IF2211 Strategi Algoritma

Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force

Laporan Tugas Kecil 1

Disusun untuk memenuhi tugas mata kuliah Strategi Algoritma
pada Semester 2 (dua) Tahun Akademik 2023/2024

Disusun oleh:

Rizqika Mulia Pratama (13522126)

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

BANDUNG
2024

DAFTAR ISI

BAB I
DESKRIPSI MASALAH

Cyberpunk 2077 Breach Protocol adalah permainan hacking aritmatika dengan tujuan mencari cara untuk memasukkan kode-kode dari matriks kode ke dalam urutan yang dibutuhkan. Permainan ini menarik cukup banyak peminat dikarenakan dapat meningkatkan kemampuan hacking serta mengasah otak agar dapat berpikir dengan strategis dan efisien.

Cyberpunk 2077 Breach Protocol biasa dimainkan dengan menggunakan cyberdeck, sebuah perangkat yang memungkinkan V, karakter utama, untuk mengakses jaringan-jaringan musuh. Cyberdeck memiliki RAM yang terbatas, yang menentukan jumlah dan jenis quickhack yang dapat digunakan.

Quickhack adalah program-program yang dapat mempengaruhi musuh atau objek-objek yang terhubung dengan jaringan. Yang perlu diperhatikan hanyalah kode-kode yang tersedia di matriks kode dan urutan yang dibutuhkan. Kode-kode terdiri dari huruf dan angka, misalnya 1C, BD, 55, dan E9. Urutan yang dibutuhkan terdiri dari satu atau lebih baris kode, misalnya 1C BD 55, BD E9 1C, dan 55 1C E9. Setiap baris kode yang berhasil dimasukkan memberikan efek tertentu, seperti mengurangi biaya RAM, menambahkan waktu, atau mendapatkan komponen.

Pada awal permainan, V harus memilih kode dari baris pertama matriks kode, kemudian dari kolom yang sesuai dengan kode yang dipilih, dan seterusnya, sampai semua baris kode yang dibutuhkan terisi atau waktu habis. Permainan berakhir ketika V berhasil memasukkan semua baris kode yang dibutuhkan atau gagal melakukannya.

Gambar 1 Permainan Breach Protocol
(Sumber: https://cyberpunk.fandom.com/wiki/Quickhacking)

BAB II
SPESIFIKASI TUGAS

- Buatlah program sederhana dalam bahasa C/C#/C++/Java/Python/JavaScript/Go/PHP yang mengimplementasikan *algoritma Brute Force* untuk mencari solusi paling optimal permainan Breach Protocol.
- **Input**:
    1. **Matriks permainan** dapat dihasilkan dengan membaca sebuah berkas berekstensi .txt yang memiliki format sebagai berikut:

buffer_size matrix_widthmatrix_height matrix number_of_sequences sequences_1 sequences_1_reward sequences_2 sequences_2_reward … sequences_n sequences_n_reward
Contoh: 7 66 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

    2. Matriks & sekuens permainan (beserta dengan bobot rewardnya) dapat juga dihasilkan secara otomatis oleh program dengan masukan via CLI berupa *jumlah_token_unik*, *token*, *ukuran_buffer*, *ukuran_matriks*, *jumlah_sekuens*, dan

ukuran_maksimal_sekuens (tidak harus terurut). Output untuk fitur ini harus menampilkan Matriks dan sekuens permainan yang digunakan.

Contoh:
5
BD 1C 7A 55 E9
7
6 6
3
4

- Output:
    1. Bobot reward atau reward maksimal yang dapat digunakan (diisi dengan nol bila tidak ada sekuens yang berhasil didapatkan)
    2. Isi dari buffer (bila tidak ada sekuens yang memenuhi maka tidak usah).
    3. Koordinat dari setiap token secara terurut (bila tidak ada sekuens yang memenuhi maka tidak usah).
    4. Waktu eksekusi program dalam ms (tidak termasuk waktu membaca masukan dan menyimpan solusi).
    5. *Prompt* untuk menyimpan solusi dalam sebuah berkas berekstensi .txt
    6. Format luaran yang tercetak pada terminal dan berkas berekstensi .txt (untuk berkas berekstensi .txt tidak perlu ada *prompt*) harus sesuai dengan urutan format di bawah ini.

