LAPORAN TUGAS KECIL I IF2211 STRATEGI ALGORITMA
SEMESTER II 2023/2024
"PENYELESAIAN CYBERPUNK 2077 BREACH PROTOCOL DENGAN ALGORITMA BRUTE FORCE"

Ibrahim Ihsan Rasyid 13522018

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

DAFTAR ISI

BAB I
DESKRIPSI MASALAH

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.

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.

Ilustrasi kasus :

Diberikan matriks sebagai berikut dan ukuran buffernya adalah tujuh

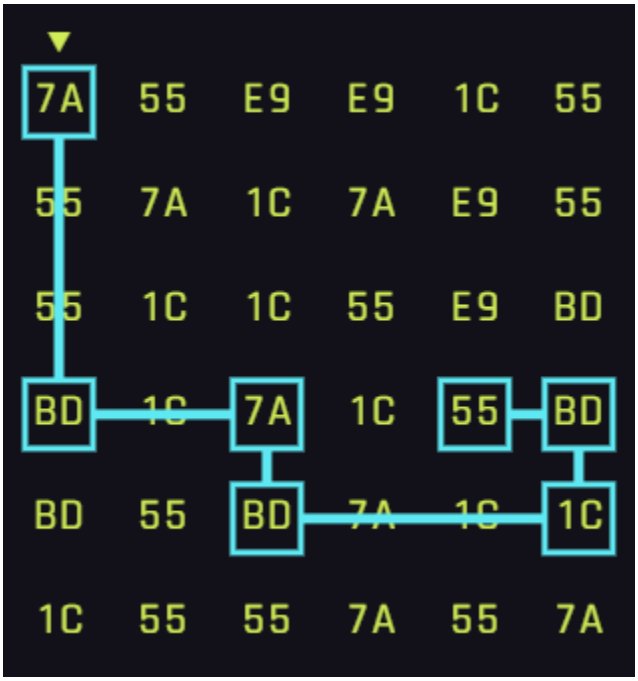
7A	55	E9	E9	1C	55
55	7A	1C	7A	E9	55
55	1C	1C	55	E9	BD
BD	1C	7A	1C	55	BD
BD	55	BD	7A	1C	1C
1C	55	55	7A	55	7A

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



Gambar 1 Contoh Solusi
(Sumber: https://cyberpunk-hacker.com/)

BAB II
TEORI SINGKAT ALGORITMA

Algoritma *Brute Force* merupakan salah satu strategi algoritma yang melakukan pendekatan secara langsung (straight-forward) untuk memecahkan suatu masalah, biasanya langsung berdasarkan pada pernyataan masalah dan definisi konsep yang terlibat. Algoritma *brute force* memecahkan persoalan dengan sangat sederhana, langsung, jelas caranya, dan "*just do it!"*.

Algoritma *Brute Force* pada umumnya sederhana dan mudah diimplementasikan, namun membutuhkan resource yang banyak baik dalam segi memori maupun waktu eksekusi karena algoritma ini memeriksa semua kemungkinan. Oleh karena itu, Algoritma *Brute Force* biasanya hanya digunakan untuk permasalahan-permasalahan yang memiliki masukan tidak terlalu banyak atau ketika waktu eksekusi dan memori bukanlah hal yang perlu diutamakan. Algoritma *Brute Force* sering digunakan sebagai basis pembanding dengan algoritma lain yang lebih mangkus

Meskipun bukan metode *problem solving* yang mangkus, hampir semua persoalan dapat diselesaikan dengan algoritma *brute force*. Ini adalah kelebihan *brute force*. Sukar menunjukkan persoalan yang tidak dapat diselesaikan dengan metode *brute force*. Bahkan, ada persoalan yang hanya dapat diselesaikan dengan *brute force*, seperti mencari elemen terbesar dalam sebuah senarai

Pada tugas kecil ini, persoalan yang ingin diselesaikan adalah mencari solusi dari permainan Breach Protocol yang paling optimal untuk setiap kombinasi matriks, sekuens, dan ukuran buffer dengan menggunakan algoritma brute force. Program akan mencari tiap langkah yang mungkin dilalui, menghitung bobot hadiah, lalu mencari langkah yang memiliki bobot hadiah terbesar.

