

**Laporan Tugas Kecil 1 IF2211 Strategi Algoritma
Semester 2 Tahun Akademik 2023/2024**



Disusun oleh :
Muhammad Roihan - 13522152

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

1. Deskripsi Tugas



Gambar 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.

2. Algoritma Brute Force

Untuk menyelesaikan persoalan ini digunakan algoritma brute force dengan langkah - langkah sebagai berikut :

1. Mencari semua kemungkinan *path* yang dapat diciptakan. Kemungkinan *path* yang dicari tidak hanya path yang memiliki panjang buffer maksimum tapi juga yang lebih kecil dari itu. Hal ini dilakukan karena bisa jadi terdapat *path* yang memiliki panjang yang lebih kecil namun memiliki total hadiah yang lebih besar sehingga dapat menghasilkan solusi yang paling optimal.
2. Penelusuran *path* dimulai dari *cell - cell* yang berada dibaris paling atas. *Path* akan ditelusuri dengan pola horizontal dan vertikal secara bergantian. Sebagai contoh, misalkan kita menelusuri sebuah path yang dimulai dari posisi (0,0), maka program akan mencari *cell - cell* secara vertikal yang belum pernah dikunjungi. Anggap cell yang dipilih selanjutnya adalah cell pada posisi (2,0). Setelah itu program kembali mencari *cell - cell* yang belum pernah dikunjungi namun secara horizontal. Hal ini dilakukan berulang ulang hingga path memiliki panjang buffer maksimum.
3. Selanjutnya, akan dicari sebuah *path* yang memiliki total hadiah yang paling banyak. Sebuah *path* ini didapatkan dengan mengiterasi semua kemungkinan *path* yang sudah dicari sebelumnya. Jika, terdapat dua *path* yang memiliki total hadiah yang sama maka akan diambil yang memiliki panjang *path* paling pendek.

3. Source Program

A. main.py

```
src > main.py > ...
1  from FileSystem import *
2  from Solver import *
3  import os
4
5  os.system("cls")
6  print("Pilih Cara Input")
7  print("1. Input File atau 2. Random")
8  choice = input()
9
10
11 if choice == "1":
12     data = input_file()
13     dataParser(data)
14     solveFile()
15     print("Ingin menyimpan hasil (Y/N)")
16     choice = input()
17     if choice == "Y":
18         saveFile()
19
20 if choice == "2":
21     inputRand()
22     printMatSeq()
23     solveFile()
24     print("Ingin menyimpan hasil (Y/N)")
25     choice = input()
26     if choice == "Y":
27         saveFileRandom()
28
```

B. Cell.py

```
src > Cell.py > ...
1  class Cell:
2      def __init__(self, row, column, visited, value):
3          self.row = row
4          self.column = column
5          self.visited = visited
6          self.value = value
7
```

C. FileSystem.py



```
src > FileSystem.py > ...
1  import os
2
3
4  def input_file():
5      global data
6      os.system("cls")
7      print("Masukkan nama file (Pastikan file telah berada di folder test) :")
8      fileName = input()
9      data = []
10     with open("../test/" + fileName) as my_file:
11         dataTmp = []
12         for line in my_file:
13             splitedLine = line.split(" ")
14             for arr in splitedLine:
15                 dataTmp.append(arr.rstrip())
16             data.append(dataTmp)
17             dataTmp = []
18     return data
19
```

D. Solver.py (Bagian algoritmanya saja)

```
src > Solver.py > dataParser
167 def resetArr(arr):
168     for i in range(len(arr)):
169         for j in range(len(arr[0])):
170             arr[i][j].visited = False
171
172
173 def solve(arr, sequences):
174     for i in range(len(arr)):
175         compareArr(arr[i], sequences)
176
177
178 def resetCekSeq(sequences):
179     global cekSeq
180     cekSeq = []
181     for i in range(len(sequences)):
182         cekSeq.append(False)
183
184
185 def compareArr(arr, sequences):
186     global maxVal, seqAns, rowAns, colAns, ans
187     resetCekSeq(sequences)
188     ans = 0
189     for i in range(len(sequences)):
190         for j in range(len(arr)):
191             if arr[j].value == sequences[i][0]:
192                 cek = True
193                 a = 0
194                 b = j
195                 while cek and b < len(arr) and a < len(sequences[i]):
196                     if arr[b].value != sequences[i][a]:
197                         cek = False
198                     a += 1
199                     b += 1
200                 if a == len(sequences[i]) and cek and (not cekSeq[i]):
201                     ans += seqVal[i]
202                     cekSeq[i] = True
203                 if ans > maxVal or (ans == maxVal and len(arr) < len(sequences[i])):
```

src > Solver.py > dataParser

```
204         seqAns = []
205         rowAns = []
206         colAns = []
207         maxVal = ans
208         for x in range(len(arr)):
209             seqAns.append(arr[x].value)
210             rowAns.append(arr[x].row)
211             colAns.append(arr[x].column)
212
213
214     def Visit(row, col, cnt, path, visitRow, buffMax, arr):
215         global Paths
216         cellNow = arr[row][col]
217         cellNow.visited = True
218         path.append(cellNow)
219         Paths.append(path.copy())
220
221         if cnt == buffMax - 1:
222             Paths.append(path.copy())
223             cellNow.visited = False
224             path.pop()
225             return
226         else:
227             limit = len(arr) if visitRow else len(arr[0])
228
229             for i in range(limit):
230                 if visitRow:
231                     if arr[i][col].visited:
232                         continue
233                     Visit(i, col, cnt + 1, path, False, buffMax, arr)
234                 else:
235                     if arr[row][i].visited:
236                         continue
237                     Visit(row, i, cnt + 1, path, True, buffMax, arr)
238
239             cellNow.visited = False
240             path.pop()
```