Contoh output:
50
7A BD 7A BD 1C BD 55
1, 1
1, 4
3, 4
3, 5
6, 5
6, 4
5, 4

300 ms

Apakah ingin menyimpan solusi? (y/n)

BAB III
TEORI SINGKAT

Algoritma brute force adalah salah satu cara untuk menemukan solusi dari suatu masalah yang tidak diketahui secara pasti. Cara ini dilakukan dengan mencoba semua kemungkinan solusi yang ada, satu per satu, sampai menemukan solusi yang benar atau yang paling mendekati. Algoritma ini sangat mudah untuk dipahami dan diterapkan, karena tidak memerlukan pengetahuan atau asumsi apapun tentang masalah yang dihadapi. Namun, algoritma ini juga memiliki kelemahan yang sangat besar, yaitu membutuhkan waktu dan sumber daya yang sangat banyak, terutama jika masalah yang ditangani memiliki ruang solusi yang sangat besar atau kompleks.

Salah satu bidang yang sering menggunakan algoritma brute force adalah bidang komputasi, khususnya dalam hal hacking. Hacking adalah kegiatan untuk memasuki atau mengambil alih sistem komputer yang dilindungi oleh keamanan. Salah satu cara untuk melakukan hacking adalah dengan memecahkan kode yang mengamankan sistem tersebut, seperti password atau kunci enkripsi. Algoritma brute force dapat digunakan untuk mencari kode yang benar dengan cara menelusuri semua kombinasi karakter yang mungkin, mulai dari yang paling sederhana hingga yang paling rumit, sampai menemukan kombinasi yang cocok dengan kode yang dicari.

Algoritma brute force memiliki kelebihan yaitu dapat menjamin menemukan solusi yang benar atau optimal, asalkan waktu dan sumber daya yang tersedia cukup. Algoritma ini juga tidak bergantung pada informasi atau asumsi apapun tentang masalah yang ditangani, sehingga dapat digunakan untuk masalah yang tidak diketahui secara pasti. Namun, algoritma ini juga memiliki kekurangan yaitu sangat tidak efisien dan boros, karena harus mencoba semua kemungkinan solusi tanpa mempertimbangkan kriteria atau batasan apapun. Efektivitas algoritma ini sangat dipengaruhi oleh ukuran dan kompleksitas ruang solusi, serta panjang dan tingkat kesulitan kode yang dicari.

BAB IV
CARA KERJA ALGORITMA

1. **Pembangkitan Semua *Path***: Fungsi generate_all_paths dirancang untuk membentuk semua kemungkinan *path* dari sebuah matriks kode. Fungsi ini memulai dengan posisi awal baris pertama matriks dan ukuran buffer yang ditentukan. Dalam setiap panggilan rekursif, fungsi ini melakukan iterasi melalui setiap elemen dalam baris atau kolom yang sama dengan posisi awal, dan menambahkan elemen tersebut ke *path* jika elemen tersebut cocok dengan salah satu elemen dalam sekuens yang dibutuhkan. Kemudian, fungsi ini memanggil dirinya sendiri dengan posisi awal yang baru (elemen yang baru ditambahkan), ukuran buffer yang dikurangi satu, dan arah yang berubah (jika sebelumnya horizontal, maka berubah menjadi vertikal, dan sebaliknya). Proses ini berlanjut sampai ukuran buffer mencapai satu, yang berarti sebuah *path* telah lengkap dibentuk. Pada titik ini, fungsi mengembalikan *path* tersebut. Karena fungsi ini memanggil dirinya sendiri untuk setiap elemen dalam baris atau kolom yang sama dengan posisi awal, semua kemungkinan *path* akan dibangkitkan.

