

# **LAPORAN TUGAS KECIL 1**

## **STRATEGI ALGORITMA**

Dosen: Nur Ulfa Maulidevi.

**IF2211-22 Strategi Algoritma**

**Nama : Ahmad Farid Mudrika**

**NIM : 13522008**

**SEKOLAH TEKNIK ELEKTRO**

**DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**2023**

## BAB I DESKRIPSI MASALAH

### ABSTRAKSI



Gambar 1 Permainan Breach Protocol

(Sumber: <https://cyberpunk.fandom.com/wiki/Quickhacking>)

**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.
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.

**Ilustrasi kasus :**

Diberikan matriks sebagai berikut dan ukuran buffernya adalah tujuh

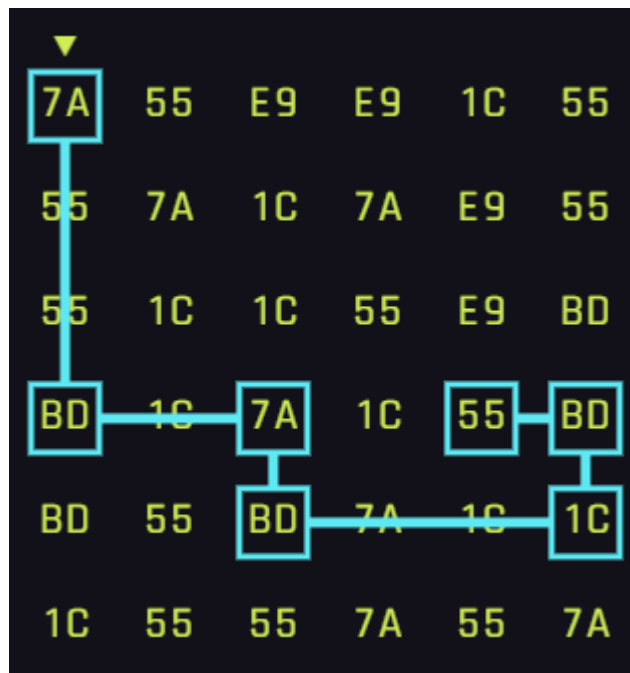
|    |    |    |    |    |    |
|----|----|----|----|----|----|
| 7A | 55 | E9 | E9 | 1C | 55 |
| 55 | 7A | 1C | 7A | E9 | 55 |
| 55 | 1C | 1C | 55 | E9 | BD |
| BD | 1C | 7A | 1C | 55 | BD |
| BD | 55 | BD | 7A | 1C | 1C |
| 1C | 55 | 55 | 7A | 55 | 7A |

Dengan sekuens sebagai berikut:

1. BD E9 1C dengan hadiah berbobot 15.
2. BD 7A BD dengan hadiah berbobot 20.
3. BD 1C BD 55 dengan hadiah berbobot 30.

Maka solusi yang optimal untuk matriks dan sekuens yang diberikan adalah sebagai berikut:

- Total bobot hadiah : 50 poin
- Total langkah : 6 langkah



Gambar 2 Contoh Solusi

(Sumber: <https://cyberpunk-hacker.com/>)

Tugas anda adalah menemukan solusi dari **permainan Breach Protocol** yang paling optimal untuk setiap kombinasi matriks, sekuens, dan ukuran buffer dengan menggunakan ***algoritma brute force***.

## **BAB II**

### **TEORI SINGKAT**

Brute force adalah sebuah pendekatan langsung(straight forward) untuk memecahkan suatu masalah, biasanya didasarkan pada pernyataan masalah (problem statement) dan definisi konsep yang dilibatkan. Algoritma brute force memecahkan masalah dengan sangat sederhana, langsung dan dengan cara yang jelas (obvious way). Di dalam pencocokan string, terdapat istilah teks dan pattern. Teks merupakan kata yang dicari dan dicocokkan dengan pattern.

Dalam menyelesaikan masalah yang diberi, saya menggunakan algoritma brute force dengan cara sebagai berikut:

1. Cari semua kemungkinan sekuens token sesuai ukuran buffer.
2. Tentukan poin yang didapat dari semua kemungkinan tersebut.
3. Ambil sekuens dengan poin tertinggi.

