IF2211 Strategi Algoritma

Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force

Laporan Tugas Kecil 1

Disusun untuk memenuhi tugas mata kuliah Strategi Algoritma
pada Semester 2 (dua) Tahun Akademik 2023/2024

Disusun oleh:

Auralea Alvinia Syaikha (13522148)

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

BANDUNG
2024

DAFTAR ISI

BAB 1

DESKRIPSI MASALAH DAN ALGORITMA

1.1 Cyberpunk 2077 Breach Protocol

Breach Protocol dalam Cyberpunk 2077 adalah permainan hacking yang melibatkan pemecahan masalah aritmatika dengan tujuan menyusun kode-kode dari matriks kode ke dalam urutan yang ditentukan. Game ini menarik minat banyak orang karena dapat membantu meningkatkan keterampilan hacking seseorang dan melatih kemampuan berpikir secara strategis dan efisien. Saat memulai permainan, pemain harus memilih kode dari baris pertama matriks kode, lalu mencari kode yang sesuai dari kolom yang tepat, dan seterusnya, hingga semua baris kode yang dibutuhkan terisi atau waktu habis. Permainan berakhir ketika semua baris kode yang diperlukan berhasil dimasukkan oleh pemain atau ketika pemain gagal melakukannya.

Gambar 1 Permainan Breach Protocol
(Sumber: https://cyberpunk.fandom.com/wiki/Quickhacking)

1.2 Algoritma *Brute Force*

Algoritma brute force merupakan metode penyelesaian masalah algoritmik yang sederhana dan langsung. Pendekatan ini didasarkan pada pernyataan masalah (problem statement) serta konsep yang terlibat dalam masalah yang sedang dihadapi. Salah satu ciri khas algoritma brute force adalah konsep penyelesaian yang mudah dipahami, jelas, dan intuitif.

Dalam penerapan algoritma brute force, langkah pertama adalah mencari semua kemungkinan solusi yang mungkin. Ini seringkali melibatkan enumerasi atau penghitungan semua kemungkinan kombinasi yang mungkin. Setelah semua solusi mungkin teridentifikasi, langkah berikutnya adalah menguji masing-masing solusi untuk memastikan bahwa mereka memenuhi kriteria atau syarat yang ditentukan. Solusi yang tidak memenuhi syarat akan dibuang, dan solusi terbaik akan dipilih sebagai jawaban akhir.

Salah satu keuntungan dari algoritma brute force adalah bahwa ia dapat menemukan solusi jika solusi tersebut ada. Namun, kelemahannya adalah bahwa pendekatan ini seringkali tidak efisien dan membutuhkan waktu yang cukup lama, terutama ketika jumlah solusi yang mungkin sangat besar. Kompleksitas waktu dari algoritma brute force biasanya lebih buruk dari polinomial, sering kali dinyatakan dalam notasi Big O (O(n)), di mana n adalah jumlah elemen yang terlibat dalam masalah yang sedang dipecahkan. Oleh karena itu, meskipun dapat memberikan solusi, algoritma brute force tidak selalu menjadi pilihan terbaik ketika efisiensi waktu menjadi pertimbangan utama.

1.3 Algoritma Penyelesaian *Cyberpunk 2077 Breach Protocol* dengan Pendekatan *Brute Force*

Dalam menemukan solusi untuk permasalahan Cyberpunk 2077 Breach Protocol secara algoritmik, penulis menerapkan pendekatan brute force. Cara penyelesaian permasalahan dalam algoritma tersebut dapat diuraikan secara deskriptif sebagai berikut:

1. **Inisialisasi variabel**: Pada awalnya, dilakukan inisialisasi variabel-variabel seperti stack, taken_tokens, dan start_time. Variabel stack digunakan untuk menyimpan token-token yang telah dipilih untuk membentuk sebuah urutan, taken_tokens untuk menyimpan token-token yang telah dipilih, dan start_time untuk merekam waktu mulai pencarian solusi.
2. **Pemanggilan fungsi 'searchSequence'**: Dilakukan panggilan fungsi searchSequence untuk melakukan pencarian urutan token yang memenuhi syarat dengan menggunakan pendekatan rekursif. Fungsi ini dipanggil dengan parameter-parameter yang diperlukan, yaitu matriks, stack yang masih kosong, koordinat awal row dan col yang ditetapkan pada posisi (0, 0), ukuran buffer, daftar solusi (solution) yang masih kosong, dan status check_horizontal yang awalnya True (menandakan pencarian pertama kali dilakukan secara horizontal).
3. **Pengecekan ukuran buffer**: Fungsi searchSequence melakukan pengecekan. Jika ukuran buffer sudah mencapai 1, artinya sebuah urutan yang memenuhi kriteria telah ditemukan dan urutan tersebut ditambahkan ke dalam daftar solusi (solution).
4. **Pencarian token secara horizontal**: Jika check_horizontal adalah True, berarti pencarian dilakukan secara horizontal. Fungsi searchSequence akan melakukan iterasi untuk setiap kolom

pada baris yang sama. Untuk setiap kolom, jika token belum dilewati (passedToken), token tersebut ditambahkan ke dalam stack, dan fungsi searchSequence dipanggil kembali dengan buffer dikurangi 1 dan check_horizontal diubah menjadi False (menandakan akan dilakukan pencarian vertikal selanjutnya).

5. **Pencarian token secara vertikal**: Jika check_horizontal adalah False, berarti pencarian dilakukan secara vertikal. Fungsi searchSequence akan melakukan iterasi untuk setiap baris pada kolom yang sama. Untuk setiap baris, jika token belum dilewati, token tersebut ditambahkan ke dalam stack, dan fungsi searchSequence dipanggil kembali dengan buffer dikurangi 1 dan check_horizontal diubah menjadi True (menandakan akan dilakukan pencarian horizontal selanjutnya).
6. **Rekursi**: Proses di atas akan terus berulang secara rekursif hingga ukuran buffer mencapai 1. Saat ukuran buffer mencapai 1, proses rekursi akan berhenti, dan urutan token yang memenuhi kriteria akan ditambahkan ke dalam daftar solusi (solution).
7. **Kembali ke panggilan rekursi sebelumnya**: Setelah menemukan solusi untuk buffer saat ini, fungsi searchSequence akan kembali ke level rekursi sebelumnya dan mencoba token lain untuk melanjutkan pencarian.
8. **Berhenti**: Setelah semua kemungkinan telah dijelajahi, proses pencarian solusi akan berhenti, dan kontrol akan dikembalikan ke fungsi pemanggil searchSequence.

BAB 2

IMPLEMENTASI ALGORITMA DALAM PYTHON

Dalam pengembangan program ini, penulis menggunakan bahasa pemrograman Python. Adapun struktur dari program ini terbagi menjadi 7 file, yaitu main.py, recursive.py, readFile.py, readCLI.py, matrix.py, sequence.py, tokenin.py.

2.1 File main.py

File ini adalah driver utama dari program, yang mana tidak mengandung fungsi tertentu di dalamnya. Isi dari file ini hanya berupa menu utama program dan deklarasi variabel yang akan digunakan.

Nama Variabel	Deskripsi
`user_input`	Variabel ini menyimpan input yang diberikan oleh pengguna, yaitu pilihan menu utama (1 atau 2) untuk membaca input dari file atau dari CLI.
`file_name`	Variabel ini menyimpan nama file yang diinputkan oleh pengguna saat memilih untuk membaca input dari file.
`file_path`	Variabel ini menyimpan path lengkap menuju file yang akan dibaca, dihasilkan dengan menggabungkan nama file dengan direktori test.
`buffer_size`	Variabel ini menyimpan ukuran buffer yang dibaca dari file input, menunjukkan berapa banyak token yang perlu dipilih dalam pencarian solusi.
`matrix`	Variabel ini menyimpan matriks yang akan digunakan untuk mencari solusi.
`sequences`	Variabel ini menyimpan daftar urutan token yang harus dipilih untuk mendapatkan solusi yang optimal.
`number_of_unique_tokens`	Variabel ini menyimpan jumlah token unik yang diinputkan oleh pengguna saat memilih untuk membaca input dari CLI.
`unique_tokens`	Variabel ini menyimpan daftar token unik yang diinputkan oleh pengguna saat memilih untuk membaca input dari CLI.

