

LAPORAN TUGAS KECIL 1

IF2211 Strategi Algoritma

Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force



Nama: Dhidit Abdi Aziz

NIM: 13522040

Kelas: 02

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

1 Deskripsi Permasalahan



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:

- Token – terdiri dari dua karakter alfanumerik seperti E9, BD, dan 55.
- Matriks – terdiri atas token-token yang akan dipilih untuk menyusun urutan kode.
- Sekuens – sebuah rangkaian token (dua atau lebih) yang harus dicocokkan.
- Buffer – jumlah maksimal token yang dapat disusun secara sekuensial.

Aturan permainan Breach Protocol antara lain:

- Pemain bergerak dengan pola horizontal, vertikal, horizontal, vertikal (bergantian) hingga semua sekuens berhasil dicocokkan atau buffer penuh.
- Pemain memulai dengan memilih satu token pada posisi baris paling atas dari matriks.
- Sekuens dicocokkan pada token-token yang berada di buffer.
- Satu token pada buffer dapat digunakan pada lebih dari satu sekuens.
- Setiap sekuens memiliki bobot hadiah atau reward yang variatif.
- 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:

- BD E9 1C dengan hadiah berbobot 15.
- BD 7A BD dengan hadiah berbobot 20
- 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



Melalui Tugas Kecil 1 ini, saya mengimplementasikan program untuk menemukan solusi optimum pada setiap kombinasi matriks, sekuens, dan ukuran buffer dengan algoritma Brute Force

2 Algoritma Brute Force

Brute force merupakan pendekatan yang *straightforward* untuk memecahkan suatu persoalan. Artinya, algoritma tersebut akan mengevaluasi seluruh kemungkinan solusi yang dapat terjadi. Dengan demikian, solusi pasti akan didapatkan walau memerlukan waktu yang cenderung lebih lambat. Apabila algoritma ini diterapkan pada permasalahan “Cyberpunk 2077 Breach Protokol”, kita perlu mencacah setiap kemungkinan kombinasi token yang mungkin untuk mencari bobot nilai tertinggi yang dapat diperoleh dari kombinasi token pada suatu matriks. Untuk merealisasikannya, saya menggunakan fungsi rekursif yang akan mencoba seluruh permutasi kemungkinan kombinasi token. Saya juga menerapkan prinsip backtracking agar program dapat lebih efisien dalam mencari seluruh kemungkinan

Secara sederhana, apabila diberikan batasan maksimal buffer sepanjang X token pada matriks berukuran $[M][N]$, saya akan mencari seluruh permutasi X token yang mungkin terjadi pada matriks tersebut. Proses pencarian akan dimulai dari kolom pertama.

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

Looping akan dilakukan pada seluruh elemen baris pertama dari pojok kiri hingga pojok kanan untuk mencari token pertama dalam urutan buffer. Setelah didapatkan token pertama, maka looping akan dilakukan lagi pada kolom yang sama dengan letak token pertama untuk mencari token kedua. Selanjutnya, setelah didapatkan token pertama, looping akan diterapkan pada seluruh elemen pada baris yang sama dengan token kedua. Proses ini akan dilakukan secara berulang hingga diperoleh permutasi X token. Setiap mengambil token pada matriks, nilai token pada matriks tersebut akan diganti sebagai mark sehingga program tidak akan mengambil dua permutasi yang sama.

Selanjutnya, X token tersebut akan diperiksa untuk mengetahui seberapa banyak point yang dimilikinya. Perlu dicatat bahwa di awal sudah dilakukan inisiasi untuk variabel penyimpanan poin sementara. Apabila, nilai poin yang dimiliki X token lebih besar daripada nilai point maksimal sementara, maka nilai point maksimal sementara akan diganti. X token juga akan sementara dianggap sebagai permutasi token paling optimum.

Akan terdapat kasus dimana di tengah jalan sudah didapatkan urutan token yang mampu memberikan nilai poin tertinggi. Nilai poin tertinggi disini maksudnya total dari seluruh poin sekuens. Pada kasus tersebut, program dapat langsung dihentikan tanpa perlu melanjutkan proses pencacahan permutasi.

