LAPORAN TUGAS KECIL 1
STRATEGI ALGORITMA

Disusun Oleh:
13522002 Ariel Herfrison

Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2024

A. Algoritma Brute Force

0. Menerima input dari user, baik dari file maupun secara acak.
1. Dengan menggunakan konsep *backtrack* dan rekursif, mencari semua kemungkinan sequence. Memanfaatkan 2 fungsi, yaitu *horizontal* dan *vertical*:
    a. Horizontal:
    Mengiterasi semua elemen dari suatu row untuk mencari semua kemungkinan sequence yang dimulai atau dilanjutkan dari elemen tersebut. Apabila belum mencapai basis, dilanjutkan dengan memanggil fungsi *vertical*.
    b. Vertical:
    Mengiterasi semua elemen dari suatu column untuk mencari semua kemungkinan sequence yang dimulai atau dilanjutkan dari elemen tersebut. Apabila belum mencapai basis, dilanjutkan dengan memanggil fungsi *horizontal*.

    Horizontal dan *vertical* memanfaatkan akumulator sequence. *Horizontal* dan *vertical* mencapai basis ketika akumulator sudah penuh (banyak sequence sudah mencapai ukuran buffer). Setelah mencapai basis, kedua fungsi mengembalikan reward dari sequence yang terakumulasi beserta sequencenya. Algoritma dimulai dengan memanggil fungsi *horizontal* pada row pertama (1).
2. Membandingkan *reward* antara semua kemungkinan sequence dengan cara membandingkan sequence yang dikembalikan oleh fungsi yang dipanggil dengan *reward* yang tersimpan oleh fungsi pemanggil. Menyimpan sequence dengan *reward* terbesar dan mengembalikannya sampai mencapai fungsi pertama.
3. Mengeluarkan solusi pada layar dan/atau file yang berupa sequence dengan reward terbesar.

B. Source Code

A. CLI

```
import os.path
import time
import random

input_type = input("Apakah ingin memasukkan input melalui file
atau secara acak?(file/acak) ")
while input_type != 'file' and input_type != 'acak':
    print("Invalid input")
     input_type = input("Apakah ingin memasukkan input melalui
file atau secara acak?(file/acak) ")

if input_type == 'file':
    file_name = input("Enter absolute path: ")
    while not os.path.isfile(file_name):
        print("File not found")
```

```
        file_name = input("Enter absolute path: ")

    file = open(file_name,'r')

    buffer_size = int(file.readline().rstrip())
    dimension = file.readline().rstrip().split()
    width = int(dimension[0])
    height = int(dimension[1])
       matrix  =  [file.readline().rstrip().split()  for  i  in
range(height)]
    sequences_amount = int(file.readline().rstrip())
    sequences = []
    sequences_reward = []
    for i in range(sequences_amount):
        sequences.append(file.readline().rstrip().split())
        sequences_reward.append(int(file.readline().rstrip()))

    file.close()

else:
    token_amount = int(input("Masukkan jumlah token unik: "))
   tokens = input("Masukkan token-token (terpisah dengan spasi):
").split()
    while len(tokens) != token_amount:
        print("Jumlah token tidak sesuai")
         tokens  =  input("Masukkan  token-token  (terpisah  dengan
spasi): ").split()
    buffer_size = int(input("Masukkan ukuran buffer: "))
    dimension = input("Masukkan ukuran matriks (width x height,
terpisah dengan spasi): ").split()
    width = int(dimension[0])
    height = int(dimension[1])
    sequences_amount = int(input("Masukkan jumlah sekuens: "))
     max_sequence_length  =  int(input("Masukkan  ukuran  maksimal
sekuens: "))

    matrix = [[random.choice(tokens) for i in range(width)] for
j in range(height)]
        sequences   =   [[random.choice(tokens)   for   i   in
range(random.randint(1,max_sequence_length))]    for    j    in
range(sequences_amount)]
       sequences_reward  =  [random.randint(1,50)  for  i  in
range(sequences_amount)]

    print(f"\nBuffer Size : {buffer_size}")
```

```
    print(f"Matrix Size : {width} x {height}")
    print(f"Matrix :")
    for i in range(height):
        print(f"    {matrix[i]}")
    print(f"Jumlah Sequence : {sequences_amount}")
    print(f"Sekuens dan Reward :")
    for i in range(sequences_amount):
        print(f"    {sequences[i]} - {sequences_reward[i]}")

# cek sequence unik (kalau tidak unik, reward = 0)
for i in range(sequences_amount-1):
    for j in range(i+1,sequences_amount):
        if len(sequences[i]) == len(sequences[j]):
            is_same = True
            k = 0
            while is_same and k < len(sequences[i]):
                if sequences[i][k] != sequences[j][k]:
                    is_same = False
                k += 1
            if is_same:
                sequences_reward[j] = 0

matrix_2 = [[1 for i in range(width)] for j in range(height)]

def checkReward(buffer):
    reward = 0
    for i in range(sequences_amount):
        has_reward = False
        j = 0
             while  not  has_reward  and  j  <  (len(buffer)-
len(sequences[i])+1):
            is_reward = True
            k = 0
            while is_reward and k < len(sequences[i]):
                if (buffer[j+k] != sequences[i][k]):
                    is_reward = False
                k += 1
            if is_reward:
                reward += sequences_reward[i]
                has_reward = True
            j += 1

    return reward

def horizontal(buffer_size,row,ctr,buffer,coor_buffer):
```

```
    max_reward = checkReward(buffer)
    max_buffer = buffer
    max_coor = coor_buffer
    if (ctr < buffer_size):

        for i in range (width):
            if (matrix_2[row][i] != 0):
                matrix_2[row][i] = 0
                buffer.append(matrix[row][i])
                coor_buffer.append(str(i+1)+','+str(row+1))
                new_buffer = buffer.copy()
                new_coor_buffer = coor_buffer.copy()
                                                    new_reward     =
vertical(buffer_size,i,ctr+1,new_buffer,new_coor_buffer)
                buffer.pop()
                coor_buffer.pop()
                matrix_2[row][i] = 1

                if new_reward[0] > max_reward:
                    max_reward = new_reward[0]
                    max_buffer = new_reward[1]
                    max_coor = new_reward[2]
                      elif  (new_reward[0]  ==  max_reward)  and
(len(new_reward[1])<len(max_buffer)):
                    max_buffer = new_reward[1]
                    max_coor = new_reward[2]

    return (max_reward,max_buffer,max_coor)

def vertical(buffer_size,column,ctr,buffer,coor_buffer):
    max_reward = checkReward(buffer)
    max_buffer = buffer
    max_coor = coor_buffer
    if (ctr < buffer_size):

        for i in range(height):
            if (matrix_2[i][column] != 0):
                matrix_2[i][column] = 0
                buffer.append(matrix[i][column])
                coor_buffer.append(str(column+1)+','+str(i+1))
                new_buffer = buffer.copy()
                new_coor_buffer = coor_buffer.copy()
                                                    new_reward     =
horizontal(buffer_size,i,ctr+1,new_buffer,new_coor_buffer)
                buffer.pop()
```

```
                coor_buffer.pop()
                matrix_2[i][column] = 1

                if new_reward[0] > max_reward:
                    max_reward = new_reward[0]
                    max_buffer = new_reward[1]
                    max_coor = new_reward[2]
                      elif (new_reward[0] == max_reward) and
(len(new_reward[1])<len(max_buffer)):
                    max_buffer = new_reward[1]
                    max_coor = new_reward[2]

    return (max_reward,max_buffer,max_coor)

print("\nSolusi:")
start = round(time.time()*1000)
max = horizontal(buffer_size,0,0,[],[])
print(max[0])
for tokens in max[1]:
    print(tokens,end=" ")
print()
for coors in max[2]:
    print(coors)
end = round(time.time()*1000)
print(f"\n{end-start} ms\n")

is_simpan = input("Apakah ingin menyimpan solusi?(y/n) ")

while (is_simpan != 'y' and is_simpan != 'n'):
    print("Invalid input")
    is_simpan = input("Apakah ingin menyimpan solusi?(y/n) ")

if is_simpan == 'y':
    file_output_name = input("Enter absolute path: ")

    file_output = open(file_output_name,'w')
    file_output.write(str(max[0])+'\n')
    for tokens in max[1]:
        file_output.write(tokens+" ")
    file_output.write('\n')
    for coors in max[2]:
        file_output.write(coors+'\n')

    file_output.close()
```

