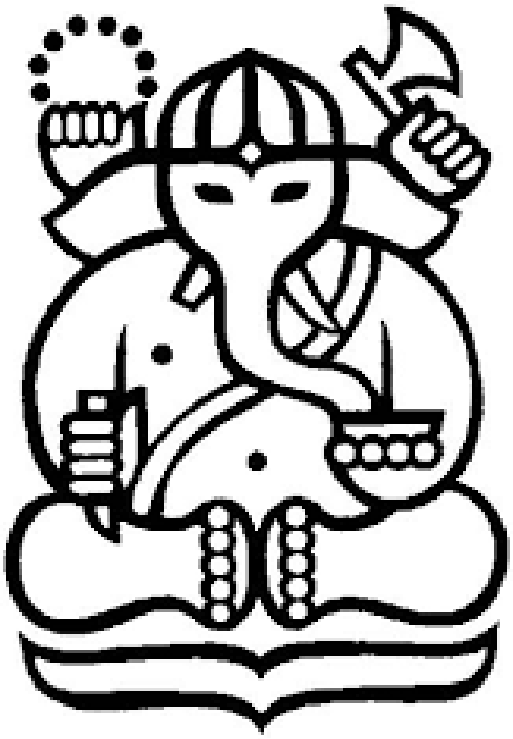
Laporan Tugas Kecil 1

IF2211 Strategi Algoritma

Penyelesaian Minigame Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force



Disusun Oleh

Fabian Radenta Bangun 13522105

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

Semester IV - 2024

A. Algoritma Penyelesaian Minigame Cyberpunk 2077 Breach Protocol dengan Brute Force

Berikut adalah rincian langkah berpikir dalam menemukan solusi :

1. Pilih token yang berada pada paling kiri atas pada matriks sebagai titik awal pencarian solusi
2. Kemudian akan dicari semua kemungkinan sequence yang ada untuk dimasukkan kedalam array solution
3. Pencarian akan dimulai dari titik awal yang berada pada kiri atas matriks secara vertikal kebawahnya
4. Kemudian pencarian akan terus berjalan secara bergantian vertikal, horizontal, vertikal, horizontal, dan seterusnya mulai dari selang 1 indeks, hingga ujung matriks.
5. Setelah semua kemungkinan sequence yang dihasilkan sudah masuk ke dalam array solution maka akan diperiksa apakah sequence yang menghasilkan point terdapat pada semua kemungkinan sequence yang telah ditemukan tadi
6. Kemudian dipilih sequence yang menghasilkan poin maksimum sebagai solusi
7. Jika poin maksimum nilainya 0 maka tidak ada solusi dari kasus tersebut

B. Source Code Program

```python
import os
import time

# CLASS
class Token :
    def __init__(self, token, absis, oordinat) :
        self.token = token
        self.absis = absis
        self.oordinat = oordinat

class Matrix :
    def __init__(self, data, width, height) :
        self.data = data
        self.width = width
        self.height = height

    def displayMatrix(self) :
        for i in range (self.height) :
            for j in range (self.width) :
                if (j == self.width-1) :
                    print(self.data[i][j].token)
                else :
                    print(f"{self.data[i][j].token} ",end='')

class Sequence :
    def __init__(self, sequence, point) :
        self.sequence = sequence
        self.point = point

    def displayData(self):
        first = True
        for seq in self.sequence:
            if first:
                print(seq, end="")
                first = False
            else:
                print(" " + seq, end="")
        print("\nPoints : " + str(self.point))
```

```python

# FUNCTION AND PROCEDURE
def isIn(row, column, stack) :
    # memeriksa apakah current token adalah token yang sedang menjadi pivot
    if stack != []:
        for token in stack:
            if (token.absis, token.oordinat) == (column, row):
                return True
    return False

def displaySequences(sequences) :
    # untuk menampilkan sequences yang ada ke terminal
    first = True
    print("-- Sequences --")
    for i in range (len(sequences)):
        if first:
            Sequence.displayData(sequences[i])
            first = False
        else:
            print("")
            Sequence.displayData(sequences[i])

