Tugas Kecil
IF2211 Strategi Algoritma
Penyelesaian Cyberpunk 2077 Breach Protocol Dengan Algoritma Brute Force

Oleh :

Muhammad Rasheed Qais Tandjung 13522158

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2024

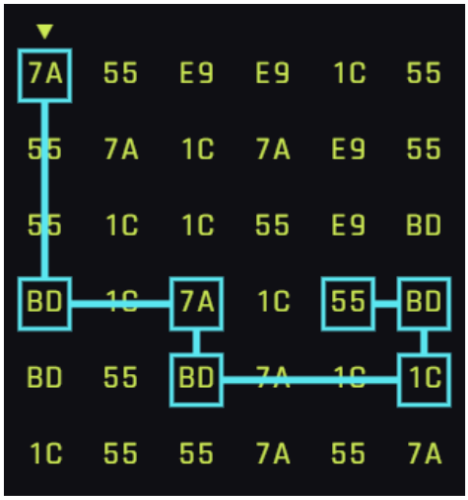
Bab I
DESKRIPSI TUGAS

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.



Gambar 2 Contoh Solusi
(Sumber: https://cyberpunk-hacker.com/)

Bab II
PENJELASAN ALGORITMA MENGGUNAKAN METODE *BRUTE FORCE*

Algoritma yang digunakan untuk menyelesaikan permasalahan ini adalah algoritma *brute-force* bertipe *exhaustive search* yaitu algoritmanya akan mengecek semua kemungkinan rute yang dapat dibuat, dan mengambil rute yang menghasilkan poin terbesar. Alur dari algoritma seperti berikut:

1. Lakukan iterasi *i traversal [0 .. COL]* untuk memilih kolom memulai. Algoritma akan selalu memulai pencarian dari kotak paling atas di kolom tersebut.
2. Untuk setiap iterasi, akan didapatkan koordinat x dan y, yaitu koordinat kotak pertama. Koordinat tersebut akan diberikan ke fungsi *cari(x, y, isVertical),* dengan parameter *isVertical = True*, yang menandakan bahwa algoritma akan selanjutnya melakukan pencarian secara vertikal.
3. Di dalam fungsi *cari*, program akan melakukan iterasi satu baris atau satu kolom penuh, tergantung parameter *isVertical*. Fungsi *cari* merupakan fungsi rekursif yang akan memanggil dirinya sendiri pada setiap iterasi baris/kolom, sampai buffer yang menyimpan rutenya sudah penuh. Gambaran besar dari fungsi *cari* dalam bentuk *pseudocode* sebagai berikut:

```
function cari(x, y, isVertical)
    if bufferPenuh:
        {hitung jumlah poin}
    else if isVertical:
        i traversal [0..ROW]
            cari(x, i, False)
    else: {isVertical = False}
        j traversal [0..COL]
            cari(j, y, True)
```

4. Jika buffer sudah penuh pada fungsi *cari*, maka program akan menghitung jumlah poin dari jalan yang sudah dilalui, dan membandingkannya dengan jumlah poin maksimal yang sudah tersimpan sebelumnya (jika baru memulai, maka nilai tersebut adalah 0).
5. Algoritma selesai setelah semua rute sudah dicek, dan di akhir akan mengembalikan rute dengan poin terbesar.

Bab III
SOURCE CODE PROGRAM DENGAN BAHASA *PYTHON*

```python
from time import time, sleep
from random import randint
from termcolor import colored
from datetime import datetime
import os.path
import sys


def slowprint(str):
    for char in str:
        sys.stdout.write(char)
        sys.stdout.flush()
        sleep(0.01)
    sleep(0.5)
```

```python
class Game:
    def __init__(self):

        self.maxPoints = 0
        self.maxSequence = ''
        self.path = []

    def formatSequence(self, seq):
        seqStr = ''
        for i in range(len(seq)):
            if not i % 2 and i:
                seqStr += ' '
            seqStr += seq[i]
        return seqStr
```

```python
    def traverse(self, x, y, path, visited, isVertical):
        if len(path) / 2 == self.MAX_BUFFER:
            points = 0
            for seq in self.seq:
                if seq in path:
                    points += self.seq[seq]

            if points > self.maxPoints:
                self.maxPoints = points
                self.maxSequence = path
                self.originalPath = visited
                self.path = [(x + 1, y + 1) for (y, x) in visited]
        elif isVertical:
            for i in range(self.ROW):
                if (i, x) not in visited:
                    newVisited = [x for x in visited] + [(i, x)]
                    self.traverse(x, i, path + self.grid[i][x], newVisited, False)
        else:
            for j in range(self.COL):
                if (y, j) not in visited:
                    newVisited = [x for x in visited] + [(y, j)]
                    self.traverse(j, y, path + self.grid[y][j], newVisited, True)

