# Tugas Kecil 1 IF2211 Strategi Algoritma

# Penyelesaian Cyberpunk 2077 Breach Protocol

# dengan Algoritma Brute Force

# Semester II Tahun 2023/2024


Disusun oleh

Sa’ad Abdul Hakim (13522092)

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**BANDUNG**

# 2024

## Daftar Isi

# BAB 1

## Algoritma Bruteforce

### 1.1 Pendahuluan

Algoritma *Brute Force* adalah metode pencarian solusi yang langsung dan sistematis untuk memecahkan suatu persoalan. Pendekatan ini berdasarkan pada deskripsi problematika dan penerapan definisi atau konsep yang terlibat tanpa memanfaatkan informasi kontekstual atau struktur yang mungkin ada. Algoritma ini secara eksplisit mencoba semua kemungkinan solusi, seperti uji coba seluruh kombinasi kata sandi, kunci enkripsi, atau input lainnya, tanpa memanfaatkan pengetahuan tambahan yang dapat mempercepat proses pencarian.

Breach Protocol dalam permainan video Cyberpunk 2077 adalah sebuah *mini game* simulasi peretasan jaringan lokal yang menghadirkan tantangan meretas ICE (*Intrusion Countermeasures Electronics*). Dalam *mini game* ini, pemain harus mengelola komponen-komponen kunci, termasuk token yang terdiri dari dua karakter alfanumerik seperti E9, BD, dan 55, matriks yang berisi token-token untuk membentuk urutan kode, sekuens yang merupakan rangkaian token yang harus dicocokkan, dan buffer yang membatasi jumlah maksimal token yang dapat disusun secara sekuensial. Aturan permainan mencakup gerakan pemain dengan pola horizontal dan vertikal bergantian, pemilihan awal token pada baris paling atas matriks, pencocokan sekuens pada token di buffer, dan pengumpulan bobot hadiah atau reward yang bervariasi untuk setiap sekuens. Para pemain ditantang untuk mencocokkan sekuens dengan panjang minimal dua token, dengan tujuan mencapai kecocokan atau mengisi buffer hingga penuh.

### 1.2 Deskripsi Langkah

Program ini menerapkan algoritma *brute force* dalam menemukan solusi yang paling optimal berdasarkan komponen-komponen yang ada. Program akan mencari semua kemungkinan/kombinasi token yang ada sesuai dengan aturan permainan dan input pengguna. Seiring dengan itu program akan mengevaluasi *reward* dari setiap kemungkinan/kombinasi token untuk mencari kombinasi buffer yang paling optimal lalu kemudian ditampilkan pada output program.