`matrix_size`	Variabel ini menyimpan ukuran matriks yang diinputkan oleh pengguna saat memilih untuk membaca input dari CLI.
`buffer_size`	Variabel ini menyimpan ukuran buffer yang diinputkan oleh pengguna saat memilih untuk membaca input dari CLI.
`num_sequences`	Variabel ini menyimpan jumlah urutan yang diinputkan oleh pengguna saat memilih untuk membaca input dari CLI.
`max_sequence`	Variabel ini menyimpan maksimum urutan yang diinputkan oleh pengguna saat memilih untuk membaca input dari CLI.
`stack`	Variabel ini digunakan untuk menumpuk token saat mencari solusi.
`taken_tokens`	Variabel ini digunakan untuk menyimpan token yang telah dipilih dalam pencarian solusi.
`start_time`	Variabel ini menyimpan waktu awal saat program mulai berjalan, digunakan untuk mengukur waktu eksekusi.
`end_time`	Variabel ini menyimpan waktu saat program selesai dieksekusi, digunakan untuk mengukur waktu eksekusi.
`result_string`	Variabel ini digunakan untuk menyimpan hasil akhir dari solusi yang ditemukan dan waktu eksekusi, yang kemudian dapat disimpan ke dalam file jika diinginkan pengguna.

2.2 File recursive.py

File ini berisi sejumlah fungsi-fungsi yang diperlukan untuk menjalankan algoritma pencarian solusi optimal dalam konteks program yang lebih besar. Fungsi-fungsi ini secara kolektif bertanggung jawab untuk melakukan operasi-operasi kunci dalam pencarian solusi, seperti membandingkan urutan token, menghitung reward, mencari urutan optimal, dan melakukan pencarian rekursif dalam matriks.

Nama Fungsi	Deskripsi
`isSequenceEqual(tokens, sequence)`	Membandingkan dua array of tokens dan mengembalikan nilai boolean. Fungsi ini akan mengembalikan True jika sequence yang diinput terdapat dalam tokens, dan False jika tidak.
`makeListToken(tokens)`	Mengonversi array token menjadi array string. Fungsi ini digunakan untuk membuat list dari token dalam bentuk string.
`calculateReward(sequences, tokens)`	Menghitung skor yang mungkin dari urutan token (sequences) dan sebuah solusi (tokens). Fungsi ini akan mengembalikan jumlah reward yang diperoleh dari sequences yang cocok dengan tokens.
`findOptimalSequence(solutions, sequences)`	Algoritma untuk mencari urutan optimal dari solusi. Fungsi ini melakukan perulangan pada setiap solusi yang diberikan, menghitung reward dari setiap solusi menggunakan fungsi calculateReward, dan mengembalikan urutan dengan reward tertinggi.
`passedToken(row, col, stack)`	Mendeteksi apakah sebuah token telah dilewati atau tidak dalam pencarian solusi. Fungsi ini mengembalikan nilai True jika token dengan koordinat (row, col) telah dilewati dalam stack, dan False jika tidak.
`searchSequence(matrix : Matrix, stack, row, col, buffer, solution, check_horizontal)`	Mencari urutan token yang memenuhi kriteria tertentu. Fungsi ini menggunakan pendekatan rekursif untuk menelusuri matriks (matrix) dan menambahkan token ke stack hingga buffer mencapai 1.

2.3 File readCLI.py

File ini bertanggung jawab untuk menghasilkan matriks dan sekeuns acak yang digunakan dalam program pencarian solusi optimal.

Nama Fungsi	Deskripsi
`generateMatrix(tokens, matrix_size)`	Menghasilkan matriks dengan ukuran tertentu yang diisi dengan token-token acak dari daftar yang diberikan. Ini berguna untuk membangkitkan matriks yang akan digunakan dalam pencarian solusi optimal. Fungsi ini mengembalikan objek Matrix yang merepresentasikan matriks yang dibuat.
`generateSequences(tokens, buffer_size, num_sequences, max_sequence)`	Menghasilkan daftar urutan acak dengan panjang dan jumlah tertentu, bersama dengan nilai hadiah yang terkait. Urutan acak ini digunakan sebagai masukan untuk algoritma pencarian solusi optimal. Fungsi ini mengembalikan daftar objek Sequence yang mewakili urutan dan nilai hadiahnya.
`isValidToken(token)`	Memeriksa apakah token yang diberikan adalah alfanumerik dan memiliki panjang maksimum dua karakter. Hal ini penting untuk memastikan bahwa token yang digunakan sesuai dengan aturan yang diberlakukan dalam program pencarian solusi optimal.

