Laporan Tugas Kecil 1
IF2211 Strategi Algoritma

Penyelesaian *Cyberpunk 2077 Breach Protocol*
dengan Algoritma Brute Force

Disusun oleh:

Christopher Brian 13522106

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2023

Daftar Isi

Bab 1: Algoritma *Brute Force*

1.1. *Input*/*Output*

File handling dalam program ini dibuat menggunakan bantuan modul tkinter. Penghitungan waktu dalam program ini dibuat menggunakan bantuan modul time. Jika pengguna memilih opsi *input* menggunakan *file* .txt, pengguna akan diminta memilih *file* tersebut dalam *pop-up window*. Begitu juga ketika pengguna memilih opsi menyimpan hasil ke dalam *file* .txt. Selain itu, setiap *input* juga memiliki *validity checking*, pengguna akan diminta untuk mengubah isi *file* .txt atau mengulangi *input* jika terdapat *input* yang tidak valid. Pada opsi *input* menggunakan *file* .txt, jika terdapat *input* yang tidak valid, program akan langsung berhenti. Pada opsi yang sama, metode strip() digunakan untuk membaca baris demi baris, sedangkan metode split() digunakan untuk memisahkan beberapa input pada baris yang sama. Pada opsi *input* dengan *file* .txt, pengguna akan diminta *input* ukuran *buffer*, lebar dan tinggi matriks, matriks, jumlah sekuens, serta masing-masing sekuens dan *reward*-nya. Pada opsi *input* melalui *command line interface*, pengguna akan diminta *input* jumlah token unik, token, ukuran *buffer*, lebar dan tinggi matriks, jumlah sekuens, serta ukuran maksimal sekuens. *Output* kedua opsi kurang lebih sama, yaitu skor tertinggi, *buffer*, koordinat setiap token secara terurut, waktu eksekusi program dalam ms, serta *prompt* untuk menyimpan solusi dalam sebuah *file* .txt.

1.2. Generasi Acak

Opsi *input* menggunakan *command line interface* mengharuskan generasi matriks & sekuens beserta *reward*-nya secara acak dari token, ukuran matriks, dan ukuran maksimum sekuens yang di-*input*. Hal ini dilakukan menggunakan modul random.

1.2.1. Generasi Matriks Acak

Metode yang digunakan dalam program ini untuk memilih token acak untuk setiap elemen matriks adalah random.choice. Dalam program, fungsi yang bertanggung jawab adalah generate-random_matrix dengan parameter berupa daftar token dan ukuran matriks serta mengembalikan sebuah matriks.

1.2.2. Generasi Sekuens Acak

Untuk menentukan ukuran sekuens, digunakan metode random.randint() untuk memilih integer acak antara 2 sampai ukuran maksimum token. Sekuens acak dibentuk menggunakan metode random.sample() berdasarkan daftar token dan ukuran sekuens yang telah ditentukan sebelumnya. *Reward* acak ditentukan dengan metode random.randint() untuk memilih integer acak antara -100 sampai 100. Fungsi yang bertanggung jawab untuk ini adalah generate_random_sequence yang menerima *input* daftar token dan ukuran maksimum sekuens, serta mengembalikan sekuens acak beserta *reward*-nya yang bernilai acak juga.

1.3. Pencarian Solusi dengan Algoritma *Brute Force*

Pencarian solusi secara garis besar terdiri dari tiga bagian, yaitu generasi dan iterasi semua kemungkinan *series*, pengecekan sekuens yang terdapat dalam *series*, dan pencarian *series* dengan skor tertinggi.

