

LAPORAN TUGAS KECIL 01
IF2211 STRATEGI ALGORITMA

Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force



Disusun oleh:

Denise Felicia Tiowanni (13522013)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2024

DAFTAR ISI

BAB I DESKRIPSI MASALAH DAN ALGORITMA.....	3
1.1 Algoritma Brute Force.....	3
1.2 Cyberpunk 2077 Breach Protocol.....	3
1.3 Algoritma Penyelesaian Cyberpunk 2077 Breach Protocol dengan Brute Force.....	4
BAB II IMPLEMENTASI PROGRAM.....	6
2.1 cyberpunk.py.....	6
2.2 helper.py.....	6
2.3 IO.py.....	7
BAB III SOURCE CODE.....	8
3.1 Github <i>Repository</i>	8
3.2 Source Code Program.....	8
3.2.1 cyberpunk.py.....	8
3.2.2 helper.py.....	11
3.2.3 IO.py.....	11
3.2.4 GUI.....	14
BAB IV IMPLEMENTASI DAN UJI COBA.....	22
4.1 Keyboard (CLI) Input.....	22
4.2 File Input.....	25
4.3 GUI.....	28
BAB V PENUTUP.....	33
5.1 Kesimpulan.....	33
5.2 Lampiran.....	33
DAFTAR PUSTAKA.....	34

BAB I

DESKRIPSI MASALAH DAN ALGORITMA

1.1 Algoritma Brute Force

Algoritma Brute Force merupakan suatu pendekatan yang lempang (langsung dan sederhana) dalam membantu memecahkan suatu persoalan. Biasanya, algoritma ini didasarkan pada pernyataan yang terdapat dalam persoalan dan definisi atau konsep yang terlibat. Pendekatan Brute Force ini memecahkan persoalan dengan cara yang sangat jelas dan langsung, tanpa memperhitungkan strategi yang rumit, di mana algoritma tersebut secara langsung mengeksekusi langkah-langkah yang diperlukan untuk menyelesaikan persoalan tanpa mempertimbangkan optimasi atau strategi yang lebih canggih.

Algoritma Brute Force umumnya tidak efisien karena membutuhkan volume komputasi yang besar dan waktu yang lama untuk menyelesaikan persoalan. Meskipun demikian, Algoritma Brute Force masih memiliki kegunaan, terutama untuk persoalan dengan ukuran masukan yang kecil. Algoritma ini dapat diterapkan pada hampir semua jenis persoalan, meski tidak selalu memberikan solusi yang optimal atau efisien. Bahkan terdapat beberapa persoalan yang hanya dapat diselesaikan dengan menggunakan Algoritma Brute Force, seperti mencari elemen terbesar dalam sebuah daftar.

1.2 Cyberpunk 2077 Breach Protocol

Cyberpunk 2077 Breach Protocol adalah minigame meretas pada permainan video Cyberpunk 2077. Minigame ini merupakan simulasi peretasan jaringan local dari ICE (Intrusion Countermeasures Electronics) pada permainan Cyberpunk 2077. Komponen pada permainan ini antara lain adalah:

1. Token – terdiri dari dua karakter alfanumerik seperti E9, BD, dan 55.
2. Matriks – terdiri atas token-token yang akan dipilih untuk menyusun urutan kode.

3. Sekuens – sebuah rangkaian token (dua atau lebih) yang harus dicocokkan.
4. Buffer – jumlah maksimal token yang dapat disusun secara sekuensial.

Permainan ini memiliki aturan yang mencakup pemain bergerak secara bergantian secara horizontal dan vertikal hingga mencocokkan semua sekuens atau buffer penuh. Pemain memulai dengan memilih satu token di baris teratas matriks, lalu mencocokkan sekuens pada token di buffer, di mana satu token dapat digunakan untuk lebih dari satu sekuens. Setiap sekuens memiliki bobot hadiah yang bervariasi, dengan panjang minimal dua token.

1.3 Algoritma Penyelesaian Cyberpunk 2077 Breach Protocol dengan Brute Force

Untuk menyelesaikan persoalan Cyberpunk 2077 Breach Protocol secara algoritmik dengan Algoritma Brute Force, berikut adalah langkah-langkah yang penulis lakukan.

1. Pembacaan Input

Program akan meminta pengguna untuk memilih jenis masukan, baik itu dari file ataupun dari keyboard. Jika pengguna memilih masukan dari file, program meminta nama file (.txt) dan membaca isi file tersebut menggunakan fungsi `read_file`. Jika pengguna memilih masukan dari keyboard, program meminta beberapa informasi seperti jumlah token unik, token, ukuran buffer, ukuran matriks, jumlah sekuens, ukuran maksimal sekuens, dan hadiah untuk setiap sekuens. Kemudian, program secara acak menghasilkan matriks dan sekuens untuk digunakan dalam algoritma.

2. Pencarian Semua Sekuens dan Koordinat

Program menggunakan fungsi `all_sequences` untuk mencari semua kemungkinan sekuens yang mungkin dari matriks yang diberikan. Algoritma ini menggunakan rekursi untuk mengeksplorasi setiap kemungkinan pergerakan ke atas atau ke samping untuk membentuk sekuens. Setiap sekuens dan koordinat yang ditemukan kemudian disimpan.

