikan nilai ukuran buffer, matriks, sekuens beserta poinnya, dan ukuran matriks.
arrange_seq	Fungsi untuk menyusun sekuens secara random sesuai dengan ketentuan yang telah diinput pengguna melalui fungsi manual_input. Poin dari tiap sekuens juga dirandom dari -50 sampai 50. Sedangkan ukuran tiap sekuens dirandom dari 2 – ukuran max sekuens yang diinput pengguna.
arrange_matrix	Fungsi untuk menyusun elemen matriks (token) secara random sesuai dengan token-token yang telah diinput pengguna.
manual_input	Fungsi untuk membuat pengguna melakukan input secara manual melalui CLI. Yang ditanyakan adalah jumlah token unik, tokennya apa saja, ukuran buffer, ukuran matriks, jumlah sekuens, dan ukuran max tiap sekuens.

2.2. File solver.py

File ini berisi fungsi-fungsi yang digunakan untuk menyelesaikan permasalahan Cyberpunk 2077 Breach Protocol dengan algoritma brute force.

Nama Fungsi	Deskripsi
check_move	Fungsi untuk mengecek apakah step masih bisa dilakukan selama current_row dan current_col masih lebih kecil dari ukuran matriks dan juga elemen[current_row][current_col] belum pernah digunakan sebelumnya.
collect_possible_path	Fungsi untuk mendapatkan kumpulan path yang merupakan kombinasi dari beberapa sekuens dengan pendekatan brute force. Mengembalikan semua kemungkinan path (urutan token) dan koordinatnya, masing-masing dalam tipe data array of array
find_path	Fungsi untuk mencari kombinasi sekuens secara iteratif dimulai dari pencarian vertikal lalu horizontal lalu vertikal lagi dan berulang sampai ukuran curr_buffer sama dengan ketentuan ukuran_buffer.
is_subsequence	Fungsi untuk mencari apakah sebuah sekuens terdapat di dalam suatu path atau tidak.
count_points	Fungsi untuk mencari poin yang didapat oleh suatu path dengan memanfaatkan fungsi is_subsequence.
collect_possible_point	Fungsi untuk menyimpan dan mengembalikan poin tiap path dalam array of integer.

2.3. File output.py

File ini berisi fungsi-fungsi yang digunakan untuk membaca input sesuai dengan pilihan pengguna.

Nama Fungsi	Deskripsi
save_file	Fungsi untuk menyimpan hasil program dalam berkas txt dan disimpan ke dalam folder ‘saved’.

2.4. File main.py

File ini berisi merupakan driver utama dari program ini sehingga tidak terdapat satupun fungsi di dalamnya. Hal-hal yang dilakukan dalam file ini hanyalah memanggil fungsi-fungsi dari file lain untuk dijalankan.

BAB III

SOURCE CODE PROGRAM

3.1 Repositori Github

Repositori program dapat diakses melalui tautan GitHub berikut :

