

LAPORAN TUGAS KECIL 1
IF2211 STRATEGI ALGORITMA

CYBERPUNK
BREACH PROTOCOL
S O L V E R

Disusun oleh:

Muhammad Naufal Aulia (13522074)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2024

DAFTAR ISI

DAFTAR ISI.....	1
BAB I.....	2
1.1 Cyberpunk 2077 Breach Protocol.....	2
1.2 Algoritma Brute Force.....	2
1.3 Algoritma Brute Force dalam Penyelesaian Cyberpunk 2077 Breach Protocol.....	2
BAB II.....	5
2.1 Source Code: Algoritma Brute Force.....	5
2.2 Source Code: Input (File & Auto Keyboard) dan Output (Save File).....	9
BAB III.....	12
3.1 Tampilan Program (GUI).....	12
3.2 Input File.....	15
3.3 Input Keyboard.....	20
BAB IV.....	22
4.1 Link Repository.....	22
4.2 Checklist.....	22
REFERENSI.....	23

BAB I

DESKRIPSI MASALAH

1.1 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.
2. Sekuens – sebuah rangkaian token (dua atau lebih) yang harus dicocokkan.
3. Buffer – jumlah maksimal token yang dapat disusun secara sekuensial.

Aturan permainan Breach Protocol antara lain:

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

1.2 Algoritma Brute Force

Brute force adalah sebuah pendekatan yang lempang (straightforward) dalam memecahkan suatu masalah, didasarkan pada pernyataan masalah (problem statement) dan melibatkan definisi konsep. Algoritma ini memecahkan masalah secara langsung dengan cara yang jelas. Dengan mencoba semua kemungkinan solusi secara langsung, algoritma ini tidak memperhatikan efisiensi sehingga pendekatan ini kadang memakan waktu dan sumber daya yang besar

1.3 Algoritma Brute Force dalam Penyelesaian Cyberpunk 2077 Breach Protocol

Dalam menyelesaikan permainan cyberpunk 2077 breach protocol, penulis menggunakan algoritma brute force. Algoritma brute force yang akan mencoba semua kemungkinan solusi yang

ada, dan memilih solusi yang paling optimal. Adapun langkah-langkah yang penulis lakukan adalah sebagai berikut:

1. Program mendapat input berupa matrix berisi token sebagai elemennya, beberapa sekuens yang harus dicocokkan, dan buffer sebagai jumlah maksimal token yang dapat disusun secara sekuensial.
2. Jika input dilakukan melalui masukkan keyboard, maka program akan membuat matrix secara random, sekuens secara random, dan reward masing-masing sekuens secara random.
3. Dalam mempersiapkan brute force pada persoalan ini, akan dibuat sebuah array penampung koordinat yang telah digunakan/dikunjungi yakni `used_coordinates`
4. Buat sebuah fungsi untuk memastikan pergerakan enumerasi sesuai, yakni untuk gerak horizontal harus berada di dalam panjang baris dan gerak vertikal harus berada di dalam panjang kolom. Koordinat keduanya harus belum pernah dikunjungi (gunakan pengecekan di array `used_coordinates`)
5. Untuk enumerasi semua kemungkinan susunan sekuens, akan disiapkan dua array penampung hasil susunan sekuens dan koordinatnya. Untuk ini, disiapkan sebuah fungsi `enumerate_combinations` yang akan melakukan gerakan horizontal menggunakan sebuah loop sepanjang row, tambahkan koordinat ke dalam array `used_coordinates`, lanjut panggil ulang fungsi `enumerate_combinations` namun dengan mengubah arah menjadi vertikal dan mengurangi buffer. Ubah juga parameter kombinasi sesuai keberjalanan loop. Setelah selesai secara rekursif, buffer akan habis dan semua susunan kemungkinan akan ditambahkan ke dalam array hasil susunan sekuens dan koordinatnya. Kemudian hapus koordinat yang sudah digunakan tadi agar dapat memulai enumerasi kemungkinan sel matrix selanjutnya. Dengan begitu, set gerakan horizontal pertama telah selesai. Lanjutkan dengan gerakan vertikal menggunakan semua proses yang sama seperti sebelumnya kecuali dengan mengubah pergerakan loop terfokus di row kemudian mengubah arahnya menjadi horizontal kembali.
6. Gunakan fungsi `enumerate_combinations` untuk memulai enumerasi dari setiap sel di baris pertama (dengan loop sepanjang kolom) dan lakukan semua proses enumerasi yang telah dijelaskan sebelumnya. Nantinya semua kombinasi akan ditambahkan ke array penampung hasil ketika sudah mencapai satu sekuens.
7. Lakukan penilaian reward terhadap semua kemungkinan susunan sekuens yang telah dienumerasi. Untuk ini, buat sebuah fungsi `rewarding` yang akan mengembalikan list

reward_candidate berisi jumlah reward untuk setiap kemungkinan susunan sekuens. Untuk melakukan penilaian reward, gunakan loop untuk setiap kemungkinan susunan sekuens, lalu gunakan loop lagi untuk setiap sekuens yang ada di dalam list sekuens yang diberikan. Jika sekuens yang ada di dalam list sekuens ada di dalam kemungkinan susunan sekuens, maka jumlahkan rewardnya. Setelah semua penilaian selesai, cari index reward maksimal dari list reward_candidate dan ambil reward maksimalnya. Dari index tersebut, ambil susunan sekuens dan koordinatnya yang sesuai.

8. Dari proses tersebut didapatlah hasil solusi yang akan ditampilkan berupa susunan optimal sekuens dan koordinatnya, untuk mendapat reward yang maksimal

BAB II

IMPLEMENTASI ALGORITMA

Implementasi algoritma ini penulis buat dalam bahasa pemrograman Python yang terdiri dari satu file yaitu `cyberpunk77.py` yang berisi implementasi algoritma brute force yang telah dijelaskan sebelumnya sekaligus strukturisasi GUI tkinter sebagai antarmuka pengguna dalam menjalankan program.

