**Laporan Tugas Kecil 1 IF2211 Strategi Algoritma**

**Semester II tahun 2023/2024**

Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force

Disusun oleh: Rayhan Ridhar Rahman (13522160)

**A screenshot of a computer

Description automatically generated**Cartoon of a person wearing a hat and coat

Description automatically generated

Idk, I just want Ishmael to impale me after this tucil :<

**PROGRAM STUDI TEKNIK INFORMATIKA SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA INSTITUT TEKNOLOGI BANDUNG 2023**

# Daftar Isi

# Penjelasan Algoritma

Algoritma brute force adalah pendekatan untuk mencari solusi dengan cara yang lempang. Algoritma ini dapat diterapkan untuk memecahkan hampir semua permasalahan dengan cara yang sederhana dan mudah dimengerti. Namun algoritma ini jarang sekali menghasilkan algoritma yang cepat dan juga dipandang tidak kreatif.

Terdapat beberapa komponen dan aturan dalam melakukan *Breach Protocol* pada game *Cyberpunk 2077* ini sesuai dengan spesifikasi tugas kecil. Komponen-komponen yang termasuk 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.

Kemudian aturan-aturan permainan ini yaitu:

1. Pemain bergerak secara horizontal dan vertikal (bergiliran) pada matriks, hingga buffer penuh atau tidak bisa lagi bergerak,
2. Dimulai dari bergerak horizontal pada baris pertama kemudian dilanjutkan pada kolom yang sama dengan pilihan, dan terus berganti,
3. Kemudian sekuens dicocokkan dengan token-token di buffer,
4. Token bisa digunakan oleh lebih dari satu sekuens,
5. Setiap sekuens memiliki bobot yang variative,
6. Sekuens setidaknya memiliki Panjang 2 token

Dengan ini, tugas kecil ini meminta untuk membuat algoritma yang dapat menentukan buffer yang tepat untuk nilai paling maksimum. Pemecahan yang diambil adalah, memeriksa buffer yang mungkin, kemudian mencocokkannya dengan sekuens. Maka dari itu terdapat algoritma yang menyusun buffer dan algoritma yang mencocokkannya. Terdapat 2 kelas buffer, 1 kelas matriks, dan 1 kelas sekuens pada pengerjaan. Buffer terbagi menjadi 2 yakni buffer template dan buffer optimal. Buffer template akan terus-menerus berubah sampai akhir algoritma mencari kemungkinan buffer, sementara buffer optimal akan berubah jika menemukan buffer yang berbobot lebih tinggi. Hal ini berkebalikan dengan Traveling Salesperson Problem.

# Source Code

Tugas ini dilaksanakan dengan menggunakan Bahasa pemrograman *Python* dengan library yang *os*, *random*, dan *time*

## Exception class

class NotCorrectColumnCountError(Exception):

    def \_\_init\_\_(self, message="Found a line with incorrect column size"):

        self.message = message

        super().\_\_init\_\_(self.message)

class NotEnoughTokens(Exception):

    def \_\_init\_\_(self, message="Doesn't have enough unique token input"):

        self.message = message

        super().\_\_init\_\_(self.message)

Kelas ini digunakan untuk beberapa *exception* yang mungkin terjadi.

## Matrix class

class BreachMatrix:

    def \_\_init\_\_(self):

        self.matrix = []

        self.row = 0

        self.col = 0

    def fill\_empty\_matrix(self,tokens):

        for i in range(self.row):

            self.matrix.append(random.choices(tokens, k = self.col))

    def append\_to\_matrix(self,Row):

        self.matrix.append(Row)

    def print\_matrix(self):

        for i in range(self.row):

            for j in range(self.col - 1):

                print(self.matrix[i][j], end=" ")

            print(self.matrix[i][self.col - 1], end="\n")

Kelas ini digunakan untuk membentuk matriks yang akan digunakan dengan atribut konten matriks, Panjang matriks, dan lebar matriks.