B. GUI

```
import os.path
import time
import random
import tkinter as tk
from tkinter import *
from tkinter.filedialog import askopenfile
from tkinter.filedialog import asksaveasfile

window = tk.Tk(className="python Cyberpunk 2077 Breach Protocol
Solver")
window.geometry("1024x576")
window.configure(background="#16151b")
window.resizable(False,False)

def func():
    buffer_size = int(buffer_entry.get().rstrip())
                matrix       =       matrix_entry.get('1.0','end-
1c').rstrip().split("\n")
    matrix = [line.split() for line in matrix]
    height = len(matrix)
    width = len(matrix[0])
              sequences      =      sequence_entry.get('1.0','end-
1c').rstrip().split("\n")
    sequences = [line.split() for line in sequences]
    sequences_amount = len(sequences)
           sequences_reward     =     reward_entry.get('1.0','end-
1c').rstrip().split("\n")
         sequences_reward    =    [int(amount)    for    amount    in
sequences_reward]

    # cek sequence unik (kalau tidak unik, reward = 0)
    for i in range(sequences_amount-1):
        for j in range(i+1,sequences_amount):
            if len(sequences[i]) == len(sequences[j]):
                is_same = True
                k = 0
                while is_same and k < len(sequences[i]):
                    if sequences[i][k] != sequences[j][k]:
                        is_same = False
                    k += 1
                if is_same:
                    sequences_reward[j] = 0

    matrix_2 = [[1 for i in range(width)] for j in range(height)]
```

```
    def checkReward(buffer):
        reward = 0
        for i in range(sequences_amount):
            has_reward = False
            j = 0
                 while  not  has_reward  and  j  <  (len(buffer)-
len(sequences[i])+1):
                is_reward = True
                k = 0
                while is_reward and k < len(sequences[i]):
                    if (buffer[j+k] != sequences[i][k]):
                        is_reward = False
                    k += 1
                if is_reward:
                    reward += sequences_reward[i]
                    has_reward = True
                j += 1

        return reward

    def horizontal(buffer_size,row,ctr,buffer,coor_buffer):
        max_reward = checkReward(buffer)
        max_buffer = buffer
        max_coor = coor_buffer
        if (ctr < buffer_size):

            for i in range (width):
                if (matrix_2[row][i] != 0):
                    matrix_2[row][i] = 0
                    buffer.append(matrix[row][i])
                    coor_buffer.append(str(i+1)+','+str(row+1))
                    new_buffer = buffer.copy()
                    new_coor_buffer = coor_buffer.copy()
                                                    new_reward    =
vertical(buffer_size,i,ctr+1,new_buffer,new_coor_buffer)
                    buffer.pop()
                    coor_buffer.pop()
                    matrix_2[row][i] = 1

                    if new_reward[0] > max_reward:
                        max_reward = new_reward[0]
                        max_buffer = new_reward[1]
                        max_coor = new_reward[2]
```

```
                        elif (new_reward[0] == max_reward) and
(len(new_reward[1])<len(max_buffer)):
                        max_buffer = new_reward[1]
                        max_coor = new_reward[2]

        return (max_reward,max_buffer,max_coor)

    def vertical(buffer_size,column,ctr,buffer,coor_buffer):
        max_reward = checkReward(buffer)
        max_buffer = buffer
        max_coor = coor_buffer
        if (ctr < buffer_size):

            for i in range(height):
                if (matrix_2[i][column] != 0):
                    matrix_2[i][column] = 0
                    buffer.append(matrix[i][column])
                  coor_buffer.append(str(column+1)+','+str(i+1))
                    new_buffer = buffer.copy()
                    new_coor_buffer = coor_buffer.copy()
                                                    new_reward    =
horizontal(buffer_size,i,ctr+1,new_buffer,new_coor_buffer)
                    buffer.pop()
                    coor_buffer.pop()
                    matrix_2[i][column] = 1

                    if new_reward[0] > max_reward:
                        max_reward = new_reward[0]
                        max_buffer = new_reward[1]
                        max_coor = new_reward[2]
                        elif (new_reward[0] == max_reward) and
(len(new_reward[1])<len(max_buffer)):
                        max_buffer = new_reward[1]
                        max_coor = new_reward[2]

        return (max_reward,max_buffer,max_coor)

    start = round(time.time()*1000)
    max = horizontal(buffer_size,0,0,[],[])
    end = round(time.time()*1000)
    duration = end-start

    popup = Toplevel(master=window,bg="#16151b")
    popup.geometry("450x440")
    popup.title("Solution")
```

```
    popup.resizable(False,False)

                                    text_frame                  =
tk.Frame(master=popup,height='100',bg="#16151b")
    text_frame.pack(fill='x',side='top')
    text_frame.pack_propagate(False)
                                     solution                   =
tk.Text(master=text_frame,bd=0,bg="#16151b",fg='#b3cc50')
    solution.pack(fill='x',padx=10,pady=10)
    solution.insert(END, f"Reward: {max[0]}\nSequence: ")
    for tokens in max[1]:
        solution.insert(END, f"{tokens} ")
    solution.insert(END, "\nSteps: ")
    for coors in max[2]:
        solution.insert(END, f"{coors} ")
    solution.config(state='disabled')

                                   matrix_frame                 =
tk.Frame(master=popup,height='250',bg="#16151b")
    matrix_frame.pack(fill='x',side='top')
    matrix_frame.pack_propagate(False)
                                    matrix_text                 =
tk.Text(master=matrix_frame,bd=0,font=("Bebas",18),bg="#16151b"
,fg='#b3cc50')
     matrix_text.pack(fill='both',padx=10,pady=10,side='top',exp
and=True)
    matrix_text.tag_config('highlight',background='#606e0c')
    for i in range(height):
        for j in range(width):
            if (str(j+1)+','+str(i+1)) not in max[2]:
                matrix_text.insert(END, f"{matrix[i][j]} ")
            else:
               matrix_text.insert(END, matrix[i][j],'highlight')
                matrix_text.insert(END, ' ')
        matrix_text.insert(END,'\n')
    matrix_text.config(state='disabled')
                                    time_frame                  =
tk.Frame(master=popup,height='50',width=450,bg="#16151b")
    time_frame.pack(side='top')
    time_frame.pack_propagate(False)
                                     time_text                  =
tk.Text(master=time_frame,bd=0,bg="#16151b",fg='#b3cc50')
    time_text.pack(padx=10,pady=10,side='left')
    time_text.insert(END, f"time: {duration} ms")
    time_text.config(state='disabled')
```

```
    def save():
        file = asksaveasfile(mode='w',defaultextension=".txt")
        if not file is None:
            file.write(str(max[0])+'\n')
            for tokens in max[1]:
                file.write(tokens+" ")
            file.write('\n')
            for coors in max[2]:
                file.write(coors+'\n')
        file.close()

        save_button   =   tk.Button(master=popup,text="Save   as
File",command=save)
    save_button.pack(padx=10,side='top',anchor='w')

def upload():
    file = askopenfile(mode='r')

    buffer_entry.delete(0,END)
    matrix_entry.delete('1.0',END)
    sequence_entry.delete('1.0',END)
    reward_entry.delete('1.0',END)

    buffer_entry.insert(END,file.readline().rstrip())
    dimension = file.readline().rstrip().split()
    for i in range(int(dimension[1])):
        matrix_entry.insert(END,file.readline())
    sequences_amount = int(file.readline().rstrip())
    for i in range(sequences_amount):
        sequence_entry.insert(END,file.readline())
        reward_entry.insert(END,file.readline())

    file.close()

def openRandomize():

    def randomize():
        token_amount = int(token_num_entry.get().rstrip())
        tokens = token_entry.get().rstrip().split()
        tokens = [tokens[i] for i in range(token_amount)]
        buffer_size = int(buffer_size_entry.get().rstrip())
        dimension = matrix_size_entry.get().rstrip().split()
        width = int(dimension[0])
        height = int(dimension[1])
```

```
        sequences_amount = int(sequence_num_entry.get().rstrip())
                                        max_sequence_length        =
int(sequence_max_entry.get().rstrip())
         matrix = [[random.choice(tokens) for i in range(width)]
for j in range(height)]
              sequences   =   [[random.choice(tokens)   for   i   in
range(random.randint(1,max_sequence_length))]    for    j    in
range(sequences_amount)]
           sequences_reward  =  [random.randint(1,50)  for  i  in
range(sequences_amount)]

         buffer_entry.delete(0,END)
         matrix_entry.delete('1.0',END)
         sequence_entry.delete('1.0',END)
         reward_entry.delete('1.0',END)

         buffer_entry.insert(END,buffer_size)

         for line in matrix:
             for element in line:
                 matrix_entry.insert(END,f"{element} ")
             matrix_entry.insert(END,"\n")

         for line in sequences:
             for element in line:
                 sequence_entry.insert(END,f"{element} ")
             sequence_entry.insert(END,"\n")

         for element in sequences_reward:
             reward_entry.insert(END,f"{element}\n")

         popup.destroy()

    popup = Toplevel(master=window,bg="#16151b")
    popup.geometry("400x400")
    popup.title("Randomize")
    popup.resizable(False,False)

     token_num  =  tk.Label(master=popup,text='Jumlah  Token  Unik:
')
    token_num.pack(padx=5,pady=(5,0))
    token_num_entry = tk.Entry(master=popup)
    token_num_entry.pack(padx=5,pady=(2,5))
    token = tk.Label(master=popup,text='Tokens: ')
    token.pack(padx=5,pady=(5,0))
```

```
    token_entry = tk.Entry(master=popup)
    token_entry.insert(END,"BD 1C 7A 55 E9")
    token_entry.pack(padx=5,pady=(2,5))
       buffer_size_label   =   tk.Label(master=popup,text='Ukuran
Buffer: ')
    buffer_size_label.pack(padx=5,pady=(5,0))
    buffer_size_entry = tk.Entry(master=popup)
    buffer_size_entry.pack(padx=5,pady=(2,5))
      matrix_size  =  tk.Label(master=popup,text='Ukuran  Matriks
(width x height): ')
    matrix_size.pack(padx=5,pady=(5,0))
    matrix_size_entry = tk.Entry(master=popup)
    matrix_size_entry.insert(END,"6 6")
    matrix_size_entry.pack(padx=5,pady=(2,5))
     sequence_num  =  tk.Label(master=popup,text='Jumlah  Sekuens:
')
    sequence_num.pack(padx=5,pady=(5,0))
    sequence_num_entry = tk.Entry(master=popup)
    sequence_num_entry.pack(padx=5,pady=(2,5))
     sequence_max  =  tk.Label(master=popup,text='Ukuran  Maksimal
Sekuens: ')
    sequence_max.pack(padx=5,pady=(5,0))
    sequence_max_entry = tk.Entry(master=popup)
    sequence_max_entry.pack(padx=5,pady=(2,5))

                            randomize_button_border             =
tk.Frame(master=popup,height=33,width=73,bg='#16151b',highlight
background='#b3cc50',highlightcolor='#b3cc50',highlightthicknes
s=3,bd=0)
    randomize_button_border.pack(pady=20)
    randomize_button_border.pack_propagate(False)
                                    randomize_button                =
tk.Button(master=randomize_button_border,command=randomize,bg='
#16151b',text='Randomize',fg='#b3cc50',font='Bebas',activebackg
round='#606e0c',bd=0)
    randomize_button.pack(expand=True)
    randomize_button.pack_propagate(False)

title = tk.Frame(master=window,height='100',bg="#16151b")
title.pack(fill=tk.X)
title.pack_propagate(False)
title_label = tk.Label(master=title,text="Cyberpunk 2077 Breach
Protocol Solver",bg="#16151b",fg="#d1ed5b",font=("Bebas",18))
title_label.pack(side="left",padx=20)
```

```
frame = tk.Frame(master=window,height='150',bg='#16151b')
frame.pack(fill=tk.X)
frame.pack_propagate(False)
buffer                                                          =
tk.Frame(master=frame,height='150',width='540',bg='#16151b')
buffer.pack(side='left')
buffer.pack_propagate(False)
bufferbox                                                       =
tk.LabelFrame(master=buffer,height=145,width=500,bg='#16151b',h
ighlightbackground='#b3cc50',highlightcolor='#b3cc50',highlight
thickness=3,bd=0)
bufferbox.pack(expand=True,anchor='w',padx=20,side='left')
bufferbox.pack_propagate(False)
buffer_title_label   =   tk.Label(master=bufferbox,text="Specify
Buffer Size",bg='#b3cc50',fg='#ffffff',font='Bebas',anchor='w')
buffer_title_label.pack(fill='x')
buffer_entry_text = tk.StringVar()
buffer_entry                                                    =
Entry(master=bufferbox,textvariable=buffer_entry_text,justify='
center')
buffer_entry_text.set(4)
buffer_entry.pack(pady=10)

button_frame                                                    =
tk.Frame(master=frame,height='150',width='484',bg='#16151b')
button_frame.pack()
button_frame.pack_propagate(False)
button_border                                                   =
tk.Frame(master=button_frame,height=53,width=103,bg='#16151b',h
ighlightbackground='#b3cc50',highlightcolor='#b3cc50',highlight
thickness=3,bd=0)
button_border.pack(side='left',padx=69.5)
button_border.pack_propagate(False)
solve_button                                                    =
tk.Button(master=button_border,command=func,height=50,width=100
,bg='#16151b',text='Solve',fg='#b3cc50',font='Bebas',activeback
ground='#606e0c',bd=0)
solve_button.pack(expand=True)
solve_button.pack_propagate(False)
button_border_2                                                 =
tk.Frame(master=button_frame,height=53,width=103,bg='#16151b',h
ighlightbackground='#b3cc50',highlightcolor='#b3cc50',highlight
thickness=3,bd=0)
button_border_2.pack(side='left',padx=69.5)
button_border_2.pack_propagate(False)
```