2. **Pencarian Solusi Terbaik**: Fungsi breach_protocol dirancang untuk mencari *path* yang memberikan nilai *reward* maksimal dari sebuah daftar sekuens yang dibutuhkan. Fungsi ini pertama-tama memanggil fungsi generate_all_paths untuk mendapatkan semua *path* yang mungkin dari matriks. Kemudian, fungsi ini melakukan iterasi untuk setiap *path*. Untuk setiap *path*, fungsi ini mengisi buffer dengan elemen-elemen di *path* tersebut, dan menghitung total *reward* yang didapat dari buffer tersebut. Total *reward* dihitung dengan membandingkan isi buffer dengan setiap urutan dalam daftar urutan yang dibutuhkan, dan menambahkan *reward* dari setiap urutan yang cocok. Jika total *reward* lebih besar dari nilai *reward* maksimal sejauh ini, maka nilai *reward* maksimal, isi buffer terbaik, dan koordinat terbaik diperbarui dengan nilai-nilai dari *path* tersebut. Akhirnya, fungsi ini mengembalikan nilai *reward* maksimal, isi buffer terbaik, koordinat terbaik, dan waktu eksekusi program.

3. **Penyimpanan Solusi**: Fungsi save_solution digunakan untuk menyimpan solusi terbaik ke dalam sebuah file teks. Fungsi ini pertama-tama memeriksa apakah folder bernama “test” sudah ada, jika belum, fungsi ini akan membuatnya. Kemudian, fungsi ini membuat sebuah file teks dengan nama yang diberikan oleh pengguna. Setelah itu, fungsi ini menulis solusi ke dalam file teks dengan format yang ditentukan, termasuk nilai *reward* maksimal, isi buffer terbaik, koordinat terbaik, dan waktu eksekusi program. Jika tidak ada *path* yang optimal atau tidak ada urutan yang bisa dibentuk (nilai *reward* maksimal adalah 0), fungsi ini juga menulis pesan khusus ke file teks.

BAB V
SOURCE PROGRAM

functions.py

```python
import os
import time
import random

def generate_all_paths(matrix, buffer_size, start=(0,0), is_horizontal=True):
    i, j = start
    if buffer_size == 1:
        if is_horizontal:
            return [[(i, new_j)] for new_j in range(len(matrix[0])) if new_j != j]
        else:
            return [[(new_i, j)] for new_i in range(len(matrix)) if new_i != i]
    else:
        paths = []
        if is_horizontal:
            for new_j in range(len(matrix[0])):
                if new_j != j:
                    smaller_paths = generate_all_paths(matrix, buffer_size - 1, start=(i, new_j), is_horizontal=False)
                    for path in smaller_paths:
                        paths.append([(i, new_j)] + path)
        else:
            for new_i in range(len(matrix)):
                if new_i != i:
                    smaller_paths = generate_all_paths(matrix, buffer_size - 1, start=(new_i, j), is_horizontal=True)
                    for path in smaller_paths:
                        paths.append([(new_i, j)] + path)
        return paths
```

```python
def breach_protocol(matrix, sequences, buffer_size):
    start_time = time.time()
    max_reward = 0
    best_buffer = []
    best_coordinates = []

    paths = generate_all_paths(matrix, buffer_size)

    for path in paths:
        buffer = [matrix[i][j] for i, j in path]
        total_reward = sum(reward for seq, reward in sequences if ''.join(seq) in ''.join(buffer))
        if total_reward > max_reward:
            max_reward = total_reward
            best_buffer = buffer
            best_coordinates = [(i+1, j+1) for i, j in path]

    execution_time = (time.time() - start_time) * 1000

    return max_reward, best_buffer, best_coordinates, execution_time


def generate_matrix_and_sequence(num_unique_tokens, tokens, buffer_size, matrix_width, matrix_height, num_sequences, max_sequence_length):
    matrix = [[random.choice(tokens) for i in range(matrix_width)] for j in range(matrix_height)]
    sequences_reward = [(random.choices(tokens, k=random.randint(1, max_sequence_length)), random.randint(1, 30)) for z in range(num_sequences)]
    return matrix, sequences_reward
```

```python
def save_solution(max_reward, best_buffer, best_coordinates, execution_time, matrix, sequences):
    folder_path = "../test"
    if not os.path.exists(folder_path):
        os.makedirs(folder_path)

    filename = input("Input file name to save the solution: ")
    file_path = os.path.join(folder_path, f"{filename}.txt")

    with open(file_path, 'w') as file:
        file.write("Buffer size: 7\n")
        file.write("Matrix:\n")
        for row in matrix:
            file.write(' '.join(row) + '\n')
        file.write("Number of Sequences: 3\n")
        file.write("Sequences and Rewards:\n")
        for sequence, reward in sequences:
            file.write(f"Sequence: {' '.join(sequence)}, Reward: {reward}\n")
        file.write("\n")
        if max_reward == 0:
            file.write("Maximum reward weight: 0\n")
            file.write("Tidak ada jalur yang optimal atau tidak ada urutan yang bisa dibentuk.\n")
        else:
            file.write(f"Maximum reward weight: {max_reward}\n")
            file.write(f"Contents of the buffer: {' '.join(best_buffer)}\n")
            file.write("Coordinates of each token in order:\n")
            for coordinate in best_coordinates:
                file.write(f"{coordinate}\n")
        file.write(f"Program execution time in ms: {execution_time} ms\n")

    print(f"The solution has been saved into {filename}.txt")
```