https://github.com/wigaandini/Tucil1_13522053

3.2 Source Code Program

3.2.1 input.py

input.py > ...

```python
import numpy as np
import random

def read_file(filename):
    with open(filename, "r") as f:
        buffer_size = int(f.readline().strip())
        matrix_width, matrix_height = map(int, f.readline().split())
        matrix = []
        for _ in range(matrix_height):
            row = f.readline().split()
            matrix.append(row)
        number_of_sequences = int(f.readline().strip())
        sequences = []
        for _ in range(number_of_sequences):
            sequence = f.readline().split()
            reward = int(f.readline().strip())
            sequences.append((sequence, reward))
    return buffer_size, matrix, sequences, matrix_width, matrix_height


def arrange_seq(seq_num, max_seq_size, token_name):
    sequences_with_points = []
    for _ in range(seq_num):
        seq_size = random.randint(2, max_seq_size)
        seq = random.sample(token_name, k=seq_size)
        points = random.randint(-50, 50)
        sequences_with_points.append((seq, points))
    return sequences_with_points
```

```python
def arrange_matrix(matrix_height, matrix_width, tokens):
    matrix = []
    for i in range(matrix_height):
        row = [random.choice(tokens) for j in range(matrix_width)]
        matrix.append(row)
    return matrix


def manual_input():
    number_of_unique_tokens = int(input("\nEnter the number of unique tokens: "))
    tokens = []
    while True:
        token_name = input("Enter the tokens: ").split()
        tokens.extend(token_name)
        # Cek jumlah token sama/ga sama number unique tokens
        if len(token_name) == number_of_unique_tokens:
            break
        else:
            print("The total number of tokens does not match the number of unique tokens. Please try again.")
    buffer_size = int(input("Enter the buffer size: "))
    matrix_height, matrix_width = map(int, input("Enter the matrix_height and matrix_width (separated by space): ").split())
    seq_num = int(input("Enter the number of sequences: "))
    max_seq_size = int(input("Enter the max size of the sequence: "))
    matrix = arrange_matrix(matrix_height, matrix_width, tokens)
    sequences = arrange_seq(seq_num, max_seq_size, token_name)

    return buffer_size, matrix, sequences, matrix_width, matrix_height, tokens, seq_num, max_seq_size
```

3.2.2 solver.py

solver.py > is_subsequence

```python
def check_move(row, col, matrix, used):
    return 0 <= row < len(matrix) and 0 <= col < len(matrix[0]) and (row, col) not in used


def collect_possible_path(matrix, buffer_size):
    def find_path(row, col, curr_buffer_length, vertical):
        used.add((row, col))
        curr_token.append(matrix[row][col])
        curr_coor.append((col + 1, row + 1))
        if curr_buffer_length == buffer_size - 1:
            seq.append(curr_token[:])
            coor.append(curr_coor[:])
        else:
            if vertical:
                moves = [(1, 0), (-1, 0)]
            else:
                moves = [(0, 1), (0, -1)]

            for i, j in moves:
                curr_row, curr_col = row, col
                for _ in range(buffer_size - curr_buffer_length):
                    curr_row, curr_col = curr_row + i, curr_col + j
                    if check_move(curr_row, curr_col, matrix, used):
                        find_path(curr_row, curr_col, curr_buffer_length + 1, not vertical)

            used.remove((row, col))
            curr_token.pop()
            curr_coor.pop()

    seq, coor = [], [] # Store semua path
    curr_coor, curr_token = [], [] # Store per path
    used = set() # Buat detect dia udah pernah dipake ato belom, kalo udah berarti diskip

    for k in range(len(matrix[0])):
        find_path(0, k, 0, True)
    return seq, coor

```

```python
# Buat cek apakah sequence ada di path
def is_subsequence(sequence, path):
    sequence_length = len(sequence)
    path_length = len(path)

    for i in range(path_length - sequence_length + 1):
        if path[i:i+sequence_length] == sequence:
            return True
    return False


# Buat hitung point dari setiap path sesuai dengan sequence yang ada
def count_points(seq, path):
    points = {tuple(sequence[0]): 0 for sequence in seq}
    for sub_path in seq:
        if is_subsequence(sub_path[0], path):
            points[tuple(sub_path[0])] += sub_path[1]
    return points


# Buat store semua point dari setiap path yang mungkin
def collect_possible_point(seq, path):
    arr_point = []
    for i in range(len(path)):
        points = count_points(seq, path[i])
        total_points = sum(points.values())
        arr_point.append(total_points)
    return arr_point
```

3.2.3 output.py

```python
import os

# Saving file to folder "saved"
def save_file(filename, possible_point, max_point, path_choosen, coordinates, exe_time):
    save_folder = "saved"
    os.makedirs(save_folder, exist_ok=True)
    filepath = os.path.join(save_folder, filename)
    with open(filepath, "w") as f:
        if not possible_point:
            f.write("No possible path found.")
        else:
            f.write("{} \n".format(max_point))
            for i in range(len(path_choosen)):
                f.write(path_choosen[i])
                if i != len(path_choosen) - 1:
                    f.write(" ")
            f.write("\n")
            for i in range(len(coordinates)):
                f.write("{} , {}\n".format(coordinates[i][0], coordinates[i][1]))
            f.write("{} ms\n".format(exe_time))

