

**Laporan Tugas Kecil 1 IF2211 Strategi Algoritma
Semester II Tahun Akademik 2023/2024**

**Penyelesaian *Cyberpunk 2077 Breach Protocol* Dengan
Algoritma Brute Force**



Disusun Oleh:
Edbert Eddyson Gunawan - 13522039
K-01

**Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2024**

Daftar Isi

Daftar Isi.....	1
1. Deskripsi Tugas.....	2
2. Algoritma Bruteforce.....	3
3. Source Code.....	5
3.1 main.py.....	5
3.2 RF.py.....	6
3.3 Solution.py.....	7
4. Test Case.....	9
GitHub.....	13

1. Deskripsi Tugas



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.

2. Algoritma Bruteforce

Pada laporan ini, algoritma yang digunakan oleh penulis adalah bruteforce. Berikut langkah-langkah penyelesaian

1. *Generate* semua kemungkinan pola token sesuai dengan panjang buffer. Misal, buffer berukuran 6 maka akan dibuat pola

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

1

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

2

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

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

4

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

5

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

6

Menghasilkan “7A 55 7A 55 E9 1C” hingga pola terakhir yaitu,

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

1

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

2

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

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

4

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

5

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

6

Yang menghasilkan pola “55 7A 55 1C 1C BD” dengan syarat **sel matriks (token) yang sudah di pilih, tidak dapat dipilih kembali.**

Disini, disimpan titik koordinat setiap token yang dipilih, dan *string* hasilnya dalam *array*.

2. Kemudian, dilakukan pengecekan *sequences* dengan himpunan solusi. Dengan cara mencari apakah *substring* dari solusi mengandung *sequences* yang ingin dicari.
3. Dengan meng-iterasi semua himpunan solusi, dilakukan pencatatan maksimum skor yang mungkin didapatkan. Jika maksimum skor ditemukan, maka catat juga *string* solusi dan titik koordinat setiap token yang dipilih.

3. Source Code

Penulis menerapkan prinsip OOP. Sehingga penulis membagi file menjadi 3 file yakni main.py, RF.py, Solution.py.

3.1 main.py

```
# libraries
from colorama import Style, Fore

# external files
import RF
import Solution

class Util:
    def __init__(self):
        pass

    def showBanner(self):
        print("Make Sure Your Input File is in test folder!")

class App:
    def __init__(self):
        self.rf = RF.RF()
        self.ut = Util()
        self.sol = None

        self.main()

    def main(self):
        if(self.rf.choice == 3):
            self.exit_program()
        elif(self.rf.choice == 1):
            self.rf.read_file()
        elif(self.rf.choice == 2):
            self.rf.generate_auto()

        self.sol = Solution.Solution(self.rf.matrix_size, self.rf.sequence, self.rf.matrix, self.rf.buffer_size)
        self.sol.main()

    def exit_program(self):
        exit(0)

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

3.2 RF.py

```
import random
import numpy as np

class RF:
    def __init__(self):
        self.fname = ''
        self.buffer_size = 0
        self.matrix_size = None
        self.matrix = []
        self.sequence_size = 0
        self.sequence = []
        self.choice = 0

        self.input_prompt()

    def input_prompt(self):
        choice = input("Apakah Anda ingin generate otomatis? (y/n)\n> ")
        if(choice == 'n'):
            self.choice = 1
        elif(choice == 'y'):
            self.choice = 2
        else:
            self.choice = 3

    def showBanner(self):
        print("Make Sure Your Input File is in test folder!")

    def generate_auto(self):
        token_n = input("Masukkan jumlah token\n> ")
        token = np.array(input("Masukkan token\n> ").split(' '))
        buffer_size = int(input("Masukkan ukuran buffer\n> "))
        matrix_size = list(map(lambda x: int(x), input("Masukkan ukuran matriks (row, col)\n> ").split(' ')))
        seq_size = int(input("Masukkan banyak sequence\n> "))
        seq_len = int(input("Masukkan maksimum panjang sequence\n> "))

        self.buffer_size = buffer_size
        self.matrix_size = matrix_size
        self.sequence_size = seq_size

        self.matrix = np.random.randint(0, token_n, size=(matrix_size[0], matrix_size[1]))
        self.matrix = token[self.matrix]
        print(self.matrix)

        rand_seq_size = np.random.randint(2, seq_len+1, size=(seq_size))

        self.sequence = []

        for x in rand_seq_size:
            self.sequence.append((list(token[(np.random.randint(0, token_n, size=(x)))]), random.randint(10, 50)))

        for x in self.sequence:
            print(" ".join(x[0]), "|", x[1])

    def read_file(self):
        self.fname = "test/" + input('input filename\n> ')
        f = open(self.fname, 'r')
        f = f.read()
        f = f.split('\n')
        f[:] = [x for x in f if x.strip('\n')]

