LAPORAN TUGAS KECIL
IF2211 STRATEGI ALGORITMA

Disusun Oleh :
M Athaullah Daffa Kusuma M (13522044)

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2024

Daftar Isi

Bab 1 : Deskripsi Masalah

1.1. Cyberpunk 2077

Cyberpunk 2077 merupakan gim bergenre *action role-playing game* yang dikembangkan dan di-*publish* oleh CD Projekt Red pada tahun 2020 lalu. Latar gim ini berada di sebuah kota fiksi metropolis, California, di dunia Cyberpunk. Pemain akan bermain sebagai V, pedagang yang tiba-tiba mendapatkan "bio-chip" cyber yang berisi informasi-informasi milik *rockstar* legendaris dan teroris Johnny Silverhand. Seiring berjalannya waktu, kesadaran johnny suatu saat akan mengambil alih kesadaran V, mereka berdua nantinya akan bekerja sama agar bisa berpisah dan menyelamatkan nyawa V.

1.2. Cyberpunk 2077 Breach Protocol

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.

Adapula aturan permainan Breach Protocol sebagai berikut:

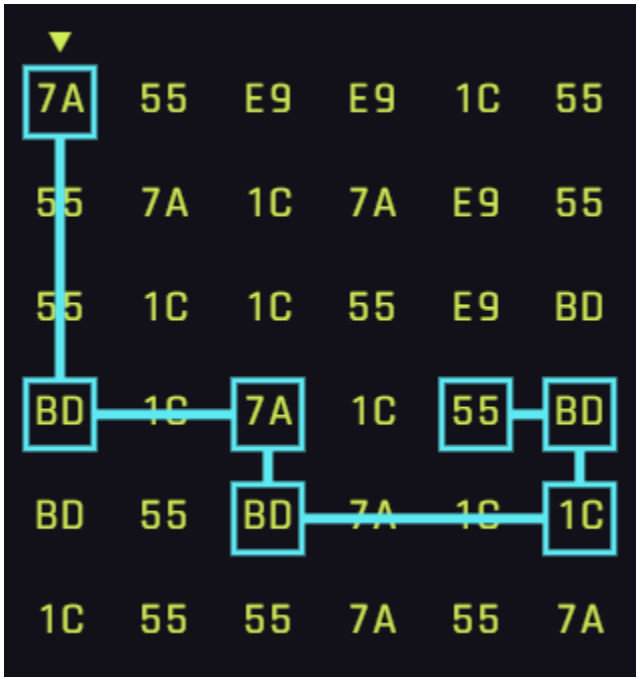
1. Pemain bergerak dengan pola horizontal, vertikal, horizontal, vertikal (bergantian) hingga semua sekuens berhasil dicocokkan atau buffer penuh.
2. Pemain memulai dengan memilih satu token pada posisi baris paling atas dari matriks.
3. Sekuens dicocokkan pada token-token yang berada di buffer.
4. Satu 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



Bab 2 : Landasar Teori

2.1. Algoritma *Brute Force*

Brute force merupakan salah satu algoritma dari sekian banyaknya jenis algoritma yang ada. Brute force merupakan algoritma yang paling simpel dan mudah. Brute force adalah algoritma dimana programmer menjalankan sesuai dengan apa yang diingkan *problem* tersebut. Akurasi dari brute force sangatlah akurat, dimana biasanya algoritma ini akan mengecek semua kemungkinan jawaban yang ada. Untuk contohnya ketika ada problem dimana kita diminta untuk mencari jalan terpendek dari titik A sampai titik B, banyak jalan yang tersedia. Algoritma brute force akan mengecek semua kemungkinan jalan yang dapat dilewati dari titik A untuk sampai ke titik B. Algoritma ini mempunyai satu kelemahan yang paling terlihat, yaitu waktunya yang sangat tidak efisien.

2.2. Implementasi *Brute Force* dalam menyelesaikan Breach Protocol

Brute force di sini akan diterapkan untuk menyelesaikan masalah Breach Protocol. Program nantinya akan memulai mengiterasi row pertama dari matriks, dimana program nantinya akan mencoba semua kemungkinan yang ada. Setelah program memilih salah satu elemen dari row pertama matriks, nantinya program akan mengiterasi elemen lain yang memiliki kolom yang sama seperti elemen sebelumnya. Setelah program memilih salah satu elemen dari kolom tersebut, nantinya program akan memilih elemen lain yang memiliki row yang sama seperti elemen barusan. Program akan mengulagi hal ini sampai batas buffer yang dimiliki. Akan tetapi, program juga akan mengevaluasi setiap 2 elemen dipilih atau lebih, tujuannya yaitu agar program mendapatkan sequence yang terpendek dan memberi reward paling banyak. Program akan mencoba seluruh kemungkinan yang ada, karena itulah algoritma yang dipakai merupakan algoritma brute force.

