LAPORAN TUGAS KECIL 1

IF2211 Strategi Algoritma

Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force

Disusun Oleh :

Abdul Rafi Radityo Hutomo (13522089)

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

Daftar Isi

BAB 1

Deskripsi Permasalahan

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.

Pada tugas kecil ini, dibuat sebuah program untuk mencari solusi yang paling optimal dari permainan breach protocol dengan menggunakan algoritma brute force.

BAB 2
Algoritma Program

Pada Tugas Kecil ini diimplementasikan algoritma brute force, yaitu dengan cara menghitung nilai buffer untuk setiap urutan gerakan yang dapat dilakukan pada permainan ini kemudian mengambil urutan gerakan yang menghasilkan nilai buffer maksimal. Secara rinci algoritma dilakukan sesuai dengan langkah berikut :

1. Mulai dengan buffer kosong
2. Hitung nilai buffer pada kondisi ini
3. Pilih salah satu *Starting Token* yang berada pada baris teratas dan masukkan ke dalam buffer
4. Hitung nilai buffer pada kondisi ini, bandingkan dengan nilai maks yang ditemui sejauh ini, apabila lebih optimal (lebih besar nilainya atau bernilai sama tetapi dengan ukuran buffer lebih kecil) maka simpan nilai tersebut, serta langkah-langkah yang dibutuhkan untuk ke buffer tersebut
5. Jika terdapat kemungkinan gerakan yang dapat dilakukan, yaitu ada token yang belum dikunjungi yang berada pada posisi vertikal/horizontal dari posisi saat ini dan buffer belum penuh, maka ambil token tersebut dan tambahkan ke buffer, kemudian kembali ke langkah 4
6. Jika sudah tidak terdapat kemungkinan gerakan, kembali ke posisi sebelumnya dan kembali ke langkah 4
7. Jika buffer kosong, pilih *Starting Token* yang lainnya hingga semua *Starting Token* telah dihitung nilainya

Tangkapan layar source code hasil implementasi pada bahasa C dengan pendekatan rekursif :

```
12
13  void findOptimumSequence(TokenMatrix M, Token* Buffer, Sequence* Seqs, Coordinate* visited, int SeqCount,
14    int bufferSize, int horizontal, int currentLength, Coordinate* CurrentOptimumCoords, int* CurrentMax, int* maxBufferSize){
15      if (SeqCount == 0) {return;}
16      int currentPoint = BufferPoint(Buffer, Seqs, SeqCount, currentLength);
17
18      if (currentPoint > *CurrentMax || (currentPoint >= *CurrentMax && currentLength < *maxBufferSize)){
19          copyCoord(visited, CurrentOptimumCoords);
20          *CurrentMax = currentPoint;
21          *maxBufferSize = currentLength;
22      }
23
24      if (currentLength == bufferSize){return;}
25
26      int i;
27      Coordinate currentCoord;
28      if (currentLength > 0){
29          currentCoord = visited[currentLength - 1];
30      }
31      else{
32          currentCoord.x = 0;
33          currentCoord.y = 1;
34      }
35
36      if (horizontal){
37          traversal(i, 1, M.width){
38              Coordinate nextCoord = {i, currentCoord.y};
39              if (NOT HasCoord(visited, nextCoord)){
40                  visited[currentLength] = nextCoord;
41                  visited[currentLength + 1] = NullCoordinate;
42                  Buffer[currentLength] = ACCESS(M, nextCoord.x, nextCoord.y);
43                  Buffer[currentLength + 1] = NullToken;
44                  findOptimumSequence(M, Buffer, Seqs, visited, SeqCount, bufferSize,
45                    NOT horizontal, currentLength + 1, CurrentOptimumCoords, CurrentMax, maxBufferSize);
46              }
47          }
48      }
49      else{
50          traversal(i, 1, M.height){
51              Coordinate nextCoord = {currentCoord.x, i};
52              if (NOT HasCoord(visited, nextCoord)){
53                  visited[currentLength] = nextCoord;
54                  visited[currentLength + 1] = NullCoordinate;
55                  Buffer[currentLength] = ACCESS(M, nextCoord.x, nextCoord.y);
56                  Buffer[currentLength + 1] = NullToken;
57                  findOptimumSequence(M, Buffer, Seqs, visited, SeqCount, bufferSize,
58                    NOT horizontal, currentLength + 1, CurrentOptimumCoords, CurrentMax, maxBufferSize);
59              }
60          }
61      }
62  }
```

