

**LAPORAN TUGAS KECIL I**  
**IF2211 STRATEGI ALGORITMA**

Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force



Disusun oleh:

Imam Hanif Mulyarahman 13522030

**Program Studi Teknik Informatika**  
**Sekolah Teknik Elektro dan Informatika**  
**Institut Teknologi Bandung**

**2024**

## **BAGIAN I**

### **ALGORITMA BRUTE FORCE**

Algoritma Brute force adalah algoritma yang menggunakan pendekatan yang lempang (straight forward) dalam memecahkan suatu persoalan. Algoritma ini biasanya didasarkan pada pernyataan persoalan (problem statement) dan definisi atau konsep yang dilibatkan. Pada tugas ini digunakan algoritma brute force untuk menyelesaikan permasalahan di dalam permainan Cyberpunk 2077 Breach Protocol.

Cyberpunk 2077 Breach Protocol adalah minigame meretas pada permainan video Cyberpunk 2077. Di dalam permainan ini diberikan suatu matriks dengan ukuran sembarang yang berisi berbagai token acak. Terdapat beberapa rangkaian token yang memiliki poin. Semua kemungkinan rangkaian token yang memenuhi syarat akan di-enumerasi satu-satu untuk selanjutnya dicari rangkaian token yang memiliki nilai poin yang paling optimal. Tugas pemain adalah mencari rangkaian token yang menghasilkan nilai yang optimal dari berbagai kemungkinan rangkaian yang ada.

Adapun aturan permainan Breach Protocol antara lain:

1. Pemain bergerak dengan pola horizontal, vertikal, horizontal, vertikal (bergantian) 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.

Algoritma brute force pada program ini dimulai dari proses pendataan setiap kemungkinan rangkaian token yang bisa dibuat. Rangkaian diperoleh melalui suatu proses pemilihan selang-seling secara berulang sampai ukuran buffer maksimal atau tidak ada token yang memenuhi. Selanjutnya setiap rangkaian tersebut disimpan dalam suatu array yang menampung setiap kemungkinan yang memenuhi syarat. Setiap rangkaian akan diperiksa jumlah poin yang dimilikinya. Rangkaian terpendek dengan poin tertinggi adalah rangkaian yang paling optimal.

Algoritma yang digunakan ini merupakan algoritma yang sederhana. Algoritma ini mencari semua kemungkinan solusi lalu membandingkannya untuk mendapatkan solusi yang paling optimal. Algoritma ini lebih cocok untuk mengatasi masalah yang memiliki sedikit kemungkinan karena memiliki kompleksitas waktu yang tinggi.

## BAGIAN II

### SOURCE CODE PROGRAM

Program ini dibuat menggunakan Bahasa Python dengan menggunakan library random dan time. Program ini disimpan dalam folder src dengan nama tucil.py. Adapun isi source codenya sebagai berikut.

```
import time
import random

class Token :
    def __init__(self, data, koor) :
        self.data = data
        self.koor = koor

    def __str__(self):
        return f"{self.data}"

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

    def __str__(self) :
        return f"{self.sequence}"

class pohon :
    def __init__(self, data, koor) :
        self.value = Token(data, koor)
        self.children = []
        self.parent = None

    def add_child(self, child) :
        child.parent = self
        self.children.append(child)

    def get_level(self) :
        level = 0
        p = self.parent
        while p:
            level += 1
            p = p.parent
        return level
```

```

def display_tree(self) :
    space = " " * self.get_level() * 2
    print(space + str(self.value.data))
    if self.children :
        for child in self.children :
            child.display_tree()

def isTaken(self, ortu) :
    taken = False
    p = ortu
    while not taken and p :
        if self.value.koor == p.value.koor:
            taken = True
            break
        else :
            p = p.parent
    return taken

def cari_seq(self, idx_brs, idx_kol, vertikal, ctr, matrix) :
    if ctr > 0 :
        if vertikal :
            for i in range(0, len(matrix)) :
                child = pohon(matrix[i][idx_kol].data, (i, idx_kol))
                if not child.isTaken(self) :
                    self.add_child(child)
                    vertikal = False
                    child.cari_seq(i, idx_kol, vertikal, ctr-1,
matrix)
        else :
            for i in range(0, len(matrix[0])) :
                child = pohon(matrix[idx_brs][i].data, (idx_brs, i))
                if not child.isTaken(self) :
                    self.add_child(child)
                    vertikal = True
                    child.cari_seq(idx_brs, i, vertikal, ctr-1, matrix)

def bacaData(namaFile) :
    try :
        with open(namaFile, 'r') as file:
            lines = file.readlines()

        matrix = []
        for line in lines:
            row = [str(num) for num in line.strip().split()]