Program ini menggunakan library-library python sebagai berikut:

- `os`: untuk mengakses file
- `random`: untuk menghasilkan randomisasi
- `time`: untuk menghitung waktu eksekusi
- `sys`: untuk mengakses path file
- `tkinter`: untuk membuat GUI

2.1 Source Code: Algoritma Brute Force

```
# BRUTE FORCE ALGORITHM
def solve():
    # Inisialisasi penyimpanan koordinat yang sudah digunakan/dikunjungi
    used_coordinates = set()

    # Fungsi untuk mengecek gerakan enumerasi kombinasi adalah valid
    def is_valid_move(row, col, direction):
        if direction == 'horizontal':
            return col < len(matrix_arr[0]) and (row, col) not in used_coordinates
        elif direction == 'vertical':
            return row < len(matrix_arr) and (row, col) not in used_coordinates

    # Fungsi untuk mengenumerasi kombinasi dari matrix token
    sequences_result = []
    coordinate_result = []

    def enumerate_combinations(row, col, direction, buffer_size, combination,
                               combination_coord):
        # Sudah mencukupi buffer size, tambahkan kombinasi dan koordinatnya ke dalam list
        hasil

        if buffer_size == 0:
            coordinate_result.append(combination_coord)
            return sequences_result.append(combination)
```

```

# Enumerasi tiap sel matrix secara berarah sesuai aturan, horizontal lalu vertikal
if direction == 'horizontal':
    for i in range(len(matrix_arr[0])):
        if is_valid_move(row, i, direction):
            used_coordinates.add((row, i))
            enumerate_combinations(row, i, 'vertical', buffer_size - 1, combination
+ [matrix_arr[row][i]], combination_coord + [(row, i)])
            used_coordinates.remove((row, i))
        elif direction == 'vertical':
            for i in range(len(matrix_arr)):
                if is_valid_move(i, col, direction):
                    used_coordinates.add((i, col))
                    enumerate_combinations(i, col, 'horizontal', buffer_size - 1,
combination + [matrix_arr[i][col]], combination_coord + [(i, col)])
                    used_coordinates.remove((i, col))

# Timer proses
start_time = process_time()

# Melakukan enumerasi kombinasi dari setiap sel dimulai dari row pertama sel pertama
for i in range(len(matrix_arr[0])):
    used_coordinates.clear()
    used_coordinates.add((0, i))
    enumerate_combinations(0, i, 'vertical', buffer_size - 1, [matrix_arr[0][i]], [(0,
i)])

# Update hasil koordinat (dari 0-based ke 1-based)
coordinate_result_update = []
for sub_list in coordinate_result:
    updated_sub_list = []
    for tup in sub_list:
        updated_tuple = tuple(element + 1 for element in tup)
        updated_sub_list.append(updated_tuple)
    coordinate_result_update.append(updated_sub_list)

# Swap elemen koordinat (row, col) ke (col, row)
coordinate_result_reversed = []
for sub_list in coordinate_result_update:
    reversed_sub_list = [(tup[1], tup[0]) for tup in sub_list]
    coordinate_result_reversed.append(reversed_sub_list)

```

```

# Fungsi untuk mekanisme rewarding
def rewarding(sequences_result, sequences_list, reward_list):
    reward_candidate = []
    for array_result in sequences_result:
        sum_reward = 0
        for index in range(len(sequences_list)):
            array_list = sequences_list[index]
            reward = reward_list[index]

            array_result_str = ' '.join(array_result)
            array_list_str = ' '.join(array_list)
            if array_list_str in array_result_str:
                sum_reward += reward

        reward_candidate.append(sum_reward)
    return reward_candidate

# FINAL SOLUTION
global max_reward
global sequences_result_final
global coordinate_result_final
global timer

# cari index reward maksimal dari list reward_candidate
index_reward = rewarding(sequences_result, sequence_list, reward_list).index(max(rewarding(sequences_result, sequence_list, reward_list)))
max_reward = max(rewarding(sequences_result, sequence_list, reward_list))

# hasil solusi
sequences_result_final = ' '.join(sequences_result[index_reward])
coordinate_result_final = coordinate_result_reversed[index_reward]

stop_time = process_time()
timer = round((stop_time - start_time)*1000, 2)

# OUTPUTING SOLUTION
print("\nHASIL: ")
if(max_reward == 0):
    print(f'Reward maksimal: {max_reward}')
    print('Sekuens: - ')
    print('Koordinat: - ')

```



```

print(f'\nWaktu eksekusi: {timer} ms')

coordinate_result[index_reward] = []
sequences_result_final = '

else:
    print(f'Reward maksimal: {max_reward}')
    print(f'Sekuens: {sequences_result_final}')
    print('Koordinat: ')
    for coord in coordinate_result_final:
        print(f'{coord}')
    print(f'\nWaktu eksekusi: {timer} ms')

# Display GUI
draw_matrix_with_lines(matrix_arr, coordinate_result[index_reward], page2)
max_reward_result.config(text=max_reward)
sequence_result.config(text=sequences_result_final)
time_result.config(text=f'{timer} ms')

draw_matrix_with_lines(matrix_arr, coordinate_result[index_reward], page3)
max_reward_result3.config(text=max_reward)
sequence_result3.config(text=sequences_result_final)
time_result3.config(text=f'{timer} ms')

# Save Button
save_path = resource_path("assets/save.png")
save_btn_img = PhotoImage(file=save_path)
save_label = Label(page2, image=save_btn_img, bg='#0B0F28')
save_label.image = save_btn_img
save_label.place(x=64.8, y=595)

save_label3 = Label(page3, image=save_btn_img, bg='#0B0F28')
save_label3.image = save_btn_img
save_label3.place(x=64.8, y=595)

save_btn = Button(page2, text='S A V E', font=('Microsoft YaHei UI',13), bg='#1C2A41',
fg='#95E9FA', relief=FLAT, command=save_file)
save_btn.place(x=137, y=624)

save_btn3 = Button(page3, text='S A V E', font=('Microsoft YaHei UI',13), bg='#1C2A41',
fg='#95E9FA', relief=FLAT, command=save_file)
save_btn3.place(x=137, y=624)