main.py

```
import functions
import os

def file_mode():
    nama_file = input("Masukkan nama file (tanpa format): ")

    try:
        with open("input_file/" + nama_file + ".txt", 'r') as file:
            lines = file.readlines()

            try:
                buffer_size = int(lines[0])
            except ValueError:
                raise ValueError("Buffer value should be integer.")

            try:
                matrix_width, matrix_height = map(int, lines[1].split())
            except (ValueError, IndexError):
                raise ValueError("The format for filling in the length and width of the matrix does not match.")

            if matrix_height != len(lines[2:2+matrix_height]) or any(len(line.strip().split()) != matrix_width for line in lines[2:2+matrix_height]):
                raise ValueError("The matrix dimensions do not match the expected width and height.")

            matrix = []
            for line in lines[2:2+matrix_height]:
                row = []
                i = 0
                while i < len(line.strip()):
                    if line[i] != ' ':
                        row.append(line[i:i+2])
                        i += 2
                    else:
                        i += 1
                matrix.append(row)
```

```
            try:
                num_sequences = int(lines[2+matrix_height])
            except ValueError:
                raise ValueError("Number of sequences should be integer.")

            if num_sequences * 2 != len(lines) - (3 + matrix_height):
                raise ValueError("The number of sequences and rewards does not match the expected number of sequences.")

            sequences_rewards = []

            for i in range(3+matrix_height, len(lines), 2):
                sequence = lines[i].strip().split()
                try:
                    reward = int(lines[i+1])
                except ValueError:
                    raise ValueError("Reward should be.")
                if reward <= 0:
                    raise ValueError("Reward harus merupakan bilangan bulat positif.")
                sequences_rewards.append((sequence, reward))

        print("Buffer size:", buffer_size)
        print("Matrix width:", matrix_width)
        print("Matrix height:", matrix_height)
        print("Matrix:")
        for row in matrix:
            print(' '.join(row))
        print("Number of Sequences:", num_sequences)
        print("Sequences and Rewards:")
        for sequence, reward in sequences_rewards:
            print(f"Sequence: {' '.join(sequence)}, Reward: {reward}")


        max_reward, best_buffer, best_coordinates, execution_time = functions.breach_protocol(matrix, sequences_rewards, buffer_size)
        print()
```

```
        if max_reward == 0:
            print("Maximum reward weight: ", max_reward)
            print("Tidak ada jalur yang optimal atau tidak ada urutan yang bisa dibentuk.")
        else:
            print("Maximum reward weight: ", max_reward)
            print("Contents of the buffer: ", ' '.join(best_buffer))
            print("Coordinates of each token in order: ")
            for i in range(len(best_coordinates)):
                print(best_coordinates[i])
            print("Program execution time in ms: ", execution_time)
            print()

        save = input("Do you want to save the solution? (y/n): ")

        if save == 'y' or save == 'Y':
            functions.save_solution(max_reward, best_buffer, best_coordinates, execution_time, matrix, sequences_rewards)
    except FileNotFoundError:
        print("File tidak ditemukan.")
    except ValueError as e:
        print("Error:", e)

def manual_mode():
    try:
        while True:
            try:
                num_unique_tokens = int(input("Enter the number of unique tokens: "))
                break
            except ValueError:
                print("Invalid input. Please enter an integer.")
```

```
        while True:
            try:
                tokens = input("Enter the tokens (separated by space): ").split()
                if len(tokens) != num_unique_tokens:
                    print("Number of tokens does not match the specified number of unique tokens. Please try again.")
                    continue
                break
            except ValueError:
                print("Invalid input. Please enter tokens separated by spaces.")

        while True:
            try:
                buffer_size = int(input("Enter the buffer size: "))
                if buffer_size <= 0:
                    print("Buffer size must be a positive integer. Please try again.")
                    continue
                break
            except ValueError:
                print("Invalid input. Please enter an integer for buffer size.")

        while True:
            try:
                matrix_width, matrix_height = map(int, input("Enter the matrix size (width and height, separated by space): ").split())
                if matrix_width <= 0 or matrix_height <= 0:
                    print("Matrix dimensions must be positive integers. Please try again.")
                    continue
                break
            except ValueError:
                print("Invalid input. Please enter two integers separated by space.")
```

```
    while True:
        try:
            num_sequences = int(input("Enter the number of sequences: "))
            if num_sequences <= 0:
                print("Number of sequences must be a positive integer. Please try again.")
                continue
            break
        except ValueError:
            print("Invalid input. Please enter an integer for the number of sequences.")

    while True:
        try:
            max_sequence_length = int(input("Enter the maximum sequence length: "))
            if max_sequence_length <= 0:
                print("Maximum sequence length must be a positive integer. Please try again.")
                continue
            break
        except ValueError:
            print("Invalid input. Please enter an integer for maximum sequence length.")


    matrix, sequences_rewards = functions.generate_matrix_and_sequence(num_unique_tokens, tokens, buffer_size, matrix_width, matrix_height, num_sequences, max_sequenc

    print("Matrix: ")
    for row in matrix:
        print(' '.join(row))
    print("Sequences and Rewards:")
    for sequence, reward in sequences_rewards:
        print(f"Sequence: {' '.join(sequence)}, Reward: {reward}")

    max_reward, best_buffer, best_coordinates, execution_time = functions.breach_protocol(matrix, sequences_rewards, buffer_size)
    print()
```