Usai perhitungan point, akan dilakukan backtracking dengan cara token ke-X pada permutasi dikembalikan nilainya kepada matriks. Hal ini perlu dilakukan agar pada saat token ke-(X-1) berganti, token ke-X dapat peluang untuk terpilih kembali. Hal yang sama juga akan terjadi apabila seluruh kemungkinan untuk token ke-(X-1) sudah terpilih. Maka token ke-(X-1) tersebut akan dikembalikan nilainya kepada matriks, sehingga proses pergantian nilai token ke-(X-2) dapat terjadi.

Ketika seluruh permutasi sudah dihitung poinnya, akan didapatkan point tertinggi dan urutan token paling optimal. Kedua variabel inilah yang akan menjadi solusi pada permasalahan Cyberpunk 2077 Breach Protocol.

3 Source Code

Kode program dibagi menjadi dua buah file terpisah:

3.1 Main.py

Merupakan file terjadinya proses inisiasi program, input pengguna, hingga pencarian solusi

```
# Import library dan fungsi dari file util
import random
from util import *
import time

# Inisiasi Variabel Global
final_seq = []
final_points = 0
final_l_loc = []
done = False

# Mengatur format koordinat
def print_final_l_loc():
    global final_l_loc
    for i in range (len(final_l_loc)):
        final_l_loc[i][0] += 1
        final_l_loc[i][1] += 1

        temp = final_l_loc[i][0]
        final_l_loc[i][0] = final_l_loc[i][1]
        final_l_loc[i][1] = temp

    print("%d," % final_l_loc[i][0], end=" ")
    print("%d" % final_l_loc[i][1], end="\n")

# Menghitung point
def hitung(l_seq, string_seq, l_points):
    sum = 0
    string_l_seq = arrtostring(l_seq)
    for i in range(len(string_seq)):
        if(string_seq[i] in string_l_seq):
            sum += l_points[i]
```

```

    return sum

# Fungsi Bruteforce
def solver(matrix, h, w, temp_seq, max_buffer, temp_l_loc, string_seq,
l_points, curr_loc, nowvertical):
    global final_seq # List penampung sekuens final
    global final_points # Point maksimal
    global final_l_loc # Koordinat dari tiap sekuens final
    global done # Ketika setiap sekuens sudah ditemukan dalam satu
rangkaian buffer sehingga diraih point maksimum, program berhenti

    max_points = 0
    for i in range(len(l_points)):
        max_points += l_points[i]

    if not done:
        if(len(temp_seq) == max_buffer): #Basis rekursi. Jika buffer penuh
maka stop, hitung point, atau backtracking
        # Ketika panjang temp_seq == max buffer
        temp_point = hitung(temp_seq, string_seq, l_points)
        if(temp_point>final_points):
            final_points = temp_point
            final_seq.clear()
            final_seq.extend(temp_seq)
            final_l_loc.clear()
            final_l_loc.extend(temp_l_loc)

            if (final_points == max_points):
                done = True

    else:
        if(nowvertical == False): # Iterasi kiri-kanan
            for i in range (w):
                new_loc = [0,0]
                new_loc[0] = curr_loc[0]
                new_loc[1] = curr_loc[1]
                new_loc[1] = i
                if(matrix[new_loc[0]][new_loc[1]] == '??'):
                    continue

                temp_l_loc.append(new_loc)
                temp_seq.append(matrix[new_loc[0]][new_loc[1]])