Implementasi program Breach Protocol yang dibuat oleh penulis menggunakan langkah-langkah berikut:

1. Program meminta metode input dari pengguna. Terdapat dua pilihan, yaitu input dari import file berekstensi .txt atau meminta program untuk men-generate matriks dan sekuens beserta bobot hadiahnya
2. Program menyimpan matriks, sekuens, dan ukuran buffer
3. Program melakukan brute force dengan matriks, sekuens, dan ukuran buffer tersebut untuk mendapatkan bobot hadiah terbesar beserta langkah-langkahnya dalam mendapatkannya. Brute force dilakukan dengan rekursif dengan mengenumerasikan setiap kemungkinan langkah secara vertikal/horizontal secara bergantian
4. Program selesai mendapatkan bobot hadiah terbesar beserta langkah-langkahnya dan waktu eksekusi lalu menampilkannya
5. Program meminta input pengguna untuk menyimpan hasil pencarian tersebut ke file berekstensi .txt atau tidak. Jika iya, program akan meminta input nama file dari pengguna

BAB III
IMPLEMENTASI PADA PROGRAM

A. Kode Utama Implementasi Brute Force

```
def protocol(matrix, buff_size, vert, seq, rc, prev_idx, sequences, steps):
    if buff_size == 0:
        score = 0
        process_seq = ""
        for elmt in seq:
            process_seq += elmt+" "
        for seq_elmt in sequences:
            if seq_elmt in process_seq:
                score += sequences[seq_elmt]
        return score, steps

    else:
        max_score = 0
        best_steps = []
        if vert:
            for i in range(0, len(matrix)):
                if prev_idx != [i, rc]:
                    seq.append(matrix[i][rc])
                    score, step = protocol(matrix, buff_size-1, not vert, seq,
i, [i, rc], sequences, [*steps, (i, rc)])
                    seq.pop()
                    if score > max_score:
                        max_score = score
                        best_steps = step


        else:
            for i in range(0, len(matrix[rc])):
                if prev_idx != [rc, i]:
                    seq.append(matrix[rc][i])
                    score, step = protocol(matrix, buff_size-1, not vert, seq,
i, [rc, i], sequences, [*steps, (rc, i)])
                    seq.pop()
                    if score > max_score:
                        max_score = score
                        best_steps = step

        return max_score, best_steps
```

B. Kode untuk Generate Random

```
def generateRandom(unique_token, tokens, matrix_size, num_seq, max_seq_len):
    tokens = tokens.split()
    print("Membuat matriks...")
    time.sleep(1)
    m = []
```

```
    matrix_size = matrix_size.split()
    matrix_height = int(matrix_size[1])
    matrix_width = int(matrix_size[0])
    for i in range(matrix_height):
        n = []
        for j in range(matrix_width):
            token_num = random.randint(0, unique_token-1)
            n.append(tokens[token_num])
        m.append(n)
    print("Matriks berhasil dibuat!")
    time.sleep(0.5)
    print("Membuat sekuens beserta rewardnya...")
    time.sleep(1)
    sequences = {}
    for i in range(num_seq):
        seq_len = random.randint(2, max_seq_len)
        seq = ""
        for i in range(seq_len):
            seq_num = random.randint(0, unique_token-1)
            seq += tokens[seq_num]+" "
        seq_reward = random.randint(1, 10)
        seq_reward *= 5
        sequences[seq] = seq_reward
    print("Sekuens berhasil dibuat!")
    time.sleep(0.5)

    return m, sequences
```

C. Kode untuk Import file .txt

```
def readFile(filePath):
    lines = []
    if os.path.isfile("test/" + filePath):
        try:
            with open("test/" + filePath) as file:
                lines = [line.rstrip() for line in file]
            print("File ditemukan! Melakukan parsing...")
        except IOError as e:
            print("File tidak bisa dibuka.")
            exit(0)
    else:
        print('{} :File tidak ditemukan.'.format(filePath))
        exit(0)
    return lines

def parseFile(lines):
    buff_size = lines[0]
    matrix_size = lines[1]
    matrix_height = int(matrix_size.split()[1])
    data_matrix = lines[2:2+matrix_height]
    matrix_token = matrix.readMatrix(data_matrix)
    num_seq = int(lines[matrix_height+2])
    sequences = {}
```

```
    for i in range(0, 2*num_seq, 2):
        sequences[lines[matrix_height+3+i]] = int(lines[matrix_height+4+i])

    parsedLines = {
        "buffer size"           : buff_size,
        "matrix size"           : matrix_size,
        "matrix of token"       : matrix_token,
        "number of sequence"    : num_seq,
        "sequences"             : sequences
    }
    print("File berhasil di-parse!")
    return parsedLines
```