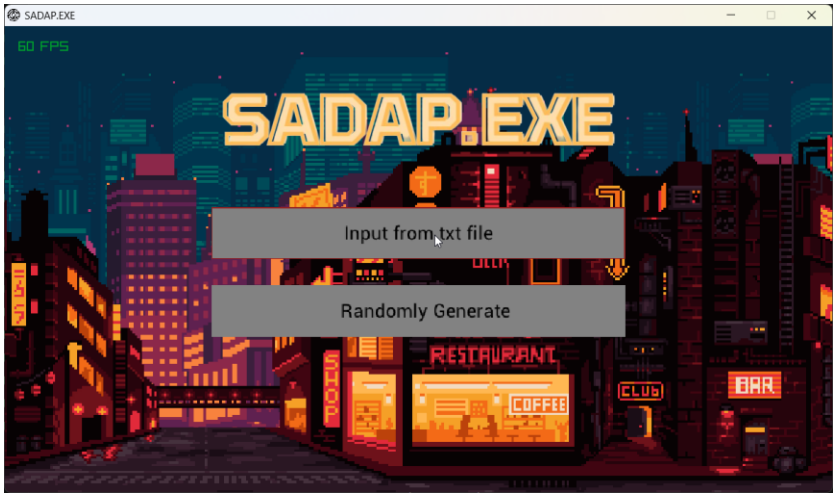
BAB 3
Pengujian Program

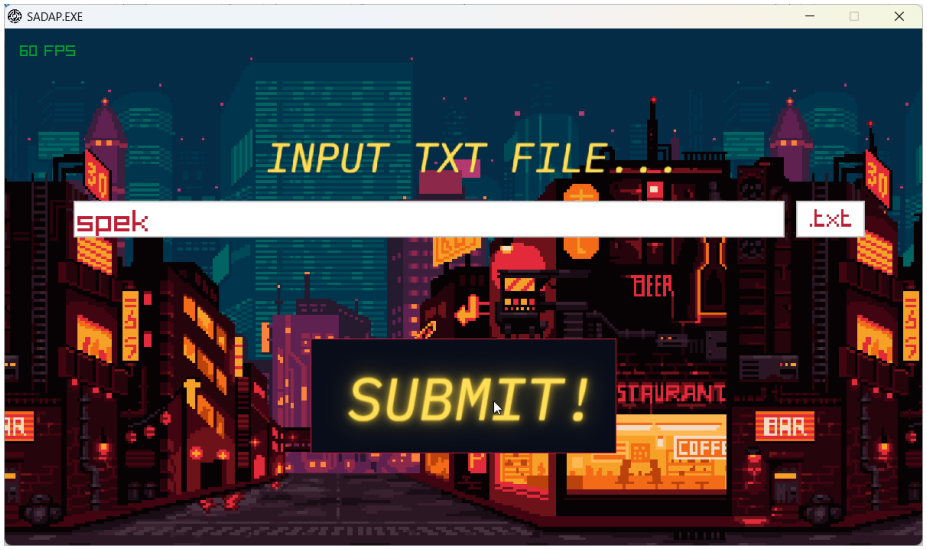
Data Uji 1 :