        # assign values
        self.buffer_size = int(f[0])
        self.matrix_size = list(map(lambda x: int(x), f[1].split(' ')))
        self.matrix = list(map(lambda x: x.split(' '), f[2:2+self.matrix_size[0]]))
        self.sequence_size = int(f[2+self.matrix_size[0]])
        for i in range(self.sequence_size):
            seq, val = f[2+self.matrix_size[0] + 1 + 2 * i], f[2+self.matrix_size[0] + 2 + 2 * i]
            self.sequence.append((seq.split(' '), int(val)))
```

3.3 Solution.py

```
import numpy as np
import time
import multiprocessing

class Solution:
    def __init__(self, matrix_size, sequence, matrix, buffer_size):
        self.row = matrix_size[0]
        self.col = matrix_size[1]
        self.sequence = sequence
        self.matrix = matrix
        self.ans_pool = []
        self.max_depth = buffer_size
        self.max_score = 0
        self.max_coor = []
        self.time_elapsed = 0

        if (buffer_size > sum([x[1] for x in sequence])):
            self.max_depth = sum([x[1] for x in sequence])

    def main(self):
        visited = np.zeros((self.row, self.col))

        start_t = time.time()
        self.brute_force(0, 0, True, "", 0, [], visited)
        self.check_ans()
        end_t = time.time()

        self.time_elapsed = end_t - start_t

        self.show_banner()
        self.show_answer()

    def brute_force(self, row, col, b, ans, depth, coor, visited):
        if (depth == self.max_depth): # basis max depth
            # print("called")
            self.ans_pool.append((ans, coor))
            return

        if (b): # b = True -> Horizontal
            for j in range(self.col):
                visited2 = np.copy(visited)
                if (visited2[row][j] == 0):
                    visited2[row][j] = 1
                    ans2 = ans + self.matrix[row][j] + " "
                    self.brute_force(row, j, not b, ans2, depth+1, [*coor, (row, j)], visited2)
        else: # b = False -> Vertical
            for i in range(self.row):
                visited2 = np.copy(visited)
                if (visited2[i][col] == 0):
                    visited2[i][col] = 1
                    ans2 = ans + self.matrix[i][col] + " "
                    self.brute_force(i, col, not b, ans2, depth+1, [*coor, (i, col)], visited2)

    def printMatrix(self, visited):
        for x in list(map(lambda x: " ".join(str(x)), visited)):
            print(x)
        print("=====")

    def check_ans(self):
        max_score = 0
        max_coor = []
        # print(self.ans_pool)
        for x in self.ans_pool:
            temp_score = 0
            for i in range(len(self.sequence)):
                s = " ".join(self.sequence[i][0])
                # print(s, " | ", x)
                if s in x[0]:
                    # print(s, " | ", x)
                    temp_score += self.sequence[i][1]

            # print(temp_score)
            if (temp_score > max_score):
                max_score = temp_score
                max_coor = x[1]

        self.max_score = max_score
        self.max_coor = max_coor
```



```

def show_banner(self):
    print("=====")
    print("|                SOLUTION                |")
    print("=====")

def show_answer(self):
    print("Max score:\t", self.max_score)
    print("Sequence:\t", " ".join(list(map(lambda x: self.matrix[x[0]][x[1]], self.max_coor))))
    print("Coordinates:\n", "\n".join(list(map(lambda x: "("+str(x[0]+1)+" "+str(x[1]+1)+")", self.max_coor))))
    print("Time elapsed:\t", round(self.time_elapsed * 1000, 2), "ms")
    choice = input("Apakah anda ingin menyimpan solusi? (y/n)\n> ")

    if(choice == 'y'):
        fname = input("Masukkan nama file\n> ")
        self.saveToFile(fname)

