TUGAS KECIL 1
IF2211 STRATEGI ALGORITMA

Penyelesaian Cyberpunk 2077 Breach Protocol Dengan Algoritma

Brute Force

Disusun oleh :

M. Hanief Fatkhan Nashrullah   13522100

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023

Daftar Isi

BAB I
Algoritma Brute Force dalam Breach Protocol

Breach protocol merupakan minigame yang menjadi salah satu mekanik dalam permainan video Cyberpunk 2077. *Minigame* ini memiliki empat komponen utama, yaitu token, dua karakter alfanumerik, kemudian ada *code matrix*, rangkaian token berbentuk matrix yang akan dicocokkan dengan *sequence*, lalu ada *sequence*, rangkaian yang terdiri dari lebih dari satu token, dan *buffer* sebagai batas jumlah token yang dapat digunakan secara sekuensial.

Minigame ini memiliki aturan sebagai berikut :

1. Pencocokkan harus dimulai dari token yang ada pada baris pertama.
2. Token (jalur) yang digunakan harus bergantian dari vertikal dan horizontal dengan jalur pertama dari baris pertama bergerak secara vertikal.
3. Token yang digunakan tidak harus bersebelahan, token bisa digunakan selama tidak bertentangan dengan aturan no.2 dan berada pada baris atau kolom yang sama.
4. Token yang sudah diambil tidak bisa diambil lagi dalam satu kali pencocokan.
5. *Sequence* dicocokkan pada token yang telah digunakan pada *buffer*.
6. Satu *buffer* bisa digunakan oleh beberapa *sequence*.
7. Token yang digunakan tidak boleh melebihi ukuran *buffer*.
8. Setiap *sequence* memiliki bobot *reward* yang dapat bervariasi.

Gambar 1 Minigame Breach Protocol
(sumber : https://cyberpunk.fandom.com/wiki/Quickhacking?file=Breach_Protocol_Information_Screenshot_02.png)

Algoritma *brute force* merupakan algoritma yang cukup sederhana. Algoritma ini memerika satu per satu dari seluruh kemungkinan yang ada untuk mencari solusi yang optimal. Algoritma ini akan digunakan untuk menyelesaikan *Minigame Breach Protocol* tersebut. Langkah langkahnya adalah :

1. Periksa apakah *buffer* yang digunakan sudah penuh dengan token atau belum. Jika sudah maka tidak ada token yang dapat diambil lagi. Jika belum, dapat lanjut ke langkah selanjutnya.
2. Periksa seluruh token dan koordinat yang dapat digunakan dari posisi koordinat awal.
3. Jika ada token yang digunakan, cocokkan dengan *sequence* yang tersedia.
4. Jika terdapat *sequence* yang cocok, periksa apakah *reward* yang diberikan lebih besar dari *reward* awal (default *reward* = 0).
5. Ulangi langkah no.1 dengan posisi koordinat terakhir yang digunakan.

6. Jika sudah tidak ada koordinat dan token yang dapat digunakan dari koordinat terakhir, kembali ke koordinat sebelumnya (*backtrack)* dan lanjut ke koordinat dan token lain.
7. Algoritma berakhir saat semua kemungkinan sudah diperiksa.

BAB II
Source Program

Program ini ditulis dengan menggunakan bahasa pemrograman Python
Daftar Module/Library :
- Tkinter
- CustomTkinter
- OS
- sys
- Random
- datetime
- webbrowser

