NAMA : Renaldy Arief Susanto
NIM : 13522022

LAPORAN TUGAS KECIL 1 IF2211 STRATEGI ALGORITMA

A. Deskripsi Permasalahan

Cyberpunk 2077 Breach Protocol adalah minigame meretas pada permainan video Cyberpunk 2077. *Minigame* ini merupakan simulasi peretasan jaringan lokal 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.

NAMA : Renaldy Arief Susanto
NIM : 13522022

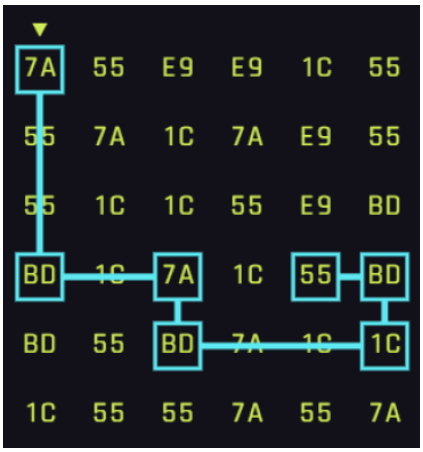
7A	55	E9	E9	1C	55
55	7A	1C	7A	E9	55
55	1C	1C	55	E9	BD
BD	1C	7A	1C	55	BD
BD	55	BD	7A	1C	1C
1C	55	55	7A	55	7A

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
- Isi buffer : **7A BD 7A BD 1C BD 55**



Gambar 1 Solusi optimal pada contoh kasus (Sumber: https://cyberpunk-hacker.com/)

Mahasiswa ditugaskan untuk membuat program sederhana dalam bahasa C/C#/C++/Java/Python/JavaScript/Go/PHP yang mengimplementasikan algoritma Brute Force untuk mencari solusi paling optimal permainan Breach Protocol

B. Desain Algoritma

Saya merancang algoritma dengan langkah-langkah sebagai berikut.

1. Semua kemungkinan jalur yang dapat dimainkan pada matriks ditelusuri dan dimuat ke dalam sebuah *array*.
2. Semua jalur tersebut kemudian dikonversi menjadi sebuah *string*.
3. Untuk setiap *string* tersebut, dilakukan pengujian ada/tidaknya tiap-tiap sekuens yang diberikan satu per satu. Nilai atau bobot dari jalur tersebut akan ditambah dengan setiap nilai atau bobot dari sekuens yang ditemukan (nilainya akan 0 jika tidak ditemukan sekuens sama sekali).
4. Seluruh nilai akan dibandingkan satu per satu untuk dicari maksimumnya. Jalur yang mempunyai nilai maksimum dan buffer terpendek akan menjadi *output* dari program.

Algoritma ini bersifat *brute force* karena seluruh kemungkinan diuji satu per satu dan program tidak diberikan logika untuk menyaring solusi atau pun berhenti di tengah jalan.

Adapun Kompleksitas dari algoritma terdiri dari dua komponen: traversal jalur dan pencarian *string* sekuens. Traversal jalur mempunyai kompleksitas O(N * M * N * M … * N) dengan N x M adalah dimensi matriks serta banyaknya N ditambah banyaknya M adalah ukuran buffer. Untuk setiap jalur itu, kompleksitas pencarian nilai/bobot dari sekuens adalah O(M * N) dengan M adalah panjang buffer dan N adalah banyaknya sekuens. Secara keseluruhan, kompleksitasnya adalah O((N *M * N * … * M) * s * k), dengan keterangan N x M adalah dimensi matriks, s adalah panjang buffer maksimal, dan k adalah banyaknya sekuens yang diberikan.

C. Implementasi Program

Program dibuat dalam bahasa Python dengan antarmuka berupa CLI (*Command Line Input*). Program dimulai dengan meminta sebuah masukan awal agar pengguna bisa memilih antara dua metode input: *file input* atau CLI (sebagaimana ditentukan

dalam spesifikasi). Program kemudian memberikan instruksi pada pengguna untuk memasukkan parameter permainan.