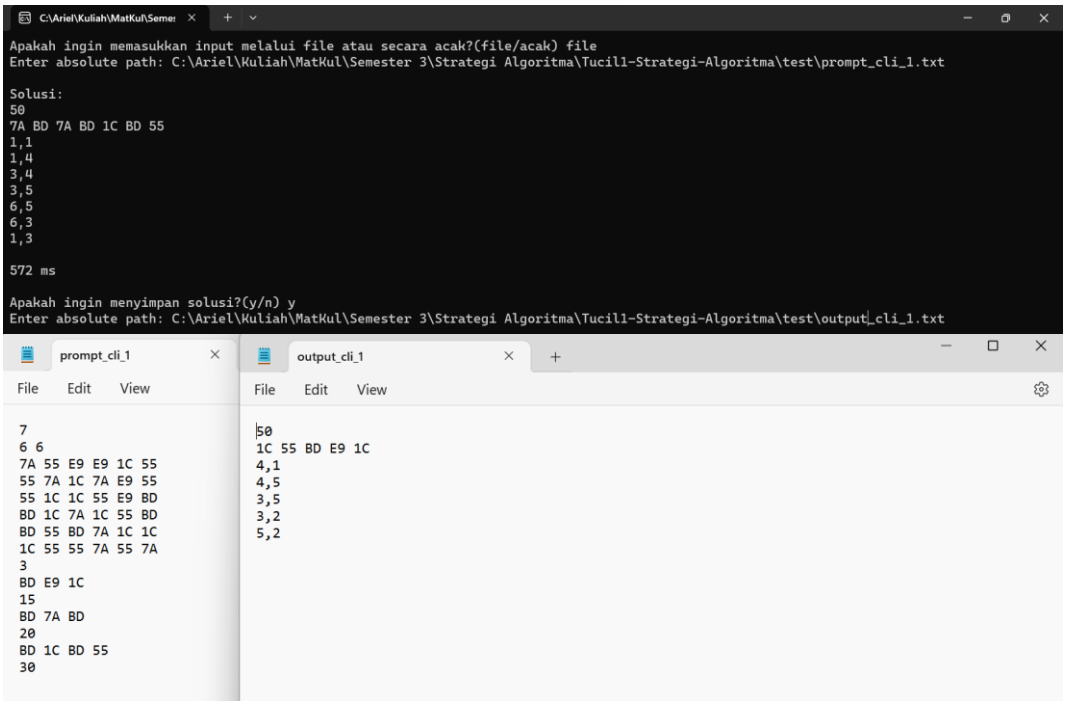
```
randomize_button                                                =
tk.Button(master=button_border_2,command=openRandomize,height=5
0,width=100,bg='#16151b',text='Randomize',fg='#b3cc50',font='Be
bas',activebackground='#606e0c',bd=0)
randomize_button.pack(expand=True)
randomize_button.pack_propagate(False)

matrix = tk.Frame(master=window,bg='#16151b')
matrix.pack(fill='both',expand=True)
matrix.pack_propagate(False)
matrixbox                                                       =
tk.LabelFrame(master=matrix,height=286,width=500,bg='#16151b',h
ighlightbackground='#b3cc50',highlightcolor='#b3cc50',highlight
thickness=3,bd=0)
matrixbox.pack(expand=True,anchor='w',padx=(20,10),side='left')
matrixbox.pack_propagate(False)
matrix_title   =   tk.Label(master=matrixbox,text="Enter   Matrix
Code",bg='#b3cc50',fg='#ffffff',font='Bebas',anchor='w')
matrix_title.pack(fill='x')
matrix_entry                                                    =
tk.Text(master=matrixbox,width=454,height=280,bg='#16151b',bd=0
,fg='#b3cc50',font='Bebas',insertbackground='#b3cc50')
matrix_entry.insert(END,"7A 55 E9 ...\n55 7A 1C ...\n...")
matrix_entry.pack(padx=10,pady=10)

sequencebox                                                     =
tk.LabelFrame(master=matrix,height=286,width=225,bg='#16151b',h
ighlightbackground='#b3cc50',highlightcolor='#b3cc50',highlight
thickness=3,bd=0)
sequencebox.pack(expand=True,anchor='w',padx=10,side='left')
sequencebox.pack_propagate(False)
sequence_title     =     tk.Label(master=sequencebox,text="Enter
Sequences",bg='#b3cc50',fg='#ffffff',font='Bebas',anchor='w')
sequence_title.pack(fill='x')
sequence_entry                                                  =
tk.Text(master=sequencebox,width=454,height=280,bg='#16151b',bd
=0,fg='#b3cc50',font='Bebas',insertbackground='#b3cc50')
sequence_entry.insert(END,"BD 55 7A\n...")
sequence_entry.pack(padx=10,pady=10)

rewardbox                                                       =
tk.LabelFrame(master=matrix,height=286,width=225,bg='#16151b',h
ighlightbackground='#b3cc50',highlightcolor='#b3cc50',highlight
thickness=3,bd=0)
rewardbox.pack(expand=True,anchor='w',padx=(10,20),side='left')
```

```
rewardbox.pack_propagate(False)
reward_title       =       tk.Label(master=rewardbox,text="Enter
Reward",bg='#b3cc50',fg='#ffffff',font='Bebas',anchor='w')
reward_title.pack(fill='x')
reward_title.pack_propagate(False)
reward_entry                                                    =
tk.Text(master=rewardbox,width=454,height=280,bg='#16151b',bd=0
,fg='#b3cc50',font='Bebas',insertbackground='#b3cc50')
reward_entry.insert(END,"15\n...")
reward_entry.pack(padx=10,pady=10)
reward_entry.pack_propagate(False)

upload_button    =    tk.Button(master=title,text="Import    from
File",command=upload)
upload_button.pack(side="left",padx=20)

window.mainloop()
```