1.3.1. Generasi dan Iterasi *Series*

Fungsi generate_series bertanggung jawab untuk generasi semua *series* yang mungkin dari matriks. Fungsi ini menerima parameter matriks dan ukuran *buffer*, serta mengembalikan *list series*. Di dalam fungsi ini, terdapat fungsi iterate yang bertanggung jawab untuk mengiterasi semua kemungkinan *series*. Fungsi ini menerima parameter temp_series (*series* sementara yang sedang dibangun), temp_coordinates (daftar koordinat dari semua elemen dalam *series*, dan parameter *boolean* same_row yang jika bernilai *True* menandakan pemilihan elemen secara vertikal (dalam kolom yang sama) dan jika bernilai *False* menandakan pemilihan elemen secara horizontal (dalam baris yang sama). *List series* diinisialisasi sebagai *list* kosong. Kemudian, fungsi akan bekerja menurut *case-case* yang sudah ditentukan. Jika panjang temp_series sama dengan ukuran *buffer* dan semua koordinat unik, *series* beserta *list* koordinat akan ditambahkan ke dalam *list series*. Jika *series* masih kosong, untuk setiap kolom, akan dilakukan iterasi dengan elemen di baris pertama kolom ke-I sebagai elemen pertama dan temp_series, [(0, i)] sebagai temp_coordinate, dan same_row bernilai *True* yang menandakan pemilihan elemen selanjutnya akan dilakukan secara horizontal. Jika *buffer* tidak kosong dan belum penuh, akan dilakukan iterasi secara rekursif. Jika same_row bernilai *True*, untuk setiap baris yang bukan merupakan baris dari elemen sebelumnya, akan dilakukan iterasi dengan temp_series adalah temp_series ditambah elemen matriks baris ke-I kolom elemen sebelumnya, koordinat elemen tersebut ditambahkan ke temp_coordinates, dan same_row bernilai *False*. Jika same_row bernilai *False*, untuk setiap kolom, jika kolom bukan merupakan kolom elemen sebelumnya, akan dilakukan iterasi dengan elemen matriks baris elemen sebelumnya kolom ke-i ditambahkan ke temp_series, koordinat elemen tersebut ditambahkan ke temp_coordinates, dan same_row bernilai *True*. Selain itu, digunakan juga fungsi unique_coordinates untuk memastikan semua elemen dalam *series* unik, yaitu dengan mengecek apakah panjang dari *set* temp_coordinates sama dengan panjang temp_coordinates (set harus unik, temp_coordinates tidak harus).

1.3.2. Pengecekan Sekuens dalam *Series*

Fungsi check_sequence_in_series bertanggung jawab untuk mengecek sekuens mana saja yang terdapat dalam sebuah *series*. Fungsi ini menerima parameter series dan *dictionary* sequence_and_reward serta mengembalikan total_score untuk *series* tersebut. Variabel total_score diinisialisasi dengan 0 dan series_string diubah ke dalam *string* dengan menghilangkan spasi yang memisahkan elemennya. Setelah itu, dilakukan iterasi untuk setiap sekuens dan *reward* dalam sequence_and_reward. Jika sekuens terdapat dalam *series* (merupakan *substring*), maka *reward* untuk sekuens tersebut ditambahkan ke dalam total_score.

1.3.3. Pencarian Solusi Terbaik

Fungsi find_highest_score_series bertanggung jawab untuk mencari *series* dengan skor terbaik. Fungsi ini menerima parameter matrix, buffer_size, dan sequence_and_reward, serta mengembalikan best_series, highest_score, best_coordinates, dan first_series. Pertama, fungsi ini akan memanggil fungsi generate_series untuk menghasilkan semua *series* yang mungkin. Kemudian, highest_score diinisialisasi sebagai 0 dan best_series serta best_coordinates diinisialisasi sebagai None. Untuk setiap *series* dan *coordinates*, akan diiterasi semua *series* dan dihitung total_scorenya menggunakan fungsi check_sequence_in_series. Jika total_score melebihi highest_score saat itu, nilai highest_score di-*update* menjadi nilai total_score saat itu. Untuk case saat highest_score adalah 0, fungsi juga menyimpan data *series* pertama yang di-*generate* dalam variable first_series.

