

Tugas Kecil IF2211 Strategi Algoritma
Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force
Semester II Tahun 2023/2024



Disusun Oleh:

Muhammad Zaki – 13522136

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA INSTITUT TEKNOLOGI BANDUNG
TAHUN AJARAN 2023/2024

Bab 1

Deskripsi Masalah

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
1. 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

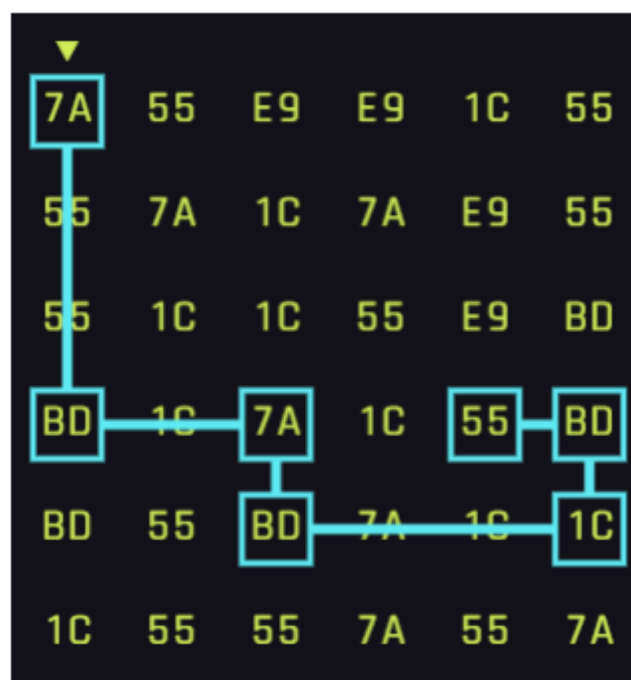
7A	55	E9	E9	1C	55
55	7A	1C	7A	E9	55
55	1C	1C	55	E9	BD
BD	1C	7A	1C	55	BD
BD	55	BD	7A	1C	1C
1C	55	55	7A	55	7A

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



Gambar 1 : Contoh Solusi

BAB 2

Implementasi dan Pengujian

2.1 Penjelasan Algoritma yang digunakan

Disini saya menggunakan algoritma brute force yang juga merupakan recursive

```
def search(row, col, currentSequence, direction, allSequences):
    if row < 0 or row >= matrixSizerow or col < 0 or col >= matrixSizecol:
        return allSequences

    currentSequence.append((board[row][col], row, col))

    if len(currentSequence) == int(bufferSize):
        #print("Found sequence:", currentSequence)
        allSequences.append(list(currentSequence))
        currentSequence.pop()
        return allSequences

    if direction == "vertical":
        for new_row in range(row + 1, matrixSizerow):
            allSequences = search(new_row, col, currentSequence, "horizontal", allSequences)
        for new_row in range(row - 1, -1, -1):
            allSequences = search(new_row, col, currentSequence, "horizontal", allSequences)
    elif direction == "horizontal":
        for new_col in range(col + 1, matrixSizecol):
            allSequences = search(row, new_col, currentSequence, "vertical", allSequences)
        for new_col in range(col - 1, -1, -1):
            allSequences = search(row, new_col, currentSequence, "vertical", allSequences)

    currentSequence.pop()
    return allSequences
```

Gambar 2: Algoritma search

Pada algoritma ini saya menandai arah yang telah digunakan, karena aturan pada *breach protocol* ini adalah sehabis vertical dilanjutkan dengan horizontal begitu pun sebaliknya, jadi setelah dilakukan pencarian secara vertikal maka parameter selanjutnya adalah melakukan pencarian secara horizontal. Disini saya menggunakan recursive dengan basis yang pertama adalah jika sudah melebihi atau kurang dari baris ataupun kolom, yang kedua jika sequence sudah mencapai buffer size maka akan dihe ntikan, selain itu akan tetap dilakukan pencarian.