# BAB 2

## *Source* Program

```
import random
import time



GREEN = "\033[32m"
END = "\033[0m"
print(f"""{GREEN}
 _____          _                                  _         _____  _____
______ ______
/  __ \        | |                                | |       / __  \|  _
||___  /|___  /
| /  \/ _   _ | |__    ___  _ __  _ __   _   _  _ __  | | __    `' / /'| |/' |
/ /    / /
| |    | | | || '_ \  / _ \| '__|| '_ \ | | | || '_ \ | |/ /      / /  |  /| |  /
/    / /
| \__/\| |_| || |_) ||  __/| |   | |_) || |_| || | | ||   <     ./ /___\ |_/ /./
/   ./ /
 \____/ \__, ||_.__/  \___||_|   | .__/  \__,_||_| |_||_|\_\    \_____/ \___/ \_/
\_/
         __/ |                   | |
        |___/                    |_|
______                          _         _____              _
_
| ___ \                        | |        | ___ \            | |
| |
| |_/ / _ __   ___   __ _   ___ | |__      | |_/ / _ __   ___  | |_   ___    ___
___  | |
| ___ \| '__| / _ \ / _` | / __|| '_ \     |  __/ | '__| / _ \ | __| / _ \  / __|
/ _ \ | |
| |_/ /| |   |  __/| (_| || (__ | | | |    | |    | |   | (_) || |_ | (_) || (__
| (_) || |
\____/ |_|    \___| \__,_| \___||_| |_|    \_|    |_|    \___/  \__| \___/  \___|
\___/ |_|
{END}""")
```

```
print("Selamat datang dalam permainan Breach Protocol")
print("Silahkan pilih masukan yang anda inginkan")
print("1. File .txt")
print("2. Masukkan manual")

pilih = int(input("=> "))
while pilih != 1 and pilih != 2:
    print("Masukkan salah")
    print("Silahkan pilih kembali")
    pilih = int(input("=> "))

if pilih == 1:
    filename = input("Masukkan nama file: ")
    file = open('../test/'+filename,'r')
    buffer_length = int(file.readline())
    content = file.readline()
    content = content.split()
    content = [int(num) for num in content]
    mwidth = content[0]
    mheight = content[1]
    matrix = [["" for i in range(mwidth)] for j in range(mheight)]
    for i in range(mheight):
        content = file.readline()
        content = content.split()
        content = [token for token in content]
        for j in range(mwidth):
            matrix[i][j] = content[j]
    content = int(file.readline())
    sequence_list = []
    for i in range(content):
        content = file.readline()
        templist = content.split()
        content = int(file.readline())
        sequence_list.append((content, templist))
    file.close()
else:
    token_count = int(input())
    token = input()
```

```
token = token.split()    print()
print("Matrix")
for row in matrix:
    for element in row:
        print(element, end=" ")
    print()
print()
print("Sequence")
for i in range(len(sequence_list)):
    sekuens = sequence_list[i][1]
    for element in sekuens:
        print(element, end=" ")
    print()
    print(sequence_list[i][0])
print()

buffer_length = int(input())
content = input()
content = content.split()
content = [int(num) for num in content]
mwidth = content[0]
mheight = content[1]
matrix = [["" for i in range(mwidth)] for j in range(mheight)]
for i in range(mheight):
    for j in range(mwidth):
        elmt = random.choice(token)
        matrix[i][j] = elmt
sequence_count = int(input())
max_sequence = int(input())
sequence_list = []
for i in range(sequence_count):
    length = random.randint(2, max_sequence)
    sequence = []
    for j in range(length):
        sequence.append(random.choice(token))
    if i > 0:
        for i in range(len(sequence_list)):
            listcheck = sequence_list[i][1]
            while listcheck == sequence:
```

```
                    sequence.pop()
                    sequence.append(random.choice(token))
        sequence_reward = random.randint(-100, 100)
        sequence_list.append((sequence_reward, sequence))


def reward_count(sequence, buffer):
    reward = 0
    for i in range(len(sequence)):
        sub_list = sequence[i][1]
        if is_sublist(sub_list, buffer):
            reward += sequence[i][0]
    return reward

def is_sublist(sub_list, list):
    for i in range(len(list)-len(sub_list)+1):
        if list[i:i + len(sub_list)] == sub_list:
            return True
    return False

def is_elmt(sublist, koordinat):
    return sublist in koordinat

def reward_check(rwrd, buff, koord):
    global reward, buffer, koordinat
    if rwrd >= reward:
        if rwrd == reward:
            if len(buff) < len(buffer):
                reward = rwrd
                buffer = buff.copy()
                koordinat = koord.copy()
        else:
            reward = rwrd
            buffer = buff.copy()
            koordinat = koord.copy()


def add_buffer(currbuffer, koor, bool, templength):
    if bool:
```

```
        i = koor[len(koor)-1][0]
        for j in range(mwidth):
            if j != koor[len(koor)-1][1] and not is_elmt([i, j], koor):
                currbuffer.append(matrix[i][j])
                koor.append([i, j])
                templength += 1
                bool = False
                reward = reward_count(sequence_list, currbuffer)
                reward_check(reward, currbuffer, koor)
                if templength < buffer_length:
                    add_buffer(currbuffer, koor, bool, templength)
                currbuffer.pop()
                koor.pop()
                templength -= 1
    else:
        j = koor[len(koor)-1][1]
        for i in range(mheight):
            if i != koor[len(koor)-1][0] and not is_elmt([i, j], koor):
                currbuffer.append(matrix[i][j])
                koor.append([i, j])
                templength += 1
                bool = True
                reward = reward_count(sequence_list, currbuffer)
                reward_check(reward, currbuffer, koor)
                if templength < buffer_length:
                    add_buffer(currbuffer, koor, bool, templength)
                currbuffer.pop()
                koor.pop()
                templength -= 1

reward = 0
buffer = []
koordinat = []
start = time.time()

if buffer_length != 0:
    for i in range(mwidth):
        buffertemp = []
        koordinattemp = []
```

```
        templength = 0
        buffertemp.append(matrix[0][i])
        koordinattemp.append([0, i])
        templength += 1
        rewardtemp = reward_count(sequence_list, buffertemp)
        if i == 0:
            reward = rewardtemp
            buffer = buffertemp.copy()
            koordinat = koordinattemp.copy()
        else:
            reward_check(rewardtemp, buffertemp, koordinattemp)
        if buffer_length > 1:
            add_buffer(buffertemp, koordinattemp, False, templength)
    print(reward)
    for element in buffer:
        print(element, end=' ')
    print()
    for i in range(len(koordinat)):
        print(koordinat[i][1]+1, end=", ")
        print(koordinat[i][0]+1)
    print()
else:
    print("Tidak ada solusi")
    print()

end = time.time()
print((end-start)*1000, end=" ms")
print()
save = input("Apakah ingin menyimpan solusi? (y/n) ")
save = save.lower()

if save == "y":
    filename = input("Masukkan nama file: ")
    file = open("../test/"+filename,"w")
    if buffer_length == 0:
        file.write("Tidak ada solusi\n")
    else:
        file.write(str(reward))
        file.write("\n")
```

```
        for element in buffer:
            file.write(str(element))
            file.write(" ")
        file.write("\n")
        for i in range(len(koordinat)):
            file.write(str(koordinat[i][1]+1))
            file.write(", ")
            file.write(str(koordinat[i][0]+1))
            file.write("\n")
        file.write("\n")
    file.write(str((end-start)*1000))
    file.write(" ms\n")
    file.close()
```

