LAPORAN TUGAS KECIL I

IF2211 STRATEGI ALGORITMA

Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force

Disusun oleh:

Marvin Scifo Y. Hutahaean 13522110

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2024

Daftar Isi

BAB I
PENJELASAN DASAR

A. Algoritma *Brute Force*

Algoritma *Brute Force*, adalah salah satu pendekatan algoritma yang pendekatannya lempang (*straightforward*) dalam sebuah penyelesaian persoalan. Algoritma ini didasarkan kepada pernyataan pada persoalan (*problem statement*) dan definisi konsep yang dilibatkan. *Brute Force* memecahkan persoalan dengan sangat sederhana, langsung, dan jelas. Biasanya *Brute Force* melibatkan semua informasi yang ada pada sebuah data. Adapun contoh-contoh persoalan yang menggunakan algoritma *Brute Force* adalah

1. Mencari elemen terbesar atau terkecil dengan membandingkan suatu elemen dengan semua elemen yang ada
2. Pencarian beruntun dengan membandingkan nilai yang ingin kit acari dalam sebuah list dengan semua elemen yang ada dalam list tersebut

B. Aplikasi Algoritma *Brute Force terhadap Cyberpunk 2077 Breach Protocol*

Pada tugas kecil ke-1 dari IF2211 Strategi Algoritma, penulis ditugaskan untuk membuat simulasi dari sebuah *minigame* dari Cyberpunk 2077 yang bernama Breach Protocol. Dari permainan ini, pemain diminta untuk mencari kombinasi alfanumerik yang identik dengan solusi alfanumerik yang ada untuk mendapatkan poin. Lebih banyak solusi yang didapat, lebih banyak poin yang didapatkan.

Cyberpunk 2077 Breach Protocol bisa dibuat simulasinya dengan menggunakan Algoritma *Brute Force*. Ini artinya program akan mencari semua kombinasi alfanumerik yang mungkin dari matriks yang disediakan dan dicocokan dengan solusi alfanumerik-nya untuk dicari nilainya. Program yang dibuat akan memberikan solusi yang paling optimal yaitu yang kombinasi alfanumeriknya paling pendek dan poinnya yang paling besar. Beginilah penjelasan singkat akan aplikasi algoritma *Brute Force* terhadap Cyberpunk 2077 Breach Protocol.

1. Program akan mulai dengan memproses elemen yang paling awal lalu ditambahkan ke list dalam bentuk koordinat. Lalu program akan memproses elemen selanjutnya secara vertikal. Elemen tersebut akan dimasukkan ke list berisi koordinat lalu program akan memproses elemen selanjutnya secara horizontal. Perlu diketahui bahwa jika terdapat koordinat yang sama dengan yang ada list, elemen tersebut akan dihindari dan program akan lanjut melakukan iterasi ke elemen yang lain.
2. Jika list sudah penuh dan elemen terakhir sudah mencapai ambang matriks (misalnya elemen 6,6), program akan menghapus elemen terakhir tersebut dan jika masih ada elemen terakhir yang mencapai ambang, program akan menghapusnya lagi. Setelah itu, program akan melakukan *backtrack* untuk mencari kombinasi lain dan iterasi yang baru akan dimulai.
3. Semua kemungkinan list yang ada akan dimasukkan ke dalam sebuah list (list berisi list). Setelah semuanya selesai, list berisi list tersebut akan berisi semua kemungkinan

list koordinat yang akan dikonversi menjadi token di matriks (koordinatnya (1,1) maka akan dikonversi menjadi token elemen (0,0).

4. List akan diproses dan dicari solusi yang paling optimal untuk diberi informasinya ke terminal.

BAB II
SOURCE CODE

Tucil ini diimplementasikan dengan menggunakan Bahasa pemrograman Java dengan compiler versi 20+. Berikut adalah fungsi dan prosedur yang diimplementasikan dalam program ini.