Source code :

```python
# Program Cyberpunk 2077 Breach Protocol Solver dengan Brute Force

# M. Hanief Fatkhan Nashrullah
# 13522100
# 11/02/2024

# Desc : Sebuah program untuk mencari solusi dari minigame meretas pada permainan video Cyberpunk 2077.
#        Minigame ini merupakan simulasi peretasan jaringan local dari ICE
#        (Intrusion Countermeasures Electronics) pada permainan Cyberpunk 2077.
#        *+GUI :D


# Import Library
import customtkinter
import tkinter
import os
import sys
import random
import webbrowser
from datetime import datetime

# OS Check
if os.name=="nt":
    exe_path = sys.argv[0]
    os.chdir(os.path.dirname(exe_path))
    file_path = os.getcwd()
else:
    file_path = os.getcwd()

# Default Color Theme
customtkinter.set_appearance_mode("dark")
customtkinter.set_default_color_theme("dark-blue")

# Batasan Input Token
ALPHANUMERIC = {'a','b','c','d','e','f',
                'A','B','C','D','E','F',
                '0','1','2','3','4','5',
                '6','7','8','9','0'}

NUMBER = {'0','1','2','3','4','5','6','7','8','9','0'}
```

```python
# App
class App(customtkinter.CTk):

    def __init__(self):

        # Tab Random Input
        def randomizer():
            self.mainProgram = customtkinter.CTkFrame(self,fg_color="#141414",width=500,corner_radius=0)
            self.mainProgram.grid(row=0,column=1,sticky="nsew")
            global CUSTOM
            CUSTOM = False

            # Unique Token Count
            self.label_uniqueTokenNum = customtkinter.CTkLabel(
                self.mainProgram,
                text="Unique Token Count",
                font=("Terminal",20))
            self.label_uniqueTokenNum.grid(row=0,column=0,padx=(20,0),pady=(30,10),sticky="w")
            self.entry_uniqueTokenNum = customtkinter.CTkEntry(self.mainProgram, width=200,font=("Terminal",14))
            self.entry_uniqueTokenNum.grid(row=1,column=0,padx=(20,0),pady=(0,10),sticky="w")

            # Unique Token
            self.label_uniqueToken = customtkinter.CTkLabel(
                self.mainProgram,
                text="Unique Token",
                font=("Terminal",20))
            self.label_uniqueToken.grid(row=2,column=0,padx=(20,0),pady=(30,10),sticky="w")
            self.entry_uniqueToken = customtkinter.CTkEntry(self.mainProgram, width=200,font=("Terminal",14))
            self.entry_uniqueToken.grid(row=3,column=0,padx=(20,0),pady=(0,10),sticky="w")

            # Buffer Size
            self.label_bufferSize = customtkinter.CTkLabel(
                self.mainProgram,
                text="Buffer Size",
                font=("Terminal",20))
            self.label_bufferSize.grid(row=4,column=0,padx=(20,0),pady=(30,10),sticky="w")
            self.entry_bufferSize = customtkinter.CTkEntry(self.mainProgram, width=200,font=("Terminal",14))
            self.entry_bufferSize.grid(row=5,column=0,padx=(20,0),pady=(0,10),sticky="w")

            # Unique Sequence Count
            self.label_uniqueSequenceCount = customtkinter.CTkLabel(
                self.mainProgram,
                text="Unique Sequence Count",
                font=("Terminal",20))
            self.label_uniqueSequenceCount.grid(row=0,column=1,padx=(75,0),pady=(30,10),sticky="w")
            self.entry_uniqueSequenceCount = customtkinter.CTkEntry(self.mainProgram, width=200,font=("Terminal",14))
            self.entry_uniqueSequenceCount.grid(row=1,column=1,padx=(75,0),pady=(0,10),sticky="w")

            # Max Sequence Size
            self.label_maxSequenceSize = customtkinter.CTkLabel(
                self.mainProgram,
                text="Max Sequence Size",
                font=("Terminal",20))
            self.label_maxSequenceSize.grid(row=2,column=1,padx=(75,0),pady=(30,10),sticky="w")
            self.entry_maxSequenceSize = customtkinter.CTkEntry(self.mainProgram, width=200,font=("Terminal",14))
            self.entry_maxSequenceSize.grid(row=3,column=1,padx=(75,0),pady=(0,10),sticky="w")

            # Matrix Size
            self.label_codeMatrixSize = customtkinter.CTkLabel(
                self.mainProgram,
                text="Code Matrix Size",
                font=("Terminal",20))
            self.label_codeMatrixSize.grid(row=4,column=1,padx=(75,0),pady=(30,10),sticky="w")
            self.entry_codeMatrixSizeRow = customtkinter.CTkEntry(self.mainProgram, width=50,font=("Terminal",14))
            self.entry_codeMatrixSizeRow.grid(row=5,column=1,padx=(75,0),pady=(0,10),sticky="w")
            self.label_times = customtkinter.CTkLabel(
                self.mainProgram,
                text="X",
                font=("Terminal",20))
            self.label_times.grid(row=5,column=1,padx=(135,0),pady=(0,10),sticky="w")
            self.entry_codeMatrixSizeCol = customtkinter.CTkEntry(self.mainProgram, width=50,font=("Terminal",14))
            self.entry_codeMatrixSizeCol.grid(row=5,column=1,padx=(155,0),pady=(0,10),sticky="w")

            # Solve Button
            self.button_solve = customtkinter.CTkButton(self.mainProgram,text="Solve",font=
("Terminal",14),command=solution)
            self.button_solve.grid(row=6,column=0,sticky="sw",pady=(50,0),padx=(22.5,0))

