

Laporan Tugas Kecil 1

IF2211 Strategi Algoritma Semester II tahun 2023/2024

Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force



Dibuat oleh :

Hugo Sabam Augusto - 13522129

1. Deskripsi Tugas



Breach Protocol dalam Cyberpunk 2077 adalah minigame peretasan yang mensimulasikan intrusi ke dalam jaringan ICE (Intrusion Countermeasures Electronics) dalam permainan. Komponennya mencakup Token (dua karakter alfanumerik seperti E9, BD, dan 55), Matriks (kumpulan token untuk membentuk kode), Sekuens (rangkaian token yang harus dicocokkan), dan Buffer (jumlah maksimal token yang dapat disusun secara sekuensial).

Aturan permainan melibatkan pemain bergerak secara bergantian horizontal dan vertikal sampai semua sekuens cocok atau buffer penuh. Pemain memulai dengan memilih satu token di posisi paling atas matriks, dan sekuens dicocokkan dengan token di buffer. Satu token dapat digunakan untuk lebih dari satu sekuens, dan setiap sekuens memiliki bobot hadiah yang berbeda. Minimal panjang sekuens adalah dua token.

2. Algoritma Brute Force

a. Inisialisasi

Pertama sudah dipastikan matriks harus diinisialisasi, lalu panggil fungsi *generate Sequences* beserta parameter-parameternya

b. Membuat Sekuensial

Diawali dengan menginisialisasi list 'sequences', lalu iterasi setiap baris matriks. Setiap baris memulai jalur baru dari kolom pertama. Setelah itu mulau memasuki ke fungsi rekursif nya *explorePath*.

c. Rekursif

Metode *explorePath* berfungsi untuk mengeksplorasi semua kemungkinan langkah dalam sebuah jalur. Pada awalnya, metode ini memeriksa apakah tidak ada langkah tersisa dalam jalur (*remainingSteps == 0*), jika iya, maka urutan jalur ditambahkan ke dalam list *sequences*. Jika masih terdapat langkah tersisa, metode melakukan iterasi melalui arah yang mungkin, yaitu vertikal atau horizontal. Untuk setiap arah, langkah-langkah dilakukan sebanyak *maxSteps*. Selanjutnya, posisi baru ditentukan berdasarkan arah dan langkah, kemudian dilakukan pengecekan kevalidan posisi baru serta memastikan belum ada dalam jalur saat ini. Jika posisi baru valid, jalur baru dibuat dengan menambahkan posisi baru, dan metode rekursif panggil *explorePath* dengan parameter yang diperbarui. Dengan demikian, metode *explorePath* secara berulang menjelajahi semua kemungkinan langkah dalam jalur yang valid.

d. Validasi koordinat/posisi + backtracking

Untuk validasi koordinat atau posisi, saya membuat fungsi *isValidPosition*. Fungsi ini memeriksa apakah suatu posisi berada dalam batas matriks. Tak hanya itu, dibuat juga *containsPosition*. Sesuai dengan namanya, fungsi ini memeriksa apakah suatu posisi sudah ada dalam jalur saat ini sehingga sangat membantu backtracking

3. Source Program

Untuk membuat program ini, bahasa yang saya gunakan adalah bahasa Java (Java version "20.0.2" 2023-07-18). Berikut kode sumber deklarasi object-object, algoritma *Brute Force*, Input dan Output.

Deklarasi object

a. Matrix.java

```
public class Matrix {
    public int row, col;
    public String[][] Matrix;

    // Constructor
    public Matrix(int row, int col){
        this.row = row;
        this.col = col;
        this.Matrix = new String[row][col];
    }

    //Getter
    public String getElmt(int i, int j){
        return this.Matrix[i][j];
    }

    public int getRowEff(){
        return this.row;
    }

    public int getColEff(){
        return this.col;
    }
}
```

```
//Setter  
public void setElmt(int i, int j, String x){  
    this.Matrix[i][j] = x;  
}  
  
}
```