Bab 3 : Implementasi

3.1. Solver.py

Source code berikut merupakan source code yang memberikan sequence terpendek dan termaksimal dengan input berupa matriks dan batasan2 tertentu lainnya

```
import time

sizebuffer = 0
colrow = []
matrixnya = []
size = 0
sequences = []

def txtParse(txtString):
    arrRes = []
    strtemp = ""
    for i in txtString:
        if i == "\n":
            arrRes.append(strtemp)
            strtemp = ""
        else:
            strtemp += i
    if strtemp != "":
        arrRes.append(strtemp)
    return arrRes

def proses(arrRes):
    global size; global sizebuffer; global colrow; global
matrixnya; global sequences
    sizebuffer = int(arrRes[0])
    colrow = [int(i) for i in arrRes[1].split()]
    matrixnya = [[0 for i in range(colrow[0])] for j in
range(colrow[1])]
    for i in range(colrow[1]):
        matrixnya[i] = [str(a) for a in arrRes[2+i].split()]
    size = int(arrRes[2+colrow[1]])
    sequences = [["a", "b"] for j in range(size)]
    for i in range(size):
        sequences[i][0] = arrRes[3+colrow[1]+(2*i)]
        sequences[i][1] = int(arrRes[4+colrow[1]+(2*i)])

maxval = -1
maxarr = ""
maxcoor = []
```

```
def evaluate(arr, coor):
    global maxval; global maxarr; global maxcoor
    strtemp = ""
    for i in range(len(arr)):
        strtemp += arr[i]
        if i != len(arr)-1:
            strtemp += " "
    valtemp = 0
    for i in range(len(sequences)):
        if sequences[i][0] in strtemp:
            valtemp += sequences[i][1]
    if valtemp > maxval or (valtemp == maxval and len(strtemp) <
len(maxarr)):
        maxarr = strtemp
        maxval = valtemp
        maxcoor = coor

def uniquecoor(coor):
    for i in range(len(coor)):
        for j in range(i+1, len(coor)):
            if coor[i] == coor[j]:
                return False
    return True

def iterate(arrtemp, coortemp, rowbool):
    # jika buffer kosong
    if len(arrtemp) == 0:
        for i in range(colrow[0]):
            arrtemp2 = arrtemp.copy()
            coortemp2 = coortemp.copy()
            arrtemp2.append(matrixnya[0][i])
            coortemp2.append((0, i))
            iterate(arrtemp2, coortemp2, True)
    # jika buffer belum penuh
    elif len(arrtemp) < sizebuffer:
        coorrecent = coortemp[len(coortemp)-1]
        if rowbool:
            for i in range(colrow[1]):
                if i != coorrecent[0]:
                    arrtemp2 = arrtemp.copy()
                    coortemp2 = coortemp.copy()
                    arrtemp2.append(matrixnya[i][coorrecent[1]])
                    coortemp2.append((i, coorrecent[1]))
                    if uniquecoor(coortemp2):
                        evaluate(arrtemp2, coortemp2)
                    iterate(arrtemp2, coortemp2, False)
        else:
            for i in range(colrow[0]):
                if i != coorrecent[1]:
                    arrtemp2 = arrtemp.copy()
                    coortemp2 = coortemp.copy()
```

```
                    arrtemp2.append(matrixnya[coorrecent[0]][i])
                    coortemp2.append((coorrecent[0], i))
                    if uniquecoor(coortemp2):
                        evaluate(arrtemp2, coortemp2)
                    iterate(arrtemp2, coortemp2, True)

def executeSol(txtString):
    arg = txtParse(txtString)
    proses(arg)
    start_time = time.perf_counter()
    iterate([], [], True)
    print()
    print(maxval)
    print(maxarr)
    for i in maxcoor:
        print(f"{i[1]+1}, {i[0]+1}")
    end_time = time.perf_counter()
    print(f"\nElapsed time : {round((end_time-start_time)*1000,
2)}ms")
    verif = input("Apakah ingin menyimpan solusi? (y/n): ")
    if verif == "y":
        patharg = input("Masukkan nama file untuk disimpan: ")
        f = open("../test/"+patharg, "w")
        f.write(f"{maxval}\n{maxarr}")
        for i in maxcoor:
            f.write(f"\n{i[1]+1}, {i[0]+1}")
        f.close()
        print("Solusi berhasil disimpan!")
```

Program pertama akan mengparse argument txt yang diberikan, lalu dari argument berikut akan diproses dengan fungsi proses(arg) untuk mengekstrak isi informasi seperti matriks, size buffer, sequence, dan lainnya ke variabel yang ada. Lalu program akan mencatat waktu sekarang sebagai tanda waktu dimulainya iterasi. Setelah itu program memulai fungsi iterate() untuk mengiterasi semua kemungkinan yang ada, setelah selesai, program akan mengeluarkan hasil-hasil yang diinginkan. Lalu program memberikan opsi apakah pengguna ingin menyimpan hasil ke sebuah file txt atau tidak