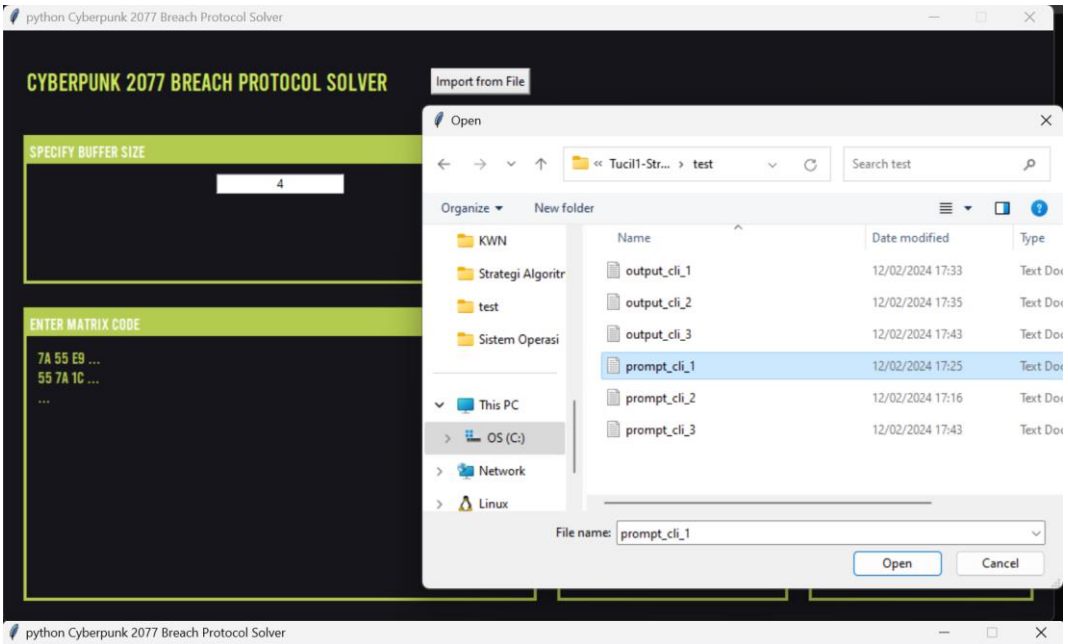
C. Uji Coba dan Hasil

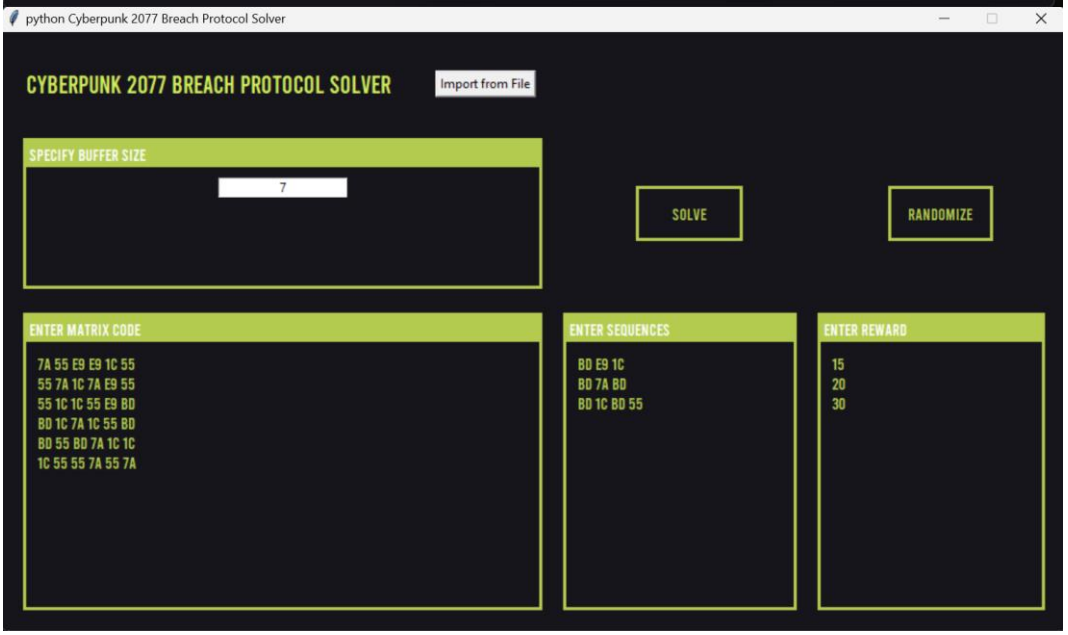
1. Test Case 1

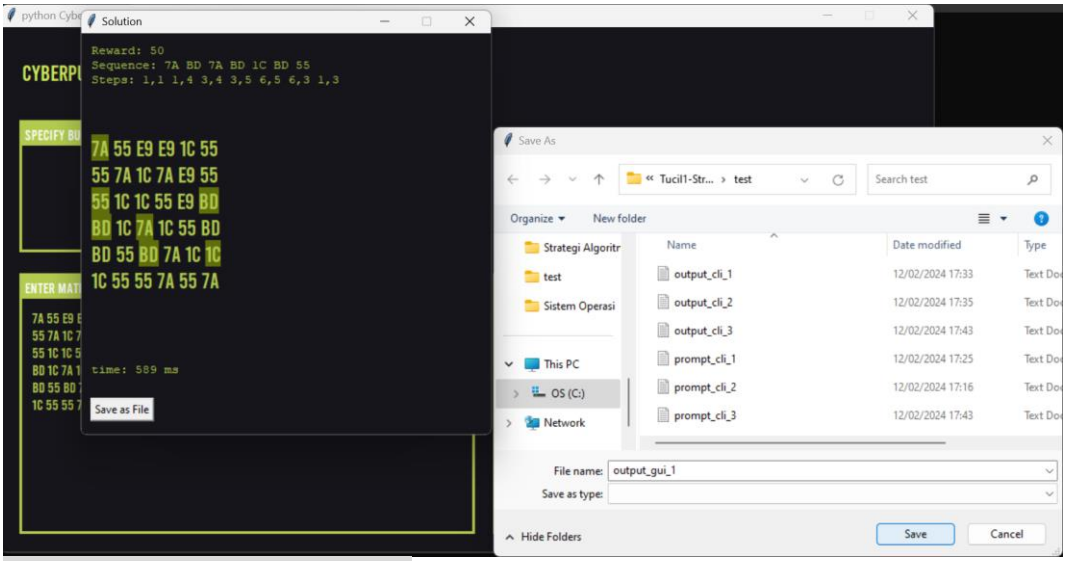
   CLI:

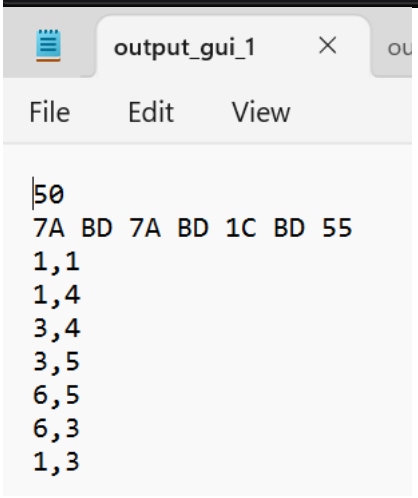


GUI:

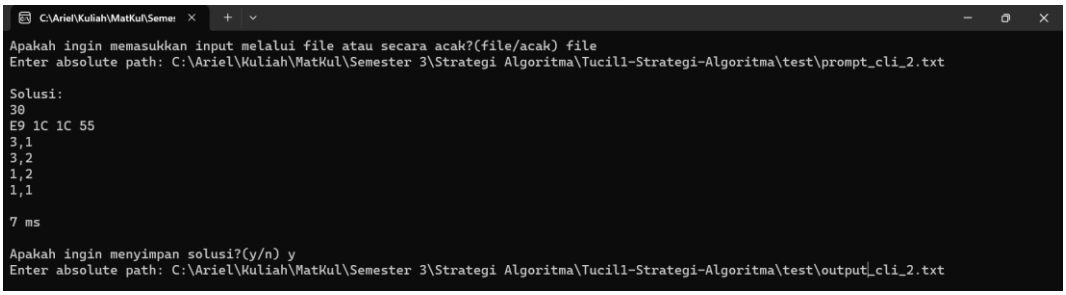

python Cyberpunk 2077 Breach Protocol Solver
CYBERPUNK 2077 BREACH PROTOCOL SOLVER
Import from File
SPECIFY BUFFER SIZE
4
ENTER MATRIX CODE
7A 55 E9 ...
55 7A 1C ...
...
Open
Tucil1-Str... › test
Search test
Organize
New folder
KWN
Strategi Algoritr
test
Sistem Operasi
This PC
OS (C:)
Network
Linux
Name
Date modified
Type
output_cli_1
12/02/2024 17:33
output_cli_2
12/02/2024 17:35
output_cli_3
12/02/2024 17:43
prompt_cli_1
12/02/2024 17:25
prompt_cli_2
12/02/2024 17:16
prompt_cli_3
12/02/2024 17:43
File name: prompt_cli_1
Open
Cancel


python Cyberpunk 2077 Breach Protocol Solver
CYBERPUNK 2077 BREACH PROTOCOL SOLVER
Import from File
SPECIFY BUFFER SIZE
7
SOLVE
RANDOMIZE
ENTER MATRIX CODE
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
ENTER SEQUENCES
BD E9 1C
BD 7A BD
BD 1C BD 55
ENTER REWARD
15
20
30





