**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**

[**Daftar Isi 2**](#_bwgya05f3dmb)

[**Bab 1: Algoritma *Brute Force* 3**](#_o78qho28cyoo)

[1.1. *Input*/*Output* 3](#_fa9asawn82ct)

[1.2. Generasi Acak 3](#_ue48wk7nss4d)

[1.2.1. Generasi Matriks Acak 3](#_k4j12u72h508)

[1.2.2. Generasi Sekuens Acak 3](#_ifczxr5yehhi)

[1.3. Pencarian Solusi dengan Algoritma *Brute Force* 4](#_ue48wk7nss4d)

[1.3.1. Generasi dan Iterasi *Series* 4](#_k4j12u72h508)

[1.3.2. Pengecekan Sekuens dalam *Series* 4](#_ifczxr5yehhi)

[1.3.3. Pencarian Solusi Terbaik 5](#_ifczxr5yehhi)

[**Bab 2: *Source Code* 6**](#_oxlh6215hv71)

[**Bab 3: Uji Coba 14**](#_syk9j4o53d35)

[4.1. Contoh 1 14](#_k219xk1ngrk0)

[4.2. Contoh 2 15](#_zdpljj9fz2hi)

[4.3. Contoh 3 16](#_v0sro8m4imzm)

[4.4. Contoh 4 17](#_5j4rro468ffb)

[4.5. Contoh 5 18](#_ngrec73ss7g2)

[4.6. Contoh 6 19](#_ngrec73ss7g2)

[**Lampiran 20**](#_a8pkkehm3kfn)

[Link Repository 20](#_cs1qg47fdbjf)

[Tabel *Checklist* 20](#_l4quh9bv67tc)

**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 generate­random\_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:

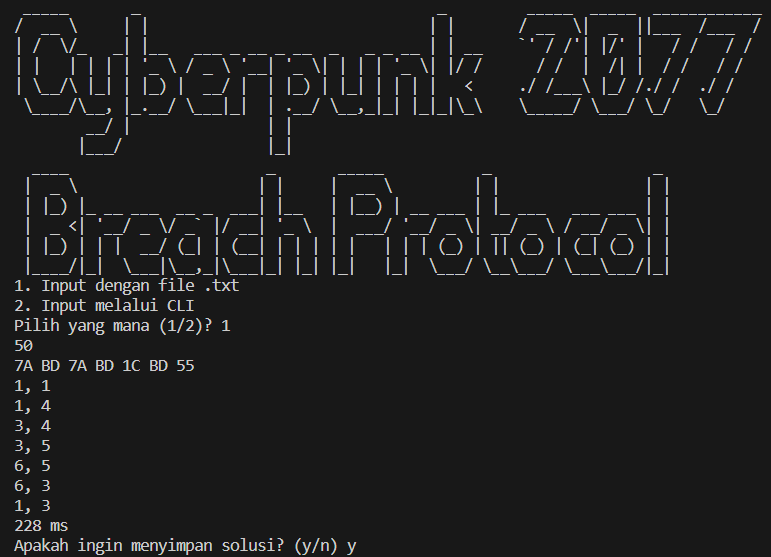
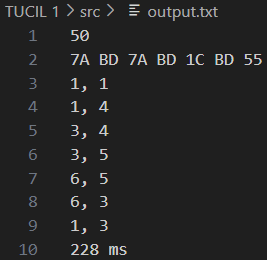
                output\_file.write("No coordinates found.\n")

            output\_file.write(f"{int((end\_time - start\_time) \* 1000)} ms\n")