```

```

        matrix.append(row)

    buffer = int(matrix[0][0])-1
    nbrs = int(matrix[1][0])
    nkol = int(matrix[1][1])
    mtrx = []
    for i in range(2,nbrs+2) :
        baris = matrix[i]
        mtrx.append(baris)
    nseq = int(matrix[nbrs+2][0])
    seq = []
    for i in range(nbrs+3,nbrs+3+nseq+nseq) :
        baris = matrix[i]
        seq.append(baris)

    lseq = makeListSequence(seq)
    return buffer, nbrs, nkol, mtrx, nseq, lseq
except FileNotFoundError :
    print("File tidak ditemukan")
    nama = input("Masukkan Path File yang benar : ")
    buffer, nbrs, nkol, mtrx, nseq, lseq = bacaData(nama)
    return buffer, nbrs, nkol, mtrx, nseq, lseq

def makeMatrixToken(matriks) :
    matrixToken = []
    for i in range(0,len(matriks)) :
        mcol = []
        for j in range(0,len(matriks[0])) :
            col = Token(matriks[i][j],(i,j))
            mcol.append(col)
        matrixToken.append(mcol)
    return matrixToken

def printMatrixToken(matriks) :
    for i in range(0,len(matriks)) :
        for j in range(0,len(matriks[0])) :
            print(matriks[i][j].data, end=" ")
        print()

def makeListSequence(seq) :
    listSequence = []
    for i in range(0,len(seq),2) :
        poin = seq[i+1][0]
        s = Sequence(seq[i], poin)
        listSequence.append(s)

```

```

    return listSequence

def printListSequence(seq) :
    for data in seq :
        print(data.sequence)
        print(data.poin)

def poinBuffer(buffer, lseq):
    total_poin = 0
    for seq in lseq :
        if cocokin_seq(seq,buffer) :
            total_poin = total_poin + int(seq.poin)
    return total_poin

def maks_poin(lbuffer, lseq) :
    maks = 0
    buff_maks = []
    for buffer in lbuffer :
        poin_buffer = poinBuffer(buffer, lseq)
        if poin_buffer > maks :
            maks = poin_buffer
            buff_maks = buffer
    return maks, buff_maks

def cocokin_seq(sequence, token) :
    if len(token) >= len(sequence.sequence) :
        for i in range(0, (len(token)-len(sequence.sequence)+1)) :
            j = 0
            while j < len(sequence.sequence) and sequence.sequence[j] ==
token[i+j].data :
                j = j + 1
            if j == len(sequence.sequence) :
                return True
        return False
    else :
        return False

def tree_to_array(root, path=[]):
    if not root:
        return []
    path = path + [root.value]
    if not root.children:
        return [path]
    paths = []
    for child in root.children:

```

```

        paths.extend(tree_to_array(child, path))

    return paths

def bacaCLI() :
    jumlah_token_unik = int(input("Masukkan Jumlah Token Unik = "))
    token = input("Masukkan Token Unik = ").split(" ")
    ukuran_buffer = int(input("Masukkan Ukuran Buffer = "))
    ukuran_matriks = input("Masukkan Ukuran Matrix = ").split(" ")
    mbaris = int(ukuran_matriks[0])
    mkolom = int(ukuran_matriks[1])
    jumlah_sekuens = int(input("Masukkan Jumlah Sekuens = "))
    ukuran_maksimal_sekuens = int(input("Masukkan Ukuran Maksimal Sekuens = "))

    matrix = ["" for i in range(mkolom)] for j in range(mkolom)]

    for i in range(mbaris) :
        for j in range(mkolom) :
            random_int = random.randint(0, jumlah_token_unik-1)
            matrix[i][j] = token[random_int]

    list_sekuens = []
    for i in range(jumlah_sekuens) :
        random_poin = 5*random.randint(0,20)
        random_nseq = random.randint(2, ukuran_maksimal_sekuens)
        lseq = []
        for j in range(random_nseq) :
            random_int = random.randint(0, jumlah_token_unik-1)
            lseq.append(token[random_int])
        sekuens = Sequence(lseq, random_poin)
        list_sekuens.append(sekuens)

    return ukuran_buffer, mbaris, mkolom, matrix, jumlah_sekuens, list_sekuens

def simpanetxt(namaFile, nilai, sekuens, waktu) :
    token = str('')
    for i in sekuens :
        token = token + str(i.data) + ' '
    koordinat = str('')
    for i in sekuens :
        koordinat = koordinat + str((i.koor[1]+1)) + ' ' + str(i.koor[0]+1) +
'\n'

    with open(namaFile, 'w') as file :
        file.write(str(nilai) + '\n')