3. Menghitung Bobot Hadiah untuk Setiap Sekuens

Program menggunakan fungsi hadiah untuk menghitung hadiah (reward) untuk setiap sekuens berdasarkan sekuens yang ditemukan sebelumnya dan hadiah yang diberikan untuk masing-masing sekuens.

4. Memilih Sekuens dengan Bobot Hadiah Maksimal

Setelah mendapatkan semua hadiah untuk sekuens yang ditemukan, program mencari sekuens dengan bobot hadiah maksimal menggunakan fungsi `max_array`. Kemudian, program mencari indeks di mana bobot hadiah maksimal ditemukan menggunakan fungsi `find_index`.

5. Menyimpan Solusi

Jika terdapat sekuens dengan bobot hadiah maksimal yang tidak sama dengan 0, program akan menyimpan solusi ke dalam file menggunakan fungsi `write_ada_solusi`. Namun, jika tidak ada sekuens yang memenuhi kriteria, program akan menyimpan pesan bahwa tidak ada solusi menggunakan fungsi `write_no_solusi`.

Algoritma Brute Force akan secara iteratif mencoba semua kemungkinan sekuens yang mungkin dari matriks yang diberikan dan menghitung hadiah untuk setiap sekuens. Kemudian, algoritma memilih sekuens dengan bobot hadiah maksimal dan menyimpan solusi jika ada.

BAB II

IMPLEMENTASI PROGRAM

2.1 cyberpunk.py

File ini merupakan *driver* utama dari program, dimana program menyimpan fungsi dengan implementasi Algoritma Brute Force dalam mencari segala kemungkinan sekuens yang ada serta berisi menu utama dari program. Pada file ini, digunakan *library* yaitu ‘time’ untuk menghitung waktu eksekusi program.

Fungsi	Deskripsi
all_sequences	Mencari dan mengembalikan semua kemungkinan sekuens dari matriks serta menemukan titik koordinatnya.
hadiah	Mencari bobot hadiah setiap sekuens yang ada.

2.2 helper.py

File ini menyimpan fungsi-fungsi pembantu yang digunakan dalam file *cyberpunk.py*, diantaranya adalah fungsi *max_array* dan *find_index*.

Fungsi	Deskripsi
max_array	Mencari nilai maksimum dalam suatu array.
find_index	Mencari index dari suatu nilai dalam array.

1.3 IO.py

File ini menyimpan fungsi-fungsi input dan output yang digunakan dalam file *cyberpunk.py*, seperti fungsi yang menangani input *keyboard*, *file*, serta *formatting* output.

Fungsi	Deskripsi
<code>read_file</code>	Membaca dan mengembalikan <code>buffer_size</code> , <code>matrix_width</code> , <code>matrix_height</code> , <code>matrix</code> , <code>number_of_sequences</code> , <code>sequences</code> , <code>sequence_rewards</code> dari sebuah file (.txt).
<code>keyboard_input</code>	Membaca dan mengembalikan <code>jumlah_token_unik</code> , <code>token</code> , <code>buffer_size</code> , <code>matrix_width</code> , <code>matrix_height</code> , <code>matrix</code> , <code>number_of_sequences</code> , <code>ukuran_maksimal_sekuens</code> , <code>sequences</code> , <code>sequence_rewards</code> dari <i>keyboard</i> .
<code>write_to_file</code>	Menuliskan <i>array of array</i> ke file (.txt).
<code>write_array</code>	Menuliskan <i>array</i> ke file (.txt).
<code>write_ada_solusi</code>	Menuliskan <i>output</i> dengan format tertentu ketika bobot hadiah maksimal ditemukan.
<code>write_no_solusi</code>	Menuliskan <i>output</i> dengan format tertentu ketika bobot hadiah maksimal tidak ditemukan.

BAB III

SOURCE CODE

3.1 Github Repository

Segala algoritma dan source code program dapat diakses pada Github dengan tautan sebagai berikut: https://github.com/denoseu/Tucil1_13522013.

