LAPORAN TUGAS KECIL 01

IF2211 STRATEGI ALGORITMA

“Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force”

Disusun oleh:

Jason Fernando K-03 13522156

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

DAFTAR ISI

BAB 1

DESKRIPSI MASALAH

Breach Protocol dalam Cyberpunk 2077 merupakan permainan mini untuk meretas dalam permainan video tersebut. Mini game ini mensimulasikan kegiatan meretas jaringan lokal dari ICE (Intrusion Countermeasures Electronics) di dunia Cyberpunk 2077.

Beberapa komponen kunci dalam permainan ini melibatkan :

1. Token: Terdiri dari dua karakter alfanumerik seperti E9, BD, dan 55.
2. Matriks: Kumpulan token yang akan dipilih untuk membentuk urutan kode.
3. Sekuens: Serangkaian token (dua atau lebih) yang harus dipasangkan.
4. Buffer: Jumlah maksimum 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. IF2211 Strategi Algoritma – Tugas Kecil 1 1
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.

BAB 2
TEORI SINGKAT

2.1 Algoritma Brute Force

Algoritma brute force merupakan metode sederhana yang mencari solusi dengan menguji semua kemungkinan secara sistematis. Dalam pendekatannya, algoritma ini tidak memanfaatkan struktur atau pola khusus, melainkan mencoba setiap kemungkinan solusi. Keunggulan algoritma ini terletak pada deterministiknya, yang artinya memberikan solusi yang dapat diprediksi dan keandalannya yang tinggi karena pasti menemukan solusi jika ada. Namun, kekurangan utamanya terletak pada kompleksitas waktu yang cenderung tumbuh secara eksponensial seiring dengan meningkatnya ukuran masalah. Algoritma brute force biasanya efektif untuk masalah kecil, namun seringkali menjadi tidak praktis atau tidak layak secara komputasional untuk masalah yang lebih besar. Meskipun demikian, pendekatan ini tetap relevan dan berguna dalam situasi di mana solusi optimal diperlukan dan tidak ada struktur atau informasi tambahan yang dapat dioptimalkan.

BAB 3

IMPLEMENTASI PROGRAM

main.py

function read file

```python
import itertools
import time
import random
import os
import sys

def read_file(filename):
    with open(f'tes/{filename}', 'r') as file:
        buffer_size = int(file.readline())
        rows, cols = map(int, file.readline().split())
        matrix = [list(file.readline().split()) for _ in range(rows)]
        num_seqs = int(file.readline())
        sequences = []
        points = []
        for _ in range(num_seqs):
            sequences.append(file.readline().strip().split())
            points.append(int(file.readline()))
    return buffer_size, matrix, sequences, points
```

function random input

```python
def generate_random_input(rows, cols, num_seqs):
    buffer_size = random.randint(1, 5)
    matrix_size = (rows, cols)

    matrix = [[chr(random.randint(65, 90)) for _ in range(cols)] for _ in range(rows)]

    sequences = [[''.join(chr(random.randint(65, 90)) for _ in range(random.randint(1, 5)))] for _ in range(num_seqs)]

    points = [random.randint(10, 100) for _ in range(num_seqs)]

    print("Randomly Generated Input:")
    print(f"Buffer Size: {buffer_size}")
    print("Matrix:")
    for row in matrix:
        print(" ".join(row))
    print(f"Number of Sequences: {num_seqs}")
    print("Sequences:")
    for seq in sequences:
        print(" ".join(seq))
    print("Points:")
    print(" ".join(map(str, points)))

    return buffer_size, matrix, sequences, points
```