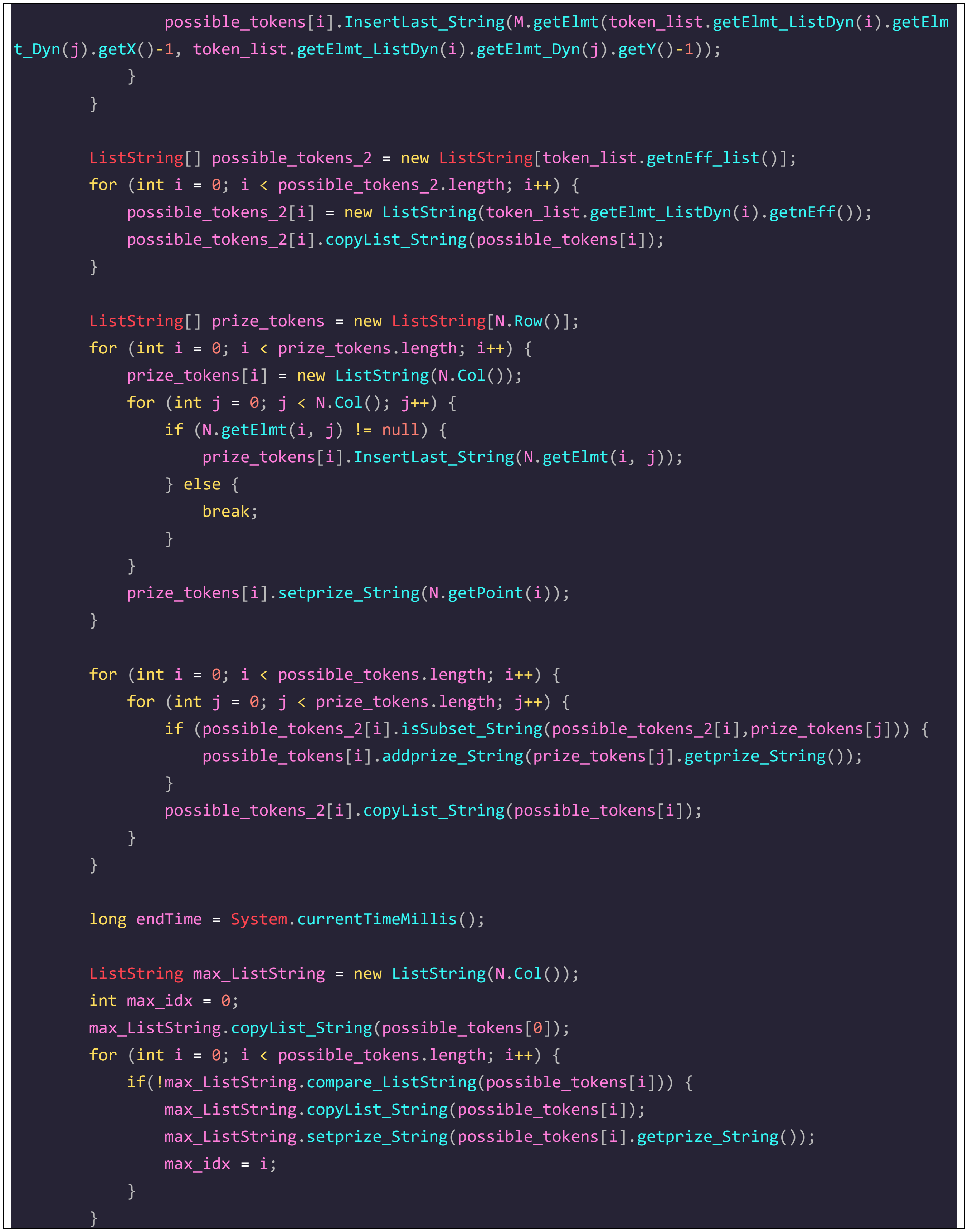
1. first_method()
2. second_method()
3. main(String[] args)
4. Read(String path, Matrix M, Matrix N)
5. Write(ListString liststring, int prize, ListDyn listcoor)
6. ListDyn(int size)
7. getnEff()
8. getsize()
9. getprize()
10. setprize(int val)
11. getElmt_Dyn(int index)
12. setElmt_Dyn(int index, Point point)
13. InsertLast_Dyn(Point point)
14. IsIn_Dyn(Point point)
15. DeleteLast_Dyn()
16. DisplayList_Dyn()
17. DisplayList_Dyn_Resultado()
18. copyList_Dyn(List_Dyn list)
19. ListListDyn(int size)
20. getnEff_list()
21. getsize_list()
22. setElmt_ListDyn(int index, ListDyn liste)
23. InsertLast_ListDyn(ListDyn list)
24. IsIn_ListDyn(ListDyn list)
25. DeleteLast_ListDyn()
26. DisplayList_ListDyn()
27. getElmt_ListDyn(int n)
28. ListString(int size)
29. getnEff_String()
30. getsize_String()
31. getprize_String()
32. setprize_String(int val)
33. addprize_String(int val)
34. getElmt_String(int index)
35. setElmt_String(int index, String string)
36. InsertLast_String(String string)