3.2. Generator.py

Source code berikut merupakan source code yang memberikan matriks breach protocol dan sequence nya secara acak/*random* dengan batasan tertentu yang ditentukan oleh user

```
import random as rd


def executeGen():
    print("masukkan input di bawah ini sesuai dengan format yang
ditentukan\n")
```

```
    jmltokenunik = int(input())
    token = [str(a) for a in input().split()]
    ukuranbuffer = int(input())
    colrow = [int(a) for a in input().split()]
    jmlseq = int(input())
    maxseq = int(input())

    thematrix = [["" for i in range(colrow[0])] for j in
range(colrow[1])]
    for i in range(colrow[1]):
        for j in range(colrow[0]):
            thematrix[i][j] = rd.choice(token)
    theseq = [["a", "b"] for i in range(jmlseq)]
    for i in range(jmlseq):
        panjangseq = rd.randint(2, maxseq)
        seqtemp = ""
        for j in range(panjangseq):
            seqtemp += rd.choice(token)
            if j != panjangseq-1:
                seqtemp += " "
        theseq[i][0] = seqtemp
        theseq[i][1] = rd.randint(5, 50)

    print("\n\nDidapatkan Matrix dengan bentuk berikut: ")
    print("Buffer Size :", ukuranbuffer)
    print(f"Matrix Width and Height : {colrow[0]} {colrow[1]}")
    for i in range(colrow[1]):
        for j in range(colrow[0]):
            print(f"{thematrix[i][j]} ", end="")
        print()
    print("Jumlah sequence :", jmlseq)
    for i in range(jmlseq):
        print(theseq[i][0])
        print(theseq[i][1])
    verif = input("Apakah ingin menyimpan solusi? (y/n): ")
    if verif == "y":
        patharg = input("Masukkan nama file untuk disimpan: ")
        f = open("../test/"+patharg, "w")
        f.write(f"{ukuranbuffer}\n{colrow[0]} {colrow[1]}\n")
        for i in range(colrow[1]):
            for j in range(colrow[0]):
                f.write(f"{thematrix[i][j]} ")
            f.write("\n")
        f.write(f"{jmlseq}")
        for i in range(jmlseq):
            f.write(f"\n{theseq[i][0]}\n{theseq[i][1]}")
        f.close()
        print("Solusi berhasil disimpan!")
```

Program nantinya akan meminta input dari CLI sesuai dengan format yang ada, lalu program akan mengeluarkan matriks dan sequence secara random dengan batasan yang telah diberikan oleh pengguna.

3.3. Main.py

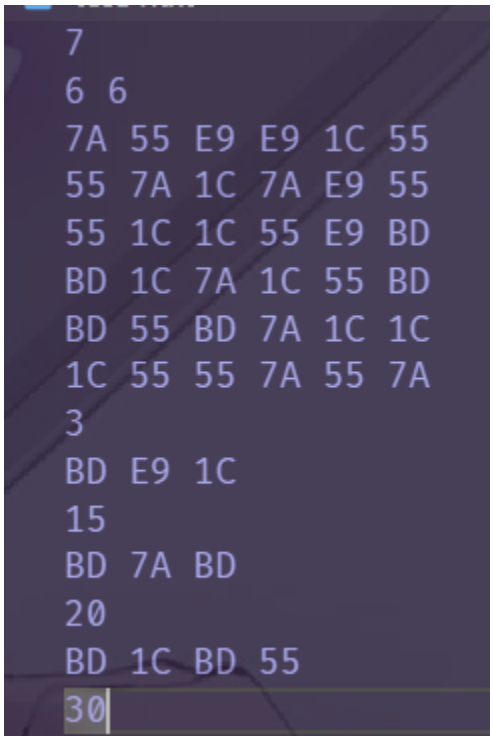
Source code berikut merupakan source code yang memberikan tampilan menu kepada pengguna.

```
from solver import *
from generator import *
import os


print("Selamat datang di permainan Breach Protocol\n")
print("silahkan pilih aksi yang hendak dilakukan:\n1.Breach
Protocol Solver\n2.Breach Protocol Generator")
val = input("silahkan pilih menu: ")
while(val != "1" and val != "2"):
    print("input anda tidak valid!")
    val = input("silahkan pilih menu: ")
val = int(val)

if val == 1:
    txtarg = input("Masukkan file txt yang hendak dimasukkan
(gunakan .txt): ")
    if not os.path.isfile("../test/"+txtarg):
        print("Tidak ditemukan file yang dimaksud!")
        exit()
    else:
        executeSol(open("../test/"+txtarg, "r").read())
else:
    executeGen()
```