```

```

solve_label.destroy()
solve_btn.destroy()
solve_label3.destroy()
solve_btn3.destroy()

```

2.2 Source Code: Input (File & Auto Keyboard) dan Output (Save File)

```

# INPUT FILE
def open_file_dialog():
    global matrix_arr
    global buffer_size
    global sequence_list
    global reward_list
    file_path = filedialog.askopenfilename()
    if file_path:
        file_name = os.path.basename(file_path)
        selected_file_label.config(text=file_name)
        with open(file_path, 'r') as file:
            buffer_size = int(file.readline().strip())
            matrix_size = file.readline().split()
            matrix_width = int(matrix_size[0])
            matrix_height = int(matrix_size[1])
            matrix_arr = []
            for i in range(matrix_height):
                matrix_arr.append(file.readline().strip().split())

            n_sequences = int(file.readline().strip())

            # Data: 2 array: 1 array untuk sequence, 1 array untuk reward
            sequence_list = []
            reward_list = []
            for i in range(n_sequences):
                sequence = file.readline().strip().split()
                reward_list.append(int(file.readline().strip()))
                sequence_list.append(sequence)

        return matrix_arr, matrix_width, matrix_height, buffer_size, matrix_size, n_sequences,
sequence_list, reward_list

# INPUT KEYBOARD

```

```

def open_keyboard_input():
    global matrix_arr
    global buffer_size
    global sequence_list
    global reward_list

    n_token = int(token_amount_input.get())
    token = str(token_input.get())
    buffer_size = int(buffer_size_input.get())
    matrix_size = str(matrix_size_input.get())
    n_sequences = int(sequence_amount_input.get())
    max_sequence_size = int(max_sequence_input.get())

    token_arr = token.split()
    matrix_width = int(matrix_size.split()[1])
    matrix_height = int(matrix_size.split()[0])

    # Matrix generator
    matrix_arr = [['' for i in range(matrix_height)] for j in range(matrix_width)]
    for i in range(matrix_width):
        for j in range(matrix_height):
            random_token = random.choice(token_arr)
            matrix_arr[i][j] = random_token

    # Sequence & reward generator
    sequence_list = []
    reward_list = []
    for i in range(n_sequences):
        sequence = []
        for j in range(random.randint(2, max_sequence_size)):
            random_token = random.choice(token_arr)
            sequence.append(random_token)
        sequence_list.append(sequence)
        reward_list.append(random.randint(10, 100))

    return matrix_arr, token_arr, matrix_width, matrix_height, n_token, token, buffer_size,
matrix_size, n_sequences, max_sequence_size, sequence_list, reward_list

# SAVE FILE GUI
def save_file():
    file_path = filedialog.asksaveasfilename(defaultextension=".txt", filetypes=[("Text
files", "*.txt"), ("All files", "*.*")])

```

```
if file_path:
    with open(file_path, 'w') as file:
        file.write(f'{max_reward}\n')
        if(max_reward != 0):
            file.write(f'{sequences_result_final}\n')
            for coord in coordinate_result_final:
                file.write(f'{coord}\n')
            file.write(f'\n{timer} ms\n')

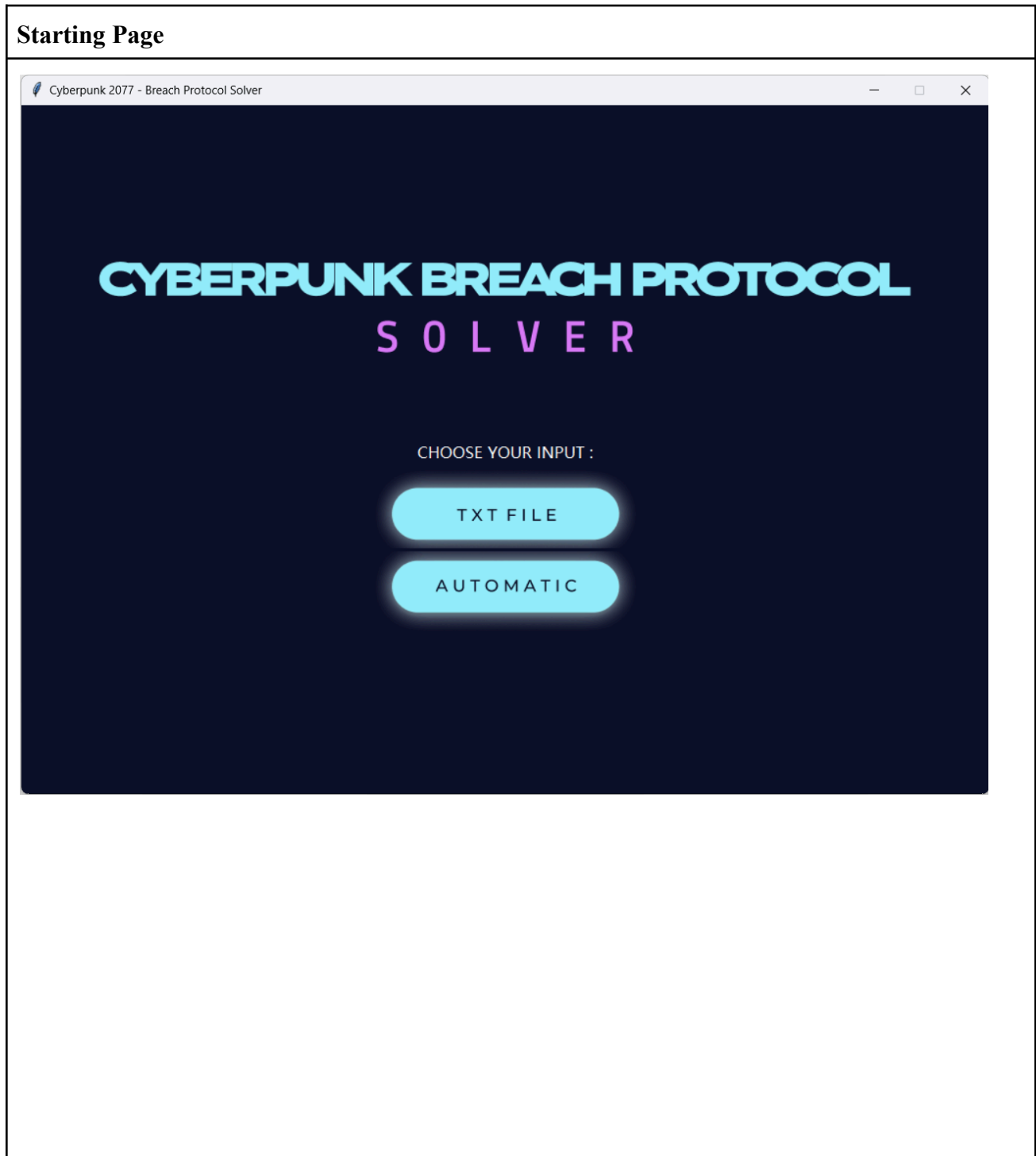
messagebox.showinfo("Save", f"Saved on {file_path}")
```