37. IsIn_String(String string)
38. DeleteLast_String()
39. DisplayList_String()
40. copyLIst_String()
41. slice_ListString(ListString list)
42. isEqual_String_Beginning(ListString list)
43. isSubset_String(ListString list1, ListString list2)
44. compare_ListString(ListString list2)
45. Matrix(int row, int col)
46. Row()
47. Col()
48. setRow(int newRow)
49. setCol(int newCol)
50. getElmt(int row, int col)
51. setElmt(int row, int col, String val)
52. getPoint(int row)
53. setPoint(int row, int rep)
54. setPointSize(int size)
55. fillEmpty()
56. displayMatrix()
57. displayCoordinate()
58. game_vertical(int col, ListDyn coor_list, ListListDyn token_list)
59. game_horizontal(int col, ListDyn coor_list, ListListDyn token_list)
60. toString(char[] array)
61. generateToken()
62. generateMatrix(int row, int col, String[] strings)
63. Point()
64. Point(int x, int y)
65. createPoint(int x, int y)
66. getX()
67. getY()
68. setX(int x)
69. setY(int y)
70. displayPoint()

Berikut ini adalah source code tucil ini.

Nama file: Driver.java

```
import java.util.*;
import java.io.*;
import java.lang.Math;

public class Driver {
    public static final String black = "\u001B[30m";
    public static final String red = "\u001B[31m";
    public static final String green = "\u001B[32m";
    public static final String yellow = "\u001B[33m";
    public static final String blue = "\u001B[34m";
    public static final String purple = "\u001B[35m";
    public static final String cyan = "\u001B[36m";
    public static final String white = "\u001B[37m";
    public static final String reset = "\u001B[0m";
    static int token_count = 0;
    static Matrix newMatrix = new Matrix(1,1);

    public static void first_method() {
        System.out.println("WARNING: File yang diambil adalah dari folder test (test.txt, maka 
test/test.txt akan diproses)!");
        System.out.print("Masukkan nama file untuk diproses (Format adalah *.txt): ");
        Scanner file_name_inpute = new Scanner(System.in);
        String file_name = file_name_inpute.nextLine();
        Matrix M = new Matrix(1000,1000);
        Matrix N = new Matrix(1000,1000);
        FileProcess.Read(file_name,M,N);
        ListDyn coor_liste = new ListDyn(N.Col());
        ListListDyn token_list = new ListListDyn(5);
        long startTime = System.currentTimeMillis();

        try {
            for (int i = 0; i < 1; i++) {
                M.game_horizontal(i, coor_liste, token_list);
            }
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.print("");
            return;
        }

        ListString[] possible_tokens = new ListString[token_list.getnEff_list()];
        for (int i = 0; i < possible_tokens.length; i++) {
            possible_tokens[i] = new ListString(token_list.getElmt_ListDyn(i).getnEff());
            for (int j = 0; j < possible_tokens[i].getsize_String(); j++) {
```



```
                possible_tokens[i].InsertLast_String(M.getElmt(token_list.getElmt_ListDyn(i).getElm
t_Dyn(j).getX()-1, token_list.getElmt_ListDyn(i).getElmt_Dyn(j).getY()-1));
            }
        }

        ListString[] possible_tokens_2 = new ListString[token_list.getnEff_list()];
        for (int i = 0; i < possible_tokens_2.length; i++) {
            possible_tokens_2[i] = new ListString(token_list.getElmt_ListDyn(i).getnEff());
            possible_tokens_2[i].copyList_String(possible_tokens[i]);
        }

        ListString[] prize_tokens = new ListString[N.Row()];
        for (int i = 0; i < prize_tokens.length; i++) {
            prize_tokens[i] = new ListString(N.Col());
            for (int j = 0; j < N.Col(); j++) {
                if (N.getElmt(i, j) != null) {
                    prize_tokens[i].InsertLast_String(N.getElmt(i, j));
                } else {
                    break;
                }
            }
            prize_tokens[i].setprize_String(N.getPoint(i));
        }

        for (int i = 0; i < possible_tokens.length; i++) {
            for (int j = 0; j < prize_tokens.length; j++) {
                if (possible_tokens_2[i].isSubset_String(possible_tokens_2[i],prize_tokens[j])) {
                    possible_tokens[i].addprize_String(prize_tokens[j].getprize_String());
                }
                possible_tokens_2[i].copyList_String(possible_tokens[i]);
            }
        }

        long endTime = System.currentTimeMillis();

        ListString max_ListString = new ListString(N.Col());
        int max_idx = 0;
        max_ListString.copyList_String(possible_tokens[0]);
        for (int i = 0; i < possible_tokens.length; i++) {
            if(!max_ListString.compare_ListString(possible_tokens[i])) {
                max_ListString.copyList_String(possible_tokens[i]);
