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)

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 yaitu *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

```
def main():
    try:
        buffer = Buffer()
        optimal_buffer = Buffer()
        BPmatrix = BreachMatrix()
        sequence = Sequence()

        print("===================== Input Methods =====================")
        print("| 1. Input from command line                           |")
        print("| 2. Input from file in test folder                    |")
        print("========================================================\n")

        selection = int(input("> Choose the Input method : "))

        while selection != 1 and selection != 2:
            selection = int(input("> Select a valid number! : "))

        print()
        if selection == 1:
            CLI_Input(buffer,BPmatrix,sequence)
        else:
            external_fileI(buffer,BPmatrix,sequence)

        optimal_buffer.buffer_size = buffer.buffer_size

        starttime = time()

        for i in range(BPmatrix.col):
            breachprotocol(buffer, BPmatrix, sequence, 0, i, optimal_buffer)


        endtime = time()

        time_used = round((endtime - starttime) * 1000)

        print("Below is the matrix!")
```

```
        BPmatrix.print_matrix()

        print("\nBelow are the sequences!")
        sequence.print_sequence()

        print("\nBelow is the most optimal buffer found!")
        optimal_buffer.print_buffer()

        print()
        print(time_used, "ms")
        print()

        print("Apakah ingin menyimpan solusi? (y/n)")
        selection = input("> ")

        while selection != "y" and selection != "Y" and selection != "n" and
selection != "N":
            selection = input("> ")

        if selection == "Y" or selection == "y":
            savingfile(optimal_buffer,time_used)


    except FileNotFoundError as fnfe:
        print("File not found or path is incorrect.", fnfe)

    except NotCorrectColumnCountError as nccce:
        print("Error in matrix formatting:", nccce)

    except NotEnoughTokens as net:
        print("Error in tokenization:", net)

    except Exception as e:
        print("An error occurred:", e)

    print("\nExitting program . . .")

if __name__ == '__main__':
    main()
```

Bagian paling terakhir adalah main, berfungsi untuk menjalankan seluruh program. Dimulai dari opsi memilih cara input, kemudian menunjukkan matriks, sekuens, dan buffer optimal, lalu memberikan prompt untuk mempertanyakan untuk melakukan simpan. Pada bagian evaluasi buffer, kolom pertama diatur pada bagian ini karena prosedur breach protocol sebelumnya menyelesaikan dengan awalan yang ditentukan dari baris paling pertama. Pada bagian ini terdapat pula semua exception yang digunakan.