        # Custom Value Tab / Import Txt Tab
        def customVal():
            global CUSTOM
            CUSTOM = True
            self.mainProgram = customtkinter.CTkFrame(self,fg_color="#141414",width=500,corner_radius=0)
            self.mainProgram.grid(row=0,column=1,sticky="nsew")

            # Enter Code Matrix
            self.label_codeMatrix = customtkinter.CTkLabel(
                self.mainProgram,
                text="Enter Code Matrix",
                font=("Terminal",20))
            self.label_codeMatrix.grid(row=0,column=0,padx=(20,0),pady=(30,10),sticky="w")
            self.entry_codeMatrix = customtkinter.CTkTextbox(self.mainProgram, width=200,font=("Terminal",14))
            self.entry_codeMatrix.grid(row=1,column=0,padx=(20,0),pady=(0,10),sticky="w")

            # Enter Sequence
            self.label_sequence = customtkinter.CTkLabel(
                self.mainProgram,
                text="Enter Sequences & Rewards",
                font=("Terminal",20))
            self.label_sequence.grid(row=0,column=1,padx=(40,0),pady=(30,10),sticky="w")
            self.entry_sequence = customtkinter.CTkTextbox(self.mainProgram, width=200,font=("Terminal",14))
            self.entry_sequence.grid(row=1,column=1,padx=(40,0),pady=(0,10),sticky="w")
```

```python
        # Buffer Size
        self.label_bufferSize = customtkinter.CTkLabel(
            self.mainProgram,
            text="Buffer Size",
            font=("Terminal",20))
        self.label_bufferSize.grid(row=2,column=0,padx=(20,0),pady=(30,10),sticky="w")
        self.entry_bufferSize = customtkinter.CTkEntry(self.mainProgram, width=200,font=("Terminal",14))
        self.entry_bufferSize.grid(row=3,column=0,padx=(20,0),pady=(0,10),sticky="w")

        # Matrix Size
        self.label_codeMatrixSize = customtkinter.CTkLabel(
            self.mainProgram,
            text="Code Matrix Size",
            font=("Terminal",20))
        self.label_codeMatrixSize.grid(row=2,column=1,padx=(50,0),pady=(30,10),sticky="w")
        self.entry_codeMatrixSizeRow = customtkinter.CTkEntry(self.mainProgram, width=50,font=("Terminal",14))
        self.entry_codeMatrixSizeRow.grid(row=3,column=1,padx=(50,0),pady=(0,10),sticky="w")
        self.label_times = customtkinter.CTkLabel(
            self.mainProgram,
            text="X",
            font=("Terminal",20))
        self.label_times.grid(row=3,column=1,padx=(110,0),pady=(0,10),sticky="w")
        self.entry_codeMatrixSizeCol = customtkinter.CTkEntry(self.mainProgram, width=50,font=("Terminal",14))
        self.entry_codeMatrixSizeCol.grid(row=3,column=1,padx=(130,0),pady=(0,10),sticky="w")

        # Open Button
        self.button_openfile = customtkinter.CTkButton(self.mainProgram,text="Open txt file",font=
("Terminal",14),command=opentxtFiles)
        self.button_openfile.grid(row=4,column=0,sticky="sw",pady=(50,0),padx=(20,0))

        # Solve Button
        self.button_solve = customtkinter.CTkButton(self.mainProgram,text="Solve",font=
("Terminal",14),command=solution)
        self.button_solve.grid(row=4,column=1,sticky="sw",pady=(50,0),padx=(50,0))

    # Open Txt Prompt
    def opentxtFiles():
        self.filename = tkinter.filedialog.askopenfilename(initialdir=file_path,title="Select a *.txt
file",filetypes=([("Text Documents","*.txt")]))
        file = open(self.filename,"r")
        buffersize = int(file.readline())
        rowAndCol= file.readline().split(" ")
        matrixRow = int(rowAndCol[0])
        matrixCol = int(rowAndCol[1])

        matrixRow_var = tkinter.StringVar()
        matrixRow_var.set(matrixRow)

        matrixCol_var = tkinter.StringVar()
        matrixCol_var.set(matrixCol)

        buffer_size_var = tkinter.StringVar()
        buffer_size_var.set(buffersize)

        self.entry_bufferSize.configure(textvariable=buffer_size_var)
        self.entry_codeMatrixSizeCol.configure(textvariable=matrixCol_var)
        self.entry_codeMatrixSizeRow.configure(textvariable=matrixRow_var)

        matrix =""
        for i in range(matrixRow):
            matrix += file.readline()
        self.entry_codeMatrix.delete("1.0", tkinter.END)
        self.entry_codeMatrix.insert("1.0", matrix)

        sequence =""
        for sequences in file:
            sequence += sequences
        self.entry_sequence.delete("1.0", tkinter.END)
        self.entry_sequence.insert("1.0", sequence)
        file.close()

    # Input Validation
    def string_validation(alphabet,length,text,equalLength):
        words = text.split(" ")
        notEmpty = False
        if len(words)==1:
            return False
        for word in words:
            if word != '':
                notEmpty = True
            if equalLength:
                if len(word)!=length:
                    return False
            for letter in word:
                if letter not in alphabet:
                    return False
        return notEmpty

    # Save to File Prompt
    def saveToFile():
        global STRING_TEMP
        try:
            save_path = tkinter.filedialog.asksaveasfilename(initialdir=file_path,title="Save