b. Result.java

```
import java.util.ArrayList;  
import java.util.List;  
  
public class Result {  
    public List<List<Integer>> listCoords;  
    public List<String> sequence;  
    public int i;  
    public int totalPoints;  
  
    public Result(int i, int totalPoints) {  
        listCoords = new ArrayList<>(i);  
        sequence = new ArrayList<>(i);  
        this.totalPoints = totalPoints;  
    }  
    public void printResult() {  
        System.out.println("Coords '[COL, ROW]' : " +  
listCoords);  
        System.out.println("Sequence: " + sequence);  
        System.out.println("Total Points: " + totalPoints);  
    }  
}
```

c. Sequence.java

```
package tools;

public class Sequence {
    public int points,lengthEff,i;
    public String[] Seq;

    //Constructor
    public Sequence(int lengthEff,int points){
        this.points = points;
        this.lengthEff = lengthEff;
        this.Seq = new String[lengthEff];

    //Getter
    }

    public int getPoints(){
        return points;
    }
    public String getSeqToken(int i){
        return Seq[i];
    }
    public int getSeqlength()
    {
        return lengthEff;
    }

    //Setter
    public void setSeqToken(int i,String x){
        this.Seq[i] = x;
    }
}
```

```

//Method
}

public void displaySequence(){
    System.out.print(points + " , ");
    for (int i = 0; i < lengthEff; i++) {
        System.out.print(getSeqToken(i) + " ");
    }
}

4. }

```

Algoritma brute force

a. Brute.java

```

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class Brute {
    public static int maxx = 0;
    public static ArrayList<Result> resultList = new ArrayList<>();

    public static void generateSequences(Matrix matrix, int
maxSteps, List<List<String>> sequences, Sequence[] prizeSeq) {
        sequences.clear();
        long startTime = System.currentTimeMillis();

        for (int startRow = 0; startRow < matrix.getRowEff(); startRow++)
        {
            List<int[]> currentPath = new ArrayList<>();
            int[] startPosition = {startRow, 0};

```

```

        currentPath.add(startPosition);
        explorePath(currentPath, startPosition, maxSteps - 1,
true,maxSteps, matrix,sequences,prizeSeq);
    }
    long endTime = System.currentTimeMillis();
    long exeTime = endTime - startTime;

    System.out.println("=====FINAL
RESULT=====");
    if (resultList.size() > 0)
    {
        Scanner scanner2 = new Scanner(System.in);
        Result maxResult = maxPointFromResult(resultList);
        maxResult.printResult();
        System.out.println("Execution Time: " + exeTime + " ms");
        String outputParent = "../test/output/";
        System.out.println("Apakah hasil output mau di save? (y/n) :
");

        String inputCmd = scanner2.nextLine();
        if(inputCmd.equals("y")){
            System.out.print("Ketik nama file anda (tanpa .txt) : ");
            String fileName = scanner2.nextLine();
            Save.writeToTextFile(prizeSeq,exeTime,matrix, maxResult,
outputParent + fileName + ".txt");
            scanner2.close();
        }
    }
    else
    {
        System.out.println("No Solution");
    }
}

private static void explorePath(List<int[]> currentPath, int[]
currentPosition, int remainingSteps, boolean isVertical,int
maxSteps,Matrix matrix,List<List<String>> sequences,Sequence[] prizeSeq)
{
    if (remainingSteps == 0) {

```



```

List<String> sequence = new ArrayList<>();
List<List<Integer>> listCoord = new ArrayList<>();
for (int[] position : currentPath) {
    List<Integer> coord = new ArrayList<>();
    int row = position[0];
    int col = position[1];
    sequence.add(matrix.getElmt(row, col));
    coord.add(col+1);
    coord.add(row+1);
    listCoord.add(coord);
    // System.out.println("debug1" + listCoord);
}

// hasil tiap sequence disini
Sequence[] tempPrizeSeq = prizeSeq;

int max = 0;
//pengambilan poin untuk tiap path yang diambil
for (int i = 0; i < prizeSeq.length; i++) {
    if(stringMatch(sequence, tempPrizeSeq[i]) == true){
        max = max + tempPrizeSeq[i].getPoints();
    }
}
//pengambilan nilai maximal
if (max > maxx){
    maxx = max;
    Result result = new Result(sequence.size(), maxx);
    result.listCoords = listCoord;
    result.sequence = sequence;
    // result.printResult();
    resultList.add(result);
}
sequences.add(sequence);
return;
}

