**LAPORAN TUGAS KECIL I**

**IF2211 STRATEGI ALGORITMA**

Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force

A blue and white logo

Description automatically generated

Disusun oleh:

Derwin Rustanly 13522115

**Program Studi Teknik Informatika**

**Sekolah Teknik Elektro dan Informatika**

**Institut Teknologi Bandung**

**2022**

Daftar Isi

[Bab I 3](#_Toc158491703)

[Bab II 5](#_Toc158491704)

[Bab III 16](#_Toc158491705)

# Bab I

**Penjelasan Algoritma Brute Force**

Permainan Breach Protocol merupakan salah satu sub permainan dari permainan video daring yang berjudul Cyberpunk 2077. Permainan ini meminta pengguna untuk melakukan peretasan sederhana yang meliputi 4 komponen utama, di antaranya:

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.

Secara sederhana aturan dari permainan ini adalah mengoptimalkan pencocokan pola token pada matriks dan sekuens berbobot hadiah variatif dengan memanfaatkan buffer yang memiliki batasan ukuran tertentu baik secara horizontal dan vertikal secara bergantian. Tujuan utama dari permainan ini adalah memaksimalkan bobot hadiah yang diperoleh pada buffer dengan memanfaatkan langkah yang seminimum mungkin. Salah satu algoritma penyelesaian dari persoalan Breach Protocol ini adalah dengan memanfaatkan algoritma brute force. Algoritma brute force dapat dikatakan sebagai algoritma yang paling naif dan intuitif untuk menyelesaikan permasalahan Breach Protocol tersebut karena penyelesaiannya hanya bergantung pada pernyataan yang terdapat pada persoalan serta definisi yang dilibatkan.

Dalam hal ini, algoritma brute force yang digunakan adalah sebagai berikut.

1. Setiap token pada baris pertama disimpan pada token pertama buffer, kemudian dilakukan pengelompokkan terhadap setiap token pada kolom yang berkaitan.
2. Setiap token yang terletak pada kolom yang berkaitan dengan token pertama buffer disimpan sebagai token buffer berikutnya, selanjutnya dilakukan pula pengelompokkan terhadap setiap token pada baris yang bersesuaian.
3. Setelah didapatkan buffer pertama yang setidaknya berukuran lebih dari 2 token, dilakukan perhitungan jumlah kemunculan dari sekuens pada buffer tersebut dan dilakukan evaluasi poin hadiah terhadap sekuens yang tersedia,kemudian buffer tersebut disimpan sebagai buffer dengan maksimum sementara.
4. Setelah didapatkan buffer lain yang setidaknya juga berukuran lebih dari 2 token, dilakukan perhitungan jumlah kemunculan dari sekuens pada buffer tersebut dan dilakukan evaluasi poin terhaadap sekuens yang tersedia sebagaimana tertulis pada langkah ke-4. Apabila didapatkan alternatif buffer yang lebih optimal dalam artian memiliki jumlah poin hadiah besar daripada buffer maksimum sebelumnya, dilakukan penggantian buffer maksimum dengan buffer tersebut.
5. Langkah 1 hingga 4 dilakukan secara rekursif hingga panjang buffer telah mencapai ukuran yang telah ditentukan.

Dalam hal perhitungan jumlah kemunculan sekuens pada buffer juga dapat digunakan algoritma brute force pula, yakni sebagai berikut.

1. Setiap token pada buffer dibandingkan dengan setiap token pada setiap token pada masing-masing sekuens yang tersedia.
2. Apabila token yang ditemukan berbeda, dilakukan penggantian pengecekan ke token buffer berikutnya.
3. Apabila token yang ditemukan sesuai, dilakukan pengecekan ke token buffer dan token sekuens yang berikutnya.
4. Langkah 2 dan 3 dilakukan secara iteratif hingga pengecekkan mencapai ujung token atau didapatkan sekuens yang muncul pada buffer.

# Bab II

**Penjelasan Source Program**

Dalam implementasi penyelesaian dari permainan Cyberpunk Breach Protocol, digunakan bahasa pemrograman Python untuk merealisasikan source program. Program yang direalisasikan berbasis GUI berupa aplikasi web yang dikembangkan dengan *framework* React sebagai *front-end* dan Flask sebagai *back-end*. Implementasi algoritma brute force dapat ditemukan pada file object.py yang menggunakan konsep pemrograman berorientasi objek, yakni komponen *class* dan *object* untuk merepresentasikan tipe beserta primitif fungsi dari sekuens, koordinat, dan permainan Breach Protocol tersebut sendiri. Berikut merupakan *source code* Python dari file object.py yang telah direalisasikan.

