LAPORAN TUGAS KECIL 01

IF2211 STRATEGI ALGORITMA

“Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force”

Disusun oleh:

Mohammad Andhika Fadillah K-03 13522128

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

DAFTAR ISI

BAB 1

DESKRIPSI MASALAH

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.

Aturan permainan Breach Protocol antara lain:

1. Pemain bergerak dengan pola horizontal, vertikal, horizontal, vertikal (bergantian) hingga semua sekuens berhasil dicocokkan atau buffer penuh. IF2211 Strategi Algoritma – Tugas Kecil 1 1
2. Pemain memulai dengan memilih satu token pada posisi baris paling atas dari matriks.
3. Sekuens dicocokkan pada token-token yang berada di buffer.
4. Satu token pada buffer dapat digunakan pada lebih dari satu sekuens.
5. Setiap sekuens memiliki bobot hadiah atau reward yang variatif.
6. Sekuens memiliki panjang minimal berupa dua token.

BAB 2
TEORI SINGKAT

2.1 Algoritma Brute Force

Algoritma *brute force* adalah metode yang digunakan untuk memecahkan masalah atau mencari solusi dengan cara mencoba semua kemungkinan secara berurutan hingga menemukan solusi yang benar.

Metode ini sering digunakan ketika tidak ada cara yang lebih efisien untuk menyelesaikan masalah tersebut. Algoritma *brute force* adalah metode yang sederhana dan kuat, karena dia tidak mengandalkan pengetahuan sebelumnya atau optimasi khusus.

2.2 Tujuan Algoritma Brute Force

Tujuan utama dari algoritma *brute force* adalah mencari solusi dari sebuah masalah dengan menguji semua kemungkinan secara sistematis. Beberapa tujuan utama dari penggunaan algoritma ini meliputi:

1. Pencarian solusi: Menggunakan pendekatan yang paling sederhana untuk menemukan solusi dari sebuah masalah.
2. Verifikasi: Memastikan bahwa solusi yang ditemukan adalah benar dengan menguji semua kemungkinan.
3. Pemahaman masalah: Mendapatkan pemahaman yang lebih dalam tentang masalah yang dihadapi.