```

3.2.4 main.py

```python
# Main Program

from input import *
from solver import *
from output import *
import os
import time


# Inizialiation
print()
print("                                                        Welcome to the Cyberpunk Hacker!                                  ")
print(" ___ _ _ ___ ___ ___ ___ _ _ _ _ _ _  ___ ___ ___ ___  ___ ___ ___ ___ ___ ___ _ _  ___ ___ ___ ___ ___ ___ ___ _  ")
print("|  _]| | || . ]| __]| . \| . \| | || \ || / /  [_  ]|   ||_  ||_  |  | . ]| . \| __]| . ||  _]| | |  | . \| . \| . ||_ _|| . ||  _]| . || |  ")
print("| [__\  /| . \| _] |   /|  _/| | ||   || \    / / | / | / /  / /   | . \|   /| _] |   || [__|   |  |  _/|   /| | || | || | || [__| | || |_ ")
print("`___/ |_| |___/|___]|_\_\|_|   \__||_\_||_\_\  [___] \__|/_/  /_/   |___/|_\_\|___]|_|_|`___/|_|_|  |_|  |_\_\`___' |_| `___'`___/`___'|___|")

print()
print("                              This program will help you to find the maximum point and the steps from the given matrix and sequences.                ")
print("-----------------------------------------------------------------------------------------------------------------------------------------------------------")
print("                                                         Erdianti Wiga Putri Andini                                       ")
print("                                                               13522053 - K1                                              ")
print("-----------------------------------------------------------------------------------------------------------------------------------------------------------")
print()

# Input
print("========== CHOOSE INPUT MODE ==========")
print("              File Input (1)")
print("             Manual Input (2)")
print("=========================================")
input_mode = int(input("\nChoice: "))

check = False
while not check:
    # File input
    if input_mode == 1:
        check = True
        folder_name = "input"
        file_name = input("Enter the file name: ")
        file_path = os.path.join(folder_name, file_name)
        while not os.path.exists(file_path):
            print("File not found. Please try again.")
            file_name = input("Enter the file name: ")
            file_path = os.path.join(folder_name, file_name)
        buffer_size, matrix, sequence, matrix_width, matrix_height = read_file(file_path)
        print()

    # Manual input
    elif input_mode == 2:
        check = True
        buffer_size, matrix, sequence, matrix_width, matrix_height, tokens, seq_num, max_seq_size = manual_input()
        print("\nMatrix: ")
        for i in range(matrix_height):
            for j in range(matrix_width):
                print(matrix[i][j], end=' ')
            print()
        print("\nSequences that used: ")
        for i in range(seq_num):
            print("Sequence:", sequence[i][0], "Point:", sequence[i][1])

    else:
        print("\nInvalid input. Please try again.")
        print("Choose the input mode: ")
        print("File Input (1)")
        print("Manual Input (2)")
        input_mode = int(input("\nChoice: "))

start = time.time()

# Cari kemungkinan semua path yang bisa dibuat dari elemen-elemen di matrix baris pertama
possible_path, coordinate = collect_possible_path(matrix, buffer_size)
possible_point = collect_possible_point(sequence, possible_path) # Cari point dari setiap path yang mungkin
```

```
print("========== RESULT ==========")
if not possible_point:
    print("No possible path found.")
else:
    max_point = possible_point[0]
    for i in range(len(possible_point)):
        if possible_point[i] > max_point:
            max_point = possible_point[i]
            max_index = i
    print("Maximum point:", max_point)
    print("Steps that used:", len(possible_path[max_index]) - 1)

    # Print final sequence and coordinate
    print("Final sequence: ", end='')
    final_sequence = possible_path[max_index]
    for token in final_sequence:
        print(token, end=' ')

    print("\nFinal coordinate: ")
    final_coordinate = coordinate[max_index]
    for coor in final_coordinate:
        print(coor)
```

```
end = time.time()
total_waktu = round((end - start) * 1000, 3)
print()
print("Total waktu:", total_waktu, "ms")
print("-------------------------------------------------------------------------------------------------------------------------")
print("Do you want to save the result as text file? (Y/N)")
save = input("Choice: ")
if save == "Y" or save == "y":
    file_name = input("Enter the file name: ")
    save_file(file_name, possible_point, max_point, possible_path[max_index], coordinate[max_index], total_waktu)
    print("File has been saved as", file_name, "in the folder 'saved'.")
else:
    print("File not saved.")
print("Thank you for using this program!")
```