```

```

int row = currentPosition[0];
int col = currentPosition[1];
int[][] directions;

if (isVertical) {
    directions = new int[][]{{1, 0}, {-1, 0}}; // vertikal
} else {
    directions = new int[][]{{0, 1}, {0, -1}}; // horizontal
}

for (int[] direction : directions) {
    for (int step = 1; step <= maxSteps; step++) {
        int newRow = row + direction[0] * step;
        int newCol = col + direction[1] * step;
        int[] newPosition = {newRow, newCol};

        if (isValidPosition(newPosition, matrix) &&
!containsPosition(currentPath, newPosition)) {
            List<int[]> newPath = new ArrayList<>(currentPath);
            newPath.add(newPosition);

            explorePath(newPath, newPosition, remainingSteps - 1,
!isVertical, maxSteps, matrix, sequences, prizeSeq);
        }
    }
}

private static boolean isValidPosition(int[] position, Matrix matrix)
{
    int row = position[0];
    int col = position[1];
    return row >= 0 && row < matrix.getRowEff() && col >= 0 && col <
matrix.getColEff();
}

private static boolean containsPosition(List<int[]> path, int[]
position) {

```

```

        for (int[] p : path) {
            if (p[0] == position[0] && p[1] == position[1]) {
                return true;
            }
        }
        return false;
    }

    public static void printSequences(List<List<String>> sequences) {
        for (int i = 0; i < sequences.size(); i++) {
            System.out.printf("Sequence    %d:    %s\n",    i    +    1,
sequences.get(i));
        }
    }

    public static boolean stringMatch(List<String> textArray, Sequence
patternArray) {

        for (int i = 0; i <= textArray.size() - patternArray.Seq.length;
i++) {
            int j = 0;

            while (j < patternArray.Seq.length && textArray.get(i +
j).equals(patternArray.Seq[j])) {
                j++;
            }

            if (j == patternArray.Seq.length) {
                return true;
            }
        }
        return false;
    }

    public static Result maxPointFromResult(ArrayList<Result>
resultList){
        int max = resultList.get(0).totalPoints;

```

```

        int idMax = 0;
        for (int i = 1; i < resultList.size(); i++) {
            if (resultList.get(i).totalPoints > max)
            {
                idMax = i;
            }
        }

        return resultList.get(idMax);
    }

5. }

```

Input dan Output

a. parse.java

```

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.Random;

public class parse {
    public static List<List<String>> raw_seq_list = new ArrayList<>();

    public static int readBufferSize(BufferedReader reader) throws IOException {
        String line;
        line = reader.readLine();
        int bufferSize = Integer.parseInt(line.trim());
        return bufferSize;
    }

    public static Matrix readMatrixSize(BufferedReader reader) throws IOException
    {
        String line;
        line = reader.readLine();
        String[] words = line.split("\\s+");
    }
}

```

```

        Matrix    matrix    =    new    Matrix(Integer.parseInt(words[0]),
Integer.parseInt(words[1]));
        return matrix;
    }

    public static void inputMatrixFromFile(BufferedReader reader,Matrix m) throws
IOException {
        int row = m.getRowEff();
        for (int i = 0; i < row; i++) {
            String line = reader.readLine();
            String[] words = line.split("\\s+");

            int j = 0;
            for(String token : words){
                m.setElmt(i, j, token);
                j ++;
            }
        }

        public static Sequence[] inputSequencesFromFile(BufferedReader reader) throws
IOException {
            String line = reader.readLine(); // baca size sequence
            int sequenceSize = Integer.parseInt(line.trim());
            Sequence[] arrSeq = new Sequence[sequenceSize];
            for (int i = 0; i < sequenceSize; i++) {
                line = reader.readLine(); // baca isi sequence
                String[] words = line.split("\\s+");
                line = reader.readLine(); // baca point
                int points = Integer.parseInt(line.trim());
                Sequence sequen = new Sequence(words.length, points);

                // Set token Sequence
                for (int j = 0; j < words.length; j++) {
                    sequen.setSeqToken(j, words[j]);
                }
                arrSeq[i] = sequen;
            }

            return arrSeq;

        }

        public static void displayMatrix(Matrix m) {