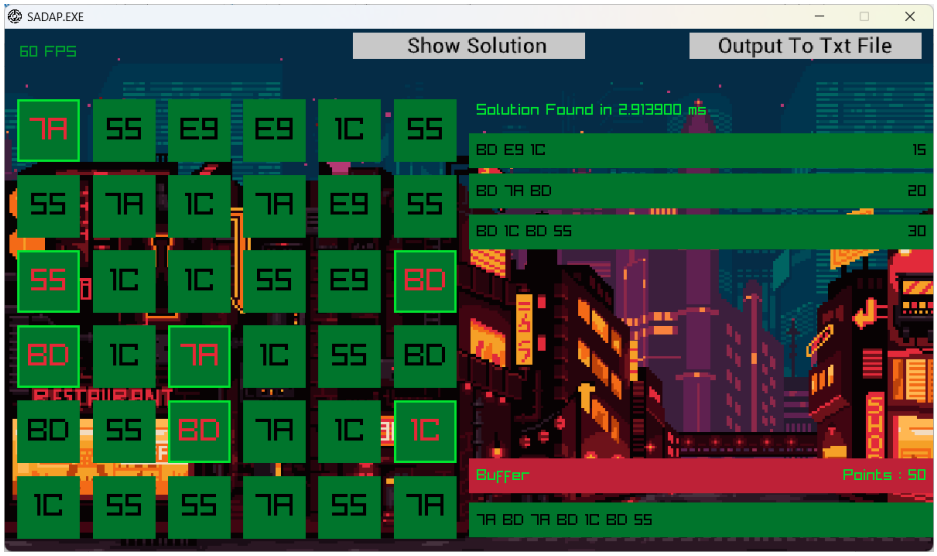
bin > tests > spek.txt

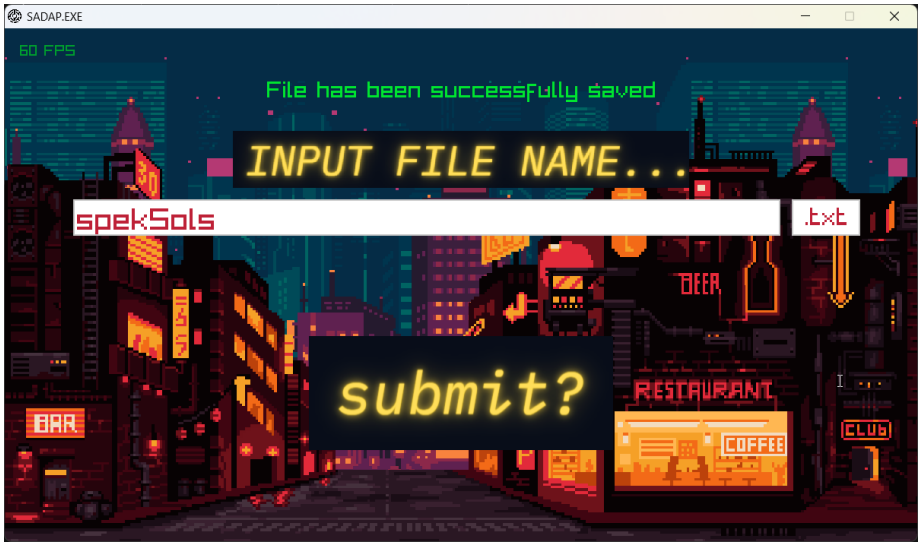
```
7
6 6
7A 55 E9 E9 1C 55
55 7A 1C 7A E9 55
55 1C 1C 55 E9 BD
BD 1C 7A 1C 55 BD
BD 55 BD 7A 1C 1C
1C 55 55 7A 55 7A
3
BD E9 1C
15
BD 7A BD
20
BD 1C BD 55
30
```

Untuk input data 1, disajikan tangkapan layar step by step input text file hingga ke output, untuk data berikutnya ditampilkan file input dan output saja. Karena output memiliki animasi yang tidak dapat disajikan dalam bentuk tangkapan layar, juga dilampirkan video singkat yang menunjukkan penggunaan program.


SADAP.EXE
60 FPS
SADAP.EXE
Input from txt file
Randomly Generate
RESTAURANT
COFFEE
CLUB
BAR


SADAP.EXE
60 FPS
INPUT TXT FILE...
spek
.txt
SUBMIT!
BEER
BAR


SADAP.EXE
60 FPS
Show Solution
Output To Txt File
Solution Found in 2.913900 ms
BD E9 1C 15
BD 7A BD 20
BD 1C BD 55 30
Buffer
Points : 50
7A BD 7A BD 1C BD 55