```

```
def findMax(self):
    start = time()
    for j in range(self.COL):
        self.traverse(j, 0, self.grid[0][j], [(0, j)], True)

    totalTime = round(time() - start, 3) * 1000
    # Print results
    slowprint(colored('\nSolve finished!\n\n', 'green'))

    if self.maxPoints == 0:
        slowprint(colored('\nMaximum Points     : ', 'yellow') + str(self.maxPoints))
        slowprint('\nNo path with points greater than 0 found.\n')

        slowprint(colored('\n\nTime taken: ', 'green') + colored(str(totalTime) + ' ms\n\n', 'yellow'))
```

```
else:
    for y in range(self.ROW):
        for x in range(self.COL):
            node = self.grid[y][x]

            if (y, x) in self.originalPath:
                print(colored(node + ' ', 'yellow'), end='')
            else:
                print(node + ' ', end='')

            sys.stdout.flush()
            sleep(0.001)
        sleep(0.01)
        print()

    slowprint(colored('\nMaximum Points     : ', 'yellow') + str(self.maxPoints))
    slowprint(colored('\nPath               : ', 'yellow') + str(self.formatSequence(self.maxSequence)))
    slowprint(colored('\nPath Coordinates   : ', 'yellow') + str(self.path))

    slowprint(colored('\n\nTime taken: ', 'green') + colored(str(totalTime) + ' ms\n\n', 'yellow'))
```

```
        # Save results
        choice = False
        while not choice:
            slowprint('Do you want to save the results to a .txt file? (Y/N): ')
            save = input()

            if save == 'Y':
                choice = True

                filename = 'test/' + self.filename
                slowprint(colored('\nSaving to ', 'green') + colored(filename, 'yellow') + colored('...', 'green'))
                sleep(1)

                with open(filename, 'w') as file:
                    file.write(str(self.maxPoints) + '\n')
                    file.write(self.formatSequence(self.maxSequence) + '\n')
                    for node in self.path:
                        file.write(str(node) + '\n')
                    file.write('\n' + f'{totalTime} ms')

                slowprint(colored('\nFile has been successfully saved!\n', 'green'))
            elif save == 'N':
                choice = True
            else:
                slowprint('Choice not recognized!\n')

    slowprint(colored('\nThanks for playing!\n', 'yellow'))
```

```
def readFile(self):
    slowprint('\nPlease input a ' + colored('file name (including the .txt extension) ', 'yellow') + 'for input.')
    slowprint('\nPlease make sure the file exists and is ' + colored('located in the "input" folder', 'yellow') +  ' as a .txt f

    found = False
    while not found:
        slowprint('File name: ')
        name = input()
        filename = 'input/' + name

        if os.path.isfile(filename):
            slowprint(colored('\nReading from file...', 'green'))
            found = True
        else:
            slowprint('\nFile not found! Please make sure ' + colored('you have inputted the correct file name ', 'yellow') + 'a
```

```
self.filename = name
with open(filename) as file:
    # Buffer length
    bufferLength = int(file.readline())

    # Matrix size
    matrixSize = [int(x) for x in file.readline().split()]
    matCol = matrixSize[0]
    matRow = matrixSize[1]

    # Matrix
    matrix = []
    for i in range(matRow):
        matrix.append(file.readline().split())

    # Sequences
    totalSeq = int(file.readline())
    seq = {}
    for i in range(totalSeq):
        sequence = file.readline().replace(' ', '').replace('\n', '')
        seq[sequence] = int(file.readline())
```

```
# Set variables
self.MAX_BUFFER = bufferLength
self.ROW = matRow
self.COL = matCol

self.grid = matrix
self.seq = seq

# Solve
self.findMax()
```

```
def randomize(self):
    slowprint('\nPlease provide the following parameters to aide the randomization process:\n')

    # Token amount
    tokenAmountValid = False

    while not tokenAmountValid: ···

    # Tokens
    tokensValid = False

    while not tokensValid: ···

    # Buffer size
    bufferSizeValid = False

    while not bufferSizeValid: ···

    # Matrix size
    matrixSizeValid = False

    while not matrixSizeValid: ···

    # Sequence amount
    sequenceAmountValid = False

    while not sequenceAmountValid: ···
```

```
# Generate matrix
matrix = [[tokens[randint(0, tokenAmount - 1)] for j in range(cols)] for i in range(rows)]

# Generate sequences
seqs = {}
for i in range(sequenceAmount): ···

timestamp = str(datetime.now().time())[:8].replace(':', '_')
self.filename = 'random_' + timestamp

# Set variables
self.grid = matrix
self.seq = seqs

self.ROW = rows
self.COL = cols
self.MAX_BUFFER = bufferSize
```

```
# Show matrix and sequences
slowprint(colored('\nGeneration finished!', 'green'))
slowprint(colored('\nMatrix:\n', 'yellow'))
for y in range(self.ROW):
    for x in range(self.COL):
        node = self.grid[y][x]
        print(node + ' ', end='')

        sys.stdout.flush()
        sleep(0.001)
    sleep(0.01)
    print()

slowprint(colored('\nSequences:', 'yellow'))
for seq in self.seq:
    slowprint('\n' + str(self.formatSequence(seq)) + ': ' + str(self.seq[seq]))

sleep(1)

# Generate solve
slowprint(colored('\n\nGenerating solve...', 'green'))
self.findMax()
```

```
def interface(self):
    slowprint('Welcome to ' + colored('Cyberpunk 2077 Breach Protocol!', 'green') + '\n')
    slowprint('Please choose your input method:\n\n')
    slowprint(colored('1. ', 'yellow') + 'Input from file\n')
    slowprint(colored('2. ', 'yellow') + 'Randomly generate puzzle\n')


    choiceValid = False
    while not choiceValid:
        slowprint('\nYour choice: ')
        choice = input()

        if choice == '1':
            choiceValid = True
            self.readFile()
        elif choice == '2':
            choiceValid = True
            self.randomize()
        else:
            slowprint('\nChoice not recognized!')
```