Struktur program dibagi menjadi tiga file inti: file *main.py*, *inputoutput.py* dan *solver.py*. Sesuai namanya, main adalah file yang dijalankan, inputoutput adalah file yang berisi fungsi untuk I/O program, dan solver berisi algoritma untuk menyelesaikan permasalahan. Pada bab ini, saya akan bahas fungsi-fungsi pada file solver saja karena proses input output kurang relevan dengan konsep yang ditekankan pada tugas ini.

Dalam implementasinya, saya membuat program ini dengan banyak menggunakan metode *built-in* yang sudah tersedia pada Python Standard Library. Berikut akan dibahas beberapa poin pengimplementasian yang saya rasa paling penting.

1. Algoritma *path generation* atau penelusuran semua jalur.

```python
# Functions to generate the paths
def generatePath(matrix, i, j, buffer : int, path : list[str], vertical : bool, result) -> None :

    result.append(path.copy())
    if buffer == 0 :
        return

    else :

        if vertical :
            for newi in range(len(matrix)) :
                if (newi,j) not in path :
                    path.append((newi,j))
                    generatePath(matrix, newi, j, buffer-1, path, not(vertical), result)
                    path.pop()

        else :
            for newj in range(len(matrix[0])) :
                if (i,newj) not in path :
                    path.append((i,newj))
                    generatePath(matrix, i, newj, buffer-1, path, not(vertical), result)
                    path.pop()

def generateAllPaths(matrix : list[list[str]], buffer : int) -> list[list[str]] :
    result = list()
    for j in range(len(matrix[0])) :
        path = [(0,j)]
        generatePath(matrix, i=0, j=j, buffer=buffer-1, path=path, vertical=True, result = result)
```

Gambar 2 : tangkapan layar *Source Code* program (1)

Fungsi ini berjalan dimulai dari baris pertama matriks, menelusuri seluruh kemungkinan secara rekursif dengan syarat penghentiannya adalah ukuran buffer nol. Poin penting pertama dari fungsi ini adalah jalur tidak diperbolehkan untuk mengandung sebuah koordinat/token yang sama

sehingga harus dicek dahulu sebelum ditambahkan ke *path*. Kedua, perlu ada variabel yang menyatakan apakah penelusuran sedang berada pada keadaan vertikal atau horizontal. Terakhir, seluruh jalur, termasuk yang belum sampai pemberhentian atau buffer nol tetap dimasukkan ke dalam array *result* karena semua kemungkinan ukuran buffer juga diinginkan.

2. Algoritma penentuan bobot sebuah jalur.

```
def pathValue(
            path_str : str,
            sequenceValues : list[int],
            sequences : list[str]
            ) -> int :

    ans = 0
    for seq,val in zip(sequences, sequenceValues) :
        idx = path_str.find(seq)
        if idx > -1 and idx % 2 == 0 :
            ans += val
    return ans
```

Gambar 3 : tangkapan layar *Source Code* program (2)

Pada fungsi ini, dicari bobot dari sebuah jalur. Sebenarnya yang dilakukan oleh fungsi ini adalah menguji apakah tiap sekuens merupakan sebuah *subarray* dari sebuah jalur. Poin penting yang ingin saya tekankan di sini adalah penggunaan metode bawaan `.find()`. Untuk menggunakan `.find()`, baik buffer mau pun sekuens perlu diubah menjadi *string* terlebih dahulu. Efek samping dari pengubahan tersebut adalah sekuens dapat ditemukan pada buffer tidak sebagaimana mestinya. Sebagai contoh, misalkan sebuah buffer terdiri dari BA AB AA, sementara sebuah sekuens adalah AA BA. Tentunya, sekuens tersebut tidak ada pada buffer. Namun, jika `.find()` digunakan, *substring* AA BA akan ditemukan pada indeks 1 (B**A AB A**A) karena keduanya telah dianggap menjadi sebuah *string*. Oleh karena itu, perlu dilakukan pengecekan hasil `.find()` untuk memastikan indeks *substring* yang dihasilkan berada pada indeks genap.