Uji Coba

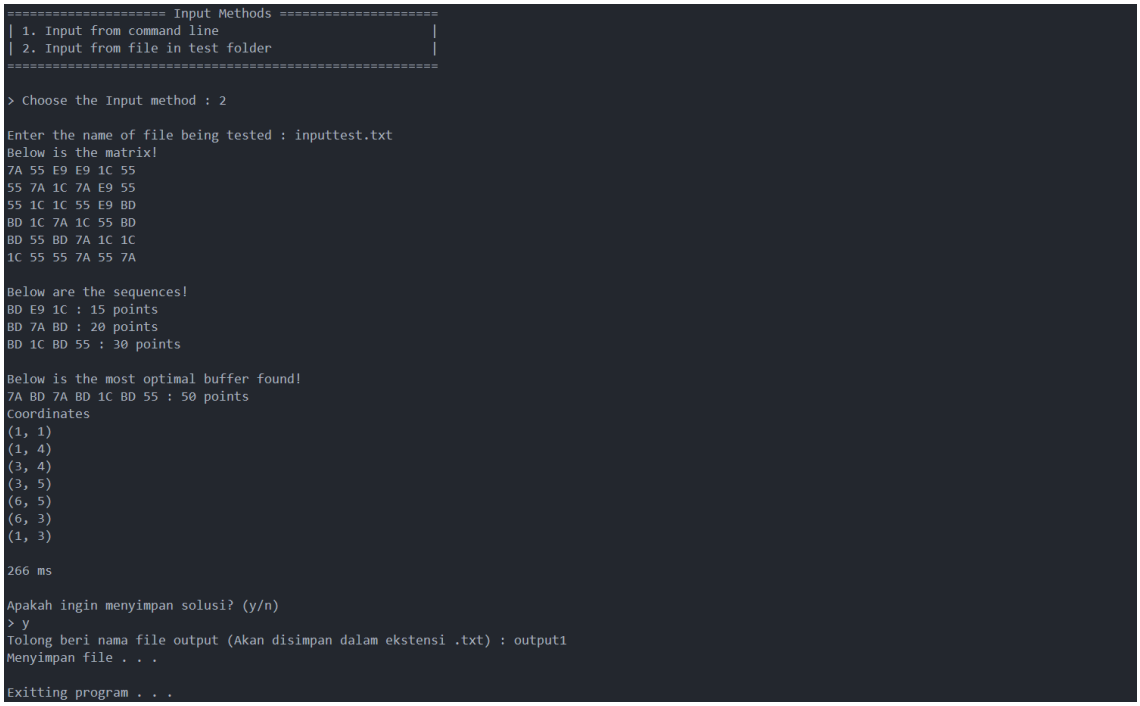

Fig. 1 contoh terminal file eksternal

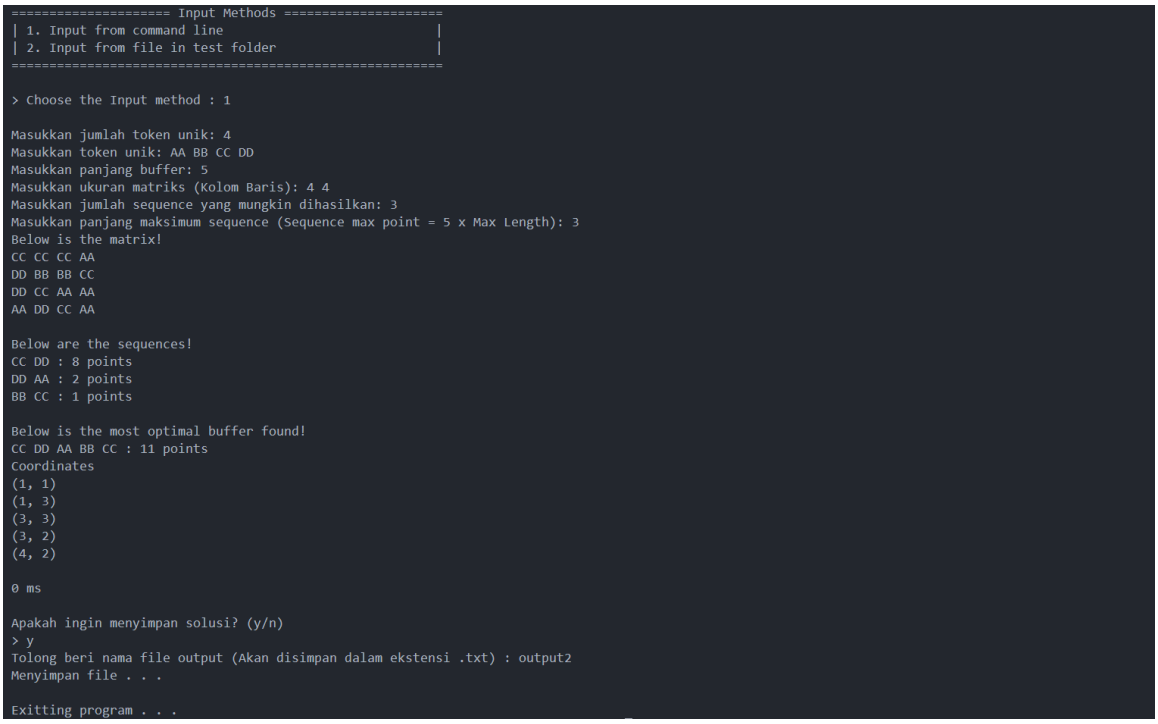

Fig. 2 contoh terminal command line input

Data Uji Coba

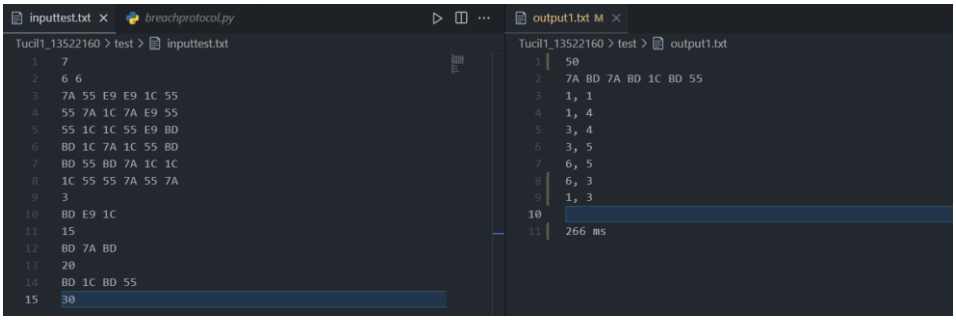

Fig. 3 input-output dari contoh di spesifikasi

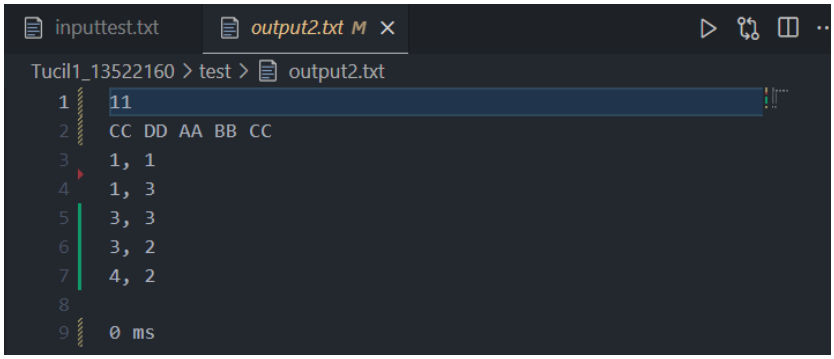

Fig. 4 output dari CLI Fig. 2