BAB 3

IMPLEMENTASI PROGRAM

3.1 Folder src

a. main.py

```
from typing import List, Tuple, Set
import time

def tampilan_awal():
    print(" _______   _______ _________________ _   _ _   _ _   _   __")
    print("/  __ \ \ / / ___ \  ___| ___ \ ___ \ | | | \ | || | / / ")
    print("| /  \/\ V /| |_/ / |__ | |_/ / |_/ / | | |  \| || |/ / ")
    print("| |     \ / | ___ \  __||    /|  __/| | | | . ` ||    \ ")
    print("| \__/\ | | | |_/ / |___| |\ \| |   | |_| | |\  || |\  \ ")
    print("\____/ \_/ \____/\____/\_| \_\_|    \___/\_| \_/\_| \_/ ")

    print("Welcome to Cyberpunk 2077 Breach Protocol!\n")
    input("click enter to get the result")
```

```
def txtInput(filename):
    with open(filename, 'r') as file:
        ukuranbuffer = int(file.readline())
        ukuranmatriks = tuple(map(int, file.readline().split()))
        matrix = [list(file.readline().split()) for _ in range(ukuranmatriks[1])]
        jumlahseqs = int(file.readline())
        sekuens = []
        points = []
        for i in range(jumlahseqs):
            sekuens.append(file.readline().strip().split())
            points.append(int(file.readline()))
    return ukuranbuffer, matrix, sekuens, points
```

```
def find_all_patterns(matrix: List[List[str]], step: int) -> List[List[Tuple[str, Tuple[int, int]]]]:
    rows = len(matrix)
    cols = len(matrix[0])
    all_paths = []

    def generate_paths(x: int, y: int, path: List[Tuple[str, Tuple[int, int]]], sign: Set[Tuple[int, int]], steps: int) -> None:

        if steps == 0:
            all_paths.append(path.copy())
            return
        sign.add((x, y))
        path.append((matrix[y][x], (x, y)))

        for xn, yn in [(0, 1), (1, 0)]:
            next_x, next_y = x + xn, y + yn
            if 0 <= next_x < cols and 0 <= next_y < rows and (next_x, next_y) not in sign:
                generate_paths(next_x, next_y, path, sign, steps - 1)

        sign.remove((x, y))
        path.pop()

    for x in range(cols):
        for y in range(rows):
            generate_paths(x, y, [], set(), step)

    return all_paths
```

```python
def count_point(matrix, path, sekuens, points):
    total_reward = 0
    buffer_tokens = []
    for token, _ in path:
        buffer_tokens.append(token)
        for seq, reward in zip(sekuens, points):
            if all(t in buffer_tokens for t in seq):
                total_reward += reward
                buffer_tokens = [t for t in buffer_tokens if t not in seq]
    return total_reward

def pola(matrix, ukuranbuffer, sekuens, points):

    point_maks = float('-inf')
    optimal_path = []

    all_paths = find_all_patterns(matrix, ukuranbuffer)
    for path in all_paths:
        path_reward = count_point(matrix, path, sekuens, points)
        if path_reward > point_maks:
            point_maks = path_reward
            optimal_path = path

    return point_maks, optimal_path
```

```python
def main():
    tampilan_awal()
    buffer_size, matrix, sequences, points = txtInput('tc6.txt')
    start_time = time.time()
    print("\n")
    maximal_point, optimal_path = pola(matrix, buffer_size, sequences, points)
    end_time = time.time()
    waktufinal = end_time - start_time
    print("Point :", maximal_point)
    print("Path :", ' '.join(token for token, _ in optimal_path))
    print("Koordinat Jalur :")
    for _, (x, y) in optimal_path:
        print(f"{x + 1}, {y + 1}")
    print("waktu final : ")
    print(waktufinal * 1000 ,"ms")

if __name__ == "__main__":
    main()
```

b. tc1.py

```
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
```

c. tc2.py

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

d. tc3.py

```
7
5 5
DD BB DD BB AA
FF BB EE DD DD
DD BB CC FF EE
EE DD EE AA EE
DD BB BB BB FF
4
BB CC EE EE
48
FF AA FF CC
40
DD CC
27
CC EE BB BB
30
```

e. tc4.py

```
6
5 5
33 11 77 33 77
77 33 33 77 33
33 33 11 77 55
55 33 33 77 33
77 33 11 55 55
2
33 55 33
35
33 55
12
```

f. tc5.py

```
7
9 9
98 AD KL AD VB AD 98 KL 23
98 23 98 AD 98 AD 98 KL 23
KL KL KL 23 5T 23 5T AD 5T
KL 5T AD 23 23 98 5T VB 23
AD 98 5T 23 VB 5T KL AD 98
98 5T 5T 5T KL KL KL 5T 5T
5T AD KL 5T KL VB 23 KL 5T
AD 98 5T 98 AD KL AD 5T 23
5T 23 5T AD KL 5T 98 AD 5T
3
VB AD 23
42
VB 5T
24
98 23 VB
29
```

g. tc6.py

```
5
4 4
8I 12 8I 8I
OP 8I OP OP
12 12 12 8I
OP 12 OP 8I
2
8I 12
26
12 12
25
```