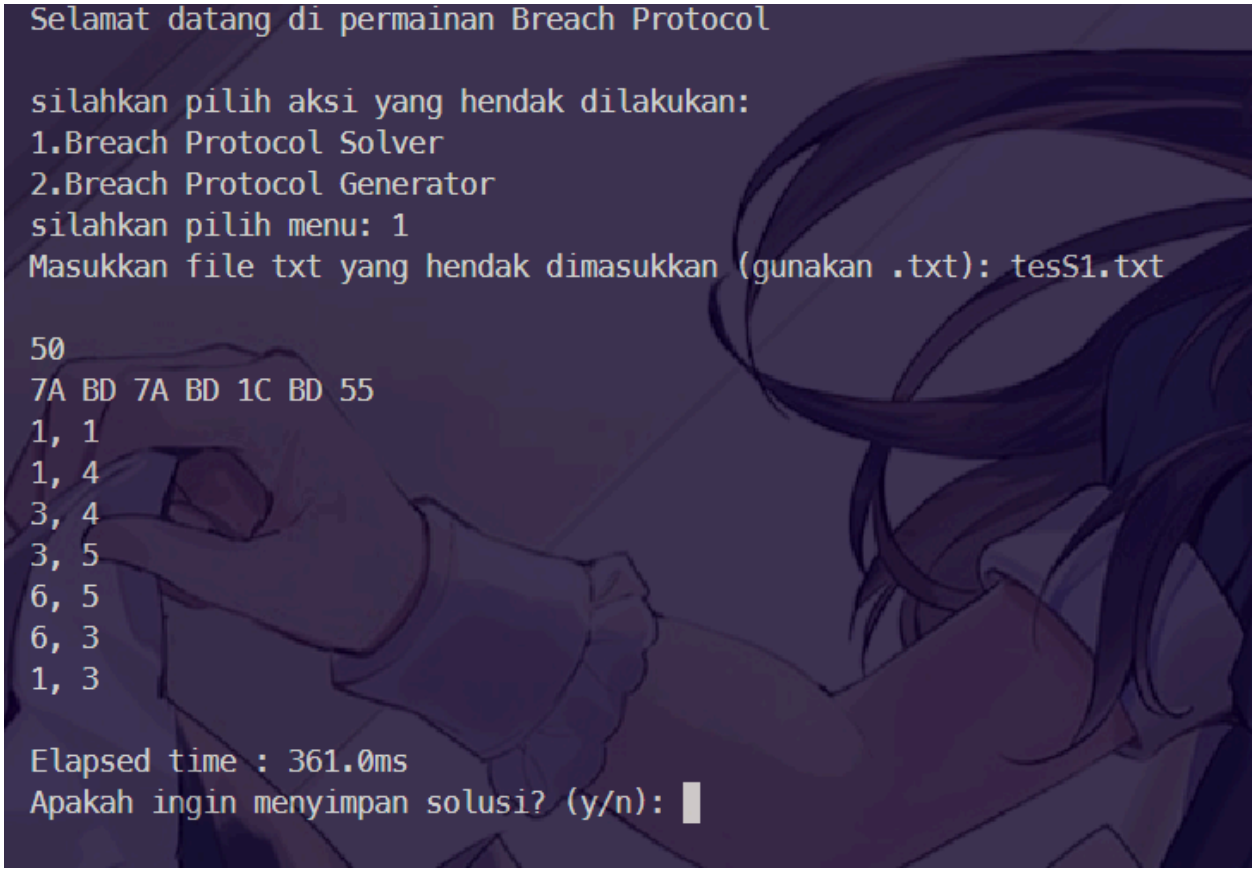
Bab 4 : Uji Coba

4.1. Test Case 1 – Solver

Dengan input berupa file yang bernama tesS1.txt dengan isi



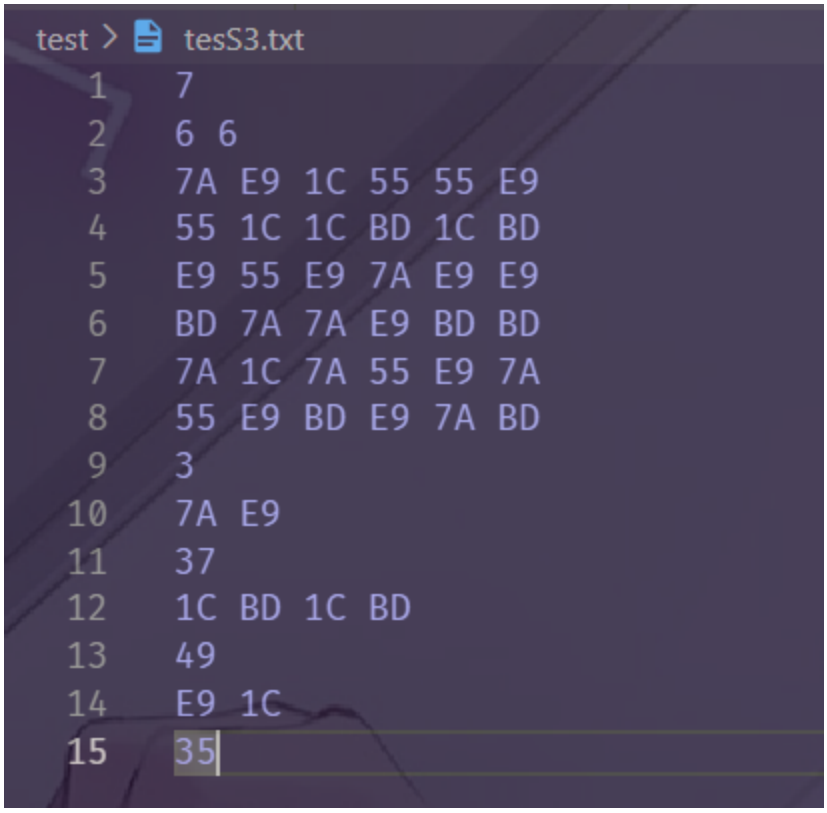
Didapat solusi yaitu



Dan ketika memilih “y” saat ingin menyimpan solusi ke dalam file “ResS1.txt”