### BAB III

#### SOURCE CODE PROGRAM

```
import numpy as np
import random
import time
def brute_force(matrix, currow, curcol, remaining_moves,
route, allroute, visited, vertical=False):
    if remaining_moves == 0:
        if tuple(route) not in visited:
            allroute.append(route.copy())
            visited.add(tuple(route))
            return

        if (matrix[currow][curcol], (currow, curcol)) not in
route:
            route.append((matrix[currow][curcol], (currow,
curcol)))
        else:
            return

    nexdir = not vertical

    if nexdir:
        for i in range(len(matrix)):
            if i != currow:
                brute_force(matrix, i, curcol, remaining_moves
- 1, route, allroute, visited, nexdir)
            else:
                for j in range(len(matrix[0])):
                    if j != curcol:
                        brute_force(matrix, currow, j, remaining_moves
- 1, route, allroute, visited, nexdir)

    route.pop()

def calculatepoint(route, sequencedict):
    result=0
    for key, value in sequencedict.items():
        sub_len = len(key)
        for i in range(len(route) - sub_len + 1):
```

```

        if route[i:i + sub_len] == key:
            result+=value
    return result
def readtxt(filename):
    try:
        with open(filename, 'r') as file:
            lines=file.readlines()
            buffer_size=int(lines[0])
            matrixsize = list(map(int, lines[1].split()))
            matrix=[list(map(str, line.strip().split()))for
line in lines[2:2+matrixsize[0]]]
            numberofsequence=int(lines[2+matrixsize[0]])
            listsequence=lines[3+matrixsize[0]:]
            sequencedict =
{tuple(listsequence[i].strip().split()): int(listsequence[i +
1]) for i in range(0, 2*numberofsequence, 2)}
        except FileNotFoundError:
            print(f"File {filename} not found.")
            return buffer_size, np.array(matrix), sequencedict

def random_matrix(tokens, m, n):
    matrix = [[random.choice(tokens) for _ in range(n)] for _
in range(m)]
    return matrix

def random_sequencedict(tokens, max_token, amount):
    sequencedict = {}
    for _ in range(amount):
        num_token = random.randint(1, max_token)
        rand_token = random.choices(tokens, k=num_token)
        value = random.randint(1, 100)
        sequencedict[tuple(rand_token)] = value
    return sequencedict

def printmatrix(matrix):
    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            print(matrix[i][j], end=' ')
        print()

def solution(max_key, start_time):
    print(f'{max_key[1]}')
    for i in range(len(max_key[0])):

```

```

        print(max_key[0][i][0], end=' ')
    print()
    for i in range(len(max_key[0])):
        print(f'{max_key[0][i][1][0]}, {max_key[0][i][1][1]}')
    print();print()
    end_time=time.time()
    print(f'{(end_time-start_time)*1000} ms')
    print();print()
    txt=input("Apakah ingin menyimpan solusi? (y/n) ")
    if txt=='y' or txt=='Y':
        fname=input("Masukkan nama file.txt : ")
        name=f'../test/{fname}'
        with open(name, 'w') as file:
            print(max_key[1], file=file)
            content=''
            for i in range(len(max_key[0])):
                content+=max_key[0][i][0]+' '
            print(content[:-1], file=file)
            for i in range(len(max_key[0])):
                print(f'{max_key[0][i][1][0]},
{max_key[0][i][1][1]}', file=file)
            print(file=file);print(file=file)
            print(f'{(end_time-start_time)*1000} ms',
file=file)
if __name__ == "__main__":
    import sys

    if len(sys.argv) >2:
        print("Usage: python main.py <txt_file>")
    elif len(sys.argv)==2:
        buffer_size, matrix, sequencedict=readtxt(sys.argv[1])
        allroute=[]
        visited=set()
        start_time = time.time()
        for i in range(len(matrix)):
            brute_force(matrix, 0, i, buffer_size,
[],allroute, visited)

            hasil = {(tuple(route),
calculatepoint(tuple(element[0] for element in route),
sequencedict)) for route in allroute}