```

```

        for (int i = 0; i < m.getRowEff(); i++) {
            for (int j = 0; j < m.getColEff(); j++) {
                System.out.print(m.getElmt(i, j) + " ");
            }
            System.out.println();
        }
    }

    public static void displaySequences(Sequence[] ses){
        System.out.println("=====Sequences=====");
        for (int i = 0; i < ses.length; i++) {
            ses[i].displaySequence();
            System.out.println();
        }
        System.out.println("=====");
    }

    public class Result {
        public Matrix matrix;
        public Sequence[] arrSeq;

        public Result(Matrix matrix, Sequence[] arrSeq) {
            this.matrix = matrix;
            this.arrSeq = arrSeq;
        }
    }

    public static void inputCli(Scanner scanner){
        Random random = new Random();
        System.out.print("Masukkan jumlah_token_unik: ");
        int sumUniqueToken = scanner.nextInt();
        scanner.nextLine();

        System.out.print("Masukkan token: ");
        String tokens = scanner.nextLine();
        String[] arrayOfTokens = tokens.split("\\s+");
        while (arrayOfTokens.length != sumUniqueToken) {
            System.out.print("Jumlah Tidak valid! Masukkan token kembali : ");
            tokens = scanner.nextLine();
            arrayOfTokens = tokens.split("\\s+");
        }
    }

```

```

        // for (String token : arrayOfTokens) {
        //     System.out.println(token);
        // }

        System.out.print("ukuran_buffer: ");
        int bufferSize = scanner.nextInt();
        scanner.nextLine();

        System.out.print("ukuran_matriks: ");
        String matrixSize = scanner.nextLine();
        String[] words = matrixSize.split("\\s+");
        while (words.length != 2) {
            System.out.print("Input tidak valid! Ketik kembali ukuran matriks (M
N): ");

            matrixSize = scanner.nextLine();
            words = matrixSize.split("\\s+");
        }
        Matrix matrix = new Matrix(Integer.parseInt(words[0]),
Integer.parseInt(words[1]));

        System.out.print("jumlah_sekuens: ");
        int sequens = scanner.nextInt();
        scanner.nextLine();

        System.out.print("ukuran_maksimal_sekuens: ");
        int maxSizeSequens = scanner.nextInt();
        scanner.nextLine();

        //Pembuatan matriks random
        for (int i = 0; i < matrix.getRowEff(); i++) {
            for (int j = 0; j < matrix.getColEff(); j++) {
                int randomIndex = random.nextInt(sumUniqueToken);
                matrix.setElmt(i, j, arrayOfTokens[randomIndex]);
            }
        }
        displayMatrix(matrix);

        //Pembuatan sequens random
        Sequence[] arrSeq = new Sequence[sequens];
        for (int i = 0; i < sequens; i++) {