3. Algoritma inti

```
def solve(matrix, buffer, sequences, sequenceValues) :

    # Generate all possible paths
    paths = generateAllPaths(matrix, buffer)

    # Convert all paths to token strings, store in second array
    path_strs = [pathToString(matrix, path) for path in paths]
    ans,ansidx = 0,0

    # Each path is evaluated. Find the index of the max/best path as well as its value
    for idx,path in enumerate(path_strs) :
        v = pathValue(path, sequenceValues, sequences)
        if v > ans :
            ans = v
            ansidx = idx
        elif v == ans and len(path_strs[idx]) < len(path_strs[ansidx]) :
            ansidx = idx

    bestpathstr = path_strs[ansidx]
    bestpathstr = " ".join([bestpathstr[i:i+2] for i in range(0, len(bestpathstr), 2)])
    bestpath = paths[ansidx]

    return ans,bestpathstr,bestpath
```

Gambar 4 : tangkapan layar *Source Code* program (3)

Di atas adalah fungsi utama yang dipanggil pada fungsi *main* untuk mendapatkan luaran. Fungsi ini melakukan langkah-langkah yang sudah diutarakan pada Bab B. Ada pun format luaran fungsi ini adalah poin terbesar, jalur dengan poin terbesar, dan isi buffer dari jalur tersebut.

4. Penggunaan *library* tambahan

   Program tidak menggunakan library di luar Python Standard Library, tetapi menggunakan fungsi `product()` dari library `itertools`, serta `randint`, `choice`, dan `sample` dari library `random` untuk mempersingkat pembuatan sekuens dan matriks random pada mode CLI. Berikut penggunaannya.

NAMA : Renaldy Arief Susanto
NIM : 13522022

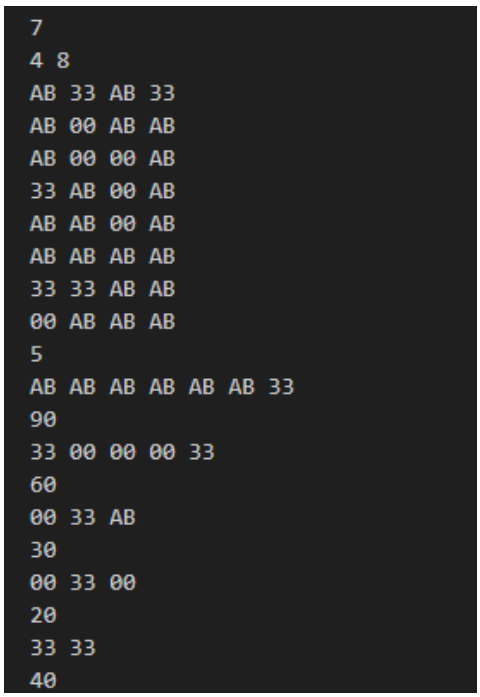
```python
from random import randint,choice,sample
from itertools import product
def generateRandom(
                    seqs_length : int,
                    maxseqlength : int,
                    tokens : set[str],
                    m_width : int,
                    m_height : int
                ) :

    tokens = list(tokens)
    matrix = [[choice(tokens) for i in range(m_width)] for j in range(m_height)]
    possibleseqs = []
    for l in range(2, maxseqlength + 1) :
        possibleseqs += product(tokens, repeat=l)
    seqs = sample(possibleseqs, seqs_length)
    seqs = list(map("".join, seqs))
    seqvals = [randint(-100, 100) for i in range(seqs_length)]

    return matrix, seqs, seqvals
```

Gambar 5 tangkapan layar *Source Code* program (4)