```

```

        matrix[new_loc[0]][new_loc[1]] = '??'

        solver(matrix, h, w, temp_seq, max_buffer, temp_l_loc,
string_seq, l_points, new_loc, True)

matrix[temp_l_loc[len(temp_l_loc)-1][0]][temp_l_loc[len(temp_l_loc)-1][1]]
= temp_seq[len(temp_seq)-1]
        temp_seq.pop(len(temp_seq) - 1)
        temp_l_loc.pop(len(temp_l_loc) - 1)

    else:
        for i in range(h):
            new_loc = [0,0]
            new_loc[0] = curr_loc[0]
            new_loc[1] = curr_loc[1]
            new_loc[0] = i
            if(matrix[new_loc[0]][new_loc[1]] == '??'):
                continue
            temp_l_loc.append(new_loc)
            temp_seq.append(matrix[new_loc[0]][new_loc[1]])
            matrix[new_loc[0]][new_loc[1]] = '??'
            solver(matrix, h, w, temp_seq, max_buffer, temp_l_loc,
string_seq, l_points, new_loc, False)

matrix[temp_l_loc[len(temp_l_loc)-1][0]][temp_l_loc[len(temp_l_loc)-1][1]]
= temp_seq[len(temp_seq)-1]
        temp_seq.pop(len(temp_seq) - 1)
        temp_l_loc.pop(len(temp_l_loc) - 1)

# Membaca file dari path
def bacafile(pathfile):
    file = open(pathfile, "r")

    #Bilangan pertama adalah buffer size
    buffer_size = int(next(file).split()[0])

    #Bilangan kedua dan ketiga adalah kolom dan baris
    kolom, baris = [int(x) for x in next(file).split()]

    #Matrix

```



```

array = []
for i in range(baris):
    line = file.readline()
    temp = []
    for x in line.split():
        temp.append(str(x))
    array.append(temp)

#Number of sequences
n_seq = int(next(file).split()[0])

#Seq and points
l_points = []
seq = []
for i in range(n_seq):
    line = file.readline()
    temp = []
    for x in line.split():
        temp.append(str(x))
    seq.append(temp)
    l_points.append(int(next(file).split()[0]))

#Menjadikan string
stringseq = []
for i in range(n_seq):
    stringseq.append(arrtoststring(seq[i]))

return buffer_size, kolom, baris, array, stringseq, l_points

# Program Utama
print('Selamat datang di game Saiberpang 2045 Breach Protokol!')
opsi = int(input('Kamu mau upload file txt atau generate otomatis nih?\nKetik 1 untuk upload file\nKetik 2 untuk otomatis\nPilihanmu: '))
while ((opsi != 1) and (opsi!=2)):
    opsi = int(input('Ketik 1 untuk upload file\nKetik 2 untuk otomatis\nPilihanmu: '))

if (opsi == 1):
    print('\nYuk masukkan path filemu!')

```

```

print("Contoh path: C:/Users/ACER/Downloads/input.txt")
path = input("Path file mu: ")

buffer_size, w, h, m, stringseq, l_points = bacafile(path)

start = time.process_time()
solver(m, h, w, [], buffer_size, [], stringseq, l_points, [0,0],
False)
duration = time.process_time() - start

if(final_points == 0):
    print("Maaf ya ternyata gaada solusinya hehe")
    print(final_points)
    print("\n%s s\n" %str(duration))

else:
    print(final_points)
    print(arr_to_string(final_seq))
    print_final_l_loc()
    print("\n%s s\n" %str(duration))

tulistxt(final_points, final_seq, final_l_loc, duration)

elif (opsi == 2):
    n_token = int(input("Masukkan jumlah token unik: "))
    token = input("Masukkan token: ")
    ukuran_buffer = int(input("Masukkan ukuran buffer: "))
    w = int(input("Masukkan jumlah kolom matriks: "))
    h = int(input("Masukkan jumlah baris matriks: "))
    n_seq = int(input("Masukkan jumlah sekuens: "))
    n_max_seq = int(input("Masukkan ukuran maksimal sekuens: "))

    l_token = []
    for x in token.split():
        l_token.append(str(x))
    print(l_token)

    # Membuat matriks
    k = 0