4.2. Test Case 2 – Solver

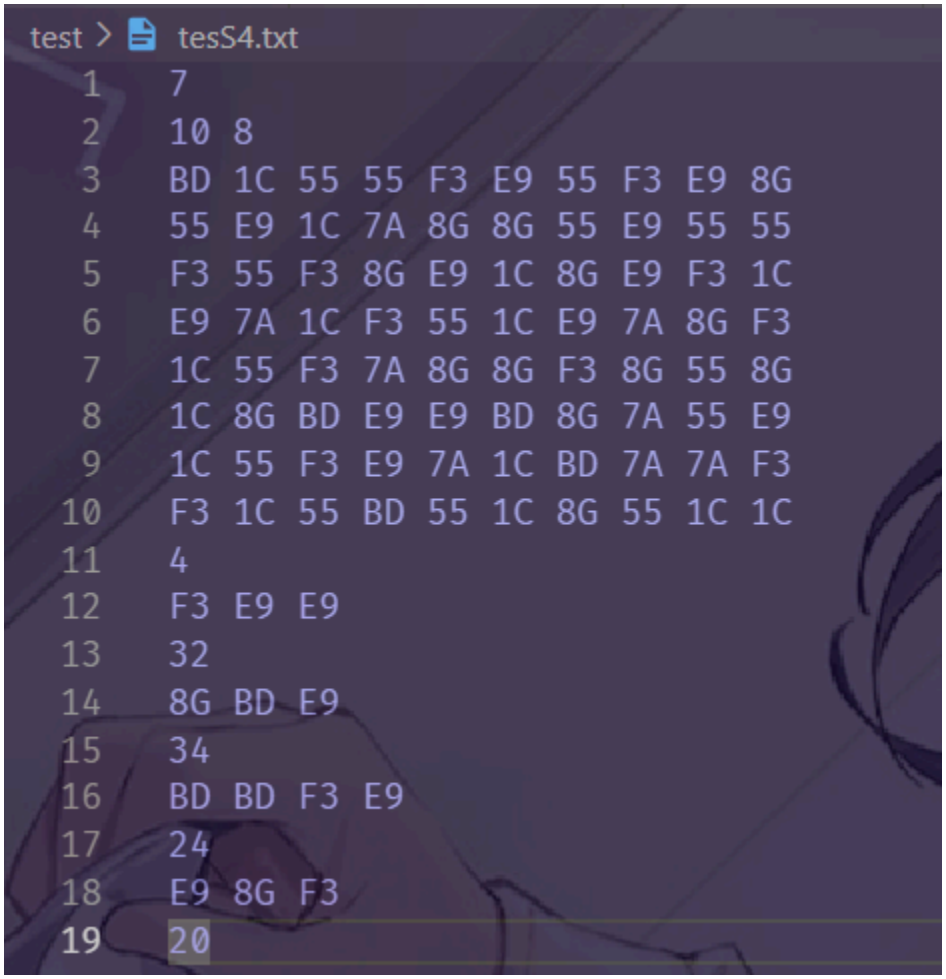
Dengan input file txt bernama “tesS3.txt” yang berisi