```

```

        Sequence sequen = new Sequence(random.nextInt(1,maxSizeSequens),
random.nextInt(30)); // random points
        for (int j = 0; j < sequen.getSeqlength(); j++) {
            sequen.setSeqToken(j,
arrayOfTokens[random.nextInt(sumUniqueToken)]);
        }
        arrSeq[i] = sequen;

    }
    displaySequences(arrSeq);
    Brute.generateSequences(matrix,bufferSize,raw_seq_list,arrSeq);
}

public static void inputTxt(Scanner scanner) {

    boolean isValidInput = false;

    do {
        String fileParent = "../test/input/";
        System.out.print("Masukkan nama file (tanpa .txt): ");
        String userInput = scanner.nextLine();
        String filePath = fileParent + userInput + ".txt";

        try (BufferedReader br = new BufferedReader(new
FileReader(filePath))) {

            int bufferSize = readBufferSize(br);
            Matrix matrix = parse.readMatrixSize(br);
            inputMatrixFromFile(br, matrix);
            Sequence[] arrSeq = parse.inputSequencesFromFile(br);
            displayMatrix(matrix);
            displaySequences(arrSeq);
            Brute.generateSequences(matrix,bufferSize,raw_seq_list,arrSeq);

            // test.printSequences(raw_seq_list);

            isValidInput = true;
        } catch (FileNotFoundException e) {

```



```

        System.err.println("File tidak ditemukan: " + filePath);
    } catch (IOException e) {
        e.printStackTrace();
    }
} while (!isValidInput);
}
}

```

b. Save.java

```

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;

public class Save {
    public static void writeToTextFile(Sequence[] arrSeq, long
exeTime, Matrix matrix, Result result, String filePath) {
        try (BufferedWriter writer = new BufferedWriter(new
FileWriter(filePath))) {
            String resultString = resultToString(result);
            String matrixString = matrixToString(matrix);
            String arrSeqString = sequencesToString(arrSeq);
            String timeString = "\nExecute Time : " +
String.valueOf(exeTime) + " ms";
            writer.write(matrixString);

            writer.write("=====\n");
            writer.write(arrSeqString);

            writer.write("=====\n");
            writer.write(resultString);
            writer.write(timeString);

            System.out.println("Output berhasil disimpan di." +
filePath);

```

```

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static String resultToString(Result result) {
        StringBuilder sb = new StringBuilder();
        sb.append("Coords          '[COL,          ROW]'          :
").append(result.listCoords).append("\n");
        sb.append("Sequence:
").append(result.sequence).append("\n");
        sb.append("Total Points: ").append(result.totalPoints);
        return sb.toString();
    }

    private static String matrixToString(Matrix matrix) {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < matrix.getRowEff(); i++) {
            for (int j = 0; j < matrix.getColEff(); j++) {
                sb.append(matrix.getElmt(i, j)).append("\t");
            }
            sb.append("\n");
        }
        return sb.toString();
    }

    private static String sequencesToString(Sequence[] arrSeq)
    {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < arrSeq.length; i++) {
            sb.append(arrSeq[i].points).append(" Points | ");
            for (int j = 0; j < arrSeq[i].lengthEff; j++) {
                sb.append(arrSeq[i].getSeqToken(j)).append(" ");
            }
            sb.append("|\n");
        }
    }

```

```

    }
    return sb.toString();
}
}

```

6. Tangkapan Layar Program

a. Percobaan ke-1 (FILE)

Input:

```

1. TXT:
2. COMMAND LINE:
3. EXIT
Pilih Input : 1
Masukkan nama file (tanpa .txt): setting

```

setting.txt

```

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

Output:

```
=====FINAL RESULT=====
Coords '[COL, ROW]' : [[1, 1], [1, 4], [3, 4], [3, 5], [6, 5], [6, 4], [5, 4]]
Sequence: [7A, BD, 7A, BD, 1C, BD, 55]
Total Points: 50
Execution Time: 100 ms
Apakah hasil output mau di save? (y/n) :
y
Ketik nama file anda (tanpa .txt) : output_setting
Output berhasil disimpan di.../test/output/output_setting.txt
```

output_setting.txt

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

=====

15 Points | BD E9 1C |

20 Points | BD 7A BD |

30 Points | BD 1C BD 55 |

=====

Coords '[COL, ROW]' : [[1, 1], [1, 4], [3, 4], [3, 5], [6, 5], [6, 4], [5, 4]]

Sequence: [7A, BD, 7A, BD, 1C, BD, 55]

Total Points: 50

Execute Time : 100 ms

b. Percobaan ke-2 (FILE)

Input:

```
1. TXT:
2. COMMAND LINE:
3. EXIT
Pilih Input : 1
Masukkan nama file (tanpa .txt): setting2
```

setting2.txt

```
8
7 7
DD FF JJ AA II II BB
CC BB BB FF BB CC AA
FF BB HH BB GG AA JJ
GG EE AA DD JJ DD II
AA JJ II FF GG GG EE
EE JJ AA AA JJ CC FF
II AA AA FF CC II EE
5
DD FF AA BB CC
20
AA CC HH
10
AA II EE JJ FF
30
GG FF
10
AA AA AA
25
```

Output:

```
=====FINAL RESULT=====
Coords '[COL, ROW]' : [[1, 4], [1, 3], [2, 3], [2, 7], [1, 7], [1, 6], [2, 6], [2, 1]]
Sequence: [GG, FF, BB, AA, II, EE, JJ, FF]
Total Points: 40
Execution Time: 1214 ms
Apakah hasil output mau di save? (y/n) :
y
Ketik nama file anda (tanpa .txt) : output_setting2
Output berhasil disimpan di.../test/output/output_setting2.txt
```

output_setting2.txt

DD	FF	JJ	AA	II	II	BB
CC	BB	BB	FF	BB	CC	AA
FF	BB	HH	BB	GG	AA	JJ
GG	EE	AA	DD	JJ	DD	II
AA	JJ	II	FF	GG	GG	EE
EE	JJ	AA	AA	JJ	CC	FF
II	AA	AA	FF	CC	II	EE

=====

20 Points | DD FF AA BB CC |

10 Points | AA CC HH |

30 Points | AA II EE JJ FF |

10 Points | GG FF |

25 Points | AA AA AA |

=====

Coords '[COL, ROW]' : [[1, 4], [1, 3], [2, 3], [2, 7], [1, 7], [1, 6], [2, 6], [2, 1]]

Sequence: [GG, FF, BB, AA, II, EE, JJ, FF]

Total Points: 40

Execute Time : 1214 ms

c. Percobaan ke-3 (FILE)

Input:

```
1. TXT:
2. COMMAND LINE:
3. EXIT
Pilih Input : 1
Masukkan nama file (tanpa .txt): setting3
```

setting3.txt

```
6
9 9
90 56 78 77 34 12 44 77 34
56 12 55 56 22 34 44 34 55
66 44 12 44 11 56 11 11 44
12 90 66 22 90 90 12 66 56
78 66 33 44 34 55 55 22 22
44 11 22 77 77 12 22 34 90
44 22 44 66 33 55 12 90 22
11 34 66 78 77 12 55 90 90
78 55 77 34 22 55 33 34 34
6
78 66
14
33
2
56 33 34 90
13
12 33 12 34 77
15
33 90 12 77
25
44 33 11 56 34
50
```

Output:

output_setting3.txt

90	56	78	77	34	12	44	77	34
56	12	55	56	22	34	44	34	55
66	44	12	44	11	56	11	11	44
12	90	66	22	90	90	12	66	56
78	66	33	44	34	55	55	22	22
44	11	22	77	77	12	22	34	90
44	22	44	66	33	55	12	90	22
11	34	66	78	77	12	55	90	90
78	55	77	34	22	55	33	34	34
=====								
14 Points 78 66								
2 Points 33								
13 Points 56 33 34 90								
15 Points 12 33 12 34 77								
25 Points 33 90 12 77								
50 Points 44 33 11 56 34								
=====								
Coords '[COL, ROW]': [[1, 1], [1, 7], [5, 7], [5, 3], [6, 3], [6, 2]]								
Sequence: [90, 44, 33, 11, 56, 34]								
Total Points: 52								
Execute Time : 179 ms								

d. Percobaan ke-4 (CLI)

Input:


```

1. TXT:
2. COMMAND LINE:
3. EXIT
Pilih Input : 2
Masukkan jumlah_token_unik: 6
Masukkan token: AA BB CC DD EE FF
ukuran_buffer: 5
ukuran_matriks: 9 4
jumlah_sekuens: 3
ukuran_maksimal_sekuens: 4

```

Output:

```

BB AA CC EE
DD BB FF DD
EE DD FF CC
EE CC FF DD
DD EE FF BB
AA BB EE AA
AA CC CC BB
AA FF EE CC
AA FF BB FF
=====Sequences=====
20 , DD DD
7 , CC BB
4 , AA CC EE
=====
=====FINAL RESULT=====
Coords '[COL, ROW]' : [[1, 2], [1, 5], [2, 5], [2, 7], [4, 7]]
Sequence: [DD, DD, EE, CC, BB]
Total Points: 27
Execution Time: 12 ms
Apakah hasil output mau di save? (y/n) :
y
Ketik nama file anda (tanpa .txt) : output1
Output berhasil disimpan di.../test/output/output1.txt

```

output1.txt

BB	AA	CC	EE
DD	BB	FF	DD
EE	DD	FF	CC
EE	CC	FF	DD
DD	EE	FF	BB
AA	BB	EE	AA
AA	CC	CC	BB
AA	FF	EE	CC

AA FF BB FF

=====

20 Points | DD DD |

7 Points | CC BB |

4 Points | AA CC EE |

=====

Coords '[COL, ROW]' : [[1, 2], [1, 5], [2, 5], [2, 7], [4, 7]]

Sequence: [DD, DD, EE, CC, BB]

Total Points: 27

Execute Time : 12 ms

e. Percobaan ke-5 (CLI)

Input:

```
1. TXT:
2. COMMAND LINE:
3. EXIT
Pilih Input : 2
Masukkan jumlah_token_unik: 6
Masukkan token: AD YF 7D 00 3F 5G
ukuran_buffer: 8
ukuran_matriks: 8 6
jumlah_sekuens: 5
ukuran_maksimal_sekuens: 4
```

Output:

```

5G 5G 00 5G 5G 3F
AD YF YF YF YF 3F
7D AD 3F 00 3F 00
3F YF AD YF YF 00
3F 3F 7D AD YF 3F
YF 7D 3F YF 5G 7D
7D YF 00 7D 00 3F
00 7D 00 5G 5G 7D
=====Sequences=====
20 , AD 00 AD
13 , 7D YF
8 , 5G 00
1 , 00 5G
13 , 00 7D
=====
=====FINAL RESULT=====
Coords '[COL, ROW]' : [[1, 1], [1, 8], [2, 8], [2, 3], [4, 3], [4, 5], [3, 5], [3, 2]]
Sequence: [5G, 00, 7D, AD, 00, AD, 7D, YF]
Total Points: 54
Execution Time: 1376 ms
Apakah hasil output mau di save? (y/n) :
y
Ketik nama file anda (tanpa .txt) : output2
Output berhasil disimpan di.../test/output/output2.txt

```

output2.txt

5G	5G	00	5G	5G	3F
AD	YF	YF	YF	YF	3F
7D	AD	3F	00	3F	00
3F	YF	AD	YF	YF	00
3F	3F	7D	AD	YF	3F
YF	7D	3F	YF	5G	7D
7D	YF	00	7D	00	3F
00	7D	00	5G	5G	7D
=====					
20 Points AD 00 AD					
13 Points 7D YF					
8 Points 5G 00					
1 Points 00 5G					
13 Points 00 7D					
=====					

Coords '[COL, ROW]' : [[1, 1], [1, 8], [2, 8], [2, 3], [4, 3], [4, 5], [3, 5], [3, 2]]

Sequence: [5G, 00, 7D, AD, 00, AD, 7D, YF]

Total Points: 54

Execute Time : 1376 ms

f. Percobaan ke-6 (CLI)

Input:

```
1. TXT:
2. COMMAND LINE:
3. EXIT
Pilih Input : 2
Masukkan jumlah_token_unik: 8
Masukkan token: 9F R0 PO CF SF ER FA 2F 1G 7C FF
Jumlah Tidak valid! Masukkan token kembali : 9F R0 PO CF SF ER FA 2F 1G 7C
Jumlah Tidak valid! Masukkan token kembali : 9F R0 PO CF SF ER FA 2F
ukuran_buffer: 8
ukuran_matriks: 10 4 5
Input tidak valid! Ketik kembali ukuran matriks (M N): 10 10
jumlah_sekuens: 8
ukuran_maksimal_sekuens: 8
```

Output:

```
CF FA 9F SF SF SF SF CF SF CF
CF R0 PO CF R0 2F CF CF 2F ER
R0 PO CF 2F 2F CF 2F PO FA FA
FA 2F ER 2F PO PO PO SF 2F CF
FA R0 SF PO PO R0 CF CF ER ER
CF 9F 9F R0 2F 2F CF R0 FA FA
PO SF 2F ER SF CF SF R0 9F 9F
FA SF 2F FA FA 2F CF R0 9F FA
SF SF 2F R0 FA 9F R0 PO 9F 9F
2F CF ER FA FA FA ER SF R0
=====Sequences=====
6 , R0 2F R0 R0
15 , ER 2F FA FA ER FA
14 , SF PO
28 , FA
1 , R0 PO
13 , 9F ER FA PO FA
24 , R0 R0 FA PO R0 FA SF
28 , CF 2F CF R0 PO SF
=====
=====FINAL RESULT=====
Coords '[COL, ROW]' : [[1, 2], [1, 10], [2, 10], [2, 5], [5, 5], [5, 7], [1, 7], [1, 8]]
Sequence: [CF, 2F, CF, R0, PO, SF, PO, FA]
Total Points: 71
Execution Time: 22520 ms
Apakah hasil output mau di save? (y/n) :
y
Ketik nama file anda (tanpa .txt) : output3
Output berhasil disimpan di.../test/output/output3.txt
```

output3.txt

CF	FA	9F	SF	SF	SF	SF	CF	SF	CF
CF	R0	PO	CF	R0	2F	CF	CF	2F	ER
R0	PO	CF	2F	2F	CF	2F	PO	FA	FA
FA	2F	ER	2F	PO	PO	PO	SF	2F	CF
FA	R0	SF	PO	PO	R0	CF	CF	ER	ER
CF	9F	9F	R0	2F	2F	CF	R0	FA	FA
PO	SF	2F	ER	SF	CF	SF	R0	9F	9F
FA	SF	2F	FA	FA	2F	CF	R0	9F	FA
SF	SF	2F	R0	FA	9F	R0	PO	9F	9F
2F	CF	ER	FA	FA	FA	FA	ER	SF	R0

=====

6 Points | R0 2F R0 R0 |

15 Points | ER 2F FA FA ER FA |

14 Points | SF PO |

28 Points | FA |

1 Points | R0 PO |

13 Points | 9F ER FA PO FA |

24 Points | R0 R0 FA PO R0 FA SF |

28 Points | CF 2F CF R0 PO SF |

<p>Coords '[COL, ROW]' : [[1, 2], [1, 10], [2, 10], [2, 5], [5, 5], [5, 7], [1, 7], [1, 8]]</p> <p>Sequence: [CF, 2F, CF, R0, PO, SF, PO, FA]</p> <p>Total Points: 71</p> <p>Execute Time : 22520 ms</p>
--

7. Lampiran

a. Tabel Checklist

Poin	Ya	Tidak
Program berhasil dikompilasi tanpa kesalahan	✓	
Program berhasil dijalankan	✓	
Program dapat membaca masukan berkas .txt	✓	
Program dapat menghasilkan masukan secara acak	✓	
Solusi yang diberikan program optimal	✓	
Program dapat menyimpan solusi dalam berkas .txt	✓	
Program memiliki GUI		✓

b. Pranala *Repository* Program

<https://github.com/miannetopokki/Tucil1Stima-2024>