

LAPORAN TUGAS KECIL 01

IF2211 STRATEGI ALGORITMA

“Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force”



Disusun oleh:

Farrel Natha Saskoro

13522145

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

DAFTAR ISI

DAFTAR ISI	2
BAB 1	3
DESKRIPSI MASALAH	3
BAB 2	4
BAB 3	5
BAB 4	12
BAB 5	19

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

strategi algoritma yang dipakai dalam menyelesaikan puzzle dalam minigame breach protocol adalah brute force dengan langkah - langkah sebagai berikut:

- inisiasi array kosong yang nantinya akan diisi oleh sebuah array of path
- mencari semua kemungkinan path (array of token yang memiliki panjang sesuai buffer inputan) yang bisa dicari dari matriks inputan
- simpan all_possible_path tersebut kedalam array kosong tadi
- cocokkan setiap path yang didapat dengan sequence inputan, jika sub-sequence dari path cocok dengan sequence inputan maka point path tersebut akan bertambah sesuai dengan point inputan
- path yang memiliki poin disimpan kedalam array kosong
- lanjut untuk mencocokkan path selanjutnya dengan sequence yang diinput seperti cara diatas
- bandingkan poin dari path yang sudah dicek dan path yang sedang dicek, jika poin path yang sedang dicek lebih besar maka path yang sudah dicek tadi akan dihapus dari array dan diganti dengan path yang baru saja dicek
- lakukan langkah di atas terus-menerus hingga semua kemungkinan path sudah dicek

BAB 3

IMPLEMENTASI PROGRAM

main.py

[illegible]

```

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

print()
print("Sequences: ")
for i in range(number_of_sequences):
    for j in range(len(sequences[i])):
        print(sequences[i][j], end=" ")
    print()
    print(rewards[i], end=" ")
    print()

print()

results = read.compare_paths(all_paths, sequences, rewards)
result.result(results, buffer_size)
end_time = time.time()
execution_time_ms = (end_time - start_time) * 1000
print(f"{round(execution_time_ms)} ms")
saveToFile(results, buffer_size)

elif choice == "2":
    print("Masukkan input anda: ")
    sumToken = int(input(""))
    tokens = input("")
    token_list = tokens.split()
    bufferSize = int(input(""))
    rows, cols = map(int, input("").split())
    sumSequence = int(input(""))
    maxSequence = int(input(""))

    start_time = time.time()
    matrix = acak.randMatrix(token_list, rows, cols)
    print()
    print("Matrix: ")
    for i in range(rows):
        for j in range(cols):
            print(matrix[i][j], end=" ")

```

```

        print()

    print()

    print("Sequences: ")
    sequences = acak.randSequences(token_list, maxSequence, sumSequence)
    rewards = acak.randRewards(sumSequence)
    for i in range(sumSequence):
        for j in range(len(sequences[i])):
            print(sequences[i][j], end=" ")
        print()
        print(rewards[i], end=" ")
        print()

    print()
    all_paths = read.generate_all_possible_path(matrix, bufferSize)
    results = read.compare_paths(all_paths, sequences, rewards)
    result.result(results, bufferSize)
    end_time = time.time()
    execution_time_ms = (end_time - start_time) * 1000
    print(f"{round(execution_time_ms)} ms")
    save.toFile(results, bufferSize)

else:
    print("Terima kasih telah menggunakan program ini!")
    exit()

```

read.py

```

from collections import deque

def read_file(file: str) -> tuple[int, int, int, list[list[str]], int, list[list[str]], list[int]]:
    with open("test/" + file, "r") as f:
        lines = f.readlines()

    buffer_size = int(lines[0].strip())
    matrix_width, matrix_height = map(int, lines[1].strip().split())
    matrix = [list(map(str, line.strip().split())) for line in