BAB IV

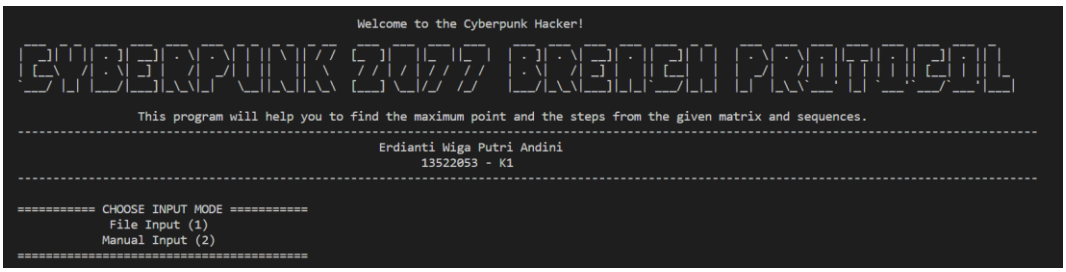
MASUKAN DAN LUARAN PROGRAM

4.1. Test Case 1 (Input dari File)

Isi case.txt

```
input > case.txt
1   7
2   6 6
3   7A 55 E9 E9 1C 55
4   55 7A 1C 7A E9 55
5   55 1C 1C 55 E9 BD
6   BD 1C 7A 1C 55 BD
7   BD 55 BD 7A 1C 1C
8   1C 55 55 7A 55 7A
9   3
10  BD E9 1C
11  15
12  BD 7A BD
13  20
14  BD 1C BD 55
15  30
```