                max_ListString.setprize_String(possible_tokens[i].getprize_String());
                max_idx = i;
            }
        }
        if (possible_tokens[max_idx].getprize_String() == 0) {
            System.out.print("Tidak ada poin paling optimal. Maka hasilnya adalah 0: ");
            System.out.print(possible_tokens[max_idx].getprize_String());
            System.out.println();
        } else {
            System.out.print("Sekuens Optimal: ");
            possible_tokens[max_idx].DisplayList_String();
            System.out.println();
            System.out.print("Poin Optimal: ");
            System.out.print(possible_tokens[max_idx].getprize_String());
            System.out.println();
            System.out.println("Koordinat Optimal: ");
            token_list.getElmt_ListDyn(max_idx).DisplayList_Dyn_Resultado();
            System.out.println();
        }

        long duration = endTime-startTime;
        System.out.print(duration);
        System.out.println(" ms");

        FileProcess.Write(max_ListString, possible_tokens[max_idx].getprize_String(),
token_list.getElmt_ListDyn(max_idx), duration);
    }

    public static void second_method() {
        Random r = new Random();

        System.out.print("Masukkan jumlah token unik: ");
        int unique_token = (new Scanner(System.in)).nextInt();
        String[] token = new String[unique_token];

        String[] token_inpute;
        while (true) {
            System.out.print("Masukkan token-token buffer: ");
            token_inpute = (new Scanner(System.in)).nextLine().trim().split(" ");
            int count = 0;
            for (int i = 0; i < token_inpute.length; i++) {
                if (token_inpute[i].length() == 2) {
                    count += 1;
                }
            }
            if (count == token_inpute.length) {
                break;
            } else {
                System.out.println("Jumlah digit setiap token harus 2!");
            }
        }

        for (int i = 0; i < unique_token; i++) {
            token[i] = token_inpute[i];
        }

        System.out.print("Masukkan ukuran buffer: ");
        int buffer_size = (new Scanner(System.in)).nextInt();

        System.out.print("Masukkan ukuran matriks: ");
        String[] matrix_row_col = (new Scanner(System.in)).nextLine().trim().split(" ");
        int matrix_row_size = Integer.parseInt(matrix_row_col[1]);
        int matrix_col_size = Integer.parseInt(matrix_row_col[0]);

        System.out.print("Masukkan jumlah sekuens: ");
        int jumlah_sekuens = (new Scanner(System.in)).nextInt();

        System.out.print("Masukkan ukuran maksimal sekuens: ");
        int ukuran_maksimal_sekuens = (new Scanner(System.in)).nextInt();

        Matrix matrix_token = new Matrix(matrix_row_size, matrix_col_size);
        matrix_token.generateMatrix(matrix_row_size, matrix_col_size, token);
        matrix_token.displayMatrix();
        System.out.println();

        ListString[] prize_token = new ListString[jumlah_sekuens];
        int rand;
        for (int i = 0; i < jumlah_sekuens; i++) {
            rand = r.nextInt(ukuran_maksimal_sekuens-1)+2;
            prize_token[i] = new ListString(rand);
            while (true) {
                for (int j = 0; j < rand; j++) {
                    prize_token[i].InsertLast_String(token[r.nextInt(token.length)]);
                }
                int count = 0;
                for (int k = 0; k < prize_token.length; k++) {
                    if (prize_token[k] != null) {
                        if (prize_token[i].isEqual_String_Beginning(prize_token[i], prize_token[k])
&& i != k) {
                            count -= 1;
                        }
                    }
                }
                if (count == 0) {
                    break;
                } else {
                    prize_token[i] = new ListString(rand);
                }
            }
            prize_token[i].setprize_String(r.nextInt(1001));
            int rando = r.nextInt(3);
            if (rando == 2) {
                prize_token[i].setprize_String(-1*prize_token[i].getprize_String());
            }
            prize_token[i].DisplayList_String();
            System.out.println();
            System.out.print(prize_token[i].getprize_String());
            System.out.println();
        }

        ListDyn coor_liste = new ListDyn(buffer_size);
        ListListDyn token_list = new ListListDyn(5);

        matrix_token.setCol(matrix_col_size);
        matrix_token.setRow(matrix_row_size);

        long startTime = System.currentTimeMillis();
        for (int i = 0; i < 1; i++) {