Untuk *source code* GUI karena terlalu panjang untuk ditampilkan, maka dapat langsung dilihat melalui pranala github repository pada bab iv di bawah dokumen ini.

BAB III

TANGKAPAN LAYAR HASIL PENGUJIAN

3.1 Tampilan Program (GUI)



Txt Input Page

Cyberpunk 2077 - Breach Protocol Solver

TXT FILE INPUT

MATRIX:

BACK

BROWSE

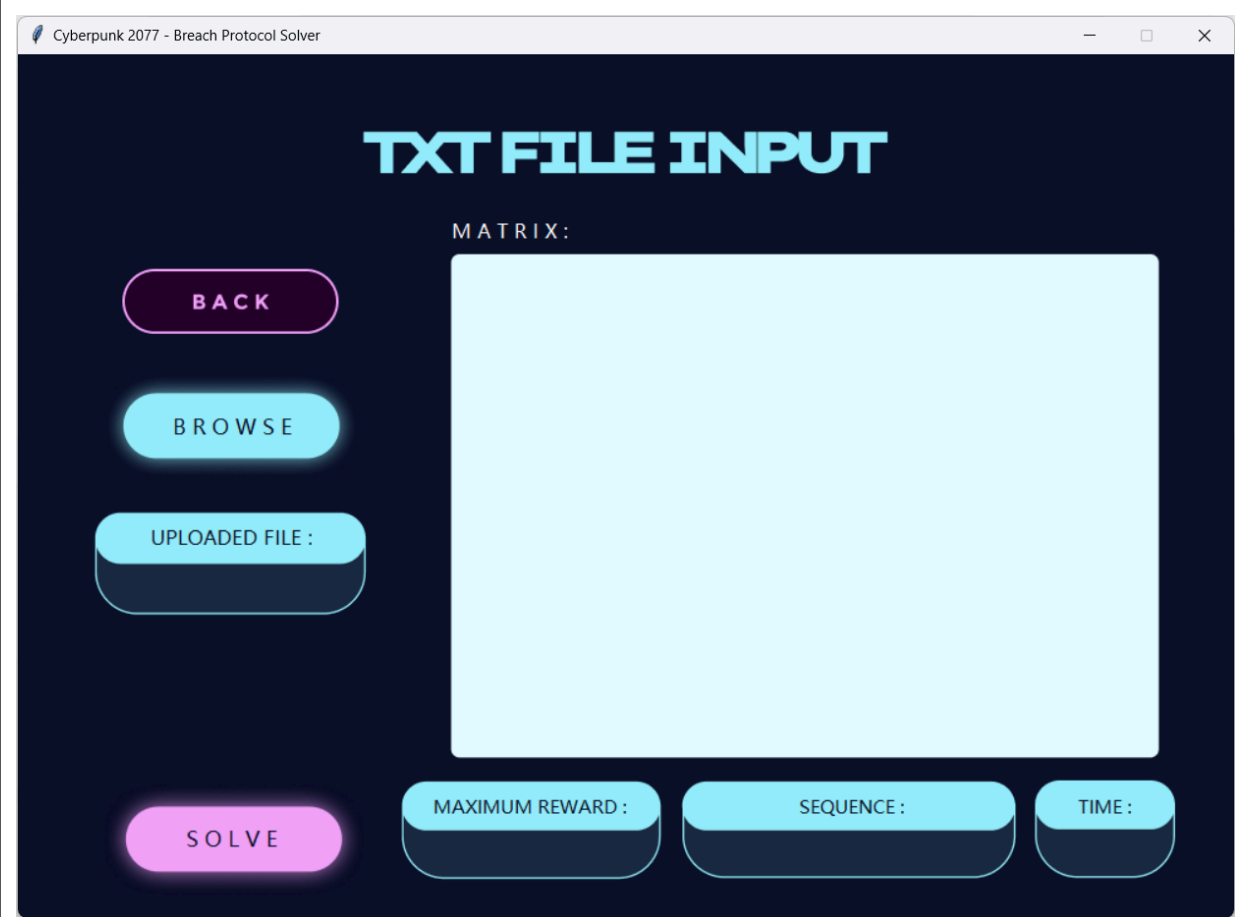
UPLOADED FILE :