As",defaultextension=".txt",filetypes=([("Text Documents","*.txt")]))
            save_writer = open(save_path,"w")
            save_writer.write(STRING_TEMP)
            save_writer.close()
        except:
            print("exception")
            pass
```

```python
# Custom Input Getter
def customInput():
    global ALPHANUMERIC
    valid = True
    try:
        bufferSize = int(self.entry_bufferSize.get())
        matrixRow = int(self.entry_codeMatrixSizeRow.get())
        matrixCol = int(self.entry_codeMatrixSizeCol.get())
        if matrixCol<2 or matrixRow<2:
            valid = False
        matrix = [[""]*matrixCol]*matrixRow
        matrixTemp = str(self.entry_codeMatrix.get("0.0",customtkinter.END)).split("\n")
        while '' in matrixTemp:
            matrixTemp.remove('')
        if matrixRow!=len(matrixTemp):
            valid = False
        for i in range(matrixRow):
            currInput = matrixTemp[i]
            if len(currInput.split(" "))!=matrixCol or string_validation(ALPHANUMERIC,0,currInput,False)==False:
                valid = False
            else:
                matrix[i] = currInput.split(" ")
        sequenceTemp = str(self.entry_sequence.get("0.0",customtkinter.END)).split("\n")
        numberOfSequence = int(sequenceTemp[0])
        sequence = [["", 0] for _ in range(numberOfSequence)]
        for i in range(1,numberOfSequence*2+1,2):
            sequence[int((i-1)/2)][0] = sequenceTemp[i].upper()
            if(string_validation(ALPHANUMERIC,0,sequenceTemp[i],False)==False):
                valid = False
        for i in range(2,numberOfSequence*2+1,2):
            sequence[int((i-1)/2)][1] = int(sequenceTemp[i])

        for i in range(len(matrix)):
            for j in range(len(matrix[0])):
                print(matrix[i][j],end=" ")
            print("")
        print(sequence)
        print(bufferSize)
        print(valid)
        return matrix,sequence,bufferSize,valid
    except:
        return [],[],0,False

# Random Input Generator
def randomInput():
    global ALPHANUMERIC
    global NUMBER
    valid = True
    try:
        numberOfUniqueToken = int(self.entry_uniqueTokenNum.get())
        tokenString = str(self.entry_uniqueToken.get())
        if (not string_validation(ALPHANUMERIC,2,tokenString,True)) or (len(tokenString.split("
"))!=numberOfUniqueToken):
            print("wrong token")
            valid = False
            raise Exception
        matrixRow = int(self.entry_codeMatrixSizeRow.get())
        matrixCol = int(self.entry_codeMatrixSizeRow.get())
        if matrixRow<2 or matrixCol<2:
            valid = False
        numberOfSequence = int(self.entry_uniqueSequenceCount.get())
        sequenceMaxSize = int(self.entry_maxSequenceSize.get())
        if sequenceMaxSize<2:
            valid = False
        bufferSize = int(self.entry_bufferSize.get())
        if bufferSize<2:
            valid = False
        tokenArray = tokenString.upper().split(" ")

        matrix = [[0] * matrixCol for i in range(matrixRow)]
        sequence = [["", 0] for _ in range(numberOfSequence)]

        for i in range(numberOfSequence):
            randomnum = random.randint(2, sequenceMaxSize)
            seq = ""
            for j in range(randomnum):
                seq += random.choice(tokenArray) + " "
            sequence[i] = [seq.strip(), 0]
            sequence[i][1] = random.randint(-100,100)
            for k in sequence:
                while sequence[i][0] in k[0] and i!=sequence.index(k):
                    sequence[i][0] = sequence[i][0][:3]
                    sequence[i][0] += random.choice(tokenArray)

        for i in range(matrixRow):
            for j in range(matrixCol):
                matrix[i][j] = tokenArray[random.randint(0,numberOfUniqueToken-1)]
    except:
        print("Other invalid input")
        valid = False
        return [],[],0,False

    return matrix,sequence,bufferSize,valid

# My Youtube Channel (pls don't click :D)
def yt_link():
    webbrowser.open_new("https://www.youtube.com/watch?v=W_czMS95ngY")
```

```python
    # Depth First Search Algorithm
    def dfs(r,c,d,m,horizontal):
        global MATR
        global ROWS
        global COLS
        global PATH
        global ELPATH
        global OPT_PATH
        global OPT_TOKEN
        global SUBSTRING
        global MAXPOINT
        currentPoint = 0
        if d==m:
            return
        if(r<0 or c<0 or r>=ROWS or c>=COLS or (r,c) in PATH):
            return
        PATH.append((r,c))
        ELPATH+=MATR[r][c]+" "
        for element in SUBSTRING:
            if element[0] in ELPATH:
                currentPoint += element[1]

        if currentPoint==MAXPOINT:
            if len(PATH)<len(OPT_PATH):
                OPT_PATH = PATH[:]
                OPT_TOKEN = ELPATH
        elif currentPoint>MAXPOINT:
            MAXPOINT -= MAXPOINT
            MAXPOINT += currentPoint
            OPT_PATH = PATH[:]
            OPT_TOKEN = ELPATH
        if horizontal:
            if  c < len(MATR[0])-c :
                for i in range(len(MATR[0])):
                    dfs(r,i,d+1,m,False)
            else:
                for i in range(len(MATR[0])-1,0,-1):