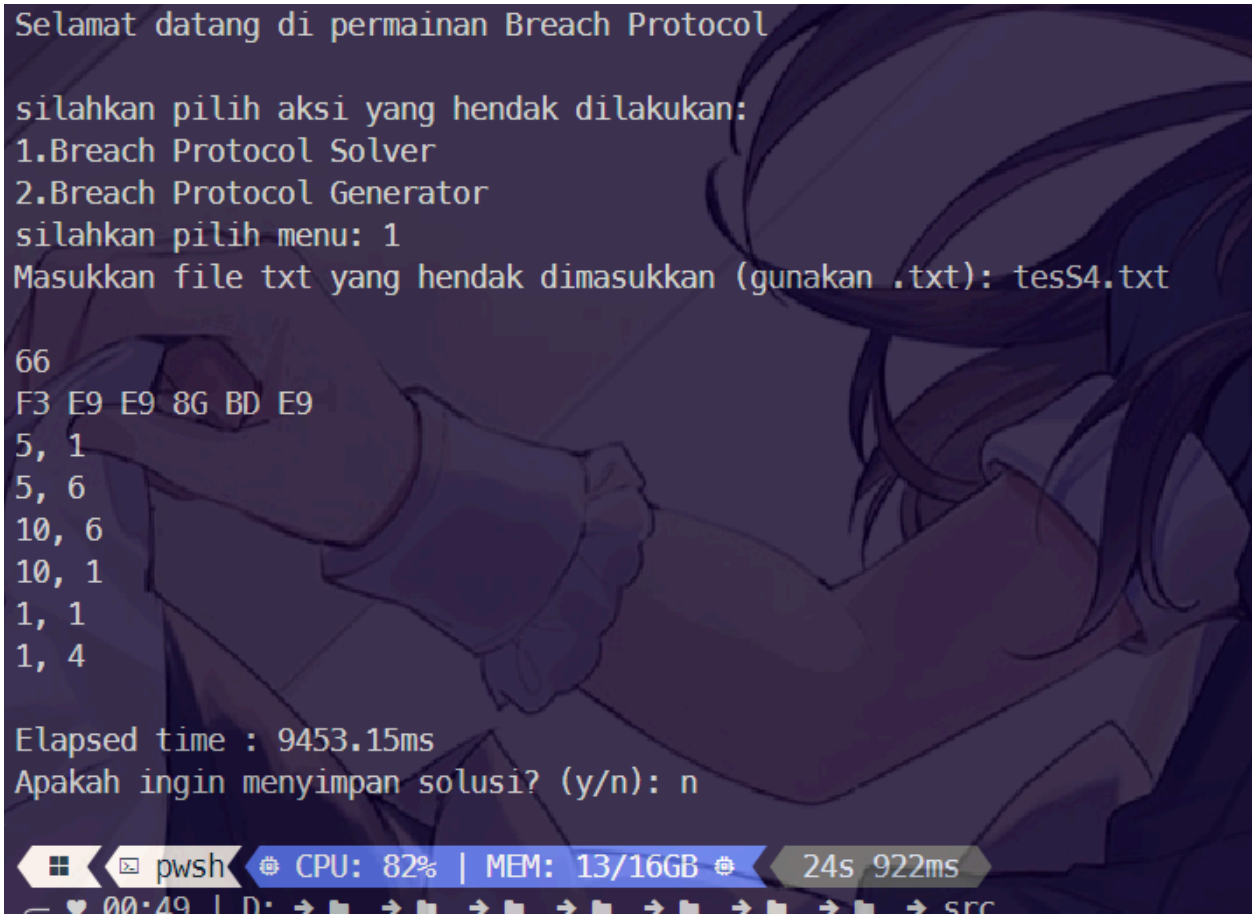
Didapat solusi sebagai berikut, dan pengguna memilih opsi “n” ketika ditanya hendak menyimpan hasil