src >  Solver.py >  dataParser

```
242
243     def solveFile():
244         start = time.time()
245
246         for i in range(matWidth):
247             resetArr(arr)
248             Visit(0, i, 0, [], True, buffMax, arr)
249         solve(Paths, sequences)
250
251         print("Output : ")
252         print(maxVal)
253         for seq in seqAns:
254             print(seq, end=" ")
255         print("")
256         for j in range(len(rowAns)):
257             print(colAns[j], ",", rowAns[j])
258         end = time.time()
259         print("")
260         print("")
261         global runtime
262         runtime = (end - start) * 1000
263         print("{:.2f} ms".format(runtime))
264
```


4. Screenshot Hasil

A. Contoh 1

```
test > input.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

test > outpu.txt
1 50
2 7A BD 7A BD 1C BD 55
3 1,1
4 1,4
5 3,4
6 3,5
7 6,5
8 6,3
9 1,3
10
11 863.54 ms

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

863.54 ms
```

B. Contoh 2

```
test > input2.txt
1 3
2 3 3
3 1C BD BD
4 1C 7A 7A
5 7A 7A BD
6 3
7 1C 1C
8 100
9 BD 7A
10 5
11 7A 1C
12 20
13 .

test > out2.txt
1 100
2 1C 1C
3 1,1
4 1,2
5
6 0.00 ms

Output :
100
1C 1C
1 , 1
1 , 2

0.00 ms
```

C. Contoh 3

```
Pilih Cara Input
1. Input File atau 2. Random
2
Masukkan jumlah token unik : 3
Masukkan token (Pisahkan dengan spasi) : 1C BD 7A
Masukkan ukuran buffer : 4
Masukkan ukuran matrix
Ukuran baris : 6
Ukuran kolom : 7
Masukkan jumlah sekuens : 3
Masukkan ukuran maksimal sekuens : 4
```

```
test > out3.txt
1 95
2 BD BD 7A 7A
3 4,1
4 4,4
5 3,4
6 3,1
7
8 Random Matrix
9 1C 7A 7A BD BD 1C 7A
10 1C 7A BD 1C 7A BD 1C
11 BD BD 7A 7A 7A 1C 7A
12 1C 1C 7A BD 7A BD 1C
13 1C 1C BD BD BD 7A 7A
14 7A 1C BD BD BD 7A 7A
```

```
Random Matrix
1C 7A 7A BD BD 1C 7A
1C 7A BD 1C 7A BD 1C
BD BD 7A 7A 7A 1C 7A
1C 1C 7A BD 7A BD 1C
1C 1C BD BD BD 7A 7A
7A 1C BD BD BD 7A 7A
```

```
Random Sequences and Reward
BD BD 7A 7A
84
7A BD 7A
44
BD
11
```

```
Output :
95
BD BD 7A 7A
4 , 1
4 , 4
3 , 4
3 , 1
```

16.11 ms

D. Contoh 4

```
Masukkan nama file (Pastikan file telah berada di folder test) :
input4.txt
Output :
0
```

366.59 ms

```
test > input4.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 1
10 ZZ ZZ
11 20
```

```
test > out4.txt
1 0
2
3
4 366.59 ms
```

E. Contoh 5

```
Pilih Cara Input
1. Input File atau 2. Random
2
Masukkan jumlah token unik : 1
Masukkan token (Pisahkan dengan spasi) : 1C
Masukkan ukuran buffer : 4
Masukkan ukuran matrix
Ukuran baris : 6
Ukuran kolom : 6
Masukkan jumlah sekuens : 3
Masukkan ukuran maksimal sekuens : 3

Random Matrix
1C 1C 1C 1C 1C 1C
1C 1C 1C 1C 1C 1C
1C 1C 1C 1C 1C 1C
1C 1C 1C 1C 1C 1C
1C 1C 1C 1C 1C 1C
1C 1C 1C 1C 1C 1C

Random Sequences and Reward
1C
93
1C 1C 1C
43
1C 1C
86

Output :
222
1C 1C 1C
1 , 1
1 , 2
2 , 2

19.63 ms
```

F. Contoh 6

```
Pilih Cara Input
1. Input File atau 2. Random
2
Masukkan jumlah token unik : 6
Masukkan token (Pisahkan dengan spasi) : AN AM PB GB GA MD
Masukkan ukuran buffer : 2
Masukkan ukuran matrix
Ukuran baris : 3
Ukuran kolom : 3
Masukkan jumlah sekuens : 3
Masukkan ukuran maksimal sekuens : 2

Random Matrix
AM MD GB
PB AN GA
PB AN AN

Random Sequences and Reward
AM
34
AN
4
MD
10

Output :
34
AM
1 , 1

0.00 ms
Ingin menyimpan hasil (Y/N)
N
```

GITHUB

https://github.com/mroi hn/Tucil1_13522152

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		✓