```



```

        max_key = max(hasil, key=lambda k: k[1])
        solution(max_key, start_time)

    else:
        tokencount=int(input("Masukkan jumlah token unik : "))
        token=input("Masukkan token yang dipisahkan spasi : ").strip().split()
        buffer_size=int(input("Masukkan ukuran buffer : "))
        matrixsize=input("Masukkan ukuran matriks : ").strip().split()
        sequencecount=int((input("Masukkan jumlah sekuens : ")))
        sequencesize=int(input("Masukkan ukuran maksimal sekuens : "))
        start_time = time.time()
        matrix=random_matrix(token, int(matrixsize[0]), int(matrixsize[1]))
        sequencedict=random_sequencedict(token, sequencesize, sequencecount)
        print("Matrix acak : ")
        printmatrix(matrix)
        print("Sekuens acak : ")
        for key, value in sequencedict.items():
            print(f"{key} : {value}")
        allroute=[]
        visited=set()
        for i in range(len(matrix)):
            brute_force(matrix, 0, i, buffer_size, [],allroute, visited)
            hasil = {(tuple(route), calculatepoint(tuple(element[0] for element in route), sequencedict)) for route in allroute}
        max_key = max(hasil, key=lambda k: k[1])
        solution(max_key, start_time)

```

## BAB IV

### TANGKAPAN LAYAR

```
PS C:\Users\User\OneDrive - Institut Teknologi Bandung\Documents\Coding\Tucil1_13522008\src> python main.py file.txt
50
7a bd 7a bd 1c bd 55
0, 0
3, 0
3, 2
4, 2
4, 5
2, 5
2, 0

950.8345127105713 ms

Apakah ingin menyimpan solusi? (y/n) y
Masukkan nama file.txt : result1.txt
```

Gambar 4.1 Tangkapan Layar penjalanan program dengan file eksternal

```
src > file.txt
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 4.2 Tangkapan Layar file eksternal yang digunakan pada 4.1

```
test > result1.txt
1 50
2 7a bd 7a bd 1c bd 55
3 0, 0
4 3, 0
5 3, 2
6 4, 2
7 4, 5
8 2, 5
9 2, 0
10
11
12 950.8345127105713 ms
13
```

Gambar 4.3 Tangkapan Layar hasil yang disimpan pada file eksternal di gambar 4.1

```
PS C:\Users\User\OneDrive - Institut Teknologi Bandung\Documents\Coding\Tucil1_13522008\src> python main.py
Masukkan jumlah token unik : 5
Masukkan token yang dipisahkan spasi : BD 1C 7A 55 E9
Masukkan ukuran buffer : 7
Masukkan ukuran matriks : 6 6
Masukkan jumlah sekuens : 4
Masukkan ukuran maksimal sekuens : 6
Matrix acak :
7A BD 55 E9 7A BD
55 55 55 1C 55 BD
BD 1C 55 BD 1C BD
1C 55 1C 55 55 E9
1C 55 1C BD 7A 55
7A 55 1C E9 E9 7A
Sekuens acak :
('55', '55', 'E9', '55') : 36
('55', '7A') : 82
('7A',) : 48
('E9', 'E9', '1C', '55') : 18
356
7A 55 55 7A 7A 55 7A
0, 4
1, 4
1, 0
5, 0
5, 5
4, 5
4, 4

959.8245620727539 ms

Apakah ingin menyimpan solusi? (y/n) n
```

Gambar 4.4 Tangkapan Layar penjalanan program tanpa file eksternal