4.3. Test Case 3 – Solver

Dengan input berupa file bernama “tesS4.txt” yang berisi

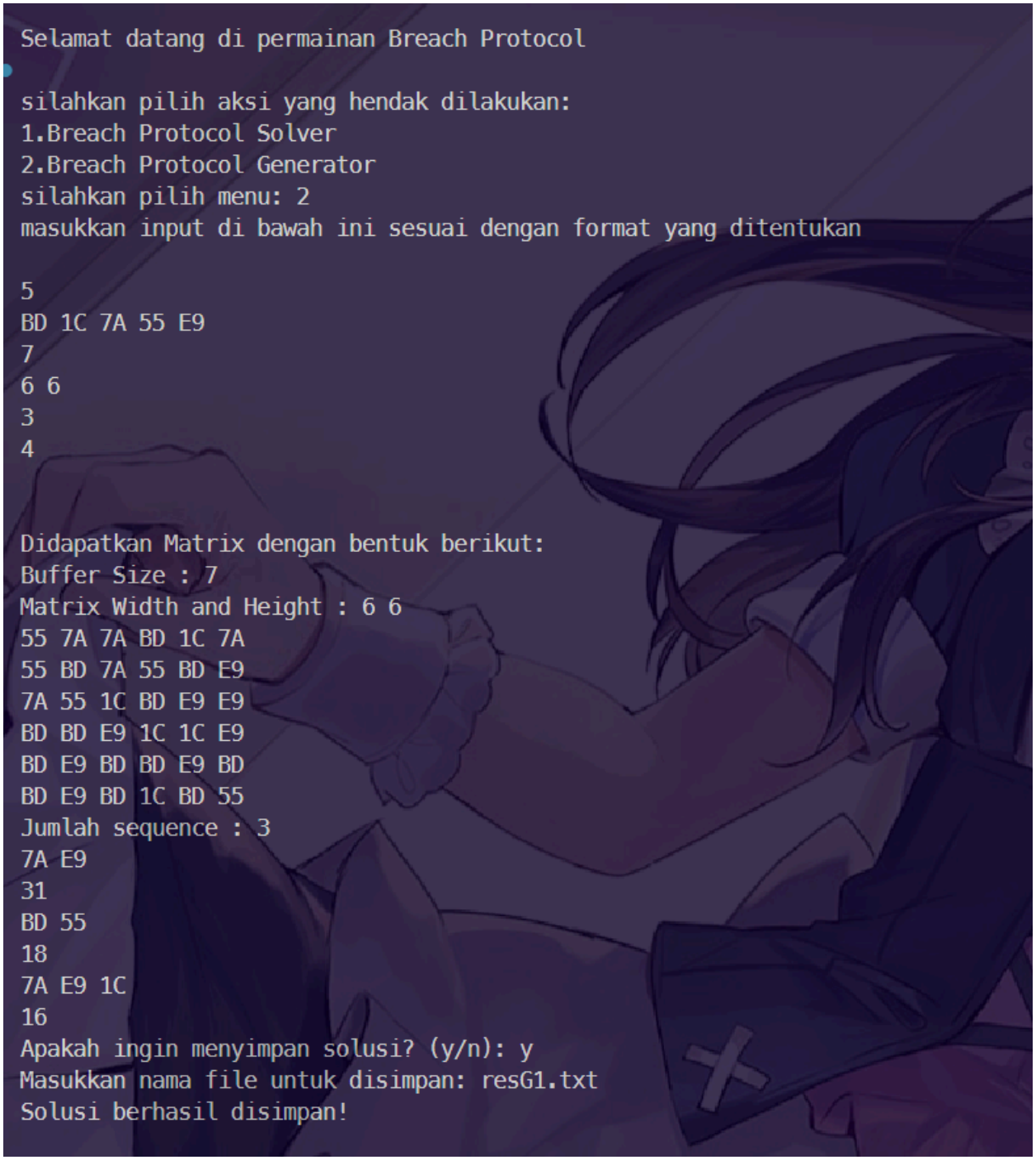


Didapat solusi sebagai berikut, dan pengguna memilih opsi “n” ketika ditanya hendak menyimpan hasil

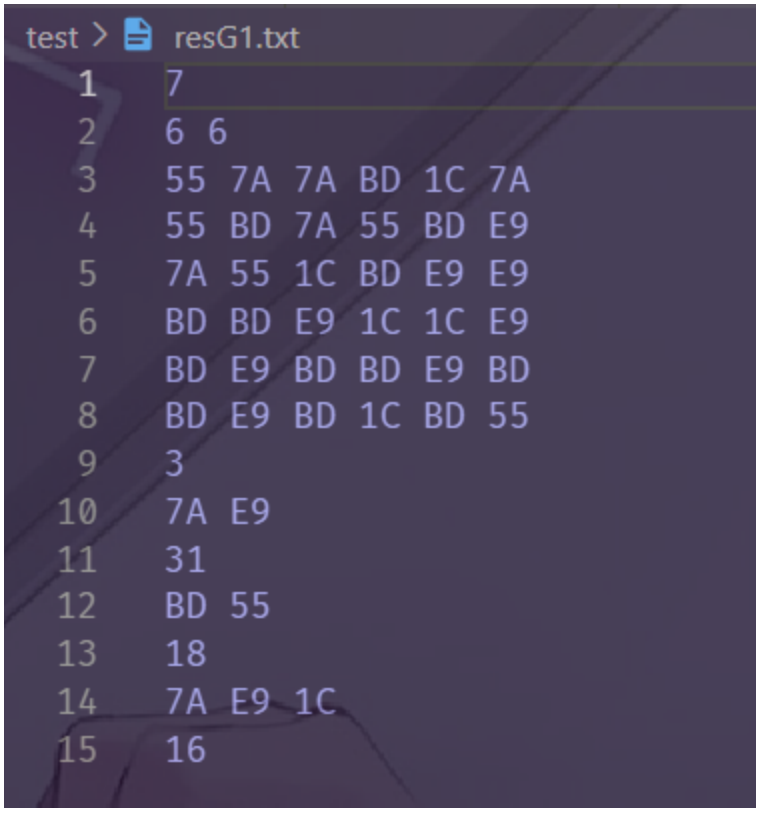


4.4. Test Case 4 – Generator

Berikut test case dengan input tertera, dan pengguna memilih “y” ketika ditanya hendak menyimpan hasil