```

```

lines[2:2+matrix_height]]
    number_of_sequences = int(lines[2+matrix_height].strip())

    sequences = []
    rewards = []
    for i in range(number_of_sequences):
        sequences.append(list(map(str,
lines[3+matrix_height+i*2].strip().split()))
        rewards.append(int(lines[4+matrix_height+i*2].strip()))

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

def generate_all_possible_path(matrix, step):
    rows, cols = len(matrix), len(matrix[0])
    all_paths = []
    queue = deque(
        [(matrix[0][x], (x, 0)), step - 1, "vertical", {(x, 0)}
        for x in range(cols)
    )

    while queue:
        path, steps, direction, visited = queue.popleft()

        if steps == 0:
            all_paths.append(path)
            continue

        x, y = path[-1][1]

        if direction == "vertical":
            for next_y in range(rows):
                if (x, next_y) not in visited:
                    queue.append(
                        (
                            path + [(matrix[next_y][x], (x, next_y))],
                            steps - 1,
                            "horizontal",
                            visited | {(x, next_y)},
                        )
                    )
            else:

```



```

        for next_x in range(cols):
            if (next_x, y) not in visited:
                queue.append(
                    (
                        path + [(matrix[y][next_x], (next_x, y))],
                        steps - 1,
                        "vertical",
                        visited | {(next_x, y)},
                    )
                )

    return all_paths

def compare_path_with_sequence(
    path: list[tuple[str, tuple[int, int]]], sequence: list[str],
) -> bool:
    for i in range(0, len(path)-len(sequence)+1):
        if all(path[i+j][0] == sequence[j] for j in range(len(sequence))):
            return True
    return False

def compare_paths(
    all_paths: list[list[tuple[str, tuple[int, int]]]],
    sequences: list[list[str]],
    rewards: list[int],
) -> tuple[list[list[tuple[str, tuple[int, int]]]], int]:
    result = []
    point = 0
    temp = 0
    for path in all_paths:
        for i in range(len(sequences)):
            if compare_path_with_sequence(path, sequences[i]):
                point += rewards[i]
        if (len(result) == 0):
            result = path
            temp = point
        else:
            if (point > temp):
                result = path
                temp = point
    point = 0

```

```
return result, temp
```

acak.py

```
import random

def randMatrix(elements: list, n: int, m: int):
    matrix = []
    for i in range(n):
        row = []
        for j in range(m):
            row.append(random.choice(elements))
        matrix.append(row)
    return matrix

def randSequences(elements: list, n: int, k: int):
    sequences = []
    for i in range(k):
        sequence = []
        for j in range(random.randint(2, n)):
            sequence.append(random.choice(elements))
        sequences.append(sequence)
    return sequences

def randRewards(k: int):
    rewards = []
    for i in range(k):
        rewards.append(random.randint(1, 100))
    return rewards
```

result.py

```
def result(list: tuple[list[tuple[str, tuple[int, int]]], int], x: int) ->
None:
    print("result: ")
    print(list[1])
    for i in range(x):
        print(list[0][i][0], end=" ")
```

```

print()
for i in range(x):
    print(str(list[0][i][1][0] + 1) + ", " + str(list[0][i][1][1] + 1))
print()