            matrix_token.game_horizontal(i, coor_liste, token_list);
        }

        ListString[] possible_tokens = new ListString[token_list.getnEff_list()];
        for (int i = 0; i < possible_tokens.length; i++) {
            possible_tokens[i] = new ListString(token_list.getElmt_ListDyn(i).getnEff());
            for (int j = 0; j < possible_tokens[i].getsize_String(); j++) {
                possible_tokens[i].InsertLast_String(matrix_token.getElmt(token_list.getElmt_ListDy
n(i).getElmt_Dyn(j).getX()-1, token_list.getElmt_ListDyn(i).getElmt_Dyn(j).getY()-1));
            }
        }

        ListString[] possible_tokens_2 = new ListString[token_list.getnEff_list()];
        for (int i = 0; i < possible_tokens_2.length; i++) {
            possible_tokens_2[i] = new ListString(token_list.getElmt_ListDyn(i).getnEff());
            possible_tokens_2[i].copyList_String(possible_tokens[i]);
        }

        for (int i = 0; i < possible_tokens.length; i++) {
            for (int j = 0; j < prize_token.length; j++) {
                if (possible_tokens_2[i].isSubset_String(possible_tokens_2[i],prize_token[j])) {
                    possible_tokens[i].addprize_String(prize_token[j].getprize_String());
                }
                possible_tokens_2[i].copyList_String(possible_tokens[i]);
            }
        }

        long endTime = System.currentTimeMillis();

        ListString max_ListString = new ListString(buffer_size);
        int max_idx = 0;
        max_ListString.copyList_String(possible_tokens[0]);
        for (int i = 0; i < possible_tokens.length; i++) {
            if(!max_ListString.compare_ListString(possible_tokens[i])) {
                max_ListString.copyList_String(possible_tokens[i]);
                max_ListString.setprize_String(possible_tokens[i].getprize_String());
                max_idx = i;
            }
        }

        if (possible_tokens[max_idx].getprize_String() == 0) {
            System.out.print("Tidak ada poin paling optimal. Maka hasilnya adalah 0: ");
            System.out.print(possible_tokens[max_idx].getprize_String());
            System.out.println();
        } else {
            System.out.print("Sekuens Optimal: ");
            possible_tokens[max_idx].DisplayList_String();
            System.out.println();
            System.out.print("Poin Optimal: ");
            System.out.print(possible_tokens[max_idx].getprize_String());
            System.out.println();
            System.out.println("Koordinat Optimal: ");
            token_list.getElmt_ListDyn(max_idx).DisplayList_Dyn_Resultado();
            System.out.println();
        }

        long duration = endTime-startTime;
        System.out.print(duration);
        System.out.println(" ms");

        FileProcess.Write(max_ListString, possible_tokens[max_idx].getprize_String(),
token_list.getElmt_ListDyn(max_idx), duration);
    }

    // Main
```

```java
    public static void main(String[] args) {
        while (true) {
            System. out. print("\033[H\033[2J");
            System.out.flush();
            System.out.println("=====================================");
            System.out.println("|| Masukkan metode yang diinginkan ||");
            System.out.println("=====================================");
            System.out.println("|| 1. Matrix From File             ||");
            System.out.println("|| 2. Random Generated Matrix      ||");
            System.out.println("|| Other: Exit                     ||");
            System.out.println("=====================================");
            System.out.println();
            System.out.print("Input: ");

            int method;
            try {
                method = (new Scanner(System.in)).nextInt();
            } catch (InputMismatchException e) {
                System. out. print("\033[H\033[2J");
                System.out.flush();
                System.out.println("=====================================");
                System.out.println("||          Terima kasih!!         ||");
                System.out.println("=====================================");
                break;
            }

            if (method == 1) {
                first_method();
            } else if (method == 2) {
                second_method();
            } else {
                System. out. print("\033[H\033[2J");
                System.out.flush();
                System.out.println("=====================================");
                System.out.println("||          Terima kasih!!         ||");
                System.out.println("=====================================");
                break;
            }
        }
    }
}
```