## Sequence class

class Sequence:

    def \_\_init\_\_(self):

        self.sequence\_list = []

        self.sequence\_points = []

        self.sequence\_number = 0

    def add\_sequence(self,newseq,newpoints):

        if not newseq in self.sequence\_list:

            self.sequence\_list.append(newseq)

            self.sequence\_points.append(newpoints)

            self.sequence\_number += 1

    def automatic\_sequences(self,tokens,seq\_n,max\_length,max\_points):

        while self.sequence\_number < seq\_n:

            token\_numbers = random.randint(2,max\_length)

            new\_seq = random.choices(tokens, k = token\_numbers)

            points = random.randint(0,max\_points)

            self.add\_sequence(new\_seq,points)

    def print\_sequence(self):

        for i in range(self.sequence\_number):

            for cseq in self.sequence\_list[i]:

                print(cseq, end=" ")

            print(":",self.sequence\_points[i],"points")

Kelas ini digunakan sebagai tempat penyimpanan semua sekuens yang dapat dicocokkan dengan atribut list berupa sekuens token, list berupa poin yang indeksnya bersesuaian, dan banyaknya sekuens yang akan dibandingkan.

## Buffer class

class Buffer:

    def \_\_init\_\_(self) -> None:

        self.buffer\_size = 0

        self.buffer\_coordinates = []

        self.buffer\_sequence = []

        self.buffer\_points = 0

        self.buffer\_used = 0

    def append\_el(self,el,iRow,iCol):

        self.buffer\_coordinates.append((iRow,iCol))

        self.buffer\_sequence.append(el)

        self.buffer\_used += 1

    def pop\_last(self):

        self.buffer\_coordinates.pop()

        self.buffer\_sequence.pop()

        if self.buffer\_used > 0:

            self.buffer\_used -= 1

    def print\_buffer(self):

        for token in self.buffer\_sequence:

            print(token, end=" ")

        print(":",self.buffer\_points,"points")

        print("Coordinates")

        for coord in self.buffer\_coordinates:

            print(f"({coord[1] + 1}, {coord[0] + 1})")

Kelas ini digunakan untuk buffer, buffer size merujuk pada panjang maksimum buffer sedangkan buffer used merujuk pada panjang yang sebenarnya digunakan, mungkin terdapat buffer yang setelah token terakhir tidak bisa bergerak kemana pun lagi atau ketika pencocokkan sekuens berakhir sebelum Panjang maksimum. Lalu terdapat juga koordinat dalam index python diisi dengan tuple row – column.

## Read input

def external\_fileI(buffer,matrix,sequence):

    file\_name = input("Enter the name of file being tested : ")

    file\_path = os.path.join(os.path.dirname(os.path.abspath(\_\_file\_\_)),"..","test",file\_name)

    with open(file\_path,'r') as rfile:

        content = rfile.readline()

        buffer.buffer\_size = int(content.strip())

        content = rfile.readline()

        matrix.col, matrix.row = map(int, content.split())

        for \_ in range(matrix.row):

            content = rfile.readline()

            if len(content.split()) != matrix.col:

                    raise NotCorrectColumnCountError

            matrix.append\_to\_matrix(content.split())

        content = rfile.readline()

        seq\_n = int(content.strip())

        for \_ in range (seq\_n):

            content = rfile.readline()

            new\_seq = content.split()

            content = rfile.readline()

            new\_points = int(content.strip())

            if len(content) > 0:

                sequence.add\_sequence(new\_seq,new\_points)

def CLI\_Input(buffer,matrix,sequence):

    n\_token = int(input("Masukkan jumlah token unik: "))

    tokens = list(set(input("Masukkan token unik: ").split()))

    if len(tokens) != n\_token:

        raise NotEnoughTokens

    buffer.buffer\_size = int(input("Masukkan panjang buffer: "))

    matrix.col, matrix.row = map(int,input("Masukkan ukuran matriks (Kolom Baris): ").split())

    seq\_n = int(input("Masukkan jumlah sequence yang mungkin dihasilkan: "))

    max\_seq\_len = int(input("Masukkan panjang maksimum sequence (Sequence max point = 5 x Max Length): "))

    max\_seq\_points = 5 \* max\_seq\_len

    sequence.automatic\_sequences(tokens,seq\_n,max\_seq\_len,max\_seq\_points)

    matrix.fill\_empty\_matrix(tokens)

Kedua prosedur tersebut menerima input dari command line (CLU) ataupun dari file eksternal untuk membentuk matriks dan sekuens serta menentukan Panjang maksimal dari buffer yang mungkin.