```

```

matrix = []
for i in range(h):
    temp = []
    for j in range(w):
        x = random.randint(0,n_token-1)
        temp.append(l_token[x])
        k = x
    matrix.append(temp)

print("\n[MATRIKS PERMAINAN]")
printmatrix(matrix)

isUnique = False
while (not isUnique):
    seq = []
    l_points = []

    for i in range(n_seq):
        temp = []
        sum = 0
        # Merandom banyaknya token dalam satu sequence
        x = random.randint(2,n_max_seq)
        for j in range(x):
            y = random.randint(0, n_token-1)
            temp.append(l_token[y])
            sum += y
        seq.append(temp)

        #Merandom poin
        z = random.randint(0,1)
        l_points.append(((x-1)**2)*10 + ((sum//x + z)*5))

    #Menjadikan string
    stringseq = []
    for i in range(n_seq):
        stringseq.append(arrtostring(seq[i]))

    #Mengecek apakah sequence sudah unique atau tidak
    isUnique = True

```

```

i = 0
while(isUnique and i<n_seq):
    j = i+1
    while(isUnique and j<n_seq):
        if(stringseq[i] == stringseq[j]):
            #Kalau ada yang sama maka ulang loop
            isUnique = False
        else:
            j += 1
    i += 1

#Mengeprint
print("\n[SEQUENCE PERMAINAN]")
for i in range(n_seq):
    print("Sequence ke-%d: " %(i+1), end='')
    print(arr_to_string(seq[i]))
    print("Dengan point sebesar: %d\n" %l_points[i])

print("Oke! Saatnya memulai permainan!\n")

mulai = time.process_time()
solver(matrix, h, w, [], ukuran_buffer, [], stringseq, l_points,
[0,0], False)
durasi = time.process_time() - mulai

if(final_points == 0):
    print("Maaf ya ternyata gaada solusinya hehe")
    print(final_points)
    print("\n%s s\n" %str(durasi))
else:
    print(final_points)
    print(arr_to_string(final_seq))
    print_final_l_loc()
    print("\n%s s\n" %str(durasi))

tulistxt(final_points, final_seq, final_l_loc, durasi)

```

3.2 Util.py

Merupakan tempat fungsi-fungsi pembantu yang sifatnya sekunder. Sebagai contoh, apabila saya memiliki urutan token ["BD", "FG", "A7", "6R"] dan ingin mengecek apakah terdapat sekuens ["FG", "A7"] pada urutan tersebut, maka masing-masing token dan sekuens akan dijadikan string yang bersambung dengan fungsi buatan `arrtostring(arr)`. Kemudian, digunakan fungsi bawaan `"in"` untuk mengecek apakah suatu string menjadi substring dari string lainnya.

```
import sys

def printmatrix(matrix):
    for i in matrix:
        print(' '.join(map(str, i)))

def arrtostring(arr):
    return (''.join(map(str, arr)))

def arr_to_string(arr):
    return (' '.join(map(str, arr)))

def steptoparagraph(step):
    text = ""
    for i in range (len(step)):
        text += str(step[i][0])
        text += ", "
        text += str(step[i][1])
        text += "\n"
    return text

def tulistxt(point, seq, step, duration):
    simpan = input("Apakah ingin menyimpan solusi? (y/n): ")
    if (simpan == "Y") or (simpan == "y"):
        print("\nYuk masukkan nama file!\nContoh: solusi.txt")
        nama_file = input("Ketik nama file: ")
        with open(nama_file, "w") as file:
            if(point== 0):
```

```

        file.write(str(point) + "\n" + str(duration) + " s")
    else:
        file.write(str(point) + "\n" + arr_to_string(seq) + "\n" +
steptoparagraph(step) + "\n" + str(duration) + " s")

    print("\nFile berhasil dibuat!")
else:
    print("Sip! Sampai jumpa!")
    sys.exit()

```

4 Testing Program

Testing ke-1

```

Selamat datang di game Saiberpang 2045 Breach Protokol!
Kamu mau upload file txt atau generate otomatis nih?
Ketik 1 untuk upload file
Ketik 2 untuk otomatis
Pilihanmu: 1

Yuk masukkan path filemu!
Contoh path: C:/Users/ACER/Downloads/input.txt
Path file mu: D:/ACER/DHIDIT/Desain/testing1.txt
50
7A BD 7A BD 1C BD 55
1, 1
1, 4
3, 4
3, 5
6, 5
6, 3
1, 3

0.46875 s

Apakah ingin menyimpan solusi? (y/n): y

Yuk masukkan nama file!
Contoh: solusi.txt
Ketik nama file: HasilTest1.txt

File berhasil dibuat!