File: object.py

class Coordinate:

    def \_\_init\_\_(self, x, y):

        self.x = x

        self.y = y

    def isCoincident(P1, P2):

        return P1.x == P2.x and P1.y == P2.y

    def isMember(P1, L):

        found = False; i = 0

        while(not(found) and i < len(L)):

            if Coordinate.isCoincident(L[i], P1):

                found = True

            i += 1

        return found

class Sequence:

    def \_\_init\_\_(self, string, val):

        self.string = string

        self.val = val

    def join(self, token):

        self.string += token

    def isSubstring(self, substring):

        return substring in self.string

    def getPoint(self, list, num):

        return sum(item.val for item in list if self.isSubstring(item.string))

class Game:

    def \_\_init\_\_(self, size, width, height, matrix, num, sequence):

        self.size = size

        self.width = width

        self.height = height

        self.matrix = matrix

        self.num = num

        self.sequence = sequence

        self.current = Sequence("",0)

        self.max = Sequence("",0)

        self.maxlen = size\*2

        self.coordinate = [Coordinate(-99,-99) for i in range(self.size)]

        self.maxcoordinate = [Coordinate(-99,-99) for i in range(self.size)]

    def infoGame(self):

        print(self.size)

        print(self.width,self.height)

        for i in range(self.height):

            for j in range(self.width):

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

            print()

        print(self.num)

        for i in range(self.num):

            print("(",end="")

            for j in range(len(self.sequence[i].string)):

                if j%2 == 0 and j > 0 :

                    print(" ",end="")

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

            print(f",{self.sequence[i].val})")

    def search(self, isVertical, idx, pivot, buffer, target):

        if idx == target:

            self.current.string = "".join(buffer)

            self.current.val = self.current.getPoint(self.sequence, self.num)

            if self.current.val >= self.max.val and len(self.current.string) <= self.maxlen:

                self.max.string = self.current.string

                self.max.val = self.current.val

                self.maxlen = len(self.max.string)

                self.maxcoordinate = self.coordinate.copy()

            return None

        if isVertical:

            for i in range(self.height-1,-1,-1):

                currentCoordinate = Coordinate(pivot, i)

                if not(Coordinate.isMember(currentCoordinate, self.coordinate)):

                    buffer[idx] = self.matrix[i][pivot]

                    self.coordinate[idx] = currentCoordinate

                    self.search(not(isVertical),idx+1, i, buffer, target)

        else:

            for j in range(self.width-1,-1,-1):

                currentCoordinate = Coordinate(j,pivot)

                if not(Coordinate.isMember(currentCoordinate, self.coordinate)):

                    buffer[idx] = self.matrix[pivot][j]

                    self.coordinate[idx] = currentCoordinate

                    self.search(not(isVertical),idx+1, j, buffer, target)

    def solution(self):

        for k in range(self.size,1,-1):

            buffer = ["" for i in range(self.size)]

            for j in range(self.width-1,-1,-1):

                isVertical = True

                buffer[0] = self.matrix[0][j]

                self.coordinate[0] = Coordinate(j,0)

                self.search(isVertical, 1,j, buffer,k)

        print(self.max.val)

        if(self.max.val != 0):

            for i in range(len(self.max.string)):

                print(self.max.string[i],end = "")

                if i%2 != 0:

                    print(" ",end="")

            print()

            for i in range(self.maxlen//2):

                print(f"{i+1}. ({self.maxcoordinate[i].x+1},{self.maxcoordinate[i].y+1})")

        else:

            self.maxlen = 0

            self.max.string = ""

Sementara itu, file app.py berfungsi untuk merealisasikan fitur masukan dari file maupun masukan secara acak, sekaligus sebagai API (Application Programming Isnterface) dari masukan dari web. Berikut merupakan *source code* Python dari file app.py.