3.2 Source Code Program

3.2.1 cyberpunk.py

```
import time
from IO import read_file, keyboard_input, write_to_file, write_array,
write_ada_solusi, write_no_solusi
from helper import max_array, find_index

#*** mencari semua sequence yang mungkin ***
def all_sequences(matrix, buffer_size):
    def is_valid_move(row, col):
        return 0 <= row < len(matrix) and 0 <= col < len(matrix[0]) and (row, col) not in visited

    def explore(row, col, buffer_index, is_vertical):
        visited.add((row, col))
        current_token.append(matrix[row][col])
        current_coordinate.append((col+1, row+1))
        if buffer_index == buffer_size - 1:
            coordinates.append(current_coordinate[:])
            sequences.append(current_token[:])
        else:
            directions = [(1, 0), (-1, 0)] if is_vertical else [(0, 1), (0, -1)]
            for dr, dc in directions:
                new_row, new_col = row, col
                for _ in range(buffer_size - buffer_index):
                    new_row, new_col = new_row + dr, new_col + dc
                    if is_valid_move(new_row, new_col):
                        explore(new_row, new_col, buffer_index + 1,
                                not is_vertical)

        visited.remove((row, col))
        current_token.pop()
        current_coordinate.pop()

    explore(0, 0, 0, True)
```



```

input_type = input("Pilih jenis masukan (1/2): ")

if input_type == '1':
    filename = input("Masukkan nama file (.txt): ")
    buffer_size, matrix_width, matrix_height, matrix,
    number_of_sequences, sequences, sequence_rewards = read_file(filename)
else:
    jumlah_token_unik, token, buffer_size, matrix_width,
    matrix_height, matrix, number_of_sequences, ukuran_maksimal_sekuens,
    sequences, sequence_rewards = keyboard_input()
    # print("Jumlah token unik:", jumlah_token_unik)
    # print("Token:", token)
    # print("Ukuran buffer:", buffer_size)
    # print("Matrix (height):", matrix_height)
    # print("Matrix (width):", matrix_width)
    print("Matrix: ")
    for i in range(matrix_height):
        for j in range(matrix_width):
            print(matrix[i][j], end=' ')
    print()
    print("Jumlah sekuens: ", number_of_sequences)

    for i in range((number_of_sequences)):
        sequence = sequences[i]
        print(sequence)
        print("Hadiah: ", sequence_rewards[i])

start = time.time()

cari_sequences, cari_coordinates = all_sequences(matrix, buffer_size)

total_hadiah = hadiah(cari_sequences, sequences, sequence_rewards)
bobot_hadiah_max = max_array(total_hadiah)
print("Bobot Hadiah: ", bobot_hadiah_max)

# mencari indeks dimana bobot hadiah maksimal ditemukan
index = find_index(total_hadiah, bobot_hadiah_max)

end = time.time()

execution_time = round(((end - start)*1000), 2)

# write_to_file(cari_coordinates, 'output_coordinate.txt')
# write_to_file(cari_sequences, 'output.txt')
# write_array(total_hadiah, 'output2.txt')

if bobot_hadiah_max != 0:
    print("Sekuens: ", end=' ')

```

3.2.2 helper.py

```
*** mencari nilai maksimum dalam suatu array ***
def max_array(array):
    max = array[0]
    for i in range(len(array)):
        if array[i] > max:
            max = array[i]
    return max

*** find index ***
def find_index(array, value):
    for i in range(len(array)):
        if array[i] == value:
            return i+1
    return -1
```

3.2.3 IO.py

```
import random

*** read file txt ***
def read_file(filename):
```

```

        with open(filename, 'r') as file:
            buffer_size = int(file.readline().strip())
            matrix_width, matrix_height = map(int, file.readline().split())
            matrix = [list(file.readline().split()) for _ in
                      range(matrix_height)]
            number_of_sequences = int(file.readline().strip())
            sequences = []
            sequence_rewards = []

            for _ in range(number_of_sequences):
                sequences.append(file.readline().split())
                sequence_rewards.append(int(file.readline().strip()))

            return buffer_size, matrix_width, matrix_height, matrix,
            number_of_sequences, sequences, sequence_rewards

    #*** read from keyboard ***
def keyboard_input():
    jumlah_token_unik = input("Jumlah token unik: ")
    token_input = input("Masukkan token: ")
    token = token_input.split()

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

    ukuran_matriks = input("Ukuran matriks: ")
    col_mtx, row_mtx = ukuran_matriks.split()
    matrix_width = int(col_mtx)
    matrix_height = int(row_mtx)

    matrix = [['' for i in range(matrix_width)] for j in
              range(matrix_height)]
    for j in range (matrix_height):
        for k in range (matrix_width):
            matrix[j][k] = random.choice(token)

    number_of_sequences = int(input("Jumlah sekuens: "))
    ukuran_maksimal_sekuens = int(input("Ukuran maksimal sekuens:
    "))

    sequences = []
    for a in range(int(number_of_sequences)):
        ukuran = random.randint(2, int(ukuran_maksimal_sekuens))
        sequence = []
        for b in range(ukuran):
            sequence.append(random.choice(token))
        sequences.append(sequence)

```

```

sequence_rewards = []
for c in range(int(number_of_sequences)):
    sequence_rewards.append(random.randint(8, 80))

    return jumlah_token_unik, token, buffer_size, matrix_width,
matrix_height, matrix, number_of_sequences, ukuran_maksimal_sekuens,
sequences, sequence_rewards

#*** write array of array ke file txt ***#
def write_to_file(sequences, filename):
    with open(filename, 'w') as file:
        for seq in sequences:
            for elem in seq:
                file.write(str(elem) + ' ')
            file.write('\n')

#*** write array ke file txt ***#
def write_array(sequences, filename):
    with open(filename, 'w') as file:
        for seq in sequences:
            file.write(str(seq) + ' ')
            file.write('\n')

#*** write file solusi ketika bobot hadiah != 0 ***#
def write_ada_solusi(index, cari_sequences, cari_coordinates,
bobot_hadiah_max, execution_time):
    filename = input("Masukkan nama file solusi: ")
    with open('test/' + filename + '.txt', 'w') as file:
        file.write("#**** Cyberpunk 2077 Breach Protocol Solution ****#
\n")
        file.write("Bobot Hadiah: " + str(bobot_hadiah_max) + '\n')
        file.write("Sekuens: ")
        for token in cari_sequences[index-1]:
            file.write(token + ' ')
        file.write('\n')
        file.write("Koordinat: \n")
        for coordinates in cari_coordinates[index-1]:
            file.write(str(coordinates) + '\n')
        file.write("Waktu eksekusi: " + str(execution_time) + " ms\n")
        file.write("#*****\n")
    return filename

#*** write file solusi ketika bobot hadiah = 0 ***#
def write_no_solusi(bobot_hadiah_max, execution_time):
    filename = input("Masukkan nama file solusi: ")
    with open('test/' + filename + '.txt', 'w') as file:

```

```

        file.write("***** Cyberpunk 2077 Breach Protocol Solution *****\n")
        file.write("Bobot Hadiah: " + str(bobot_hadiah_max) + '\n')
        file.write("Tidak ada sekuens yang memenuhi. \n")
        file.write("Waktu eksekusi: " + str(execution_time) + " ms\n")
        file.write("#*****\n")
        return filename
    
```

3.2.4 GUI.py

```

import tkinter as tk
from tkinter import simpledialog, filedialog
from PIL import Image, ImageTk, ImageDraw, ImageFont
import random, time, os

class GUI(tk.Tk):
    def max_array(array):
        max = array[0]
        for i in range(len(array)):
            if array[i] > max:
                max = array[i]
        return max

    def find_index(array, value):
        for i in range(len(array)):
            if array[i] == value:
                return i+1
        return -1

    def all_sequences(matrix, buffer_size):
        def is_valid_move(row, col):
            return 0 <= row < len(matrix) and 0 <= col <
len(matrix[0]) and (row, col) not in visited

        def explore(row, col, buffer_index, is_vertical):
            visited.add((row, col))
            current_token.append(matrix[row][col])
            current_coordinate.append((col+1, row+1))
            if buffer_index == buffer_size - 1:
                coordinates.append(current_coordinate[:])
                sequences.append(current_token[:])
            else:
                directions = [(1, 0), (-1, 0)] if is_vertical else [(0,
1), (0, -1)]
                for direction in directions:
                    new_row = row + direction[0]
                    new_col = col + direction[1]
                    if is_valid_move(new_row, new_col):
                        explore(new_row, new_col, buffer_index + 1, is_ver
tical)
        explore(0, 0, 0, True)
        return sequences
    
```

```

        for dr, dc in directions:
            new_row, new_col = row, col
            for _ in range(buffer_size - buffer_index):
                new_row, new_col = new_row + dr, new_col + dc
                if is_valid_move(new_row, new_col):
                    explore(new_row, new_col, buffer_index + 1,
not is_vertical)

                    visited.remove((row, col))
                    current_token.pop()
                    current_coordinate.pop()

sequences = []
coordinates = []
current_coordinate = []
current_token = []
visited = set()
for col_index in range(len(matrix[0])):
    explore(0, col_index, 0, True)
return sequences, coordinates

def hadiah(matrix, sequences, sequence_rewards):
total_rewards = []

for array in matrix:
    total_reward = 0
    for i in range(len(sequences)):
        sequence = sequences[i]
        reward = sequence_rewards[i]

        array_string = ' '.join(array)
        sequence_string = ' '.join(sequence)

        if sequence_string in array_string:
            total_reward += reward

    total_rewards.append(total_reward)

return total_rewards

def calculate(self, buffer_value, matrix_width, matrix_height,
matrix, no_of_sequences, sequences, sequence_rewards):
    start = time.time()
    cari_sequences, cari_coordinates = GUI.all_sequences(matrix,
buffer_value)
    rewards = GUI.hadiah(cari_sequences, sequences,
sequence_rewards)

```

```

max_reward = GUI.max_array(rewards)
index = GUI.find_index(rewards, max_reward)
end = time.time()
execution_time = end - start

print("Matrix: ")
for i in range(matrix_height):
    for j in range(matrix_width):
        print(matrix[i][j], end=' ')
    print()
print("Jumlah sekuens: ", no_of_sequences)

for i in range((no_of_sequences)):
    sequence = sequences[i]
    print(sequence)
    print("Hadiah: ", sequence_rewards[i])

print("Bobot Hadiah:", max_reward)
print("Sekuens: ", cari_sequences[index-1])
print("Koordinat: ", cari_coordinates[index-1])
print("Execution Time: ", execution_time*1000, "ms")

    self.result(matrix, cari_sequences, cari_coordinates,
max_reward, index, execution_time)

    return matrix, cari_sequences, cari_coordinates, max_reward,
index, execution_time

def result(self, matrix, cari_sequences, cari_coordinates,
max_reward, index, execution_time, event=None):
    self.canvas.delete("all")

    background_img = Image.open("src/assets/resultbg.png")
    background_img = background_img.resize((1500, 810))
    self.background_photo = ImageTk.PhotoImage(background_img)
    self.canvas.create_image(0, 0, image=self.background_photo,
anchor="nw")

    header_img = Image.open("src/assets/result_header.png")
    header_img = header_img.resize((1435, 260))
    self.header_photo = ImageTk.PhotoImage(header_img)
    self.canvas.create_image(0, 10, image=self.header_photo,
anchor="nw")

    matrix_img = Image.open("src/assets/matrix.png")
    matrix_img = matrix_img.resize((600, 530))