```

input - Notepad

File	Edit	Format	View	Help
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				

HasilTest1.txt U

Testing ke-2

```
Selamat datang di game Saiberpang 2045 Breach Protokol!
Kamu mau upload file txt atau generate otomatis nih?
Ketik 1 untuk upload file
Ketik 2 untuk otomatis
Pilihanmu: 2
Masukkan jumlah token unik: 6
Masukkan token: BD A7 55 EG 4T OL
Masukkan ukuran buffer: 8
Masukkan jumlah kolom matriks: 7
Masukkan jumlah baris matriks: 6
Masukkan jumlah sekuens: 4
Masukkan ukuran maksimal sekuens: 4
['BD', 'A7', '55', 'EG', '4T', 'OL']

[MATRIKS PERMAINAN]
4T 4T 55 OL A7 A7 4T
EG 55 4T A7 A7 4T A7
EG 55 OL BD A7 OL EG
55 BD 55 4T EG OL OL
OL 4T 55 OL 55 55 A7
BD EG 4T BD A7 55 55

[SEQUENCE PERMAINAN]
Sequence ke-1: BD 55 OL 55
Dengan point sebesar: 105

Sequence ke-2: A7 OL 55
Dengan point sebesar: 50

Sequence ke-3: EG BD BD BD
Dengan point sebesar: 95

Sequence ke-4: 4T OL BD
Dengan point sebesar: 60

Oke! Saatnya memulai permainan!
```

```
Oke! Saatnya memulai permainan!

200
4T EG BD BD BD 55 OL 55
1, 1
1, 3
4, 3
4, 6
1, 6
1, 4
6, 4
6, 5

4.5 s

Apakah ingin menyimpan solusi? (y/n): y

Yuk masukkan nama file!
Contoh: solusi.txt
Ketik nama file: HasilTest2.txt

File berhasil dibuat!
```

```
📄 HasilTest2.txt  U
```

Testing ke-3

```
Selamat datang di game Saiberpang 2045 Breach Protokol!
Kamu mau upload file txt atau generate otomatis nih?
Ketik 1 untuk upload file
Ketik 2 untuk otomatis
Pilihanmu: 2
Masukkan jumlah token unik: 4
Masukkan token: AB CD EF GH
Masukkan ukuran buffer: 5
Masukkan jumlah kolom matriks: 4
Masukkan jumlah baris matriks: 4
Masukkan jumlah sekuens: 2
Masukkan ukuran maksimal sekuens: 3
['AB', 'CD', 'EF', 'GH']

[MATRIKS PERMAINAN]
CD CD CD AB
AB CD CD GH
EF GH GH GH
EF CD GH EF

[SEQUENCE PERMAINAN]
Sequence ke-1: AB EF
Dengan point sebesar: 15

Sequence ke-2: CD AB AB
Dengan point sebesar: 40

Oke! Saatnya memulai permainan!
```

```
Oke! Saatnya memulai permainan!

15
CD CD AB EF GH
2, 1
2, 2
1, 2
1, 3
2, 3

0.0 s

Apakah ingin menyimpan solusi? (y/n): Y

Yuk masukkan nama file!
Contoh: solusi.txt
Ketik nama file: HasilTest3.txt

File berhasil dibuat!
```

```
📄 HasilTest3.txt  U
```

