# LAPORAN TUGAS KECIL 1

## IF2211 Strategi Algoritma

## Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force


Oleh:

Samy Muhammad Haikal (13522151)

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG


2023

**DAFTAR ISI**


**DAFTAR ISI** **2**
**BAB 1** **3**
1.1 Algoritma Bruteforce 3
1.2 Cyberpunk hacking puzzle 3
1.3 Algoritma Penyelesaian cyberpunk hacking puzzle dengan pendekatan bruteforce 4
**BAB II** **5**
2.1 Functions 5
**BAB III** **6**
3.1 Repository Program 6
3.2 Source Code Program 6
**BAB IV** **11**
4.1 Masukan dari file 11
4.2 Masukan random 11
**BAB V** **13**

# BAB 1
# DESKRIPSI MASALAH

## 1.1 Algoritma Bruteforce

Algoritma brute force merupakan sebuah pendekatan dari penyelesaian sebuah masalah algoritmik dengan menggunakan pendekatan yang lempeng (straightforward) untuk menyelesaikan sebuah persoalan yang telah didefinisikan. Algoritma brute force didasarkan pada pernyataan pada persoalan (problem statement) dan juga konsep yang dilibatkan oleh persoalan yang sedang dibahas.

Algoritma dengan pendekatan brute force biasanya memiliki ciri khas mempunyai konsep penyelesaian yang sederhana, langsung, jelas, dan intuitif. Pendekatan brute force seringkali melibatkan enumerasi semua kemungkinan solusi, sebelum akhirnya menghilangkan jawaban yang tidak memenuhi syarat dan mengambil solusi terbaik. Algoritma dengan pendekatan brute force dijamin akan menemukan sebuah solusi apabila solusi tersebut ada. Namun, algoritma dengan pendekatan brute force seringkali tidak mangkus atau tidak efektif, dengan O(n) yang lebih buruk dari polinomial.

## 1.2 Cyberpunk hacking puzzle

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.

### 1.3 Algoritma Penyelesaian cyberpunk hacking puzzle dengan pendekatan bruteforce

Dalam menyelesaikan breach protocol secara algoritmik, penulis menggunakan pendekatan *brute force.* Adapun langkah-langkah penyelesaian masalah dalam algoritma dapat dijelaskan sebagai berikut:

1. Program meminta input, baik dari file atau input manual yang dirandomisasi
2. Program akan mencari semua kemungkinan sekuens sel yang panjangnya kurang dari ukuran buffer. Untuk proses ini menggunakan fungsi yang bersifat rekursif
3. Program akan mengkalkulasi nilai dari setiap sekuens dengan mencocokkannya dengan sekuens berhadiah
4. Setelah dilakukan pengkalkulasian nilai, program akan menyeleksi sekuens dengan kriteria nilai terbesar dan panjang terpendek
5. Keluarkan nilai, sekuens, dan koordinat dari solusi yang sudah diseleksi

# BAB II
# IMPLEMENTASI ALGORITMA DALAM BAHASA PYTHON

Dalam pembuatan program ini, penulis menggunakan bahasa pemrograman python. Program hanya terdiri dari file main.py.

## 2.1 Functions

| Functions | Description |
|---|---|
| `def main():` | Fungsi untuk menjalankan program utama, terdapat main menu |
| `def saveFile():` | Fungsi untuk menyimpan hasil ke dalam suatu file |
| `def printmatrix(matrix):` | Fungsi untuk melakukan print matrix |
| `def generateRandomSeq(tokens, maxSeqSize):` | Fungsi untuk men-*generate* sekuens-sekuens random |
| `def inputfile(filename):` | Fungsi untuk membaca file konfigurasi jika user memilih untuk menginput dari file |
| `def find_sequences(matrix, visited, path, row, col, is_horizontal, arrseq):` | Fungsi untuk mencari semua sekuens yang ada pada matrix dengan ukuran buffer tertentu |
| `def Solve(Seqs, reward):` | Fungsi untuk mencocokkan sekuens yang ada dengan sekuens berhadiah |
| `def output(list_hasil, list_sequence):` | Fungsi untuk menampilkan hasil dari pencocokan |

# BAB III
# SOURCE CODE PROGRAM

## 3.1 Repository Program

Repository program dapat diakses dengan tautan github berikut:
https://github.com/rendangmunir/Tucil1_13522151