                    dfs(r,i,d+1,m,False)
        else:
            if  r < len(MATR)-r :
                for i in range(len(MATR)):
                    dfs(i,c,d+1,m,True)
            else:
                for i in range(len(MATR)-1,0,-1):
                    dfs(i,c,d+1,m,True)
        PATH.remove((r,c))
        ELPATH = ELPATH[:-3]
        return

    # Display Solution
    def solution():
        global MATR
        global ROWS
        global COLS
        global PATH
        global ELPATH
        global OPT_PATH
        global OPT_TOKEN
        global SUBSTRING
        global MAXPOINT
        global CUSTOM
        global STRING_TEMP
        MATR = 0
        PATH =[]
        ELPATH =""
        OPT_PATH = ()
        OPT_TOKEN = ""
        RUNTIME = 0
        MAXPOINT = 0
        if CUSTOM:
            print("Custom/File Input")
            MATR,SUBSTRING,BUFFERSIZE,valid = customInput()
        else:
            print("Randomized Input")
            MATR,SUBSTRING,BUFFERSIZE,valid = randomInput()
        if valid:
            ROWS = len(MATR)
            COLS = len(MATR[0])
            solution_window = customtkinter.CTkToplevel(self,fg_color="#141414")
            x = int((self.winfo_screenwidth()/2)-(450/2))
            y = int((self.winfo_screenheight()/2)-(800/2))
            solution_window.geometry(f"450x800+{x}+{y}")
            solution_window.title("Result")
            matrix_frame = customtkinter.CTkFrame(solution_window,fg_color="#141414",height=225)
            matrix_frame.grid(row=0,column=0,sticky="nsew")
            solution_frame =
customtkinter.CTkFrame(solution_window,fg_color="#141414",height=225)
            start=datetime.now()
            for i in range(COLS):
                dfs(0,i,0,BUFFERSIZE,False)

            for x in SUBSTRING:
                print(f"\033[38;5;227m{x[0]}\033[0m\033[38;5;87m ({x[1]})\033[0m")

            print((f"\033[38;5;227mMATRIX : \033[0m"))
            for i in range(ROWS):
                for j in range(COLS):
                    if (i,j) in OPT_PATH :
                        print(f"\033[38;5;87m{MATR[i][j]}\033[0m", end=" ")
                    else:
                        print(f"\033[38;5;227m{MATR[i][j]}\033[0m", end=" ")
                print("")

            print(f"\033[38;5;87m{MAXPOINT}\033[0m")
            print(f"\033[38;5;87m{OPT_TOKEN}\033[0m")
            for y,x in OPT_PATH:
                print(f"\033[38;5;227m{x+1},{y+1}\033[0m")
            stop = datetime.now()
            RUNTIME = stop-start
            print(f"\033[38;5;87m{round(RUNTIME.total_seconds()*1000)} ms\033[0m")
```

```python
                STRING_TEMP = ""
                STRING_TEMP += str(MAXPOINT)+"\n"
                STRING_TEMP += str(OPT_TOKEN)+"\n"
                for y,x in OPT_PATH:
                    STRING_TEMP += str(x+1)+","+str(y+1)+"\n"
                STRING_TEMP += "\n"+str(round(RUNTIME.total_seconds()*1000))+" ms"
                for i in range(ROWS):
                    for j in range(COLS):
                        if (i,j) in OPT_PATH:
                            color = "E6E6E6"
                        else:
                            color = "7F7F7F"
                        matrix_solution = customtkinter.CTkLabel(
                            matrix_frame,
                            text=MATR[i][j].upper()+" ",
                            text_color=f"#{color}",
                            font=("Terminal",16))
                        if j==0:
                            right = 80
                        else:
                            right = 0
                        if i==0:
                            top = 40
                        else:
                            top = 0
                        matrix_solution.grid(row=i, column=j,padx=(right,0),pady=(top,0),sticky="nsew")

                sequence_title = customtkinter.CTkLabel(
                        solution_frame,
                        text="List of sequences : ",
                        text_color=f"#FFFFFF",
                        font=("Terminal",12),
                        )
                sequence_title.grid(row=0, column=0,padx=(40,0),pady=(20,0),sticky="w")
                for i in range(len(SUBSTRING)):
                    sequence_list = customtkinter.CTkLabel(
                            solution_frame,
                            text=SUBSTRING[i][0]+" "+"("+str(SUBSTRING[i][1])+")\n",
                            text_color=f"#FFFFFF",
                            font=("Terminal",12),
                            )
                    if i==0:
                        sequence_list.grid(row=i+1, column=0,padx=(40,0),pady=(0,0),sticky="w")
                    else:
                        sequence_list.grid(row=i+1, column=0,padx=(40,0),pady=(0,0),sticky="w")
                opt_path_label = customtkinter.CTkLabel(
                        solution_frame,
                        text="Optimal sequence : "+str(OPT_TOKEN),
                        text_color=f"#FFFFFF",
                        font=("Terminal",12))
                opt_path_label.grid(row=len(SUBSTRING)+2, column=0,padx=(40,0),pady=(0,0),sticky="w")