function find all patterns

```python
def find_all_patterns(
    matrix: list[list[str]], step: int
) -> list[list[tuple[str, tuple[int, int]]]]:
    num_rows: int = len(matrix)
    num_cols: int = len(matrix[0])
    all_paths: list[list[tuple[str, tuple[int, int]]]] = []

    def explore_paths(
        current_x: int,
        current_y: int,
        current_path: list[tuple[str, tuple[int, int]]] = [],
        visited_cells: set[tuple[int, int]] = set(),
        current_direction: str = "vertical",
        remaining_steps: int = step,
    ) -> None:
        if remaining_steps == 0:
            all_paths.append(current_path.copy())
            return
        if current_direction == "vertical":
            for next_y in range(num_rows):
                if (current_x, next_y) not in visited_cells:
                    explore_paths(
                        current_x,
                        next_y,
                        current_path + [
                            (matrix[next_y][current_x], (current_x, next_y))
                        ],
                        visited_cells | {(current_x, next_y)},
                        "horizontal",
                        remaining_steps - 1,
                    )
        else:
            for next_x in range(num_cols):
                if (next_x, current_y) not in visited_cells:
                    explore_paths(
                        next_x,
                        current_y,
                        current_path + [
                            (matrix[current_y][next_x], (next_x, current_y))
                        ],
                        visited_cells | {(next_x, current_y)},
                        "vertical",
                        remaining_steps - 1,
                    )

    for x in range(num_cols):
        explore_paths(
            x,
            0,
            current_path=[(matrix[0][x], (x, 0))],
            visited_cells={(x, 0)},
            current_direction="vertical",
            remaining_steps=step,
        )

    return all_paths
```

function calculate point

```python
def calculate_point(matrix, path, sequences, points):
    total_reward = 0
    buffer_tokens = []
    for token, _ in path:
        buffer_tokens.append(token)
        for seq, reward in zip(sequences, points):
            if all(t in buffer_tokens for t in seq):
                total_reward += reward
                buffer_tokens = [t for t in buffer_tokens if t not in seq]
    return total_reward
```

function search optimal path

```python
def compare_path_with_sequence(
    path: list[tuple[str, tuple[int, int]]], sequence: list[str],
) -> bool:
    for i in range(0, len(path)-len(sequence)+1):
        if all(path[i+j][0] == sequence[j] for j in range(len(sequence))):
            return True
    return False

def point_path(path: list[tuple[str, tuple[int, int]]], rewards: list[int], sequences: list[list[str]]) -> int:
    for i in range(len(sequences)):
        if compare_path_with_sequence(path, sequences[i]):
            return rewards[i]
    return 0

def compare_paths(
    all_paths: list[list[tuple[str, tuple[int, int]]]],
    sequences: list[list[str]],
    rewards: list[int],
) -> tuple[list[list[tuple[str, tuple[int, int]]]], int]:
    result = []
    total_points = 0
    current_points = 0
    for path in all_paths:
        for i in range(len(sequences)):
            if compare_path_with_sequence(path, sequences[i]):
                current_points += rewards[i]
        if not result:
            result = path
            total_points = current_points
        else:
            if current_points > total_points:
                result = path
                total_points = current_points
        current_points = 0

    return result, total_points
```

```python
def optimal_pattern(matrix, buffer_size, sequences, points):
    all_paths = find_all_patterns(matrix, buffer_size)
    optimal_path, max_points = compare_paths(all_paths, sequences, points)
    return max_points, optimal_path[:-1]
```

function save result to txt

```python
def save_results(max_points, optimal_path, final_time):
    filename = input("Enter the filename to save the results: ")
    with open(f'tes/{filename}.txt', 'w') as file:
        file.write("RESULT:\n")
        file.write(f"Points obtained: {max_points}\n")
        file.write(f"Path: {' '.join(token for token, _ in optimal_path)}\n")
        file.write("Selected Path Coordinates:\n")
        for _, (x, y) in optimal_path:
            file.write(f"{x + 1}, {y + 1}\n")
        file.write(f"{final_time * 1000} ms\n")
        loader()
        clear_terminal()
    print(f"Results saved to {filename}")
```