Testing ke-4

```
Selamat datang di game Saiberpang 2045 Breach Protokol!
Kamu mau upload file txt atau generate otomatis nih?
Ketik 1 untuk upload file
Ketik 2 untuk otomatis
Pilihanmu: 2
Masukkan jumlah token unik: 5
Masukkan token: AA BB CC DD EE
Masukkan ukuran buffer: 2
Masukkan jumlah kolom matriks: 2
Masukkan jumlah baris matriks: 1
Masukkan jumlah sekuens: 2
Masukkan ukuran maksimal sekuens: 3
['AA', 'BB', 'CC', 'DD', 'EE']

[MATRIKS PERMAINAN]
EE DD

[SEQUENCE PERMAINAN]
Sequence ke-1: DD DD
Dengan point sebesar: 30

Sequence ke-2: EE DD
Dengan point sebesar: 25

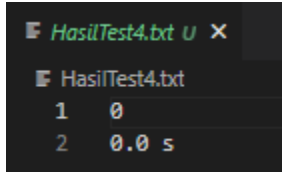
Oke! Saatnya memulai permainan!

Maaf ya ternyata gaada solusinya hehe
0
0.0 s

Apakah ingin menyimpan solusi? (y/n): Y

Yuk masukkan nama file!
Contoh: solusi.txt
Ketik nama file: HasilTest4.txt

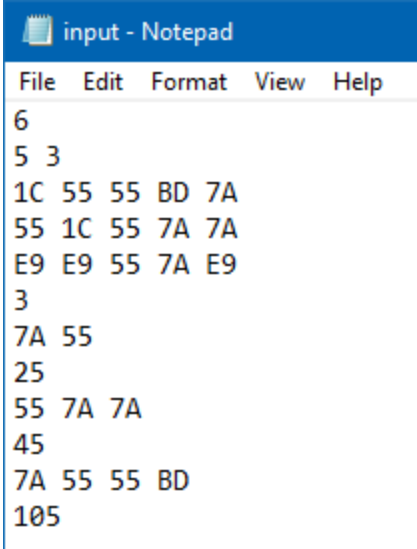
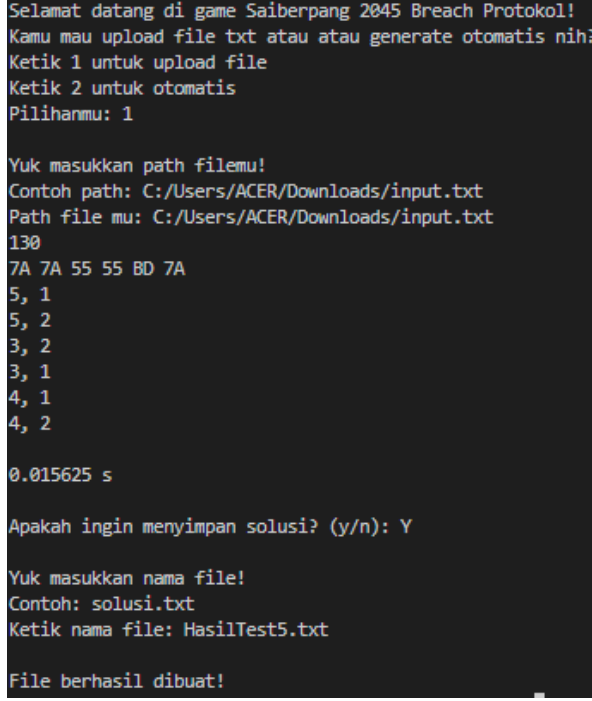
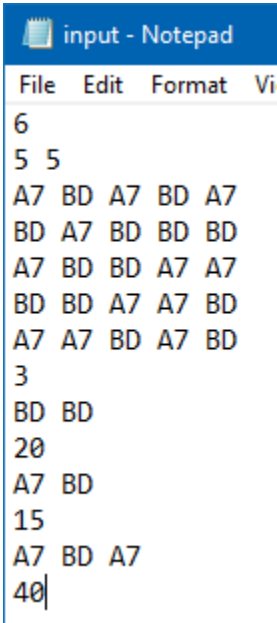
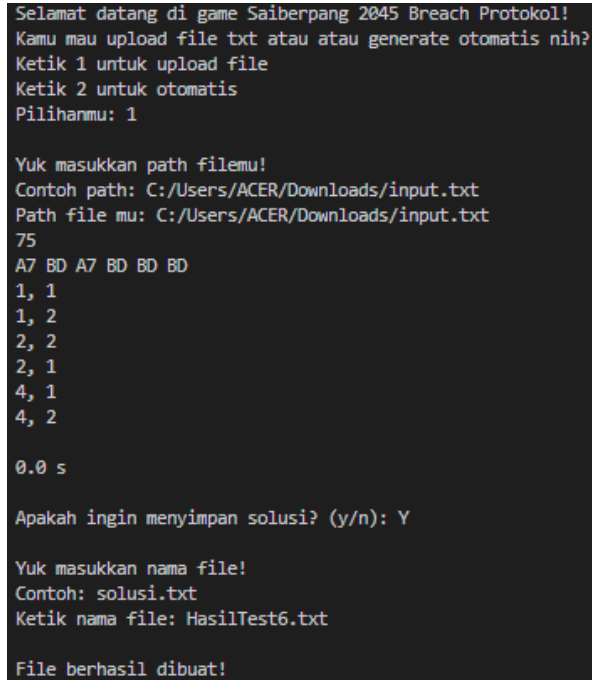
File berhasil dibuat!
```