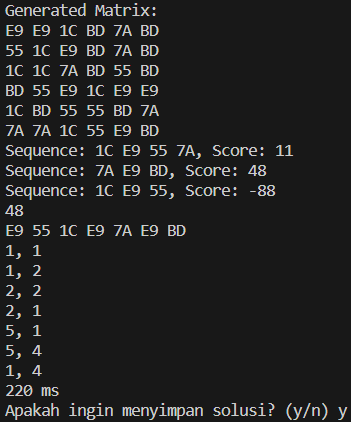
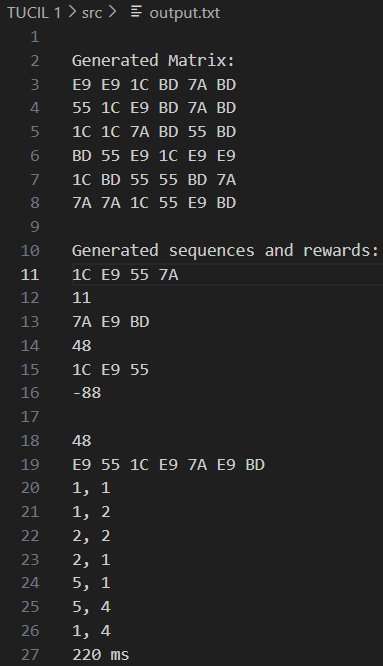
        print(f"Output saved to {output\_file\_path}")

# **Bab 4: Implementasi dan Uji Coba**

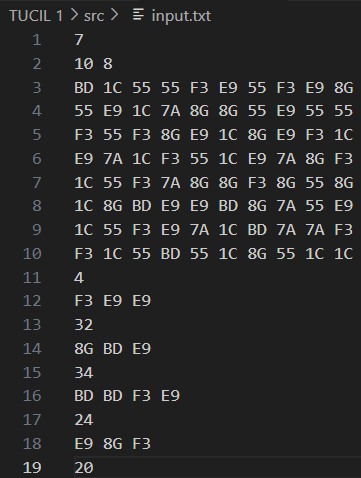
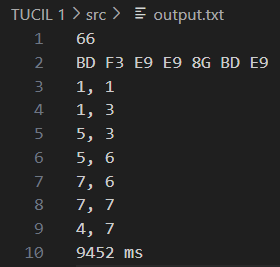
## 4.1. Contoh 1



## 4.2. Contoh 2

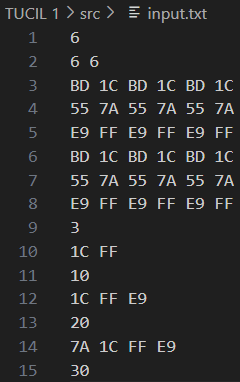


## 4.3. Contoh 3



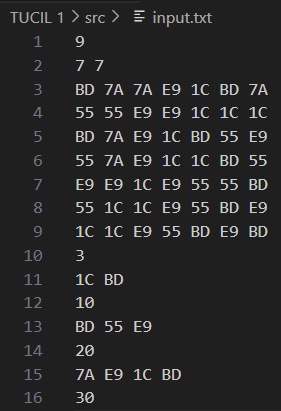
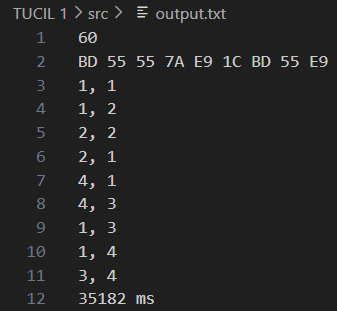


## 4.4. Contoh 4



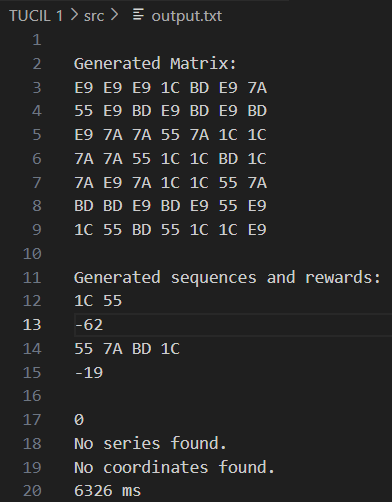


## 4.5. Contoh 5





## 4.6. Contoh 6



# **Lampiran**

## Link Repository

## <https://github.com/ChristopherBrian/TUCIL-1-STIMA>

## Tabel *Checklist*

**