D. Kode untuk Menyimpan Solusi pada File .txt

```
def simpan_solusi(score, exe_time, best_steps):
    dt = datetime.datetime.now().strftime("%d-%m-%y")
    parent_dir = "test/save_file/"
    dir = os.path.join(parent_dir, dt)
    try:
        os.mkdir(dir)
        print(f"Directory {dt} telah dibuat. Menyimpan...")
    except FileExistsError:
        print(f"Directory {dt} sudah ada. Menyimpan...")
    time.sleep(0.8)
    file_name = input("Masukkan nama file: ")
    path = os.path.join(dir, file_name)
    if os.path.exists(path):
        print(f"File {file_name} sudah ada. Overwrite otomatis...")
    else:
        print(f"Membuat file {file_name}...")
    time.sleep(0.8)
    f = open(path, "w")
    f.write(f"{score}\n")
    for step in best_steps:
        f.write(f"{step[0]+1} {step[1]+1}\n")
    f.write(f"{exe_time} ms\n")
    f.close()
    print("Selesai menyimpan")
    time.sleep(0.5)
```

E. Kode untuk Menampilkan Matriks, Sekuens, dan Hasil Pencarian

```
def displayMatrix(matrix):
    print("Matriks: ")
    time.sleep(1)
    for i in range(len(matrix)):
        for j in range(len(matrix[i])):
            print(matrix[i][j], end=" ")
        print()
        time.sleep(0.3)
```

```
def displaySequences(sequences):
    print("Sekuens beserta rewardnya: ")
    time.sleep(1)
    for sequence in sequences:
        print(f"{sequence}\n{sequences[sequence]}")
        time.sleep(0.3)

def displayResult(matrix, max_score, exe_time, best_steps):
    print(f"Skor maksimal: {max_score}")
    time.sleep(0.3)
    print("Urutan token:")
    for step in best_steps:
        print(f"{matrix[step[0]][step[1]]}", end=" ")
    print()
    time.sleep(0.3)
    print("Token berada pada koordinat berikut:")
    for step in best_steps:
        print(f"{step[0]} {step[1]}")
    time.sleep(0.3)
    print(f"Waktu eksekusi: {exe_time} ms")
```

F. Kode untuk Menerima Input Nama File atau Atribut untuk Generate Random dan Pemanggilan Fungsi Utama Brute Force

```
def play(opsi):
    if opsi == 1:
        matrix, buff_size, sequences = playFile()
    else:
        matrix, buff_size, sequences = playRandom()
    print("Melakukan pencarian...")
    time.sleep(1)
    start_time = time.time()
    max_score, best_steps = protocol(matrix, buff_size, False, [], 0, [-1,
-1], sequences, [])
    end_time = time.time()
    print("Pencarian selesai!")
    time.sleep(0.5)
    exe_time = (end_time-start_time)*1000 # dalam ms
    displayResult(matrix, max_score, exe_time, best_steps)
    return max_score, exe_time, best_steps

def playFile():
    print("Anda memilih import dari file .txt")
    time.sleep(1)
    filePath = input("Masukkan file path txt (pastikan format benar): ")
    time.sleep(0.3)
    lines = readFile(filePath)
    time.sleep(0.5)
    parsed = parseFile(lines)
    time.sleep(0.7)
    matrix_token = parsed["matrix of token"]
```

```
    buff_size = int(parsed["buffer size"])
    sequences = parsed["sequences"]
    time.sleep(0.5)
    displayMatrix(matrix_token)
    time.sleep(0.7)
    displaySequences(sequences)
    time.sleep(0.7)

    return matrix_token, buff_size, sequences

def playRandom():
    print("Anda memilih generate random")
    time.sleep(0.3)
    unique_token = int(input("Banyaknya token unik: "))
    tokens = input("Token-token unik: ")
    buff_size = int(input("Ukuran buffer: "))
    matrix_size = input("Ukuran matriks: ")
    num_seq = int(input("Banyaknya sekuens: "))
    max_seq_len = int(input("Ukuran maksimal sekuens: "))
    matrix, sequences = generateRandom(unique_token, tokens, matrix_size,
num_seq, max_seq_len)
    time.sleep(0.5)
    displayMatrix(matrix)
    time.sleep(0.7)
    displaySequences(sequences)
    time.sleep(0.7)
    return matrix, buff_size, sequences
```