Dalam pencarian untuk mencapai semua kemungkinan saya menggunakan parameter untuk mengaturnya. Untuk bisa naik ataupun turun secara vertikal maka digunakan paramater baris +1 untuk turun dan juga -1 untuk naik.

Untuk mengatur agar bisa pindah ke arah kanan kiri saya menggunakan parameter kolom + 1 dan juga kolom -1.

Jika sudah mencapai basisnya maka akan me return allSequence nyaa dan dikumpulkan untuk nanti dicari manakah sequence yang mempunyai nilai bonus yang paling besar.

2.2 Source Code

```
import random,time

class Sequence:
    def __init__(self, sequenceData=None, bonus=0):
        self.sequenceData = sequenceData if sequenceData is not None else []
        self.bonus = bonus

    def __eq__(self, other):
        return self.sequenceData == other.sequenceData

def search(row, col, currentSequence, direction, allSequences):
    if row < 0 or row >= matrixSizerow or col < 0 or col >= matrixSizecol:
        return allSequences

    currentSequence.append((board[row][col], row, col))

    if len(currentSequence) == int(bufferSize):
        #print("Found sequence:", currentSequence)
        allSequences.append(list(currentSequence))
        currentSequence.pop()
        return allSequences

    if direction == "vertical":
        for new_row in range(row + 1, matrixSizerow):
            allSequences = search(new_row, col, currentSequence, "horizontal",
allSequences)
        for new_row in range(row - 1, -1, -1):
            allSequences = search(new_row, col, currentSequence, "horizontal",
allSequences)
    elif direction == "horizontal":
        for new_col in range(col + 1, matrixSizecol):
            allSequences = search(row, new_col, currentSequence, "vertical",
allSequences)
        for new_col in range(col - 1, -1, -1):
```

```

        allSequences = search(row, new_col, currentSequence, "vertical",
allSequences)

    currentSequence.pop()
    return allSequences

def printMatrix():
    for i in range(matrixSizerow):
        for j in range(matrixSizecol):
            print(board[i][j], end=" ")
        print()

def createSequence(totalSequence, maxSequence, token):
    sequences = []
    for _ in range(int(totalSequence)):
        sequenceLength = random.randint(2, int(maxSequence))
        sequenceData = [random.choice(token) for _ in range(sequenceLength)]
        bonus = random.randint(1, 10) * 5
        sequences.append(Sequence(sequenceData, bonus))
    return sequences

def printSequence(sequences):
    for i in range(int(totalSequence)):
        print("Sequence " + str(i + 1) + ": ", end="")
        for j in range(len(sequences[i].sequenceData)):
            print(sequences[i].sequenceData[j], end=" ")
        print("Bonus: " + str(sequences[i].bonus))

def traverseAndFindSequences():
    allSequences = []
    for col in range(matrixSizecol):
        search(0, col, [], "vertical", allSequences)
    return allSequences

def compareSequences(allSequences, sequences):
    bestSequence = None
    maxTotalBonus = 0

```

```

    for foundSeq in allSequences:
        totalBonus = 0
        foundSeqData = [item[0] for item in foundSeq]

        for sequence in sequences:
            if sequence.sequenceData ==
foundSeqData[:len(sequence.sequenceData)]:
                totalBonus += sequence.bonus

            if totalBonus > maxTotalBonus:
                maxTotalBonus = totalBonus
                bestSequence = foundSeq

    return bestSequence, maxTotalBonus

while True:
    #Cek input CLI or txt
    print("Masukkan jenis input :")
    print("1. CLI")
    print("2. TXT")
    inputType = input("Pilih jenis input : ")

    if int(inputType) == 1:
        totalUniqueToken = input("Masukkan jumlah token unik: ")
        token = []
        token = input("Masukkan token: ").split()

        if len(token) != int(totalUniqueToken):
            print("Jumlah token tidak sesuai")
            exit()

        while True:
            try:
                bufferSize = int(input("Masukkan ukuran buffer: "))
                if bufferSize > 0:
                    break
            else:
                print("Pilih angka lebih dari 0 !!!.")