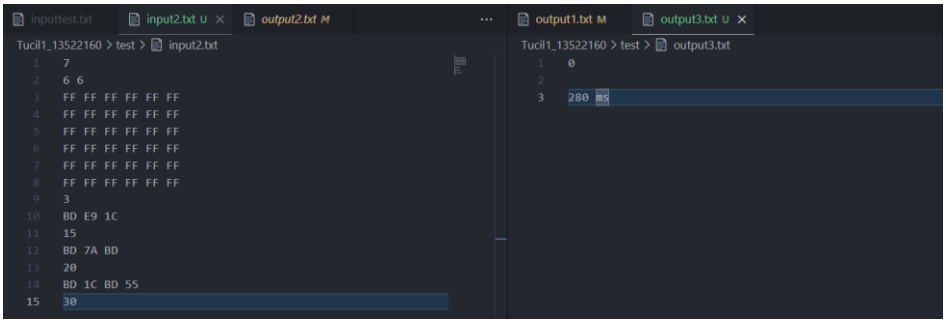

Fig. 5 input-output jika tidak menemukan sekuens apapun

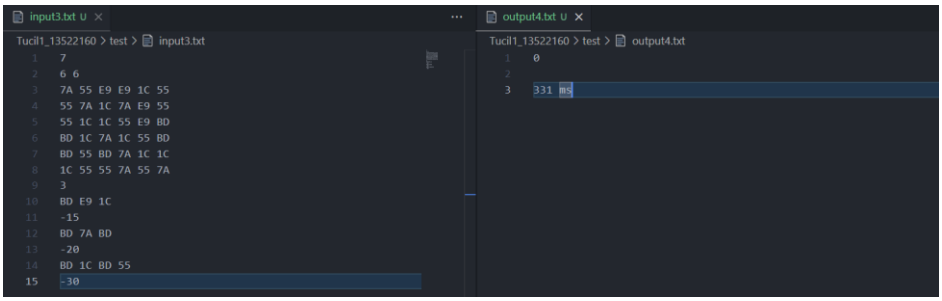

Fig. 6 input-output jika tidak menemukan poin yang lebih tinggi dari 0

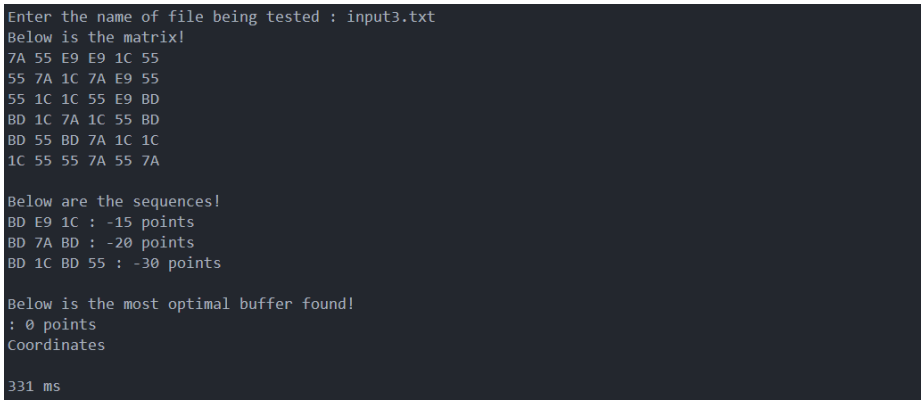

Fig. 7 terminal untuk Fig. 6

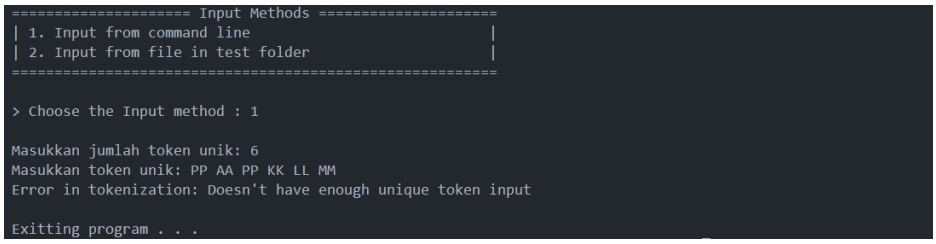

Fig. 8 contoh CLI gagal karena terdapat dua PP (duplikat) sehingga tidak sesuai