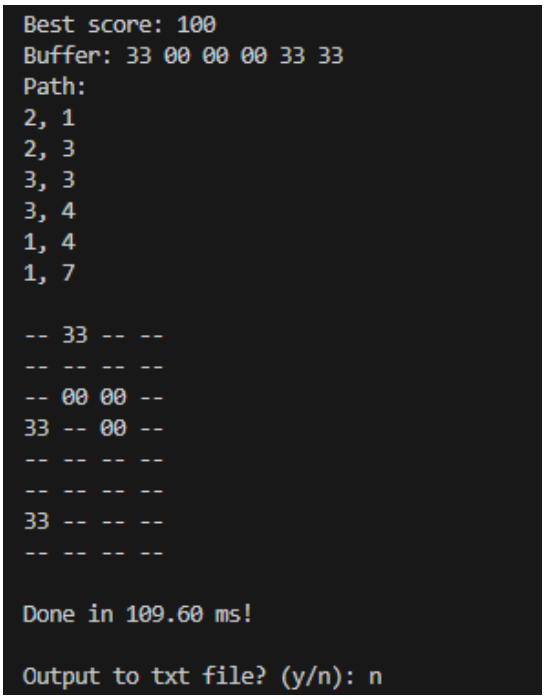
NAMA : Renaldy Arief Susanto
NIM : 13522022

D. Uji Kasus

1. Uji Kasus 1

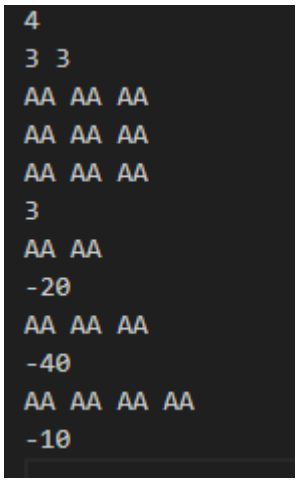


```
7
4 8
AB 33 AB 33
AB 00 AB AB
AB 00 00 AB
33 AB 00 AB
AB AB 00 AB
AB AB AB AB
33 33 AB AB
00 AB AB AB
5
AB AB AB AB AB AB 33
90
33 00 00 00 33
60
00 33 AB
30
00 33 00
20
33 33
40
```

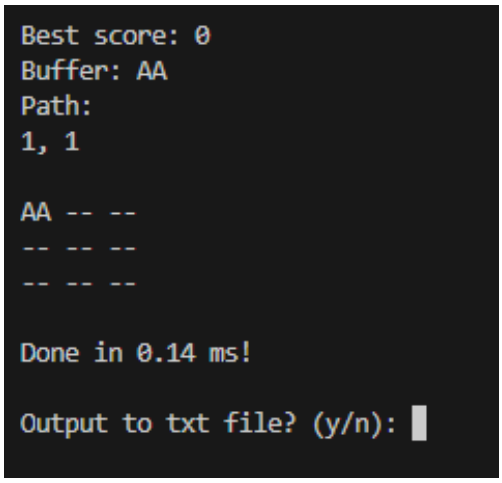


```
Best score: 100
Buffer: 33 00 00 00 33 33
Path:
2, 1
2, 3
3, 3
3, 4
1, 4
1, 7

-- 33 -- --
-- -- -- --
-- 00 00 --
33 -- 00 --
-- -- -- --
-- -- -- --
33 -- -- --
-- -- -- --

Done in 109.60 ms!

Output to txt file? (y/n): n
```

2. Uji Kasus 2



```
4
3 3
AA AA AA
AA AA AA
AA AA AA
3
AA AA
-20
AA AA AA
-40
AA AA AA AA
-10
```



```
Best score: 0
Buffer: AA
Path:
1, 1

AA -- --
-- -- --
-- -- --

Done in 0.14 ms!

Output to txt file? (y/n):
```