```

save.py

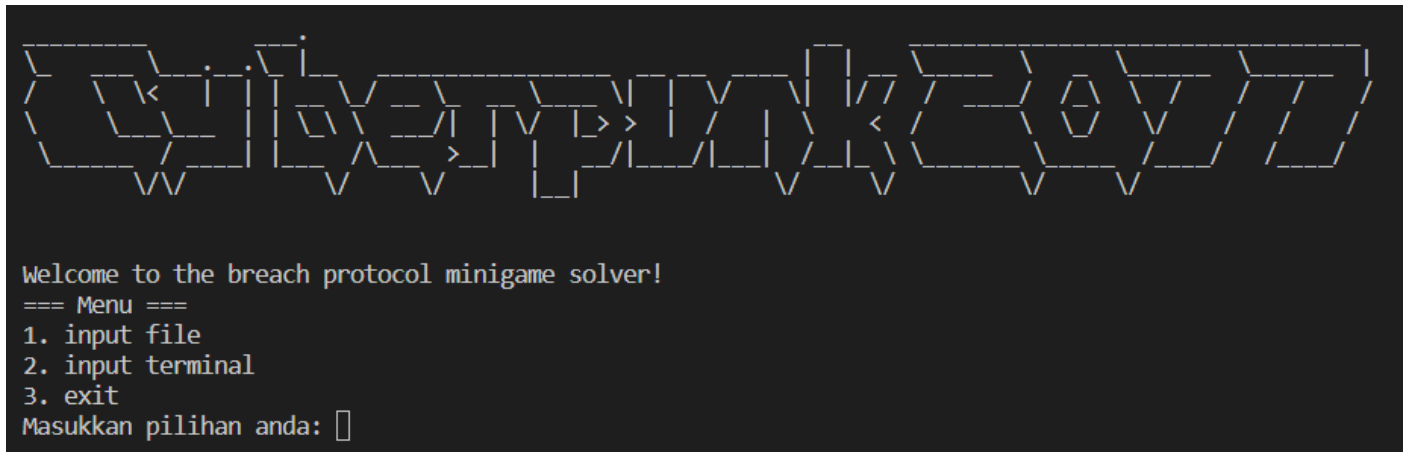
```

def toFile(list, x):
    print("Apakah anda ingin menyimpan hasil permainan? (Y/N)")
    choice = input()
    if choice == "Y" or choice == "y":
        filename = input("Masukkan nama file: ")
        path = "test/" + filename + ".txt"
        with open(path, 'w') as f:
            f.write(str(list[1]) + "\n")
            for i in range(x):
                f.write(list[0][i][0] + " ")
            f.write("\n")
            for i in range(x):
                f.write(str(list[0][i][1][0] + 1) + ", " +
str(list[0][i][1][1] + 1) + "\n")
            print("Hasil permainan berhasil disimpan di " + path)
            print("Terima kasih telah menggunakan program ini!")

```

EKSPERIMEN

Inisiasi Program



Gambar 0 tampilan awal program

test case 1

soal:

1	7
2	6 6
3	7A 55 E9 E9 1C 55
4	55 7A 1C 7A E9 55
5	55 1C 1C 55 E9 BD
6	BD 1C 7A 1C 55 BD
7	BD 55 BD 7A 1C 1C
8	1C 55 55 7A 55 7A
9	3
10	BD E9 1C
11	15
12	BD 7A BD
13	20
14	BD 1C BD 55
15	30

Gambar 1 test case 1

hasil:

```
Masukkan pilihan anda: 1
Masukkan nama file: 1.txt
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

Sequences:
BD E9 1C
15
BD 7A BD
20
BD 1C BD 55
30

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

2596 ms
Apakah anda ingin menyimpan hasil permainan? (Y/N)
Y
Masukkan nama file: ans
Hasil permainan berhasil disimpan di test/ans.txt
Terima kasih telah menggunakan program ini!
```

Gambar 2 hasil test case 1

test case 2

soal:

```
Masukkan pilihan anda: 2
Masukkan input anda:
5
BD 1C 7A 55 E9
7
6 6
3
4
```

Gambar 3 input data matriks dan sekuens

randomize matrix dan sequence:

```
Matrix:
7A 1C 55 7A BD 7A
E9 7A E9 55 E9 1C
E9 55 55 1C 1C BD
E9 BD 55 1C 7A 1C
1C E9 7A 55 7A E9
7A 1C BD 55 7A 1C

Sequences:
E9 BD
59
7A 1C
88
7A BD E9
37
```

Gambar 4 otomatisasi dari data matriks dan sekuens

hasil:

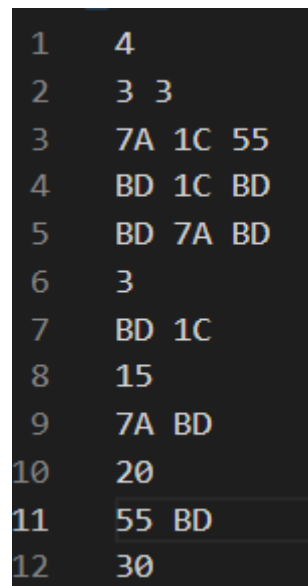
```
result:
184
7A 1C 7A BD E9 E9 BD
6, 1
6, 2
2, 2
2, 4
1, 4
1, 3
6, 3

2470 ms
Apakah anda ingin menyimpan hasil permainan? (Y/N)
Y
Masukkan nama file: ans2
Hasil permainan berhasil disimpan di test/ans2.txt
Terima kasih telah menggunakan program ini!
```

Gambar 5 hasil test case 2

test case 3

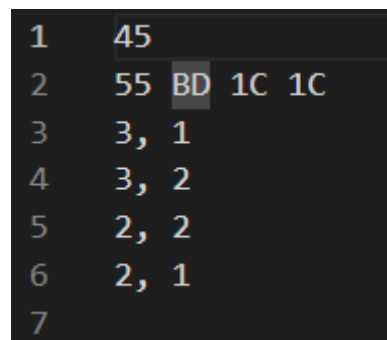
soal:



1	4
2	3 3
3	7A 1C 55
4	BD 1C BD
5	BD 7A BD
6	3
7	BD 1C
8	15
9	7A BD
10	20
11	55 BD
12	30

Gambar 6 soal test case 3

hasil:



1	45
2	55 BD 1C 1C
3	3, 1
4	3, 2
5	2, 2
6	2, 1
7	

Gambar 7 hasil test case 3

test case 4

soal:

```
1 5
2 6 6
3 AY TT 4G UU TT 23
4 TT 4G 23 UU 4G 23
5 OU 23 4G 23 4G 23
6 4G TT 4G UU 23 4G
7 OU 4G OU TT UU 4G
8 TT 4G 23 4G 23 4G
9 3
10 OU 4G 23
11 15
12 23 UU 23
13 20
14 TT OU 4G 23
15 30
```

Gambar 8 soal test case 4

hasil:

```
1 45
2 UU TT OU 4G 23
3 4, 1
4 4, 5
5 1, 5
6 1, 4
7 5, 4
8
```

Gambar 9 hasil test case 4

test case 5

soal:

```
Masukkan input anda:
5
HH FF EE 44 F4
6
4 4
4
3
```

Gambar 10 soal test case 5

hasil:

```
Matrix:
EE HH FF FF
F4 F4 FF EE
44 F4 FF F4
44 FF F4 F4

Sequences:
EE HH F4
80
HH EE HH
33
44 44 44
60
EE F4 EE
16
```

Gambar 11 hasil otomatisasi matriks dan sekuens test case 5

```
result:
96
FF EE F4 EE HH F4
4, 1
4, 2
1, 2
1, 1
2, 1
2, 2
```

Gambar 12 hasil test case 5

test case 6

soal:

```
6
AB BC CD EF GH HI
6
5 5
5
3
```

Gambar 13 soal test case 6

hasil:

```
Matrix:
BC HI CD GH HI
BC GH BC EF GH
CD BC BC EF AB
EF EF HI EF BC
AB BC BC BC BC

Sequences:
CD GH CD
9
CD EF GH
39
AB AB
61
EF CD AB
75
EF GH
10
```

Gambar 14 hasil otomatisasi matriks dan sekuens test case 6

```
result:
75
BC BC EF EF CD AB
1, 1
1, 2
4, 2
4, 3
1, 3
1, 5
```

Gambar 15 hasil test case 6

BAB 5

PENUTUP

5.1 Kesimpulan

Algoritma brute force dapat menyelesaikan berbagai permasalahan dengan baik, walaupun keefesiensiannya tergantung kepada ukuran matriks dan sekuens yang diberikan, sehingga tidak begitu optimal untuk ukuran matriks dan sekuens yang tinggi.

5.2 Link Repository

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

Link : https://github.com/fnathas/Tucil1_13522145

5.3 Tabel Checkpoint Program

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan	✓	
2. Program berhasil <i>dijalankan</i>	✓	
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		✓