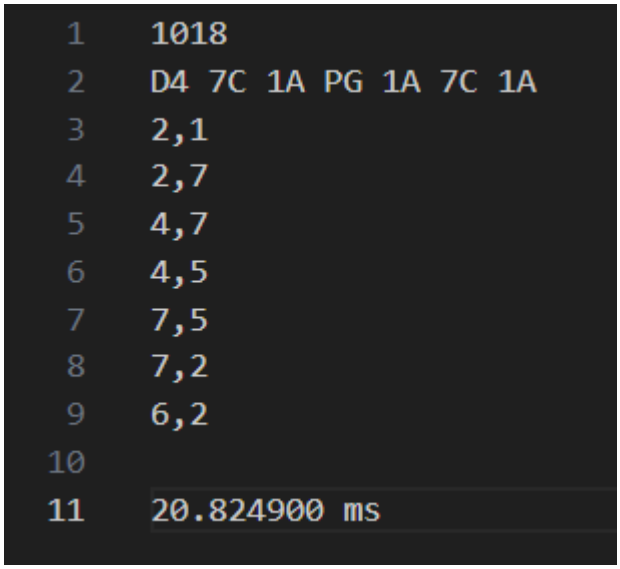

```
50
7A BD 7A BD 1C BD 55
1,1
1,4
3,4
3,5
6,5
6,3
1,3

2.913900 ms
```

Data uji 2 :

bin > tests > test2.txt

```
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
```

Hasil :



Data Uji 3 :

```
8
7 7
J7 N1 N1 N1 1A N1 J7
1A L9 J7 L9 G4 L9 G4
L9 J7 J7 N1 J7 1A N1
L9 G4 1A 1A L9 N1 1A
1A J7 J7 N1 J7 1A L9
N1 G4 N1 N1 1A J7 N1
1A N1 L9 G4 G4 G4 J7
7
G4 L9
-79
J7 G4 L9
74
L9 L9 J7 1A
336
N1 J7
165
J7 L9
33
J7 1A 1A N1
-195
G4 N1 N1
-158
```

Hasil :

```
534
J7 L9 L9 J7 1A N1 J7
1,1
1,4
5,4
5,3
6,3
6,1
7,1

122.048500 ms
```

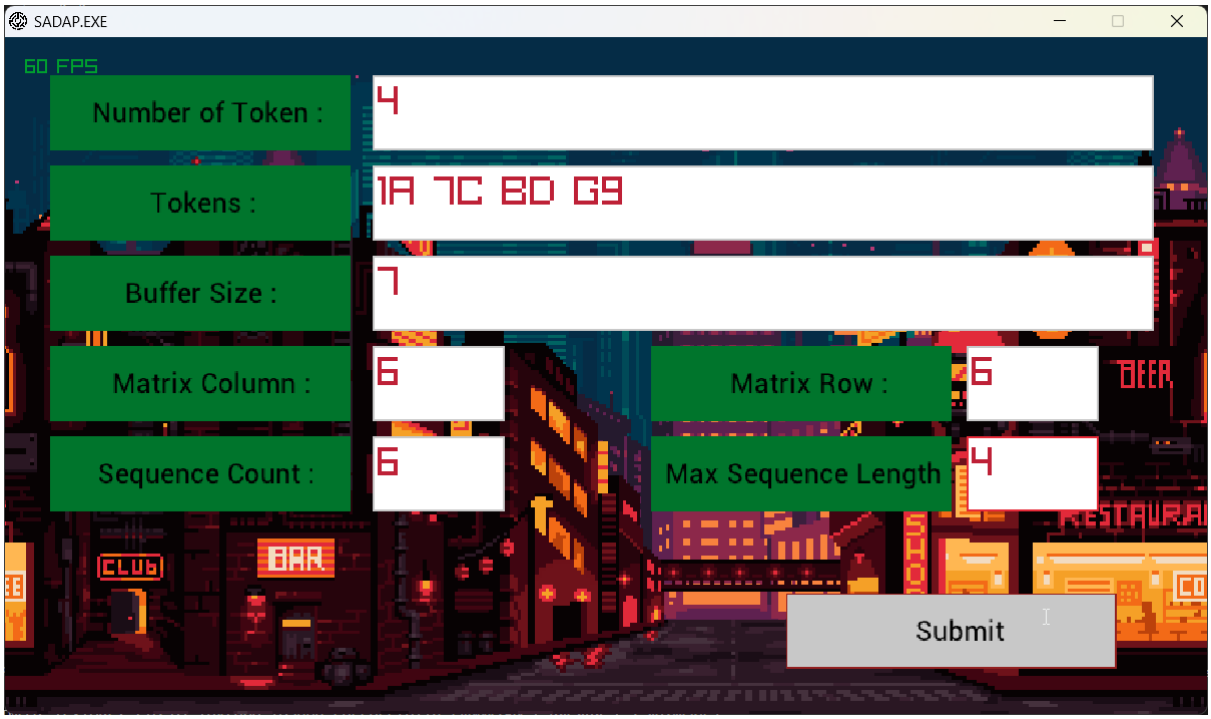
Data Uji 4 :

```
7
6 6
55 55 55 55 55 55
55 55 55 55 55 55
55 55 55 55 55 55
55 55 55 55 55 55
55 55 55 55 55 55
55 55 55 55 55 55
3
55 55 55 55
-100
55 55 55
15
55 55 55 55 55
30
```