```

```python
def findSequence(matrix, bufferSize, stack, row, col, solution, horizontal) :
    # mencari semua bentuk kemungkinan sequence kemudian memasukkan semua kemungkinan ke dalam array solution

    # base
    if (bufferSize == 1) :
        solution.append(stack[:])

    # recurrense
    else :
        if (horizontal) :
            # mencari sequence secara horizontal
            for i in range (matrix.width) :
                if not isIn(row, i, stack) :
                    stack.append(matrix.data[row][i])
                    findSequence(matrix, bufferSize - 1, stack, row, i,  solution, False)
                    stack.pop()
        else :
            # mencari sequence secara horizontal
            for i in range (matrix.height) :
                if not isIn(i, col, stack) :
                    stack.append(matrix.data[i][col])
                    findSequence(matrix, bufferSize - 1, stack, i, col,  solution, True)
                    stack.pop()

def isInSequence(bufferSequence, sequence) :
    # memeriksa apakah sequence ada di bufferSequence
    if len(sequence) > len(bufferSequence) :
        return False
    else :
        for i in range (len(bufferSequence) - len(sequence) + 1) :
            if bufferSequence[i] == sequence[0] :
                same = True
                for j in range (1, len(sequence)) :
                    if bufferSequence[i+j] != sequence[j] :
                        same = False
                        break
                if same :
                    return True
        return False

def convertToken(tokens) :
    # mengubah array of token menjadi array of string
    return [token.token for token in tokens]
```

```python
def getScore(sequences, tokens) :
    score = 0
    for subSeq in sequences:
        if isInSequence(convertToken(tokens), subSeq.sequence) :
            score += subSeq.point
    return score

def getResult(solutions, sequences) :
    max = getScore(sequences, solutions[0])
    solve = solutions[0]
    for i in range(1, len(solutions)):
        if getScore(sequences, solutions[i]) > max:
            max = getScore(sequences, solutions[i])
            solve = solutions[i]
    return max, solve

def welcome() :
    #menampilkan tampilan awal terminal
    print()
    print(10*'=' + " Cyberpunk 2077 Breach Protocol Solver " + 10*'=')
    print(59*'-')
    print()
```

```python
# MAIN PROGRAM
welcome()
auto = input("auto generate game? (y/n) ")
if (auto == 'y') :
    welcome() # Masih belum dibuat
elif (auto == 'n') :
    # mengambil masukan dari file txt
    fileName = input("Enter file name : ")
    path = os.path.join("..", "test", "input", fileName)
    try:
        file = open(path, 'r')
    except FileNotFoundError:
        print(fileName + " is not found, please recheck your filename\nExiting program...")
        exit()

    # read buffer size
    bufferSize = int(file.readline().strip())

    # read matrix size
    matrixWidth, matrixHeight = map(int, file.readline().split())

    # read matrix
    mainMatrix = []
    for i in range(matrixHeight):
        line = file.readline()
        line = line.rstrip('\n')
        line = line.split(" ")
        mainMatrix.append(line)
    for i in range(matrixHeight):
        for j in range(matrixWidth):
            mainMatrix[i][j] = Token(mainMatrix[i][j], j, i)
    mainMatrix = Matrix(mainMatrix, matrixHeight, matrixWidth)

    # read number of sequences
    numberOfSequences = int(file.readline())
```

```
    # read sequence and reward point
    arrayOfSequence = []
    for i in range (numberOfSequences*2) :
        line = file.readline()
        if (i%2 == 0) :
            tempSequence = (line.rstrip('\n')).split(" ")
        else :
            arrayOfSequence.append(Sequence(tempSequence, int(line)))

    # semua masukan sudah disimpan, kemudian masuk ke algoritma programnya di bawah

else :
    # keluar dari program jika command tidak sesuai
    print("\nCheck your command\nExiting program...")
    exit()


print("\nBuffer Size :", bufferSize,'\n')
print("Matrix : \n")
mainMatrix.displayMatrix()
print()
displaySequences(arrayOfSequence)
print()
print("Searching for solution...\n")
print()

# inisiasi awal
horizontal = True
stack = []
solutions = []

# mulai mencari semua kemungkinan sequence
findSequence(mainMatrix, bufferSize, stack,0, 0, solutions, True)

# simpan solusi berupa sequence hasil dan point nya
maxPoint, solutionSequence = getResult(solutions, arrayOfSequence)

if maxPoint == 0 :
    # jika poin maksimum yang didapatkan adalah 0 maka tidak ada sequence yang berhasil
    print("There is no solution\nExiting program...")
else :
    # cetak poin maksimum
    print(maxPoint)

    # cetak sequence yang menghasilkan poin maksimum
```

C. Hasil Eksekusi Program

D. Lampiran

Pranala menuju repository : https://github.com/fabianradenta/Tucil1_13522105

Tabel Spesifikasi Program

	Ya	Tidak
Program berhasil dikompilasi tanpa kesalahan		V

Program berhasil dijalankan	**V**	
Program dapat membaca masukan berkas .txt	**V**	
Program dapat menghasilkan masukan secara acak		**V**
Solusi yang diberikan program optimal		**V**
Program dapat menyimpan solusi dalam berkas .txt		**V**
Program memiliki GUI		**V**