SOLVE

MAXIMUM REWARD :

SEQUENCE :

TIME :



Automatic Input Page

Cyberpunk 2077 - Breach Protocol Solver

— □ ×

BACK

TOKEN AMOUNT:

TOKEN:

BUFFER SIZE:

MATRIX SIZE:

SEQUENCE AMT:

MAX SEQUENCE:

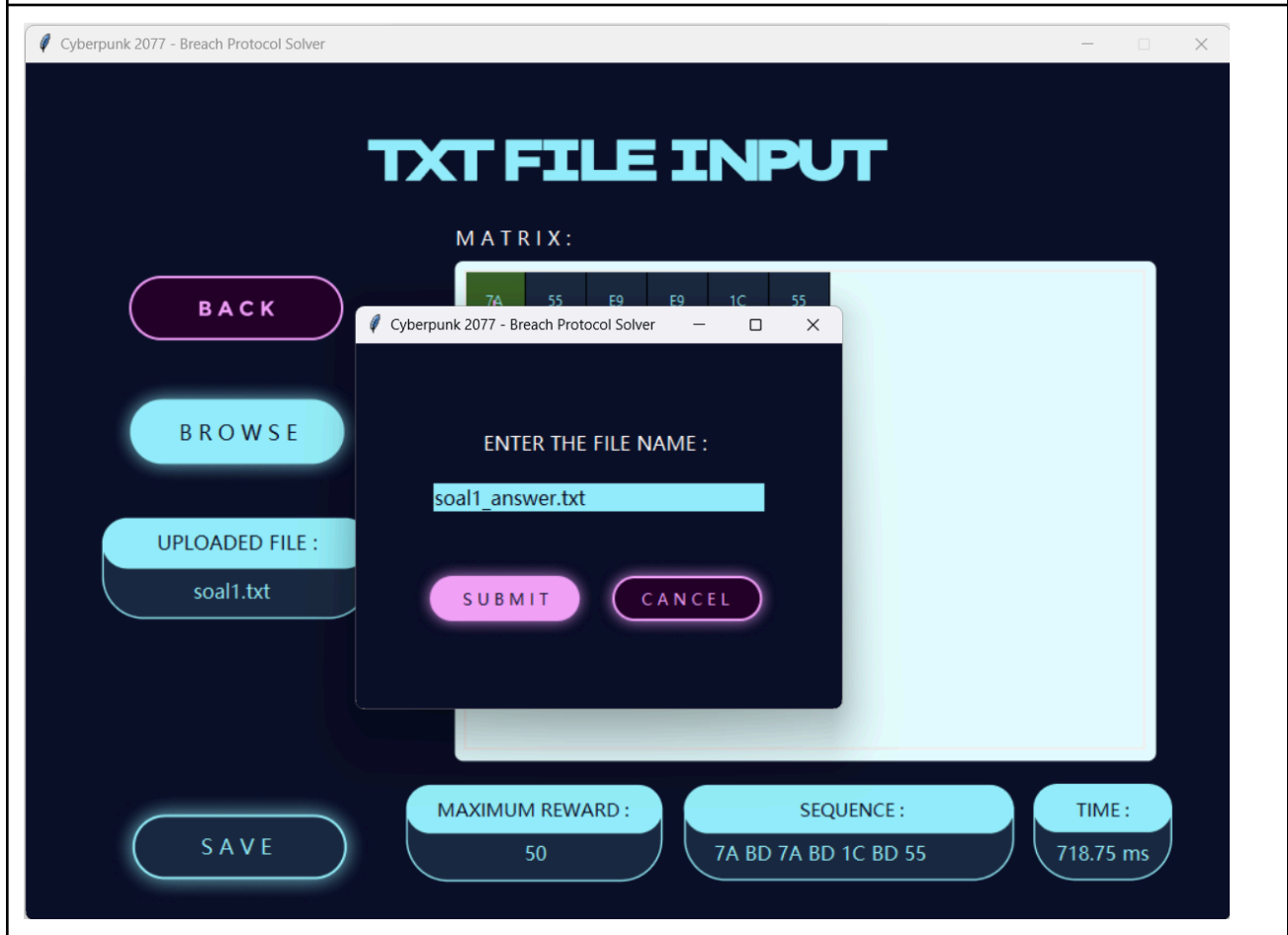
SOLVE

AUTO INPUT

MATRIX:

MAXIMUM REWARD : SEQUENCE : TIME :

Result Saving Option



3.2 Input File

Keterangan: pada output (txt), koordinat yang dihasilkan adalah berupa (col, row)

Pada display matrix output, sel token berwarna hijau adalah sel dimana lintasan sekuens optimal dimulai, dilanjutkan pada sel token berwarna kuning, dan diakhiri pada sel token berwarna merah.