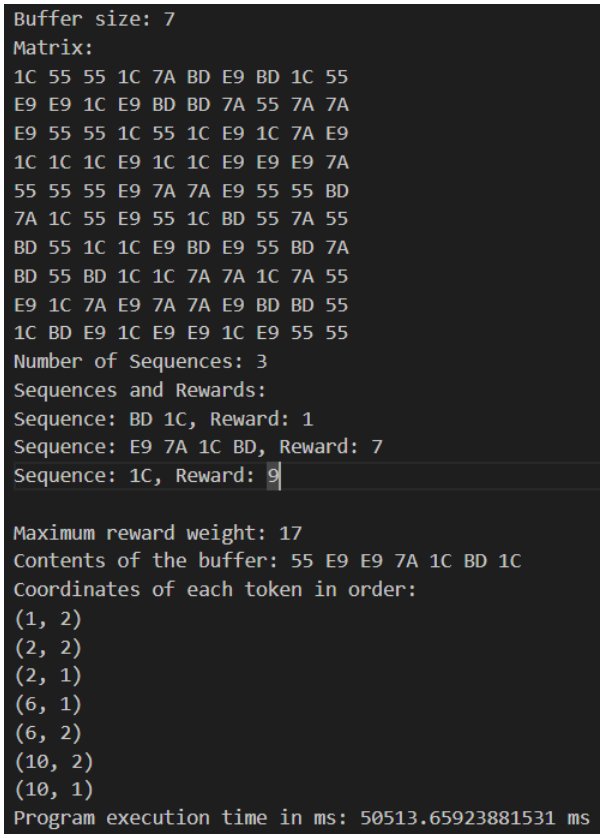
```
        if max_reward == 0:
            print("Maximum reward weight: ", max_reward)
            print("Tidak ada jalur yang optimal atau tidak ada urutan yang bisa dibentuk.")
        else:
            print("Maximum reward weight: ", max_reward)
            print("Contents of the buffer: ", ' '.join(best_buffer))
            print("Coordinates of each token in order: ")
            for i in range(len(best_coordinates)):
                print(best_coordinates[i])
            print("Program execution time in ms: ", execution_time)
            print()

        save = input("Do you want to save the solution? (y/n): ")

        if save == 'y' or save == 'Y':
            functions.save_solution(max_reward, best_buffer, best_coordinates, execution_time, matrix, sequences_rewards)

    except ValueError as e:
        print("Error:", e)

while True:
    os.system("cls")
    print("1. File Input")
    print("2. Manual Input")
    choice = int(input("Choose option: "))
    if choice == 1:
        file_mode()
        break
    elif choice == 2:
        manual_mode()
        break
    else:
        print("There is no such option.")
```