2.4 File readFile.py

File ini bertanggung jawab untuk membaca file masukan dan mencetak informasi matriks serta urutan.

Nama Fungsi	Deskripsi
`parse_file(file_name)`	Membaca isi file dengan nama tertentu dan mengurai informasi matriks dan urutan dari file tersebut. Fungsi ini mengembalikan buffer size, objek Matrix yang mewakili matriks, dan daftar objek Sequence yang mewakili urutan dan nilai hadiahnya.
`print_matrix(matrix)`	Mencetak matriks ke layar dengan format yang sesuai. Ini berguna untuk menampilkan matriks dari file masukan ke pengguna agar dapat dilihat

	dengan jelas.
`print_sequences(sequences)`	Mencetak daftar urutan beserta nilai hadiahnya ke layar dengan format yang sesuai. Ini berguna untuk menampilkan informasi urutan dan nilai hadiahnya dari file masukan ke pengguna agar dapat dilihat dengan jelas.

2.5 File matrix.py

File ini berisi definisi dari sebuah kelas bernama Matrix. Berikut adalah penjelasan tentang kelas tersebut beserta kegunaannya:

`` class Matrix: def __init__(self, width, height, data): self.width = width self.height = height self.data = data ```	Kelas ini merepresentasikan sebuah matriks. Matriks memiliki properti lebar (width), tinggi (height), dan data. Properti width dan height digunakan untuk menyimpan ukuran matriks, sedangkan properti data digunakan untuk menyimpan isi dari matriks tersebut dalam bentuk array dua dimensi. Metode inisialisasi yang digunakan untuk membuat objek Matrix baru.

2.6 File sequence.py

File ini berisi definisi dari sebuah kelas bernama Sequence. Berikut adalah penjelasan tentang kelas tersebut beserta kegunaannya:

`` class Sequence: def __init__(self, sequence, reward): self.sequence = sequence self.reward = reward ```	Kelas ini merepresentasikan sebuah urutan (sequence) yang terdiri dari token-token tertentu beserta dengan imbalan (reward) yang terkait. Ketika objek Sequence dibuat, metode ini mengatur nilai sequence dan reward sesuai dengan argumen yang diberikan. Sequence merepresentasikan urutan token-token, sedangkan reward adalah bobot yang terkait dengan urutan tersebut.

2.7 File tokenin.py

File ini berisi sebuah kelas yang digunakan untuk merepresentasikan token tertentu dalam matriks, beserta dengan posisi (koordinat)nya dalam matriks tersebut. Berikut penjelasan lebih detailnya:

`` class TokenInput: def __init__(self, token, x, y): self.token = token self.x = x self.y = y ```	Kelas ini digunakan untuk merepresentasikan token tertentu dalam matriks, beserta dengan koordinat posisinya. Token merepresentasikan nilai token tersebut, sedangkan x dan y merepresentasikan koordinat posisi token dalam matriks.