main

```python
def initial_display():
    print_ascii_main()
    print("Select Input Method:")
    print("1. Read from txt file")
    print("2. Generate Random Input")
    print("3. Exit")

def clear_terminal():
    os.system('cls' if os.name == 'nt' else 'clear')

def print_ascii_main():
    print('''

  .g8"""bgd `YMM'   `MM'`7MM"""Yp, `7MM"""YMM  `7MM"""Mq.  `7MM"""Mq. `7MMF'    `7MF'`7MN.    `7MF'`7MMF' `YMM'
.dP'     `M   VMA   ,V    MM    Yb   MM    `7    MM   `MM.    MM   `MM.  MM       M    MMN.     M    MM   .M'
dM'       `     VMA ,V     MM    dP   MM   d      MM   ,M9    MM   ,M9   MM       M    M YMb    M    MM .d"
MM              VMMP      MM"""bg.   MMmmMM      MMmmdM9     MMmmdM9    MM       M    M  `MN. M    MMMMM.
MM.              MM       MM    `Y   MM   Y  ,   MM  YM.     MM         MM       M    M   `MM.M    MM  VMA
`Mb.     ,'      MM       MM    ,9   MM     ,M   MM   `Mb.   MM         YM.      ,M    M     YMM    MM   `MM.
  `"bmmmd'     .JMML.   .JMMmmmd9  .JMMmmmmMMM .JMML. .JMM..JMML.        `bmmmmd"'  .JML.    YM  .JMML.   MMb.

        ''')
    loader()
    print("\033c", end="")
```

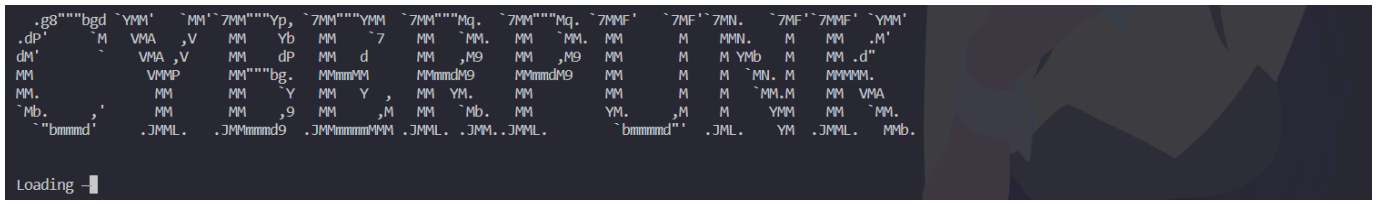
```python
def print_ascii_result():
    print("\033c", end="")
    print('''

`7MM"""Mq.  `7MM"""YMM   .M"""bgd `7MMF'   `7MF'`7MMF'      MMP""MM""YMM
  MM   `MM.   MM    `7  ,MI    "Y   MM       M    MM        P'   MM   `7
  MM   ,M9    MM   d    `MMb.       MM       M    MM             MM
  MMmmdM9     MMmmMM      `YMMNq.   MM       M    MM             MM
  MM  YM.     MM   Y  , .     `MM   MM       M    MM      ,      MM
  MM   `Mb.   MM     ,M Mb     dM   YM.     ,M    MM     ,M      MM
.JMML. .JMM..JMMmmmmMMM P"Ybmmd"     `bmmmmd"'  .JMMmmmmMMM    .JMML.

        ''')

def loader():
    chars = "/-\\|"
    for _ in range(5):
        for char in chars:
            sys.stdout.write('\r' + 'Loading ' + char)
            sys.stdout.flush()
            time.sleep(0.1)
    print()
```

```python
def main():
    initial_display()
    choice = input("Enter your choice (1, 2, or 3): ")

    if choice == '1':
        filename = input("Enter the file name: ")
        loader()
        print_ascii_result()
        buffer_size, matrix, sequences, points = read_file(filename)
    elif choice == '2':
        rows = int(input("Enter the number of rows: "))
        cols = int(input("Enter the number of columns: "))
        num_seqs = int(input("Enter the number of sequences: "))
        print_ascii_result()
        buffer_size, matrix, sequences, points = generate_random_input(rows, cols, num_seqs)
    elif choice == '3':
        clear_terminal()
        return
    else:
        print("Invalid choice. Exiting.")
        return

    start_time = time.time()
    max_points, optimal_path = optimal_pattern(matrix, buffer_size, sequences, points)
    end_time = time.time()
    final_time = end_time - start_time
    print("Points obtained:", max_points)
    print("Path:", ' '.join(token for token, _ in optimal_path))
    print("Selected Path Coordinates:")
    for _, (x, y) in optimal_path:
        print(f"{x + 1}, {y + 1}")
    print(final_time * 1000, "ms")

    save_option = input("Do you want to save the results to a text file? (yes/no): ").lower()
    if save_option == 'yes':
        save_results(max_points, optimal_path, final_time)
    else:
        clear_terminal()
```