Nama File: soal2.txt

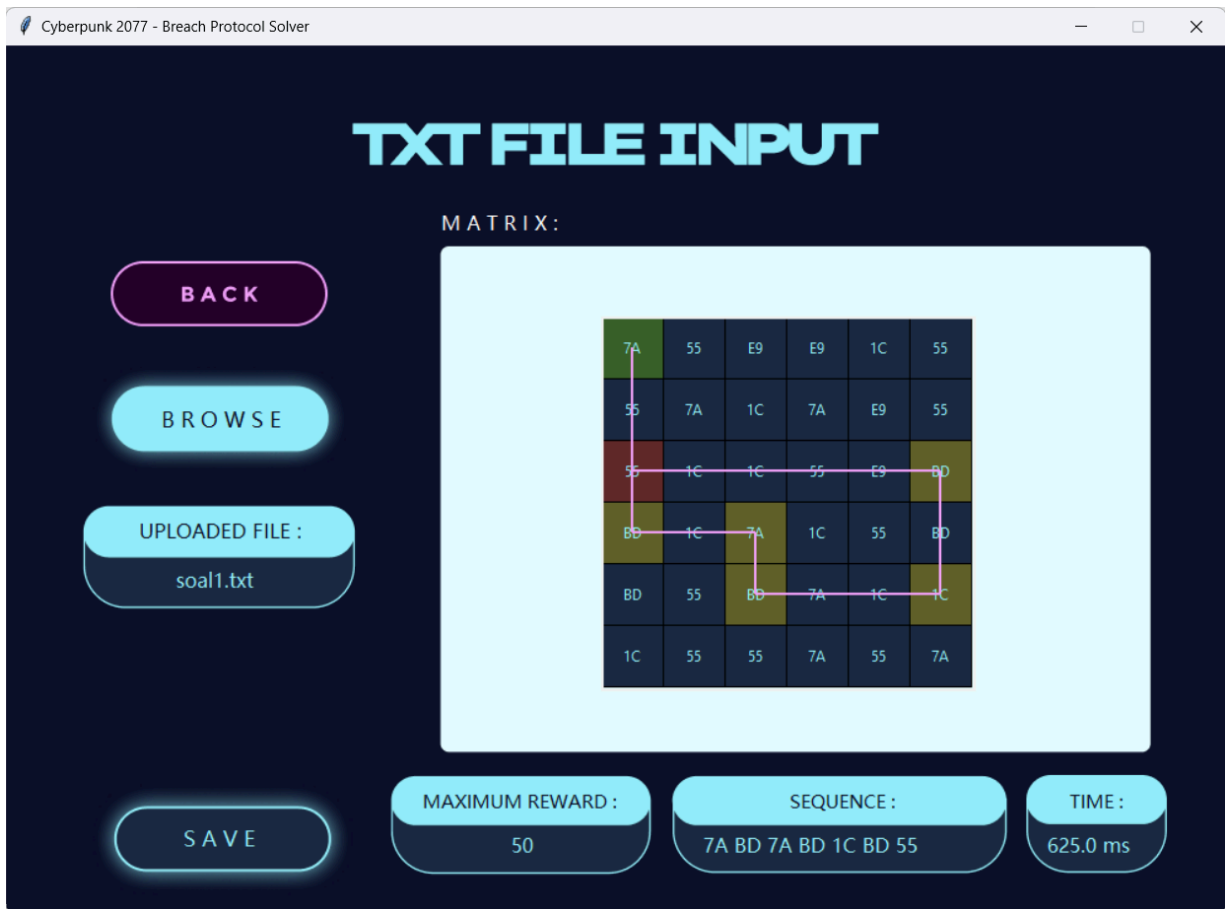
Input:

```
soal1.txt
1 7
2 6 6
3 7A 55 E9 E9 1C 55
4 55 7A 1C 7A E9 55
5 55 1C 1C 55 E9 BD
6 BD 1C 7A 1C 55 BD
7 BD 55 BD 7A 1C 1C
8 1C 55 55 7A 55 7A
9 3
10 BD E9 1C
11 15
12 BD 7A BD
13 20
14 BD 1C BD 55
15 30
```

Output (txt):

```
test > soal1_answer.txt
1 50
2 7A BD 7A BD 1C BD 55
3 (1, 1)
4 (1, 4)
5 (3, 4)
6 (3, 5)
7 (6, 5)
8 (6, 3)
9 (1, 3)
10
11 625.0 ms
12
```

Output (GUI):



Keterangan: jika diperhatikan, solusi koordinat buffer sekuens pada program berbeda dengan yang terdapat pada spek Tupil 1. Hal ini disebabkan oleh adanya kemungkinan yang sama-sama optimal dengan besar reward yang juga sama. Pada algoritma saya, enumerasi kombinasi solusi sekuens dilakukan mulai dari sel teratas dahulu (dalam lajur vertikal) dan kiri dahulu (dalam lajur horizontal) sehingga sekuens optimal pertama yang ditemukan adalah sebagaimana pada gambar di atas.