                point_label = customtkinter.CTkLabel(
                        solution_frame,
                        text="Max points : "+str(MAXPOINT),
                        text_color=f"#FFFFFF",
                        font=("Terminal",12))
                point_label.grid(row=len(SUBSTRING)+1, column=0,padx=(40,0),pady=(0,0),sticky="w")
                for i in range(len(OPT_PATH)):
                    coord = customtkinter.CTkLabel(
                            solution_frame,
                            text=str(OPT_PATH[i][1]+1)+","+str(OPT_PATH[i][0]+1)+"\n",
                            text_color=f"#FFFFFF",
                            font=("Terminal",12))
                    coord.grid(row=len(SUBSTRING)+3+i, column=0,padx=(40,0),pady=(0,0),sticky="w")
                runtime_label = customtkinter.CTkLabel(
                        solution_frame,
                        text="Execution time : "+str(round(RUNTIME.total_seconds()*1000))+" ms",
                        text_color=f"#FFFFFF",
                        font=("Terminal",12))
                runtime_label.grid(row=len(SUBSTRING)+len(OPT_PATH)+4, column=0,padx=(40,0),pady=(0,0),sticky="w")

                save_button = customtkinter.CTkButton(solution_frame,text="Save Result",font=
("Terminal",14),command=saveToFile)
                save_button.grid(row=len(SUBSTRING)+len(OPT_PATH)+5, column=0,padx=(40,0),pady=(20,0),sticky="w")
                solution_window.attributes("-topmost",True)
                solution_window.grab_set()
                solution_window.mainloop()


            else:
                invalid_window = customtkinter.CTkToplevel(self)
                invalid_window.title("")
                x = int((self.winfo_screenwidth()/2)-(250/2))
                y = int((self.winfo_screenheight()/2)-(100/2))
                invalid_window.geometry(f"250x100+{x}+{y}")
                invalid_window.maxsize(250,100)
                invalid_window.minsize(250,100)
                invalid_window.attributes("-topmost",True)
                invalid_window_label = customtkinter.CTkLabel(invalid_window,text="Input Invalid!",pady=50,font=
("Terminal",20))
                invalid_window_label.pack()
                invalid_window.grab_set()
                invalid_window.mainloop()
            try:
                self.grab_release()
            except:
                pass
```

```python
        # About Me Pop-up Window
        def about_me():
            about_me_window = customtkinter.CTkToplevel(self)
            about_me_window.title("About Me")
            x = int((self.winfo_screenwidth()/2)-(500/2))
            y = int((self.winfo_screenheight()/2)-(125/2))
            about_me_window.geometry(f"500x125+{x}+{y}")
            me_label = customtkinter.CTkLabel(
                about_me_window,
                text="Created by M. Hanief Fatkhan Nashrullah || 13522100\n GitHub : hannoobz",
                font=("Terminal",12))
            me_label.pack(pady=(20,0))
            yt_channel = customtkinter.CTkButton(
                about_me_window,
                text="My Youtube channel",
                font=("Terminal",12),
                command= yt_link
            )
            yt_channel.pack(pady=(10,0))

            about_me_window.maxsize(500,125)
            about_me_window.minsize(500,125)
            about_me_window.grab_set()
            about_me_window.mainloop()


        super().__init__()

        # App WxH
        appWidth = 800
        appHeight = 600
        screenWidth = self.winfo_screenwidth()
        screenHeight = self.winfo_screenheight()
        x = int((screenWidth/2)-(appWidth/2))
        y = int((screenHeight/2)-(appHeight/2))
        self.title('Brute Force Breach Protocol Solver')
        self.geometry(f'{appWidth}x{appHeight}+{x}+{y}')
        self.minsize(appWidth,appHeight)
        self.maxsize(appWidth,appHeight)

        # Frame
        self.modeOption = customtkinter.CTkFrame(self,fg_color="black",width=300,corner_radius=0)
        self.modeOption.grid(row=0,column=0,sticky="nsew")

        self.mainProgram = customtkinter.CTkFrame(self,fg_color="#141414",width=500,corner_radius=0)
        self.mainProgram.grid(row=0,column=1,sticky="nsew")

        # Set the weight of the columns and rows
        self.columnconfigure(0,weight=5)
        self.columnconfigure(1,weight=50)
        self.rowconfigure(0,weight=1)

        # Mode option
        self.button_randomizedValue = customtkinter.CTkButton(self.modeOption,text="Randomized",font=
("Terminal",14),command=randomizer)
        self.button_randomizedValue.pack(pady=20)

        self.button_customValue = customtkinter.CTkButton(self.modeOption,text="Custom Value",font=
("Terminal",14),command=customVal)
        self.button_customValue.pack()

        self.button_aboutMe = customtkinter.CTkButton(self.modeOption,text="About",font=
("Terminal",14),command=about_me)
        self.button_aboutMe.pack(side="bottom", pady=(0,40))

app = App()
app.mainloop()
```
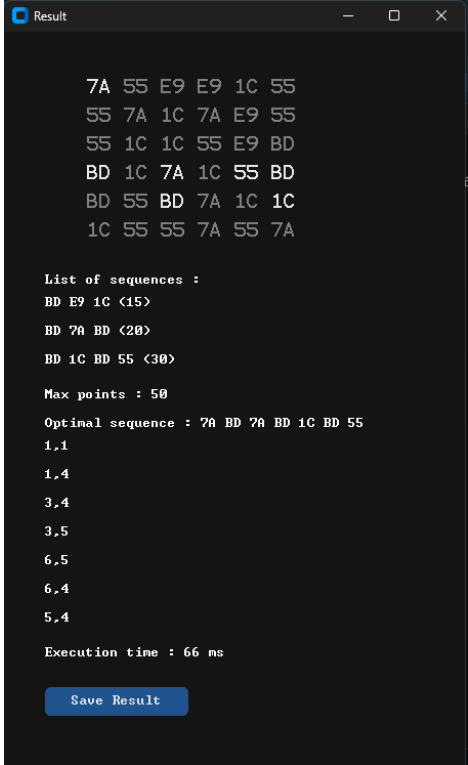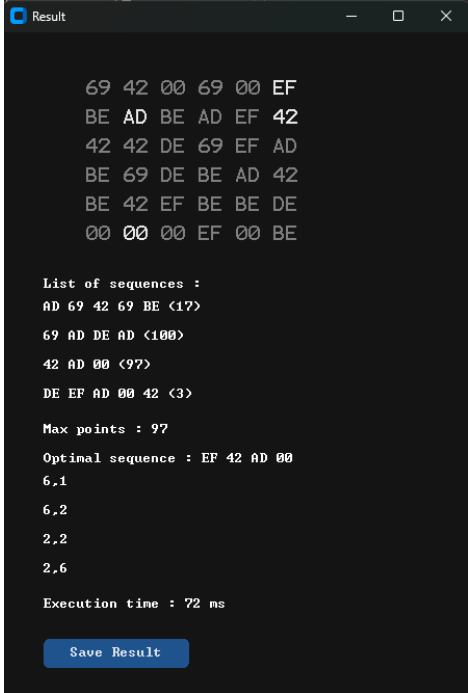
BAB III: Pengujian dan Hasil Pengujian