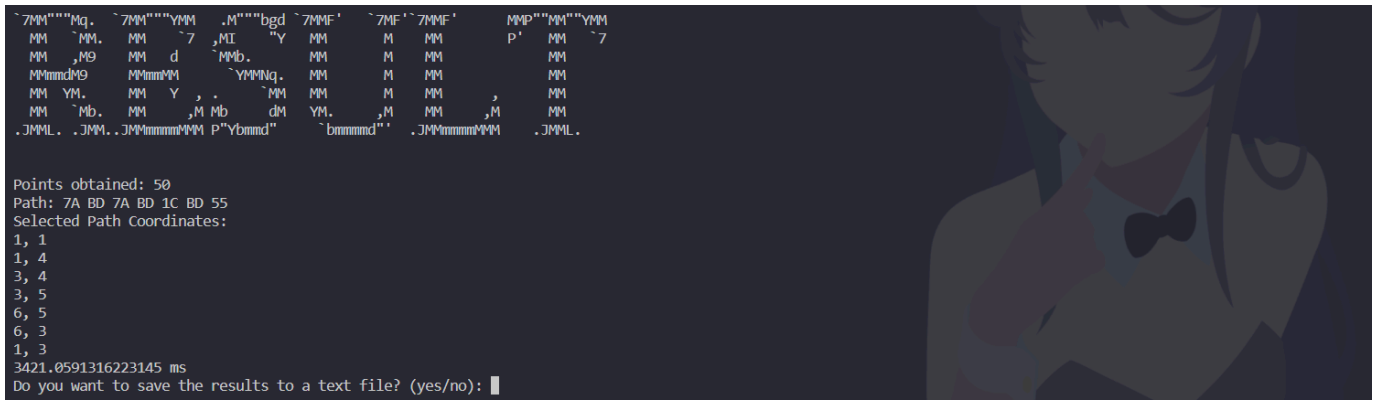
```python
if __name__ == "__main__":
    main()
```