```

```

        self.matrix_photo = ImageTk.PhotoImage(matrix_img)
        self.canvas.create_image(40, 210, image=self.matrix_photo,
anchor="nw")

        cell_width = 230 / len(matrix[0])
        cell_height = 230 / len(matrix)
        for i in range(len(matrix)):
            for j in range(len(matrix[0])):
                x0 = 150 + j * cell_width
                y0 = 350 + i * cell_height
                x1 = x0 + cell_width
                y1 = y0 + cell_height
                self.canvas.create_rectangle(x0, y0, x1, y1, fill="white",
outline="black")
                self.canvas.create_text((x0 + x1) / 2, (y0 + y1) / 2,
text=str(matrix[i][j]))

        seq_result_img = Image.open("src/assets/seq_result.png")
        seq_result_img = seq_result_img.resize((600, 280))
        self.seq_result_photo = ImageTk.PhotoImage(seq_result_img)
        self.canvas.create_image(750, 180, image=self.seq_result_photo,
anchor="nw")

        reward_result_img = Image.open("src/assets/reward_result.png")
        reward_result_img = reward_result_img.resize((250, 180))
        self.reward_result_photo = ImageTk.PhotoImage(reward_result_img)
        self.canvas.create_image(750, 470,
image=self.reward_result_photo, anchor="nw")

        time_result_img = Image.open("src/assets/time_result.png")
        time_result_img = time_result_img.resize((250, 180))
        self.time_result_photo = ImageTk.PhotoImage(time_result_img)
        self.canvas.create_image(1080, 470,
image=self.time_result_photo, anchor="nw")

        x_offset, y_offset = 1, 1
        self.canvas.create_text(920 + x_offset, 380 + y_offset,
text=cari_sequences[index-1], font=("Arial", 22), fill="#FFD11A")
        self.canvas.create_text(920, 380, text=cari_sequences[index-1],
font=("Arial", 22), fill="#2C75D4")

        self.canvas.create_text(870 + x_offset, 590 + y_offset,
text=max_reward, font=("Arial", 22), fill="#FFD11A")
        self.canvas.create_text(870, 590, text=max_reward,
font=("Arial", 22), fill="#2C75D4")

        self.canvas.create_text(1200 + x_offset, 580 + y_offset,

```

```

text=(round(execution_time*1000)), font=("Arial", 22), fill="#FFD11A")
        self.canvas.create_text(1200, 580,
text=(round(execution_time*1000)), font=("Arial", 22), fill="#2C75D4")

        self.canvas.create_text(1200 + x_offset, 600 + y_offset,
text="ms", font=("Arial", 22), fill="#FFD11A")
        self.canvas.create_text(1200, 600, text="ms", font=("Arial",
22), fill="#2C75D4")

        save_img = Image.open("src/assets/save.png")
        save_img = save_img.resize((300, 50))
        self.save_photo = ImageTk.PhotoImage(save_img)
        self.save_image = self.canvas.create_image(890, 680,
image=self.save_photo, anchor="nw")
        self.canvas.tag_bind(self.save_image, "<Button-3>",
self.save_input_clicked(matrix, cari_sequences, cari_coordinates,
max_reward, index, execution_time))

def save_input_clicked(self, matrix, cari_sequences,
cari_coordinates, max_reward, index, execution_time, event=None):
    file_path =
filedialog.asksaveasfilename(defaultextension=".txt", filetypes=[("Text
files", ".*.txt")])
    print(file_path)
    if file_path:
        if max_reward != 0:
            with open(file_path, 'w') as file:
                file.write("***** Cyberpunk 2077 Breach Protocol
Solution ****# \n")
                file.write("Bobot Hadiah: " + str(max_reward) +
'\n')
                file.write("Sekuens: ")
                for token in cari_sequences[index-1]:
                    file.write(token + ' ')
                file.write('\n')
                file.write("Koordinat: \n")
                for coordinates in cari_coordinates[index-1]:
                    file.write(str(coordinates) + '\n')
                file.write("Waktu eksekusi: " +
str(execution_time*1000) + " ms\n")

            file.write("*****\n")
        else:
            with open(file_path, 'w') as file:
                file.write("***** Cyberpunk 2077 Breach Protocol
Solution ****# \n")

```

```

        file.write("Bobot Hadiah: " + str(max_reward) +
'\n')
        file.write("Tidak ada sekuens yang memenuhi. \n")
        file.write("Waktu eksekusi: " +
str(execution_time*1000) + " ms\n")

file.write("#*****\n")
    print("File saved successfully:", file_path)
else:
    print("Save operation cancelled.")

def file_input_clicked(self, event=None):
    file_path = filedialog.askopenfilename()
    if file_path:
        file_name = os.path.basename(file_path)
        print("File selected:", file_name)
        with open(file_name, 'r') as file:
            buffer_size = int(file.readline().strip())
            matrix_width, matrix_height = map(int,
file.readline().split())
            matrix = [list(file.readline().split()) for _ in
range(matrix_height)]
            number_of_sequences = int(file.readline().strip())
            sequences = []
            sequence_rewards = []

        for _ in range(number_of_sequences):
            sequences.append(file.readline().split())

sequence_rewards.append(int(file.readline().strip()))

        GUI.calculate(self, buffer_size, matrix_width,
matrix_height, matrix, number_of_sequences, sequences,
sequence_rewards)

    else:
        print("No file selected.")

def keyboard_input_clicked(self, event=None):
    no_of_tokens = simpledialog.askstring("Input", "Jumlah token
unik:")
    tokens = simpledialog.askstring("Input", "Token (ex: AA BB
CC):")
    tokens = tokens.split()