G. Inisiasi dan Terminasi Program

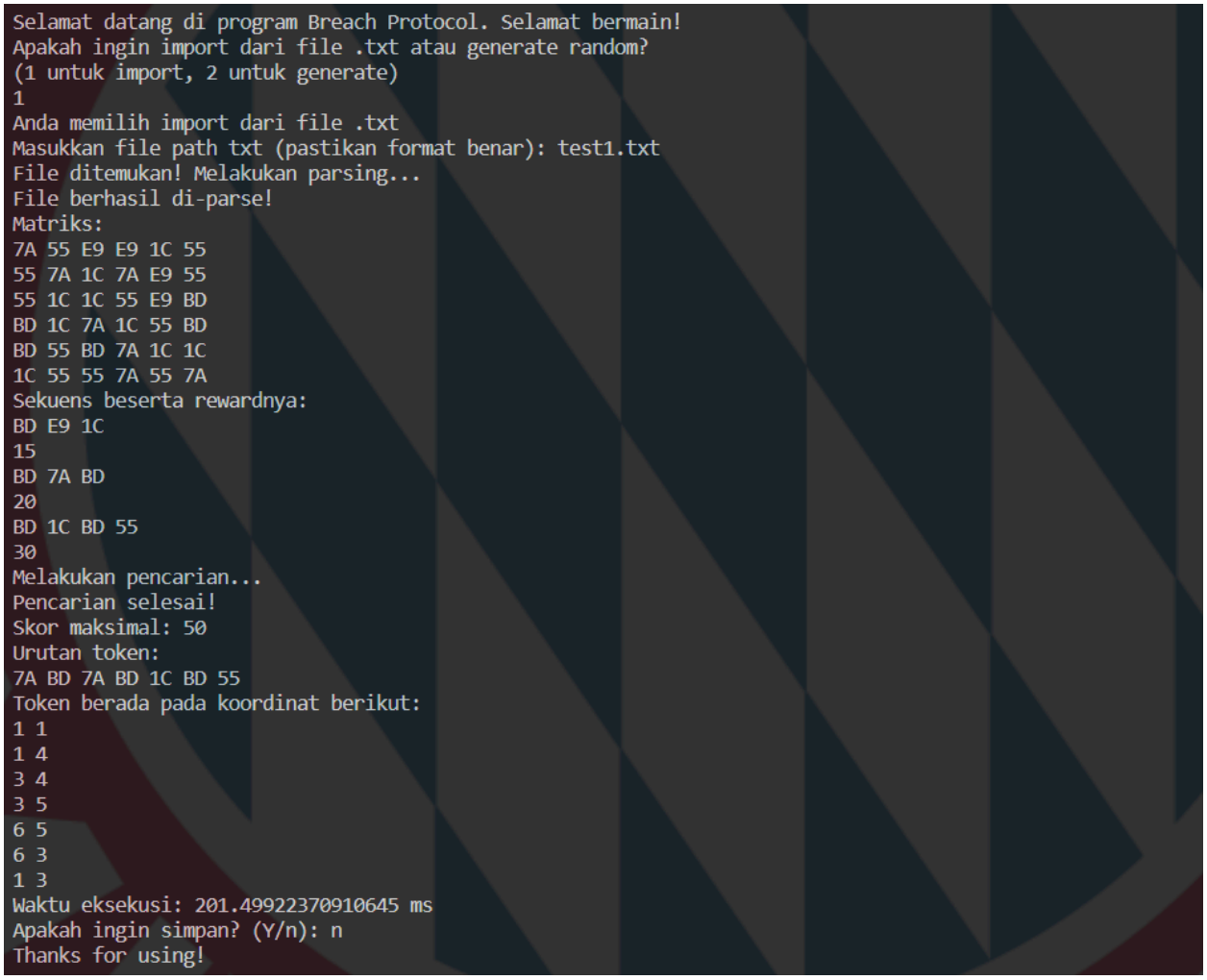
```
def opening():
    print("""
 ________   ________   _________   _______    _______   __    __         ________
_________  _______  _________  _______   _______   ___  |  /  |      /        \ /
\  /      \ /         |/       \  /       \  /       \ /  |
$$$$$$$  |$$$$$$$  |$$$$$$$$/ /$$$$$$  |/$$$$$$  |$$ |  $$ |      $$$$$$$
|$$$$$$$  |/$$$$$$  |$$$$$$$$//$$$$$$  |/$$$$$$  |/$$$$$$  |$$ |
$$ |__$$ |$$ |__$$ |$$ |___     $$ |__$$ |$$ |  $$/ $$ |__$$ |      $$ |__$$ |$$
|__$$ |$$ |  $$ |   $$ |  $$ |  $$ |$$ |  $$/ $$ |  $$ |$$ |
$$    $$< $$    $$< $$    |   $$    $$ |$$ |      $$    $$ |      $$    $$/ $$
$$< $$ |  $$ |   $$ |  $$ |  $$ |$$ |      $$ |  $$ |$$ |
$$$$$$$  |$$$$$$$  |$$$$$/    $$$$$$$$ |$$ |    __ $$$$$$$$ |      $$$$$$$/
$$$$$$$  |$$ |  $$ |   $$ |  $$ |  $$ |$$ |    __ $$ |  $$ |$$ |
$$ |__$$ |$$ |  $$ |$$ |_____ $$ |  $$ |$$ \__/  |$$ |  $$ |      $$ |      $$
|  $$ |$$ \__$$ |   $$ |  $$ \__$$ |$$ \__/  |$$ \__$$ |$$ |_____
$$    $$/ $$ |  $$ |$$       |$$ |  $$ |$$    $$/ $$ |  $$ |      $$ |      $$
|  $$ |$$    $$/    $$ |  $$    $$/ $$    $$/ $$    $$/ $$       |
$$$$$$$/  $$/   $$/ $$$$$$$$/ $$/   $$/  $$$$$$/  $$/   $$/       $$/
$$/   $$/  $$$$$$/     $$/    $$$$$$/   $$$$$$/   $$$$$$/  $$$$$$$$/
          """)
    time.sleep(0.7)
