

**LAPORAN TUGAS KECIL 01**  
**IF2211 STRATEGI ALGORITMA**  
**“Penyelesaian *Cyberpunk 2077 Breach Protocol* dengan Algoritma Brute Force”**



Disusun oleh :  
Venantius Sean Ardi Nugroho K-02 13522078

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**  
**INSTITUT TEKNOLOGI BANDUNG**  
**2024**

## **DAFTAR ISI**

### **DAFTAR ISI**

#### **BAB 1 DESKRIPSI MASALAH**

#### **BAB 2 TEORI SINGKAT**

#### **BAB 3 IMPLEMENTASI PROGRAM**

#### **BAB 4 EKSPERIMEN**

#### **LAMPIRAN**

#### **DAFTAR REFERENSI**

## BAB 1

### DESKRIPSI MASALAH

Pada game *Cyberpunk 2077* yang diciptakan oleh studio *CD Projekt Red* terdapat game yang terinspirasi oleh kegiatan meretas yaitu *Breach Protocol*. Objektif dari game tersebut adalah menemukan kombinasi sekuens yang paling optimal sehingga pemain bisa mendapatkan poin yang terbanyak. Untuk mendapatkan poin, sekuens yang dibuat oleh pemain harus mengandung kombinasi sekuens yang disediakan oleh game. Secara garis besar, berikut adalah komponen – komponen yang terdapat pada game tersebut :

- Token : token adalah kombinasi antara dua karakter alfanumerik yang membangun sebuah sekuens dan matriks
- Matriks : terdiri dari  $n * m$  jumlah token yang akan dipilih untuk menyusun urutan kode
- Sekuens: sebuah rangkaian token (dua atau lebih) yang harus dicocokkan.
- Buffer : jumlah maksimal token yang dapat disusun secara sekuensial.

Untuk membuat kombinasi, pemain harus memilih salah satu token pada posisi baris paling atas dari matriks , lalu bergerak dengan pola vertikal, horizontal, vertikal (bergantian , pasti dimulai dengan vertikal) hingga semua sekuens berhasil dicocokkan atau buffer penuh.

## BAB 2

### TEORI DAN IDE SOLUSI

Algoritma yang digunakan untuk menemukan sekuens yang optimal dalam permainan peretasan *Cyberpunk 2077* adalah dengan metode *Brute Force*. Algoritma *Brute Force* adalah algoritma yang sederhana dalam menyelesaikan suatu permasalahan, lebih spesifiknya adalah dengan cara mencari semua opsi yang mungkin sampai suatu solusi ditemukan. Ciri – ciri dari algoritma ini adalah algoritma *Brute Force* sangat bergantung pada kekuatan komputasi.

Untuk mengerti bagaimana algoritma *Brute Force* digunakan dalam menyelesaikan masalah yang diberikan. Alangkah baiknya kita membahas bagaimana alur dari program ini. Berikut merupakan solusi yang saya usulkan untuk memecahkan masalah ini :

1. Meminta data – data yang diperlukan.
2. Menemukan semua kombinasi sekuens sepanjang *buffer* yang dapat dibuat dari matrix lalu menyimpannya dalam sebuah array.
3. Untuk setiap elemen dalam array buffer, akan dicari berapa poin yang didapat dengan cara mengecek apakah suatu sekuens (input) terdapat pada buffer.
4. Masukkan semua point ke dalam array.
5. Temukan point maksimal dalam array tersebut dan ambil indeksinya.
6. Gunakan indeks tersebut untuk menemukan sekuens buffer dengan poin paling banyak.

Dalam alur tersebut, algoritma brute force paling berperan dalam step ke 2 yaitu menemukan semua kombinasi (enumerasi) sekuens yang sepanjang buffer dari matriks. Lebih tepatnya, algoritma *Brute Force* yang digunakan untuk mengenumerasi sekuens adalah algoritma *Backtrack*. Algoritma *Back Track* berguna dalam mencari semua solusi dalam suatu permasalahan, biasanya terdapat suatu *constraint* dimana solusi yang tidak memenuhi syaratnya akan dibuang. Algoritma *Back Track* yang digunakan menggunakan prinsip *Depth First Search* (DFS). Pada pembuatan sekuens, *constraint* yang perlu dipertimbangkan antara lain adalah :

1. Sekuens tidak boleh memiliki panjang lebih dari *buffer*.
2. Token yang sudah dipakai tidak bisa dipakai lagi.
3. *Sekuens* hanya boleh dibuat dengan arah vertikal – horizontal – vertikal , perlu diingat bahwa gerakan pertama pasti vertikal.
4. Token yang pertama dipilih hanya boleh dari baris paling atas.

Untuk memastikan agar token yang dipakai tidak dipakai tidak digunakan kembali, dibuat sebuah matriks boolean yang dimensinya sama dengan matriks tokennya. Pada implementasi yang saya gunakan, program akan mencoba untuk membuat semua sekuens dengan panjang =  $2 \leq n < \text{buffer}$ , sebelum membuat sekuens dengan panjang  $2 \leq n < \text{buffer}$ .

## BAB 3

### IMPLEMENTASI PROGRAM

#### 3.1 *Import statements dan classes*

Pada program tucil ini, digunakan library random pada pembuatan sekuens yang digunakan pada opsi input dengan CLI. Lebih spesifiknya, random digunakan untuk menentukan token yang membuat sekuens tersebut serta poin yang di assign pada tiap sekuens. Library time digunakan untuk menghitung *execution time*.

Dalam pembuatan tugas ini, dibuat class bernama Token dan Matrix. Class Token dibuat untuk memudahkan penyimpanan posisi dari tiap token. Class Matrix dibuat untuk meningkatkan *readability*.