2. Test Case 2
CLI:



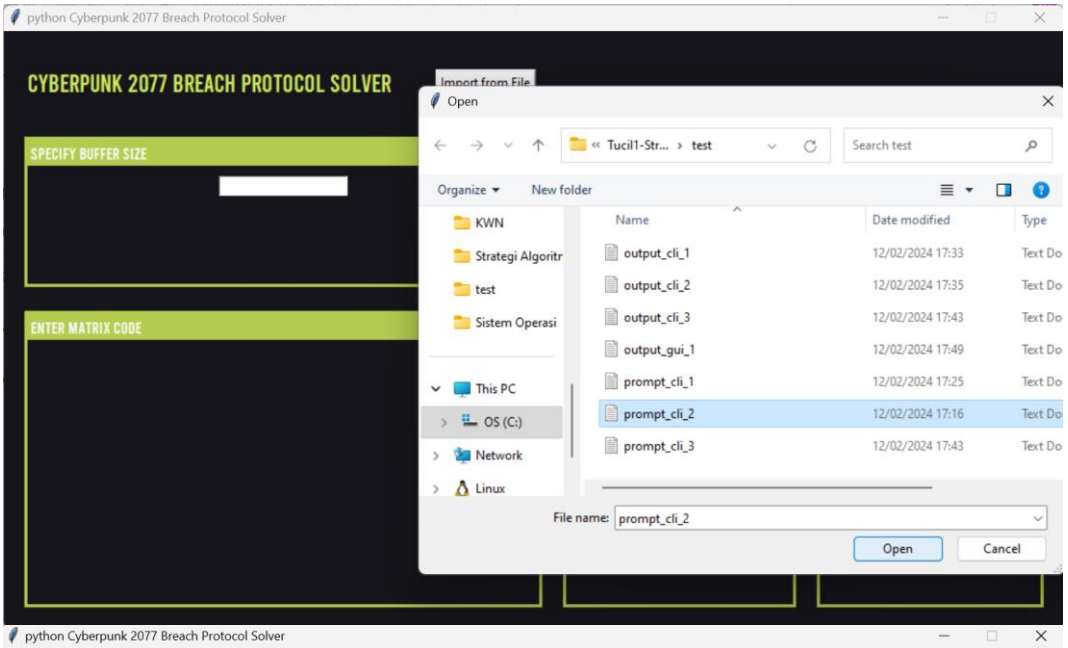
prompt_cli_2

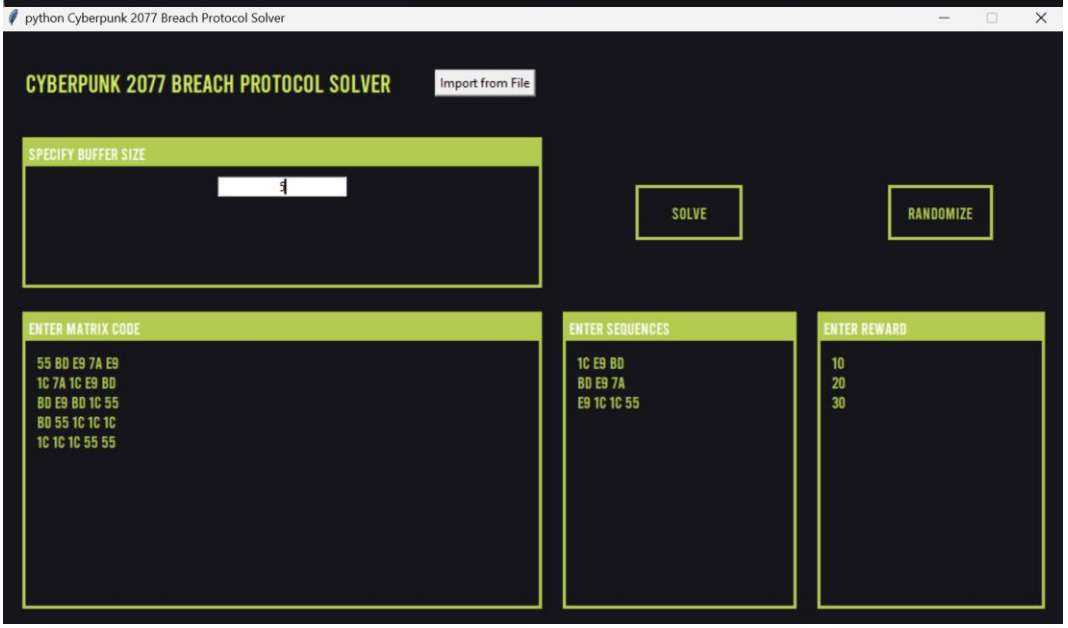
```
5
5 5
55 BD E9 7A E9
1C 7A 1C E9 BD
BD E9 BD 1C 55
BD 55 1C 1C 1C
1C 1C 1C 55 55
3
1C E9 BD
10
BD E9 7A
20
E9 1C 1C 55
30
```

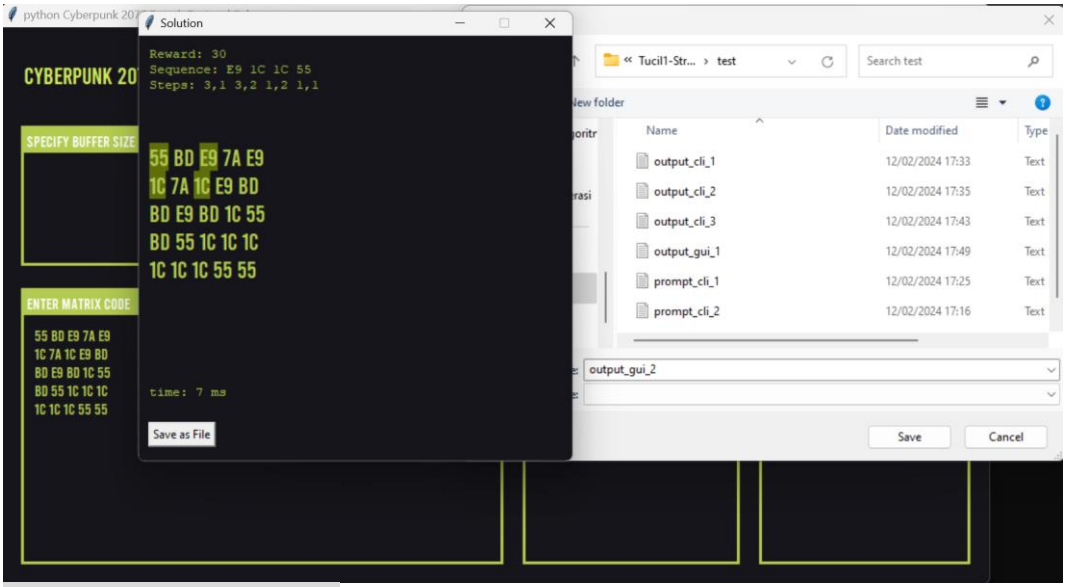
output_cli_2

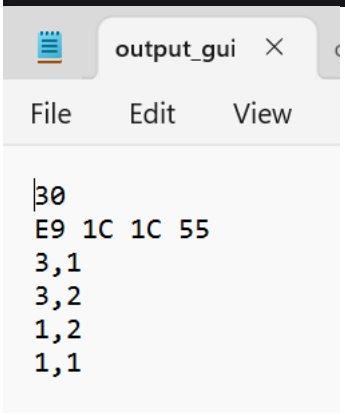
```
30
E9 1C 1C 55
3,1
3,2
1,2
1,1
```

GUI:

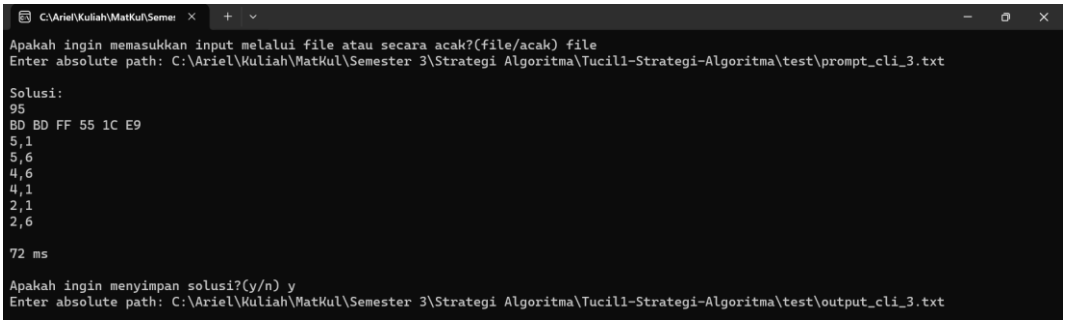


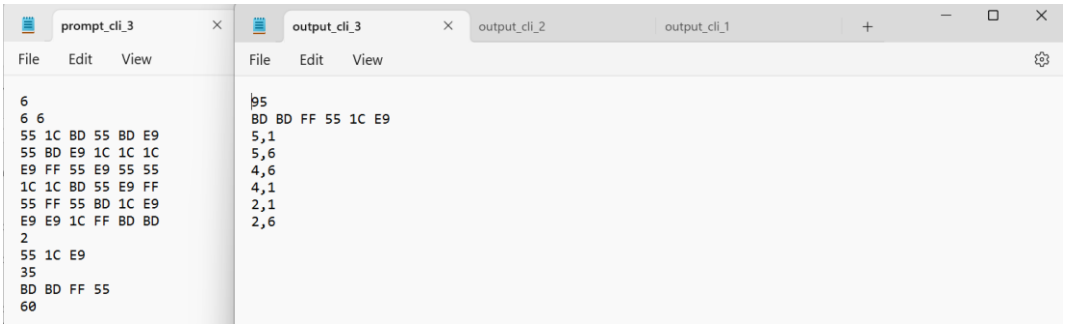




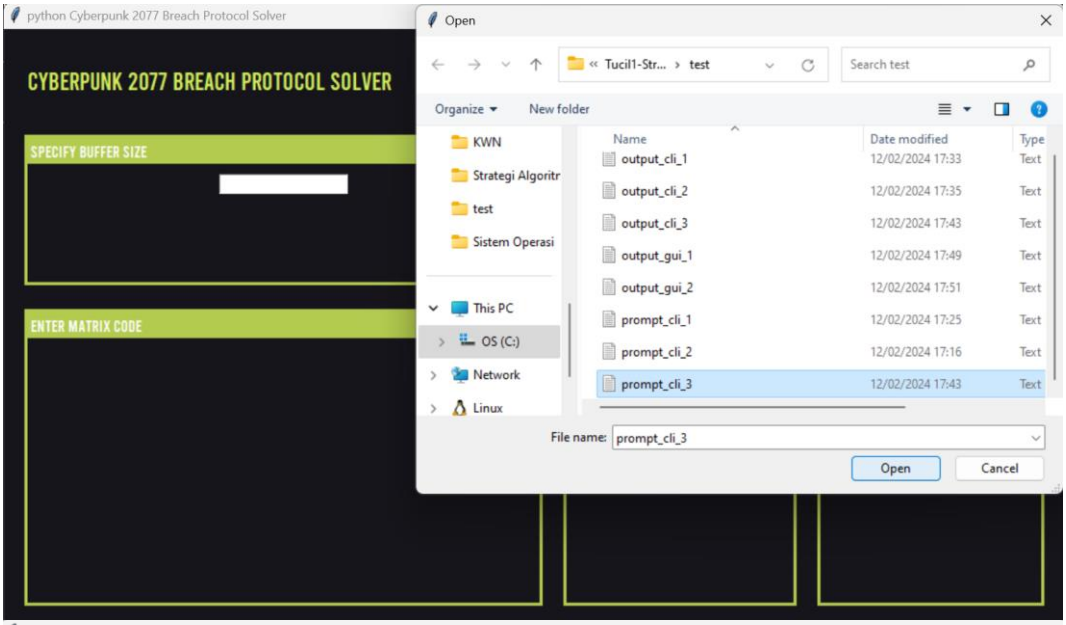


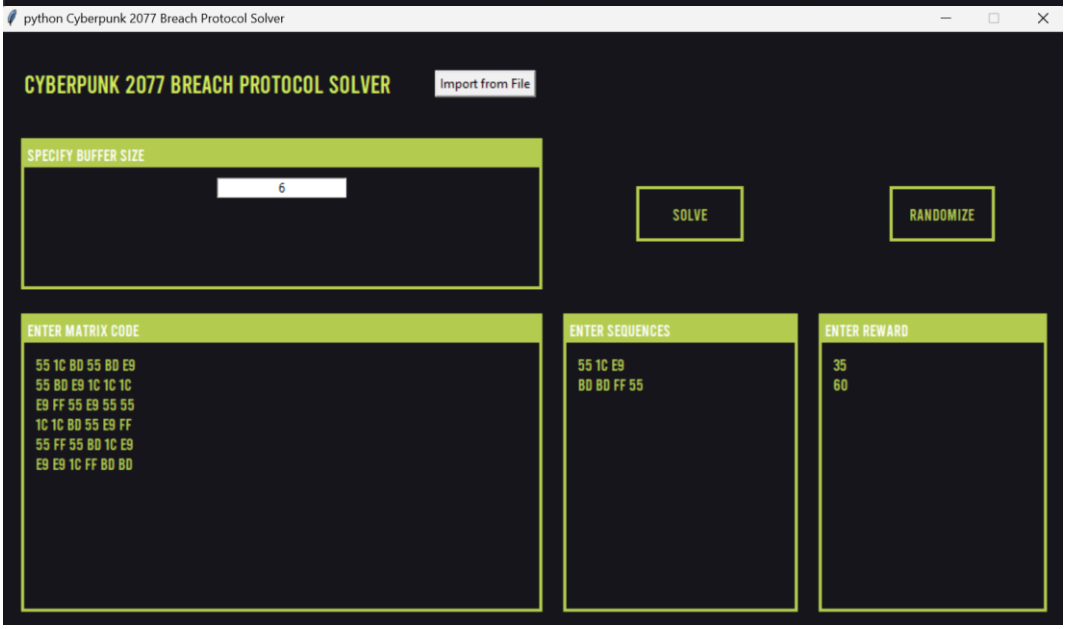
3. Test Case 3
CLI:

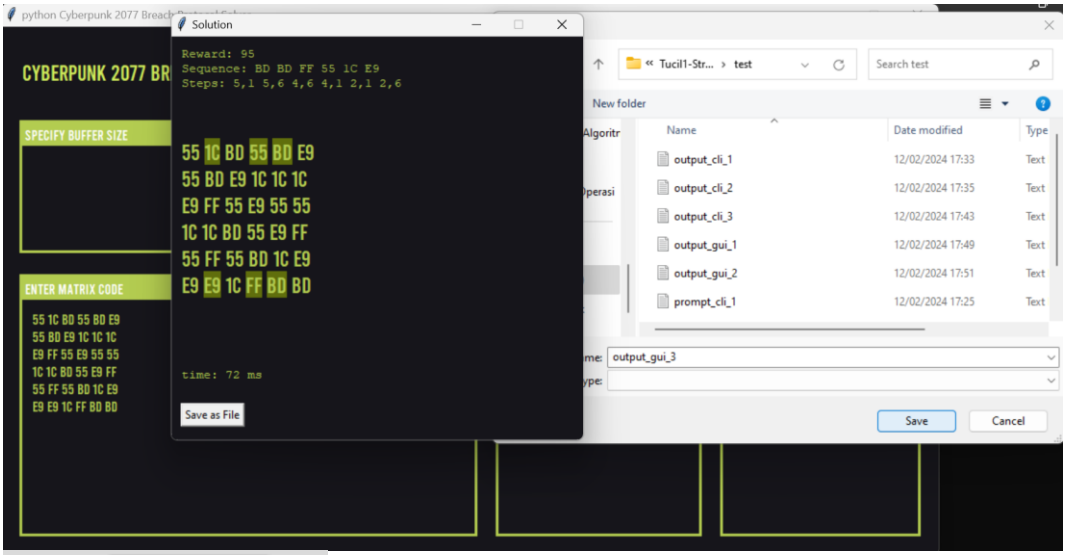


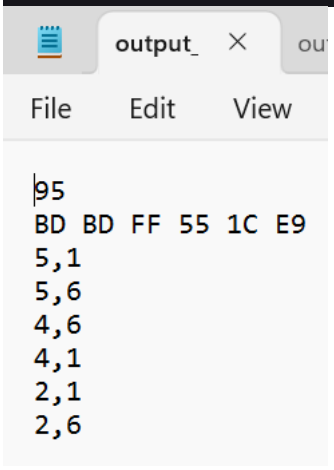

prompt_cli_3
File Edit View
6
6 6
55 1C BD 55 BD E9
55 BD E9 1C 1C 1C
E9 FF 55 E9 55 55
1C 1C BD 55 E9 FF
55 FF 55 BD 1C E9
E9 E9 1C FF BD BD
2
55 1C E9
35
BD BD FF 55
60
output_cli_3
output_cli_2
output_cli_1
File Edit View
95
BD BD FF 55 1C E9
5,1
5,6
4,6
4,1
2,1
2,6

GUI:

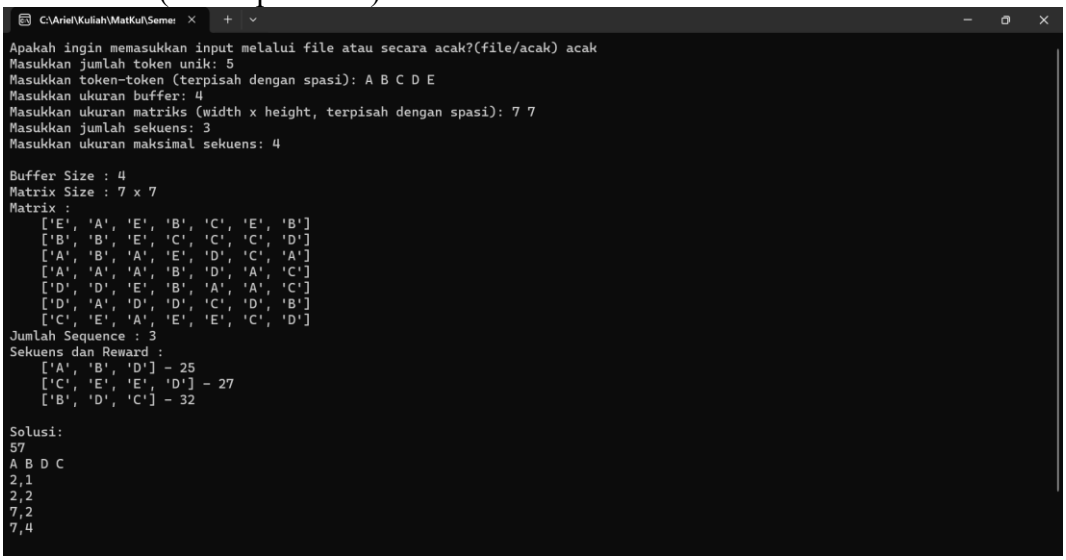

python Cyberpunk 2077 Breach Protocol Solver
CYBERPUNK 2077 BREACH PROTOCOL SOLVER
SPECIFY BUFFER SIZE
ENTER MATRIX CODE
Open
Tucil1-Str... › test
Search test
Organize
New folder
KWN
Strategi Algoritr
test
Sistem Operasi
This PC
OS (C:)
Network
Linux
Name
Date modified
Type
output_cli_1 12/02/2024 17:33 Text
output_cli_2 12/02/2024 17:35 Text
output_cli_3 12/02/2024 17:43 Text
output_gui_1 12/02/2024 17:49 Text
output_gui_2 12/02/2024 17:51 Text
prompt_cli_1 12/02/2024 17:25 Text
prompt_cli_2 12/02/2024 17:16 Text
prompt_cli_3 12/02/2024 17:43 Text
File name: prompt_cli_3
Open
Cancel