NAMA : Renaldy Arief Susanto
NIM : 13522022

3. Uji Kasus 3

```
9
6 6
KO KO RO MI SA KI
MI SA KI KO KO RO
KO KO RO MI SA KI
MI SA KI KO KO RO
KO KO RO MI SA KI
MI SA KI KO KO RO
10
KO KO RO
50
MI SA KI
50
KO KO RO MI SA KI
50
MI SA KI KO KO RO
50
KO KO KO KO
50
RO RO RO RO
200
MI MI MI MI
200
SA SA SA SA
200
KI KI KI KI
200
MI SA KO KO
100
```

```
Best score: 200
Buffer: KO MI SA KO KO KO KO KO RO
Path:
1, 1
1, 2
2, 2
2, 3
1, 3
1, 5
2, 5
2, 1
3, 1

KO KO RO -- -- --
MI SA -- -- -- --
KO KO -- -- -- --
-- -- -- -- -- --
KO KO -- -- -- --
-- -- -- -- -- --

Done in 7744.88 ms!

Output to txt file? (y/n): n
Goodbye!
PS D:\GitHub\Tucil1_13522022>
```

4. Uji Kasus 4

```
7
3 3
AA BB CC
DD EE FF
GG HH II
10
AA GG HH BB CC II
1
AA DD EE BB CC FF
2
CC II HH EE DD AA
3
DD AA
-1
EE DD
-1
AA BB
1000
HH DD
1000
EE FF DD
1000
HH BB EE
1000
AA HH
1000
EE II
1000
```

```
Best score: 1001
Buffer: CC II HH EE DD AA BB
Path:
3, 1
3, 3
2, 3
2, 2
1, 2
1, 1
2, 1

AA BB CC
DD EE --
-- HH II

Done in 0.60 ms!

Output to txt file? (y/n): n
Goodbye!
PS D:\GitHub\Tucil1_13522022>
```

NAMA : Renaldy Arief Susanto
NIM : 13522022

5. Uji Kasus 5

```
7
7 7
PG D4 PG 7C 7C 7C 7C
PG PG D4 7C 7C 1A 7C
PG D4 1A 1A 1A D4 D4
7C 7C D4 7C D4 7C 7C
D4 7C PG PG 7C 1A 1A
D4 1A PG 1A D4 D4 1A
D4 7C 7C 1A 7C PG 7C
6
D4 7C 1A
211
PG D4 7C
125
7C PG D4 PG
220
1A PG
272
1A 7C 1A
282
PG 1A 7C
253
```

```
Best score: 1018
Buffer: D4 7C 1A PG 1A 7C 1A
Path:
2, 1
2, 5
6, 5
6, 7
4, 7
4, 2
6, 2

-- D4 -- -- -- -- --
-- -- -- 7C -- 1A --
-- -- -- -- -- -- --
-- -- -- -- -- -- --
-- 7C -- -- -- 1A --
-- -- -- -- -- -- --
-- -- -- 1A -- PG --

Done in 1072.73 ms!

Output to txt file? (y/n):
```