BAB 3

SOURCE CODE PROGRAM

3.1 Source CodeProgram

3.1.1 main.py

```
import os
import time
from readCLI import generateMatrix, generateSequences, isValidToken
from recursive import searchSequence, findOptimalSequence, calculateReward
from readFile import parse_file, print_matrix, print_sequences


# MENU
def get_menu_option():
    menu_options = ('1', '2')

    while True:
        print()
        print()
        print("                                Welcome to                                ")
        print()
        print("________                        ______     ________         _____          ______   ")
        print("___  __ )________________ _________  /_    ___  __ \_____________  /__________________  /   ")
        print("__  __  |_  ___/  _ \  __ `/  ___/_  __ \    __  /_/ /_  ___/  __ \  __/  __ \  ___/  __ \_  /   ")
        print("_  /_/ /_  /   /  __/ /_/ // /__ _  / / /    _  ____/_  /   / /_/ / /_/ /_/ / /__ / /_/ /  /    ")
        print("/_____/ /_/    \___/\__,_/ \___/ /_/ /_/     /_/     /_/    \____/\__/ \____/\___/ \____//_/     ")
        print()
        print()
        print()
        print("╔═════════════════ MENU ═════════════╗")
        print("║                                    ║")
        print("║  1. Read from file                 ║")
        print("║  2. Input from CLI                 ║")
        print("║                                    ║")
        print("╚════════════════════════════════════╝")
        print()
        user_input = input(">> Enter your option (1 / 2): ")



        if user_input in menu_options:
            return user_input
        else:
            print()
            print("OPTION NOT AVAILABLE! Please enter either 1 or 2.")
            time.sleep(1)

user_input = get_menu_option()
```

```python
# READ FILE INPUT
if user_input == '1':
    file_name = input("\n>> Please input your file name with .txt: ")
    file_path = os.path.join("..", "test", file_name)
    buffer_size, matrix, sequences = parse_file(file_path)

    print()
    print("\n Reading File...")
    time.sleep(2)
    print()
    print("----------------------------------------")
    print("                FILE INFO               ")
    print("----------------------------------------")
    print("\nBuffer Size:", buffer_size)
    print("Matrix Width:", matrix.width, ", Height:", matrix.height)
    print("\n--- MATRIX ---")
    print_matrix(matrix)

    print("\nNumber of Sequences:", len(sequences))
    print("\n--- SEQUENCES ---")
    print_sequences(sequences)
    print()
    print("\n Generating optimal solution...")


# READ CLI INPUT
elif user_input == '2':
    number_of_unique_token = int(input("\nInsert number of unique tokens: "))
    unique_tokens = input("Insert tokens: ").split()

    # Validate tokens
    for token in unique_tokens:
        if not isValidToken(token):
            print(f"Error: Invalid token '{token}'. Token must be alphanumeric and have at most 2 characters.")

    matrix_size = tuple(map(int, input("Insert matrix size (row)(column): ").split()))

    if len(unique_tokens) != number_of_unique_token:
        print("Error: Number of tokens doesn't match.")

    matrix = generateMatrix(unique_tokens, matrix_size)

    print("\n--- GENERATED MATRIX ---")
    for row in matrix.data:
        print(' '.join(row))
```

```
    buffer_size = int(input("\nInsert buffer size: "))
    num_sequences = int(input("Insert number of sequences: "))
    max_sequence = int(input("Insert maximum sequences: "))

    sequences = generateSequences(unique_tokens, buffer_size, num_sequences, max_sequence)

    print("\n--- GENERATED SEQUENCES ---")
    for i, seq in enumerate(sequences, start=1):
        print(f"Sequence {i}: {' '.join(seq.sequence)}")
        print(f"Reward sequence {i}: {seq.reward}")


#EXECUTION
print()
print()
print("----------------------------------------")
print("              YOUR RESULT               ")
print("----------------------------------------")
stack = []
taken_tokens = []
start_time = time.time()
searchSequence(matrix, stack, 0, 0, buffer_size+1, taken_tokens, True)
max_reward, optimal_solution = findOptimalSequence(taken_tokens, sequences)
result_string = ""      # ??
end_time = time.time()

if max_reward != 0:
    print(f"\nMaximum reward: {max_reward}")
    result_string += str(max_reward) + "\n"
    check = True
    print("Optimal sequence:", end=" ")
    for token in optimal_solution:
        print(token.token, end=" ")
        if not check:
            result_string += " " + token.token
        else:
            result_string += token.token
            check = False

    print("")

    print("Path coordinates:")
    for token in optimal_solution:
        print(str(token.x+1) + ", " + str(token.y+1))
        result_string += '\n' + str(token.x+1) + ", " + str(token.y+1)
```

```
    execution_time = float(end_time - start_time)*1000
    print("\nExecution time:", execution_time, "ms")
    result_string += f"\nExecution time: {execution_time} ms\n"

else:
    print("\nNO OPTIMAL SOLUTION")
    result_string += "NO OPTIMAL SOLUTION\n"


# SAVING SOLUTION
save_solution = input("\n>> Want to save the solution? (Y/N): ")
save_solution = save_solution.upper()


if save_solution == 'Y':
    save_file = input("Enter file name to save: ")
    save_path = os.path.join("..", "solution", save_file)
    save = open(save_path, 'w')

    save.write(result_string)
    save.close()

    print(f"\nSolution saved to '{save_file}'.")
    print("\nThank you for using the program!")

elif save_solution == 'N':
    print("\nThank you for using the program!")
else:
    print("\nOption is invalid. Please enter either Y or N!")
```