## 3.2 Source Code Program

```
import time
import random


buffsize = 0
row = 0
col = 0
list_sequence = []
matrix = []
buffhasil = ''
maxSeq =[]


def main():
    global buffhasil, list_sequence, matrix, buffsize
    print("===== MENU UTAMA =====")
    print("1. Input File")
    print("2. Input manual/Random")
    print("3. Exit")
    pilihan = int(input("Pilihan (1/2/3) : "))
    if (pilihan==1):
        file_loc = input("Masukkan alamat file: ")
        inputfile(file_loc)
        print(matrix)
        x = []
        start = time.time()
        for col in range(len(matrix[0])):
            find_sequences(matrix, [], [], 0, col, True, x)
        list_hasil = Solve(x, list_sequence)
        end = time.time()
        duration = round((end-start)*1000)
        output(list_hasil, x)
```

```
        print(f'{duration} ms')
        buffhasil+=f'{duration} ms'
        saveFile()
    elif (pilihan==2):
        tokenCnt = int(input("Masukkan jumlah token unik : "))
        token = input("Masukkan token : ")
        token = token.split()
        buffsize = int(input("Masukkan ukuran buffer : "))
        size = input("Masukkan ukuran matrix : ")
        size = size.split()
        col = int(size[0])
        row = int(size[1])
        for i in range(row):
            matrix.append([])
            for j in range(col):
                rand_int = random.randint(0,tokenCnt-1)
                matrix[i].append(token[rand_int])

        SeqCnt = int(input("Masukkan jumlah sekuens : "))
        maxSeqSize = int(input("Masukkan ukuran maksimal seqkuens : "))
        for i in range(SeqCnt):#generate sequence
            seq = generateRandomSeq(token, maxSeqSize)
            while(seq in list_sequence):
                seq = generateRandomSeq(token, maxSeqSize)
            reward = random.randint(0,50)
            list_sequence.append([seq, reward])

        #output matrix and sequences
        print("Matrix: ")
        for row in matrix:
            print(' '.join(map(str, row)))
        for i in range(len(list_sequence)):
                sequences = ' '.join(list_sequence[i][0][j:j+2] for j in
range(0, len(list_sequence[i][0]), 2))
            print(f'Sequence: {sequences}')
            print(f'Reward: {list_sequence[i][1]}')

        x=[]
        start = time.time()
```

```
        for i in range(len(matrix[0])):
            find_sequences(matrix, [], [], 0, i, True, x)
        list_hasil = Solve(x, list_sequence)
        end = time.time()
        duration = round((end-start)*1000)

        output(list_hasil,x)
        print(f'{duration} ms')
        buffhasil+=f'{duration} ms'
        saveFile()

def printmatrix(matrix):
    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            print(f'{matrix[i][j]}')
            if (j<len(matrix[0])-1):
                print(" ")
        print("\n")

def saveFile():
    global buffhasil
    saveChoice = input("Apakah anda ingin menyimpan solusi? (y/n) ")
    if (saveChoice in ['y', 'Y']):
        outputName = input("Masukkan nama file: ")
        with open ("../test/"+outputName, "w") as file:
            file.write(buffhasil)

def generateRandomSeq(tokens, maxSeqSize):
    global list_sequence
    sequence = ''
    sequence_size = random.randint(2,maxSeqSize)
    for _ in range(sequence_size):
        token = random.choice(tokens)
        sequence += token
    return sequence




def inputfile(filename):
```

```
    global buffsize, col, row, matrix, list_sequence
    contents = []
    with open('../test/'+filename, 'r') as file:
        for line in file:
            contents.append(line)
    buffsize = int(contents[0])
    size = contents[1].split()
    col = int(size[0])
    row = int(size[1])
    for i in range(row):
        matCol = contents[2+i].split()
        matrix.append(matCol)
    SeqCnt = int(contents[2+row])
    for i in range(0,SeqCnt*2,2):
         list_sequence.append([contents[3+row+i].replace(" ", '').strip(),
int(contents[4+row+i])])




def  find_sequences(matrix,  visited,  path,  row,  col,  is_horizontal,
arrseq):
    global buffsize
    # Append the current cell to the path
    path.append(matrix[row][col])
    visited.append([row,col])

    # Basis rekursi
    if len(path) <= buffsize:
        arrseq.append([path[:], visited[:]])
        # print(arrs)

    # Define possible moves (vertically first, then horizontally)
    if (len(path)<buffsize):
        if is_horizontal:
            for i in range(len(matrix)):
                new_row, new_col = i,col
                if [new_row,new_col] not in visited:
                        find_sequences(matrix, visited.copy(), path.copy(),
new_row, new_col, not is_horizontal, arrseq)
```

```
        else:
            for i in range(len(matrix[0])):
                new_row, new_col = row,i
                if [new_row,new_col] not in visited:
                        find_sequences(matrix, visited.copy(), path.copy(),
new_row, new_col, not is_horizontal,arrseq)

def Solve(Seqs, reward):
    global list_sequence, maxSeq
    list_hasil = []
    # print(Seqs)
    for sequence in Seqs:
        hasil = 0
        answer = ''.join(sequence[0])
        # print(answer)
        for x in reward:
            # print(answer, x[0], x[1])
            if x[0] in answer:
                hasil+=x[1]
        list_hasil.append(hasil)
    # print(list_hasil)
    maxVal = max(list_hasil)
      max_indexes = [index for index, value in enumerate(list_hasil) if
value == maxVal]
    solutions = []
    for i in range(len(max_indexes)):
        solutions.append(Seqs[max_indexes[i]])
    maxSeq = []
    if (len(solutions)<=1):
        maxSeq = solutions[0]
    else:
        for i in range(len(solutions)-1):
            if (i==0):
                maxSeq = solutions[i]
            if (len(maxSeq[0])>len(solutions[i+1][0])):
                maxSeq = solutions[i+1]

    return list_hasil

def output(list_hasil, list_sequence):
```

```
    global buffhasil, maxSeq
    print(max(list_hasil))
    buffhasil+=f'{max(list_hasil)}\n'
    sequence = ' '.join(maxSeq[0])
    print(sequence)
    buffhasil+=f'{sequence}\n'
    for i in range(len(maxSeq[1])):#print koordinat
        print(f'{maxSeq[1][i][1]+1}, {maxSeq[1][i][0]+1}')
        buffhasil+=f'{maxSeq[1][i][1]+1}, {maxSeq[1][i][0]+1}\n'

main()
```