```

```

        except ValueError:
            print("INPUT HANYA INTEGER !.")

    matrixSize = input("Masukkan ukuran matriks : ").split()
    matrixSizerow = int(matrixSize[0])
    matrixSizecol = int(matrixSize[1])
    totalSequence = input("Masukkan jumlah sequence: ")
    maxSequence = input("Masukkan jumlah maksimum sequence: ")

    sequences = createSequence(totalSequence, maxSequence, token)

    print("Sequence: ")
    printSequence(sequences)
    print()

    board = [[0 for i in range(matrixSizecol)] for j in range(matrixSizerow)]

    for i in range(matrixSizerow):
        for j in range(matrixSizecol):
            board[i][j] = random.choice(token)

elif int(inputType) == 2:
    fileName = input("Masukkan nama file : ")
    filePath = "../test/" + fileName + ".txt"
    with open(filePath, "r") as file:
        lines = file.readlines()
        bufferSize = int(lines[0].split()[0])

        if bufferSize <= 0:
            print("Buffer size harus lebih dari 0")
            exit()

        matrixSize = lines[1].split()
        matrixSizerow = int(matrixSize[0])
        matrixSizecol = int(matrixSize[1])

        board = [[0 for i in range(matrixSizecol)] for j in
range(matrixSizerow)]

```



```

        for i in range(matrixSizerow):
            val = lines[i + 2].split()
            for j in range(matrixSizecol):
                board[i][j] = (val[j])

        totalSequence = int(lines[matrixSizerow + 2])
        if totalSequence <= 0:
            print("Jumlah sequence harus lebih dari 0")
            exit()

        sequences = []
        for i in range(matrixSizerow + 3, matrixSizerow + 2 +
(int(totalSequence) * 2), 2):
            sequenceData = lines[i].split()
            bonus = int(lines[i+1].strip())
            sequences.append(Sequence(sequenceData, bonus))
        printSequence(sequences)
    else:
        print("Input tidak valid")

    print("Matriks: ")
    printMatrix()
    print()

    startTime = time.time()
    allSequences = traverseAndFindSequences()
    endTime = time.time()

    #DEBUG
    # print("All Sequences:")
    # print(allSequences)

    bestSequence, totalBonus = compareSequences(allSequences, sequences)
    elapsedTimeMs = (endTime - startTime) * 1000

    print(totalBonus)
    if (totalBonus > 0):

```

```

        for item in bestSequence:
            print(" ".join(map(str, item[0:1])), end=" ")
        print()

        for item in bestSequence:
            print(", ".join(map(str, item[1:3])))

        print("\n" + str(elapsedTimeMs) + " ms \n")

choice = input("Apakah ingin menyimpan solusi? (y/n) \n")

if choice == "y" or choice == "Y":
    outputFileName = input("Masukkan nama file : ")
    filePath = "../test/" + outputFileName + ".txt"
    with open(filePath, "w") as file:
        file.write(str(totalBonus) + "\n")
        if (totalBonus != 0):
            for item in bestSequence:
                file.write(" ".join(map(str, item[0:1])) + " ")
            file.write("\n")
            for item in bestSequence:
                file.write(", ".join(map(str, item[1:3]))) + "\n"
            file.write("\n" + str(elapsedTimeMs) + " ms")
        print("Solusi berhasil disimpan di " + filePath)

print("Continue Playing ? (y/n)")
exitChoice = input()
if exitChoice == "y" or exitChoice == "Y":
    continue
else:
    break

```

2.3 Test Case

2.3.1 Test Case 1 (Input CLI)

```
Pilih jenis input : 1
Masukkan jumlah token unik: 3
Masukkan token: AA BB CC
Masukkan ukuran buffer: 3
Masukkan ukuran matriks : 3 4
Masukkan jumlah sequence: 3
Masukkan jumlah maksimum sequence: 2
Sequence:
Sequence 1: CC AA Bonus: 35
Sequence 2: BB AA Bonus: 30
Sequence 3: BB CC Bonus: 10

Matriks:
BB AA CC AA
AA AA BB CC
CC BB AA BB

35
CC AA BB
0, 2
2, 2
2, 3

0.0 ms


Apakah ingin menyimpan solusi? (y/n)
y
Masukkan nama file : solusi1
Solusi berhasil disimpan di ../test/solusi1.txt
Press anything to exit...
```