BAB VI
PENGETESAN PROGRAM

Input	Output
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	Buffer size: 7 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 Number of Sequences: 3 Sequences and Rewards: Sequence: BD E9 1C, Reward: 15 Sequence: BD 7A BD, Reward: 20 Sequence: BD 1C BD 55, Reward: 30 Maximum reward weight: 50 Contents of the buffer: 55 BD 7A BD 1C BD 55 Coordinates of each token in order: (1, 6) (4, 6) (4, 3) (5, 3) (5, 6) (3, 6) (3, 1) Program execution time in ms: 458.5874080657959 ms

7 8 8 55 7A 7A 7A 7A BD 7A BD 55 55 7A BD 55 BD BD 55 7A 7A BD 55 BD BD 7A BD BD 55 55 7A BD 55 BD 55 7A 55 7A BD BD 55 BD BD 7A 7A 7A 7A BD 55 7A BD 7A 3 7A 2 7A E9 28 BD E9 7A E9 28	Buffer size: 7 Matrix: BD E9 55 55 E9 E9 BD 55 E9 7A 1C 7A 1C 55 E9 1C E9 7A E9 1C E9 E9 1C 55 E9 1C 1C BD 1C 55 55 7A BD E9 BD BD 7A 7A 7A 1C 55 E9 55 55 BD E9 BD E9 55 55 BD BD 1C 7A 1C 55 1C 55 BD 7A E9 E9 E9 1C Number of Sequences: 3 Sequences and Rewards: Sequence: 7A, Reward: 2 Sequence: 7A E9, Reward: 28 Sequence: BD E9 7A E9, Reward: 28 Maximum reward weight: 58 Contents of the buffer: 55 BD E9 7A E9 Coordinates of each token in order: (1, 3) (5, 3) (5, 2) (2, 2) (2, 1) Program execution time in ms: 48.49529266357422 ms
8 7 7 DD LL CC DD BB BB BB CC DD LL LL CC CC LL BB DD CC LL DD BB LL CC LL DD CC BB DD DD BB LL CC CC LL DD DD DD BB DD LL LL BB BB DD LL LL CC CC DD CC 5 BB BB DD CC 28 LL CC BB BB 5 CC BB CC BB CC 17 LL BB 6 BB BB CC 20	Buffer size: 8 Matrix: DD LL CC DD BB BB BB CC DD LL LL CC CC LL BB DD CC LL DD BB LL CC LL DD CC BB DD DD BB LL CC CC LL DD DD DD BB DD LL LL BB BB DD LL LL CC CC DD CC Number of Sequences: 5 Sequences and Rewards: Sequence: BB BB DD CC, Reward: 28 Sequence: LL CC BB BB, Reward: 5 Sequence: CC BB CC BB CC, Reward: 17 Sequence: LL BB, Reward: 6 Sequence: BB BB CC, Reward: 20 Maximum reward weight: 54 Contents of the buffer: LL BB BB DD CC BB BB CC Coordinates of each token in order: (1, 2) (6, 2) (6, 6) (4, 6) (4, 1) (3, 1) (3, 6) (2, 6) Program execution time in ms: 22219.494819641113 ms