Bab 2: *Source Code*

```
# Import modul yang digunakan
import time
import random
import tkinter as tk
from tkinter import filedialog

# ASCII art
print(r"
_____       _                              _         _____  _____  __________
_ ")
print(r"/  __ \     | |                            | |       /
__  \|  _  ||___  /___  / ")
print(r"| /  \/_   _| |__   ___ _ __ _ __  _   _ _ __ | | __    `' / /'| |/'
|   / /   / /  ")
print(r"| |   | | | | '_ \ / _ \ '__| '_ \| | | | '_ \| |/ /      / /  |  /| |  /
/   / /   ")
print(r"| \__/\ |_| | |_) |  __/ |  | |_) | |_| | | | |   <     ./ /___\ |_/ /./
/  ./ /    ")
print(r" \____/\__, |_.__/ \___|_|  | .__/ \__,_|_| |_|_|\_\    \_____/ \___/
\_/   \_/     ")
print(r"         __/ |               |
|                                          ")
print(r"        |___/                |_|
          ")

print(r"  ____                      _       _____           _                  _
")
print(r" |  _ \                    | |     |  __ \         | |                |
|")
print(r" | |_) |_ __ ___  __ _  ___| |__   | |__) | __ ___ | |_ ___   ___ ___ |
|")
print(r" |  _ <| '__/ _ \/ _` |/ __| '_ \  |  ___/ '__/ _ \| __/ _ \ / __/ _ \|
|")
print(r" | |_) | | |  __/ (_| | (__| | | | | |   | | | (_) | || (_) | (_| (_) |
|")
print(r" |____/|_|  \___|\__,_|\___|_| |_| |_|   |_|  \___/ \__\___/
\___\___/|_|")

# Fungsi untuk generasi sekuens random
def generate_random_sequence(tokens, max_tokens):
    sequence_length = random.randint(2, max_tokens)
    random_sequence = ' '.join(random.sample(tokens, sequence_length))
```

```
    random_score = random.randint(-100, 100)  # Adjust the score range as needed
    return random_sequence, random_score

# Fungsi untuk generasi matriks random berdasarkan token yang diinput
def generate_random_matrix(tokens, width, height):
    return [[random.choice(tokens) for _ in range(width)] for _ in range(height)]

# Fungsi untuk generasi semua series yang mungkin secara iteratif
def generate_series(matrix, buffer_size):
    rows, cols = len(matrix), len(matrix[0])
    series_list = []

    # Fungsi untuk iterasi semua kemungkinan series
    def iterate(temp_series, temp_coordinates, same_row):
        nonlocal series_list

        if len(temp_series) == buffer_size and
unique_coordinates(temp_coordinates):
            series_list.append((temp_series.copy(), temp_coordinates.copy()))

        elif len(temp_series) == 0:
            for i in range(cols):
                iterate([matrix[0][i]], [(0, i)], True)

        elif len(temp_series) < buffer_size:
            last_coordinate = temp_coordinates[-1]
            if same_row:
                for i in range(rows):
                    if i != last_coordinate[0]:
                        iterate(
                            temp_series + [matrix[i][last_coordinate[1]]],
                            temp_coordinates + [(i, last_coordinate[1])],
                            False,
                        )
            else:
                for i in range(cols):
                    if i != last_coordinate[1]:
                        iterate(
                            temp_series + [matrix[last_coordinate[0]][i]],
                            temp_coordinates + [(last_coordinate[0], i)],
                            True,
                        )

    # Fungsi untuk memastikan semua elemen dalam series unik
```

```
    def unique_coordinates(temp_coordinates):
        return len(set(temp_coordinates)) == len(temp_coordinates)

    iterate([], [], True)
    return series_list

# Fungsi untuk mengecek sekuens apa saja yang terdapat dalam series
def check_sequence_in_series(series, sequence_and_reward):
    total_score = 0
    series_string = ''.join(map(str, series))

    for sequence_string, reward in sequence_and_reward.items():
        if sequence_string.replace(" ", "") in series_string.replace(" ", ""):
            total_score += reward

    return total_score

# Fungsi untuk mencari series dengan total skor terbanyak
def find_highest_score_series(matrix, buffer_size, sequence_and_reward):
    result = generate_series(matrix, buffer_size)

    highest_score = 0
    best_series = None
    best_coordinates = None

    for series, coordinates in result:
        total_score = check_sequence_in_series(series, sequence_and_reward)
        if total_score > highest_score:
            highest_score = total_score
            best_series = series.copy()
            best_coordinates = coordinates.copy()

    first_series = result[0][0]

    return best_series, highest_score, best_coordinates, first_series

# Pilihan metode input
print("1. Input dengan file .txt")
print("2. Input melalui CLI")
input_option = int(input("Pilih yang mana (1/2)? "))

while (input_option != 1) and (input_option != 2):
    input_option = int(input("Pilihan tidak tersedia.\nSilahkan pilih kembali
(1/2) "))
```

```
# Input dengan file .txt
if input_option == 1:
    # Pilih file input
    root = tk.Tk()
    root.withdraw()
    file_path = filedialog.askopenfilename(
        title="Select a Text File",
        filetypes=[("Text files", "*.txt"), ("All files", "*.*")]
    )

    # Penghitungan waktu dimulai
    start_time = time.time()

    # File handling
    with open(file_path, 'r') as file:
        buffer_size = int(file.readline().strip())

        if (buffer_size) <= 0:
            print("Ukuran buffer harus lebih dari 0.\nPerbaiki file input lalu
jalankan kembali program.")
            exit(1)

        second_line = list(map(int, file.readline().strip().split()))

        matrix_width = second_line[0]

        if (matrix_width) <= 0:
            print("Lebar matriks harus lebih dari 0.\nPerbaiki file input lalu
jalankan kembali program.")
            exit(1)

        matrix_height = second_line[1]

        if (matrix_height) <= 0:
            print("Tinggi matriks harus lebih dari 0.\nPerbaiki file input lalu
jalankan kembali program.")
            exit(1)

        matrix = [[] for _ in range(matrix_height)]

        for i in range(matrix_height):
            matrix_row = file.readline().strip().split()
```

```
            if len(matrix_row) != matrix_width:
                print(f"Jumlah elemen di baris {i+1} tidak sesuai dengan lebar
matriks.\nPerbaiki file input lalu jalankan kembali program.")
                exit(1)

            for element in matrix_row:
                if not element.isalnum() or len(element) != 2:
                    print(f"Elemen '{element}' di baris {i+1} bukan merupakan
token yang valid.\nPerbaiki file input lalu jalankan kembali program.")
                    exit(1)

            matrix[i].extend(matrix_row)

        if len(matrix) != matrix_height:
            print("Jumlah baris tidak sesuai dengan tinggi matriks.\nPerbaiki
file input lalu jalankan kembali program.")
            exit(1)

        sequence_and_reward = {}
        number_of_sequences = int(file.readline().strip())

        if (number_of_sequences) <= 0:
            print("Jumlah sekuens harus lebih dari 0.\nPerbaiki file input lalu
jalankan kembali program.")
            exit(1)

        loaded_sequences = 0

        for _ in range(number_of_sequences):
            sequence = file.readline().strip()
            reward = int(file.readline().strip())
            if len(sequence.split()) < 2:
                print("Sekuens harus terdiri dari minimal dua token.\nPerbaiki
file input lalu jalankan kembali program.")
            sequence_and_reward[sequence] = reward
            loaded_sequences += 1

        if loaded_sequences != number_of_sequences:
            print("Jumlah sekuens tidak sesuai.\nPerbaiki file input lalu
jalankan kembali program.")
            exit(1)

    file.close()
```

```
# Input melalui CLI
elif input_option == 2:
    token_number = int(input("Jumlah token: "))

    while (token_number) <= 0:
        print("Jumlah token harus lebih dari 0.\nMasukkan angka yang valid.")
        token_number = int(input("Jumlah token: "))

    while True:
        token_line = input("Token (dipisahkan oleh spasi): ").split()

        valid_tokens = all(token.isalnum() and len(token) == 2 for token in
token_line)

        if valid_tokens:
            break
        else:
            print("Terdapat token yang tidak valid. Harap masukkan token lagi.")

    buffer_size = int(input("Ukuran buffer: "))

    while (buffer_size) <= 0:
        print("Ukuran buffer harus lebih dari 0.")
        buffer_size = int(input("Ukuran buffer: "))

    matrix_size = input("Ukuran matriks (lebar dan tinggi dipisahkan oleh spasi):
").split()

    matrix_width = int(matrix_size[0])

    matrix_height = int(matrix_size[1])

    while (matrix_width <= 0) or (matrix_height <= 0):
        print("Lebar dan tinggi matriks harus lebih dari 0.")
        matrix_size = input("Ukuran matriks (lebar dan tinggi dipisahkan oleh
spasi): ").split()
        matrix_width = int(matrix_size[0])
        matrix_height = int(matrix_size[1])

    sequence_number = int(input("Jumlah sekuens: "))

    while (sequence_number) <= 0:
        print("Jumlah sekuens harus lebih dari 0.")
        sequence_number = int(input("Jumlah sekuens: "))
```

```
    maximum_sequence_size = int(input("Ukuran maksimum sekuens: "))

    while (maximum_sequence_size) <= 0:
        print("Ukuran maksimal sekuens harus lebih dari 0.")
        maximum_sequence_size = int(input("Ukuran maksimum sekuens: "))

    start_time = time.time()

    # Generasi matriks random
    matrix = generate_random_matrix(token_line, matrix_width, matrix_height)

    # Output matriks
    print("\nGenerated Matrix:")
    for row in matrix:
        print(' '.join(row))

    # Generasi sekuens dan reward random
    generated_sequences = []
    for _ in range(sequence_number):
        random_sequence, random_score = generate_random_sequence(token_line,
maximum_sequence_size)
        generated_sequences.append((random_sequence, random_score))
        print(f"Sequence: {random_sequence}, Score: {random_score}")

    # Pencarian sekuens terbaik
    sequence_and_reward = {seq: score for seq, score in generated_sequences}

best_series, highest_score, best_coordinates, first_series =
find_highest_score_series(matrix, buffer_size, sequence_and_reward)

# Penghitungan waktu selesai
end_time = time.time()

print(highest_score)
if best_series is not None:
    print(" ".join(best_series))
else:
    print("No series found.")
if best_coordinates is not None:
    for coordinates in best_coordinates:
        print(f"{coordinates[1] + 1}, {coordinates[0] + 1}")
else:
    print("No coordinates found.")
```