Input File	
Input	Output
tc1.txt 7 6 6 7A 55 E9 E9 1C 55 55 7A 1C 7A E9 55 55 1C 1C 55 E9 BD BD 1C 7A 1C 55 BD BD 55 BD 7A 1C 1C 1C 55 55 7A 55 7A 3 BD E9 1C 15 BD 7A BD 20 BD 1C BD 55 30	
tc2.txt 7 6 6 69 42 00 69 00 EF BE AD BE AD EF 42 42 42 DE 69 EF AD BE 69 DE BE AD 42 BE 42 EF BE BE DE 00 00 00 EF 00 BE 4 AD 69 42 69 BE 17 69 AD DE AD 100 42 AD 00 97 DE EF AD 00 42 3	

tc3.txt	

tc3.txt

```
8
7 7
01 F0 B8 F0 DE DE F0
F0 B8 DE E5 DE F0 AD
E5 DE AD DE DE 01 AD
AD F0 B8 AD E5 E5 01
01 E5 01 F0 01 F0 F0
E5 F0 F0 AD B8 B8 DE
B8 B8 E5 DE E5 DE 01
7
F0 01 B8 F0
38
E5 F0 AD E5 E5
61
E5 B8 B8
25
DE DE 01 DE F0
83
DE 01 F0 F0 DE 01 B8
17
F0 AD DE AD E5 01
-12
F0 01 DE B8 E5 DE
-46
```