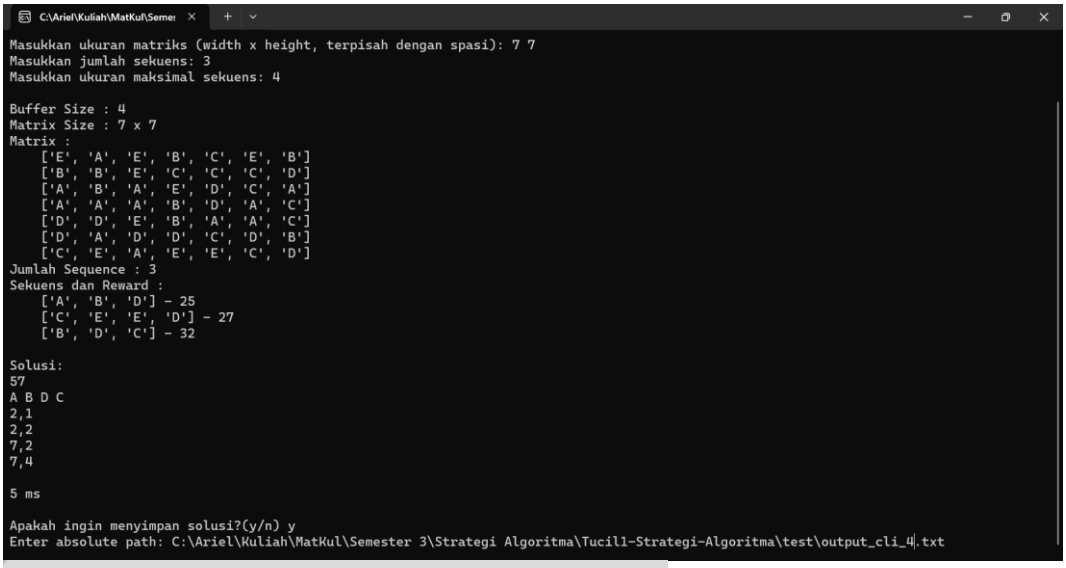

python Cyberpunk 2077 Breach Protocol Solver
CYBERPUNK 2077 BREACH PROTOCOL SOLVER
Import from File
SPECIFY BUFFER SIZE
6
SOLVE
RANDOMIZE
ENTER MATRIX CODE
55 1C BD 55 BD E9
55 BD E9 1C 1C 1C
E9 FF 55 E9 55 55
1C 1C BD 55 E9 FF
55 FF 55 BD 1C E9
E9 E9 1C FF BD BD
ENTER SEQUENCES
55 1C E9
BD BD FF 55
ENTER REWARD
35
60

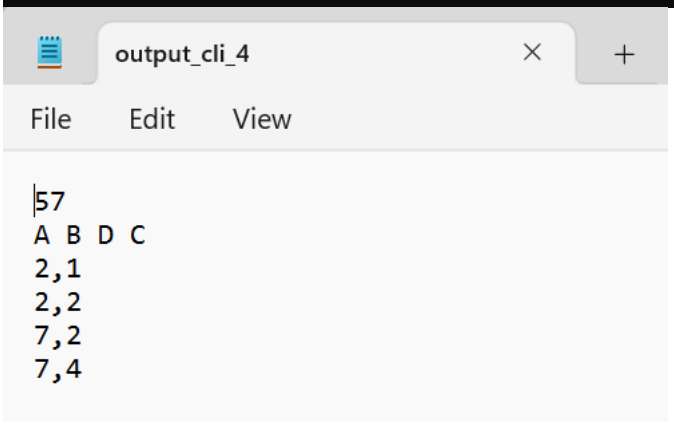




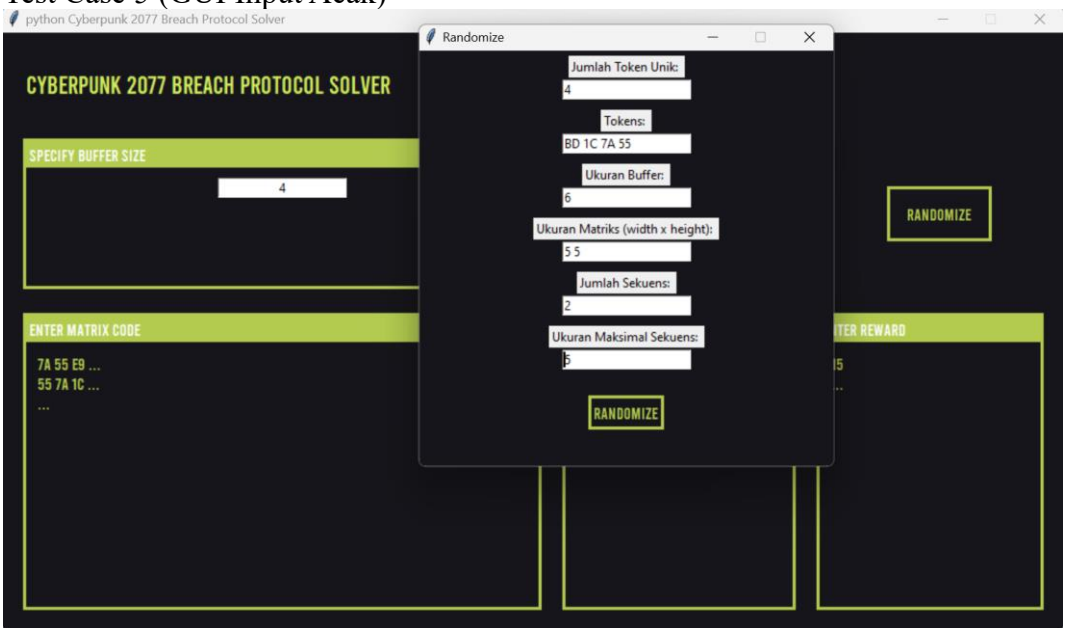
4. Test Case 4 (CLI Input Acak)