```

```

buffer_value = simpledialog.askstring("Input", "Ukuran buffer:")
buffer_value = int(buffer_value)

matrix_size = simpledialog.askstring("Input", "Ukuran matriks:")
col_mtx, row_mtx = matrix_size.split()
matrix_width = int(row_mtx)
matrix_height = int(col_mtx)

matrix = [[['' for i in range(matrix_width)] for j in
range(matrix_height)]
for j in range (matrix_height):
    for k in range (matrix_width):
        matrix[j][k] = random.choice(tokens)

no_of_sequences = simpledialog.askstring("Input", "Jumlah
sekuens:")
no_of_sequences = int(no_of_sequences)

max_tokens_per_sequences = simpledialog.askstring("Input",
"Ukuran maksimal sekuens:")
max_tokens_per_sequences = int(max_tokens_per_sequences)

sequences = []
for a in range(int(no_of_sequences)):
    ukuran = random.randint(2, int(max_tokens_per_sequences))
    sequence = []
    for b in range(ukuran):
        sequence.append(random.choice(tokens))
    sequences.append(sequence)

sequence_rewards = []
for c in range(int(no_of_sequences)):
    sequence_rewards.append(random.randint(8, 80))

GUI.calculate(self, buffer_value, matrix_width, matrix_height,
matrix, no_of_sequences, sequences, sequence_rewards)

def __init__(self, *args, **kwargs):
super().__init__(*args, **kwargs)

self.title("Cyberpunk 2077 Breach Protocol Solver")

self.geometry("2560x1600")

self.canvas = tk.Canvas(self, width=2560, height=1600)
self.canvas.pack(fill=tk.BOTH, expand=True)

```

```
background_img = Image.open("src/assets/homebg.png")
background_img = background_img.resize((1500, 810))
self.background_photo = ImageTk.PhotoImage(background_img)
self.canvas.create_image(0, 0, image=self.background_photo,
anchor="nw")

header_img = Image.open("src/assets/header.png")
header_img = header_img.resize((1435, 325))
self.header_photo = ImageTk.PhotoImage(header_img)
self.canvas.create_image(0, 100, image=self.header_photo,
anchor="nw")

file_img = Image.open("src/assets/fileinput.png")
file_img = file_img.resize((280, 130))
self.file_photo = ImageTk.PhotoImage(file_img)
self.file_image = self.canvas.create_image(350, 500,
image=self.file_photo, anchor="nw")
self.canvas.tag_bind(self.file_image, "<Button-1>",
self.file_input_clicked)

keyboard_img = Image.open("src/assets/keyboardinput.png")
keyboard_img = keyboard_img.resize((280, 130))
self.keyboard_photo = ImageTk.PhotoImage(keyboard_img)
self.keyboard_image = self.canvas.create_image(800, 500,
image=self.keyboard_photo, anchor="nw")
self.canvas.tag_bind(self.keyboard_image, "<Button-1>",
self.keyboard_input_clicked)

if __name__ == "__main__":
    app = GUI()
    app.mainloop()
```

BAB IV

IMPLEMENTASI DAN UJI COBA

4.1 File Input



```
Input File: input1.txt

Masukkan nama file (.txt): input1.txt
Bobot Hadiah: 50
Sekuens: 7A BD 7A BD 1C BD 55
Koordinat:
(1, 1)
(1, 4)
(3, 4)
(3, 5)
(6, 5)
(6, 4)
(5, 4)
Waktu eksekusi: 101.26 ms
Apakah ingin menyimpan solusi? (y/n) y
Masukkan nama file solusi: 1
Solusi telah disimpan dalam file 1.txt!

Hasil Penyimpanan Jawaban

1.txt
***** Cyberpunk 2077 Breach Protocol Solution ****#
Bobot Hadiah: 50
Sekuens: 7A BD 7A BD 1C BD 55
Koordinat:
(1, 1)
(1, 4)
(3, 4)
(3, 5)
(6, 5)
(6, 4)
(5, 4)
Waktu eksekusi: 101.26 ms
*****#
```

Input File: **input2.txt**

```
Masukkan nama file (.txt): input2.txt
Bobot Hadiah: 9
Sekuens: CC CC CC
Koordinat:
(2, 1)
(2, 4)
(3, 4)
Waktu eksekusi: 0.2 ms
Apakah ingin menyimpan solusi? (y/n) y
Masukkan nama file solusi: 2
Solusi telah disimpan dalam file 2.txt!
```

Hasil Penyimpanan Jawaban

2.txt

```
***** Cyberpunk 2077 Breach Protocol Solution ****#
Bobot Hadiah: 9
Sekuens: CC CC CC
Koordinat:
(2, 1)
(2, 4)
(3, 4)
Waktu eksekusi: 0.2 ms
*****#
```

Input File: **input3.txt**

```
Masukkan nama file (.txt): input3.txt
Bobot Hadiah: 50
Sekuens: 7A BD 7A BD 1C BD 55
Koordinat:
(1, 1)
(1, 4)
(3, 4)
(3, 5)
(6, 5)
(6, 7)
(7, 7)
Waktu eksekusi: 999.21 ms
Apakah ingin menyimpan solusi? (y/n) y
Masukkan nama file solusi: 3
Solusi telah disimpan dalam file 3.txt!
```

Hasil Penyimpanan Jawaban

3.txt

```
***** Cyberpunk 2077 Breach Protocol Solution ****#
Bobot Hadiah: 50
Sekuens: 7A BD 7A BD 1C BD 55
Koordinat:
(1, 1)
(1, 4)
(3, 4)
(3, 5)
(6, 5)
(6, 7)
(7, 7)
Waktu eksekusi: 999.21 ms
*****#
```