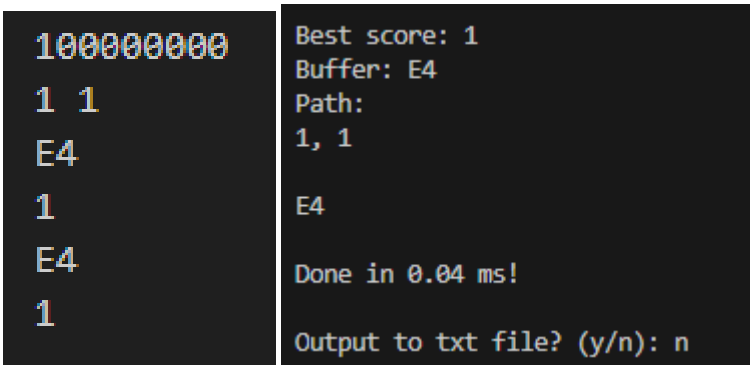
6. Uji Kasus 6 – CLI dengan parameter matriks 20x20, buffer 5, banyak sekuens 15 dan panjang maksimum sekuens 5.

```
---------*----------
Generated matrix:
QQ AA JK JK JK QQ B7 V4 K3 2L JK AA QQ QQ V4
X2 AA B7 V4 X2 X2 X2 2L 91 91 JK AA 2L AA K3
AA AA 91 X2 QQ K3 B7 2L X2 2L L9 V4 V4 B7 K3
K3 X2 K3 QQ K3 QQ B7 V4 B7 JK AA B7 JK 91 X2
V4 B7 JK V4 K3 QQ B7 X2 QQ 91 QQ K3 2L X2 K3
91 JK B7 X2 L9 2L K3 JK B7 K3 QQ K3 JK JK QQ
QQ X2 V4 AA QQ AA K3 K3 V4 L9 K3 JK X2 L9 JK
2L 91 QQ V4 X2 91 K3 X2 AA QQ L9 91 AA 91 2L
B7 L9 AA L9 91 JK QQ B7 L9 91 V4 QQ 2L 2L AA
91 2L QQ X2 L9 K3 QQ JK X2 AA 2L B7 V4 K3 B7
V4 2L V4 JK V4 QQ L9 2L X2 V4 AA L9 K3 JK V4
K3 V4 QQ 91 V4 V4 91 K3 K3 K3 JK QQ 2L B7 JK
QQ V4 X2 V4 V4 QQ K3 L9 JK 2L JK 2L 91 2L AA
L9 AA 2L L9 K3 B7 2L 91 L9 2L QQ V4 AA K3 X2
QQ X2 V4 91 2L 91 QQ AA X2 JK 2L 2L B7 K3 K3
```

```
Generated sequences:
QQ X2 QQ K3 X2
-60
AA JK K3 91 K3
-34
QQ K3 B7 2L V4
-94
B7 AA V4 91 B7
-63
B7 V4 K3 B7 X2
-18
AA X2 QQ QQ
-28
B7 B7 91 B7 JK
-3
L9 2L V4 X2
-45
K3 B7 X2 V4 2L
-47
QQ L9 JK 91 QQ
76
X2 X2 JK V4 K3
36
AA B7 L9 X2
-94
QQ JK AA 2L 2L
37
2L X2 X2 2L K3
-67
```

```
Best score: 76
Buffer: QQ L9 JK 91 QQ
Path:
14, 1
14, 7
12, 7
12, 8
3, 8

-- -- -- -- -- -- -- -- -- -- -- -- -- QQ --
-- -- -- -- -- -- -- -- -- -- -- -- -- -- --
-- -- -- -- -- -- -- -- -- -- -- -- -- -- --
-- -- -- -- -- -- -- -- -- -- -- -- -- -- --
-- -- -- -- -- -- -- -- -- -- -- -- -- -- --
-- -- -- -- -- -- -- -- -- -- -- -- -- -- --
-- -- -- -- -- -- -- -- -- -- -- -- JK -- L9 --
-- -- QQ -- -- -- -- -- -- -- -- -- 91 -- -- --
-- -- -- -- -- -- -- -- -- -- -- -- -- -- --
-- -- -- -- -- -- -- -- -- -- -- -- -- -- --
-- -- -- -- -- -- -- -- -- -- -- -- -- -- --
-- -- -- -- -- -- -- -- -- -- -- -- -- -- --
-- -- -- -- -- -- -- -- -- -- -- -- -- -- --
-- -- -- -- -- -- -- -- -- -- -- -- -- -- --
-- -- -- -- -- -- -- -- -- -- -- -- -- -- --

Done in 2268.28 ms!

Output to txt file? (y/n): n
Goodbye!
PS D:\GitHub\Tucil1_13522022>
```

7. Uji Kasus 7 – Edge case



E. Tautan Repository GitHub

https://github.com/AldyDPP/Tucil1_13522022