```

```
    print("Selamat datang di program Breach Protocol. Selamat bermain!")

def initiate():
    print("""Apakah ingin import dari file .txt atau generate random?
(1 untuk import, 2 untuk generate)""")
    opsi = int(input())
    while opsi != 1 and opsi != 2:
        print("Masukan salah! Ulangi")
        opsi = int(input())
    return opsi

def outro(score, exe_time, best_steps):
    simpan = input("Apakah ingin simpan? (Y/n): ")
    while simpan.lower() != "y" and simpan.lower() != "n":
        simpan = input("Masukan tidak benar! (Y/n): ")
    if simpan.lower() == "y":
        simpan_solusi(score, exe_time, best_steps)

def exitGame():
    print("Thanks for using!")
    print("""
 ________  __    __  ________  __    _  ___   _    __   __  ________  __   __
___   ____
|       ||  | |  ||   _   ||  |  | ||   | | |  |  | |  ||       ||  | |  |  |
| |    |
|_     _||  |_|  ||  |_|  ||   |_| ||   |_| |  |  |_|  ||   _   ||  | |  |
|___| |_    |
  |   |  |       ||       ||       ||      _|  |       ||  | |  ||  |_|  |
___    |   |
  |   |  |       ||       ||  _    ||     |_   |_     _||  |_|  ||       |  |
|   |   |
  |   |  |   _   ||   _   || | |  ||    _  |    |   |  |       ||       |
|___|  _|   |
  |___|  |__| |__||__| |__||_|  |__||___| |_|    |___|  |_______||_______|
|____|
          """)
    time.sleep(0.5)
```