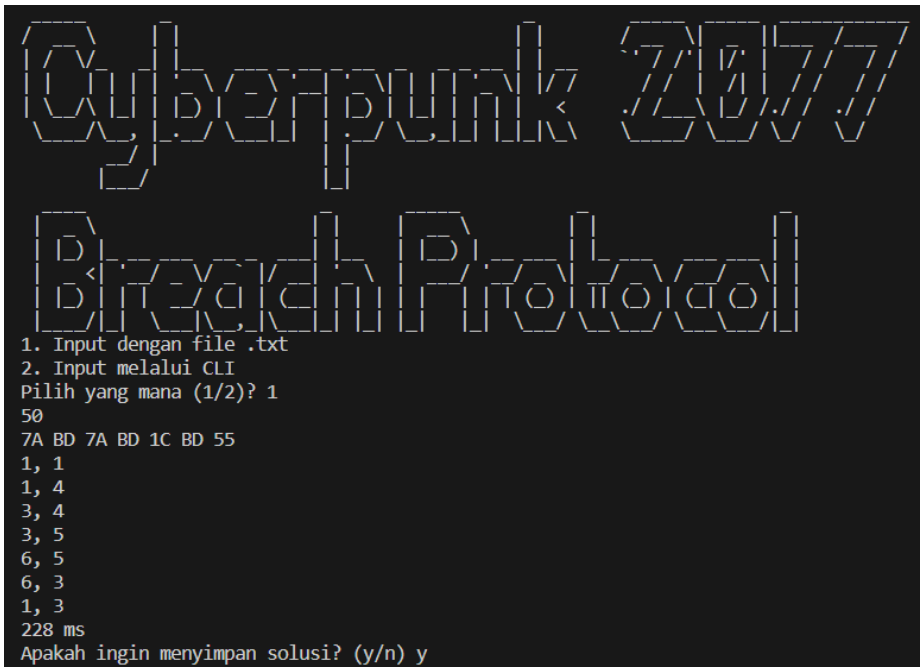
```
print(f"{int((end_time - start_time) * 1000)} ms")

# Opsi menyimpan output dalam file .txt
file_option = input("Apakah ingin menyimpan solusi? (y/n) ")

# File handling
if file_option == "y":
    output_file_path = filedialog.asksaveasfilename(
        title="Save Output As",
        defaultextension=".txt",
        filetypes=[("Text files", "*.txt"), ("All files", "*.*")]
    )

    if not output_file_path:
        print("Output save cancelled.")

    else:
        with open(output_file_path, "w") as output_file:
            if input_option == 2:
                output_file.write("\nGenerated Matrix:\n")
                for row in matrix:
                    output_file.write(' '.join(row) + "\n")
                output_file.write("\nGenerated sequences and rewards:\n")
                for sequence, reward in generated_sequences:
                    output_file.write(f"{sequence}\n{reward}\n")
                output_file.write("\n")
            output_file.write(str(highest_score) + "\n")
            if best_series is not None:
                output_file.write(" ".join(best_series) + "\n")
            else:
                output_file.write("No series found.\n")
            if best_coordinates is not None:
                for coordinates in best_coordinates:
                    output_file.write(f"{coordinates[1] + 1}, {coordinates[0] +
1}\n")
            else:
                output_file.write("No coordinates found.\n")
            output_file.write(f"{int((end_time - start_time) * 1000)} ms\n")

        print(f"Output saved to {output_file_path}")
```