```

```

        file.write(token + '\n')
        file.write(koordinat + '\n')
        file.write(str(f"{{(waktu)*10**3:.03f}} ms"))

def mulai() :
    print("Selamat datang di Cyberpunk 2077 Breach Protokol\n")

    print("Pilihan yang tersedia")
    print("1. Menggunakan File txt")
    print("2. Menggunakan Command Line Interface\n")

    pilih = input("Masukkan nomor yang diinginkan\n")
    while pilih != '1' and pilih != '2' :
        print("Masukkan tidak valid")
        pilih = input("Masukkan nomor yang diinginkan\n")

    ukuran_buffer, mbaris, mkolom, matrix, jumlah_sekuens, list_sekuens = int,
int, int, [], int, []
    if pilih == '1' :
        nama = input("Masukkan path menuju file : ")
        ukuran_buffer, mbaris, mkolom, matrix, jumlah_sekuens, list_sekuens =
        bacaData(nama)
    elif pilih == '2' :
        ukuran_buffer, mbaris, mkolom, matrix, jumlah_sekuens, list_sekuens =
        bacaCLI()

    start = time.time()
    mtoken = makeMatrixToken(matrix)
    print()
    printMatrixToken(mtoken)
    print()
    printListSequence(list_sekuens)
    print()

    hasil_akhir = []
    for i in range (1,ukuran_buffer+1):
        for j in range(0,len(mtoken[0])) :
            seq2 = pohon(mtoken[0][j].data,(0,j))
            seq2.cari_seq(0,j,True,i,mtoken)
            result_array = tree_to_array(seq2)
            for k in result_array :
                hasil_akhir.append(k)
    maks, buffer_maks = maks_poin(hasil_akhir,list_sekuens)
    print(maks)
    for i in buffer_maks :

```



```

        print(i.data, end=" ")
    print()
    for i in buffer_maks :
        print((i.koor[1]+1), (i.koor[0]+1))
    end = time.time()
    print(f"\nTime taken: {(end-start)*10**3:.03f} ms\n")

    simpan = input("Apakah ingin menyimpan solusi? (y/n)\n")
    while simpan != 'y' and simpan != 'n' and simpan != 'Y' and simpan != 'N' :
        print('Masukkan tidak valid')
        simpan = input("Apakah ingin menyimpan solusi? (y/n)\n")

    if simpan == 'y' or simpan == 'Y' :
        nama_simpan = input("Masukkan nama file yang ingin disimpan : ")
        simpan_txt(nama_simpan, maks, buffer_maks, end-start)

mulai()

```

## BAGIAN 3

### HASIL UJI KASUS

Input: input1.txt

```
PS C:\Users\ASUS\Documents\Tucil1_13522030> & C:/Users/ASUS/
Documents/Tucil1_13522030/src/tucil.py
Selamat datang di Cyberpunk 2077 Breach Protokol

Pilihan yang tersedia
1. Menggunakan File txt
2. Menggunakan Command Line Interface

Masukkan nomor yang diinginkan
1
Masukkan path menuju file : test/input1.txt

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

['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

Time taken: 1250.078 ms

Apakah ingin menyimpan solusi? (y/n)
y
Masukkan nama file yang ingin disimpan : test/tc1.txt
```

Input: input2.txt

```
PS C:\Users\ASUS\Documents\Tucil1_13522030> python -u
Selamat datang di Cyberpunk 2077 Breach Protokol

Pilihan yang tersedia
1. Menggunakan File txt
2. Menggunakan Command Line Interface

Masukkan nomor yang diinginkan
1
Masukkan path menuju file : test/input2.txt

A1 A2 A3
A4 A5 A6
A7 A8 A9

['A1', 'A5']
20
['A5', 'A9']
15
['A5', 'A3']
10

0

Time taken: 3.906 ms

Apakah ingin menyimpan solusi? (y/n)
y
Masukkan nama file yang ingin disimpan : test/tc2.txt
```

## Input : input3.txt

```
PS C:\Users\ASUS\Documents\Tucill_13522030> python -u
Selamat datang di Cyberpunk 2077 Breach Protokol

Pilihan yang tersedia
1. Menggunakan File txt
2. Menggunakan Command Line Interface

Masukkan nomor yang diinginkan
1
Masukkan path menuju file : test/input3.txt

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

['7A', '1C', 'BD']
20
['BD', '7A']
15
['1C', 'BD']
5

40
7A BD 7A 1C BD
1 1
1 4
2 4
2 3
4 3

Time taken: 44.876 ms

Apakah ingin menyimpan solusi? (y/n)
y
Masukkan nama file yang ingin disimpan : test/tc3.txt
```