# BAB 3

## Testing Program

### 3.1 Tampilan Awal


### 3.2 Testing dengan Input File

| File Input | Output | File Output |
|---|---|---|

**input1.txt**

```
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
```

**Output**

```
Selamat datang dalam permainan Breach Protocol
Silahkan pilih masukan yang anda inginkan
1. File .txt
2. Masukkan manual
=> 1
Masukkan nama file: input1.txt
50
7A BD 7A BD 1C BD 55
1, 1
1, 4
3, 4
3, 5
6, 5
6, 3
1, 3

225.48151016235352 ms
Apakah ingin menyimpan solusi? (y/n) y
Masukkan nama file: output1.txt
```

**output1.txt**

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

225.48151016235352 ms
```

<table>
<tr>
<td>

input2.txt U

Tucil1_13522092 > test > input2.txt

```
7
7 7
7A 55 E9 E9 1C 55 1C
55 7A 1C 7A E9 55 BD
55 1C 1C 55 E9 BD 7A
BD 1C 7A 1C 55 BD 7A
BD 55 BD 7A 1C 1C 55
1C 55 55 7A 55 7A BD
7A BD E9 BD 7A E9 E9
4
E9 1C
10
BD 7A 55
20
E9 1C BD 55
30
7A 55 1C BD
40
```

</td>
<td>

```
Selamat datang dalam permainan Breach Protocol
Silahkan pilih masukan yang anda inginkan
1. File .txt
2. Masukkan manual
=> 1
Masukkan nama file: input2.txt
70
E9 1C BD 7A 55 1C BD
3, 1
3, 2
7, 2
7, 3
1, 3
1, 6
7, 6

1004.9424171447754 ms
Apakah ingin menyimpan solusi? (y/n) y
Masukkan nama file: output2.txt
```

</td>
<td>

output2.txt U

Tucil1_13522092 > test > output2.txt

```
70
E9 1C BD 7A 55 1C BD
3, 1
3, 2
7, 2
7, 3
1, 3
1, 6
7, 6

1004.9424171447754 ms

```

</td>
</tr>
<tr>
<td>

input3.txt U

Tucil1_13522092 > test > input3.txt