Tampilan pertama saat program baru dijalankan

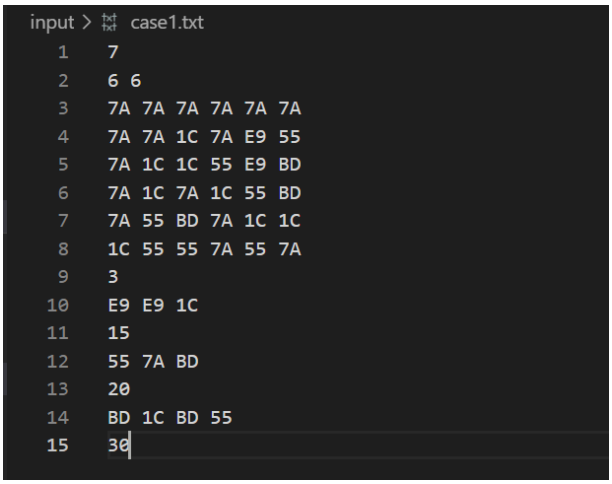


Output penyelesaian program case.txt pada terminal

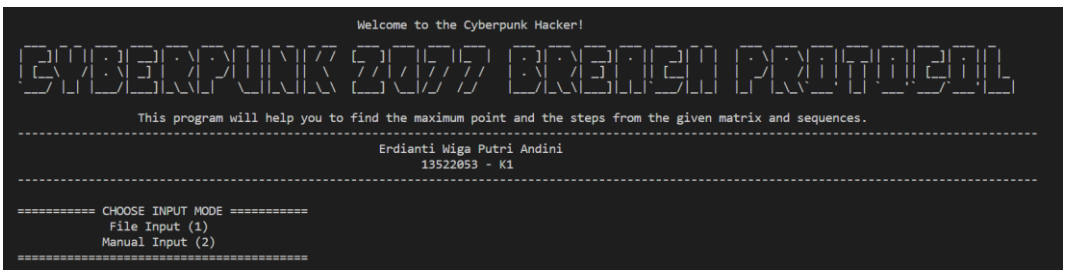
Output penyelesaian program case.txt yang telah disimpan pada file ok.txt

4.2. Test Case 2 (Input dari File)

Isi case1.txt



Tampilan pertama saat program baru dijalankan



Output penyelesaian program case1.txt pada terminal

```
Choice: 1
Enter the file name: case1.txt

========== RESULT ==========
Maximum point: 50
Steps that used: 6
Final sequence: 7A 55 7A BD 1C BD 55
Final coordinate:
(2, 1)
(2, 6)
(6, 6)
(6, 3)
(3, 3)
(3, 5)
(2, 5)

Total waktu: 214.553 ms
------------------------------------------------------------------------------------------------------------------------------------------------------
Do you want to save the result as text file? (Y/N)
Choice: y
Enter the file name: ok1.txt
File has been saved as ok1.txt in the folder 'saved'.
Thank you for using this program!
```

Output penyelesaian program case1.txt yang telah disimpan pada file ok1.txt

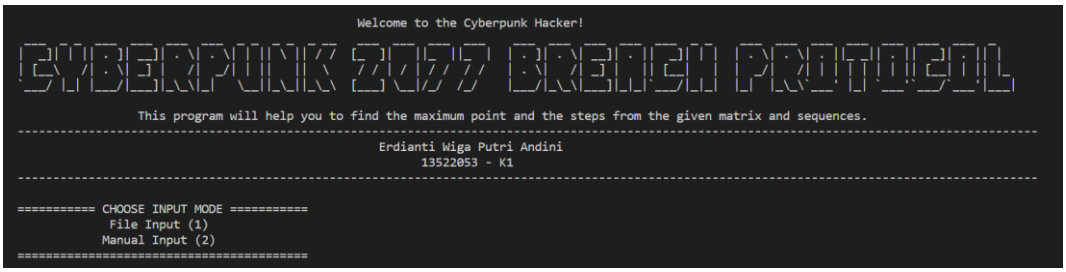
```
saved > ok1.txt
 1
 2   ___  _ _  ___  ___  ___  ___  _ _  _ _  _ _  ___  ___  ___  ___  ___  ___  _ _  ___  ___  ___  ___  ___  ___  ___  _
 3   | _ )| | || . )| __|| . \| . \| | || \ || / /  [_  ]|  ||_  ||_  |  | . )| . \| __]| . || _]| | |  | . \| . \| . ||_ _|| . ||  _]| . || |
 4   | [__\   /| . \| _] |   /|  _/| | ||   ||  \    / / | / | / /  / /   | . \|   /| _] |   || [__|   |  |  _/|   /| | | | | | || [__| | || |_
 5   `___/ |_| |___/|___]|_\_\|_|   \__||_\_||_\_\  [___] \__|/_/  /_/    |___/|_\_\|___]|_|_|`___/|_|_|  |_|  |_\_\`___' |_| `___'`___/`___'|___|
 6
 7   Maximum point: 50
 8   Final sequence: 7A 55 7A BD 1C BD 55
 9   Steps that used: 6
10   Final coordinate:
11   2 , 1
12   2 , 6
13   6 , 6
14   6 , 3
15   3 , 3
16   3 , 5
17   2 , 5
18   Execution time: 214.553 ms
```