## Write output

def savingfile(buffer,searchtime):

    file\_name = input("Tolong beri nama file output (Akan disimpan dalam ekstensi .txt) : ") + ".txt"

    file\_path = os.path.join(os.path.dirname(os.path.abspath(\_\_file\_\_)),"..","test",file\_name)

    if os.path.exists(file\_path):

        print("Sudah ada file dengan nama tersebut, apakah ingin ditimpa? (y/n)")

        selection = input("> ")

        while selection != "y" and selection != "Y" and selection != "n" and selection != "N":

            selection = input("> ")

        if selection == "n" or selection == "N":

            savingfile(buffer,searchtime)

        else:

            print("Menimpa file . . .")

            writeoutput(buffer,searchtime,file\_path)

    else:

        print("Menyimpan file . . .")

        writeoutput(buffer,searchtime,file\_path)

def writeoutput(buffer,searchtime,filepath):

    with open(filepath,'w') as wfile:

        line = str(buffer.buffer\_points) + "\n"

        wfile.write(line)

        line = ""

        for i in range(buffer.buffer\_used):

            line = line + buffer.buffer\_sequence[i]

            if i < buffer.buffer\_used - 1:

                line = line + " "

            else:

                line = line + "\n"

        wfile.write(line)

        for i in range(buffer.buffer\_used):

            line = str(buffer.buffer\_coordinates[i][1] + 1) + ", " + str(buffer.buffer\_coordinates[i][0] + 1) + "\n"

            wfile.write(line)

        line = "\n" + str(searchtime) + " ms"

        wfile.write(line)

Algoritma tersebut adalah untuk melakukan penulisan ke file .txt baru di folder test, dengan format yang sesuai dengan spesifikasi.

## Breach protocol algorithm

def breachprotocol(buffer, matrix, sequence, iRow, iCol, optimalBuffer):

    breachMatrix = matrix.matrix

    if not (iRow,iCol) in buffer.buffer\_coordinates:

        buffer.append\_el(breachMatrix[iRow][iCol],iRow,iCol)

    loop = 0

    newRow = (iRow + buffer.buffer\_used % 2) % matrix.row

    newCol = (iCol + (buffer.buffer\_used + 1) % 2) % matrix.col

    while buffer.buffer\_used < buffer.buffer\_size and (newRow != iRow or newCol != iCol):

        if not (newRow,newCol) in buffer.buffer\_coordinates:

            breachprotocol(buffer, matrix, sequence, newRow, newCol, optimalBuffer)

            loop += 1

        newRow = (newRow + buffer.buffer\_used % 2) % matrix.row

        newCol = (newCol + (buffer.buffer\_used + 1) % 2) % matrix.col

    if loop == 0:

        sequence\_search(buffer, sequence, optimalBuffer)

    buffer.pop\_last()

def sequence\_search(buffer, sequence, optimalBuffer):

    checked\_sequence = buffer.buffer\_sequence

    compared\_sequences = sequence.sequence\_list

    point\_distribution = sequence.sequence\_points

    total\_points = 0

    used\_size = 0

    buf\_len = len(checked\_sequence)

    seq\_n = sequence.sequence\_number

    for i in range(seq\_n):

        seqt\_len = len(compared\_sequences[i])

        j = 0

        while j + seqt\_len <= buf\_len and checked\_sequence[j : j + seqt\_len] != compared\_sequences[i]:

             j += 1

        if j + seqt\_len <= buf\_len:

            total\_points += point\_distribution[i]

            if used\_size < j + seqt\_len:

                used\_size = j + seqt\_len

Kedua prosedur tersebut bermaksud untuk mencari buffer paling optimal dengan awalan kolom yang ditentukan (diperjelas pada main). Dalam program yang dibuat, buffer paling optimal adalah buffer dengan poin terbesar dengan ukuran terkecil. Pencarian kemungkinan buffer dilakukan secara looping dan rekursif. Terdapat variabel loop untuk memastikan buffer yang dipilih untuk dicocokkan sudah buffer seutuhnya dan bukan merupakan sub-buffer dari yang lain. Kemudian pada prosedur sequence search, buffer yang lolos akan dikompresi sampai akhir buffer merupakan akhir sekuens yang terakhir.

## Main

# Uji Coba

# Lampiran

Pranala repository GitHub:

<https://github.com/Rinnearu/Tucil1_13522160>