Input File: **input4.txt**

```
Masukkan nama file (.txt): input4.txt
Bobot Hadiah: 40
Sekuens: 7A 55 7A 55 BD E9 1C
Koordinat:
(1, 1)
(1, 2)
(2, 2)
(2, 5)
(3, 5)
(3, 8)
(5, 8)
Waktu eksekusi: 128.32 ms
Apakah ingin menyimpan solusi? (y/n) y
Masukkan nama file solusi: 4
Solusi telah disimpan dalam file 4.txt!
```

Hasil Penyimpanan Jawaban

```
4.txt
***** Cyberpunk 2077 Breach Protocol Solution ****#
Bobot Hadiah: 40
Sekuens: 7A 55 7A 55 BD E9 1C
Koordinat:
(1, 1)
(1, 2)
(2, 2)
(2, 5)
(3, 5)
(3, 8)
(5, 8)
Waktu eksekusi: 128.32 ms
*****#
```

4.2 Keyboard (CLI) Input



Input 1

```
Jumlah token unik: 5
Masukkan token: A1 B2 C3 D4 E5
Ukuran buffer: 8
Ukuran matriks: 8 8
Jumlah sekuens: 4
Ukuran maksimal sekuens: 4
Matrix:
C3 E5 D4 A1 D4 C3 E5 B2
B2 B2 C3 C3 D4 E5 A1 B2
C3 C3 E5 B2 A1 B2 A1 B2
A1 D4 C3 E5 E5 B2 A1 A1
B2 C3 E5 B2 D4 C3 E5
E5 C3 E5 A1 B2 E5 A1 A1
C3 C3 B2 E5 B2 A1 D4 A1
D4 C3 E5 E5 C3 E5 D4 D4
Jumlah sekuens: 4
['C3', 'D4', 'B2', 'E5']
Hadiah: 16
['D4', 'C3', 'B2', 'D4']
Hadiah: 15
['A1', 'B2', 'E5', 'E5']
Hadiah: 71
['E5', 'C3']
Hadiah: 18
Bobot Hadiah: 105
Sekuens: A1 B2 E5 E5 C3 D4 B2 E5
Koordinat:
(4, 1)
(4, 3)
(3, 3)
(3, 8)
(5, 8)
(5, 5)
(4, 5)
(4, 7)
Waktu eksekusi: 5133.5 ms
Apakah ingin menyimpan solusi? (y/n) y
Masukkan nama file solusi: keyboard1
Solusi telah disimpan dalam file keyboard1.txt!
```

Hasil Penyimpanan Jawaban

```
📄 keyboard1.txt
***** Cyberpunk 2077 Breach Protocol Solution ****#
Bobot Hadiah: 105
Sekuens: A1 B2 E5 E5 C3 D4 B2 E5
Koordinat:
(4, 1)
(4, 3)
(3, 3)
(3, 8)
(5, 8)
(5, 5)
(4, 5)
(4, 7)
Waktu eksekusi: 5133.5 ms
*****#
```

Input 2

```
Jumlah token unik: 7
Masukkan token: AA BB CC DD EE FF GG
Ukuran buffer: 4
Ukuran matriks: 3 4
Jumlah sekuens: 1
Ukuran maksimal sekuens: 4
Matrix:
AA AA GG
CC FF EE
GG EE BB
BB AA EE
Jumlah sekuens: 1
['AA', 'GG']
Hadiah: 62
Bobot Hadiah: 62
Sekuens: AA GG EE AA
Koordinat:
(1, 1)
(1, 3)
(2, 3)
(2, 4)
Waktu eksekusi: 0.28 ms
Apakah ingin menyimpan solusi? (y/n) y
Masukkan nama file solusi: keyboard2
Solusi telah disimpan dalam file keyboard2.txt!
```

Hasil Penyimpanan Jawaban

```
📄 keyboard2.txt
***** Cyberpunk 2077 Breach Protocol Solution ****#
Bobot Hadiah: 62
Sekuens: AA GG EE AA
Koordinat:
(1, 1)
(1, 3)
(2, 3)
(2, 4)
Waktu eksekusi: 0.28 ms
*****#
```

Input 3

```
Jumlah token unik: 5
Masukkan token: A2 B4 C6 D8 E3
Ukuran buffer: 6
Ukuran matriks: 6 3
Jumlah sekuens: 3
Ukuran maksimal sekuens: 4
Matrix:
A2 D8 D8 C6 B4 D8
C6 D8 E3 B4 D8 A2
C6 A2 D8 C6 A2 A2
Jumlah sekuens: 3
['E3', 'E3']
Hadiah: 39
['E3', 'C6', 'B4']
Hadiah: 11
['B4', 'E3', 'E3', 'D8']
Hadiah: 54
Bobot Hadiah: 0
Apakah ingin menyimpan solusi? (y/n) y
Masukkan nama file solusi: keyboard3
Solusi telah disimpan dalam file keyboard3.txt!
```

Hasil Penyimpanan Jawaban

```
📄 keyboard3.txt
***** Cyberpunk 2077 Breach Protocol Solution *****
Bobot Hadiah: 0
Tidak ada sekuens yang memenuhi.
Waktu eksekusi: 6.46 ms
*****#
```