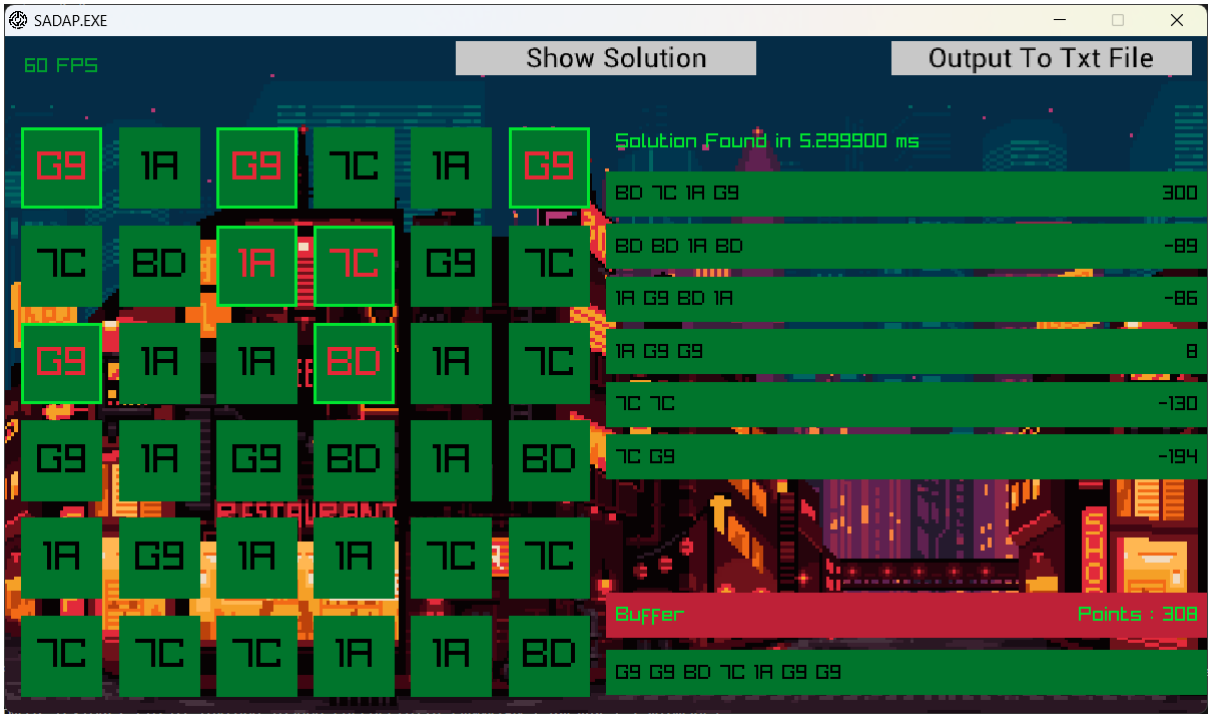
Hasil :

```
15
55 55 55
1,1
1,2
2,2

2.608200 ms
```

Data Uji 5 (Random) :



Hasil :




```
1   308
2   G9 G9 BD 7C 1A G9 G9
3   1,1
4   1,3
5   4,3
6   4,2
7   3,2
8   3,1
9   6,1
10
11  5.299900 ms
```

Data Uji 6 :

```
2
2 2
AB CD
CD AB
2
AB AB
100
CD CD
20
```


```
0

0.001100 ms
```

BAB 4
LAMPIRAN

Link repository (GitHub) : https://github.com/abdulrafirh/Tucil1_13522089.git

Link Video Singkat :
https://drive.google.com/file/d/1O48hL1ZaWP7Wye1OcrHOiLxCZ3wxKWcz/view?usp=drive_link