BAB IV
TANGKAPAN LAYAR PROGRAM

1. **test1.txt**

Input:	Output:
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	50 7A BD 7A BD 1C BD 55 (1, 1) (1, 4) (3, 4) (3, 5) (6, 5) (6, 3) (1, 3) 81.0 ms

```
$ python ./src/hack.py
Welcome to Cyberpunk 2077 Breach Protocol!
Please choose your input method:

1. Input from file
2. Randomly generate puzzle

Your choice: 1

Please input a file name (including the .txt extension) for input.
Please make sure the file exists and is located in the "input" folder as a .txt file.

File name: test1.txt

Reading from file...
```

```
Solve finished!

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

Maximum Points     : 50
Path               : 7A BD 7A BD 1C BD 55
Path Coordinates   : [(1, 1), (1, 4), (3, 4), (3, 5), (6, 5), (6, 3), (1, 3)]

Time taken: 81.0 ms

Do you want to save the results to a .txt file? (Y/N): Y

Saving to test/test1.txt...
File has been successfully saved!

Thanks for playing!
```

2. **test2.txt**

Input:	Output:
6 6 6 BD 1C BD 1C BD 1C 55 7A 55 7A 55 7A E9 FF E9 FF E9 FF BD 1C BD 1C BD 1C 55 7A 55 7A 55 7A	30 BD BD 1C FF E9 55 (1, 1) (1, 4) (2, 4) (2, 3) (1, 3)