```

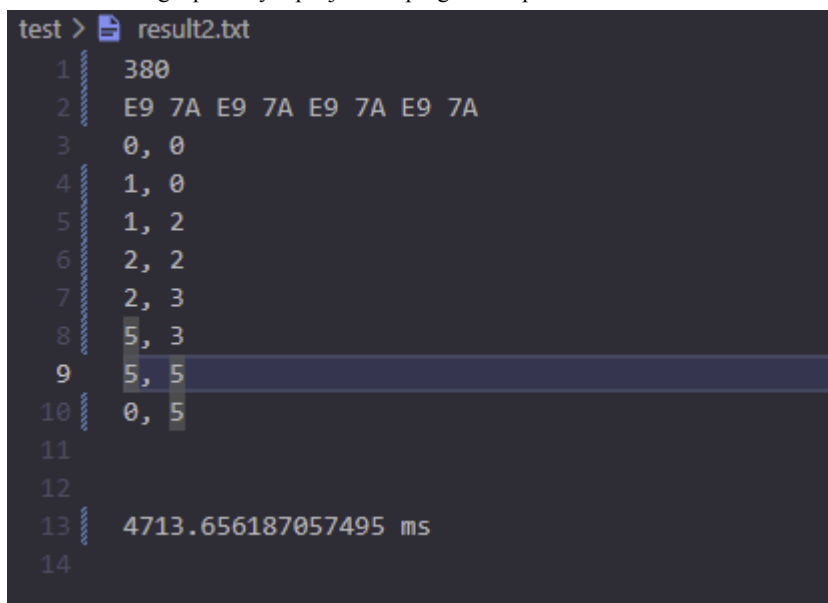
PS C:\Users\User\OneDrive - Institut Teknologi Bandung\Documents\Coding\Tucil1_13522008\src> python main.py
Masukkan jumlah token unik : 8
Masukkan token yang dipisahkan spasi : BD 1C 7A 55 E9 2B 9S A2
Masukkan ukuran buffer : 8
Masukkan ukuran matriks : 6 6
Masukkan jumlah sekuens : 3
Masukkan ukuran maksimal sekuens : 5
Matrix acak :
E9 7A A2 BD 7A 7A
7A 55 E9 1C 1C 2B
2B 9S 7A E9 BD 9S
A2 7A 7A 1C 9S BD
55 55 2B E9 7A A2
1C 55 2B 7A E9 E9
Sekuens acak :
('E9', '1C', '2B', '55') : 23
('55', '55', 'A2', '7A') : 36
('E9', '7A') : 95
380
E9 7A E9 7A E9 7A E9 7A
0, 0
1, 0
1, 2
2, 2
2, 3
5, 3
5, 5
0, 5

4713.656187057495 ms

Apakah ingin menyimpan solusi? (y/n) y
Masukkan nama file.txt : result2.txt

```

Gambar 4.5 Tangkapan Layar penjalanan program tanpa file eksternal



```

test > result2.txt
1 380
2 E9 7A E9 7A E9 7A E9 7A
3 0, 0
4 1, 0
5 1, 2
6 2, 2
7 2, 3
8 5, 3
9 5, 5
10 0, 5
11
12
13 4713.656187057495 ms
14

```

Gambar 4.6 Tangkapan Layar hasil yang disimpan pada file eksternal di gambar 4.5

## **BAB V**

### **PENUTUP**

#### 5.1 Kesimpulan

Program yang saya buat dapat menghitung jalur paling optimal untuk mendapat poin dalam *minigame* Cyberpunk 2077 Breach Protocol. Permasalahan ini diselesaikan menggunakan algoritma Brute Force, yaitu mencoba semua kemungkinan yang ada dan memilih yang terbaik.

#### 5.2 Saran

5.2.1 Memperbanyak *testcase* pada spesifikasi tugas kecil

#### 5.3 Komentar & Refleksi

Setelah mengerjakan tugas besar ini, pemahaman saya mengenai materi-materi yang dipelajari sepanjang kelas Strategi Algoritma sangat meningkat. Tugas kecil ini juga sangat membantu kami dalam memahami cara kerja Brute Force. Untuk kedepannya saya rasa manajemen waktu pengerjaan tugas kecil dapat ditingkatkan lagi agar tidak mepet dengan deadline..

## LAMPIRAN

| 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                             |    | ✓     |

## **PRANALA GITHUB REPOSITORY**

[https://github.com/frdmmm/Tucil1\\_13522008](https://github.com/frdmmm/Tucil1_13522008)