BAB 4

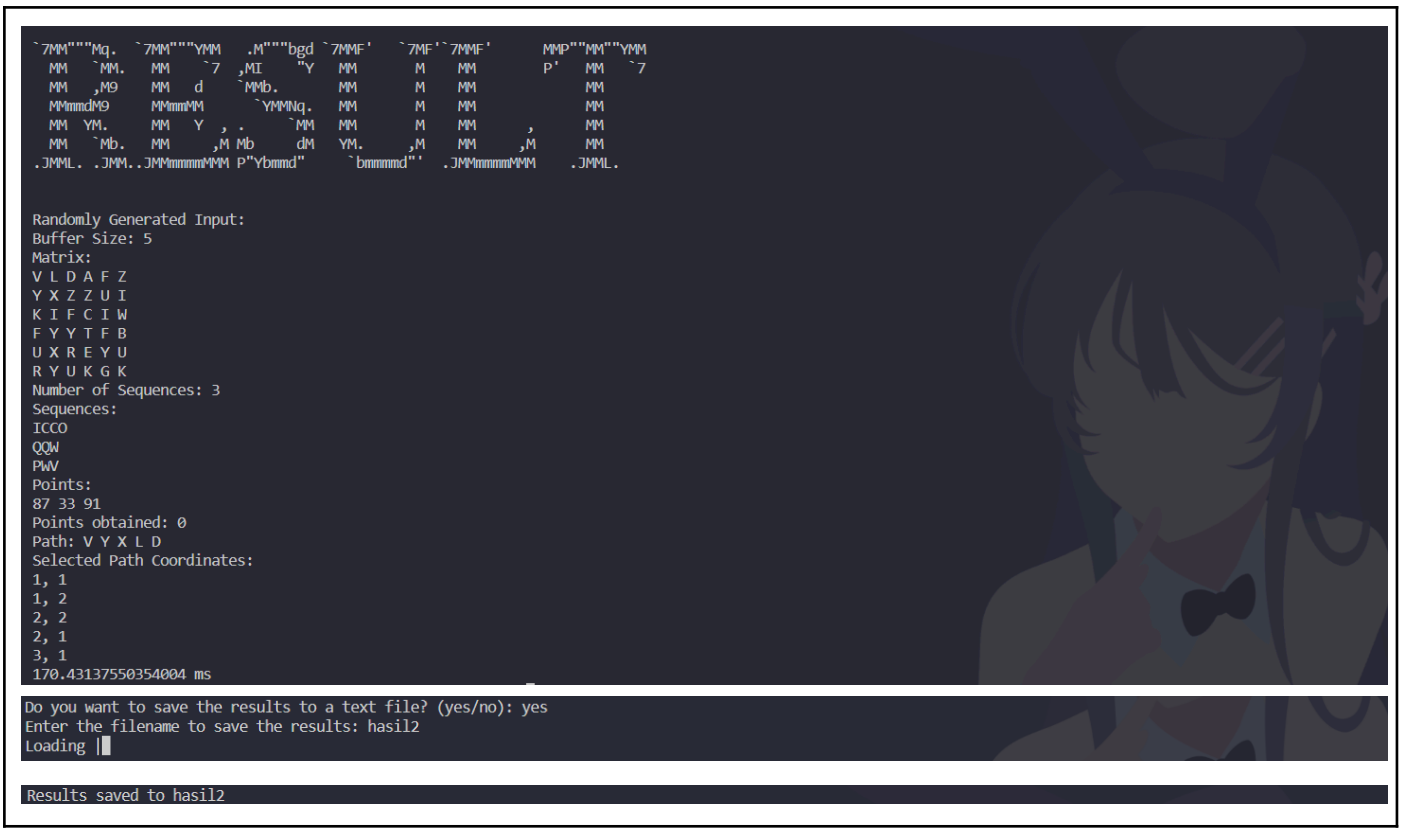
EKSPERIMEN



read file

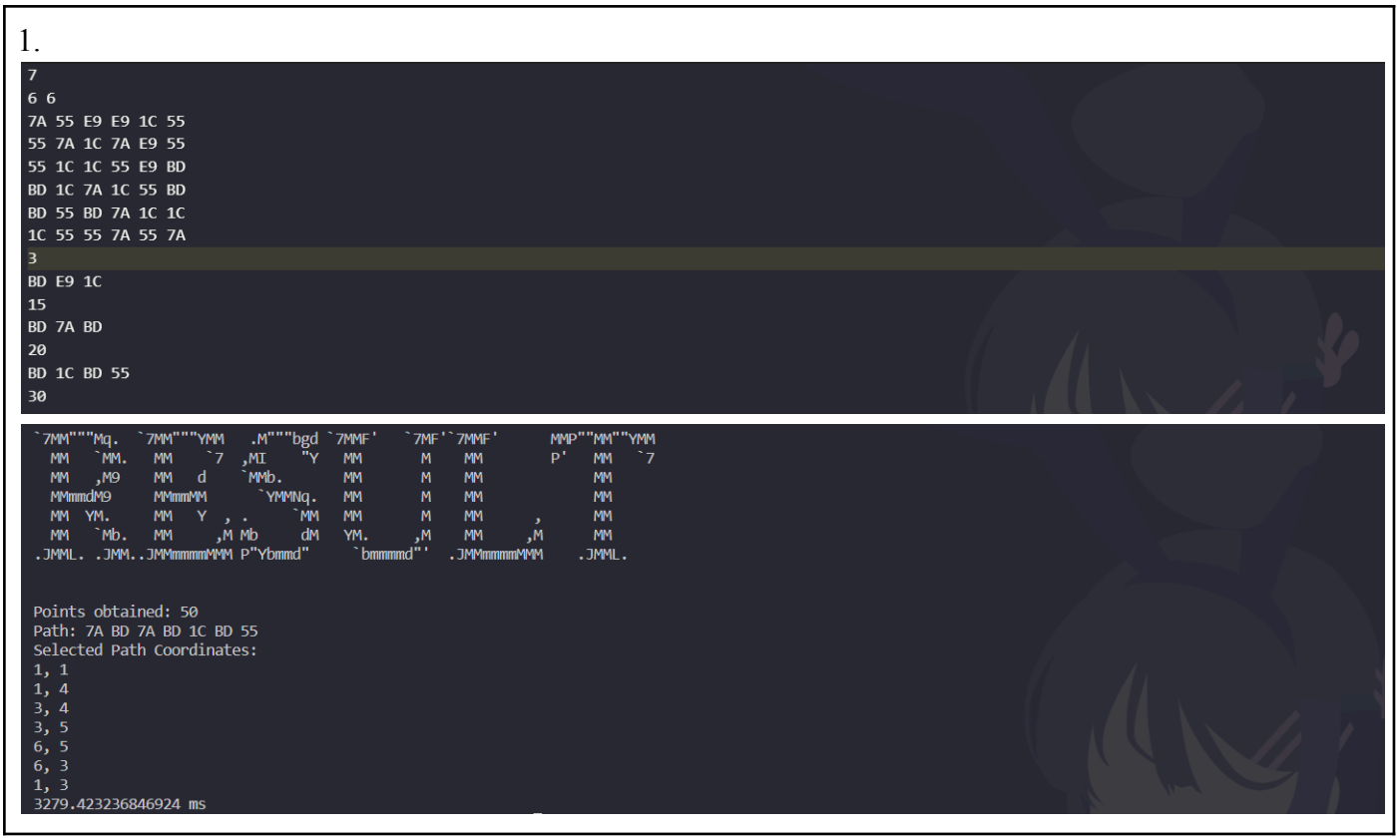


random input


`7MM"""Mq. `7MM"""YMM .M"""bgd `7MMF' `7MF'`7MMF' MMP""MM""YMM
MM `MM. MM `7 ,MI "Y MM M MM P' MM `7
MM ,M9 MM d `MMb. MM M MM MM
MMmmdM9 MMmmMM `YMMNq. MM M MM MM
MM YM. MM Y , . `MM MM M MM , MM
MM `Mb. MM ,M Mb dM YM. ,M MM ,M MM
.JMML. .JMM..JMMmmmmMMM P"Ybmmd" `bmmmmd"' .JMMmmmmMMM .JMML.
Randomly Generated Input:
Buffer Size: 5
Matrix:
V L D A F Z
Y X Z Z U I
K I F C I W
F Y Y T F B
U X R E Y U
R Y U K G K
Number of Sequences: 3
Sequences:
ICCO
QQW
PWV
Points:
87 33 91
Points obtained: 0
Path: V Y X L D
Selected Path Coordinates:
1, 1
1, 2
2, 2
2, 1
3, 1
170.43137550354004 ms
Do you want to save the results to a text file? (yes/no): yes
Enter the filename to save the results: hasil2
Loading
Results saved to hasil2

test case

1.


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
`7MM"""Mq. `7MM"""YMM .M"""bgd `7MMF' `7MF'`7MMF' MMP""MM""YMM
MM `MM. MM `7 ,MI "Y MM M MM P' MM `7
MM ,M9 MM d `MMb. MM M MM MM
MMmmdM9 MMmmMM `YMMNq. MM M MM MM
MM YM. MM Y , . `MM MM M MM , MM
MM `Mb. MM ,M Mb dM YM. ,M MM ,M MM
.JMML. .JMM..JMMmmmmMMM P"Ybmmd" `bmmmmd"' .JMMmmmmMMM .JMML.
Points obtained: 50
Path: 7A BD 7A BD 1C BD 55
Selected Path Coordinates:
1, 1
1, 4
3, 4
3, 5
6, 5
6, 3
1, 3
3279.423236846924 ms