E9 FF E9 FF E9 FF 3 1C FF 10 1C FF E9 20 7A 1C FF E9 30	(1, 2) 20.0 ms

```
$ python ./src/hack.py
Welcome to Cyberpunk 2077 Breach Protocol!
Please choose your input method:

1. Input from file
2. Randomly generate puzzle

Your choice: 1

Please input a file name (including the .txt extension) for input.
Please make sure the file exists and is located in the "input" folder as a .txt file.

File name: test2.txt

Reading from file...
```

```
Solve finished!

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

Maximum Points     : 30
Path               : BD BD 1C FF E9 55
Path Coordinates   : [(1, 1), (1, 4), (2, 4), (2, 3), (1, 3), (1, 2)]

Time taken: 20.0 ms

Do you want to save the results to a .txt file? (Y/N): Y

Saving to test/test2.txt...
File has been successfully saved!

Thanks for playing!
```

3. **test3.txt**

7 10 8 BD 1C 55 55 F3 E9 55 F3 E9 8G 55 E9 1C 7A 8G 8G 55 E9 55 55 F3 55 F3 8G E9 1C 8G E9 F3 1C E9 7A 1C F3 55 1C E9 7A 8G F3 1C 55 F3 7A 8G 8G F3 8G 55 8G 1C 8G BD E9 E9 BD 8G 7A 55 E9 1C 55 F3 E9 7A 1C BD 7A 7A F3 F3 1C 55 BD 55 1C 8G 55 1C 1C 4 F3 E9 E9 32 8G BD E9 34 BD BD F3 E9 24 E9 8G F3 20	

```
$ python ./src/hack.py
Welcome to Cyberpunk 2077 Breach Protocol!
Please choose your input method:

1. Input from file
2. Randomly generate puzzle

Your choice: test3.txt

Choice not recognized!
Your choice: 1

Please input a file name (including the .txt extension) for input.
Please make sure the file exists and is located in the "input" folder as a .txt file.

File name: test3.txt
```

```
Solve finished!

BD 1C 55 55 F3 E9 55 F3 E9 8G
55 E9 1C 7A 8G 8G 55 E9 55 55
F3 55 F3 8G E9 1C 8G E9 F3 1C
E9 7A 1C F3 55 1C E9 7A 8G F3
1C 55 F3 7A 8G 8G F3 8G 55 8G
1C 8G BD E9 E9 BD 8G 7A 55 E9
1C 55 F3 E9 7A 1C BD 7A 7A F3
F3 1C 55 BD 55 1C 8G 55 1C 1C

Maximum Points     : 66
Path               : BD F3 E9 E9 8G BD E9
Path Coordinates   : [(1, 1), (1, 3), (5, 3), (5, 6), (7, 6), (7, 7), (4, 7)]

Time taken: 2195.0 ms

Do you want to save the results to a .txt file? (Y/N): Y

Saving to test/test3.txt...
File has been successfully saved!
```

4. **random_12_25_38**

```
$ python ./src/hack.py
Welcome to Cyberpunk 2077 Breach Protocol!
Please choose your input method:

1. Input from file
2. Randomly generate puzzle

Your choice: 2

Please provide the following parameters to aide the randomization process:

Token amount: 5

Tokens (seperated by spaces ; each token consist of two characters): AA BB CC DD EE

Buffer size: 5

Matrix size (cols rows): 6 6

Sequence amount: 4

Maximum sequence length (>= 2): 5
```

```
Generation finished!
Matrix:
CC BB AA EE AA DD
EE DD DD EE CC DD
BB CC BB AA EE AA
BB DD CC CC AA AA
CC DD DD BB CC CC
BB CC CC AA EE BB

Sequences:
CC DD AA: 29
AA CC: 7
BB CC: 31
CC AA: 49
```

```
Solve finished!

CC BB AA EE AA DD
EE DD DD EE CC DD
BB CC BB AA EE AA
BB DD CC CC AA AA
CC DD DD BB CC CC
BB CC CC AA EE BB

Maximum Points      : 87
Path                : CC BB CC AA CC
Path Coordinates    : [(1, 1), (1, 4), (4, 4), (4, 3), (2, 3)]

Time taken: 9.0 ms

Do you want to save the results to a .txt file? (Y/N): Y

Saving to test/random_12_25_38...
File has been successfully saved!
```