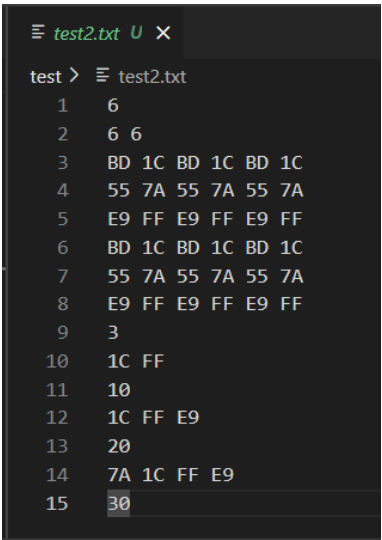
def saveToFile(self, fname):
    fname = "test/solution-" + fname # "solution-test1.txt"
    f = open(fname, "w")

    f.write("=====\n")
    f.write("|                SOLUTION                |\n")
    f.write("=====\n")
    f.write("Max score:\t{}\n".format(self.max_score))
    f.write("Sequence:\t{}\n".format(" ".join(list(map(lambda x: self.matrix[x[0]][x[1]], self.max_coor))))))
    f.write("Coordinates:\n{}\n".format("\n".join(list(map(lambda x: "("+str(x[0]+1)+" "+str(x[1]+1)+")",
self.max_coor))))))
    f.write("Time elapsed:\t{} ms".format(round(self.time_elapsed*1000, 2)))

    print(f"File saved in {fname}")

```

4. Test Case

Test Case Input	Output
File test1.txt 8 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	<pre> Apakah Anda ingin generate otomatis? (y/n) > n input filename > test1.txt ===== SOLUTION ===== Max score: 50 Sequence: 7A BD 7A BD 1C BD 55 55 Coordinates: (1, 1) (4, 1) (4, 3) (5, 3) (5, 6) (3, 6) (3, 1) (2, 1) Time elapsed: 1587.97 ms Apakah anda ingin menyimpan solusi? (y/n) > n PS C:\Users\Asus Tuf Gaming\Documents\Github Desktop\Tucil1_13522039> </pre>
 <pre> test > test2.txt 1 6 2 6 6 3 BD 1C BD 1C BD 1C 4 55 7A 55 7A 55 7A 5 E9 FF E9 FF E9 FF 6 BD 1C BD 1C BD 1C 7 55 7A 55 7A 55 7A 8 E9 FF E9 FF E9 FF 9 3 10 1C FF 11 10 12 1C FF E9 13 20 14 7A 1C FF E9 15 30 </pre>	<pre> Apakah Anda ingin generate otomatis? (y/n) > n input filename > test2.txt ===== SOLUTION ===== Max score: 30 Sequence: BD BD 1C FF E9 55 Coordinates: (1, 1) (4, 1) (4, 2) (3, 2) (3, 1) (2, 1) Time elapsed: 70.0 ms Apakah anda ingin menyimpan solusi? (y/n) > y Masukkan nama file > test2.txt File saved in test/solution-test2.txt PS C:\Users\Asus Tuf Gaming\Documents\Github Desktop\Tucil1_13522039> </pre>

	<pre> ≡ solution-test2.txt U X test > ≡ solution-test2.txt 1 ===== 2 SOLUTION 3 ===== 4 Max score: 30 5 Sequence: BD BD 1C FF E9 55 6 Coordinates: 7 (1, 1) 8 (4, 1) 9 (4, 2) 10 (3, 2) 11 (3, 1) 12 (2, 1) 13 Time elapsed: 70.0 ms </pre>
<pre> Apakah Anda ingin generate otomatis? (y/n) > y Masukkan jumlah token > 5 Masukkan token > AA BB CC DD EE Masukkan ukuran buffer > 7 Masukkan ukuran matriks (row, col) > 8 8 Masukkan banyak sequence > 3 Masukkan maksimum panjang sequence > 5 [['BB' 'EE' 'CC' 'EE' 'DD' 'EE' 'AA' 'DD'] ['DD' 'EE' 'CC' 'AA' 'CC' 'DD' 'EE' 'AA'] ['DD' 'DD' 'CC' 'DD' 'CC' 'AA' 'EE' 'CC'] ['AA' 'BB' 'DD' 'DD' 'EE' 'BB' 'CC' 'CC'] ['EE' 'AA' 'AA' 'DD' 'CC' 'EE' 'DD' 'BB'] ['CC' 'DD' 'AA' 'EE' 'DD' 'AA' 'DD' 'AA'] ['BB' 'EE' 'BB' 'AA' 'CC' 'EE' 'EE' 'EE'] ['CC' 'AA' 'BB' 'AA' 'BB' 'AA' 'BB' 'EE']] AA EE BB 12 AA BB 30 AA CC EE DD 11 </pre>	<pre> ===== SOLUTION ===== Max score: 42 Sequence: BB DD AA BB AA EE BB Coordinates: (1, 1) (2, 1) (2, 8) (5, 8) (5, 2) (7, 2) (7, 1) Time elapsed: 4034.09 ms Apakah anda ingin menyimpan solusi? (y/n) > n </pre>
<pre> Apakah Anda ingin generate otomatis? (y/n) > y Masukkan jumlah token > 7 Masukkan token > AA BB CC DD EE FF GG Masukkan ukuran buffer > 4 Masukkan ukuran matriks (row, col) > 8 8 Masukkan banyak sequence > 5 Masukkan maksimum panjang sequence > 4 [['CC' 'BB' 'FF' 'EE' 'CC' 'AA' 'DD' 'CC'] ['CC' 'CC' 'DD' 'FF' 'AA' 'FF' 'GG' 'DD'] ['FF' 'GG' 'BB' 'EE' 'BB' 'FF' 'CC' 'BB'] ['DD' 'EE' 'CC' 'BB' 'GG' 'DD' 'EE' 'BB'] ['BB' 'AA' 'CC' 'EE' 'CC' 'GG' 'DD' 'CC'] ['FF' 'CC' 'FF' 'FF' 'BB' 'AA' 'GG' 'CC'] ['DD' 'CC' 'EE' 'AA' 'FF' 'AA' 'FF' 'EE'] ['FF' 'FF' 'AA' 'DD' 'CC' 'EE' 'GG' 'FF']] DD DD 40 AA EE 26 FF EE GG EE 26 EE FF 16 AA CC 26 </pre>	<pre> ===== SOLUTION ===== Max score: 66 Sequence: DD DD AA CC Coordinates: (1, 7) (5, 7) (5, 2) (2, 2) Time elapsed: 14.16 ms Apakah anda ingin menyimpan solusi? (y/n) > n </pre>