## Input : Command Line Interface

```
Selamat datang di Cyberpunk 2077 Breach Protokol

Pilihan yang tersedia
1. Menggunakan File txt
2. Menggunakan Command Line Interface

Masukkan nomor yang diinginkan
2
Masukkan Jumlah Token Unik = 5
Masukkan Token Unik = BD 1C 7A 55 E9
Masukkan Ukuran Buffer = 7
Masukkan Ukuran Matrix = 6 6
Masukkan Jumlah Sekuens = 3
Masukkan Ukuran Maksimal Sekuens = 4

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

['1C', '1C']
20
['BD', '1C', 'E9', 'BD']
30
['E9', 'E9', '7A']
30
```

```

60
BD 1C E9 BD E9 E9 7A
2 1
2 5
6 5
6 4
1 4
1 5
4 5

Time taken: 7979.374 ms
Apakah ingin menyimpan solusi? (y/n)

```

Input : Command Line Interface

```

PS C:\Users\ASUS\Documents\Tucil1_13522030> python
Selamat datang di Cyberpunk 2077 Breach Protokol

Pilihan yang tersedia
1. Menggunakan File txt
2. Menggunakan Command Line Interface

Masukkan nomor yang diinginkan
2
Masukkan Jumlah Token Unik = 7
Masukkan Token Unik = BD 1C E9 7A PW 55 LM
Masukkan Ukuran Buffer = 6
Masukkan Ukuran Matrix = 4 4
Masukkan Jumlah Sekuens = 6
Masukkan Ukuran Maksimal Sekuens = 4

1C E9 E9 1C
LM BD PW 1C
PW E9 BD 1C
55 BD 1C LM

['7A', 'LM', '7A']
95
['LM', '7A', '55', 'E9']
95
['BD', '55', 'LM']
80
['55', '55']
55
['BD', '1C', 'LM', 'PW']
100
['7A', 'BD', '1C']
65

```

```

80
E9 BD 55 LM
2 1
2 4
1 4
1 2

Time taken: 62.769 ms

Apakah ingin menyimpan solusi? (y/n)
Y
Masukkan nama file yang ingin disimpan : test/tc5.txt

```

## Input : Command Line Interface

```
PS C:\Users\ASUS\Documents\Tucil1_13522030> python
Selamat datang di Cyberpunk 2077 Breach Protokol

Pilihan yang tersedia
1. Menggunakan File txt
2. Menggunakan Command Line Interface

Masukkan nomor yang diinginkan
2
Masukkan Jumlah Token Unik = 7
Masukkan Token Unik = BD 1C E9 7A PW 55 LM
Masukkan Ukuran Buffer = 8
Masukkan Ukuran Matrix = 7 7
Masukkan Jumlah Sekuens = 5
Masukkan Ukuran Maksimal Sekuens = 6

55 BD 7A PW E9 7A 7A
1C LM 7A 55 7A BD LM
BD 1C 1C 55 55 LM 1C
BD E9 E9 BD BD E9 1C
1C 1C 7A E9 1C 7A BD
7A E9 PW 7A 55 LM 55
1C LM PW 7A 7A E9 55

['1C', 'E9', '55', 'PW']
55
['LM', 'LM', 'E9']
50
['E9', 'PW', 'BD']
70
['55', 'E9', 'LM', '7A']
70
['BD', 'E9', '55', 'LM', '1C', 'BD']
10
```

```
140
7A 55 E9 LM 7A E9 PW BD
7 1
7 6
2 6
2 2
5 2
5 1
4 1
4 4

Time taken: 230857.349 ms

Apakah ingin menyimpan solusi? (y/n)
y
Masukkan nama file yang ingin disimpan : test/tc6.txt
```

## LINK REPOSITORY

[https://github.com/HanifIHM/Tucil1\\_13522030](https://github.com/HanifIHM/Tucil1_13522030)

## LAMPIRAN

Poin	Ya	Tidak
1. Program dapat berhasil dikompilasi tanpa kesalahan	V	
2. Program berhasil dijalankan	V	
3. Program dapat membaca masukan berkas .txt	V	
4. Program dapat menghasilkan masukan secara acak	V	
5. Solusi yang diberikan program optimal	V	
6. Program dapat menyimpan Solusi dalam berkas .txt	V	
7. Program memiliki GUI		V