Result window (tc3):

```
        01 F0 B8 F0 DE DE F0
        F0 B8 DE E5 DE F0 AD
        E5 DE AD DE DE 01 AD
        AD F0 B8 AD E5 E5 01
        01 E5 01 F0 01 F0 F0
        E5 F0 F0 AD B8 B8 DE
        B8 B8 E5 DE E5 DE 01

List of sequences :
F0 01 B8 F0 (38)
E5 F0 AD E5 E5 (61)
E5 B8 B8 (25)
DE DE 01 DE F0 (83)
DE 01 F0 F0 DE 01 B8 (17)
F0 AD DE AD E5 01 (-12)
F0 01 DE B8 E5 DE (-46)

Max points : 121
Optimal sequence : DE DE 01 DE F0 01 B8 F0
5,1
5,3
6,3
6,1
7,1
7,4
3,4
3,6

Execution time : 1669 ms
```

tc4.txt

```
8
7 7
B8 69 EE E3 B8 FF E3
EE EA EA 99 FF 69 FF
CC 69 CC EA CC B8 CC
FF FF CA 42 42 42 CC
EE EA CA CA CA B8 99
FF B8 B8 99 69 99 69
EA CC CA B8 CC 69 CA
5
E3 69 B8 69 FF
-61
B8 69 E3 CC B8 42
87
42 EE 42 CA
-79
E3 E3 CC
-17
EE B8 69 FF
-7
```

Result window (tc4):

```
        B8 69 EE E3 B8 FF E3
        EE EA EA 99 FF 69 FF
        CC 69 CC EA CC B8 CC
        FF FF CA 42 42 42 CC
        EE EA CA CA CA B8 99
        FF B8 B8 99 69 99 69
        EA CC CA B8 CC 69 CA

List of sequences :
E3 69 B8 69 FF (-61)
B8 69 E3 CC B8 42 (87)
42 EE 42 CA (-79)
E3 E3 CC (-17)
EE B8 69 FF (-7)

Max points : 0
Optimal sequence :
Execution time : 1573 ms

Save Result
```
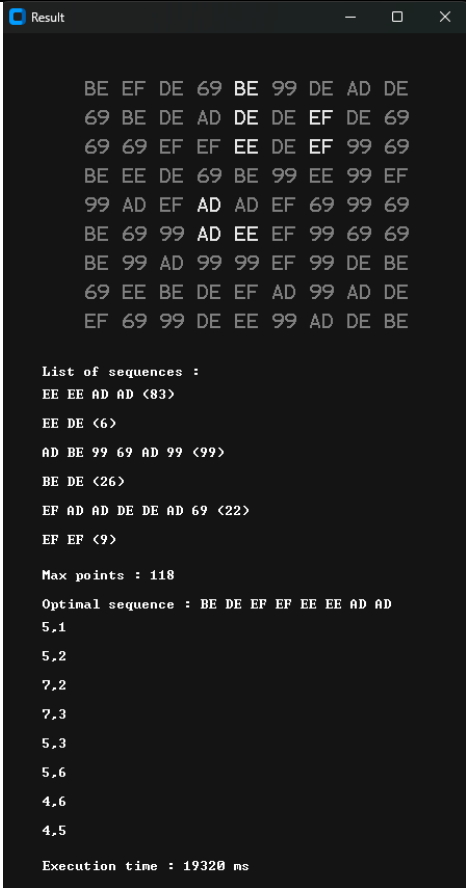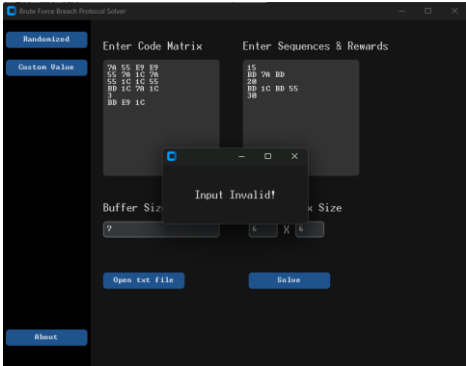
tc5.txt	
8	
9 9	
BE EF DE 69 BE 99 DE AD DE	
69 BE DE AD DE DE EF DE 69	
69 69 EF EF EE DE EF 99 69	
BE EE DE 69 BE 99 EE 99 EF	
99 AD EF AD AD EF 69 99 69	
BE 69 99 AD EE EF 99 69 69	
BE 99 AD 99 99 EF 99 DE BE	
69 EE BE DE EF AD 99 AD DE	
EF 69 99 DE EE 99 AD DE BE	
6	
EE EE AD AD	
83	
EE DE	
6	
AD BE 99 69 AD 99	
99	
BE DE	
26	
EF AD AD DE DE AD 69	
22	
EF EF	
9	

invalidmatrixtc.txt	
7	
6 6	
7A 55 E9 E9	
55 7A 1C 7A	
55 1C 1C 55	
BD 1C 7A 1C	
3	
BD E9 1C	
15	
BD 7A BD	
20	
BD 1C BD 55	
30	

Randomized Input	
Input	Output

Lampiran

1. *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	✓	

2. Pranala *repository*
   https://github.com/hannoobz/Tucil1_13522100