HasilTest4.txt U X

1	0
2	0.0 s

Testing ke-5

 <pre> input - Notepad File Edit Format View Help 6 5 3 1C 55 55 BD 7A 55 1C 55 7A 7A E9 E9 55 7A E9 3 7A 55 25 55 7A 7A 45 7A 55 55 BD 105 </pre>	 <pre> Selamat datang di game Saiberpang 2045 Breach Protokol! Kamu mau upload file txt atau generate otomatis nih? Ketik 1 untuk upload file Ketik 2 untuk otomatis Pilihanmu: 1 Yuk masukkan path filemu! Contoh path: C:/Users/ACER/Downloads/input.txt Path file mu: C:/Users/ACER/Downloads/input.txt 130 7A 7A 55 55 BD 7A 5, 1 5, 2 3, 2 3, 1 4, 1 4, 2 0.015625 s Apakah ingin menyimpan solusi? (y/n): Y Yuk masukkan nama file! Contoh: solusi.txt Ketik nama file: HasilTest5.txt File berhasil dibuat! </pre>
Testing ke-6	
 <pre> input - Notepad File Edit Format Vi 6 5 5 A7 BD A7 BD A7 BD A7 BD BD BD A7 BD BD A7 A7 BD BD A7 A7 BD A7 A7 BD A7 BD 3 BD BD 20 A7 BD 15 A7 BD A7 40 </pre>	 <pre> Selamat datang di game Saiberpang 2045 Breach Protokol! Kamu mau upload file txt atau generate otomatis nih? Ketik 1 untuk upload file Ketik 2 untuk otomatis Pilihanmu: 1 Yuk masukkan path filemu! Contoh path: C:/Users/ACER/Downloads/input.txt Path file mu: C:/Users/ACER/Downloads/input.txt 75 A7 BD A7 BD BD BD 1, 1 1, 2 2, 2 2, 1 4, 1 4, 2 0.0 s Apakah ingin menyimpan solusi? (y/n): Y Yuk masukkan nama file! Contoh: solusi.txt Ketik nama file: HasilTest6.txt File berhasil dibuat! </pre>

5 Lampiran

No	Poin	Ya	Tidak
1	Program berhasil dikompilasi tanpa kesalahan	V	
2	Program berhasil dijalankan	V	
3	Program dapat membaca masukan berkas.txt	V	
4	Program dapat menghasilkan masukan secara acak	V	
5	Solusi yang diberikan program optimal	V	
6	Program dapat menyimpan solusi dalam berkas.txt	V	
7	Program memiliki GUI		V

6 Pranala Github

https://github.com/dhiabziz/TUCIL1_13522040