## BAB IV
## MASUKAN DAN LUARAN PROGRAM

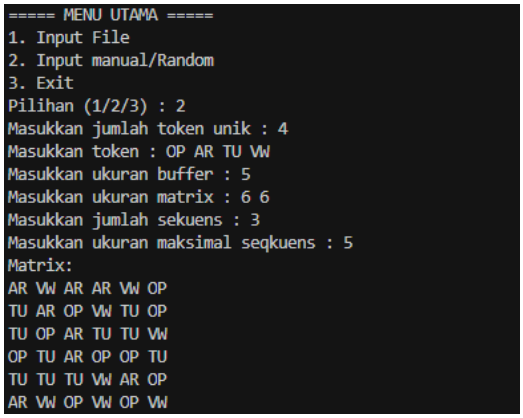
### 4.1 Masukan dari file

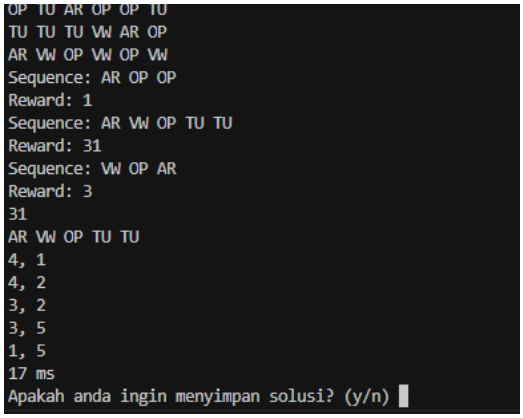
```
===== MENU UTAMA =====
1. Input File
2. Input manual/Random
3. Exit
Pilihan (1/2/3) : 1
Masukkan alamat file: test.txt
50
7A BD 7A BD 1C BD 55
1, 1
1, 4
3, 4
3, 5
6, 5
6, 3
1, 3
546 ms
Apakah anda ingin menyimpan solusi? (y/n)
```

File “test.txt”

```
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
```

### 4.2 Masukan random


===== MENU UTAMA =====
1. Input File
2. Input manual/Random
3. Exit
Pilihan (1/2/3) : 2
Masukkan jumlah token unik : 4
Masukkan token : OP AR TU VW
Masukkan ukuran buffer : 5
Masukkan ukuran matrix : 6 6
Masukkan jumlah sekuens : 3
Masukkan ukuran maksimal seqkuens : 5
Matrix:
AR VW AR AR VW OP
TU AR OP VW TU OP
TU OP AR TU TU VW
OP TU AR OP OP TU
TU TU TU VW AR OP
AR VW OP VW OP VW


OP TU AR OP OP TU
TU TU TU VW AR OP
AR VW OP VW OP VW
Sequence: AR OP OP
Reward: 1
Sequence: AR VW OP TU TU
Reward: 31
Sequence: VW OP AR
Reward: 3
31
AR VW OP TU TU
4, 1
4, 2
3, 2
3, 5
1, 5
17 ms
Apakah anda ingin menyimpan solusi? (y/n)

# BAB V
# 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 di berkas .txt | ✅ | |
| 7. Program memiliki GUI | | ✅ |