A screenshot of a code editor with a dark background and light-colored text. The code defines two classes: Token and Matrix. The Token class has an \_\_init\_\_ method that takes symbol, x, and y as arguments and assigns them to self.symbol, self.x, and self.y respectively. It also has a \_\_str\_\_ method that returns self.symbol. The Matrix class has an \_\_init\_\_ method that takes rows and cols as arguments and assigns them to self.rows and self.cols. It also initializes self.data as a list of lists, where each inner list represents a row of zeros. The \_\_str\_\_ method for Matrix returns a string representation of the matrix, with rows separated by newlines and elements separated by spaces.

```
import random
import time

class Token:
    def __init__(self, symbol:str,x:int,y:int):
        self.symbol = symbol
        self.x = x
        self.y = y

    def __str__(self):
        return self.symbol

class Matrix:
    def __init__(self, rows:int, cols:int):
        self.rows = rows
        self.cols = cols
        self.data = [[0 for _ in range(cols)] for _ in range(rows)]

    def __str__(self):
        return '\n'.join([' '.join(map(str, row)) for row in
self.data])
```

Gambar 1. *Import Statement dan Class* yang dibuat

### 3.2 Algoritma enumerasi

Berikut adalah kode enumerasi sekuens dengan cara yang telah dijelaskan pada bab 2.

```
def enumerate_sequence(matrix: Matrix, n:int):
    def backtrack(path, visited, last_i, last_j, last_movement):
        if len(path) == n:
            sequence.append(path)
            return
        last_char = path[-1] if path else None
        for i in range(len(matrix.data)):
            for j in range(len(matrix.data[0])):
                if matrix.data[i][j] != last_char and not visited[i][j]:
                    if last_i != -1 and last_movement == "vertical" and i != last_i:
                        continue
                    if last_j != -1 and last_movement == "horizontal" and j != last_j:
                        continue
                    visited[i][j] = True
                    next_movement = "vertical" if last_movement == "horizontal" else "horizontal"
                    backtrack(path + [matrix.data[i][j]], visited, i, j, next_movement)
                    visited[i][j] = False

    sequence = []
    while n >= 2:
        visited = [[False for _ in range(len(matrix.data[0]))] for _ in range(len(matrix.data))]
        for j in range(len(matrix.data[0])):
            visited[0][j] = True
            backtrack([matrix.data[0][j]], visited, 0, j, "horizontal")
            visited[0][j] = False
        n -= 1
    return sequence
```

Gambar 2. Fungsi “enumerate\_sequence”

### 3.3 Algoritma Membandingkan Sekuens

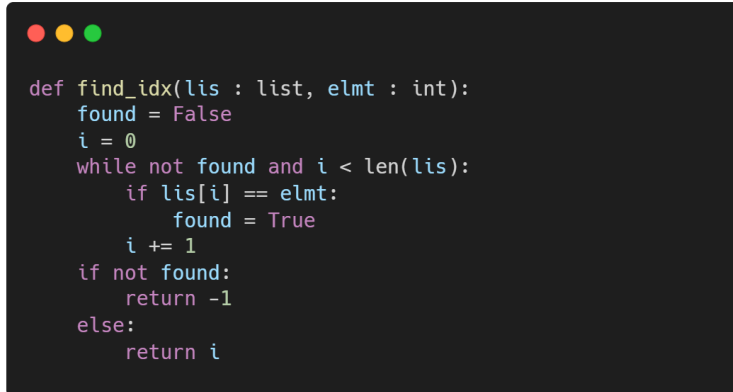
Fungsi di bawah ini digunakan untuk menentukan apakah suatu sekuens terdapat dalam suatu sekuens lainnya.

```
def compare_sequence(compared, reference):
    length_compared = len(compared)
    length_reference = len(reference)
    if length_reference > length_compared:
        return False
    else:
        c = 0
        r = 0
        while c < length_compared:
            if compared[c].symbol == reference[r]:
                r += 1
                if r == length_reference:
                    return True
            else:
                r = 0
                c += 1
        return False
```

Gambar 3. Fungsi “compare\_sequence”

### 3.4 Fungsi *find\_idx*

Fungsi ini digunakan menemukan index suatu elemen dalam sebuah array.

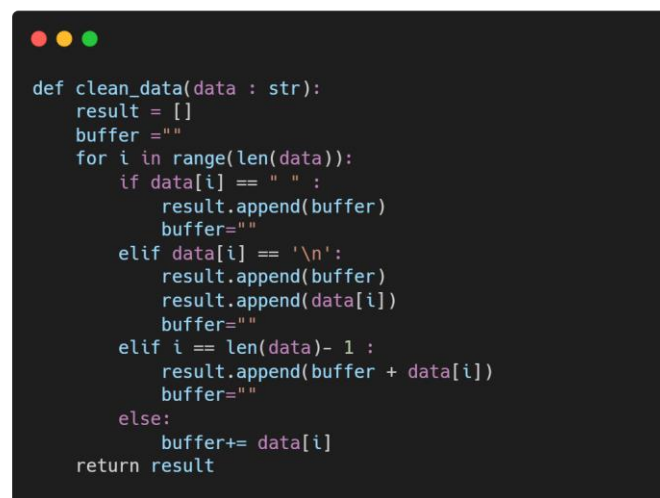
A screenshot of a code editor with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The code defines a function `find_idx` that takes a list `lis` and an integer `elmt` as arguments. It initializes `found` to `False` and `i` to `0`. A `while` loop runs as long as `not found` and `i < len(lis)`. Inside the loop, if `lis[i] == elmt`, `found` is set to `True` and `i` is incremented by 1. After the loop, if `not found`, it returns `-1`; otherwise, it returns `i`.

```
def find_idx(lis : list, elmt : int):  
    found = False  
    i = 0  
    while not found and i < len(lis):  
        if lis[i] == elmt:  
            found = True  
            i += 1  
    if not found:  
        return -1  
    else:  
        return i
```

Gambar 4. Fungsi “find\_idx”

### 3.5 Fungsi *clean\_data*

Fungsi di bawah ini membantu kita dalam mengubah string menjadi elemen – elemen pada sebuah array. Digunakan dalam mengubah inputan txt menjadi array supaya *development* lebih gampang.

A screenshot of a code editor with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The code defines a function `clean_data` that takes a string `data` as an argument. It initializes `result` to an empty list and `buffer` to an empty string. A `for` loop iterates over each character in `data`. If the character is a space, the current `buffer` is appended to `result` and `buffer` is reset. If the character is a newline, the current `buffer` and the newline character are appended to `result`, and `buffer` is reset. If the character is the last one in the string, the current `buffer` and the character are appended to `result`, and `buffer` is reset. Otherwise, the character is simply added to the `buffer`. Finally, the `result` list is returned.

```
def clean_data(data : str):  
    result = []  
    buffer = ""  
    for i in range(len(data)):  
        if data[i] == " "  
            result.append(buffer)  
            buffer=""  
        elif data[i] == '\n':  
            result.append(buffer)  
            result.append(data[i])  
            buffer=""  
        elif i == len(data)- 1 :  
            result.append(buffer + data[i])  
            buffer=""  
        else:  
            buffer+= data[i]  
    return result
```

Gambar 5 Fungsi “clean\_data”

### 3.6 Kamus Global

Kumpulan deklarasi variabel yang digunakan secara global dalam main.

A screenshot of a code editor with a dark background and light-colored text. The code is written in a Python-like syntax and represents global variable declarations. It starts with a comment line '#kamus\_global' followed by several lines of variable assignments. The variables include 'jumlah\_token\_unik', 'token\_selection', 'buffer\_size', 'matrix\_width', 'matrix\_height', 'matrix', 'num\_of\_sequence', 'array\_of\_points', 'array\_of\_sequence', 'sequence\_combination', 'result\_point\_array', 'max\_points', 'idx\_of\_max', 'seq\_result', 'result\_pos', and 'executionTime'. The values assigned are mostly -1, [], 0, or Matrix(0,0).

```
#kamus_global
jumlah_token_unik = -1
token_selection = []
buffer_size = -1
matrix_width = -1
matrix_height = -1
matrix = Matrix(0,0)
num_of_sequence = -1
array_of_points = []
array_of_sequence = []
sequence_combination = []
result_point_array = []
max_points = 0
idx_of_max = -1
seq_result = []
result_pos = []
executionTime = 0
```

Gambar 6 Kamus Global



## 3.7 Main

### 3.7.1 Masukkan Data

Terdapat dua cara untuk memasukkan data dalam program, yaitu menggunakan .txt file dan juga melalui inputan CLI. Bila Anda memilih inputan CLI, maka sekuens dan poin dari sekuens tersebut akan dibuat secara random. Bila Anda memilih menggunakan .txt file, pastikan Anda sudah mengikuti format yang ada pada spek.

```
#main
option = input("How would you want to input the data ? \n 1. file \n 2. command line \n >> ")
while option not in ['1','2']:
    print("Invalid option, please only write 1 or 2")
    option = input("How would you want to input the data ? \n 1. file \n 2. command line \n >> ")

if option == '1':
    file_open=input(str("Apa nama file yang ingin dibaca ? "))
    data = open("input/"+file_open)
    data_contents = clean_data(data.read())
    buffer_size= int(data_contents[0])
    matrix_width = int(data_contents[2])
    matrix_height = int(data_contents[3])
    matrix = Matrix(matrix_height,matrix_width)
    i = 0
    n_ent = 0
    curr_col = 1
    curr_row = 1
    #fill matrix
    while n_ent < 2+matrix_height:
        if n_ent < 2:
            if data_contents[i]== '\n':
                n_ent +=1
        else:
            if data_contents[i]== '\n':
                curr_col = 1
                n_ent +=1
                curr_row +=1
            else:
                temp = Token(data_contents[i],curr_col,curr_row)
                matrix.data[curr_row-1][curr_col-1] = temp
                curr_col +=1
        i +=1
    array_of_sequence = []
    array_of_points = []
    num_of_sequence = data_contents[i]
    i +=2
    isPoint = False
    seq_buf = []
    while i < len(data_contents):
        if not isPoint:
            if data_contents[i] == '\n':
                array_of_sequence.append(seq_buf)
                seq_buf = []
                isPoint = not isPoint
            else:
                seq_buf.append(data_contents[i])
        else:
            if data_contents[i] != '\n':
                array_of_points.append(int(data_contents[i]))
            else:
                isPoint = not isPoint
        i+=1
    data.close()
```

```

else:
    jumlah_token_unik = int(input("Masukkan jumlah token yang ingin diinput: "))
    input_string= input("Masukkan token - token unik tersebut dengan tiap token dipisah dengan spasi (e.g.
BD 1C 7A ): ")
    token_selection = input_string.split()
    if jumlah_token_unik != len(token_selection):
        print("Jumlah token unik tidak sama dengan jumlah token unik yang diinput")
    else:
        buffer_size= int(input("Masukkan besar buffer : "))
        matrix_size = input("Masukkan dimensi matrix (e.g. 6 6) : ").split()
        matrix_width= int(matrix_size[0])
        matrix_height = int(matrix_size[1])
        matrix = Matrix(matrix_height,matrix_width)
        for i in range(matrix_height):
            for j in range(matrix_width):
                temp = Token(random.choice(token_selection),j+1,i+1)
                matrix.data[i][j] = temp
        print("Membuat matrix secara acak ... ")
        time.sleep(2)
        print("Inilah matrix yang digunakan: ")
        print(matrix)
        num_of_sequence = int(input("Masukkan berapa jumlah sequence yang akan dibuat : "))
        max_sequence_size = int(input("Masukkan panjang maksimal sequence tersebut : "))
        for i in range(num_of_sequence):
            array_of_points.append(random.randint(10,100))
            temp = []
            for j in range(random.randint(2,max_sequence_size)):
                temp.append(random.choice(token_selection))
            array_of_sequence.append(temp)
        print("Membuat sequence secara acak ... ")
        time.sleep(2)
        for i in range(len(array_of_sequence)):
            print(array_of_sequence[i], "@", array_of_points[i], "Points")

```

Gambar 7. Penginputan Data pada Program

### 3.7.2 Alur Utama

```
start = time.time()
sequence_combination = enumerate_sequence(matrix,buffer_size)
if sequence_combination == []:
    print("Tidak ada sekuens yang bisa digenerate ... ")
    out= input(("Ketik apapun untuk keluar dari program ... "))
else:
    for i in range(len(sequence_combination)):
        points = 0
        for j in range(len(array_of_sequence)):
            if compare_sequence(sequence_combination[i],array_of_sequence[j]):
                points += array_of_points[j]
        result_point_array.append(points)

    max_points = max(result_point_array)
    idx_of_max = find_idx(result_point_array,max_points)

    if idx_of_max != -1:
        result_temp = sequence_combination[idx_of_max]
        for i in range(len(result_temp)):
            seq_result.append(result_temp[i].symbol)
            result_pos.append([result_temp[i].x,result_temp[i].y])
    end = time.time()
    executionTime = (end - start)*1000

    print("Poin maksimal yang bisa didapat adalah :", max_points)
    print('\n')
    print("Dengan sequence sebagai berikut :",seq_result)
    print('\n')
    print("Berikut adalah koordinat tiap simbol pada sequence tersebut : ")
    for i in range(len(result_pos)):
        print(result_pos[i])

    print("execution time :",executionTime,"ms")
```

```
print("Poin maksimal yang bisa didapat adalah :", max_points)
print('\n')
print("Dengan sequence sebagai berikut :",seq_result)
print('\n')
print("Berikut adalah koordinat tiap simbol pada sequence tersebut : ")
for i in range(len(result_pos)):
    print(result_pos[i])

print("execution time :",executionTime,"ms")

option_save = str(input("Apakah Anda ingin menyimpan hasil program ? (y/n) "))
while option_save not in ["y","n"]:
    print("Bukan pilihan valid, tolong ketik hanya y untuk yes atau n untuk no !")
    option_save=str(input("Apakah Anda ingin menyimpan hasil program ? (y/n) "))
if option_save == "y":
    file_name = str(input("Apa nama filenya ? "))
    save = open("../test/"+file_name,'w')
    save.write(str(max_points))
    save.write('\n')
    save.write(str(seq_result))
    save.write('\n')
    for i in range(len(result_pos)):
        save.write(str(result_pos[i]))
        save.write('\n')
    save.write(str(executionTime))
    save.close()
out= input(("Ketik apapun untuk keluar dari program ... "))
```

Gambar 8. Snippet Alur Utama

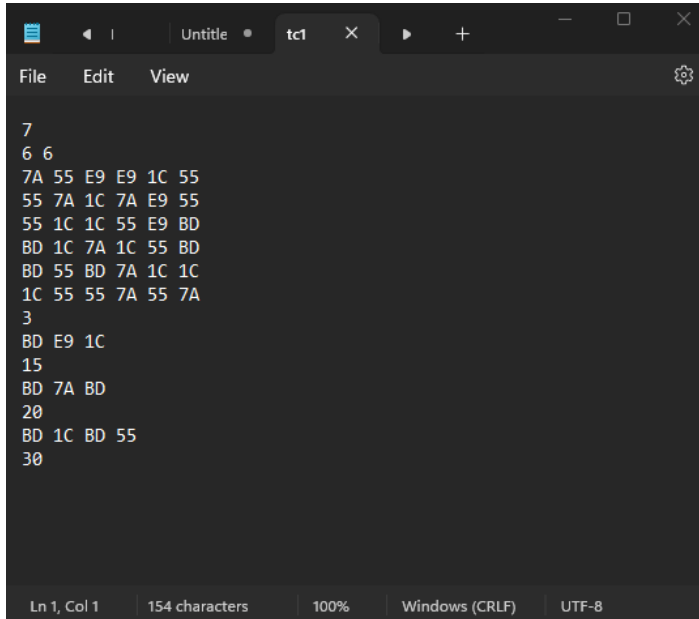
## BAB 4

### EKSPERIMEN

#### 4.1 Dengan Input File .txt

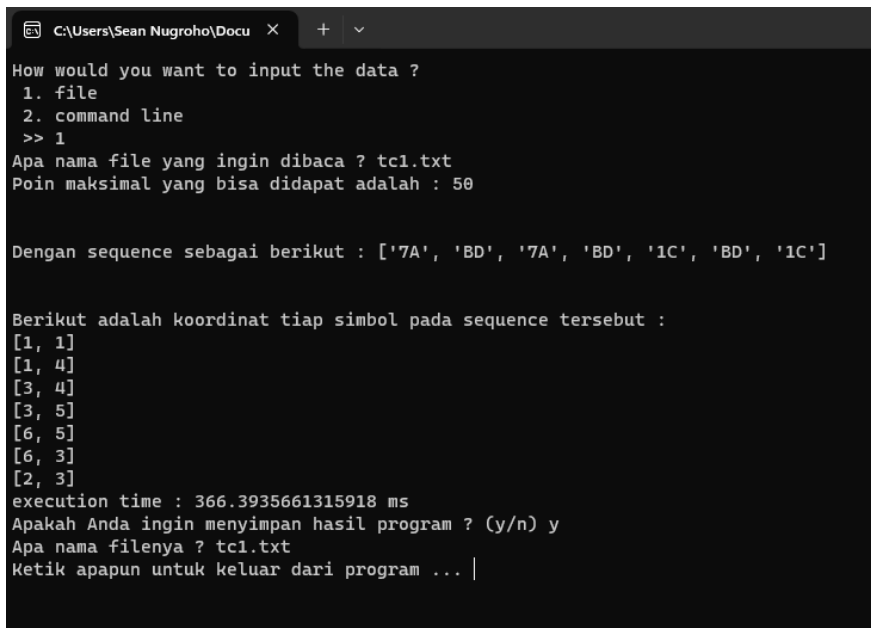
##### 1. Test case 1

Input:



```
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
```

Output:



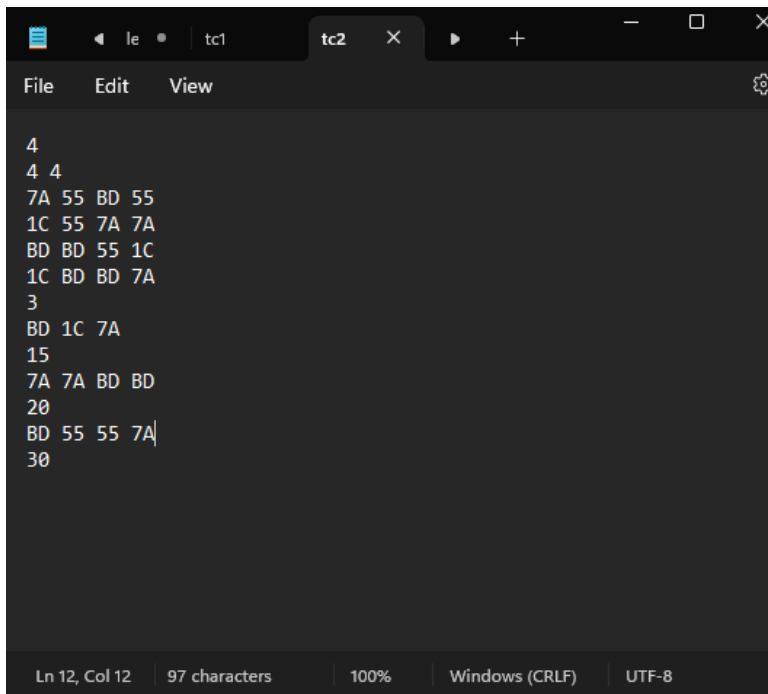
```
C:\Users\Sean Nugroho\Docu x + v
How would you want to input the data ?
1. file
2. command line
>> 1
Apa nama file yang ingin dibaca ? tc1.txt
Poin maksimal yang bisa didapat adalah : 50

Dengan sequence sebagai berikut : ['7A', 'BD', '7A', 'BD', '1C', 'BD', '1C']

Berikut adalah koordinat tiap simbol pada sequence tersebut :
[1, 1]
[1, 4]
[3, 4]
[3, 5]
[6, 5]
[6, 3]
[2, 3]
execution time : 366.3935661315918 ms
Apakah Anda ingin menyimpan hasil program ? (y/n) y
Apa nama filenya ? tc1.txt
Ketik apapun untuk keluar dari program ... |
```

## 2. Test case 2

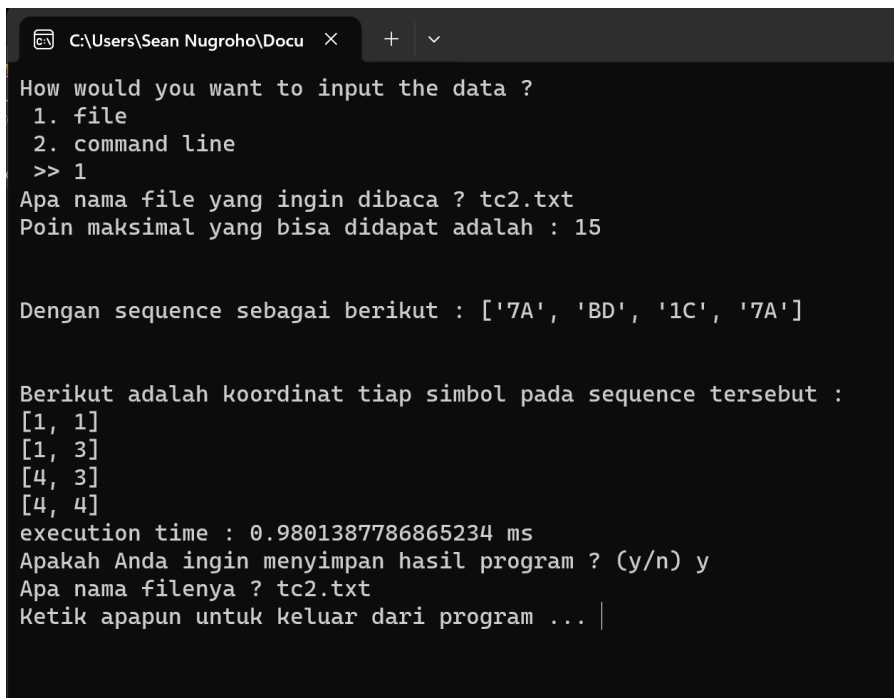
Input :



```
4
4 4
7A 55 BD 55
1C 55 7A 7A
BD BD 55 1C
1C BD BD 7A
3
BD 1C 7A
15
7A 7A BD BD
20
BD 55 55 7A
30
```

Ln 12, Col 12 | 97 characters | 100% | Windows (CRLF) | UTF-8

Output :



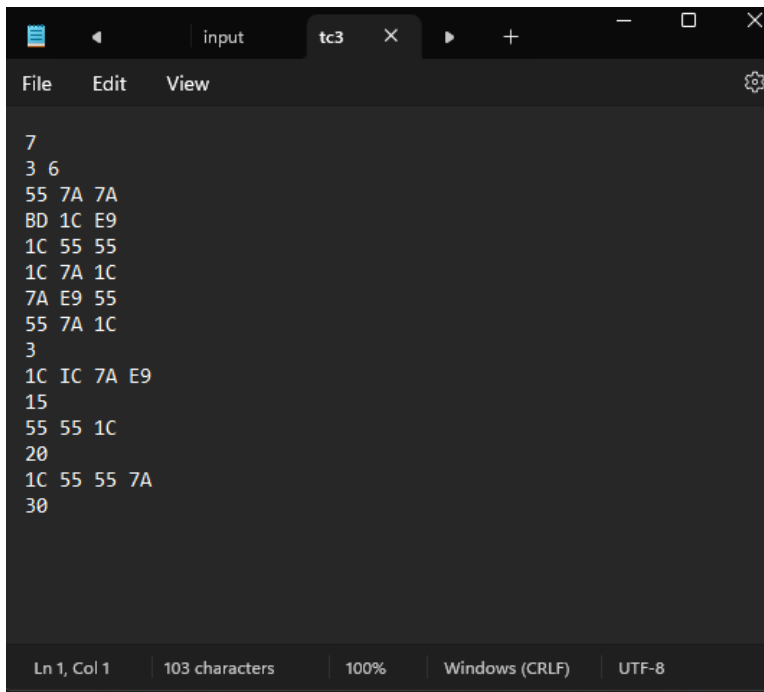
```
C:\Users\Sean Nugroho\Docu x + v
How would you want to input the data ?
1. file
2. command line
>> 1
Apa nama file yang ingin dibaca ? tc2.txt
Poin maksimal yang bisa didapat adalah : 15

Dengan sequence sebagai berikut : ['7A', 'BD', '1C', '7A']

Berikut adalah koordinat tiap simbol pada sequence tersebut :
[1, 1]
[1, 3]
[4, 3]
[4, 4]
execution time : 0.9801387786865234 ms
Apakah Anda ingin menyimpan hasil program ? (y/n) y
Apa nama filenya ? tc2.txt
Ketik apapun untuk keluar dari program ... |
```

### Test case 3

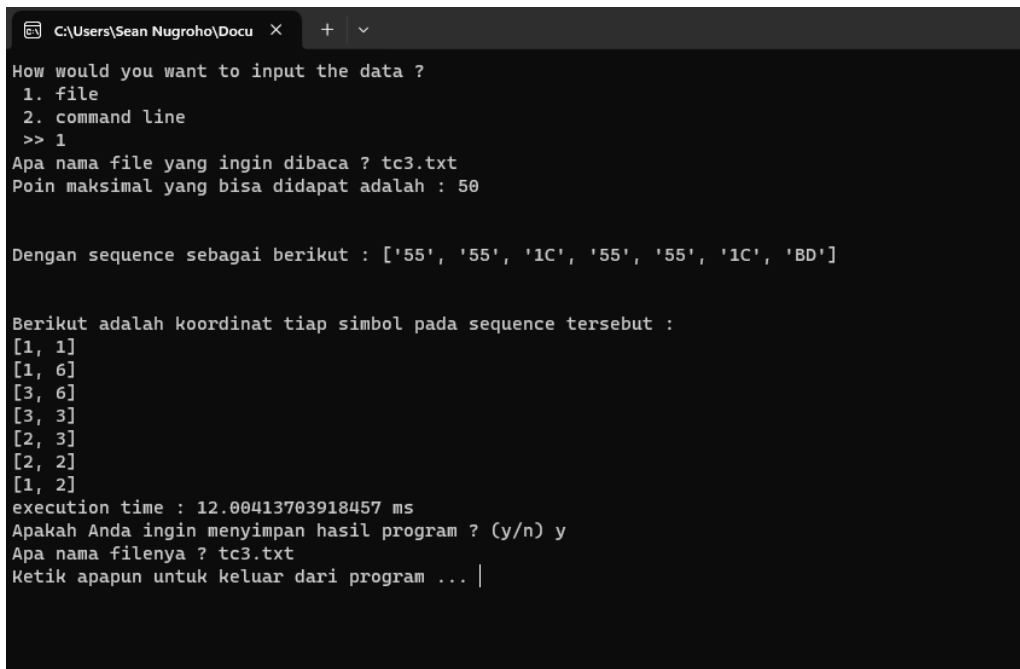
Input :



```
7
3 6
55 7A 7A
BD 1C E9
1C 55 55
1C 7A 1C
7A E9 55
55 7A 1C
3
1C 1C 7A E9
15
55 55 1C
20
1C 55 55 7A
30
```

Ln 1, Col 1 | 103 characters | 100% | Windows (CRLF) | UTF-8

Output:



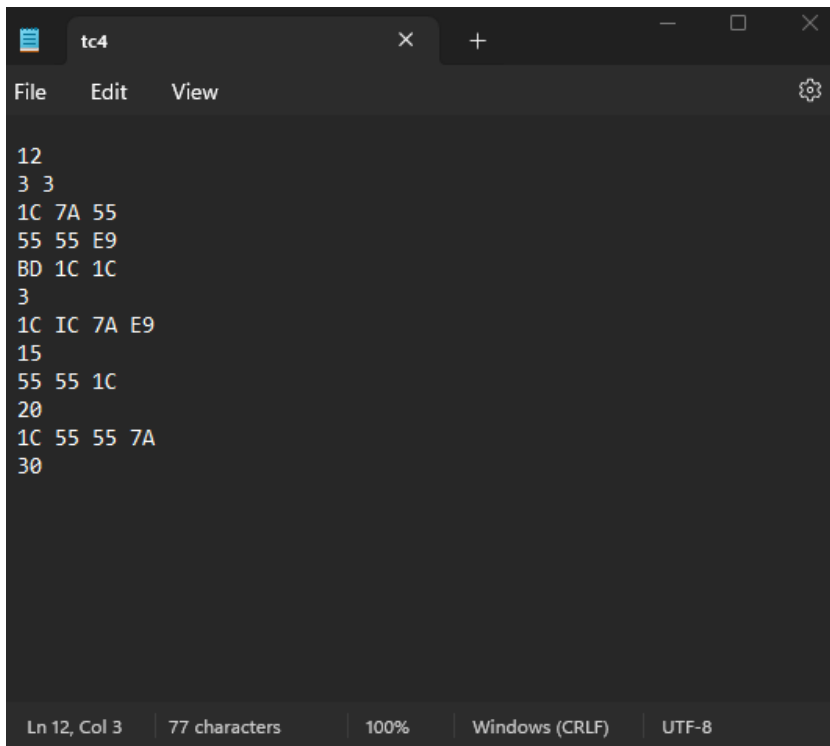
```
C:\Users\Sean Nugroho\Docu X + v
How would you want to input the data ?
1. file
2. command line
>> 1
Apa nama file yang ingin dibaca ? tc3.txt
Poin maksimal yang bisa didapat adalah : 50

Dengan sequence sebagai berikut : ['55', '55', '1C', '55', '55', '1C', 'BD']

Berikut adalah koordinat tiap simbol pada sequence tersebut :
[1, 1]
[1, 6]
[3, 6]
[3, 3]
[2, 3]
[2, 2]
[1, 2]
execution time : 12.00413703918457 ms
Apakah Anda ingin menyimpan hasil program ? (y/n) y
Apa nama filenya ? tc3.txt
Ketik apapun untuk keluar dari program ... |
```

## Test Case 4

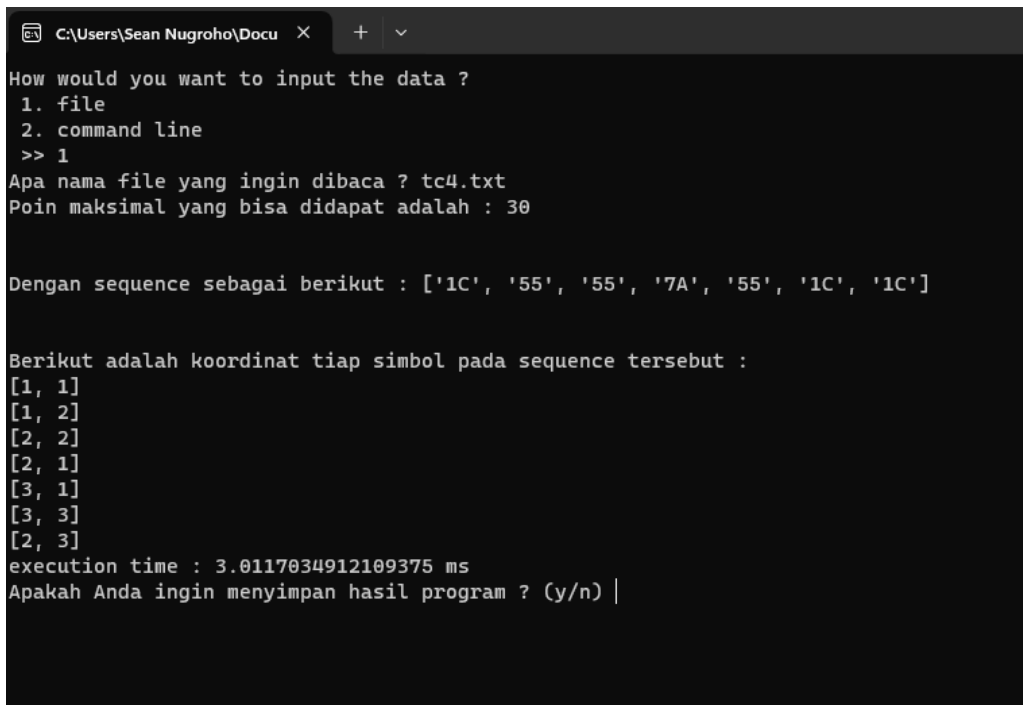
Input :



```
12
3 3
1C 7A 55
55 55 E9
BD 1C 1C
3
1C 1C 7A E9
15
55 55 1C
20
1C 55 55 7A
30
```

Ln 12, Col 3 | 77 characters | 100% | Windows (CRLF) | UTF-8

Output:



```
C:\Users\Sean Nugroho\Docu >
How would you want to input the data ?
1. file
2. command line
>> 1
Apa nama file yang ingin dibaca ? tc4.txt
Poin maksimal yang bisa didapat adalah : 30

Dengan sequence sebagai berikut : ['1C', '55', '55', '7A', '55', '1C', '1C']

Berikut adalah koordinat tiap simbol pada sequence tersebut :
[1, 1]
[1, 2]
[2, 2]
[2, 1]
[3, 1]
[3, 3]
[2, 3]
execution time : 3.0117034912109375 ms
Apakah Anda ingin menyimpan hasil program ? (y/n) |
```

## 4.2 Input Lewat CLI

### Test Case 5

```
C:\Users\Sean Nugroho\Docu  X  +  v
How would you want to input the data ?
1. file
2. command line
>> 2
Masukkan jumlah token yang ingin diinput: 5
Masukkan token - token unik tersebut dengan tiap token dipisah dengan spasi (e.g. BD 1C 7A ): BD 1C 7A 55 E9
Masukkan besar buffer : 5
Masukkan dimensi matrix (e.g. 6 6) : 3 7
Membuat matrix secara acak ...
Inilah matrix yang digunakan:
BD BD 7A
7A BD 1C
BD E9 E9
1C 1C 55
1C 55 1C
7A 1C 55
BD 55 1C
Masukkan berapa jumlah sequence yang akan dibuat : 3
Masukkan panjang maksimal sequence tersebut : 5
Membuat sequence secara acak ...
['1C', '1C'] @ 79 Points
['7A', '7A'] @ 15 Points
['E9', '7A'] @ 19 Points
Poin maksimal yang bisa didapat adalah : 79

Dengan sequence sebagai berikut : ['BD', '7A', 'BD', '1C', '55']

Berikut adalah koordinat tiap simbol pada sequence tersebut :
[1, 1]
[1, 2]
[2, 2]
[2, 4]
[3, 4]
execution time : 2.030611038208008 ms
Apakah Anda ingin menyimpan hasil program ? (y/n) y
Apa nama filenya ? tc5.txt
Ketik apapun untuk keluar dari program ... |
```



## Test Case 6

```
C:\Users\Sean Nugroho\Docu X + v
How would you want to input the data ?
1. file
2. command line
>> 2
Masukkan jumlah token yang ingin diinput: 4
Masukkan token - token unik tersebut dengan tiap token dipisah dengan spasi (e.g. BD 1C 7A ): BD 1C 7A 55
Masukkan besar buffer : 9
Masukkan dimensi matrix (e.g. 6 6) : 7 7
Membuat matrix secara acak ...
Inilah matrix yang digunakan:
BD 7A 55 7A 55 BD BD
55 BD 1C 55 7A 7A 55
7A 55 1C BD BD 55 1C
1C 7A BD 1C BD 7A 55
1C 7A 1C 7A 1C 7A BD
7A BD 55 1C 7A 1C 7A
BD 55 55 1C BD BD 55
Masukkan berapa jumlah sequence yang akan dibuat : 3
Masukkan panjang maksimal sequence tersebut : 4
Membuat sequence secara acak ...
['BD', '7A', 'BD', '7A'] @ 17 Points
['1C', '55'] @ 83 Points
['7A', '7A', 'BD'] @ 43 Points
Poin maksimal yang bisa didapat adalah : 143

Dengan sequence sebagai berikut : ['BD', '7A', 'BD', '7A', '7A', 'BD', '55', '1C', '7A']

Berikut adalah koordinat tiap simbol pada sequence tersebut :
[1, 1]
[1, 3]
[4, 3]
[4, 1]
[2, 1]
[2, 2]
[1, 2]
[1, 5]
[2, 5]
execution time : 43793.147563934326 ms
Apakah Anda ingin menyimpan hasil program ? (y/n) y
Apa nama filenya ? tc6.txt
Ketik apapun untuk keluar dari program ... |
```

## Test Case 7

```
C:\Users\Sean Nugroho\Docu X + v
How would you want to input the data ?
1. file
2. command line
>> 2
Masukkan jumlah token yang ingin diinput: 3
Masukkan token - token unik tersebut dengan tiap token dipisah dengan spasi (e.g. BD 1C 7A ): BD 1C 7A
Masukkan besar buffer : 12
Masukkan dimensi matrix (e.g. 6 6) : 3 3
Membuat matrix secara acak ...
Inilah matrix yang digunakan:
BD BD 1C
1C 7A 7A
1C BD 7A
Masukkan berapa jumlah sequence yang akan dibuat : 5
Masukkan panjang maksimal sequence tersebut : 5
Membuat sequence secara acak ...
['BD', 'BD'] @ 66 Points
['BD', '7A', '1C', '7A', 'BD'] @ 13 Points
['BD', 'BD', '7A', 'BD'] @ 45 Points
['7A', 'BD'] @ 74 Points
['1C', '7A', 'BD', '7A'] @ 68 Points
Poin maksimal yang bisa didapat adalah : 208

Dengan sequence sebagai berikut : ['BD', '1C', '7A', 'BD', '1C', '7A']

Berikut adalah koordinat tiap simbol pada sequence tersebut :
[1, 1]
[1, 2]
[2, 2]
[2, 1]
[3, 1]
[3, 2]
execution time : 2.0558834075927734 ms
Apakah Anda ingin menyimpan hasil program ? (y/n) y
Apa nama filenya ? tc7.txt
Ketik apapun untuk keluar dari program ... |
```

## Test Case 8

```
C:\Users\Sean Nugroho\Docu X + v
How would you want to input the data ?
1. file
2. command line
>> 2
Masukkan jumlah token yang ingin diinput: 4
Masukkan token - token unik tersebut dengan tiap token dipisah dengan spasi (e.g. BD 1C 7A ): BD 1C 7A 55
Masukkan besar buffer : 4
Masukkan dimensi matrix (e.g. 6 6) : 7 1
Membuat matrix secara acak ...
Inilah matrix yang digunakan:
BD 1C 7A 55 7A 7A BD
Masukkan berapa jumlah sequence yang akan dibuat : 3
Masukkan panjang maksimal sequence tersebut : 4
Membuat sequence secara acak ...
['55', 'BD'] @ 79 Points
['BD', '7A'] @ 31 Points
['7A', 'BD', '1C'] @ 21 Points
Tidak ada sekuens yang bisa digenerate ...
Ketik apapun untuk keluar dari program ... |
```

## LAMPIRAN

Pranala ke repository :

[https://github.com/Leaguemen/Tucil1\\_Stima](https://github.com/Leaguemen/Tucil1_Stima)

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 dalam berkas .txt	✓	
7. Program memiliki GUI		✓

## DAFTAR REFERENSI

<https://www.freecodecamp.org/news/brute-force-algorithms-explained/>

<https://www.simplilearn.com/tutorials/data-structure-tutorial/backtracking-algorithm>

<https://www.baeldung.com/cs/backtracking-algorithms>

[https://github.com/AJason36/Tucil1\\_13521100/tree/main](https://github.com/AJason36/Tucil1_13521100/tree/main)