Gambar 3 : TC 1

2.3.2 Test Case 2 (Input File)

```
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
```

Gambar 4 : Input via File

```
test >  solusi2.txt  
1 0  
2  
3 105.1492691040039 ms
```

Gambar 5 : Hasil TC 2

2.3.3 Test Case 3 (Input CLI)

```
Masukkan jumlah token unik: 5
Masukkan token: BD 1C 7A 55 E9
Masukkan ukuran buffer: 7
Masukkan ukuran matriks : 6 6
Masukkan jumlah sequence: 3
Masukkan jumlah maksimum sequence: 4
Sequence:
Sequence 1: BD BD BD Bonus: 35
Sequence 2: 7A E9 Bonus: 30
Sequence 3: 7A 7A Bonus: 20


Matriks:
E9 1C BD BD BD 7A
BD E9 BD E9 1C 1C
BD 7A E9 7A 1C 55
BD 1C 1C E9 55 7A
55 7A 1C BD 7A E9
BD 7A 55 55 BD 55

35
BD BD BD BD 7A 1C 1C
0, 2
1, 2
1, 0
2, 0
2, 1
3, 1
3, 2

102.45561599731445 ms
```

Gambar 6 : Hasil TC 3

2.3.4 Test Case 4(Input File)

```
test >  soal4.txt
1 5
2 5 5
3 7A 55 E9 1C BD
4 55 7A 1C BD E9
5 1C E9 55 7A BD
6 BD 1C 7A 55 E9
7 E9 BD 55 1C 7A
8 2
9 BD E9 1C
10 15
11 BD 55 BD
12 20
13
```

Gambar 7 : Input TC 4

```
15
BD E9 1C 55 7A
0, 4
1, 4
1, 2
2, 2
2, 3
1.1000633239746094 ms
```

Gambar 8: Hasil TC 4

2.3.5 Test Case 5 (Input File)

```
soal5.txt
7
7 7
7A 55 E9 1C BD 55 1C
55 7A 1C BD E9 BD 7A
1C E9 55 7A BD 1C 55
BD 1C 7A 55 E9 7A E9
E9 BD 55 1C 7A 55 1C
1C 7A BD E9 55 E9 BD
55 1C E9 7A 1C BD 7A
3
BD E9 1C
-15
BD 7A BD
20
BD 1C BD 55
30
```

Gambar 9: Input TC 5

```

30
BD 1C BD 55 1C BD E9
0, 4
6, 4
6, 5
4, 5
4, 6
5, 6
5, 5

372.96438217163086 ms

```

Gambar 10: Hasil TC 5

2.3.6 Test Case 6 (Input CLI)

```

Pilih jenis input : 1
Masukkan jumlah token unik: 3
Masukkan token: AA BB 55
Masukkan ukuran buffer: 4
Masukkan ukuran matriks : 8 8
Masukkan jumlah sequence: 3
Masukkan jumlah maksimum sequence: 3
Sequence:
Sequence 1: 55 55 AA Bonus: 5
Sequence 2: 55 AA Bonus: 30
Sequence 3: AA AA Bonus: 10

Matriks:
AA 55 55 55 55 AA 55 BB
55 BB 55 AA BB BB 55 AA
AA AA AA AA AA AA 55 55
BB 55 55 AA BB BB AA 55
AA AA AA 55 55 AA BB BB
BB BB BB 55 BB AA BB 55
BB 55 BB 55 AA BB AA AA
BB 55 BB 55 BB AA AA AA

30
55 AA AA 55
0, 1
2, 1
2, 2
3, 2

2.1026134490966797 ms

```

Gambar 11: TC 6

Lampiran

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		✓

Repository Github

https://github.com/mzaki9/Tucil1_13522136