3.1.2 recursive.py

```python
from matrix import Matrix
from sequence import Sequence
from tokenin import TokenInput


def isSequenceEqual(tokens, sequence):
    # Check if a sequence is equal to a list of tokens
    if len(sequence) > len(tokens):
        return False
    else:
        for i in range(len(tokens) - len(sequence) + 1):
            if tokens[i:i+len(sequence)] == sequence:
                return True
        return False


def makeListToken(tokens):
    # Create a list of tokens from a list of TokenInput
    list_token = []
    for token in tokens:
        list_token.append(token.token)
    return list_token



def calculateReward(sequences, tokens):
    # Calculate the reward of a sequence
    reward = 0

    for seq in sequences:
        if isSequenceEqual(makeListToken(tokens), seq.sequence):
            reward += seq.reward
    return reward


def findOptimalSequence(solutions, sequences):
    # Find the optimal sequence
    max_reward = 0
    optimal_sequence = []
    for solution in solutions:
        reward = calculateReward(sequences, solution)
        if reward > max_reward:
            max_reward = reward
            optimal_sequence = solution
    return max_reward, optimal_sequence
```

```python
def passedToken(row, col, stack):
    # Detect whether a token has been passed or not
    if stack == []:
        return False
    else:
        for token in stack:
            if (token.x == col) and (token.y == row):
                return True


def searchSequence(matrix : Matrix, stack, row, col, buffer, solution, check_horizontal):
    # Recursive function to search for a sequence
    if buffer == 1:
        solution.append(list(stack))
    else:
        if check_horizontal:
            # Search for horizontal sequence
            for i in range (matrix.width):
                if not passedToken(row, i, stack):
                    stack.append(TokenInput(matrix.data[row][i], i, row))
                    searchSequence(matrix, stack, row, i, buffer-1, solution, False)
                    stack.pop()
        else:
            # Search for vertical sequence
            for i in range (matrix.height):
                if not passedToken(i, col, stack):
                    stack.append(TokenInput(matrix.data[i][col], col, i))
                    searchSequence(matrix, stack, i, col, buffer-1, solution, True)
                    stack.pop()
```

3.1.3 readCLI.py

```python
import random
import string
from matrix import Matrix
from sequence import Sequence


def generateMatrix(tokens, matrix_size):
    matrix = [['_' for _ in range(matrix_size[1])] for _ in range(matrix_size[0])]
    random_tokens = random.choices(tokens, k=matrix_size[0]*matrix_size[1])
    for i in range(len(random_tokens)):
        row = i // matrix_size[1]
        col = i % matrix_size[1]
        matrix[row][col] = random_tokens[i]
    return Matrix(matrix_size[1], matrix_size[0], matrix)

def generateSequences(tokens, buffer_size, num_sequences, max_sequence):
    sequences = []
    for _ in range(num_sequences):
        sequence = random.choices(tokens, k=random.randint(2, max_sequence))
        sequences.append(Sequence(sequence, random.randint(-50, 100)))
        # Remove buffer elements
        for _ in range(buffer_size):
            if len(sequence) > buffer_size:
                sequence.pop(0)
    return sequences

def isValidToken(token):
    # Check if token is alphanumeric and has at most 2 characters
    return token.isalnum() and len(token) <= 2
```