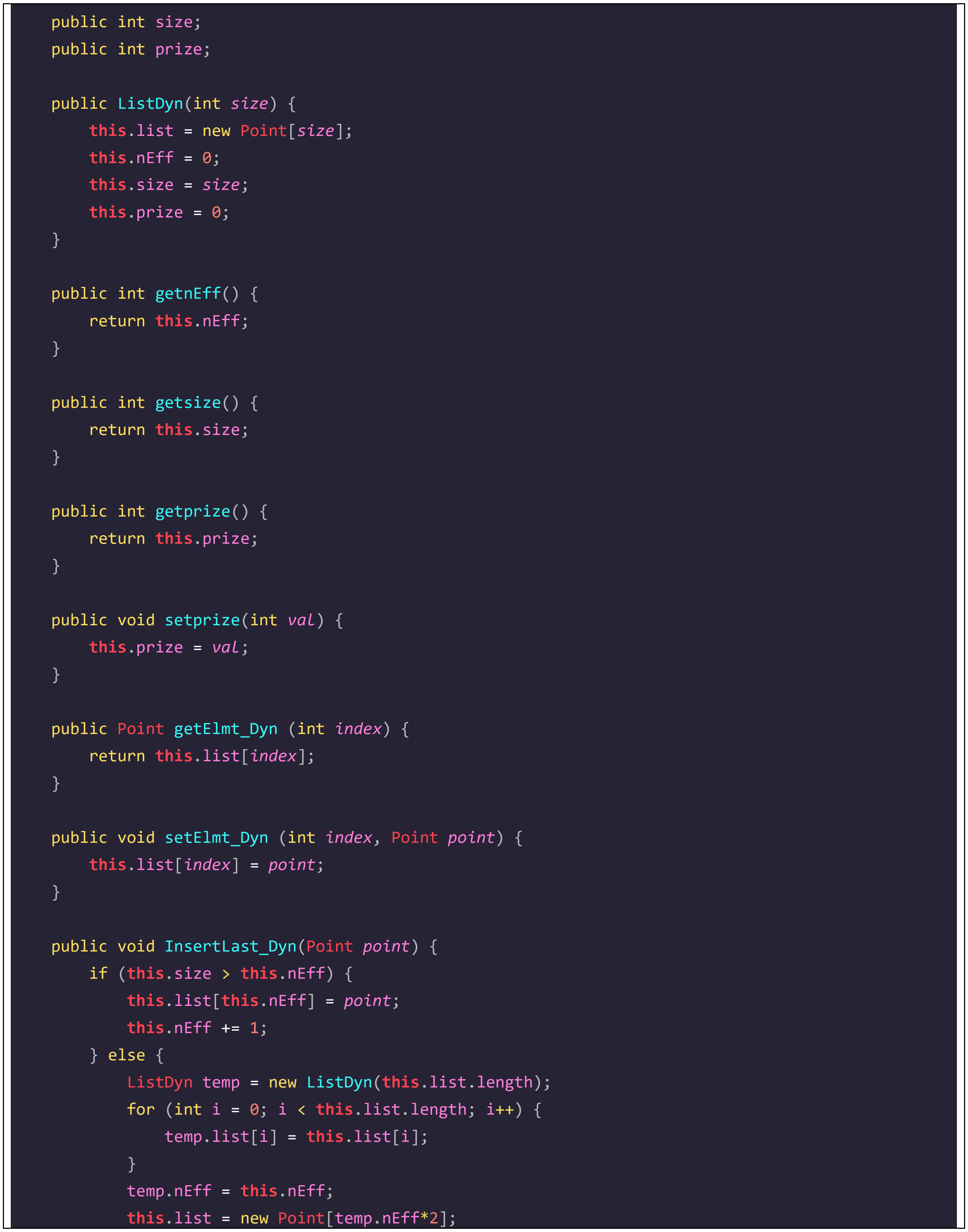
Nama file: FileProcess.java

```java
import java.io.*;
import java.util.*;
```

```
public class FileProcess {
    static Scanner scInt = new Scanner(System.in);
    static Scanner scFlt = new Scanner(System.in);
    static Scanner scStr = new Scanner(System.in);

    public static void Read(String path, Matrix M, Matrix N){
            try{
                String test = "../test/";
                File f = new File(test.concat(path));
                Scanner reader = new Scanner(f);
                if (reader.hasNextLine()) {
                    String buffer_size_string = reader.nextLine();
                    int buffer_size = Integer.parseInt(buffer_size_string);
                    N.setCol(buffer_size);
                }
                if (reader.hasNextLine()) {
                    String[] matrix_size_string = reader.nextLine().trim().split(" ");
                    int matrix_row = Integer.parseInt(matrix_size_string[0]);
                    int matrix_col = Integer.parseInt(matrix_size_string[1]);
                    M.setCol(matrix_row);
                    M.setRow(matrix_col);
                }

                for (int i = 0; i < M.Row(); i++) {
                    if (reader.hasNextLine()) {
                        String[] alpha_numeric_string = reader.nextLine().trim().split(" ");
                        for(int j = 0; j < M.Col(); j++) {
                            M.setElmt(i,j,alpha_numeric_string[j]);
                        }
                        for (int k = 0; k < M.Row(); k++) {
                            for (int l = 0; l < M.Col(); l++) {
                                if (M.getElmt(k, l) != null) {
                                    if (M.getElmt(k, l).length() != 2) {
                                        System.out.println("Terdapat elemen berdigit tidak sama
dengan 2! Menutup program...");
                                        System.exit(0);
                                    }
                                }
                            }
                        }
                    }
                }
                if (reader.hasNextLine()) {
                    String point_size_string = reader.nextLine();
                    int point_size = Integer.parseInt(point_size_string);
                    N.setRow(point_size);
                    N.setPointSize(point_size);
                }
                for (int i = 0; i < N.Row(); i++) {
                    if (reader.hasNextLine()) {
                        String[] buffer_token_string = reader.nextLine().trim().split(" ");
                        for (int j = 0; j < buffer_token_string.length; j++) {
                            N.setElmt(i,j,buffer_token_string[j]);
                        }
                        if (reader.hasNextLine()) {
                            N.setPoint(i, Integer.parseInt(reader.nextLine()));
                        }
                        for (int k = 0; k < N.Row(); k++) {
                            for (int l = 0; l < N.Col(); l++) {