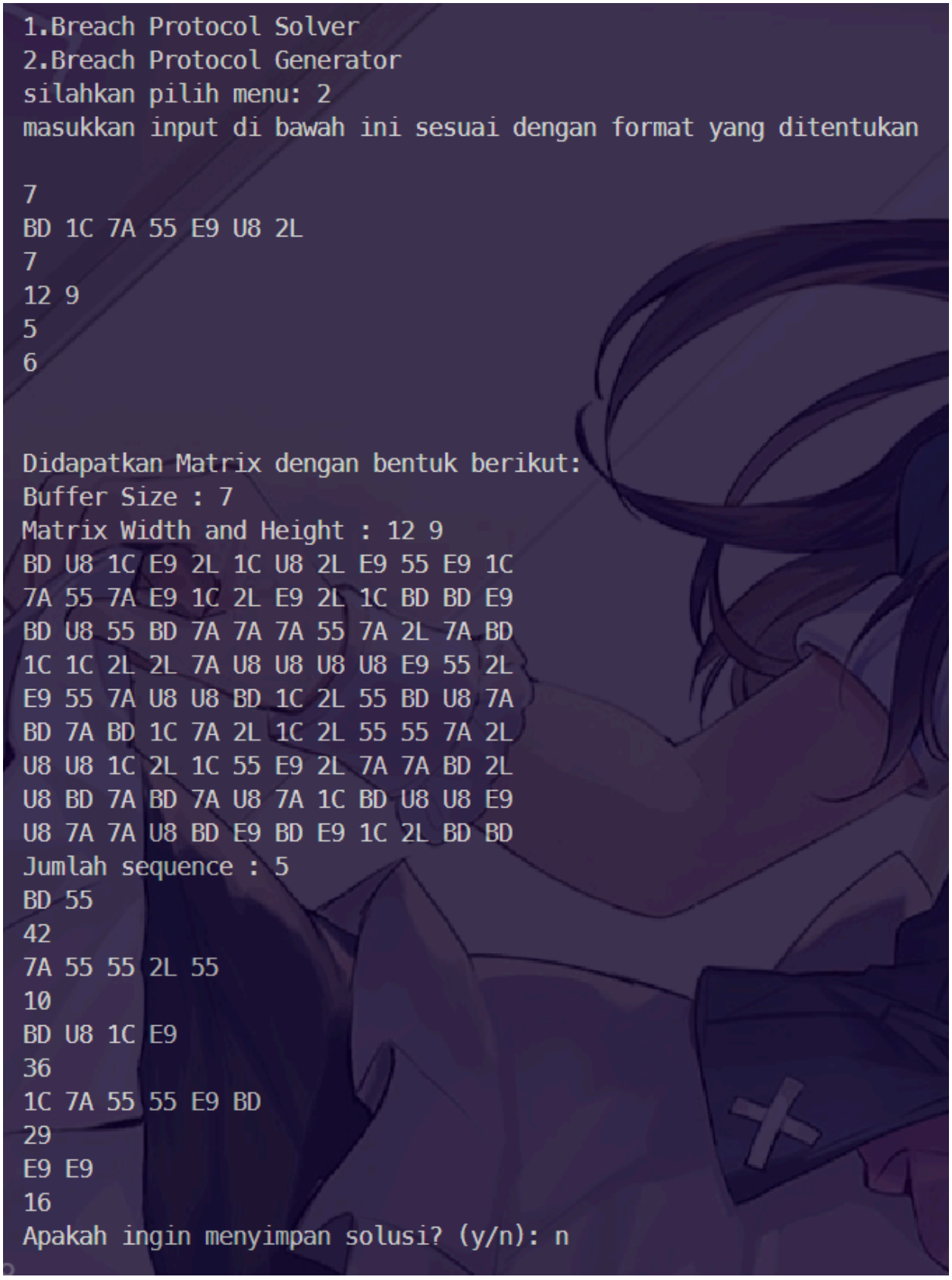


Dan berikut isi dari file resG1.txt



4.5. Test Case 5 – Generator

Berikut test case dengan input tertera, dan pengguna memilih “n” ketika ditanya hendak menyimpan hasil



```
1.Breach Protocol Solver
2.Breach Protocol Generator
silahkan pilih menu: 2
masukkan input di bawah ini sesuai dengan format yang ditentukan

7
BD 1C 7A 55 E9 U8 2L
7
12 9
5
6


Didapatkan Matrix dengan bentuk berikut:
Buffer Size : 7
Matrix Width and Height : 12 9
BD U8 1C E9 2L 1C U8 2L E9 55 E9 1C
7A 55 7A E9 1C 2L E9 2L 1C BD BD E9
BD U8 55 BD 7A 7A 7A 55 7A 2L 7A BD
1C 1C 2L 2L 7A U8 U8 U8 U8 E9 55 2L
E9 55 7A U8 U8 BD 1C 2L 55 BD U8 7A
BD 7A BD 1C 7A 2L 1C 2L 55 55 7A 2L
U8 U8 1C 2L 1C 55 E9 2L 7A 7A BD 2L
U8 BD 7A BD 7A U8 7A 1C BD U8 U8 E9
U8 7A 7A U8 BD E9 BD E9 1C 2L BD BD
Jumlah sequence : 5
BD 55
42
7A 55 55 2L 55
10
BD U8 1C E9
36
1C 7A 55 55 E9 BD
29
E9 E9
16
Apakah ingin menyimpan solusi? (y/n): n
```

4.6. Test Case 6 – Generator

Berikut test case dengan input tertera, dan pengguna memilih “n” ketika ditanya hendak menyimpan hasil

Lampiran

Link Repository

https://github.com/NameLessAth/Tucil1_13522044

Sheets Poin

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		✓