File: app.py

import os

import random

import time

from object import Sequence, Game

from flask import Flask, request, jsonify

from flask\_cors import CORS

app = Flask(\_\_name\_\_)

CORS(app)

dir = 'static'

os.makedirs(dir, exist\_ok=True)

@app.route('/upload', methods=['POST'])

def upload\_file():

# Check if the POST request contains a file

if 'file' not in request.files:

return jsonify({'error': 'No file part'}), 400

file = request.files['file']

# If the user does not select a file, the browser may send an empty file without a filename

if file.filename == '':

return jsonify({'error': 'No selected file'}), 400

# Save the uploaded file locally

print("name: " + file.filename)

file\_name = dir+'/'+file.filename

file.save(file\_name)

game = txt\_reader(file\_name)

start = time.time()

game.solution()

end = time.time()

dur = (end-start)\*1000

response = {

'sequences' : [[[game.sequence[i].string[j:j+2] for j in range(0,len(game.sequence[i].string),2)], game.sequence[i].val] for i in range(len(game.sequence))] ,

'width': game.width,

'height': game.height,

'matrix' : game.matrix,

'maxbuffer' : [game.max.string[i:i+2] for i in range(0,len(game.max.string),2)],

'maxvalue' : game.max.val,

'maxcoordinate' : [(game.maxcoordinate[i].y, game.maxcoordinate[i].x) for i in range(game.maxlen//2)],

'duration' : dur

}

return jsonify(response), 200

@app.route('/input', methods=['POST'])

def input\_random():

input\_data = request.get\_json()

numToken = int(input\_data.get("numToken"))

listToken = input\_data.get("listToken").split(" ")

size = int(input\_data.get("size"))

width = int(input\_data.get("width"))

height = int(input\_data.get("height"))

num = int(input\_data.get("num"))

seqSize = int(input\_data.get("seqSize"))

if numToken <= 0 or numToken == None or listToken == [] or listToken == None or size <= 0 or size == None or width <= 0 or width == None or height <= 0 or height == None or num <= 0 or num == None or seqSize <= 0 or seqSize == None:

return jsonify({'error': 'Missing filename, reward, buffer, or coordinates'}), 400

mat = [[listToken[random.randint(0,numToken-1)] for j in range(width)] for i in range(height)]

sequence = [Sequence("",0) for i in range(num)]

for i in range(num):

check = False

temp = Sequence("",0)

while(not(check)):

for j in range(random.randint(2,seqSize)):

temp.string += listToken[random.randint(0,numToken-1)]

temp.val = random.randrange(10,40,5)

if temp not in sequence:

check = True

sequence[i].string = temp.string

sequence[i].val = temp.val

game = Game(size,width,height,mat,num,sequence)

start = time.time()

game.solution()

end = time.time()

dur = (end-start)\*1000

response = {

'sequences' : [[[game.sequence[i].string[j:j+2] for j in range(0,len(game.sequence[i].string),2)], game.sequence[i].val] for i in range(len(game.sequence))] ,

'width': game.width,

'height': game.height,

'matrix' : game.matrix,

'maxbuffer' : [game.max.string[i:i+2] for i in range(0,len(game.max.string),2)],

'maxvalue' : game.max.val,

'maxcoordinate' : [(game.maxcoordinate[i].y, game.maxcoordinate[i].x) for i in range(game.maxlen//2)],

'duration' : dur

}

return jsonify(response), 200

@app.route('/save', methods=['POST'])

def save\_file():

data = request.get\_json()

file = data.get('filename')

reward = data.get('reward')

buffer = data.get('buffer')

coordinates = data.get('coordinates')

duration = data.get('duration')

if file == "" or reward == None or buffer == [] or coordinates == []:

return jsonify({'error': 'Missing filename, reward, buffer, or coordinates'}), 400

file = "../../test/output/"+file

with open(file, 'w') as savefile:

savefile.write(str(reward)+"\n")

for token in buffer:

savefile.write(token)

savefile.write(" ")

savefile.write("\n")

for coordinate in coordinates:

savefile.write("("+str(coordinate[1]+1)+","+str(coordinate[0]+1)+")\n")

savefile.write("\n")

savefile.write(str(duration) + "ms")

print(reward)

print(buffer)

print(coordinates)

print(duration)

return jsonify({'message': 'File saved successfully'}), 200

def txt\_reader(path):

with open(path, "r") as file:

size = int(file.readline())

width, height = map(int, file.readline().split(" "))

mat = [[0 for i in range(width)] for j in range(height)]

for i in range(height):

for j in range(width):

mat[i][j] = file.read(2)

file.read(1)

n = int(file.readline())

seq = ["" for i in range(n)]

for i in range(n):

s = "".join(file.readline().strip().split(" "))

val = int(file.readline())

seq[i] = Sequence(s, val)

return Game(size,width,height,mat,n,seq)

def stdin():

numToken = int(input("Masukkan jumlah token unik: "))

while numToken <= 0:

print("Jumlah token unik setidaknya 1")

numToken = int(input("Masukkan jumlah token unik: "))

