

Laporan Tugas Kecil 1

Strategi Algoritma

IF2211



**Penyelesaian Cyberpunk 2077 Breach Protocol dengan
Algoritma Brute Force**

Oleh Yusuf Ardian Sandi

NIM : 13522015

Semester II Tahun 2023/2024

Algoritma Brute Force

Pada program ini, algoritma brute force digunakan untuk menyelesaikan dua permasalahan, yaitu untuk mencari semua kemungkinan buffer yang mungkin dan mencocokkan semua buffer yang telah dicari tersebut dengan *sequences* menggunakan metode *string matching*.

Dalam mencari semua kemungkinan buffer yang mungkin menjadi solusi, digunakan fungsi rekursif untuk menjelajahi setiap rute yang mungkin. Basis dari fungsi tersebut ialah ukuran dari buffer itu sendiri, yaitu saat mencapai ukuran tertentu maka rekursi akan terhenti. Sementara itu, jika tidak memenuhi kondisi tersebut, maka program akan melakukan rekursi dengan menyesuaikan arah rutenya, yaitu vertikal atau horizontal (tergantung pada data variabel sebelumnya). Karena dalam sebuah buffer tidak diperbolehkan untuk mengunjungi koordinat yang sama pada matriks, dibuatlah matriks boolean yang menyatakan apakah koordinat tertentu sudah pernah dikunjungi atau belum. Terakhir, setiap selesai mengunjungi suatu koordinat, program akan menyimpan koordinatnya pada variabel yang sama untuk menyimpan kumpulan buffer sehingga memudahkan pengaksesan data.

Kemudian, algoritma brute force juga digunakan untuk menerapkan metode *string matching* pada pencocokan buffer dengan semua *sequence* yang ada. Mula-mula program akan mencari token yang sama dengan token pertama dari *sequence* yang akan dicocokkan. Setelah ditemukan, program akan terus menelusuri hingga semua token dibelakang *sequence* dan memeriksa apakah setiap token dan urutannya ialah cocok atau tidak. Program akan melakukan perulangan itu terus menerus hingga akhir token buffer yang bisa dicocokkan dengan token pertama dari *sequence*. Namun, jika program telah menemukan *sequence* yang cocok, maka program tidak akan melanjutkan pencarian.

Source Program

```
# Modul input

import random
# Input dari File (TO DO VALIDASI TIAP MASUKKAN)
def inputFile():      # Diasumsikan masukkan file melalui txt sesuai
dengan format yang diinginkan
    fileName = input("Masukkan nama file dengan ekstensinya: ")
    print()

    try:
        with open("../test/" + fileName, 'r') as file:
            # Membaca buffer_size
            buffer_size = int(file.readline())

            # Membaca matrix_width dan height dan memvalidasi bahwa
            matrix_width dan matrix_height harus berupa bilangan bulat yang lebih besar
            dari 0

            line = file.readline().strip()
            matrix_width, matrix_height = line.split() # Memisahkan data
            menjadi dua bagian berdasarkan spasi
            if int(matrix_width) <= 0 or int(matrix_height) <= 0:
                raise ValueError("Lebar dan tinggi matrix harus lebih besar
                dari 0.")

            col_matrix = int(matrix_width)
            row_matrix = int(matrix_height)
```

```

# Membaca Matrix
matrix = []
for _ in range(row_matrix):
    row = list(map(str, file.readline().strip().split()))
    # Pastikan panjang setiap baris sesuai dengan jumlah kolom
    yang diharapkan
    if len(row) != col_matrix:
        raise ValueError("Kesalahan pada ukuran matrix.")
    # Memvalidasi setiap elemen matrix
    for elem in row:
        if not (elem.isalnum() and len(elem) == 2):
            raise ValueError("Setiap token dalam matrix harus
alfanumerik, terdiri dari dua buah karakter, dan dalam bentuk uppercase.")
        if (any(map(lambda x: x.isalpha(), elem)) and not
elem.isupper()):
            raise ValueError("Setiap token dalam matrix harus
alfanumerik, terdiri dari dua buah karakter, dan dalam bentuk uppercase.")
        matrix.append(row)

# Membaca banyak_seq dan memvalidasi bahwa banyak_seq harus
berupa bilangan bulat yang lebih besar dari 0
banyak_seq = int(file.readline().strip())
if banyak_seq <= 0:
    raise ValueError("Banyak sequence harus lebih besar dari
0.")

# Membaca sequences dan memvalidasi bahwa:
# 1. Jumlah sequence yang dimasukkan sesuai dengan yang
diinginkan
# 2. Setiap sequence terdiri dari token yang harus alfanumerik
dan terdiri dari dua buah karakter serta harus dalam bentuk uppercase
# 3. Reward harus berupa bilangan bulat (bisa saja berupa
bilangan negatif)
sequences = []
reward_seq = []
for _ in range(banyak_seq):
    sequence_tokens = list(map(str,
file.readline().strip().split()))
    sequences.append(sequence_tokens)
    reward_seq.append(int(file.readline().strip()))
    # if len(sequences[-1]) > buffer_size:
    #     raise ValueError("Sequence tidak boleh lebih panjang
dari buffer.")

    # print(sequences[-1][0])
    # if not all(map(lambda x: x.isalnum() and len(x) == 2 and
x.isupper(), sequences[-1])):