4.3. Test Case 3 (Input dari File)

Isi case2.txt

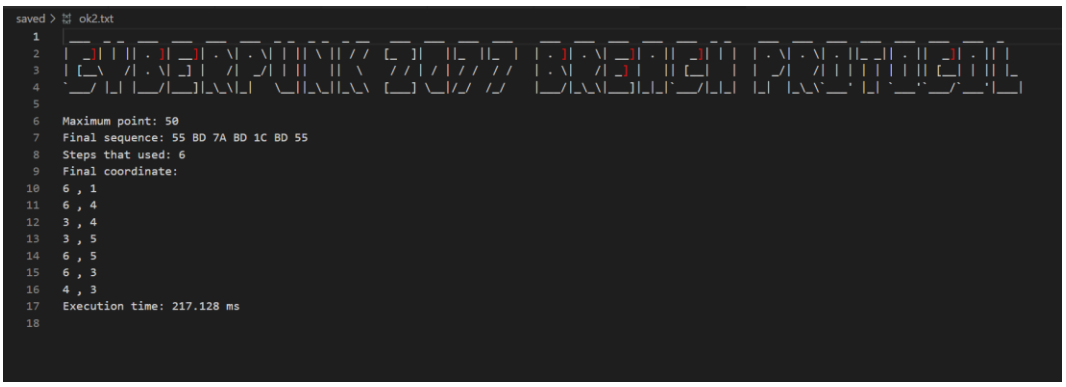
```
input > case2.txt
 1   7
 2   6 6
 3   7A 55 E9 E9 1C 55
 4   55 7A 1C 7A E9 55
 5   55 1C 1C 55 E9 BD
 6   E9 1C 7A 1C 55 BD
 7   E9 55 BD 7A 1C 1C
 8   1C 55 55 7A 55 7A
 9   3
10   E9 E9 1C
11   15
12   BD 7A BD
13   20
14   BD 1C BD 55
15   30
```

Tampilan pertama saat program baru dijalankan



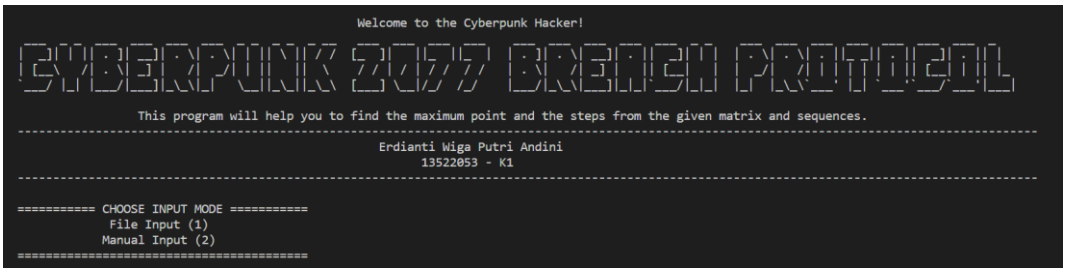
Output penyelesaian program case2.txt pada terminal