3.1.4 readFile.py

```python
import time
from matrix import Matrix
from sequence import Sequence


def parse_file(file_name):
    with open(file_name, 'r') as file:
        buffer_size = int(file.readline().strip())
        matrix_width, matrix_height = map(int, file.readline().strip().split())

        matrix_data = []
        for _ in range(matrix_height):
            row = file.readline().strip().split()
            matrix_data.append(row)

        matrix = Matrix(matrix_width, matrix_height, matrix_data)

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

        sequences = []
        for _ in range(num_sequences):
            sequence = file.readline().strip().split()
            reward = int(file.readline().strip())
            sequences.append(Sequence(sequence, reward))

    return buffer_size, matrix, sequences

def print_matrix(matrix):
    for row in matrix.data:
        print(" ".join(row))

def print_sequences(sequences):
    for i, seq in enumerate(sequences, 1):
        print(f"Sequence {i}: {seq.sequence}, Reward: {seq.reward}")
```

3.1.5 matrix.py

```
class Matrix:
    def __init__(self, width, height, data):
        self.width = width
        self.height = height
        self.data = data
```

3.1.6 sequence.py

```
class Sequence:
    def __init__(self, sequence, reward):
        self.sequence = sequence
        self.reward = reward
```

3.1.7 tokenin.py

```
class TokenInput:
    def __init__(self, token, x, y):
        self.token = token
        self.x = x
        self.y = y
```

3.1 Repository Program

Repository program dapat diakses melalui tautan GitHub berikut ini:
[auraleaas/Tucil1_13522148: Tugas Kecil mata kuliah Strategi Algoritma - Cyberpunk 2077 Breach Protocol Solver (github.com)]

BAB 4

TESTING PROGRAM

4.1 Input dari File (user)

INPUT	OUTPUT
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 *Note In file: (test.txt) (sol1.txt)	``` ---------- FILE INFO ---------- Buffer Size: 7 Matrix Width: 6 , Height: 6 --- MATRIX --- 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 Number of Sequences: 3 --- SEQUENCES --- Sequence 1: ['BD', 'E9', '1C'], Reward: 15 Sequence 2: ['BD', '7A', 'BD'], Reward: 20 Sequence 3: ['BD', '1C', 'BD', '55'], Reward: 30 Generating optimal solution... ---------- YOUR RESULT ---------- Maximum reward: 50 Optimal sequence: 7A BD 7A BD 1C BD 55 Path coordinates: 1, 1 1, 4 3, 4 3, 5 6, 5 6, 3 1, 3 Execution time: 270.9171772003174 ms >> Want to save the solution? (Y/N): y >> Enter file name to save with .txt: sol1.txt Solution saved to 'sol1.txt'. ```

7 10 8 BD 1C 55 55 F3 E9 55 F3 E9 8G 55 E9 1C 7A 8G 8G 55 E9 55 55 F3 55 F3 8G E9 1C 8G E9 F3 1C E9 7A 1C F3 55 1C E9 7A 8G F3 1C 55 F3 7A 8G 8G F3 8G 55 8G 1C 8G BD E9 E9 BD 8G 7A 55 E9 1C 55 F3 E9 7A 1C BD 7A 7A F3 F3 1C 55 BD 55 1C 8G 55 1C 1C 4 F3 E9 E9 32 8G BD E9 34 BD BD F3 E9 24 E9 8G F3 20 *Note In file: (test1.txt) (sol2.txt)	``` --------------------------------------- YOUR RESULT --------------------------------------- Maximum reward: 66 Optimal sequence: BD F3 E9 E9 8G BD E9 Path coordinates: 1, 1 1, 3 5, 3 5, 6 7, 6 7, 7 4, 7 Execution time: 12530.200481414795 ms >> Want to save the solution? (Y/N): y >> Enter file name to save with .txt: sol2.txt Solution saved to 'sol2.txt'. Thank you for using the program! ```
7 7 7 55 7A 7A 7A 7A BD 7A BD 55 55 7A BD 55 BD BD 55 7A 7A BD 55 BD BD 7A BD BD 55 55 7A BD 55 BD 55 7A 55 7A BD BD 55 BD BD 7A 7A 7A 7A BD 55 7A BD 7A 3 7A 2 7A E9 28 BD E9 7A E9 28 *Note In file: (test2.txt) (sol3.txt)	``` --------------------------------------- YOUR RESULT --------------------------------------- Maximum reward: 2 Optimal sequence: 55 BD 55 7A 7A 55 7A Path coordinates: 1, 1 1, 2 2, 2 2, 1 3, 1 3, 2 4, 2 Execution time: 1097.0573425292969 ms >> Want to save the solution? (Y/N): y >> Enter file name to save with .txt: sol3.txt Solution saved to 'sol3.txt'. ```