                                if (N.getElmt(k, l) != null) {
                                    if (N.getElmt(k, l).length() != 2) {
                                        System.out.println("Terdapat elemen berdigit tidak sama 
dengan 2! Menutup program...");
                                        System.exit(0);
                                    }
                                }
                            }
                        }
                    }
                }
            } catch (FileNotFoundException e) {
                System.out.println("Nama file tidak ditemukan!");
            }
        }

    public static void Write(ListString liststring, int prize, ListDyn listcoor, long time) {
        System.out.println();
        System.out.println("====================================");
        System.out.println("||    Simpan output dalam file?   ||");
        System.out.println("||--------------------------------||");
        System.out.println("|| (Y/y) : YES                    ||");
        System.out.println("|| Other Response : NO            ||");
        System.out.println("====================================");
        System.out.println();
        try {
            char input = (new Scanner(System.in)).next().charAt(0);
            if (input == 'Y' || input == 'y') {
                System.out.print("Masukkan sebuah path: ");
                String path = scStr.nextLine();
                try {
                    String test = "../test/";
                    FileWriter writer = new FileWriter(test.concat(path));
                    for (int i = 0; i < liststring.getnEff_String(); i++) {
                        writer.write(liststring.getElmt_String(i));
                        if (i != liststring.getnEff_String()-1) {
                            writer.write(" ");
                        }
                    }
                    writer.write("\n");
                    writer.write(Integer.toString(prize));
                    writer.write("\n");
                    for (int i = 0; i < listcoor.getnEff(); i++) {
                        writer.write(Integer.toString(listcoor.getElmt_Dyn(i).getY()));
                        writer.write(",");
                        writer.write(Integer.toString(listcoor.getElmt_Dyn(i).getX()));
                        if (i != liststring.getnEff_String()-1) {
                            writer.write("\n");
                        }
                    }
                    writer.write("\n");
                    writer.write(Integer.toString((int)time));
                    writer.write(" ms");
                    writer.close();
                    System.out.println("Berhasil menulis ke file " + path);
                } catch (IOException e) {
                    System.out.println("Nama file tidak ditemukan!");
                }
            }
        } catch (InputMismatchException e) {
            char input = (new Scanner(System.in)).next().charAt(0);
        }



    }
}
```