listToken = input("Masukkan token: ").split()

while len(listToken) != numToken:

print("Harap masukkan token dengan jumlah yang sesuai")

listToken = input("Masukkan token: ").split()

size = int(input("Masukkan ukuran buffer: "))

while size <= 0:

print("Ukuran buffer setidaknya 1")

size = int(input("Masukkan ukuran buffer: "))

width,height = map(int,input("Masukkan ukuran matriks dalam format m n: ").split())

mat = [[listToken[random.randint(0,numToken-1)] for j in range(width)] for i in range(height)]

num = int(input("Masukkan jumlah sekuens: "))

sequence = [Sequence("",0) for i in range(num)]

seqSize = int(input("Masukkan ukuran maksimal sekuens: "))

for i in range(num):

check = False

temp = Sequence("",0)

while(not(check)):

for j in range(random.randint(2,seqSize)):

temp.string += listToken[random.randint(0,numToken-1)]

temp.val = random.randrange(10,40,5)

if temp not in sequence:

check = True

sequence[i].string = temp.string

sequence[i].val = temp.val

return Game(size,width,height,mat,num,sequence)

def txt\_writer(path, game: Game):

with open(path, "w") as file:

file.write(str(game.max.val)+"\n")

for i in range(len(game.max.string)):

file.write(game.max.string[i])

if i%2 != 0:

file.write(" ")

file.write("\n")

for i in range(game.size):

file.write(f"{i+1}. ({game.maxcoordinate[i].x+1},{game.maxcoordinate[i].y+1})\n")

file.write("\n")

file.write(f"{(end-start)\*1000:.2f} ms")

if \_\_name\_\_ == "\_\_main\_\_":

path = "../../test/input/file.txt"

# game = txt\_reader(path)

game = stdin()

game.infoGame()

start = time.time()

game.solution()

end = time.time()

print(f"{(end-start)\*1000:.2f} ms")

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

if (isPrintable == "Y" or isPrintable == "y"):

output = input("Masukkan nama file output: ")

txt\_writer("../../test/output/"+output, game)

# Bab III

**Pengujian dan Tangkapan Layar**

1. Pengujian dengan metode masukan dari file sesuai dengan spesifikasi

File Input:

A black screen with white text

Description automatically generated

Tampilan Program:

A screenshot of a video game

Description automatically generated

File output

A screenshot of a computer

Description automatically generated

1. Pengujian dengan metode masukan file dengan sekuens berbobot negatif, kecuali salah satu sekuens

File Input:

A screenshot of a computer

Description automatically generated

Tampilan Program:

A screenshot of a computer

Description automatically generated

File Output:

A screenshot of a computer program

Description automatically generated

1. Pengujian dengan metode masukan file dengan keseluruhan sekuens yang berbobot negatif

File Input:

A screenshot of a computer

Description automatically generated

Tampilan Program:

Screens screenshot of a video game

Description automatically generated

File Output:

A screenshot of a computer

Description automatically generated­­­­­

1. Pengujian dengan metode masukan acak sesuai dengan spesifikasi

Tampilan Masukan:

A screenshot of a computer

Description automatically generated

Tampilan Program:

A screenshot of a computer

Description automatically generated

File Output:

A screenshot of a computer

Description automatically generated

1. Pengujian dengan metode masukan acak yang berbeda

Tampilan Masukan:

A screenshot of a computer

Description automatically generated

Tampilan Program:

A screenshot of a computer

Description automatically generated

File Output:

A screenshot of a computer

Description automatically generated

1. Pengujian dengan metode masukan acak yang berbeda

Tampilan Masukan:

A screenshot of a computer

Description automatically generated

Tampilan Program:

A screenshot of a video game

Description automatically generated

File Output:

A screenshot of a computer

Description automatically generated

# Bab IV

**Evaluasi**

|  |  |  |
| --- | --- | --- |
| Poin | Ya | Tidak |
| 1. Program berhasil dikompilasi tanpa kesalahan | ✓ |  |
| 1. Program berhasil dijalankan | ✓ |  |
| 1. Program dapat membaca masukan berkas .txt | ✓ |  |
| 1. Program dapat menghasilkan masukan secara acak | ✓ |  |
| 1. Solusi yang diberikan optimal | ✓ |  |
| 1. Program dapat menyimpan solusi dalam berkas .txt | ✓ |  |
| 1. Program memiliki GUI | ✓ |  |

Pranala menuju repositori: <https://github.com/DerwinRustanly/Tucil1_13522115>