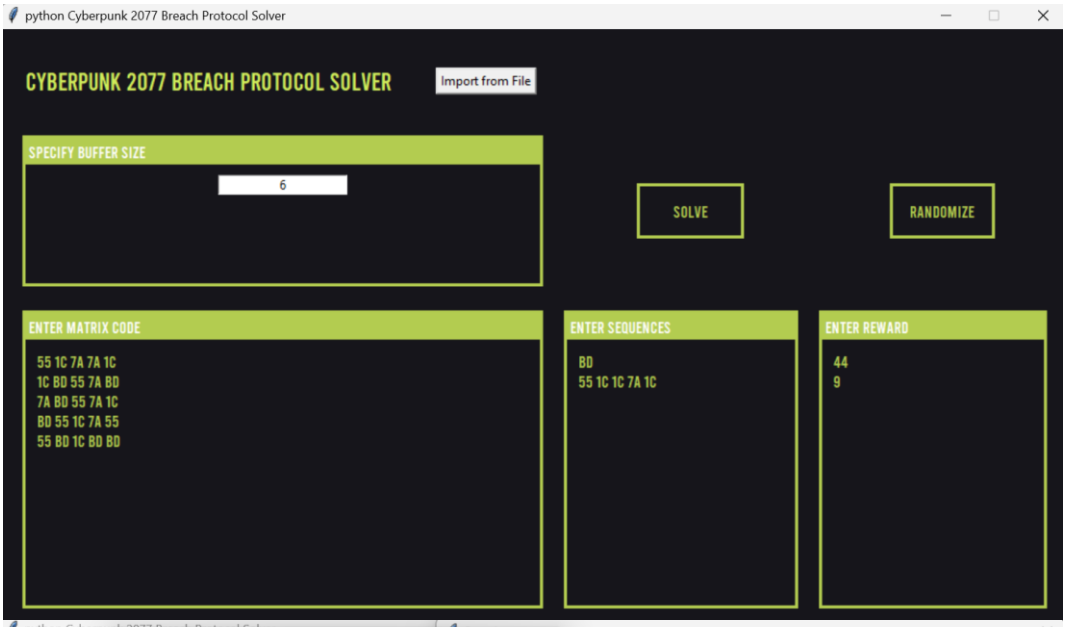


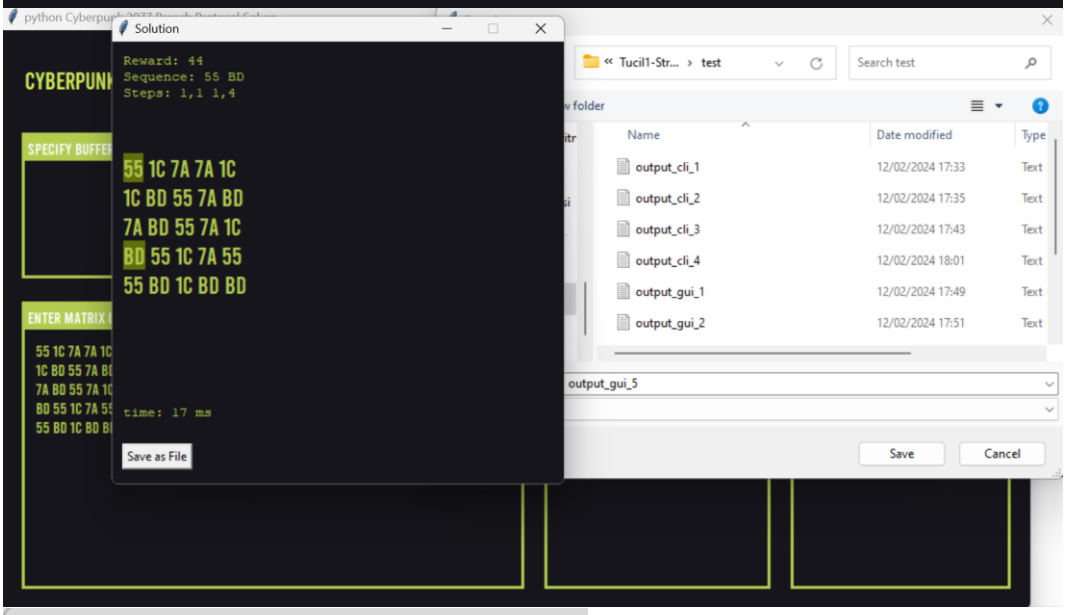


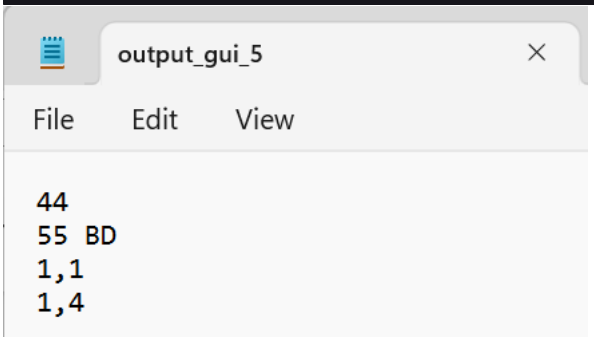


5. Test Case 5 (GUI Input Acak)

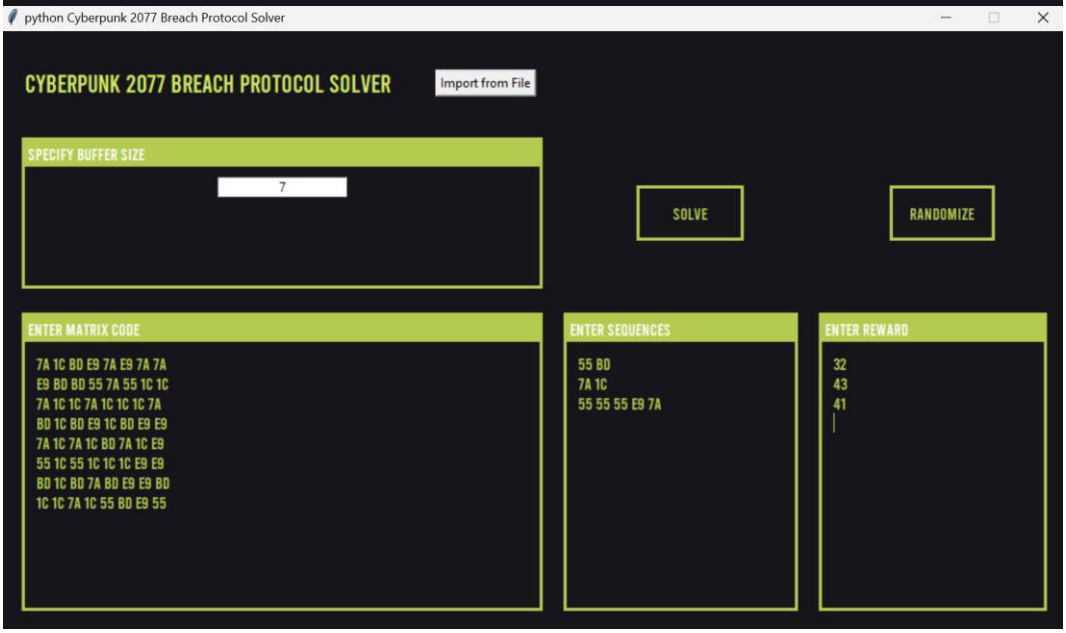


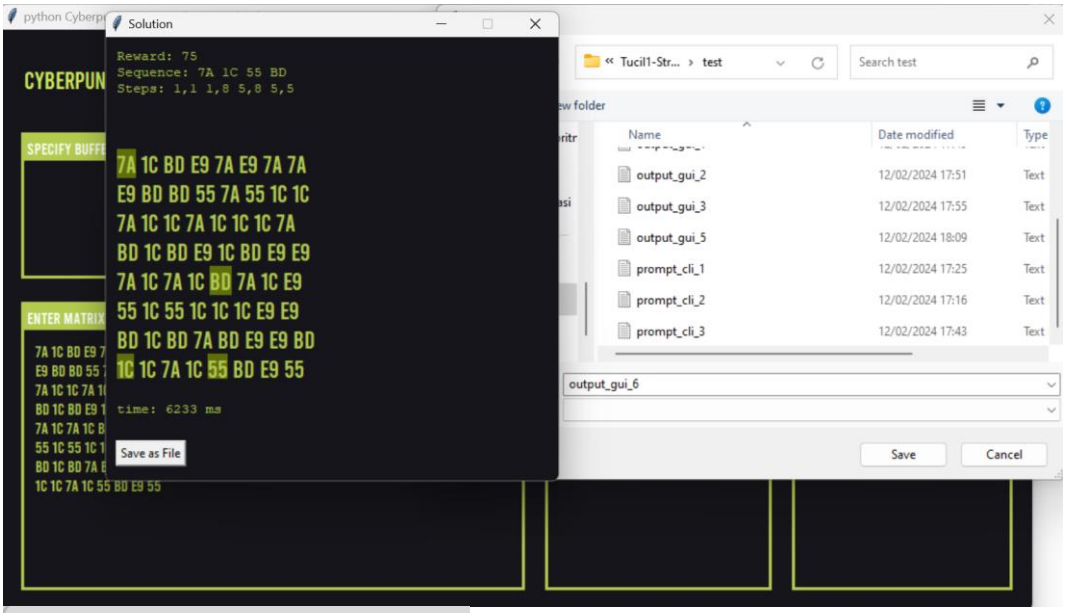

python Cyberpunk 2077 Breach Protocol Solver
CYBERPUNK 2077 BREACH PROTOCOL SOLVER
Import from File
SPECIFY BUFFER SIZE
6
SOLVE
RANDOMIZE
ENTER MATRIX CODE
55 1C 7A 7A 1C
1C BD 55 7A BD
7A BD 55 7A 1C
BD 55 1C 7A 55
55 BD 1C BD BD
ENTER SEQUENCES
BD
55 1C 1C 7A 1C
ENTER REWARD
44
9

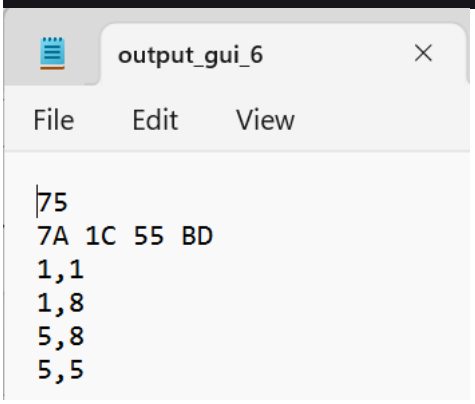

Solution
Reward: 44
Sequence: 55 BD
Steps: 1,1 1,4
55 1C 7A 7A 1C
1C BD 55 7A BD
7A BD 55 7A 1C
BD 55 1C 7A 55
55 BD 1C BD BD
time: 17 ms
Save as File
Tucil1-Str... › test
Search test
Name
Date modified
Type
output_cli_1
output_cli_2
output_cli_3
output_cli_4
output_gui_1
output_gui_2
12/02/2024 17:33
12/02/2024 17:35
12/02/2024 17:43
12/02/2024 18:01
12/02/2024 17:49
12/02/2024 17:51
Text
output_gui_5
Save
Cancel


output_gui_5
File
Edit
View
44
55 BD
1,1
1,4

6. Test Case 6 (GUI Input Acak)







Pranala Repository

https://github.com/Ariel-HS/Tucil1-Strategi-Algoritma.git

Checklist

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	✓	