Output penyelesaian program case2.txt yang telah disimpan pada file ok2.txt



4.4. Test Case 4 (Input CLI)

Tampilan pertama saat program baru dijalankan

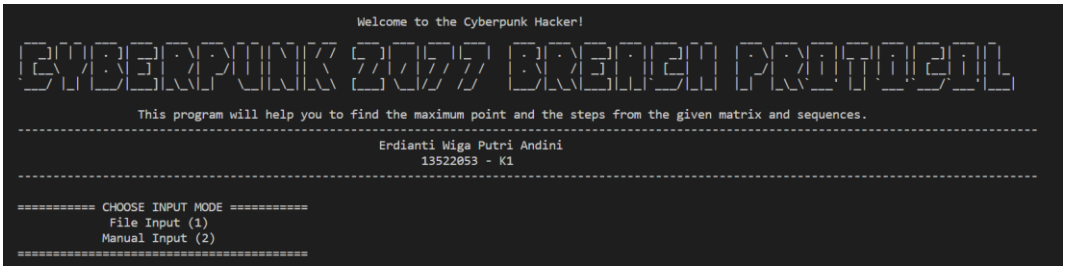


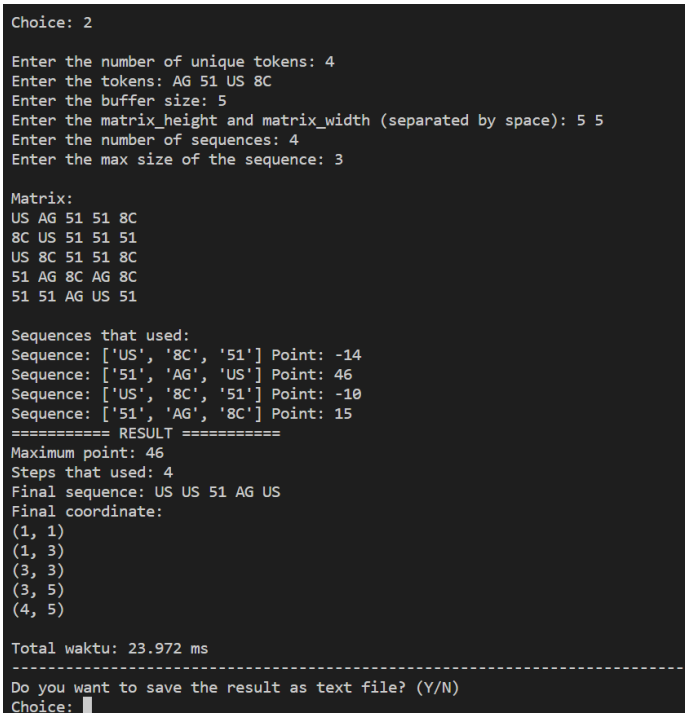
Output penyelesaian program pada terminal

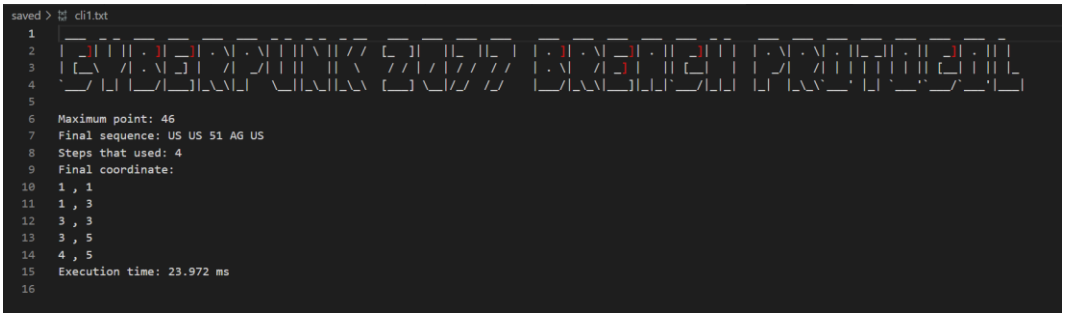
```
Choice: 2

Enter the number of unique tokens: 5
Enter the tokens: 7A BD 1C 39 55
Enter the buffer size: 7
Enter the matrix_height and matrix_width (separated by space): 4 5
Enter the number of sequences: 3
Enter the max size of the sequence: 4

Matrix:
39 55 1C 39 55
BD 39 55 39 55
7A 7A BD 7A 55
55 1C 55 55 7A

Sequences that used:
Sequence: ['1C', '39', 'BD', '7A'] Point: -27
Sequence: ['39', '1C', 'BD'] Point: 3
Sequence: ['39', 'BD', '1C', '7A'] Point: -43
========== RESULT ==========
Maximum point: 3
Steps that used: 6
Final sequence: 39 BD 39 39 1C BD 7A
Final coordinate:
(1, 1)
(1, 2)
(4, 2)
(4, 1)
(3, 1)
(3, 3)
(4, 3)

Total waktu: 41.82 ms
```

