LAPORAN TUGAS KECIL 1 IF2211 Strategi Algoritma

Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force

Disusun oleh
Shafiq Irvansyah (13522003)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2024

Daftar Isi

BAB 1

ALGORITMA BRUTE FORCE

1.1. Pendahuluan

Algoritma brute force adalah pendekatan langsung (*straight forward*) dalam memecahkan masalah dengan mencoba semua kemungkinan solusi secara sistematis. Meskipun sederhana, metode ini efektif dalam menemukan solusi optimal dalam berbagai konteks, seperti *sorting,* pencarian pola, *string matching*, dan berbagai persoalan lainnya. Namun, pendekatan ini memerlukan waktu dan sumber daya yang besar, sehingga pendekatan ini sering kali digunakan sebagai acuan untuk mendesain algoritma lainnya yang lebih efektif dan efisien.

Cyberpunk 2077 Breach Protocol adalah *minigame* meretas dari *video game* Cyberpunk 2077. Dalam *minigame* ini, pemain akan disimulasikan untuk meretas jaringan local dari ICE (Intrusion Countermeasures Electronics) dalam *video game* tersebut. Komponen utama dari minigame ini mencakup Token, Matriks, Sekuens, dan Buffer. Pemain akan mengikuti aturan yang meliputi gerakan horizontal dan vertikal secara bergantian, memulai dengan memilih satu token di posisi teratas matriks kemudian mencocokkan sekuens pada token dalam buffer. Satu token dapat digunakan untuk beberapa sekuens dan setiap sekuens memiliki bobot hadiah yang bervariasi, dengan panjang minimal sekuens adalah dua token.

1.2. Pengaplikasian

Program ini menerapkan algoritma brute force untuk mencoba semua kemungkinan token buffer dari matriks yang diberikan sesuai dengan aturan yang telah ditentukan sebelumnya. Selain itu, algoritma brute force juga digunakan untuk mencocokkan string dari semua urutan yang diberikan terhadap semua kombinasi buffer yang telah ditemukan untuk mencari poin yang terbesar.

BAB 2

SOURCE PROGRAM

```
import numpy as np
import time as time
import random
import os

matrix_main = []
sequences = []
combination = []
width = 0
height = 0

def intro():
    print("\n
    \n
    \n")

print("
\n
\n
\n
\n
\n
\n
")

def coorToToken(li):
    result = []
    for coor in li:
        result.append(matrix_main[coor[0]][coor[1]])
    return result
```

```
def getLastSubsetIdx(input_set, subset):
    # Nyari indeks kemunculan terakhir dari subset dalam set
    subset = subset[0]
    last_index = -1
    for i in range(len(input_set) - len(subset) + 1):
        if input_set[i:i + len(subset)] == subset:
            last_index = i + len(subset)

    return last_index

def optimizeSequence(lists, main):
    last = -1
    for list in lists:
        if (getLastSubsetIdx(coorToToken(main), list)> last):
            last = getLastSubsetIdx(coorToToken(main), list)

    if(last == -1):
        return main
    else:
        for i in range(len(main)-last):
            main.pop()
        return main

def generate_sequences(tokens, max_length):
    return random.choices(tokens, k= random.randint(2,max_length))

def generate_matrix(rows, cols, items):
    matrix = []
    for _ in range(rows):
        row = []
        for _ in range(cols):
            row.append(random.choice(items))
        matrix.append(row)
    return matrix

def getOccurrences(main_list, sublist):
    count = 0
```

```
    sublist_length = len(sublist)

    for i in range(len(main_list) - sublist_length + 1):
        if main_list[i:i + sublist_length] == sublist:
            count += 1
    return count

def getCombination(coor, urutan, vertical, n):
    urutan = urutan.copy()
    urutan.append(coor)
    if (n>1):
        if vertical:
            for i in range (height):
                if (not ([i,coor[1]] in urutan)):
                    getCombination([i,coor[1]], urutan, not vertical, n-1)
        else:
            for i in range (width):
                if (not ([coor[0],i] in urutan)):
                    getCombination([coor[0],i], urutan, not vertical, n-1)
    elif (n==1):
        combination.append(urutan)

def display_array(arr):
    return ' '.join(map(str, arr))

def print_matrix(matrix):
    for row in matrix:
        print("|", end=" ")
        print(" ".join(map(str, row)), end=" ")
        print("|")

def eliminate_after_subset(original_list, subset):
    subset_str = ''.join(subset)
    original_str = ''.join(original_list)
    subset_index = original_str.find(subset_str)
    if subset_index != -1:
        original_list = original_list[:subset_index + len(subset)]
    return original_list
```

```
def main():
    global matrix_main
    global sequences
    global combination
    global width
    global height

    intro()
    type = int(input("Pilih Metode Input : \n1. File \n2. Command Line Interface \n(Ketik 1 atau 2)\n>> "))

    if (type == 1):
        nama_file = input("Masukkan nama file: \n>> ")
        # Looping mencari file
        while True:
            try:
                with open(f"../test/{nama_file}", "r") as file:
                    print("File ditemukan!")
                break
            except FileNotFoundError:
                print("File tidak ditemukan. Coba lagi.")
            nama_file = input("Masukkan nama file: \n>> ")

        file = open(f"../test/{nama_file}", "r")

        # Buffer Size
        line = file.readline()
        buffer = int(line)

        # Matrix size dan matrix handle
        line = file.readline()
        size = line.split()
        width = int(size[0])
        height = int(size[1])

        for i in range(height):
```

```
            elements_in_line = []
            line = file.readline()
            rows = line.split()
            for element in rows :
                elements_in_line.append(element)
            matrix_main.append(elements_in_line)

        # Number of Sequences and Sequences
        line = file.readline()
        total_sequence = int(line)
        for i in range(total_sequence):
            sequence = [] # hapus aja kalo ga bikin error
            line = file.readline()
            sequence = line.split()

            line = file.readline()
            reward  = int(line)
            sequences.append([sequence,reward])
        file.close()

    elif (type ==2):
        jumlah_token_unik = input("Jumlah token unik: \n>> ")
        token = input("Token: \n>> ")
        token = token.split()
        buffer = int(input("Ukuran buffer: \n>> "))
        ukuran_matriks = input("Ukuran matrix (width height): \n>> ")
        jumlah_sekuens = int(input("Jumlah sekuens: \n>> "))
        ukuran_maksimal_sekuens = int(input("Ukuran maksimal sekuens: \n>> "))

        for i in range (jumlah_sekuens):

sequences.append([generate_sequences(token,ukuran_maksimal_sekuens),random.randrange(1
0,101,10)])
        ukuran_matriks = ukuran_matriks.split()
        width = int(ukuran_matriks[0])
        height = int(ukuran_matriks[1])
        matrix_main = generate_matrix(height, width,token)
```

```
    print("\n=====================================================")
    print("Generated Matrix:")
    print_matrix(matrix_main)
    print("\nGenerated Sequence:")
    for item in sequences:
        print(' - '.join(map(str, item[0])),":" ,item[1])

    print("=====================================================")

# Kalkulasi semua kemungkinan
start =time.time()

for i in range(width):
    getCombination([0,i],[],True, buffer)

max = [0,0]

# Kalkulasi point terbesar
for kombinasi in combination:
    count = 0
    for sekuens in sequences:
        count += sekuens[1]*getOccurrences(coorToToken(kombinasi),sekuens[0])
    if (count > max[1]):
        max = [kombinasi, count]
        # print("Updated max: ", max)

max[0] = optimizeSequence(sequences, max[0])


end = time.time()

# Menampilkan Hasil
print("\n=======================result=========================")
if max[1]==0:
    print("0")
else:
    print(max[1])
    for coor in max[0]:
```

```
            print(matrix_main[coor[0]][coor[1]], end = ' ')
        print()
        for coor in max[0]:
            print(f"{coor[1]+1}, {coor[0]+1}")
    print()
    print((end-start)*1000, "ms")
    print("=======================================================")
    finish = input("Apakah ingin menyimpan solusi? (y/n) \n>> ")


    if finish == 'y':
        index = 1
        while True:
            os.chdir('../test')
            filename = f"output({index}).txt"
            if not os.path.exists(filename):
                with open(filename, 'w') as file:
                    if max[1] == 0:
                        file.write("0\n")
                    else:
                        file.write(str(max[1]))
                        file.write("\n")
                        for coor in max[0]:
                            file.write(str(matrix_main[coor[0]][coor[1]]) + ' ')
                        file.write("\n")
                        for coor in max[0]:
                            file.write(f"{coor[1]+1}, {coor[0]+1}\n")
                    file.write(f"{(end-start)*1000} ms")
                break
            index += 1

main()
```

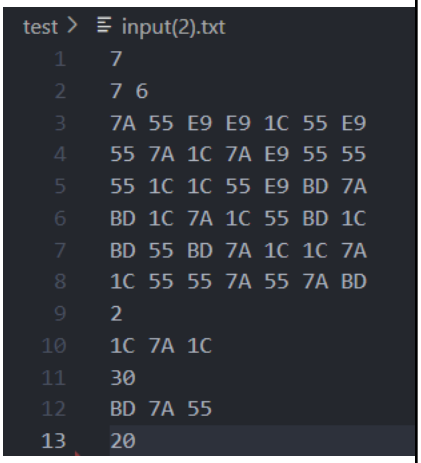
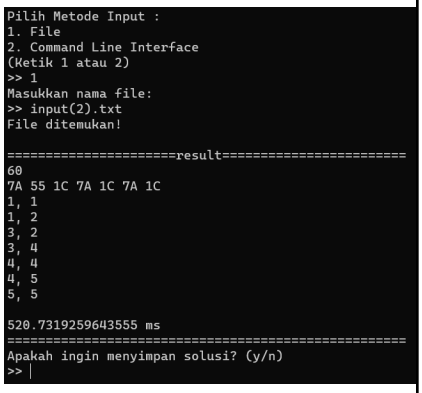
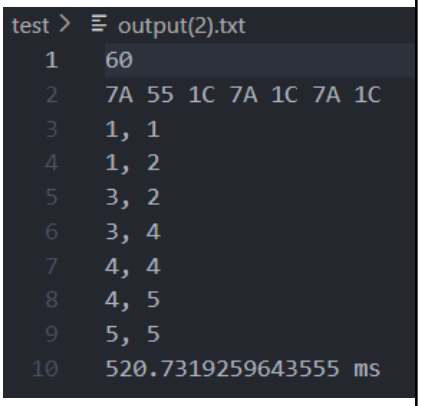
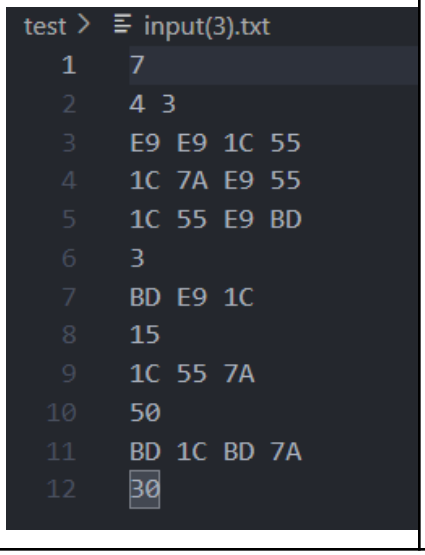
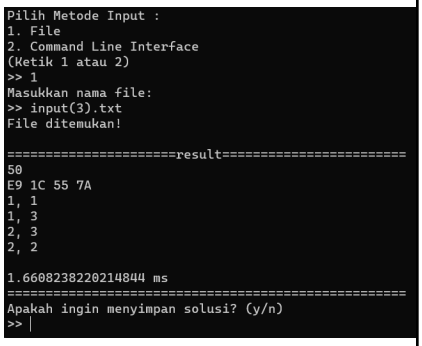
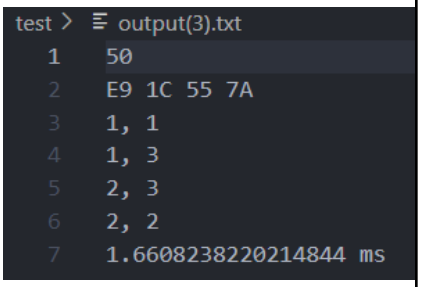
BAB 3

TESTING PROGRAM

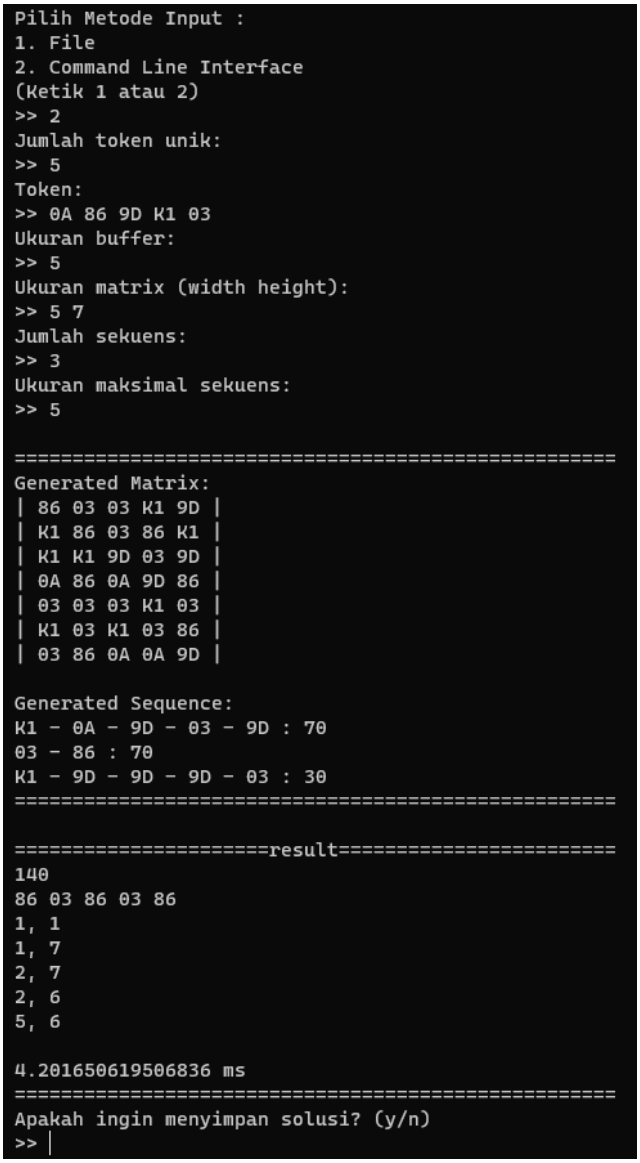
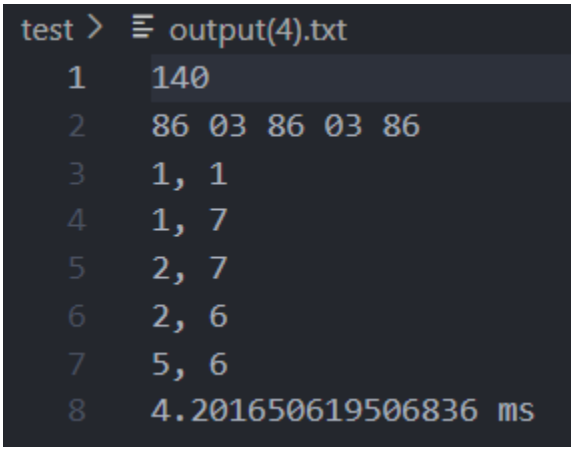
3.1. Initial

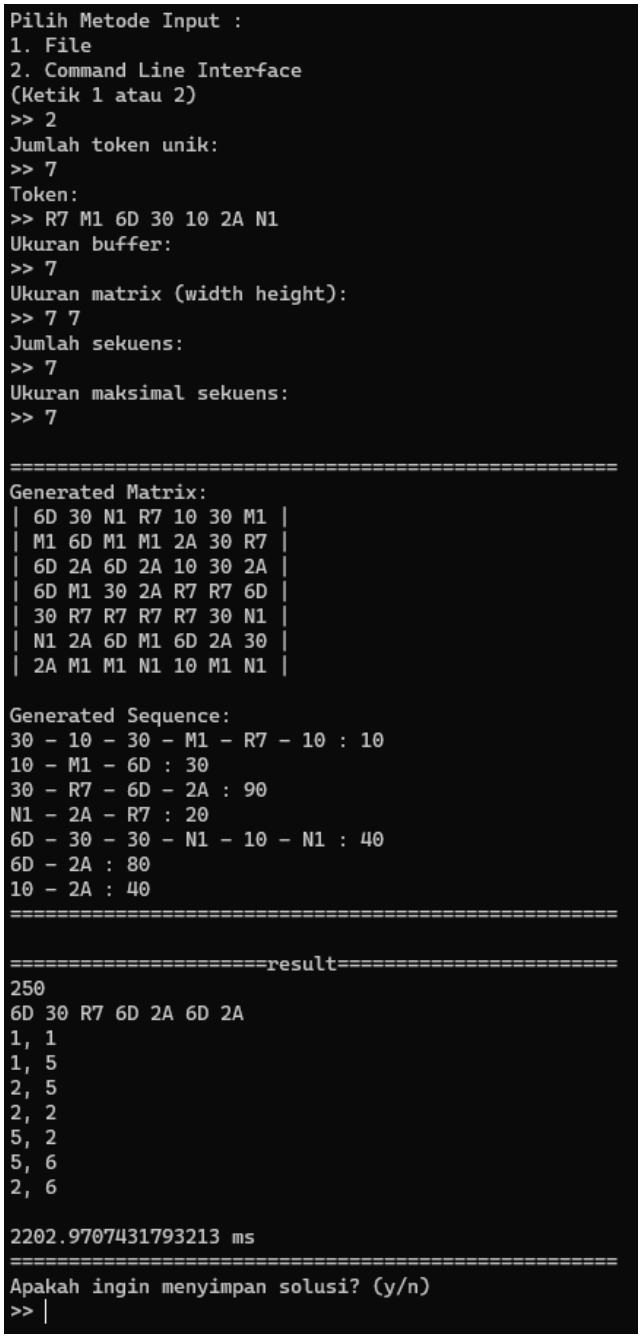
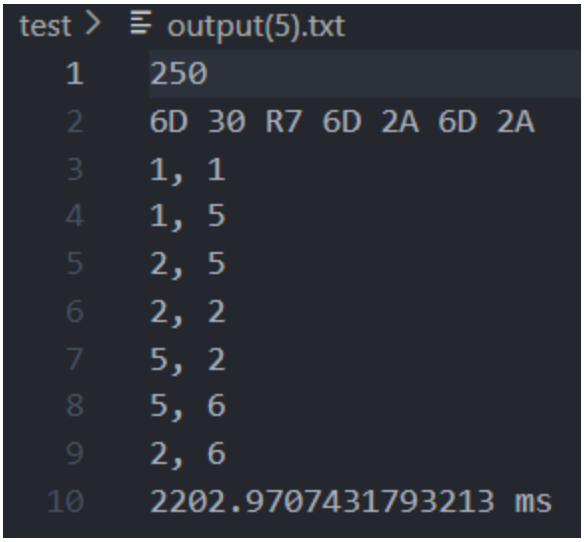
3.2. Input File .txt

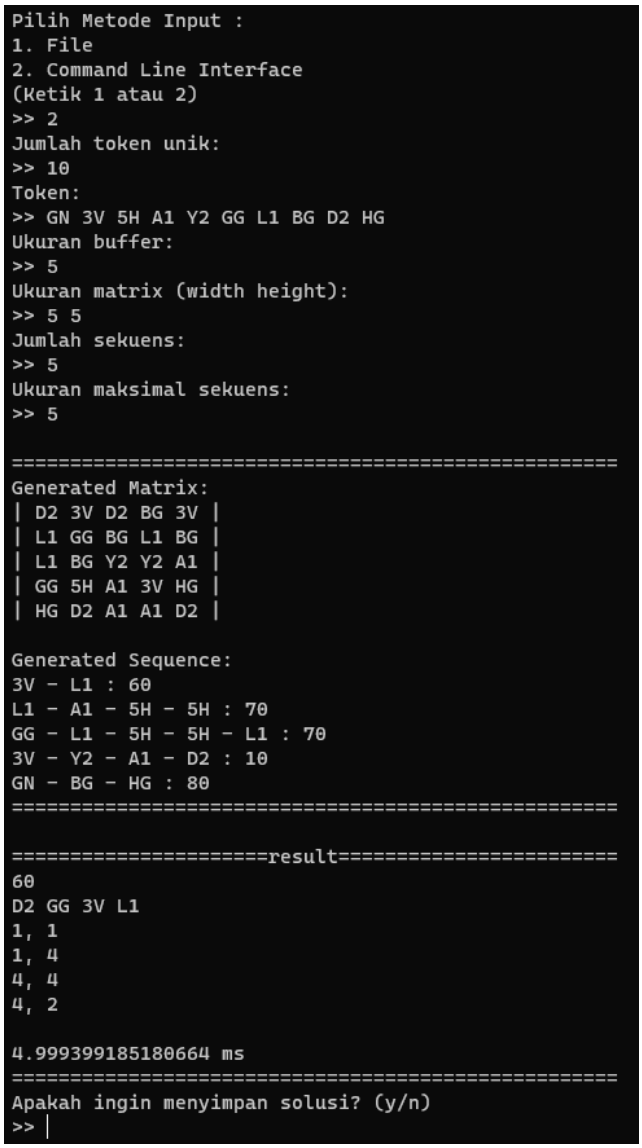
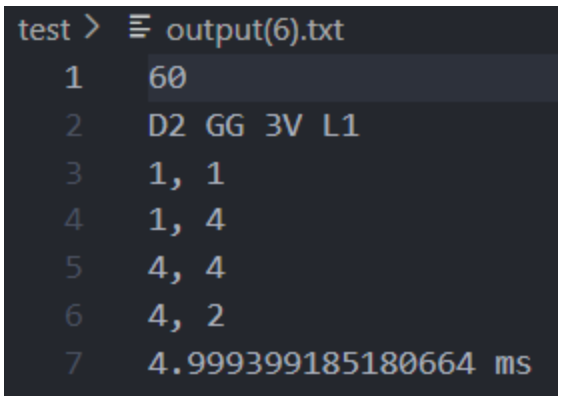
No.	File input .txt	Terminal	File output .txt
1	test > input(1).txt 7 6 6 7A 55 E9 E9 1C 55 55 7A 1C 7A E9 55 55 1C 1C 55 E9 BD BD 1C 7A 1C 55 BD BD 55 BD 7A 1C 1C 1C 55 55 7A 55 7A 3 BD E9 1C 15 BD 7A BD 20 BD 1C BD 55 30	Pilih Metode Input : 1. File 2. Command Line Interface (Ketik 1 atau 2) >> 1 Masukkan nama file: >> input(1).txt File ditemukan! =====result===== 50 7A BD 7A BD 1C BD 55 1, 1 1, 4 3, 4 3, 5 6, 5 6, 3 1, 3 312.9076957702637 ms ===== Apakah ingin menyimpan solusi? (y/n) >>	test > output(1).txt 50 7A BD 7A BD 1C BD 55 1, 1 1, 4 3, 4 3, 5 6, 5 6, 3 1, 3 312.9076957702637 ms

No.	File input .txt	Terminal	File output .txt
2			
3			

3.3. Input *Command Line Interface*

No.	Terminal	File output .txt
1		

2	 Pilih Metode Input : 1. File 2. Command Line Interface (Ketik 1 atau 2) >> 2 Jumlah token unik: >> 7 Token: >> R7 M1 6D 30 10 2A N1 Ukuran buffer: >> 7 Ukuran matrix (width height): >> 7 7 Jumlah sekuens: >> 7 Ukuran maksimal sekuens: >> 7 ========== Generated Matrix: | 6D 30 N1 R7 10 30 M1 | | M1 6D M1 M1 2A 30 R7 | | 6D 2A 6D 2A 10 30 2A | | 6D M1 30 2A R7 R7 6D | | 30 R7 R7 R7 R7 30 N1 | | N1 2A 6D M1 6D 2A 30 | | 2A M1 M1 N1 10 M1 N1 | Generated Sequence: 30 - 10 - 30 - M1 - R7 - 10 : 10 10 - M1 - 6D : 30 30 - R7 - 6D - 2A : 90 N1 - 2A - R7 : 20 6D - 30 - 30 - N1 - 10 - N1 : 40 6D - 2A : 80 10 - 2A : 40 ========== ==========result========== 250 6D 30 R7 6D 2A 6D 2A 1, 1 1, 5 2, 5 2, 2 5, 2 5, 6 2, 6 2202.9707431793213 ms ========== Apakah ingin menyimpan solusi? (y/n) >>	 test > output(5).txt 1 250 2 6D 30 R7 6D 2A 6D 2A 3 1, 1 4 1, 5 5 2, 5 6 2, 2 7 5, 2 8 5, 6 9 2, 6 10 2202.9707431793213 ms

3	 Pilih Metode Input : 1. File 2. Command Line Interface (Ketik 1 atau 2) >> 2 Jumlah token unik: >> 10 Token: >> GN 3V 5H A1 Y2 GG L1 BG D2 HG Ukuran buffer: >> 5 Ukuran matrix (width height): >> 5 5 Jumlah sekuens: >> 5 Ukuran maksimal sekuens: >> 5 ========== Generated Matrix: | D2 3V D2 BG 3V | | L1 GG BG L1 BG | | L1 BG Y2 Y2 A1 | | GG 5H A1 3V HG | | HG D2 A1 A1 D2 | Generated Sequence: 3V - L1 : 60 L1 - A1 - 5H - 5H : 70 GG - L1 - 5H - 5H - L1 : 70 3V - Y2 - A1 - D2 : 10 GN - BG - HG : 80 ========== ==========result========== 60 D2 GG 3V L1 1, 1 1, 4 4, 4 4, 2 4.999399185180664 ms ========== Apakah ingin menyimpan solusi? (y/n) >>	 test > output(6).txt 1 60 2 D2 GG 3V L1 3 1, 1 4 1, 4 5 4, 4 6 4, 2 7 4.999399185180664 ms

LAMPIRAN

Link repository: https://github.com/shafiqIrv/Tucil1_13522003

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	☐	☑