```

Apakah Anda ingin generate otomatis? (y/n)
> y
Masukkan jumlah token
> 5
Masukkan token
> AA BB CC DD EE
Masukkan ukuran buffer
> 8
Masukkan ukuran matriks (row, col)
> 6 6
Masukkan banyak sequence
> 4
Masukkan maksimum panjang sequence
> 6
[['CC' 'EE' 'EE' 'EE' 'AA' 'EE']
 ['DD' 'CC' 'AA' 'DD' 'AA' 'CC']
 ['DD' 'EE' 'DD' 'DD' 'AA' 'DD']
 ['EE' 'BB' 'AA' 'CC' 'CC' 'AA']
 ['BB' 'CC' 'AA' 'AA' 'EE' 'EE']
 ['BB' 'CC' 'EE' 'AA' 'DD' 'CC']]
CC EE AA | 46
BB AA AA CC DD AA | 30
DD CC CC CC DD | 13
DD EE DD BB EE BB | 26

```

```

=====
|                               SOLUTION                               |
=====
Max score:      46
Sequence:      CC DD CC EE AA AA AA EE
Coordinates:
(1, 1)
(2, 1)
(2, 2)
(1, 2)
(1, 5)
(2, 5)
(2, 3)
(1, 3)
Time elapsed:   1870.73 ms
Apakah anda ingin menyimpan solusi? (y/n)
> y
Masukkan nama file
> generated1.txt
File saved in test/solution-generated1.txt

```

```

≡ solution-generated1.txt U X
test > ≡ solution-generated1.txt
1  =====
2  |                               SOLUTION                               |
3  =====
4  Max score:  46
5  Sequence:   CC DD CC EE AA AA AA EE
6  Coordinates:
7  (1, 1)
8  (2, 1)
9  (2, 2)
10 (1, 2)
11 (1, 5)
12 (2, 5)
13 (2, 3)
14 (1, 3)
15 Time elapsed: 1870.73 ms

```

```

Apakah Anda ingin generate otomatis? (y/n)
> y
Masukkan jumlah token
> 5
Masukkan token
> AA BB CC DD EE
Masukkan ukuran buffer
> 4
Masukkan ukuran matriks (row, col)
> 2 2
Masukkan banyak sequence
> 3
Masukkan maksimum panjang sequence
> 5
[['EE' 'CC']
 ['CC' 'EE']]
CC CC BB EE BB | 44
BB CC | 15
AA CC BB | 47

```

```

=====
|                SOLUTION                |
=====
Max score:      0
Sequence:
Coordinates:

Time elapsed:    0.0 ms
Apakah anda ingin menyimpan solusi? (y/n)
> y
Masukkan nama file
> generated-empty.txt
File saved in test/solution-generated-empty.txt

```

≡ *solution-generated-empty.txt* U ×

test > ≡ solution-generated-empty.txt

```

1  =====
2  |                SOLUTION                |
3  =====
4  Max score:  0
5  Sequence:
6  Coordinates:
7
8  Time elapsed:  0.0 ms

```

GitHub

Kode dapat diakses pada laman https://github.com/WazeAzure/Tucil1_13522039

Jika tidak dapat diakses, mohon kontak penulis di LINE yenyenhui atau TEAMS.