4.2 Input dari Generate Random (CLI)

INPUT	OUTPUT
Insert number of unique tokens: 5 Insert tokens: BD 1C 55 7A E9 Insert matrix size (row)(column): 6 6 Insert buffer size: 7 Insert number of sequences: 3 Insert maximum sequences: 4 (sol4.txt)	--- GENERATED MATRIX --- 55 1C 55 E9 7A 55 7A BD 1C 55 55 1C 1C E9 1C 7A 7A BD 7A 7A E9 E9 BD E9 1C E9 55 1C BD 1C 7A 55 1C E9 7A 1C --- GENERATED SEQUENCES --- Sequence 1: BD 55 7A E9 Reward sequence 1: 25 Sequence 2: BD 55 E9 Reward sequence 2: 31 Sequence 3: 1C 55 1C BD Reward sequence 3: 55 ---------- YOUR RESULT ---------- Maximum reward: 86 Optimal sequence: 55 1C 55 1C BD 55 E9 Path coordinates: 1, 1 1, 5 3, 5 3, 2 2, 2 2, 6 4, 6 Execution time: 272.402286529541 ms

```
Insert number of unique tokens: 3
Insert tokens: BW RG JJ
Insert matrix size (row)(column): 5 4
```

```
Insert buffer size: 4
Insert number of sequences: 3
Insert maximum sequences: 5
```

(sol5.txt)

```
--- GENERATED MATRIX ---
RG BW BW BW BW RG
RG RG JJ JJ BW RG
RG BW JJ RG RG JJ
JJ JJ JJ RG JJ JJ
```

```
--- GENERATED SEQUENCES ---
Sequence 1: RG JJ RG
Reward sequence 1: 43
Sequence 2: JJ BW JJ
Reward sequence 2: -25
Sequence 3: RG BW RG BW
Reward sequence 3: 12


----------------------------------------
              YOUR RESULT
----------------------------------------

Maximum reward: 43
Optimal sequence: RG RG JJ RG
Path coordinates:
1, 1
1, 2
4, 2
4, 3

Execution time: 1.0800361633300781 ms

>> Want to save the solution? (Y/N): Y
>> Enter file name to save with .txt: sol5.txt

Solution saved to 'sol5.txt'.

Thank you for using the program!
```

```
Insert number of unique tokens: 4
Insert tokens: LL BD CC AI
Insert matrix size (row)(column): 5 4
```

```
Insert buffer size: 5
Insert number of sequences: 5
Insert maximum sequences: 4
```

(sol6.txt)

```
--- GENERATED MATRIX ---
BD BD CC LL
AI CC LL AI
AI LL BD AI
AI BD LL LL
BD AI BD LL
```

```
--- GENERATED SEQUENCES ---
Sequence 1: AI CC CC
Reward sequence 1: 26
Sequence 2: AI LL BD
Reward sequence 2: 79
Sequence 3: LL BD
Reward sequence 3: 81
Sequence 4: LL BD LL
Reward sequence 4: 28
Sequence 5: BD CC CC
Reward sequence 5: 100


---------------------------------------
              YOUR RESULT
---------------------------------------

Maximum reward: 188
Optimal sequence: BD AI LL BD LL
Path coordinates:
1, 1
1, 2
3, 2
3, 3
2, 3

Execution time: 4.518985748291016 ms

>> Want to save the solution? (Y/N): y
>> Enter file name to save with .txt: sol6.txt

Solution saved to 'sol6.txt'.

Thank you for using the program!
```

BAB 5

LAMPIRAN

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

DAFTAR PUSTAKA

Rinaldi Munir. "Aljabar dan Geometri untuk Informatika 2023/2024."
informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/stima23-24.htm

“How to solve Breach Protocol hacking puzzles in Cyberpunk 2077 2.0”, *Dexerto*
How to solve Breach Protocol hacking puzzles in Cyberpunk 2077 2.0 - Dexerto

“Cyberpunk Quickhacking”, *Cyberpunk Wiki*
https://cyberpunk.fandom.com/wiki/Quickhacking)