2.

```
7
7 7
A2 B4 C6 D8 E0 F1 G3
H5 I7 J9 K0 L1 M2 N3
O4 P5 Q6 R7 S8 T9 UA
VB WC XD YE ZF 1G 2H
3I 4J 5K 6L 7M 8N 9O
AP BQ CR DS ET FU GV
3I 4J 5K 6L 7M 8N 9O
2
B4 C6 D8 E0
6
J9 K0 L1 M2
8
```

```
`7MM"""Mq.  `7MM"""YMM   .M"""bgd `7MMF'   `7MF'`7MMF'      MMP""MM""YMM
  MM   `MM.   MM    `7  ,MI    "Y   MM       M    MM        P'   MM   `7
  MM   ,M9    MM   d    `MMb.       MM       M    MM             MM
  MMmmdM9     MMmmMM      `YMMNq.   MM       M    MM             MM
  MM  YM.     MM   Y  , .     `MM   MM       M    MM      ,      MM
  MM   `Mb.   MM     ,M Mb     dM   YM.     ,M    MM     ,M      MM
.JMML. .JMM..JMMmmmmMMM P"Ybmmd"     `bmmmmd"'  .JMMmmmmMMM    .JMML.


Points obtained: 0
Path: A2 H5 I7 B4 C6 J9 K0
Selected Path Coordinates:
1, 1
1, 2
2, 2
2, 1
3, 1
3, 2
4, 2
10252.936601638794 ms
```

3.

```
4
5 5
J1 K2 L3 M4 N5
O6 P7 Q8 R9 SA
TB UC VD WE XF
YG ZH 1I 2J 3K
O6 P7 Q8 R9 SA
3
K2 L3 M4
7
1I 2J 3K
10
1I 3K M4
15
```

```
`7MM"""Mq.  `7MM"""YMM   .M"""bgd `7MMF'   `7MF'`7MMF'      MMP""MM""YMM
  MM   `MM.   MM    `7  ,MI    "Y   MM       M    MM        P'   MM   `7
  MM   ,M9    MM   d    `MMb.       MM       M    MM             MM
  MMmmdM9     MMmmMM      `YMMNq.   MM       M    MM             MM
  MM  YM.     MM   Y  , .     `MM   MM       M    MM      ,      MM
  MM   `Mb.   MM     ,M Mb     dM   YM.     ,M    MM     ,M      MM
.JMML. .JMM..JMMmmmmMMM P"Ybmmd"     `bmmmmd"'  .JMMmmmmMMM    .JMML.


Points obtained: 0
Path: J1 O6 P7 K2
Selected Path Coordinates:
1, 1
1, 2
2, 2
2, 1
7.006645202636719 ms
```

4.

```
6
7 7
M1 N2 O3 P4 Q5 R6 S7
T8 U9 VA WB XC YD ZE
1F 2G 3H 4I 5J 6K 7L
8M 9N AO BP CQ DR ES
FT GU HV IW JX KY LZ
MI NJ OK PL QM RN SO
T8 U9 VA WB XC YD ZE
2
O3 P4 Q5 R6
6
M1 T8 U9 N2 O3 VA
10
```

```
`7MM"""Mq.  `7MM"""YMM   .M"""bgd `7MMF'   `7MF'`7MMF'      MMP""MM""YMM
  MM   `MM.   MM    `7  ,MI    "Y   MM       M    MM        P'   MM   `7
  MM   ,M9    MM   d    `MMb.       MM       M    MM             MM
  MMmmdM9     MMmmMM      `YMMNq.   MM       M    MM             MM
  MM  YM.     MM   Y  , .     `MM   MM       M    MM      ,      MM
  MM   `Mb.   MM     ,M Mb     dM   YM.     ,M    MM     ,M      MM
.JMML. .JMM..JMMmmmmMMM P"Ybmmd"     `bmmmmd"'  .JMMmmmmMMM    .JMML.


Points obtained: 10
Path: M1 T8 U9 N2 O3 VA
Selected Path Coordinates:
1, 1
1, 2
2, 2
2, 1
3, 1
3, 2
1309.2777729034424 ms
```