Bab 4: Implementasi dan Uji Coba

4.1. Contoh 1

TUCIL 1 > src > input.txt

```
7
6 6
7A 55 E9 E9 1C 55
55 7A 1C 7A E9 55
55 1C 1C 55 E9 BD
BD 1C 7A 1C 55 BD
BD 55 BD 7A 1C 1C
1C 55 55 7A 55 7A
3
BD E9 1C
15
BD 7A BD
20
BD 1C BD 55
30
```

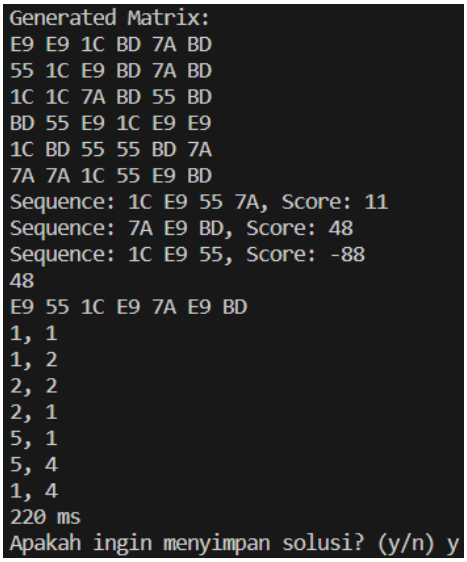


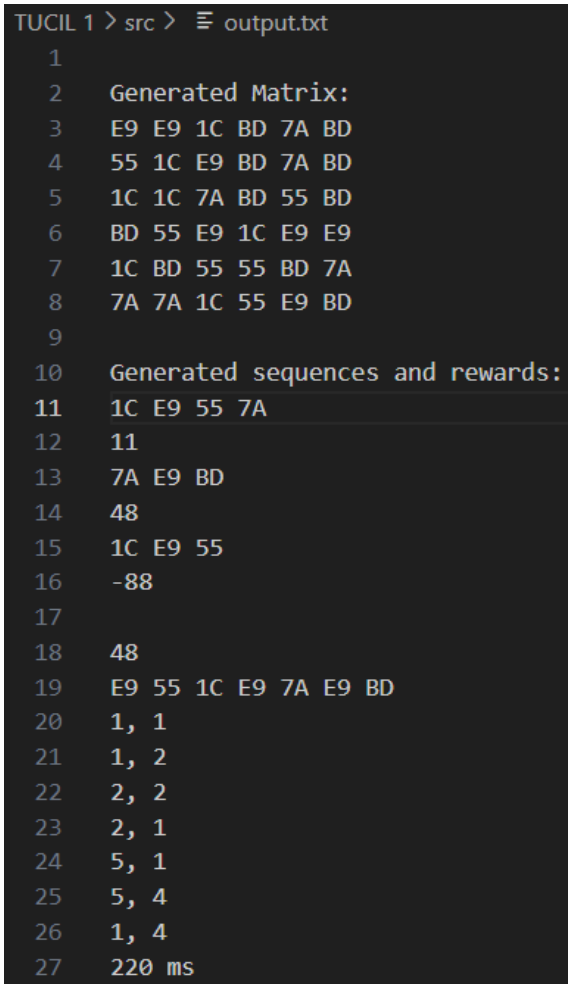
TUCIL 1 > src > output.txt

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

4.2. Contoh 2


1. Input dengan file .txt
2. Input melalui CLI
Pilih yang mana (1/2)? 2
Jumlah token: 5
Token (dipisahkan oleh spasi): BD 1C 7A 55 E9
Ukuran buffer: 7
Ukuran matriks (lebar dan tinggi dipisahkan oleh spasi): 6 6
Jumlah sekuens: 3
Ukuran maksimum sekuens: 4