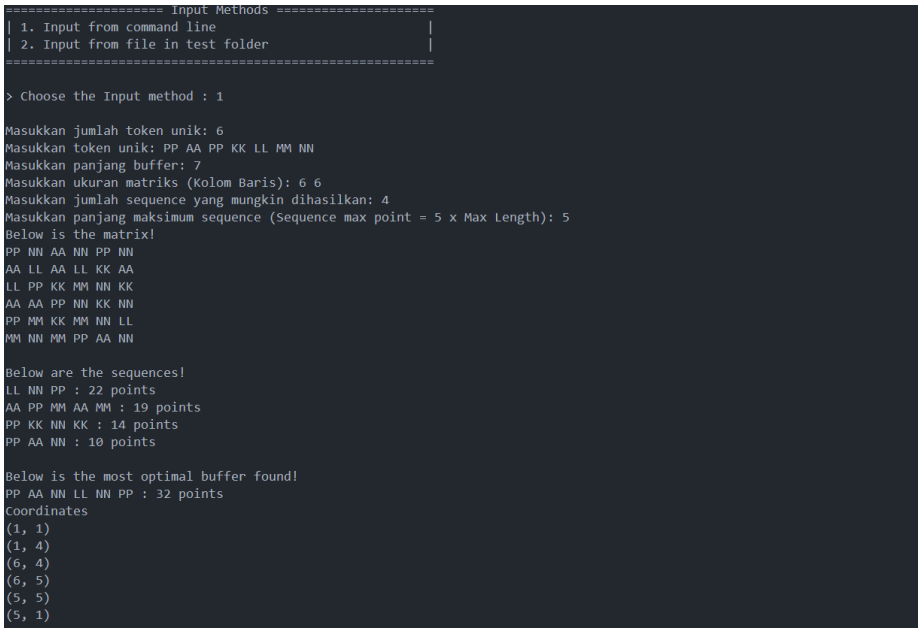

Fig. 9 contoh CLI 2 di terminal

Fig. 10 contoh CLI 3 (ekstrem)

Lampiran

Poin	Ya	Tidak
1. Program 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 optima	V	
6. Program dapat menyimpan solusi dalam berkas .txt	V	
7. Program memiliki GUI		V

Pranala repository GitHub:
https://github.com/Rinnearu/Tucil1_13522160