Nama File: soal2.txt

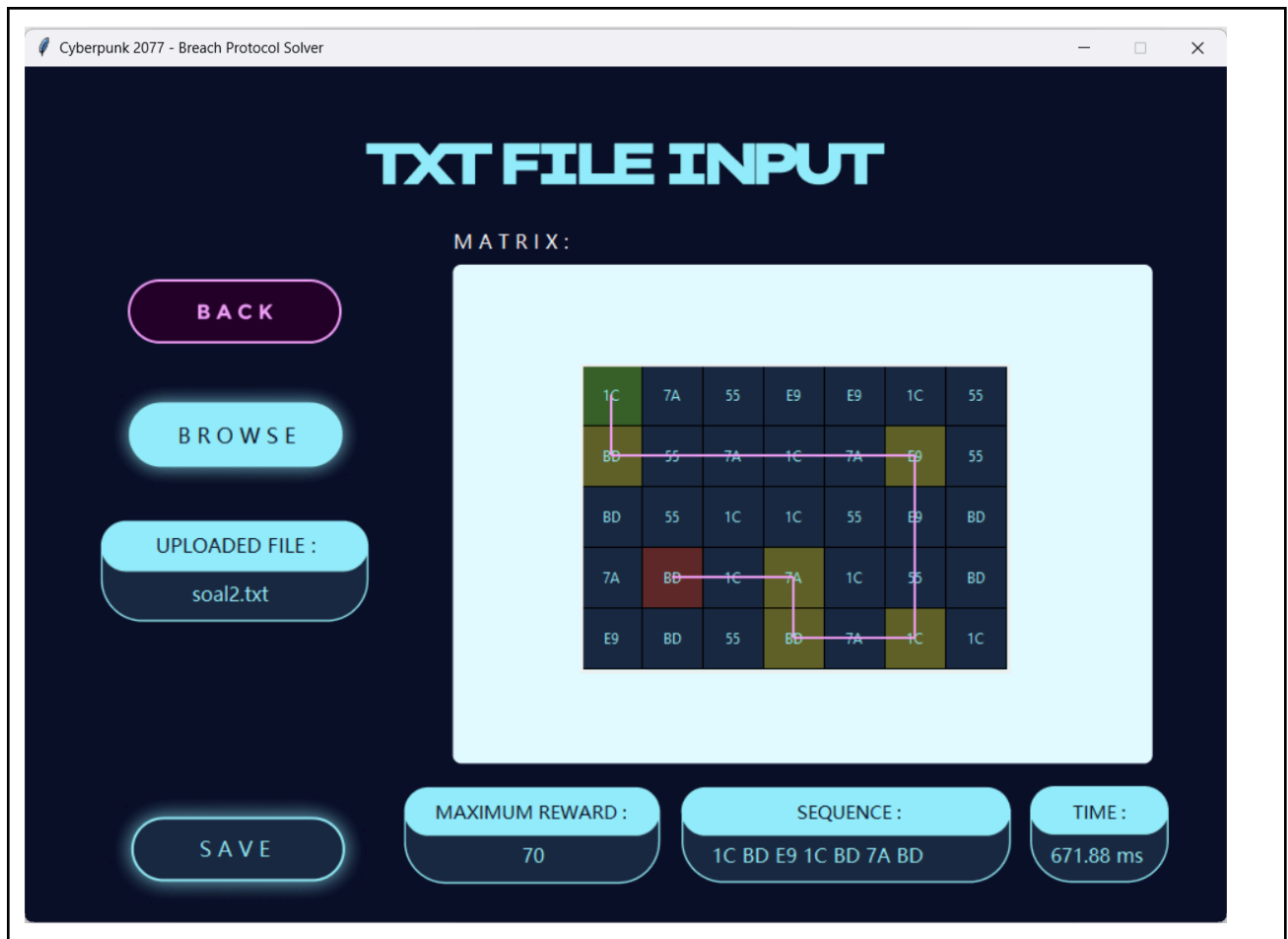
Input:

```
soal2.txt
1  7
2  7 5
3  1C 7A 55 E9 E9 1C 55
4  BD 55 7A 1C 7A E9 55
5  BD 55 1C 1C 55 E9 BD
6  7A BD 1C 7A 1C 55 BD
7  E9 BD 55 BD 7A 1C 1C
8  3
9  BD E9 1C
10 50
11 BD 7A BD
12 20
13 BD 1C BD 55
14 30
```

Output (txt):

```
test > soal2_answer.txt
1  70
2  1C BD E9 1C BD 7A BD
3  (1, 1)
4  (1, 2)
5  (6, 2)
6  (6, 5)
7  (4, 5)
8  (4, 4)
9  (2, 4)
10
11 671.88 ms
12
```

Output (GUI):



Nama File: salah.txt	
Input:	Output (txt):

```

salah.txt
1 7
2 6 6
3 7A 55 E9 E9 1C 55
4 55 7A 1C 7A E9 55
5 55 1C 1C 55 E9 BD
6 BD 1C 7A 1C 55 BD
7 BD 55 BD 7A 1C 1C
8 1C 55 55 7A 55 7A
9 3
10 AA BB CC
11 15
12 DD EE FF
13 20
14 ZZ XX GG
15 30

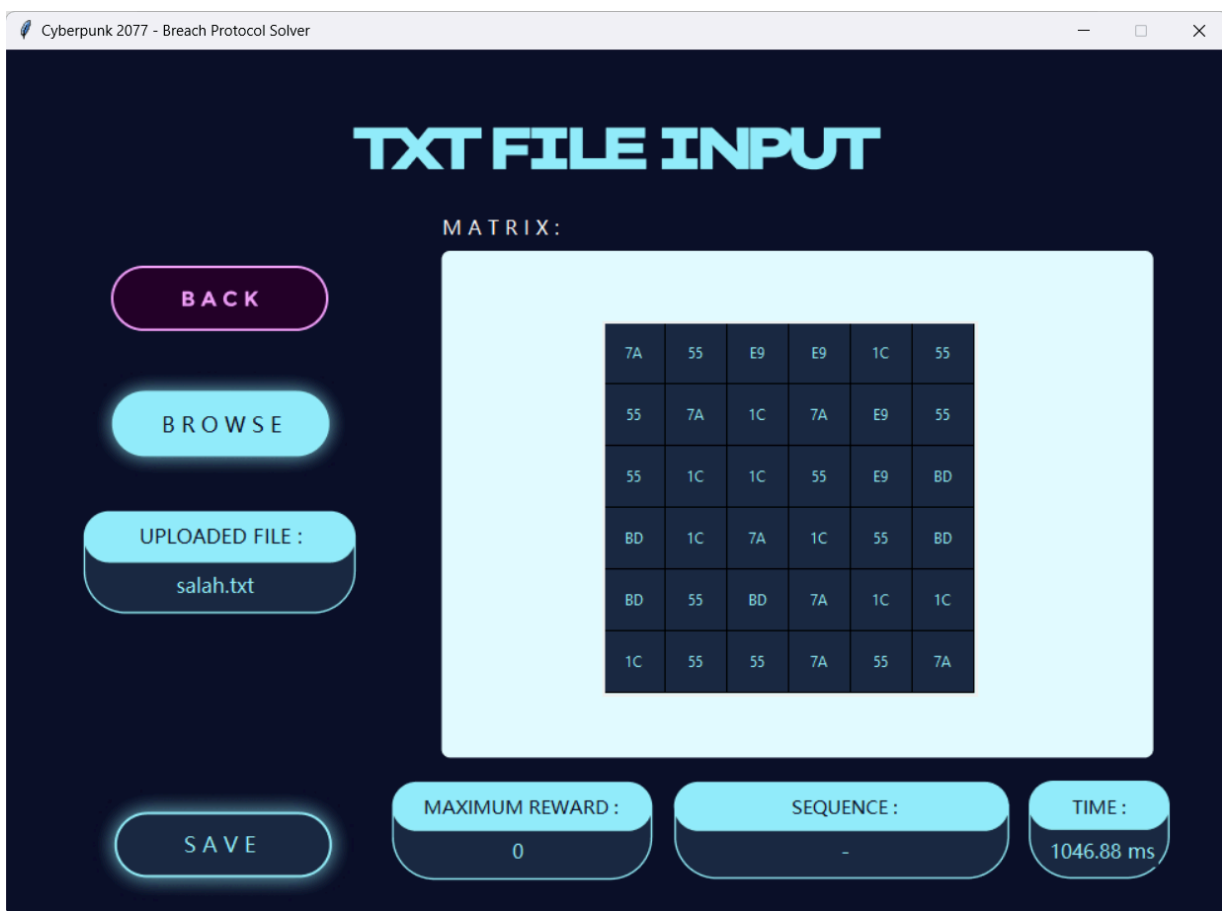
```

```

test > salah_answer.txt
1 0
2
3 1046.88 ms
4