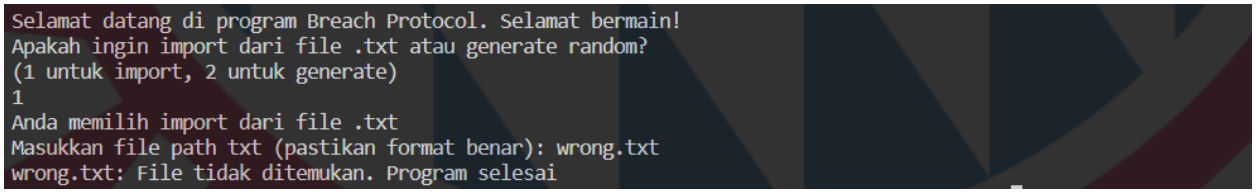
BAB IV
HASIL UJI COBA

A. Masukan dari File .txt

1. Masukan nama file benar



2. Masukan nama file salah

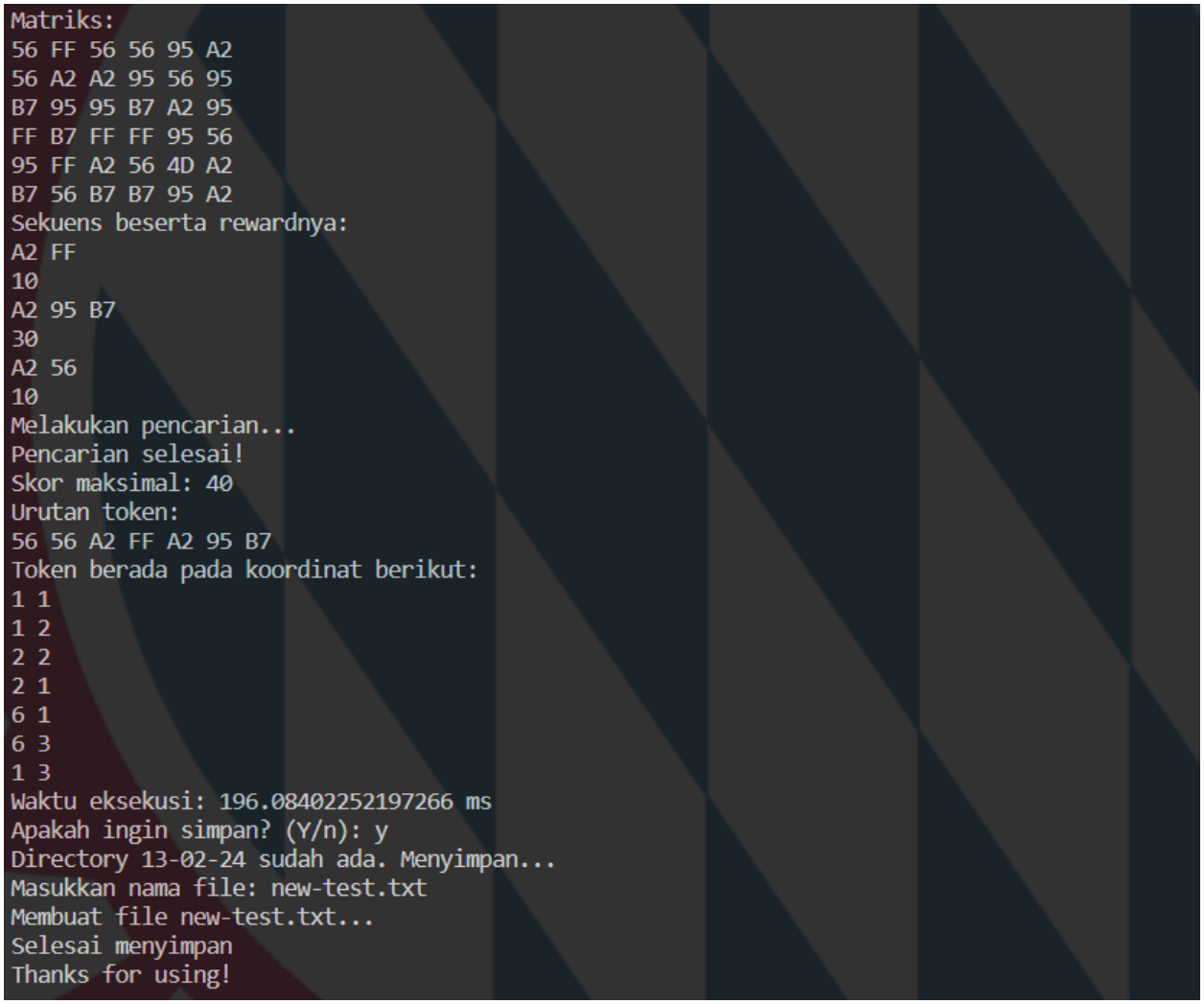


B. Masukan dari Generate Random

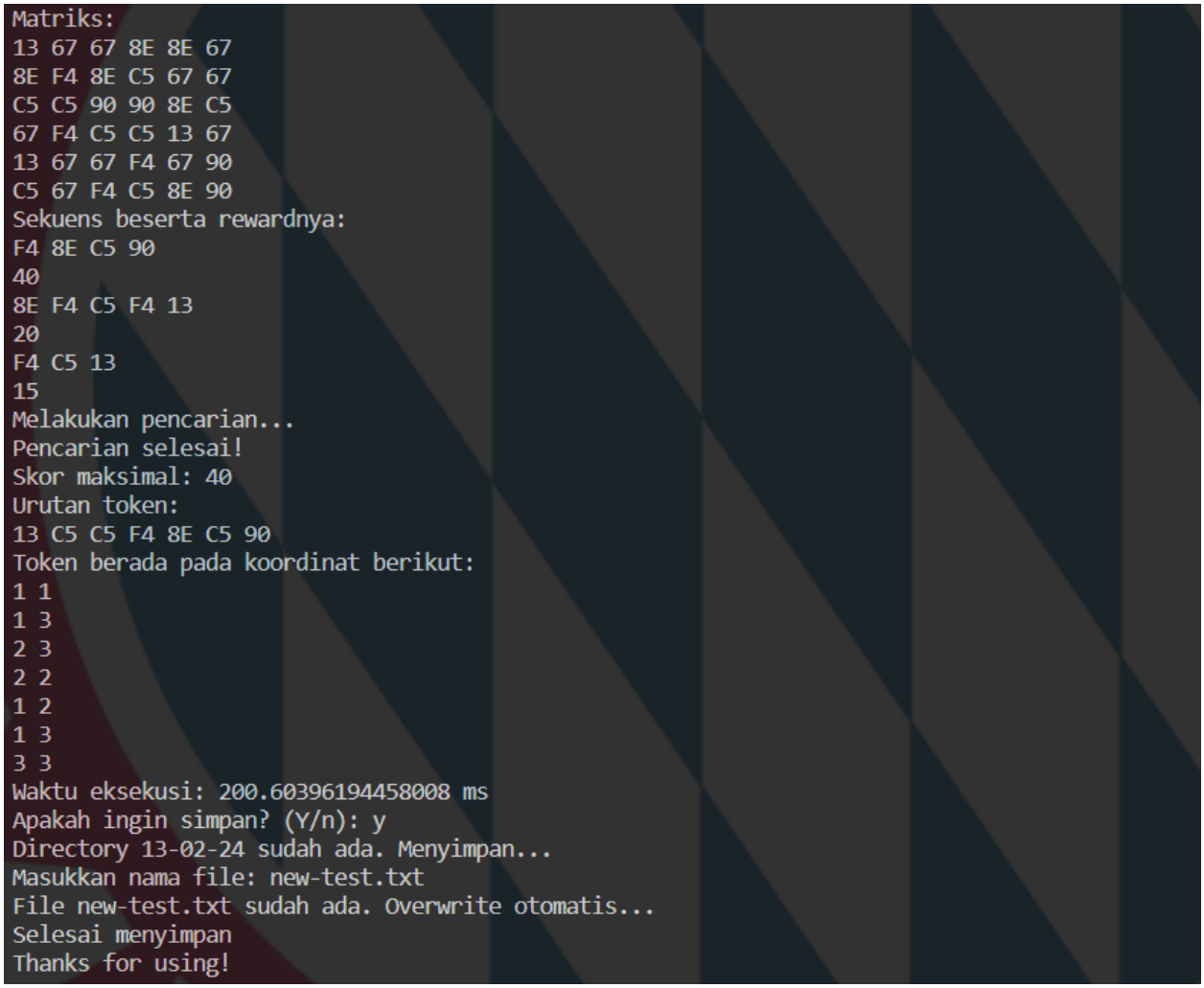
```
Selamat datang di program Breach Protocol. Selamat bermain!
Apakah ingin import dari file .txt atau generate random?
(1 untuk import, 2 untuk generate)
2
Anda memilih generate random
Banyaknya token unik: 5
Token-token unik: B3 68 C0 9A 1F
Ukuran buffer: 7
Ukuran matriks: 6 6
Banyaknya sekuens: 3
Ukuran maksimal sekuens: 4
Membuat matriks...
Matriks berhasil dibuat!
Membuat sekuens beserta rewardnya...
Sekuens berhasil dibuat!
Matriks:
B3 C0 9A C0 68 9A
68 C0 68 68 9A C0
1F 1F 1F 1F 1F 68
C0 9A 1F C0 C0 9A
1F 9A 1F 68 1F 9A
68 1F 9A 9A C0 1F
Sekuens beserta rewardnya:
B3 1F 9A
50
9A 9A B3 1F
35
68 C0
50
Melakukan pencarian...
Pencarian selesai!
Skor maksimal: 135
Urutan token:
68 C0 9A 9A B3 1F 9A
Token berada pada koordinat berikut:
5 1
5 4
6 4
6 1
1 1
1 5
2 5
Waktu eksekusi: 199.60999488830566 ms
```