```
5
4 4
7A 1C 7A E9
BD BD 55 E9
E9 7A 1C 55
55 E9 BD 1C
3
BD E9
5
7A 1C
15
55 E9 BD
25
```

</td>
<td>

```
Selamat datang dalam permainan Breach Protocol
Silahkan pilih masukan yang anda inginkan
1. File .txt
2. Masukkan manual
=> 1
Masukkan nama file: input3.txt
40
7A 1C 55 E9 BD
3, 1
3, 3
4, 3
4, 2
1, 2

2.238750457763672 ms
Apakah ingin menyimpan solusi? (y/n) y
Masukkan nama file: output3.txt
```

</td>
<td>

output3.txt U

Tucil1_13522092 > test > output3.txt

```
40
7A 1C 55 E9 BD
3, 1
3, 3
4, 3
4, 2
1, 2

2.238750457763672 ms

```

</td>
</tr>
</table>

3.3 Testing dengan Input Terminal

| Input | Output | File Output |
|---|---|---|

<table>
<tr>
<td>

```
Selamat datang dalam permainan Breach Protocol
Silahkan pilih masukan yang anda inginkan
1. File .txt
2. Masukkan manual
=> 2
5
BD 1C 7A 55 E9
7
6 6
3
4
```

</td>
<td>

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

Sequence
E9 1C 7A BD
26
7A E9
-23
7A E9 BD E9
92

69
55 7A E9 BD E9
6, 1
6, 2
2, 2
2, 3
1, 3

220.02625465393066 ms
Apakah ingin menyimpan solusi? (y/n) y
Masukkan nama file: output4.txt
```

</td>
<td>

output4.txt U

Tucil1_13522092 > test > output4.txt

```
1   69
2   55 7A E9 BD E9
3   6, 1
4   6, 2
5   2, 2
6   2, 3
7   1, 3
8
9   220.02625465393066 ms
10
```

</td>
</tr>
<tr>
<td>

```
Selamat datang dalam permainan Breach Protocol
Silahkan pilih masukan yang anda inginkan
1. File .txt
2. Masukkan manual
=> 2
5
BD 1C 7A 55 E9
6
5 4
3
3
```

</td>
<td>

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

Sequence
E9 E9 E9
95
1C E9
-78
7A 55 7A
4

95
1C BD E9 E9 E9
1, 1
1, 2
4, 2
4, 4
1, 4

7.596015930175781 ms
Apakah ingin menyimpan solusi? (y/n) y
Masukkan nama file: output5.txt
```

</td>
<td>

output5.txt U

Tucil1_13522092 > test > output5.txt

```
1   95
2   1C BD E9 E9 E9
3   1, 1
4   1, 2
5   4, 2
6   4, 4
7   1, 4
8
9   7.596015930175781 ms
10
```

</td>
</tr>
</table>

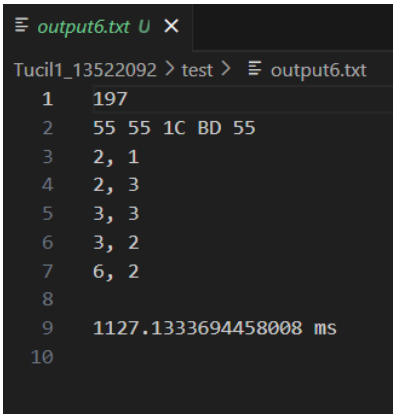
```
Selamat datang dalam permainan Breach Protocol
Silahkan pilih masukan yang anda inginkan
1. File .txt
2. Masukkan manual
=> 2
5
BD 1C 7A 55 E9
8
6 6
3
4
```

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

Sequence
55 55 1C
99
BD 55
93
55 1C
5

197
55 55 1C BD 55
2, 1
2, 3
3, 3
3, 2
6, 2

1127.1333694458008 ms
Apakah ingin menyimpan solusi? (y/n) y
Masukkan nama file: output6.txt
```



# Lampiran

Link repository: https://github.com/saadabha/Tucil1_13522092

| **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 | | ✓ |