3 2 2 7A 55 BD E9 3 55 BD 55 25 7A E9 7 7A 55 7A 18	Buffer size: 3 Matrix: 7A 55 BD E9 Number of Sequences: 3 Sequences and Rewards: Sequence: 55 BD 55, Reward: 25 Sequence: 7A E9, Reward: 7 Sequence: 7A 55 7A, Reward: 18 Maximum reward weight: 0 There is no optimal path or no sequence that can be formed. Program execution time in ms: 0.0 ms
4 5 5 7A 1C 1C 1C E9 7A BD E9 BD BD E9 7A E9 1C 55 E9 BD 1C 1C 7A E9 E9 55 BD 7A 5 7A BD BD 12 E9 12 BD 7A 26 55 5 E9 BD 1C 1C 17	Buffer size: 4 Matrix: 7A 1C 1C 1C E9 7A BD E9 BD BD E9 7A E9 1C 55 E9 BD 1C 1C 7A E9 E9 55 BD 7A Number of Sequences: 5 Sequences and Rewards: Sequence: 7A BD BD, Reward: 12 Sequence: E9, Reward: 12 Sequence: BD 7A, Reward: 26 Sequence: 55, Reward: 5 Sequence: E9 BD 1C 1C, Reward: 17 Maximum reward weight: 38 Contents of the buffer: 1C BD 7A E9 Coordinates of each token in order: (1, 2) (2, 2) (2, 1) (3, 1) Program execution time in ms: 0.0 ms

7 10 10 1C 55 55 1C 7A BD E9 BD 1C 55 E9 E9 1C E9 BD BD 7A 55 7A 7A E9 55 55 1C 55 1C E9 1C 7A E9 1C 1C 1C E9 1C 1C E9 E9 E9 7A 55 55 55 E9 7A 7A E9 55 55 BD 7A 1C 55 E9 55 1C BD 55 7A 55 BD 55 1C 1C E9 BD E9 55 BD 7A BD 55 BD 1C 1C 7A 7A 1C 7A 55 E9 1C 7A E9 7A 7A E9 BD BD 55 1C BD E9 1C E9 E9 1C E9 55 55 3 BD 1C 1 E9 7A 1C BD 7 1C 9	

BAB VII
KESIMPULAN DAN SARAN

a. Kesimpulan

Algoritma *brute force* dapat digunakan untuk mencari semua *path* yang bisa dihasilkan dari sebuah matriks di permainan Cyberpunk 2077 Breach Protocol, serta mencari *path* yang membentuk hasil pola paling optimal dan memiliki reward paling optimal. Kecepatan program bergantung pada ukuran data yang diinput, waktu cukup lama mulai terasa ketika matriks yang diinput melebihi 10 x 10, hal ini dikarenakan algoritma *brute force* memiliki kompleksitas waktu yang relatif buruk.

b. Saran

Program dapat dijadikan acuan untuk pengembangan permainan berbasis pola lainnya.

BAB VIII
DAFTAR PUSTAKA

1. Sumber Pustaka

[1] R. Munir, “Bahan Kuliah IF2211 Strategi Algoritma Algoritma Brute Force.” Accessed: Feb. 12, 2024. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf

[2]“Breach Protocol (quickhack),” *Cyberpunk Wiki*. https://cyberpunk.fandom.com/wiki/Breach_Protocol_(quickhack) (accessed Feb. 12, 2024).

[3]“How to solve Breach Protocol hacking puzzles in Cyberpunk 2077 2.0,” *Dexerto*, Nov. 30, 2023. https://www.dexerto.com/cyberpunk-2077/how-to-solve-breach-protocol-hacking-puzzles-in-cyberpunk-2077-2303399/ (accessed Feb. 12, 2024).

2. Lampiran

1. Github: https://github.com/rizqikapratamaa/Tucil1_13522126
2.

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		✔