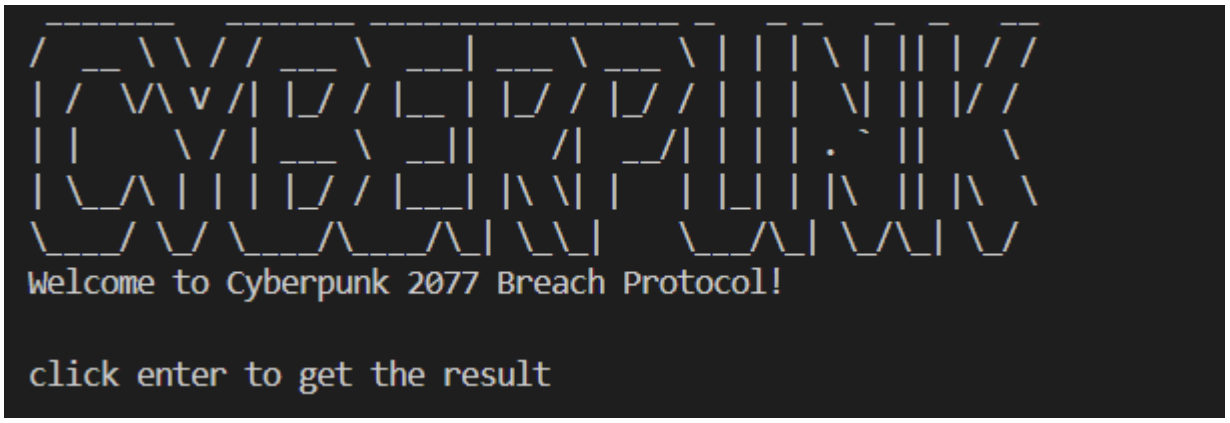
3.2 Folder test

Seharusnya untuk menyimpan hasil yang telah didapatkan

BAB 4

EKSPERIMEN

Inisiasi Program



Gambar tampilan awal program

Untuk mengganti file test case, pergi ke fungsi main lalu ubah didalam fungsi txtInput.

```
def main():
    tampilan_awal()
    buffer_size, matrix, sequences, points = txtInput('tc6.txt')
    start_time = time.time()
```

test case 1

```
Point : 50
Path : 7A 55 55 BD BD 1C 55
Koordinat Jalur :
1, 1
1, 2
1, 3
1, 4
1, 5
1, 6
2, 6
waktu final :
100.59452056884766 ms
```

test case 2

```
Point : 68
Path : 1C 8G BD E9 E9 BD 8G
Koordinat Jalur :
1, 6
2, 6
3, 6
4, 6
5, 6
6, 6
7, 6
waktu final :
147.2487449645996 ms
```

test case 3

```
Point : 48
Path : DD FF BB EE CC EE BB
Koordinat Jalur :
1, 1
1, 2
2, 2
3, 2
3, 3
3, 4
3, 5
waktu final :
87.5401496887207 ms
```

text case 4

```
Point : 70
Path : 33 55 77 33 11 55
Koordinat Jalur :
1, 3
1, 4
1, 5
2, 5
3, 5
4, 5
waktu final :
75.70505142211914 ms
```

test case 5

```
Point : 66
Path : KL AD VB 98 5T 23 VB
Koordinat Jalur :
3, 1
4, 1
5, 1
5, 2
5, 3
5, 4
5, 5
waktu final :
163.69366645812988 ms
```

test case 6

```
Point : 77
Path : 8I 12 8I 12 12
Koordinat Jalur :
1, 1
2, 1
2, 2
2, 3
2, 4
waktu final :
68.89224052429199 ms
```

BAB 5
PENUTUP

5.1 Kesimpulan

Melalui tugas besar ini, saya belajar banyak hal terkait library dan bahasa pemrograman C++ dan python. Saya juga belajar banyak tentang algoritma *brute force.* Algoritma *brute force* dapat menyelesaikan banyak persoalan algoritma, namun tidak efisien hasilnya.

5.2 Saran

- Jangan kerjain h-1 deadline, jangan terlalu banyak meremehkan pekerjaan, dan jangan terlalu bingung untuk menggunakan bahasa pemrograman apa dalam suatu tugas.

5.3 Link Repository

Link repository untuk tugas kecil 1 mata kuliah IF2211 Strategi Algoritma adalah sebagai berikut

Link : https://github.com/Andhikafdh/Tucil1_13522128

5.4 Tabel Checkpoint Program

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		✓

DAFTAR REFERENSI

https://www.cloudeka.id/id/berita/web-sec/cara-kerja-algoritma-brute-force/

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Algoritma-Brute-Force-(2016).pdf

https://www.w3schools.com/