Output penyelesaian program yang telah disimpan pada file cli.txt

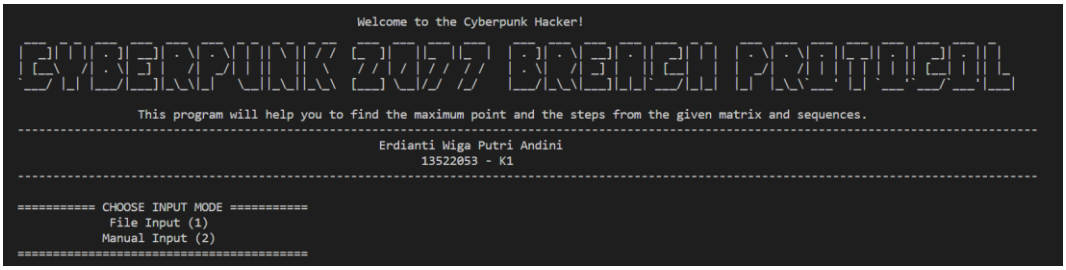
4.5. Test Case 5 (Input CLI)

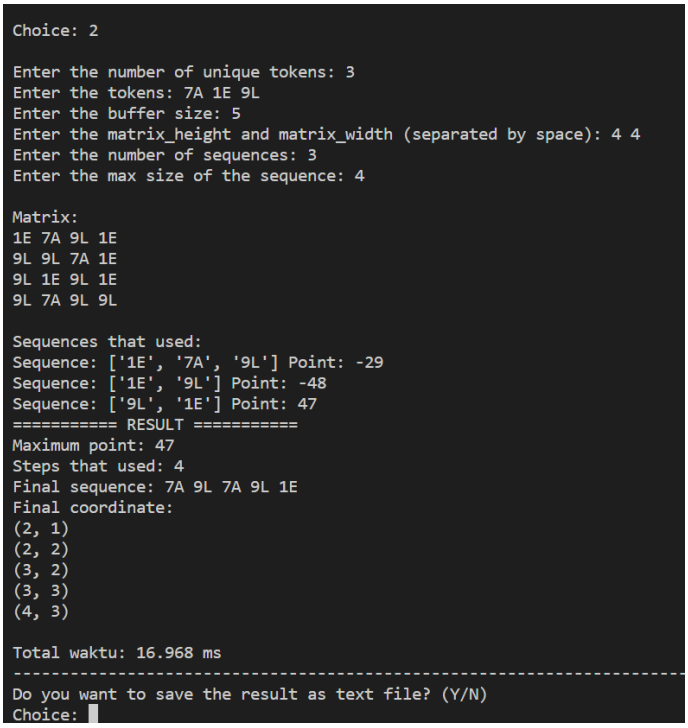
4.6. Tampilan pertama saat program baru dijalankan


Output penyelesaian program pada terminal


Output penyelesaian program yang telah disimpan pada file cli1.txt


4.7. Test Case 6 (Input CLI)

4.8. Tampilan pertama saat program baru dijalankan


Output penyelesaian program pada terminal


Output penyelesaian program yang telah disimpan pada file cli2.txt


BAB V

LAMPIRAN

No	Poin	Ya	Tidak
1.	Program berhasil dikompilasi tanpa kesalahan	V	
2.	Program berhasil dijalankan	V	
3.	Program dapat membaca masukan berkas.txt	V	
4.	Program dapat menghasilkan masukan secara acak	V	
5.	Solusi yang diberikan program optimal	V	
6.	Program dapat menyimpan solusi dalam berkas.txt	V	
7.	Program memiliki GUI		V

DAFTAR PUSTAKA

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf

[2] freeCodeCamp.org. (2022, June 20). *Brute force algorithms explained*. https://www.freecodecamp.org/news/brute-force-algorithms-explained/

[3] GeeksforGeeks. (2024, January 18). *Brute Force Approach and its pros and cons*. https://www.geeksforgeeks.org/brute-force-approach-and-its-pros-and-cons/

[4] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Tucil1-2024.pdf