C. Hasil Permainan Disimpan

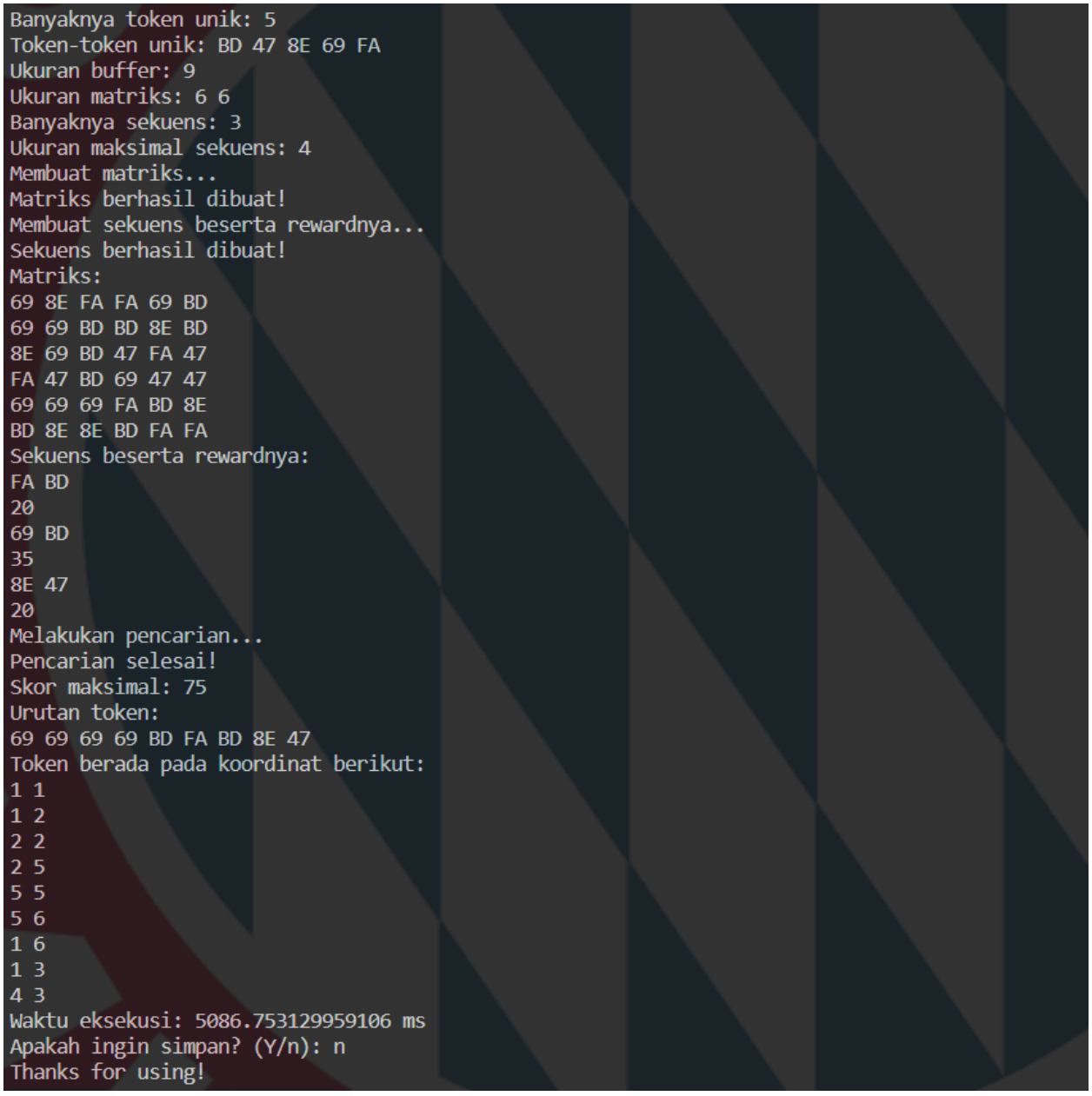
1. Hasil disimpan pada file baru



2. Hasil disimpan pada file yang sudah ada (overwrite)



D. Ukuran Buffer Besar



LAMPIRAN

- Checklist Kelengkapan Tugas

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		√

- Tautan Repository Github Implementasi Program : https://github.com/ibrahim-rasyid/tucil-1-stima