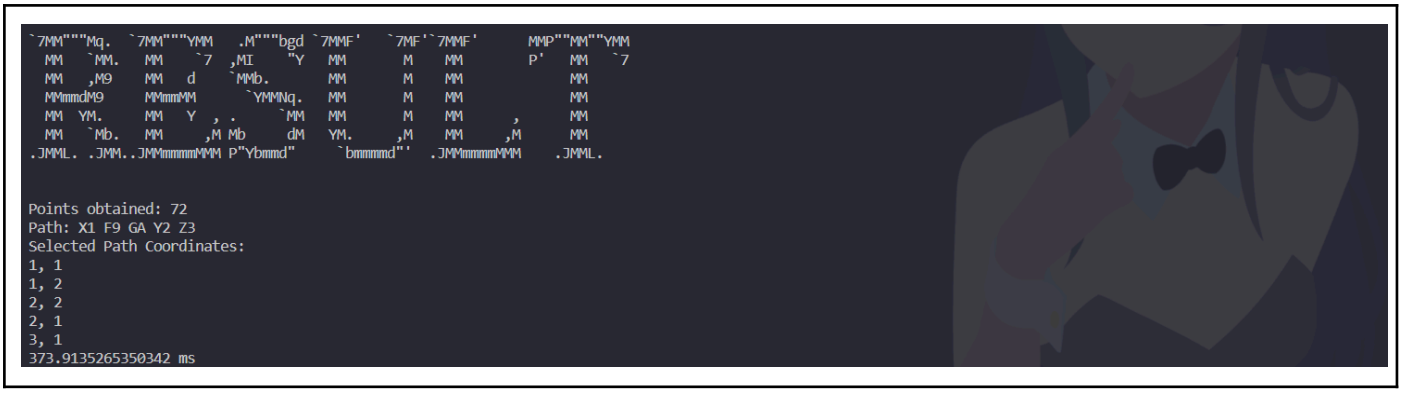
5.

```
8
6 6
P1 Q2 R3 S4 T5 U6
V7 W8 X9 YA ZB 1C
2D 3E 4F 5G 6H 7I
8J 9K AL BM CN DO
EP FQ GR HS IT JU
KV LW MX NY OZ PA
3
V7 X9 YA S4
7
W8 Q2 R3 X9
10
P1 V7 W8
15
```

```
Points obtained: 25
Path: P1 V7 W8 Q2 R3 X9 YA S4
Selected Path Coordinates:
1, 1
1, 2
2, 2
2, 1
3, 1
3, 2
4, 2
4, 1
17017.17758178711 ms
```

6.

```
5
7 7
X1 Y2 Z3 A4 B5 C6 D7
F9 GA HB IC JD KE LF
NH OI PJ QK RL SM TN
VP WQ XR YS ZT 1U 2V
4X 5Y 6Z 7A 8B 9C AD
CF DG EH FI GJ HK IL
KN LO MP NQ OR PS QT
3
X1 F9
7
F9 GA Y2
20
Y2 Z3
45
```


`7MM"""Mq.  `7MM"""YMM   .M"""bgd `7MMF'   `7MF'`7MMF'   MMP""MM""YMM
MM   `MM.   MM    `7 ,MI   "Y   MM   M   MM   P'   MM   `7
MM   ,M9   MM   d   `MMb.   MM   M   MM   MM
MMmmdM9   MMmmMM   `YMMNq.   MM   M   MM   MM
MM YM.   MM   Y , .   `MM   MM   M   MM   ,   MM
MM   `Mb.   MM   ,M Mb   dM   YM.   ,M   MM   ,M   MM
.JMML. .JMM..JMMmmmmMMM P"Ybmmd"   `bmmmmd"' .JMMmmmmMMM   .JMML.
Points obtained: 72
Path: X1 F9 GA Y2 Z3
Selected Path Coordinates:
1, 1
1, 2
2, 2
2, 1
3, 1
373.9135265350342 ms

BAB 5
PENUTUP

5.1 Kesimpulan

Algoritma brute force efektif dalam menyelesaikan permasalahan, namun keefisiensiannya terbatas pada ukuran matriks dan sekuens yang besar. Proses mencoba semua kemungkinan kombinasi dapat menjadi tidak optimal dalam hal waktu dan sumber daya. Oleh karena itu, dalam skenario dengan skala permasalahan yang besar, pertimbangan untuk menggunakan pendekatan algoritmik yang lebih canggih atau strategi optimasi yang lebih spesifik dapat menjadi kunci untuk meningkatkan efisiensi pemecahan masalah.

5.2 Link Repository

Link repository untuk tugas kecil 1 mata kuliah IF2211 Strategi Algoritma adalah sebagai berikut

Link : https://github.com/JasonFernandoo/Tucil1_13522156

5.3 Tabel Checkpoint Program

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		✓

DAFTAR REFERENSI

https://nicolas-siplis.com/blog/cyberpwned