```

```

        # raise ValueError("Setiap token dalam sequence harus
alfanumerik dan terdiri dari dua buah karakter serta harus dalam bentuk
uppercase.")
        for token in sequence_tokens:
            if not (token.isalnum() and len(token) == 2):
                raise ValueError("Setiap token dalam sequence harus
alfanumerik, terdiri dari dua buah karakter, dan dalam bentuk uppercase.")
            if (any(map(lambda x: x.isalpha(), token)) and not
token.isupper()):
                raise ValueError("Setiap token dalam sequence harus
alfanumerik, terdiri dari dua buah karakter, dan dalam bentuk uppercase.")

        # Memvalidasi jika len(sequences) == banyak_seq
        if len(sequences) != banyak_seq:
            raise ValueError("Jumlah sequence yang dimasukkan tidak
sesuai dengan yang diinginkan.")

        return buffer_size, col_matrix, row_matrix, matrix, banyak_seq,
sequences, reward_seq

except FileNotFoundError:
    print(f"File {fileName} tidak ditemukan.")
    print()
    return inputFile()
except ValueError as e:
    print("Terjadi kesalahan saat membaca file:", e)
    print()
    return inputFile()
except Exception as e:
    print("Terjadi kesalahan saat membaca file:", e)
    print()
    return inputFile()

# Input dari Terminal
def inputTerminal():

    while True:
        try:
            banyak_token = int(input("Masukkan jumlah token unik: "))
            if banyak_token <= 0:
                print("Jumlah token unik harus lebih besar dari 0.\n")
            else:
                break
        except ValueError:
            print("Masukan harus berupa bilangan bulat.\n")
    print()

    # Validasi:

```

```
# 1. Jumlah token yang dimasukkan harus sesuai dengan yang diinginkan
# 2. Setiap token harus unik
# 3. Setiap token harus alfanumerik dan terdiri dari dua buah karakter
serta harus dalam bentuk uppercase
```

```
while True:
    # Menerima input token
    token = input("Masukkan token: ").split()

    # Memeriksa panjang token
    if len(token) != banyak_token:
        print(f"Jumlah token yang dimasukkan harus {banyak_token}")
        continue

    # Memeriksa uniknya setiap token
    if len(set(token)) != banyak_token:
        print("Setiap token harus unik")
        continue

    # Memeriksa setiap token secara terpisah
    for token_elem in token:
        if not (token_elem.isalnum() and len(token_elem) == 2):
            print("Setiap token dalam sequence harus alfanumerik,
terdiri dari dua buah karakter, dan dalam bentuk uppercase.")
            break
        if (any(map(lambda x: x.isalpha(), token_elem)) and not
token_elem.isupper()):
            print("Setiap token dalam sequence harus alfanumerik,
terdiri dari dua buah karakter, dan dalam bentuk uppercase.")
            break
    else:
        # Jika semua token valid, keluar dari loop
        break
```

```
print()
```

```
while True:
    try:
        buffer_size = int(input("Masukkan ukuran buffer: "))
        if buffer_size <= 0:
            print("Ukuran buffer harus lebih besar dari 0.\n")
        else:
            break # Keluar dari perulangan saat masukan valid
    except ValueError:
        print("Masukan harus berupa bilangan bulat.\n")
print()
```

```
while True:
    try:
```

```

        col_matrix = int(input("Masukkan lebar matrix: "))
        if col_matrix <= 0:
            print("Lebar matrix harus lebih besar dari 0.\n")
        else:
            break # Keluar dari perulangan saat masukan valid
    except ValueError:
        print("Masukan harus berupa bilangan bulat.\n")
print()

while True:
    try:
        row_matrix = int(input("Masukkan tinggi matrix: "))
        if row_matrix <= 0:
            print("Tinggi matrix harus lebih besar dari 0.\n")
        else:
            break # Keluar dari perulangan saat masukan valid
    except ValueError:
        print("Masukan harus berupa bilangan bulat.\n")
print()

while True:
    try:
        banyak_seq = int(input("Masukkan banyak sequence: "))
        if banyak_seq <= 0:
            print("Banyak sequence harus lebih besar dari 0.\n")
        else:
            break # Keluar dari perulangan saat masukan valid
    except ValueError:
        print("Masukan harus berupa bilangan bulat.\n")
print()

while True:
    try:
        ukuran_max_seq = int(input("Masukkan ukuran maksimum sequence:
"))
        if ukuran_max_seq <= 0:
            print("Ukuran maksimum sequence harus lebih besar dari
0.\n")
        else:
            break # Keluar dari perulangan saat masukan valid
    except ValueError:
        print("Masukan harus berupa bilangan bulat.\n")
print()

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

matrix = []

```

```

    # melakukan random dari token yang dimasukkan dan menyusun nya sehingga
    terbentuk matrix sesuai dengan ukurannya
    for i in range(row_matrix):
        row = []
        for j in range(col_matrix):
            row.append(token[random.randint(0, banyak_token-1)])
        matrix.append(row)

    # Menampilkan matrix
    print("Berikut matriks permainan anda:")
    for i in range(row_matrix):
        for j in range(col_matrix):
            print(matrix[i][j], end=" ")
        print()
    print()

    # Proses me-random sequence dan reward
    sequences = []
    reward_seq = []
    for i in range(banyak_seq):
        # Melakukan random dari token yang dimasukkan untuk membentuk
sequence
        sequence = ""
        for j in range(random.randint(1, ukuran_max_seq)):
            sequence = sequence + token[random.randint(0, banyak_token-1)]
            if j != ukuran_max_seq-1:
                sequence = sequence + " "
        sequences.append(sequence)
        # Melakukan random reward untuk setiap sequence dari 1 hingga 50
reward_seq.append(random.randint(-50,
50))
# Concern: Reward bisa negatif

    # Menampilkan sequence dan reward
    print("Berikut sequence dan rewardnya:")
    for i in range(banyak_seq):
        print(f"Sequence {i+1}: {sequences[i]} \nReward: {reward_seq[i]}")
    print()

    return buffer_size, col_matrix, row_matrix, matrix, banyak_seq,
sequences, reward_seq

```

```

# Modul pemrosesan data dari masukkan user

```

```

def search(matrix, row_matrix, col_matrix, maxBufferSize, size, buffer, x,
y, visitedMat, udahVertikal): # Mengembalikan kumpulan buffer yang mungkin
dalam bentuk list of list of str
    results = [] # Inisialisasi list untuk menyimpan semua kemungkinan
array yang memenuhi syarat

```

```

    if len(buffer) == size or len(buffer) == maxBufferSize or maxBufferSize
< size:
        return [(buffer.copy(), [(x, y)])] # Jika buffer sudah mencapai
panjang yang diinginkan, tambahkan ke hasil bersama dengan koordinatnya

    elif udahVertikal:
        for i in range(col_matrix):
            if not visitedMat[x][i]:
                visitedMat[x][i] = True
                buffer.append(matrix[x][i])
                results.extend(search(matrix, row_matrix, col_matrix,
maxBufferSize, size, buffer, x, i, visitedMat, False))
                buffer.pop() # Kembalikan kondisi buffer sebelumnya
                visitedMat[x][i] = False

    else:
        for i in range(row_matrix):
            if not visitedMat[i][y]:
                visitedMat[i][y] = True
                buffer.append(matrix[i][y])
                results.extend(search(matrix, row_matrix, col_matrix,
maxBufferSize, size, buffer, i, y, visitedMat, True))
                buffer.pop() # Kembalikan kondisi buffer sebelumnya
                visitedMat[i][y] = False

    # Tambahkan koordinat saat memperpanjang hasil
    for result in results:
        result[1].append((x, y))

    return results

```

```

def solusi(hasilSearch, sequences, reward_seq):
    # Mengembalikan hasil search yang paling optimal
    # hasilSearch: list of tuple of list of str and list of tuple of int
    # Mengembalikan poin tertinggi dan index hasilSearch yang paling optimal
    # Namun, jika hasilSearch kosong, maka mengembalikan 0 dan -1
    poin_arr = [0] * len(hasilSearch) # Inisialisasi array untuk menyimpan
poin dari setiap hasilSearch
    # Melakukan string matching dengan setiap buffer pada hasilSearch
    for i in range(len(hasilSearch)):
        for j in range(len(sequences)):
            # print("hasilSearch[i][0]: ", hasilSearch[i][0])
            # print("sequences[j]: ", sequences[j])
            poin_arr[i] += reward_seq[j] * matching(hasilSearch[i][0],
sequences[j])
    # Mencari poin tertinggi
    poin = max(poin_arr)

```



```

    index = poin_arr.index(poin)

    return poin, index

def matching(buffer, sequence):
    # Mengembalikan 1 jika sequence ada di buffer, 0 jika tidak
    # buffer: list of str
    # sequence: list of str
    # Mengembalikan 1 jika sequence ada di buffer, 0 jika tidak
    # Menggunakan algoritma string matching (brute force)
    for i in range(len(buffer) - len(sequence) + 1):
        found = True
        for j in range(len(sequence)):
            if buffer[i+j] != sequence[j]:
                found = False
                break
        if found:
            return 1
    return 0

```

```

# Modul output

def outputTerminal(hasilSearch, poin, index):
    # Menampilkan hasil search yang paling optimal ke terminal
    # hasilSearch: list of tuple of list of str and list of tuple of int
    # poin: int
    # index: int
    # Menampilkan hasil search yang paling optimal
    noSolution = True
    for i in range(len(hasilSearch)):
        if hasilSearch[i][0] != ([], [(0, 0)]):
            noSolution = False
            break

    if noSolution or poin == 0:
        print("Tidak ada solusi yang mungkin.\nPoin: 0")

    else:
        print("Hasil solusi yang paling optimal:")
        print("Poin:", poin)
        for i in range(len(hasilSearch[index][0])):
            print(hasilSearch[index][0][i], end=" ")
        print()
        for i in range(len(hasilSearch[index][1])-2, -1, -1):
            x, y = hasilSearch[index][1][i]
            x += 1 # Tambahkan 1 pada absis
            y += 1 # Tambahkan 1 pada ordinat

```

```

        print(f"({y}, {x})", end=" ") # Cetak dengan urutan yang
dibalik
    print()

def outputFile(hasilSearch, poin, index, exe_time):
    # Meminta nama file dari pengguna
    nama_file = input("Masukkan nama file dengan ekstensinya: ")

    # Membuka file dengan mode menulis (w)
    with open("../test/" + nama_file, 'w') as file:
        noSolution = True
        for i in range(len(hasilSearch)):
            if hasilSearch[i][0] != ([], [(0, 0)]):
                noSolution = False
                break

        if noSolution or poin == 0:
            file.write("0\nTidak ada solusi yang mungkin.\n")
        else:
            file.write(str(poin) + "\n")
            for i in range(len(hasilSearch[index][0])):
                file.write(str(hasilSearch[index][0][i]) + " ")
            file.write("\n")
            for i in range(len(hasilSearch[index][1])-2, -1, -1):
                x, y = hasilSearch[index][1][i]
                x += 1 # Tambahkan 1 pada absis
                y += 1 # Tambahkan 1 pada ordinat
                file.write(f"({y}, {x})\n")
            file.write("\n")
        file.write(f"{exe_time:.2f} ms\n")

```

```

import numpy as np
from inputUser import *
from process import *
from output import *
import time

print('''\nWelcome to Cyberpunk 2077
Breach Protocol
-----

Pilih opsi input
1. File
2. Terminal
''')

opsiInput = None
while opsiInput not in [1, 2]:

```

```

    try:
        opsiInput = int(input("Masukkan opsi input (1 atau 2): "))
        if opsiInput not in [1, 2]:
            print("Masukkan tidak valid. Harap masukkan angka 1 atau 2.")
    except ValueError:
        print("Masukkan tidak valid. Harap masukkan angka 1 atau 2.")

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

if opsiInput==1:
    buffer_size, col_matrix, row_matrix, matrix, banyak_seq, sequences,
    reward_seq = inputFile()

else:
    buffer_size, col_matrix, row_matrix, matrix, banyak_seq, sequences,
    reward_seq = inputTerminal()
#   int           , int           , int           , int[][], int           , int[][] ,
int[]

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

#####
# Nge test file inputUser.py
# print("Hasil input:")
# print("Ukuran buffer:", buffer_size)
# print("Ukuran kolom matrix:", col_matrix)
# print("Ukuran baris matrix:", row_matrix)
# print("Matrix:")
# print(matrix)
# print("Banyak sequence:", banyak_seq)
# print("Sequence:")
# print(sequences)
# print("Reward sequence:")
# print(reward_seq)
# print("-----\n")
# exit()
#####

# Catat waktu mulai eksekusi
start_time = time.time()

# Mencari panjang sequence terpanjang dan terpendek
maxLen = len(sequences[0])
minLen = len(sequences[0])
for i in range(banyak_seq):
    if len(sequences[i]) > maxLen:
        maxLen = len(sequences[i])
    if len(sequences[i]) < minLen:

```

```

        minLen = len(sequences[i])

# Mencari semua buffer yang mungkin
buffer = []
visitedMat = np.zeros((row_matrix, col_matrix), dtype=bool)
for i in range(row_matrix):
    for j in range(col_matrix):
        visitedMat[i][j] = False

if buffer_size < minLen:
    print("Tidak ada buffer yang mungkin")
    hasilSearch = [[[]], [(0, 0)]]
else:
    hasilSearch = []
    for i in range(minLen, buffer_size+1):
        result = search(matrix, row_matrix, col_matrix, buffer_size, i,
buffer, 0, 0, visitedMat, True)
        if result != [[[]], [(0, 0)]]:
            # print(result)
            # print()
            # print(result[0][0])
            # print(result[0][1])
            # print()
            hasilSearch.extend(result)

        else:
            hasilSearch.extend(result)
            print("Tidak ada buffer yang mungkin")

# Mencari solusi yang paling optimal
poin, index = solusi(hasilSearch, sequences, reward_seq)

# Hitung waktu eksekusi dalam milidetik
execution_time_ms = (time.time() - start_time) * 1000

outputTerminal(hasilSearch, poin, index)

# Cetak waktu eksekusi ke terminal
print(f"\nWaktu eksekusi program: {execution_time_ms:.2f} ms")

# Apakah ingin menyimpan hasil
save = ""
while save not in ['Y', 'N', 'y', 'n']:
    save = input("\nApakah ingin menyimpan solusi? (y/n): ")
    save = save.upper() # Ubah input menjadi huruf besar (uppercase)
    if save not in ['Y', 'N', 'y', 'n']:
        print("\nMasukkan tidak valid. Silakan masukkan 'y' atau 'n'.")

```

```

print()

if save == 'Y':
    outputFile(hasilSearch, poin, index, execution_time_ms)
    print("Solusi berhasil disimpan.")
else:
    print("Solusi tidak disimpan.")

```

Program Test

Input	Output
<pre> 7 6 6 1BC 55 E9 E9 1C 55 55 7A 1C 7A E9 55 55 1CS 1C 55 E9 BD BD 1C 7A 1C 55 BD BD 55 BDS 7A 1C 1C 1C 55 55 7A 55 7A 2 BD 7A BD 20 BD 1C BD 55 30 </pre>	<pre> an-html-css\Tucil1_13522015\src> py mai n.py Welcome to Cyberpunk 2077 Breach Protocol ----- Pilih opsi input 1. File 2. Terminal Masukkan opsi input: 1 ----- Masukkan nama file dengan ekstensinya: token not 2 char.txt Terjadi kesalahan saat membaca file: Se tiap token dalam matrix harus alfanumer ik, terdiri dari dua buah karakter, dan dalam bentuk uppercase. Masukkan nama file dengan ekstensinya: </pre>
<pre> 7 6 4 7A 55 E9 E9 1C 55 55 7A 1C 7A E9 55 55 1C 1C 55 E9 BD BD 1C 7A 1C 55 BD 2 BD 7A BD 20 BD 1C BD 55 30 </pre>	<pre> Tidak ada solusi yang mungkin. Poin: 0 Waktu eksekusi program: 101.70 ms Apakah ingin menyimpan solusi? (y/n):nn S olusi tidak disimpan. </pre>

<pre> 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 1C 55 55 7A 55 7A BD 55 BD 7A 1C 1C 3 BD E9 1C 15 BD 7A BD 20 BD 1C BD 55 30 </pre>	<pre> Welcome to Cyberpunk 2077 Breach Protocol ----- Pilih opsi input 1. File 2. Terminal Masukkan opsi input: 1 ----- Masukkan nama file dengan ekstensinya: test.txt ----- Hasil solusi yang paling optimal: Poin: 50 7A BD 7A BD 1C BD 55 (1, 1) (1, 4) (3, 4) (3, 6) (6, 6) (6, 3) (1, 3) Waktu eksekusi program: 626.71 ms Apakah ingin menyimpan solusi? (y/n):yy Masukkan nama file dengan ekstensinya: hasil-test.txt Solusi berhasil disimpan. </pre>
<pre> 7 10 10 7A 55 E9 E9 1C 55 7A 55 E9 E9 1C 55 7A 55 55 55 1C 7A E9 55 55 7A E9 55 55 1C 1C 55 E9 BD BD BD 1C 7A 1C 55 BD BD 1C 7A 1C BD BD 55 BD 7A 1C 1C E9 BD BD 55 1C 55 55 7A 55 7A 7A 55 7A 1C 1C E9 1C 7A 7A 55 1C 1C E9 E9 55 55 7A 55 1C 1C E9 E9 7A 55 E9 E9 1C 55 7A 55 E9 E9 1C 55 7A 55 55 55 1C 7A E9 55 3 BD E9 1C 15 BD 7A BD 20 BD 1C BD 55 30 </pre>	<pre> Hasil solusi yang paling optimal: Poin: 50 7A BD 7A BD 1C BD 55 (1, 1) (1, 4) (10, 4) (10, 3) (7, 3) (7 , 4) (6, 4) Waktu eksekusi program: 38411.47 ms Apakah ingin menyimpan solusi? (y/n):nn Solusi tidak disimpan. </pre>

```
7
6 4
7A 55 E9 E9 1C 55
55 7A 1C 7A E9 55
55 1C 1C 55 E9 BD
BD 1C 7A 1C 55 BD
2
BD 7A BD
20
AAA 1C BD 55
30
```

```
Welcome to Cyberpunk 2077
Breach Protocol
```

```
-----
Pilih opsi input
```

1. File
2. Terminal

```
Masukkan opsi input: 1
```

```
-----
Masukkan nama file dengan ekstensinya:
target-false-token.txt
```

```
Terjadi kesalahan saat membaca file: Se
tiap token dalam sequence harus alfanum
erik, terdiri dari dua buah karakter, d
an dalam bentuk uppercase.
```

```
Masukkan nama file dengan ekstensinya:█
```

<p>Welcome to Cyberpunk 2077 Breach Protocol</p> <p>-----</p> <p>Pilih opsi input</p> <ol style="list-style-type: none"> 1. File 2. Terminal <p>Masukkan opsi input (1 atau 2): a Masukkan tidak valid. Harap masukkan angka 1 atau 2. Masukkan opsi input (1 atau 2): 2</p> <p>-----</p> <p>Masukkan jumlah token unik: a Masukan harus berupa bilangan bulat.</p> <p>Masukkan jumlah token unik: 5</p> <p>Masukkan token: 7a 55 E9 1C BD Setiap token dalam sequence harus alfanumerik, terdiri dari dua buah karakter, dan dalam bentuk uppercase. Masukkan token: 7A 55 E9 1C Jumlah token yang dimasukkan harus 5 Masukkan token: +A 55 E9 1C BD Setiap token dalam sequence harus alfanumerik, terdiri dari dua buah karakter, dan dalam bentuk uppercase. Masukkan token: 7A 55 E9 1C BD</p> <p>Masukkan ukuran buffer: 7</p> <p>Masukkan lebar matrix: 6</p> <p>Masukkan tinggi matrix: 4</p> <p>Masukkan banyak sequence: 2</p> <p>Masukkan ukuran maksimum sequence: 4</p> <p>-----</p>	<p>-----</p> <p>Berikut matriks permainan anda:</p> <pre>BD 55 7A 1C 55 7A 55 E9 E9 BD E9 BD E9 BD E9 7A 55 BD 55 E9 7A BD 7A E9</pre> <p>Berikut sequence dan rewardnya:</p> <p>Sequence 1: 7A E9 55 E9 Reward: -49</p> <p>Sequence 2: 7A 55 Reward: 8</p> <p>-----</p> <p>Tidak ada solusi yang mungkin. Poin: 0</p> <p>Waktu eksekusi program: 75.94 ms</p> <p>Apakah ingin menyimpan solusi? (y/n):nn</p> <p>Solusi tidak disimpan.</p>
---	---

Pranala Repository

<https://github.com/Yusufarsan/tucil1.git>

Lampiran

Poin	Ya	Tidak
------	----	-------

Program berhasil dikompilasi tanpa kesalahan	V	
Program berhasil dijalankan	V	
Program dapat membacamasukan berkas .txt	V	
Program dapat menghasilkan masukan secara acak	V	
Solusi yang diberikan program optimal	V	
Program dapat menyimpan solusi dalam berkas .txt	V	
Program memiliki GUI		V