5. **random_12_27_50**

```
$ python ./src/hack.py
Welcome to Cyberpunk 2077 Breach Protocol!
Please choose your input method:

1. Input from file
2. Randomly generate puzzle

Your choice: 2

Please provide the following parameters to aide the randomization process:

Token amount: 5

Tokens (seperated by spaces ; each token consist of two characters): VV WW XX YY ZZ

Buffer size: 7

Matrix size (cols rows): 7 7

Sequence amount: 4

Maximum sequence length (>= 2): 4
```

```
Generation finished!
Matrix:
XX YY VV YY WW YY YY
ZZ VV YY YY XX ZZ VV
VV ZZ XX YY YY YY WW
WW ZZ YY YY YY YY YY
XX XX XX XX WW WW VV
XX VV VV XX ZZ VV ZZ
ZZ ZZ XX XX YY VV ZZ

Sequences:
VV ZZ YY XX: 8
VV YY YY: 48
VV VV ZZ XX: 19
YY ZZ YY: 14
```

```
Generating solve...
Solve finished!

XX YY VV YY WW YY YY
ZZ VV YY YY XX ZZ VV
VV ZZ XX YY YY YY WW
WW ZZ YY YY YY YY YY
XX XX XX XX WW WW VV
XX VV VV XX ZZ VV ZZ
ZZ ZZ XX XX YY VV ZZ

Maximum Points     : 67
Path               : VV YY YY VV VV ZZ XX
Path Coordinates   : [(3, 1), (3, 4), (6, 4), (6, 6), (2, 6), (2, 3), (3, 3)]

Time taken: 301.0 ms

Do you want to save the results to a .txt file? (Y/N): y
Choice not recognized!
Do you want to save the results to a .txt file? (Y/N): y
Choice not recognized!
Do you want to save the results to a .txt file? (Y/N): Y

Saving to test/random_12_27_50...
File has been successfully saved!
```

6. **random_12_29_54**

```
$ python ./src/hack.py
Welcome to Cyberpunk 2077 Breach Protocol!
Please choose your input method:

1. Input from file
2. Randomly generate puzzle

Your choice: 2

Please provide the following parameters to aide the randomization process:

Token amount: 5

Tokens (seperated by spaces ; each token consist of two characters): AB CD EF GH IJ

Buffer size: 5

Matrix size (cols rows): 8 8

Sequence amount: 4

Maximum sequence length (>= 2): 4
```

```
Generation finished!
Matrix:
CD AB GH AB IJ GH CD IJ
AB AB EF IJ AB EF AB AB
IJ AB IJ IJ GH AB GH EF
CD IJ GH EF IJ EF CD AB
GH AB IJ AB IJ EF GH AB
GH CD EF GH EF EF CD CD
CD EF IJ AB AB EF CD EF
CD IJ IJ EF GH IJ IJ AB

Sequences:
EF EF EF: 35
GH GH AB CD: 0
IJ EF GH GH: 26
AB EF AB: 41
```

```
Generating solve...
Solve finished!

CD AB GH AB IJ GH CD IJ
AB AB EF IJ AB EF AB AB
IJ AB IJ IJ GH AB GH EF
CD IJ GH EF IJ EF CD AB
GH AB IJ AB IJ EF GH AB
GH CD EF GH EF EF CD CD
CD EF IJ AB AB EF CD EF
CD IJ IJ EF GH IJ IJ AB

Maximum Points     : 41
Path               : CD AB AB EF AB
Path Coordinates   : [(1, 1), (1, 2), (2, 2), (2, 7), (4, 7)]

Time taken: 19.0 ms

Do you want to save the results to a .txt file? (Y/N): Y

Saving to test/random_12_29_54...
File has been successfully saved!
```

BAB V
TAUTAN *REPOSITORY*

Repository program dapat dilihat pada tautan berikut:
https://github.com/trimonuter/Tucil1_IF2211Stima_13522158