```

Output (GUI):

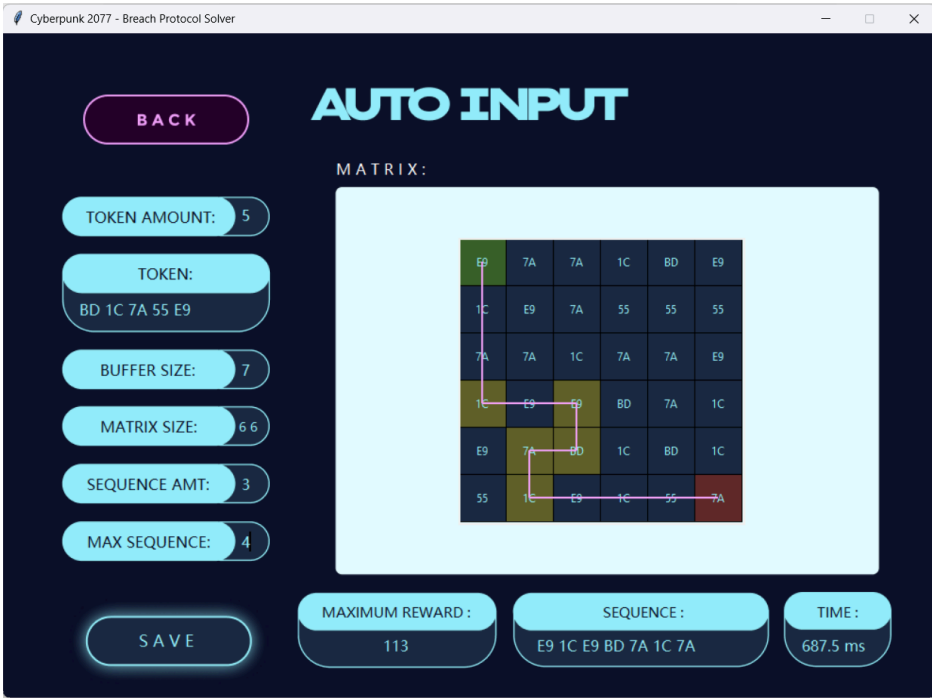


Keterangan: pada contoh input ini, dipastikan tidak ada buffer yang memenuhi sekuens sehingga didapat maximum reward 0 dan tidak ada buffer sekuens.

3.3 Input Keyboard

Keterangan: pada output (txt), koordinat yang dihasilkan adalah berupa (col, row).

Isi matrix dan isi sekuens yang akan dicocokkan adalah hasil randomisasi token unik yang dimasukkan pengguna. Matrix dirandom sesuai ukuran row dan col-nya, sekuens dirandom sebanyak 2 s.d. maksimal buffer dari pengguna, dan reward tiap sekuens dirandom dari 10 s.d. 100.

Input/Output (GUI)	Output (txt)
	<pre>auto1_answer.txt 113 E9 1C E9 BD 7A 1C 7A (1, 1) (1, 4) (3, 4) (3, 5) (2, 5) (2, 6) (6, 6) 687.5 ms</pre>

Cyberpunk 2077 - Breach Protocol Solver

BACK

TOKEN AMOUNT: 5

TOKEN: BD 1C 7A 55 E9

BUFFER SIZE: 6

MATRIX SIZE: 5 5

SEQUENCE AMT: 3

MAX SEQUENCE: 4

SAVE

MAXIMUM REWARD: 81

SEQUENCE: BD 1C 1C BD BD 7A

TIME: 15.62 ms

AUTO INPUT

MATRIX:

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

auto2_answer.txt
81
BD 1C 1C BD BD 7A
(1, 1)
(1, 5)
(3, 5)
(3, 1)
(4, 1)
(4, 2)

15.62 ms

Cyberpunk 2077 - Breach Protocol Solver

BACK

TOKEN AMOUNT: 6

TOKEN: BD 1C 7A 55 E9 FF

BUFFER SIZE: 8

MATRIX SIZE: 7 6

SEQUENCE AMT: 3

MAX SEQUENCE: 5

SAVE

MAXIMUM REWARD: 97

SEQUENCE: 1C 55 BD BD 1C BD FF BD

TIME: 7968.75 ms

AUTO INPUT

MATRIX:

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

auto3_answer.txt
97
1C 55 BD BD 1C BD FF BD
(1, 1)
(1, 2)
(2, 2)
(2, 4)
(5, 4)
(5, 5)
(1, 5)
(1, 6)

7968.75 ms

BAB IV LAMPIRAN

4.1 Link Repository

Github: https://github.com/NopalAul/Tucil1_13522074

4.2 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	✓	

REFERENSI

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-\(2022\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf)

<https://repository.unikom.ac.id/37037/1/BruteForce%28bagian%201%29.pdf>