Generated Matrix:
E9 E9 1C BD 7A BD
55 1C E9 BD 7A BD
1C 1C 7A BD 55 BD
BD 55 E9 1C E9 E9
1C BD 55 55 BD 7A
7A 7A 1C 55 E9 BD
Sequence: 1C E9 55 7A, Score: 11
Sequence: 7A E9 BD, Score: 48
Sequence: 1C E9 55, Score: -88
48
E9 55 1C E9 7A E9 BD
1, 1
1, 2
2, 2
2, 1
5, 1
5, 4
1, 4
220 ms
Apakah ingin menyimpan solusi? (y/n) y


TUCIL 1 > src > output.txt
Generated Matrix:
E9 E9 1C BD 7A BD
55 1C E9 BD 7A BD
1C 1C 7A BD 55 BD
BD 55 E9 1C E9 E9
1C BD 55 55 BD 7A
7A 7A 1C 55 E9 BD
Generated sequences and rewards:
1C E9 55 7A
11
7A E9 BD
48
1C E9 55
-88
48
E9 55 1C E9 7A E9 BD
1, 1
1, 2
2, 2
2, 1
5, 1
5, 4
1, 4
220 ms

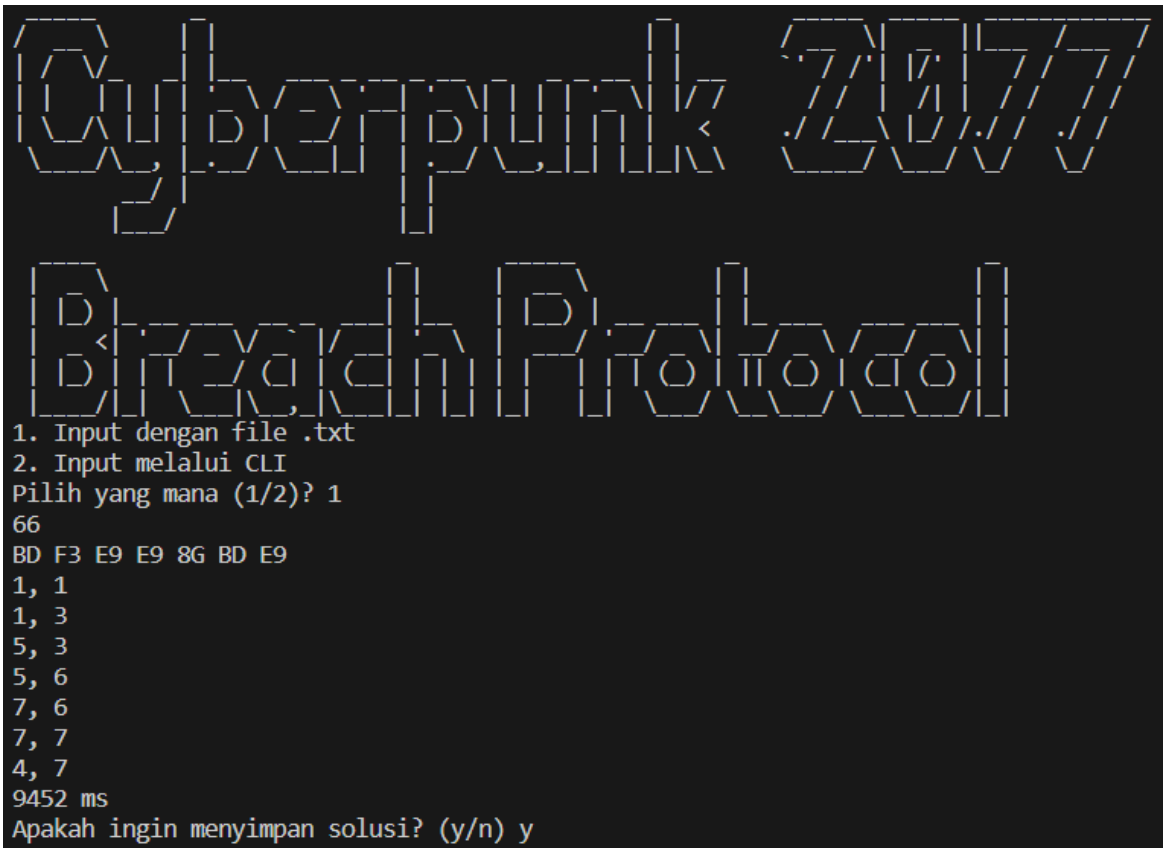
4.3. Contoh 3

TUCIL 1 > src > input.txt

```
7
10 8
BD 1C 55 55 F3 E9 55 F3 E9 8G
55 E9 1C 7A 8G 8G 55 E9 55 55
F3 55 F3 8G E9 1C 8G E9 F3 1C
E9 7A 1C F3 55 1C E9 7A 8G F3
1C 55 F3 7A 8G 8G F3 8G 55 8G
1C 8G BD E9 E9 BD 8G 7A 55 E9
1C 55 F3 E9 7A 1C BD 7A 7A F3
F3 1C 55 BD 55 1C 8G 55 1C 1C
4
F3 E9 E9
32
8G BD E9
34
BD BD F3 E9
24
E9 8G F3
20
```

TUCIL 1 > src > output.txt

```
66
BD F3 E9 E9 8G BD E9
1, 1
1, 3
5, 3
5, 6
7, 6
7, 7
4, 7
9452 ms
```



4.4. Contoh 4

TUCIL 1 > src > input.txt

```
6
6 6
BD 1C BD 1C BD 1C
55 7A 55 7A 55 7A
E9 FF E9 FF E9 FF
BD 1C BD 1C BD 1C
55 7A 55 7A 55 7A
E9 FF E9 FF E9 FF
3
1C FF
10
1C FF E9
20
7A 1C FF E9
30
```

TUCIL 1 > src > output.txt

```
30
BD BD 1C FF E9 55
1, 1
1, 4
2, 4
2, 3
1, 3
1, 2
39 ms
```

4.5. Contoh 5

TUCIL 1 > src > input.txt

```
9
7 7
BD 7A 7A E9 1C BD 7A
55 55 E9 E9 1C 1C 1C
BD 7A E9 1C BD 55 E9
55 7A E9 1C 1C BD 55