Input 4

```
Jumlah token unik: 3
Masukkan token: AB CD E8
Ukuran buffer: 5
Ukuran matriks: 3 7
Jumlah sekuens: 5
Ukuran maksimal sekuens: 3
Matrix:
CD CD E8
CD E8 E8
E8 E8 E8
E8 AB AB
AB AB AB
CD E8 AB
E8 E8 E8
Jumlah sekuens: 5
['E8', 'CD']
Hadiah: 50
['E8', 'AB']
Hadiah: 42
['E8', 'E8']
Hadiah: 56
['E8', 'AB']
Hadiah: 19
['CD', 'AB', 'AB']
Hadiah: 42
Bobot Hadiah: 167
Sekuens: E8 E8 CD E8 AB
Koordinat:
(3, 1)
(3, 2)
(1, 2)
(1, 4)
(2, 4)
Waktu eksekusi: 2.36 ms
Apakah ingin menyimpan solusi? (y/n) y
Masukkan nama file solusi: keyboard4
Solusi telah disimpan dalam file keyboard4.txt!
```

Hasil Penyimpanan Jawaban

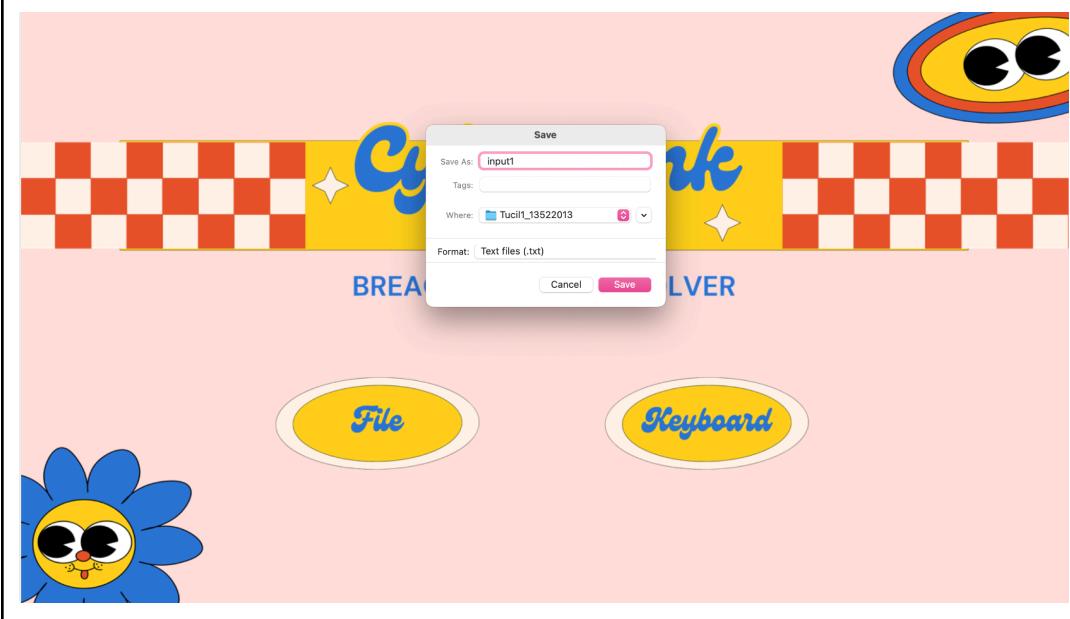
```
📄 keyboard4.txt
***** Cyberpunk 2077 Breach Protocol Solution ****#
Bobot Hadiah: 167
Sekuens: E8 E8 CD E8 AB
Koordinat:
(3, 1)
(3, 2)
(1, 2)
(1, 4)
(2, 4)
Waktu eksekusi: 2.36 ms
*****#
```

4.3 GUI

Home Page



Input: File



Result

Matrix

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

Sequence

7A BD 7A BD 1C BD 55

Reward

50

Time

105 ms

[save answer](#)

Cyberpunk 2077

Input: Keyboard

BREAKDOWN SOLVER

File

Keyboard

Input

Jumlah token unik:

OK **Cancel**



BREACH PROTOCOL SOLVER

File **Keyboard**

Result

Matrix

C3	D4	C3	D4	E5	E5	B2	C3
D4	B2	B2	E5	B2	D4	C3	C3
B2	E5	A1	C3	C3	B2	B2	A1
E5	D4	B2	D4	E5	C3	D4	B2
B2	A1	E5	C3	B2	C3	D4	D4
E5	C3	B2	E5	B2	C3	C3	B2
E5	A1	E5	A1	B2	D4	E5	D4

Sequence

C3 A1 B2 C3 D4

Reward

132

Time

16 ms

[save answer](#)

BAB V

KESIMPULAN

5.1 Kesimpulan

Persoalan Cyberpunk 2077 Breach Protocol dapat diselesaikan dengan menggunakan Algoritma Brute Force, namun pendekatan ini tidak selalu menghasilkan solusi optimal dan membutuhkan waktu yang lebih lama terutama untuk persoalan dengan ukuran masukan yang besar. Sebagai alternatif, pengembangan program dengan menggunakan algoritma yang lebih efisien dapat menjadi langkah yang baik untuk meningkatkan kinerja dan efektivitas program.

5.2 Lampiran

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

DAFTAR PUSTAKA

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

<https://www.geeksforgeeks.org/python-gui-tkinter/>

https://www.canva.com/design/DAF8lylHEvI/Z_03TOUqBgUKxQdj5QG0yA/edit?utm_content=DAF8lylHEvI&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton