Laporan Tugas Kecil 1
IF2211 Strategi Algoritma
Penyelesaian *Cyberpunk 2077 Breach Protocol* dengan Algoritma *Brute Force*

Disusun oleh:
Steven Tjhia (13522103)

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2023

Daftar Isi

Bab I
Deskripsi Masalah

**bagian ini diambil dari spesifikasi tugas*

Gambar 1 Permainan Breach Protocol
(Sumber: https://cyberpunk.fandom.com/wiki/Quickhacking)

Cyberpunk 2077 Breach Protocol adalah *minigame* meretas pada permainan video *Cyberpunk 2077*. *Minigame* ini merupakan simulasi peretasan jaringan local dari *ICE (Intrusion Countermeasures Electronics)* pada permainan *Cyberpunk 2077*. Komponen pada permainan ini antara lain 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.

Aturan permainan Breach Protocol antara lain:

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

Ilustrasi kasus :

Diberikan matriks sebagai berikut dan ukuran buffernya adalah tujuh

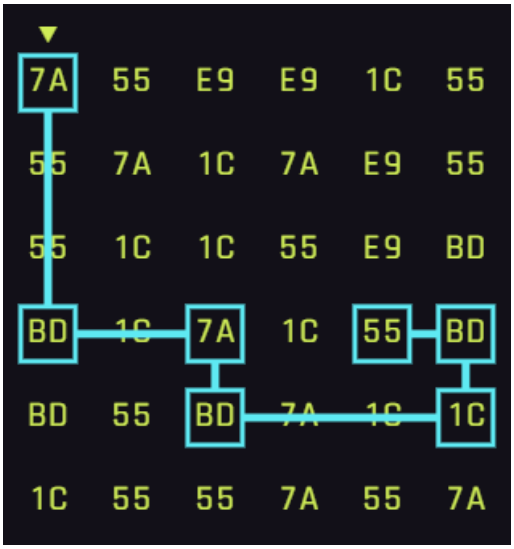
7A	55	E9	E9	1C	55
55	7A	1C	7A	E9	55
55	1C	1C	55	E9	BD
BD	1C	7A	1C	55	BD
BD	55	BD	7A	1C	1C
1C	55	55	7A	55	7A

Dengan sekuens sebagai berikut:

1. BD E9 1C dengan hadiah berbobot 15.
2. BD 7A BD dengan hadiah berbobot 20.
3. BD 1C BD 55 dengan hadiah berbobot 30.

Maka solusi yang optimal untuk matriks dan sekuens yang diberikan adalah sebagai berikut:

- Total bobot hadiah : 50 poin
- Total langkah : 6 langkah



Gambar 2 Contoh Solusi
(Sumber: https://cyberpunk-hacker.com/)

Tugas anda adalah menemukan solusi dari **permainan Breach Protocol** yang paling optimal untuk setiap kombinasi matriks, sekuens, dan ukuran buffer dengan menggunakan ***algoritma brute force***

Bab II
Algoritma Brute Force dalam Penyelesaian Cyberpunk 2077 Breach Protocol

Minigame Breach Protocol dari *Cyberpunk 2077* dapat diselesaikan dengan mencoba semua kemungkinan kombinasi token yang valid pada *buffer*. Selain digunakan untuk pencarian solusi optimal, algoritma *brute force* juga digunakan pada proses *sequence matching* untuk mengetahui hasil kalkulasi *reward* pada kombinasi token pada *buffer* saat ini. *Reward* atau *value* maksimal yang didapatkan akan diperbarui jika *reward* pada *buffer* lebih besar dari *reward* maksimum saat ini. Untuk lebih lengkapnya, algoritma *brute force* yang digunakan akan dijelaskan di bawah ini:

2.1 Algoritma *Brute Force* pada Pencarian Semua Kemungkinan Kombinasi Token

Langkah 1

Pertama, kita tentukan terlebih dahulu elemen matriks pada baris pertama matriks. Kita dapat memulai pada e_{11} (token (elemen) matriks pada baris pertama dan kolom pertama).

Langkah 2

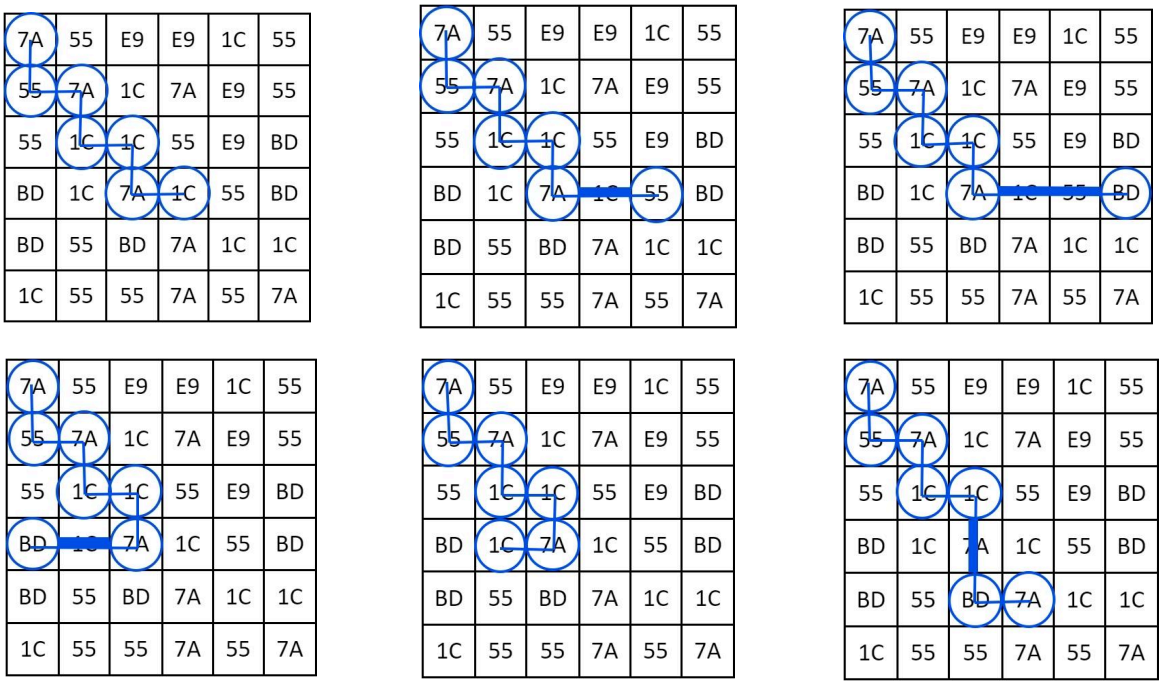
Selanjutnya, langkah yang perlu dilakukan adalah mencari token untuk dimasukkan ke dalam *buffer* secara vertikal, dimulai dengan lompat 1 langkah ke bawah. Kemudian dilanjutkan dengan lompat 1 (secara horizontal) ke kanan. Jika sudah mencapai ujung bawah atau ujung kanan matriks, lanjutkan dengan token paling atas atau paling kiri matriks. Langkah-langkah pada *Langkah 2* diulangi sampai buffer terisi penuh dan setiap setelah melakukan 1 kali lompatan, dilakukan pengecekan jumlah *reward* yang bisa didapatkan dari token-token yang *buffer*. Jika nilai *reward* yang didapatkan lebih besar dari nilai optimal sementara atau *reward* yang dihasilkan sama dengan nilai optimal tetapi jumlah token pada *buffer* lebih sedikit, maka perbarui solusi optimal saat ini.
**solusi optimal adalah solusi yang menghasilkan reward maksimal dengan jumlah token seminimal mungkin*

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

Gambar 3 Penggambaran Langkah 2

Langkah 3

Setelah melakukan lompatan bawah-kanan-bawah-kanan sebesar 1 langkah, selanjutnya dilakukan penambahan jarak lompatan terakhir sebesar 1. Jika lompatan terakhir dilakukan secara horizontal, maka proses tersebut dilakukan sebanyak (jumlah kolom matriks - 1) kali. Kemudian dilanjutkan dengan penambahan jarak lompatan sebelum lompatan terakhir sebesar 1 dan seterusnya. Sama seperti sebelumnya, *Langkah 3* diulangi sampai seluruh kombinasi token yang dimulai dari token awal (pada kolom pertama dan baris pertama matriks) didapatkan dan setiap setelah melakukan 1 kali lompatan, dilakukan pengecekan jumlah *reward* yang bisa didapatkan dari token-token yang *buffer*. Jika nilai *reward* yang didapatkan lebih besar dari nilai optimal sementara atau *reward* yang dihasilkan sama dengan nilai optimal tetapi jumlah token pada *buffer* lebih sedikit, maka perbarui solusi optimal saat ini.



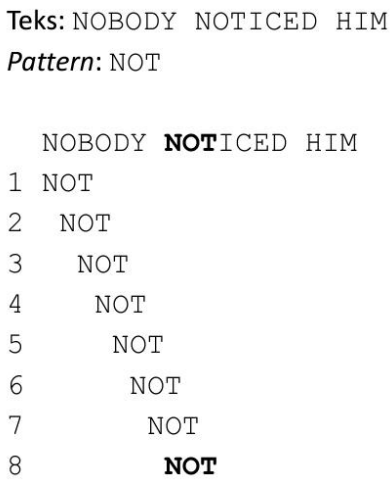
Gambar 3 Penggambaran Langkah 3

Langkah 4

Setelah didapatkan hasil dari seluruh kombinasi token yang dimulai dari kolom pertama dan baris pertama matriks, selanjutnya dilakukan pencarian hasil dari seluruh kombinasi token yang dimulai dari baris pertama matriks dan kolom kedua sampai dengan kolom terakhir. Dengan demikian seluruh hasil dari kombinasi akan didapatkan, begitu juga dengan solusi optimal untuk permasalahan tersebut.

2.2 Algoritma *Brute Force* pada Proses *Sequence Matching*

Proses Sequence Matching dilakukan untuk kalkulasi reward berdasarkan isi token pada buffer. Pada dasarnya ide dari proses *sequence matching* ini berasal dari *string matching*.



Gambar 4 Contoh Algoritma Brute Force pada proses String Matching
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf)

Langkah 1

Lakukan pengecekan panjang token pada sekuens dan panjang token pada *buffer*. Karena tidak ada kepastian bahwa ukuran sekuens tidak lebih besar dari ukuran *buffer* maka ketika terdapat ukuran sekuens yang lebih besar dari ukuran *buffer* akan langsung dilewati dan tidak dicek.

Langkah 2

Lakukan pengecekan token pertama pada sekuens dengan token pertama pada *buffer*. Jika kedua token sama, lanjutkan pengecekan token berikutnya pada masing-masing sekuens dan *buffer*. Jika pengecekan token yang dilakukan sampai akhir sekuens tidak terdapat perbedaan, maka nilai *reward* dari *buffer* tersebut ditambahkan sesuai dengan *reward* dari sekuens tersebut. Namun jika di tengah-tengah pengecekan terdapat perbedaan token, hentikan pengecekan dan lanjutkan pengecekan token berikutnya dari *buffer* dengan token pertama dari sekuens. Jika sama, lakukan pengecekan lanjutan seperti pada kasus sebelumnya. Jika berbeda, lanjutkan pengecekan token berikutnya dari *buffer* dengan token pertama dari sekuens hingga token ke-(panjang *buffer* - panjang sekuens).

Bab III
Implementasi Program dengan Python

Algoritma yang telah dijelaskan pada Bab 2 diimplementasikan dengan sebuah program Python. Alasan penggunaan program Python untuk penyelesaian Tucil1 ini adalah karena python memiliki banyak library yang dapat mempermudah dalam pemrosesan string dan list, selain itu juga GUI akan lebih mudah dibuat pada Python. Program dapat dijalankan melalui CLI ataupun GUI, implementasi yang akan ditunjukkan pada bab ini adalah implementasi pada CLI karena lebih modular daripada program dengan GUI.

Program pada CLI diimplementasikan secara modular dan terbagi menjadi 9 file, yaitu Main.py sebagai program utama, VarGlobal.py untuk menyimpan variabel global, VarInit untuk menginisiasi beberapa variabel, Input_CLI.py untuk mendapatkan input dari pengguna melalui CLI, Input_TXT untuk mendapatkan input dari pengguna melalui file ber-ekstensi .txt, SequenceMatching.py untuk menghitung reward yang bisa didapatkan dari suatu buffer, Brute.py yang memuat algoritma brute force, Process.py untuk menjalankan algoritma brute force dan menampilkan hasilnya juga, dan SaveOutput.py untuk menyimpan hasil solusi.

Main.cpp

```
from VarInit import *
from Input_TXT import *
from Input_CLI import *
from Process import *
from SaveOutput import *

# MAIN PROGRAM
print("\033[1;34;40m___  ____ _  _ ___ ____ ____ ____ ____ ____ ____")
print("|__] |__/ |  |  |  |___ |___ |  | |__/ |    |___")
print("|__] |  \ |__|  |  |___ |    |__| |  \ |___ |___")
print("-------------------------------------------------")
print("\033[1;33;40m")

print("\033[0;33;40mInput Type:")
print("1. Text file (.txt)")
print("2. Manual input (CLI)")

while True:
    try:
        input_type = int(input(">>>\033[1;33;40m "))
    except:
        print("\033[0;31;40mInvalid Type Input\033[0;33;40m")
```

```
    else:
        if (input_type == 1) or (input_type == 2):
            break
        else:
            print("\033[0;31;40mInvalid Input\033[0;33;40m")

if (input_type == 1):
    print()
    TXT_Input()
elif (input_type == 2):
    CLI_Input()
else:
    print("\033[0;31;40mInvalid Input\033[0;33;40m")

initOffsets()
DisplayResult()
wantToSave()
print("\033[1;34;40m================================================")
print(" _____ _  _   _   _  _ _  ____   _____  _   _")
print("|_   _| || | /_\ | \| | |/ /\ \ / / _ \| | | |")
print("  | | | __ |/ _ \| .` | ' <  \ V / (_) | |_| |")
print("  |_| |_||_/_/ \_\_|\_|_|\_\  |_| \___/ \___/ ")
print("================================================")

print("\033[0;37;40m")
```

Input_CLI.py

```
import random
import time
import VarGlobal

def CLI_Input():

    print("\n\033[1;33;40mPlease Complete Below Requirements")
    print("---------------------------------\033[0;33;40m")

    # jumlah token unik
    while True:
        try:
```

```
            n_token_unik = int(input("Jumlah Token Unik : \033[1;33;40m"))
        except:
            print("\033[0;31;40mInvalid Type Input\033[0;33;40m")
        else:
            break

    # token unik
    while True:
        try:
            tokens = input("\033[0;33;40mToken (XX YY ... ZZ) (case sensitive):
\033[1;33;40m")
        except:
            print("\033[0;31;40mInvalid Type Input\033[0;33;40m")
        else:
            tokens = tokens.split(" ")
            tokens = list(dict.fromkeys(tokens))
            if (len(tokens) != n_token_unik):
                # invalid
                print("\033[0;31;40mJumlah token unik harus sesuai dengan jumlah
token yang telah dimasukkan di atas\033[0;33;40m")

            else:
                # loop cek
                valid = True
                for i in range(0,n_token_unik):
                    if (len(tokens[i]) != 2):
                        print("\033[0;31;40mToken harus terdiri dari 2 karakter
alfanumerik\033[0;33;40m")
                        valid = False
                        break
                    else:
                        if (tokens[i][0] not in VarGlobal.alfanumerik) or
(tokens[i][1] not in VarGlobal.alfanumerik):
                            print("\033[0;31;40mToken harus terdiri dari 2
karakter alfanumerik\033[0;33;40m")
                            valid = False
                            break
                if valid:
                    break
```

```
    # buffer size
    while True:
        try:
            VarGlobal.buffer_size = int(input("\033[0;33;40mBuffer size :
\033[1;33;40m"))
        except:
            print("\033[0;31;40mInvalid Type Input\033[0;33;40m")
        else:
            if VarGlobal.buffer_size > 0:
                break
            else:
                print("\033[0;31;40mUkuran Buffer harus bernilai
positif\033[0;33;40m")

    # matriks size
    while True:
        try:
            VarGlobal.col, VarGlobal.row = [ int(x) for x in
input("\033[0;33;40mMatriks size (col row) : \033[1;33;40m").split(" ")]
        except:
            print("\033[0;31;40mInvalid Type Input\033[0;33;40m")
        else:
            if VarGlobal.row <= 0:
                print("\033[0;31;40mUkuran baris pada matriks harus bernilai
positif\033[0;33;40m")
            if VarGlobal.col <= 0:
                print("\033[0;31;40mUkuran kolom pada matriks harus bernilai
positif\033[0;33;40m")
            if VarGlobal.row > 0 and VarGlobal.col > 0:
                VarGlobal.matriks =
[[(tokens[int(random.randrange(0,n_token_unik))]) for j in
range(0,VarGlobal.col)] for i in range(0,VarGlobal.row)]
                break

    # ukuran maks sequence
    while True:
        try:
```

```
            max_sequence_size = int(input("\033[0;33;40mMax sequence size :
\033[1;33;40m"))
        except:
            print("\033[0;31;40mInvalid Type Input\033[0;33;40m")
        else:
            if  max_sequence_size < 2:
                print("\033[0;31;40mUkuran minimal sequence adalah
2\033[0;33;40m")
            else:
                break

    # jumlah sequence
    while True:
        try:
            VarGlobal.n_sequence = int(input("\033[0;33;40mJumlah sequence :
\033[1;33;40m"))
        except:
            print("\033[0;31;40mInvalid Type Input\033[0;33;40m")
        else:
            if VarGlobal.n_sequence > 0:
                VarGlobal.list_sequence = []
                isPossible = True
                threshold = 100
                for i in range(0, VarGlobal.n_sequence):
                    counter_threshold = 0
                    random.seed(time.time())
                    while True:
                        len_seq = int(random.randrange(2,max_sequence_size+1))
                        seq = []

                        for j in range(len_seq):
                            idx = int(random.randrange(0,n_token_unik))
                            seq.append(tokens[idx])

                        if seq not in VarGlobal.list_sequence:
                            break
                        else:
                            if counter_threshold == threshold:
                                isPossible = False
```

```
                            break
                        else:
                            counter_threshold += 1

                if isPossible:
                    VarGlobal.list_sequence.append(seq)
                else:
                    break

            if isPossible:
                break
            else:
                print("\033[0;31;40mJumlah tersebut tidak mungkin
menghasilkan sequence yang unik semua, harus lebih kecil\033[0;33;40m")
        else:
            print("\033[0;31;40mJumlah sequence harus bernilai
positif\033[0;33;40m")

    print()

    # display matriks & sequence
    random.seed(time.time())
    print("\033[1;35;40mGenerated Matriks")
    print("-----------------\033[0;35;40m")
    for i in range(0,VarGlobal.row):
        for j in range(0,VarGlobal.col):
            if j != VarGlobal.col-1:
                print(VarGlobal.matriks[i][j], end=" ")
            else:
                print(VarGlobal.matriks[i][j])
    time.sleep(1)

    print()

    random.seed(time.time())
    VarGlobal.list_sequenceValue = [int(random.randrange(-100,101)) for i in
range(0, VarGlobal.n_sequence)]
    print("\033[1;36;40mGenerated Sequence")
    print("------------------\033[0;36;40m")
```

```
    for i in range(0,VarGlobal.n_sequence):
        leng = len(VarGlobal.list_sequence[i])
        for j in range(0,leng):
            if j != leng-1:
                print(VarGlobal.list_sequence[i][j], end=" ")
            else:

print(VarGlobal.list_sequence[i][j],"--->",VarGlobal.list_sequenceValue[i])
    time.sleep(1)
    print("\033[1;34;40m")
```

Input_TXT.py

```
import os
import re
import time
import VarGlobal

def TXT_Input():
    finished = False
    while True:
        filename = input("\033[0;33;40mInsert File Name : \033[1;33;40m").lower()
        if not filename.endswith(".txt"):
            print("\033[0;31;40mOnly .txt file can be loaded\033[0;33;40m")
            continue
        try:
            path = os.path.dirname(__file__)
            os.chdir(path)
            os.chdir("../")
            path = os.getcwd()
            path = os.path.join(path,"test",filename)

            file = open(path, "r")
        except:
            print("\033[0;31;40mFile not Found\033[0;33;40m")
        else:
            if os.stat(path).st_size==0:
                print("\033[0;31;40mFile is empty")
                print("\033[1;33;40mPlease input non-empty file name")
                continue
```

```
            print("\033[1;33;40mLoading file...")
            time.sleep(1)
            line_count = 0
            isValid = True
            for line in file:
                if line_count == 0:
                    try:
                        VarGlobal.buffer_size = int(line)
                    except:
                        isValid = False
                        print("\033[0;31;40mInvalid buffer size")
                        break
                    else:
                        if VarGlobal.buffer_size <= 0:
                            isValid = False
                            print("\033[0;31;40mInvalid buffer size")
                            break
                        else:
                            print("\033[0;32;40m✔ Buffer Size")
                            time.sleep(0.3)

                elif line_count == 1:
                    m_size = line.split(" ")
                    try:
                        VarGlobal.col = int(m_size[0])
                        VarGlobal.row = int(m_size[1])
                    except:
                        isValid = False
                        print("\033[0;31;40mInvalid matrix size")
                        break
                    else:
                        if VarGlobal.col > 0 and VarGlobal.row >0:
                            last_row_line = 1 + VarGlobal.row
                            print("\033[0;32;40m✔ Matrix Size")
                            time.sleep(0.3)
                        else:
                            isValid = False
                            print("\033[0;31;40mInvalid matrix size")
                            break
```

```
                elif 2 <= line_count <= last_row_line:
                    line = re.split('\n| ', line)
                    line.pop()

                    if len(line) != VarGlobal.col:
                        isValid = False
                        print("\033[0;31;40mInvalid matrix")
                        break
                    else:
                        for i in range(0,VarGlobal.col):
                                line[i] = (line[i])
                                if len(line[i]) != 2:
                                    isValid = False
                                    print("\033[0;31;40mInvalid token in matriks
detected")
                                    break
                                else:
                                    if (line[i][0] not in VarGlobal.alfanumerik)
or (line[i][1] not in VarGlobal.alfanumerik):
                                        print("\033[0;31;40mInvalid token in
matriks detected")
                                        isValid = False
                                        break
                        if not isValid:
                            break
                        else:
                            VarGlobal.matriks.append(line)
                            print("\033[0;32;40m✔ Matrix Row", line_count-2)
                            time.sleep(0.3)


                elif line_count == (1+last_row_line):
                    try:
                        VarGlobal.n_sequence = int(line)
                    except:
                        isValid = False
                        print("\033[0;31;40mInvalid jumlah sequence")
                        break
```

```
                    else:
                        if VarGlobal.n_sequence <= 0:
                            isValid = False
                            print("\033[0;31;40mInvalid jumlah sequence")
                            break
                        else:
                            last_line = 1 + (2*VarGlobal.n_sequence)
                            print("\033[0;32;40m✔ Jumlah Sequence")
                            time.sleep(0.3)

                elif ((line_count-last_row_line)%2 == 0) and
(line_count-last_row_line <= last_line):
                    line = re.split('\n| ', line)
                    line.pop()
                    n = len(line)
                    for i in range(0,n):
                            line[i] = (line[i])
                            if len(line[i]) != 2:
                                isValid = False
                                print("\033[0;31;40mInvalid token in sequence
detected")
                                break
                            else:
                                if (line[i][0] not in VarGlobal.alfanumerik) or
(line[i][1] not in VarGlobal.alfanumerik):
                                        print("\033[0;31;40mInvalid token in sequence
detected\033[0;33;40m")
                                        isValid = False
                                        break
                    if not isValid:
                        break
                    else:
                        VarGlobal.list_sequence.append(line)
                        print("\033[0;32;40m✔
Sequence",(line_count-last_row_line)//2)
                        time.sleep(0.3)

                elif ((line_count-last_row_line)%2 == 1) and
((line_count-last_row_line) <= last_line):
```

```
                    try:
                        value = int(line)
                    except:
                        isValid = False
                        print("\033[0;31;40mInvalid sequence reward")
                        break
                    else:
                        VarGlobal.list_sequenceValue.append(value)
                        print("\033[0;32;40m✔
Sequence",(line_count-last_row_line)//2, "Reward")
                        time.sleep(0.3)
                    if (line_count-last_row_line) == last_line:
                        finished = True
                        break

                line_count += 1

            if not isValid:
                print("\033[1;31;40mFile Process Terminated\033[0;37;40m\n")
                break
            else:
                if finished:
                    print("\033[1;32;40mData successfully loaded\033[1;32;40m")
                    time.sleep(0.3)

                    print()

                    # display matriks & sequence
                    print("\033[1;35;40mMatriks")
                    print("-------\033[0;35;40m")
                    for i in range(0,VarGlobal.row):
                        for j in range(0,VarGlobal.col):
                            if j != VarGlobal.col-1:
                                print(VarGlobal.matriks[i][j], end=" ")
                            else:
                                print(VarGlobal.matriks[i][j])
                    time.sleep(1)

                    print()
```

```
                    print("\033[1;36;40mSequence")
                    print("--------\033[0;36;40m")
                    for i in range(0,VarGlobal.n_sequence):
                        leng = len(VarGlobal.list_sequence[i])
                        for j in range(0,leng):
                            if j != leng-1:
                                print(VarGlobal.list_sequence[i][j], end=" ")
                            else:

print(VarGlobal.list_sequence[i][j],"--->",VarGlobal.list_sequenceValue[i])
                    time.sleep(1)
                    print("\033[1;34;40m")
                    break
                else:
                    print("\033[0;31;40mInvalid file content")
                    print("\033[1;31;40mFile Process Terminated\033[0;37;40m\n")
            file.close()
```

SequenceMatching.py

```
def isSequenceMatch(Buffer, Sequence):
    if len(Buffer) < len(Sequence):
        return False
    def isSequenceFound(Buffer,Sequence):
        n = len(Sequence)
        for i in range(0,n):
            if (Sequence[i] != Buffer[i]):
                return False
        return True

    n = len(Buffer) - len(Sequence) + 1
    for i in range(0,n):
        if (Buffer[i] == Sequence[0]):
            if isSequenceFound(Buffer[i:],Sequence):
                return True
    return False
```

Brute.py

```
import VarGlobal
```

```
from SequenceMatching import *

def brute(TotalRow, TotalCol):
    BufferMaxValToken = []
    BufferMaxValCoordinate = []
    MaxVal = 0

    def AppendToken(Matrix, Coordinate):

        idx_row = Coordinate[0]
        idx_col = Coordinate[1]
        Token = Matrix[idx_row][idx_col]
        VarGlobal.current_buffer_token.append(Token)
        VarGlobal.current_buffer_coordinate.append(Coordinate.copy())

    def ResetCurrentBuffer():

        VarGlobal.current_buffer_token = []
        VarGlobal.current_buffer_coordinate = []

    def AppendBuffer(CurrentBufferToken, CurrentBufferCoordinate,
BufferMaxCoordinate, BufferMaxToken, MaxValue):
        global n_sequence
        bufferValue = 0
        for i in range(0,VarGlobal.n_sequence):
            if (isSequenceMatch(CurrentBufferToken, VarGlobal.list_sequence[i])):
                bufferValue += VarGlobal.list_sequenceValue[i]

        if (MaxValue < bufferValue) or ((MaxValue == bufferValue) and
(len(CurrentBufferCoordinate) < len(BufferMaxCoordinate))):
            NewMaxValue = bufferValue

            nBuffer = len(CurrentBufferToken)

            NewBufferMaxValToken = []
            for j in range(0,nBuffer):
                NewBufferMaxValToken.append(CurrentBufferToken[j])

            NewBufferMaxValCoordinate = []
```

```
            for j in range(0,nBuffer):
                NewBufferMaxValCoordinate.append(CurrentBufferCoordinate[j])

        else:
            NewMaxValue = MaxValue
            NewBufferMaxValToken = BufferMaxToken
            NewBufferMaxValCoordinate = BufferMaxCoordinate

        return [NewBufferMaxValToken, NewBufferMaxValCoordinate, NewMaxValue]

    def UpdateOffsets():
        global offsets
        i = VarGlobal.buffer_size - 2
        while (VarGlobal.offsets[0] < TotalRow):
            VarGlobal.offsets[i] = (VarGlobal.offsets[i]+1)
            if (i%2 == 0):
                if (VarGlobal.offsets[i] >= TotalRow):
                    VarGlobal.offsets[i] = (VarGlobal.offsets[i]%TotalRow)+1
                    i-=1
                else:
                    break
            else:
                if (VarGlobal.offsets[i] >= TotalCol):
                    VarGlobal.offsets[i] = (VarGlobal.offsets[i]%TotalCol)+1
                    i-= 1
                else:
                    break

    def special_case():
        global row
        global col
        # case ketika VarGlobal.matriks hanya memiliki 1 atau 2 baris dan/atau
kolom saja
        if len(VarGlobal.offsets) == 0 or VarGlobal.row <= 2 or VarGlobal.col <=
2:
            return True
        return False

    for i in range(0,TotalCol):
```

```
        VarGlobal.offsets = []
        for j in range(0,VarGlobal.buffer_size-1):
            VarGlobal.offsets.append(1)
        while True:
            ResetCurrentBuffer()
            VarGlobal.current_coordinate = [0,i]
            AppendToken(VarGlobal.matriks,VarGlobal.current_coordinate)
            Result = AppendBuffer(VarGlobal.current_buffer_token,
VarGlobal.current_buffer_coordinate, BufferMaxValCoordinate, BufferMaxValToken,
MaxVal)
            BufferMaxValToken = Result[0]
            BufferMaxValCoordinate = Result[1]
            MaxVal = Result[2]

            for j in range(0,len(VarGlobal.offsets)):
                if (j%2 == 0):
                    VarGlobal.current_coordinate[0] =
(VarGlobal.current_coordinate[0]+VarGlobal.offsets[j])%TotalRow
                else:
                    VarGlobal.current_coordinate[1] =
(VarGlobal.current_coordinate[1]+VarGlobal.offsets[j])%TotalCol

                if VarGlobal.current_coordinate not in
VarGlobal.current_buffer_coordinate:
                    AppendToken(VarGlobal.matriks, VarGlobal.current_coordinate)
                    Result = AppendBuffer(VarGlobal.current_buffer_token,
VarGlobal.current_buffer_coordinate, BufferMaxValCoordinate, BufferMaxValToken,
MaxVal)
                    BufferMaxValToken = Result[0]
                    BufferMaxValCoordinate = Result[1]
                    MaxVal = Result[2]
                else:
                    break

            if (VarGlobal.offsets == VarGlobal.max_offsets) or special_case():
                break
            else:
                UpdateOffsets()
```

```
    VarGlobal.MAX_VALUE = MaxVal
    VarGlobal.BUFFER_MAX_VALUE_TOKEN = BufferMaxValToken
    VarGlobal.BUFFER_MAX_VALUE_COORDINATE = BufferMaxValCoordinate
```

Process.py

```
import time
import VarGlobal
from BruteForce import *

def DisplayResult():

    print("Processing Result")
    print("----------------\033[0;34;40m")

    start = time.time()
    brute(len(VarGlobal.matriks), len(VarGlobal.matriks[0]))

    print("MAX VALUE             =", VarGlobal.MAX_VALUE)

    N = len(VarGlobal.BUFFER_MAX_VALUE_TOKEN)
    print("TOKEN                 =", end=" ")
    if len(VarGlobal.BUFFER_MAX_VALUE_TOKEN) == 0:
        print("\033[0;31;40mTidak ada solusi yang menghasilkan value bernilai
positif\033[0;34;40m")
    else:
        for i in range(0, N):
            print(VarGlobal.BUFFER_MAX_VALUE_TOKEN[i],end=" ")

        print()

    print("COORDINATE (col,row)  =", end=" ")
    if len(VarGlobal.BUFFER_MAX_VALUE_COORDINATE) == 0:
        print("\033[0;31;40mTidak ada solusi yang menghasilkan value bernilai
positif\033[0;34;40m")
    else:
        for i in range(0, N):

print("("+str(VarGlobal.BUFFER_MAX_VALUE_COORDINATE[i][1]+1)+","+str(VarGlobal.BU
FFER_MAX_VALUE_COORDINATE[i][0]+1)+")",end=" ")
```

```
    end = time.time()

    VarGlobal.time_processing = int((end-start)*1000)

    print()
    print("\033[0;32;40m")
    print("Finished in", VarGlobal.time_processing," ms")
    print()
```

SaveOutput.py

```
import os
import VarGlobal

def saveSolution(path):
    file = open(path, "w")
    for i in range(0,4):
        if i == 0:
            writeThis = str(VarGlobal.MAX_VALUE) + "\n"
        elif i == 1:
            writeThis = " ".join(VarGlobal.BUFFER_MAX_VALUE_TOKEN) + "\n"
        elif i == 2:
            if len(VarGlobal.BUFFER_MAX_VALUE_COORDINATE) == 0:
                writeThis = ("Tidak ada solusi yang menghasilkan value bernilai
positif")
            else:
                N = len(VarGlobal.BUFFER_MAX_VALUE_TOKEN)
                for j in range(0, N):
                    writeThis =
(str(VarGlobal.BUFFER_MAX_VALUE_COORDINATE[j][1]+1)+","+str(VarGlobal.BUFFER_MAX_
VALUE_COORDINATE[j][0]+1)+"\n")
                    if j != N-1:
                        file.write(writeThis)
        else:
            writeThis = "\n" + str(VarGlobal.time_processing) + " ms\n"
        file.write(writeThis)
    file.close()

def wantToSave():
```

```
    print("\033[0;33;40mDo you want to save the solution? (y/n)")
    isSaved = False

    while True:

        inp = input("\033[0;33;40m>>> \033[1;33;40m")

        if inp == "y":
            while True:
                filename = input("\033[0;33;40mInsert File Name (.txt):
\033[1;33;40m")

                if ".txt" not in filename:
                    print("\033[0;31;40mOnly .txt file can be
saved\033[0;33;40m")
                else:
                    break

            try:
                path = os.path.dirname(__file__)
                os.chdir(path)
                os.chdir("../")
                path = os.getcwd()
                path = os.path.join(path,"test",filename)

                file = open(path, "x")
            except:
                print("File already exist, do you want to overwrite it? (y/n)")
                while True:
                    inp2 = input("\033[0;33;40m>>> \033[1;33;40m")
                    if inp2 == "y":
                        # SAVE
                        saveSolution(path)
                        isSaved = True
                        break
                    elif inp2 == "n":
                        print("\033[0;33;40mDo you want to save the solution?
(y/n)")
                        break
```

```
                else:
                    print("\033[0;31;40mInvalid Input")

        else:
            # SAVE
            saveSolution(path)
            isSaved = True

    elif inp == "n":
        print()
        break

    else:
        print("\033[0;31;40mInvalid Input")

    if isSaved:
        print()
        print("\033[1;32;40mFile successfully saved")
        print()
        break
```

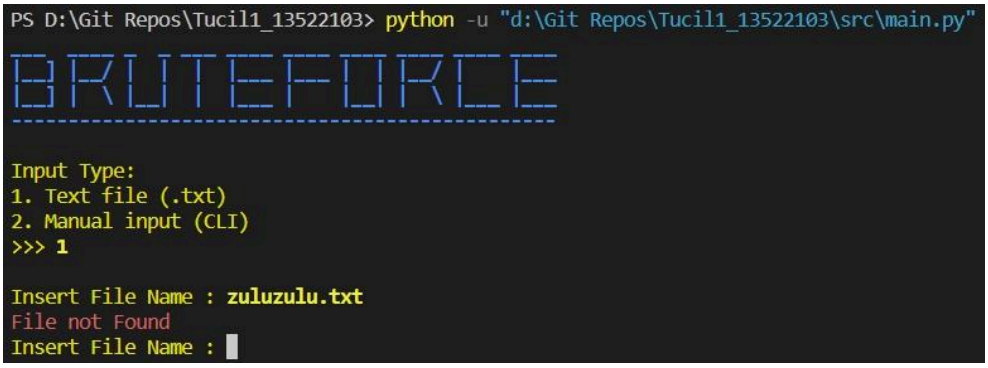
Bab IV
Eksperimen

Asumsi

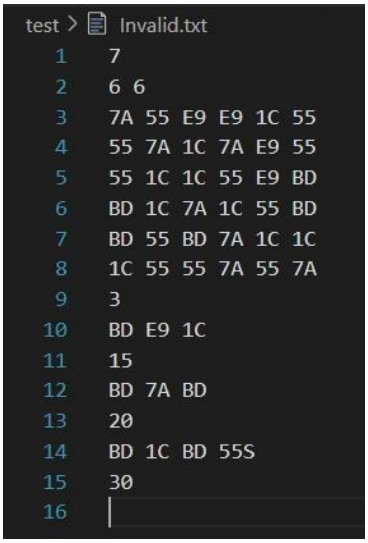
1. Seluruh file yang ingin di tes harus berada pada folder “test” (output file juga akan berada pada folder “test” (berlaku pada CLI dan GUI).
2. Jika terdapat spasi di akhir **baris matriks** atau **sequence** pada input file .txt maka file dianggap tidak valid. (Contoh terdapat pada Eksperimen 6)

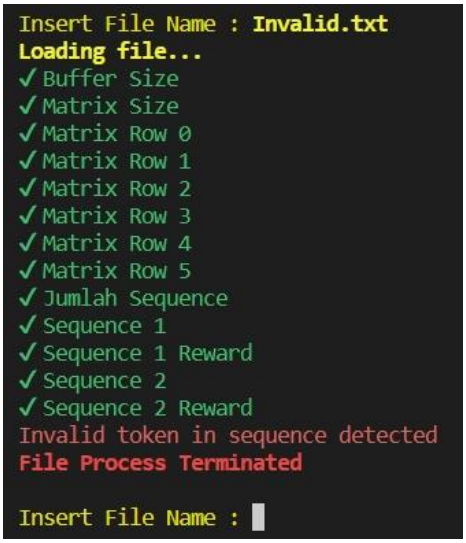
Eksperimen 1

Input File & File tidak ada



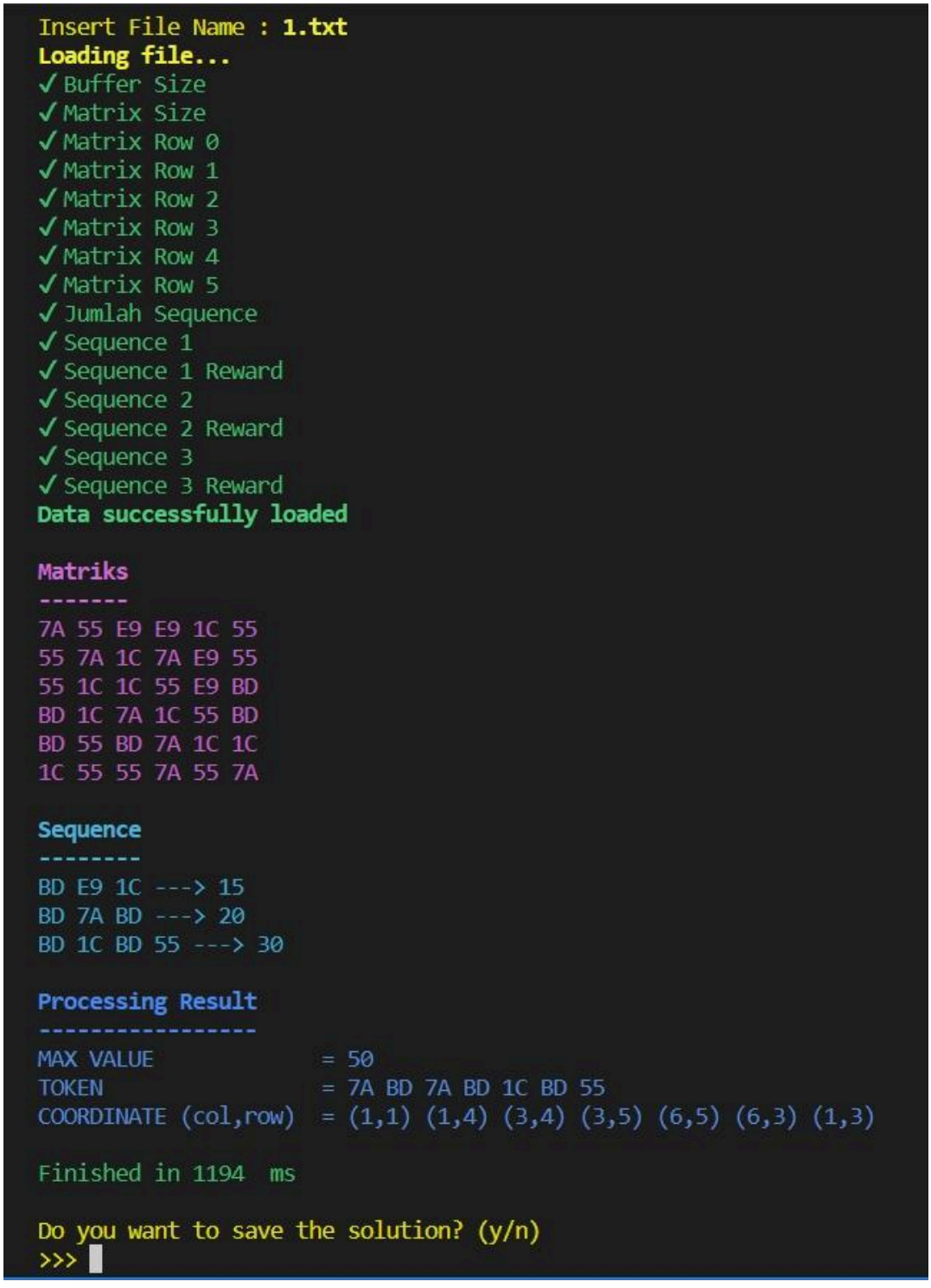
Input File & Isinya Tidak Valid



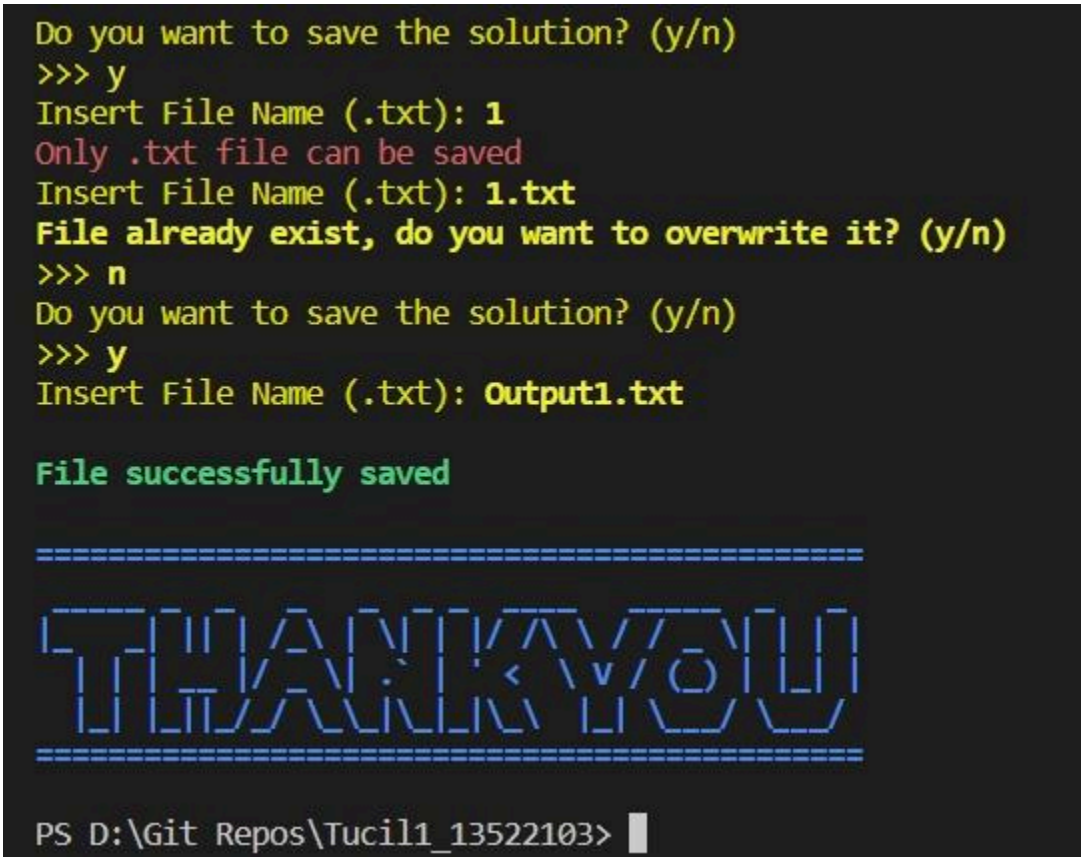


*pada line 14 terdapat token “55S”

Input File & Valid



Save : y & (Nama file tidak berakhiran dengan .txt, Nama file sudah ada, Overwrite : n, Nama file belum ada)

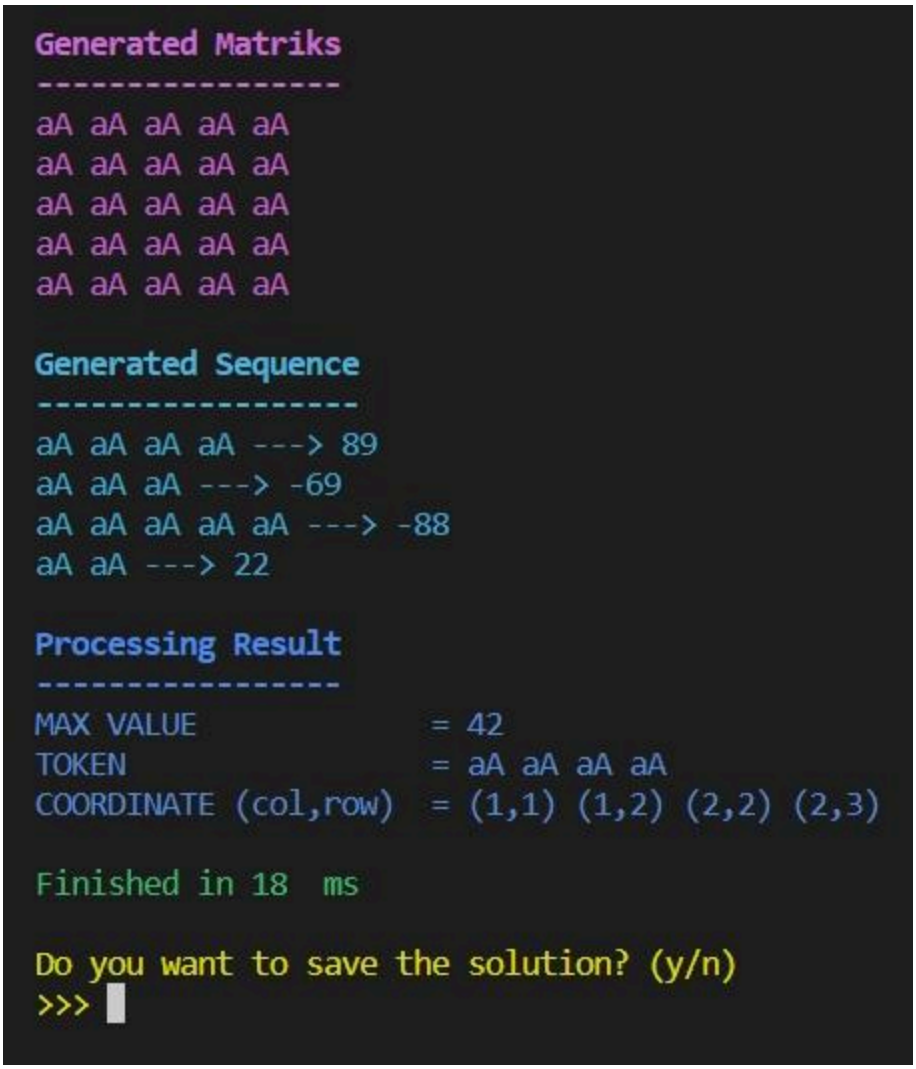


Eksperimen 2

Input Manual

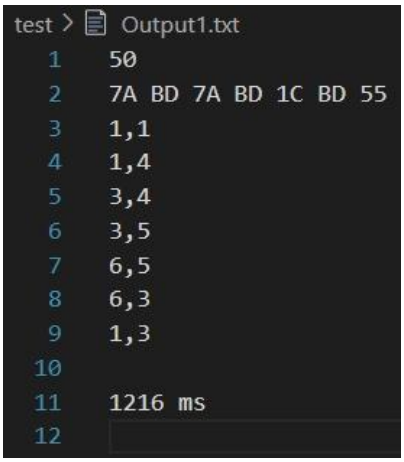
```
PS D:\Git Repos\Tucil1_13522103> python -u "d:\Git Repos\Tucil1_13522103\src\main.py"

Input Type:
1. Text file (.txt)
2. Manual input (CLI)
>>> 2
```

```
Please Complete Below Requirements
----------------------------------
Jumlah Token Unik : a
Invalid Type Input
Jumlah Token Unik : 1
Token (XX YY ... ZZ) (case sensitive): aaa
Token harus terdiri dari 2 karakter alfanumerik
Token (XX YY ... ZZ) (case sensitive): aa bb
Jumlah token unik harus sesuai dengan jumlah token yang telah dimasukkan di atas
Token (XX YY ... ZZ) (case sensitive): a!
Token harus terdiri dari 2 karakter alfanumerik
Token (XX YY ... ZZ) (case sensitive): aA
Buffer size : s
Invalid Type Input
Buffer size : 5
Matriks size (col row) : z z
Invalid Type Input
Matriks size (col row) : 0 0
Ukuran baris pada matriks harus bernilai positif
Ukuran kolom pada matriks harus bernilai positif
Matriks size (col row) : 5 5
Max sequence size : g
Invalid Type Input
Max sequence size : 5
Jumlah sequence : 7
Jumlah tersebut tidak mungkin menghasilkan sequence yang unik semua, harus lebih kecil
Jumlah sequence : 6
Jumlah tersebut tidak mungkin menghasilkan sequence yang unik semua, harus lebih kecil
Jumlah sequence : 5
Jumlah tersebut tidak mungkin menghasilkan sequence yang unik semua, harus lebih kecil
Jumlah sequence : 4
```

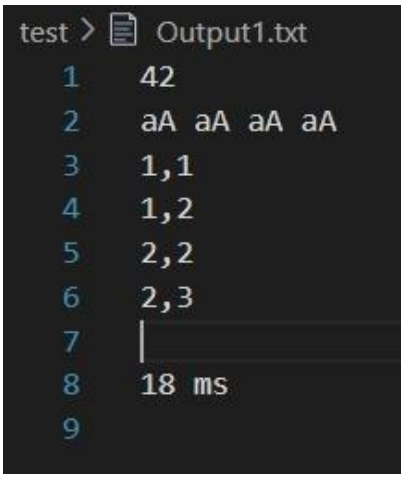


Save : y & Overwrite : y

Kondisi awal Output1.txt

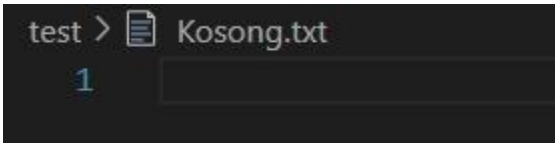


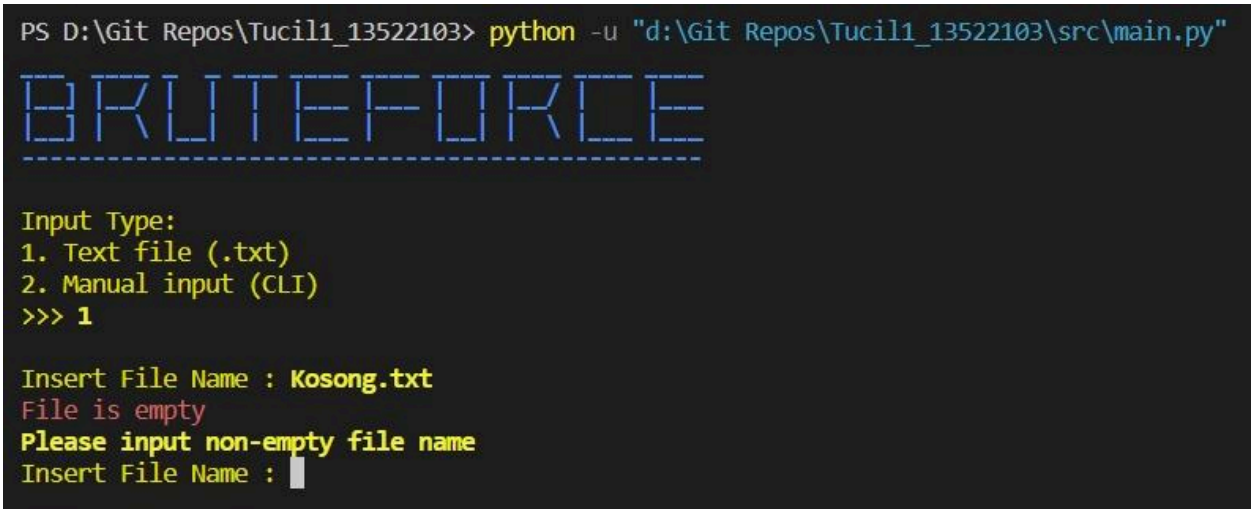
Kondisi akhir Output1.txt



Eksperimen 3

Input File & File Kosong





Input File Valid

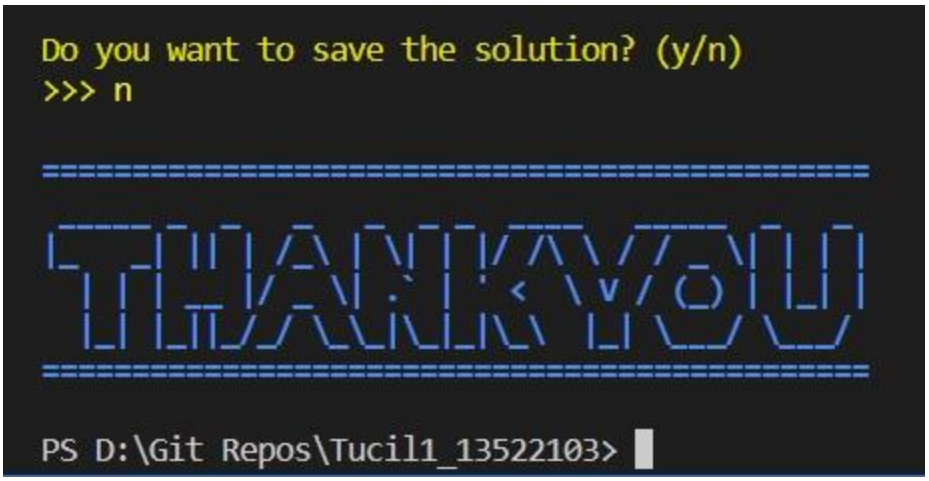
test > 3.txt

```
6
10 10
DD DD CC BB AA AA EE EE BB AA
CC EE CC BB CC DD DD DD BB EE
BB BB AA CC CC CC EE DD AA EE
EE EE AA DD EE EE AA CC AA BB
BB AA EE DD DD DD DD EE AA CC
EE BB DD EE BB CC DD DD BB CC
BB CC EE BB AA AA BB EE CC BB
EE EE CC DD BB EE DD AA AA CC
BB EE CC AA EE AA BB CC BB BB
EE CC CC AA EE CC AA CC CC CC
5
BB AA EE CC
31
BB DD
-59
AA AA AA
83
CC BB
-89
CC DD CC
96
```

```
Please input non-empty file name
Insert File Name : 3.txt
Loading file...
✓ Buffer Size
✓ Matrix Size
✓ Matrix Row 0
✓ Matrix Row 1
✓ Matrix Row 2
✓ Matrix Row 3
✓ Matrix Row 4
✓ Matrix Row 5
✓ Matrix Row 6
✓ Matrix Row 7
✓ Matrix Row 8
✓ Matrix Row 9
✓ Jumlah Sequence
✓ Sequence 1
✓ Sequence 1 Reward
✓ Sequence 2
✓ Sequence 2 Reward
✓ Sequence 3
✓ Sequence 3 Reward
✓ Sequence 4
✓ Sequence 4 Reward
✓ Sequence 5
✓ Sequence 5 Reward
Data successfully loaded
```

```
Matriks
-------
DD DD CC BB AA AA EE EE BB AA
CC EE CC BB CC DD DD DD BB EE
BB BB AA CC CC CC EE DD AA EE
EE EE AA DD EE EE AA CC AA BB
BB AA EE DD DD DD DD EE AA CC
EE BB DD EE BB CC DD DD BB CC
BB CC EE BB AA AA BB EE CC BB
EE EE CC DD BB EE DD AA AA CC
BB EE CC AA EE AA BB CC BB BB
EE CC CC AA EE CC AA CC CC CC

Sequence
--------
BB AA EE CC ---> 31
BB DD ---> -59
AA AA AA ---> 83
CC BB ---> -89
CC DD CC ---> 96

Processing Result
-----------------
MAX VALUE            = 179
TOKEN                = CC DD CC AA AA AA
COORDINATE (col,row) = (3,1) (3,6) (6,6) (6,7) (5,7) (5,1)

Finished in 11527  ms
```

Save : n



Eksperimen 4

File 4.txt

test > 4.txt

```
10
5 5
AA AA BV BV BV
BV AA AA AA AA
AA BV BV AA AA
BV BV BV BV AA
AA AA BV AA AA
10
BV AA BV BV BV AA AA BV AA
65
BV AA BV BV BV BV AA
-48
AA BV BV AA AA
-16
BV AA
-44
BV BV AA BV
-13
AA AA BV
-8
BV BV
-9
AA AA BV AA BV BV BV
9
BV AA BV BV AA
-26
BV AA BV AA AA BV
-7
```

File Input & Valid

```
PS D:\Git Repos\Tucil1_13522103> python -u "d:\Git Repos\Tucil1_13522103\src\main.py"

Input Type:
1. Text file (.txt)
2. Manual input (CLI)
>>> 1
```

```
Insert File Name : 4.txt
Loading file...
✓ Buffer Size
✓ Matrix Size
✓ Matrix Row 0
✓ Matrix Row 1
✓ Matrix Row 2
✓ Matrix Row 3
✓ Matrix Row 4
✓ Jumlah Sequence
✓ Sequence 1
✓ Sequence 1 Reward
✓ Sequence 2
✓ Sequence 2 Reward
✓ Sequence 3
✓ Sequence 3 Reward
✓ Sequence 4
✓ Sequence 4 Reward
✓ Sequence 5
✓ Sequence 5 Reward
✓ Sequence 6
✓ Sequence 6 Reward
✓ Sequence 7
✓ Sequence 7 Reward
✓ Sequence 8
✓ Sequence 8 Reward
✓ Sequence 9
✓ Sequence 9 Reward
✓ Sequence 10
✓ Sequence 10 Reward
Data successfully loaded
```

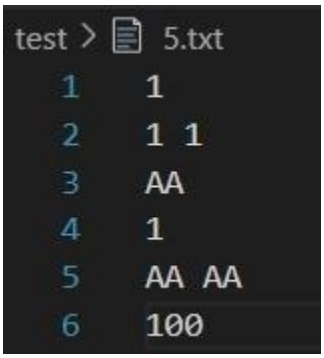
```
Matriks
-------
AA AA BV BV BV
BV AA AA AA AA
AA BV BV AA AA
BV BV BV BV AA
AA AA BV AA AA

Sequence
--------
BV AA BV BV BV AA AA BV AA ---> 65
BV AA BV BV BV BV AA ---> -48
AA BV BV AA AA ---> -16
BV AA ---> -44
BV BV AA BV ---> -13
AA AA BV ---> -8
BV BV ---> -9
AA AA BV AA BV BV BV ---> 9
BV AA BV BV AA ---> -26
BV AA BV AA AA BV ---> -7

Processing Result
-----------------
MAX VALUE            = 4
TOKEN                = BV AA BV BV BV AA AA BV AA
COORDINATE (col,row) = (3,1) (3,2) (1,2) (1,4) (2,4) (2,5) (4,5) (4,1) (1,1)

Finished in 64948  ms
```

Save : (Tidak valid, n)

```
Do you want to save the solution? (y/n)
>>> g
Invalid Input
>>> n

===============================================

===============================================

PS D:\Git Repos\Tucil1_13522103>
```

Eksperimen 5

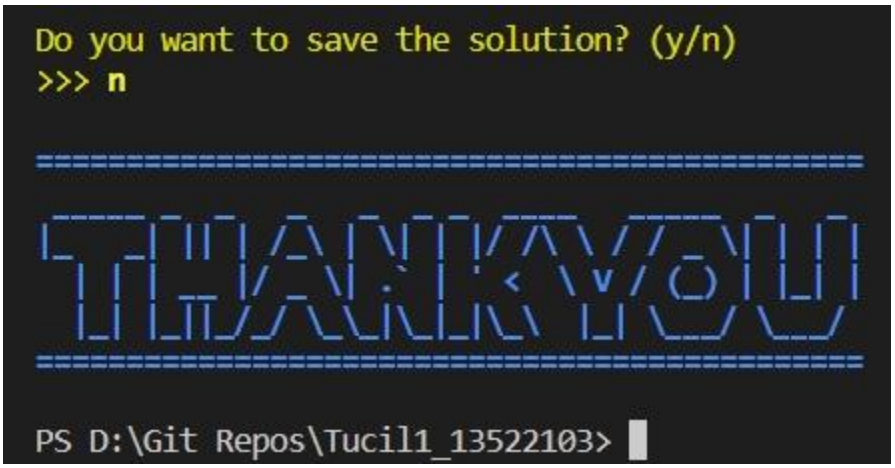
File 5.txt



File Input & Valid

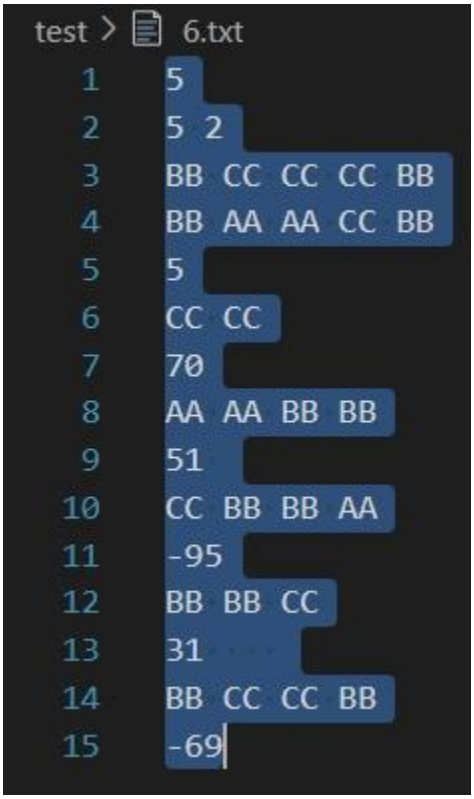
```
Insert File Name : 5.txt
Loading file...
✓ Buffer Size
✓ Matrix Size
✓ Matrix Row 0
✓ Jumlah Sequence
✓ Sequence 1
✓ Sequence 1 Reward
Data successfully loaded

Matriks
-------
AA

Sequence
--------
AA AA ---> 100

Processing Result
-----------------
MAX VALUE             = 0
TOKEN                 = Tidak ada solusi yang menghasilkan value bernilai positif
COORDINATE (col,row)  = Tidak ada solusi yang menghasilkan value bernilai positif


Finished in 0  ms
```

Save : n

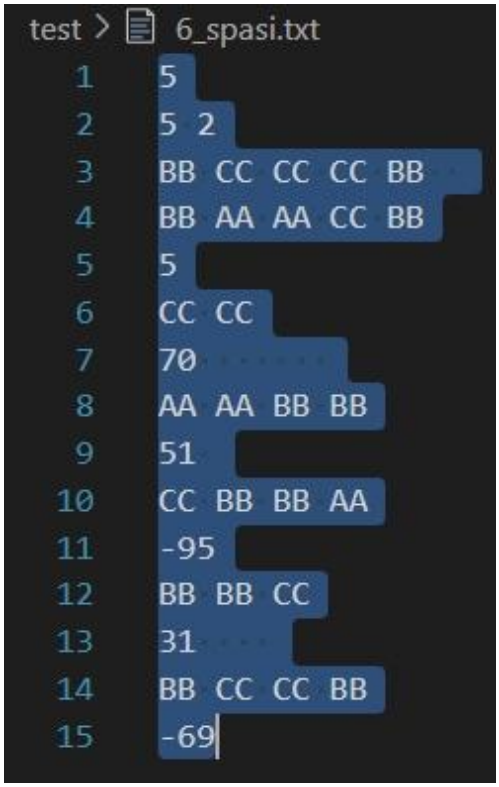


Eksperimen 6

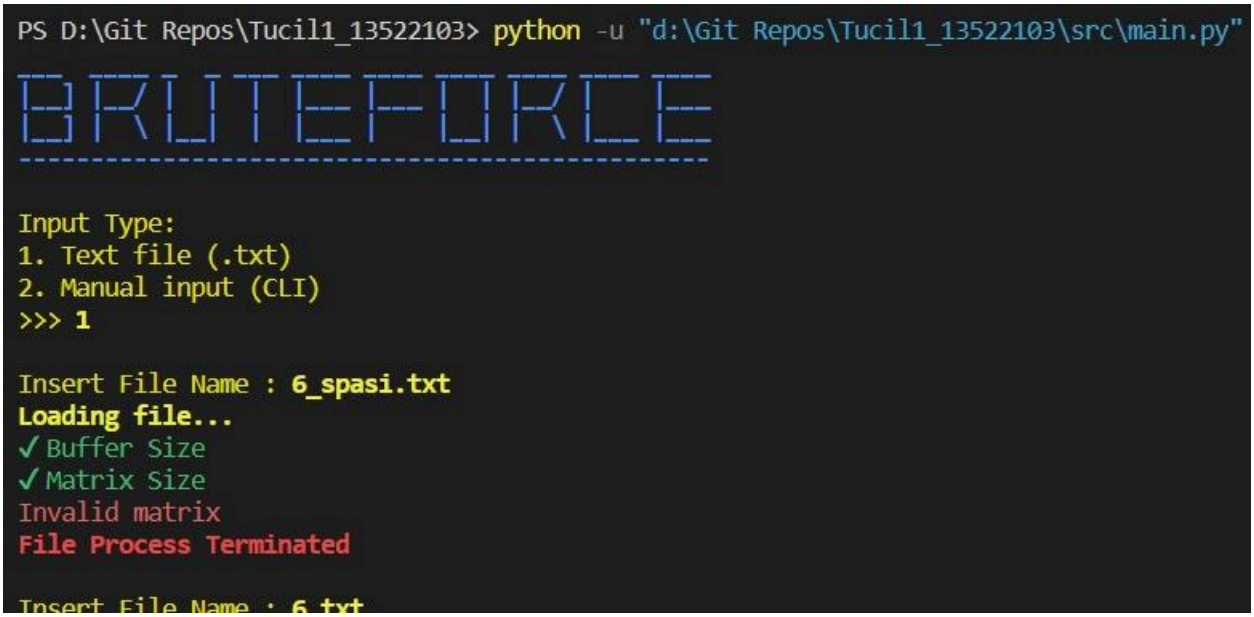
File 6.txt



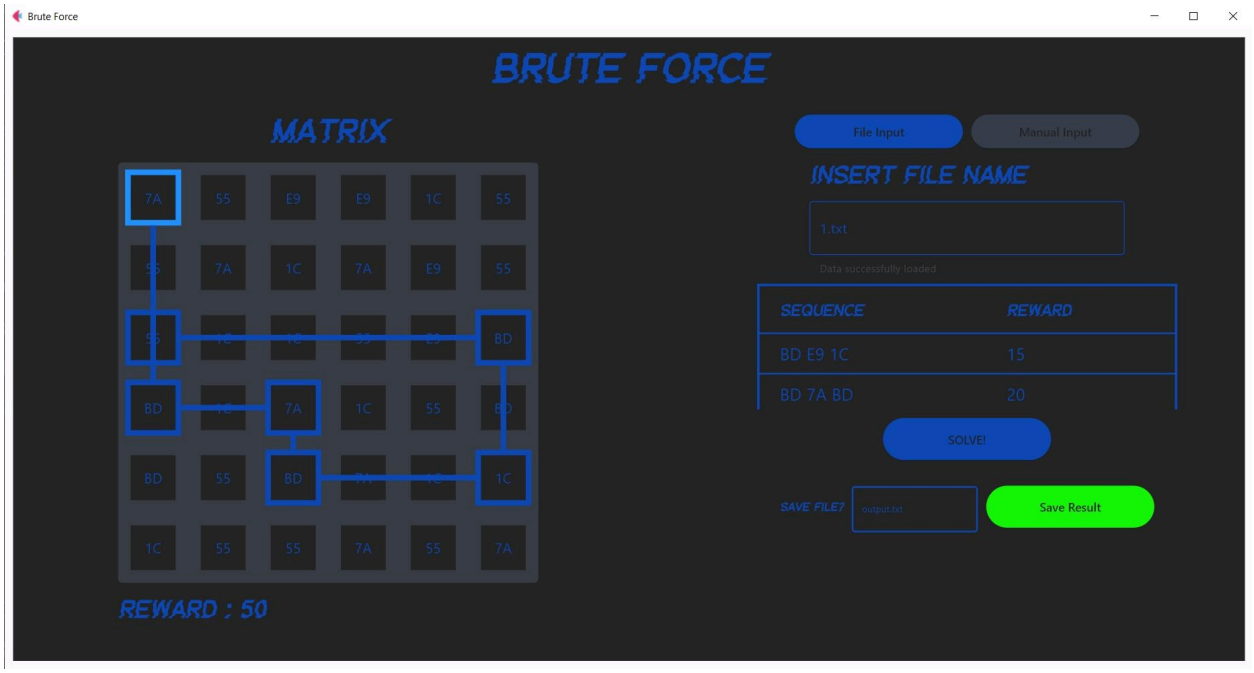
File 6_spasi.txt

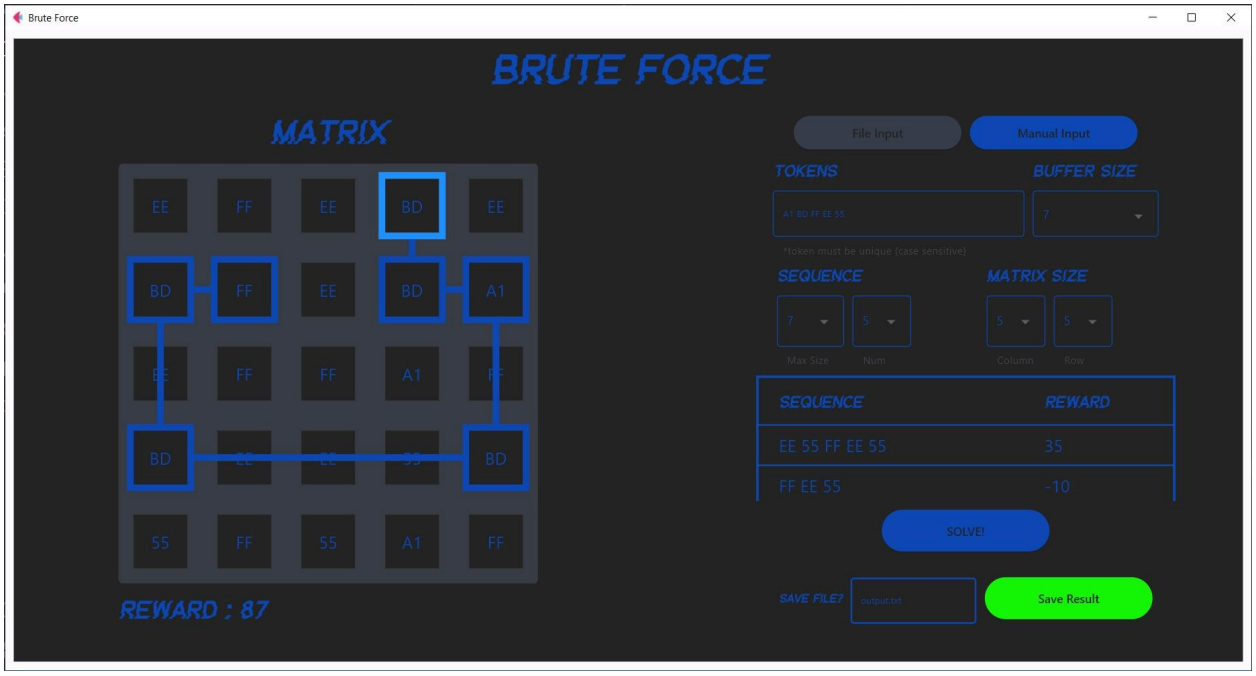


File Input & (Tidak Valid, Valid) & Save : n



Bab V
GUI


Brute Force
BRUTE FORCE
MATRIX
7A 55 E9 E9 1C 55
55 7A 1C 7A E9 55
55 1C 1C 55 E9 BD
BD 1C 7A 1C 55 ED
BD 55 BD 7A 1C 1C
1C 55 55 7A 55 7A
REWARD : 50
File Input
Manual Input
INSERT FILE NAME
1.txt
Data successfully loaded
SEQUENCE
REWARD
BD E9 1C
15
BD 7A BD
20
SOLVE!
SAVE FILE?
output.txt
Save Result


Brute Force
BRUTE FORCE
MATRIX
EE FF EE BD EE
BD FF EE BD A1
EE FF FF A1 FF
BD EE EE 55 BD
55 FF 55 A1 FF
REWARD : 87
File Input
Manual Input
TOKENS
BUFFER SIZE
A1 BD FF EE 55
7
*token must be unique (case sensitive)
SEQUENCE
MATRIX SIZE
7
5
5
5
Max Size
Num
Column
Row
SEQUENCE
REWARD
EE 55 FF EE 55
35
FF EE 55
-10
SOLVE!
SAVE FILE?
output.txt
Save Result

Lampiran

Github Repository

https://github.com/Zechtro/Tucil1_13522103

Tabel Spesifikasi

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	✓	