E9 E9 1C E9 55 55 BD
55 1C 1C E9 55 BD E9
1C 1C E9 55 BD E9 BD
3
1C BD
10
BD 55 E9
20
7A E9 1C BD
30
```

TUCIL 1 > src > output.txt

```
60
BD 55 55 7A E9 1C BD 55 E9
1, 1
1, 2
2, 2
2, 1
4, 1
4, 3
1, 3
1, 4
3, 4
35182 ms
```

```
1. Input dengan file .txt
2. Input melalui CLI
Pilih yang mana (1/2)? 1
60
BD 55 55 7A E9 1C BD 55 E9
1, 1
1, 2
2, 2
2, 1
4, 1
4, 3
1, 3
1, 4
3, 4
35182 ms
Apakah ingin menyimpan solusi? (y/n) y
```

4.6. Contoh 6

```
1. Input dengan file .txt
2. Input melalui CLI
Pilih yang mana (1/2)? 2
Jumlah token: 5
Token (dipisahkan oleh spasi): BD 1C 7A 55 E9
Ukuran buffer: 8
Ukuran matriks (lebar dan tinggi dipisahkan oleh spasi): 7 7
Jumlah sekuens: 2
Ukuran maksimum sekuens: 4

Generated Matrix:
E9 E9 E9 1C BD E9 7A
55 E9 BD E9 BD E9 BD
E9 7A 7A 55 7A 1C 1C
7A 7A 55 1C 1C BD 1C
7A E9 7A 1C 1C 55 7A
BD BD E9 BD E9 55 E9
1C 55 BD 55 1C 1C E9
Sequence: 1C 55, Score: -62
Sequence: 55 7A BD 1C, Score: -19
0
No series found.
No coordinates found.
6326 ms
Apakah ingin menyimpan solusi? (y/n) y
```

TUCIL 1 > src > output.txt

```
1
2   Generated Matrix:
3   E9 E9 E9 1C BD E9 7A
4   55 E9 BD E9 BD E9 BD
5   E9 7A 7A 55 7A 1C 1C
6   7A 7A 55 1C 1C BD 1C
7   7A E9 7A 1C 1C 55 7A
8   BD BD E9 BD E9 55 E9
9   1C 55 BD 55 1C 1C E9
10
11  Generated sequences and rewards:
12  1C 55
13  -62
14  55 7A BD 1C
15  -19
16
17  0
18  No series found.
19  No coordinates found.
20  6326 ms
```

Lampiran

Link Repository

https://github.com/ChristopherBrian/TUCIL-1-STIMA

Tabel *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		✓