Nama file: ListDyn.java

```
import java.util.*;
import java.io.File;
import java.lang.Math;

public class ListDyn {
    public Point[] list;
    public int nEff;
```



```java
    public int size;
    public int prize;

    public ListDyn(int size) {
        this.list = new Point[size];
        this.nEff = 0;
        this.size = size;
        this.prize = 0;
    }

    public int getnEff() {
        return this.nEff;
    }

    public int getsize() {
        return this.size;
    }

    public int getprize() {
        return this.prize;
    }

    public void setprize(int val) {
        this.prize = val;
    }

    public Point getElmt_Dyn (int index) {
        return this.list[index];
    }

    public void setElmt_Dyn (int index, Point point) {
        this.list[index] = point;
    }

    public void InsertLast_Dyn(Point point) {
        if (this.size > this.nEff) {
            this.list[this.nEff] = point;
            this.nEff += 1;
        } else {
            ListDyn temp = new ListDyn(this.list.length);
            for (int i = 0; i < this.list.length; i++) {
                temp.list[i] = this.list[i];
            }
            temp.nEff = this.nEff;
            this.list = new Point[temp.nEff*2];
            this.nEff = temp.nEff;
            this.size = temp.nEff*2;
            for (int i = 0; i < temp.nEff; i++) {
                this.list[i] = temp.list[i];
            }
            this.list[temp.nEff] = point;
            this.nEff += 1;
        }
    }

    public Boolean IsIn_Dyn(Point point) {
        Boolean bool = false;
        for (int i = 0; i < this.nEff; i++) {
            if (this.list[i] == point) {
                bool = true;
                break;
            }
        }
        return bool;
    }

    public void DeleteLast_Dyn() {
        this.nEff -= 1;
    }

    public void DisplayList_Dyn() {
        for (int i = 0; i < this.nEff; i++) {
            System.out.print("(");
            System.out.print(this.list[i].getX());
            System.out.print(",");
            System.out.print(this.list[i].getY());
            System.out.print(")");
            if (i != this.nEff-1) {
                System.out.print(",");
            }
        }
    }

    public void DisplayList_Dyn_Resultado() {
        for (int i = 0; i < this.nEff; i++) {
            System.out.print(this.list[i].getY());
            System.out.print(",");
            System.out.print(this.list[i].getX());
            if (i != this.nEff-1) {
                System.out.println();
            }
        }
    }

    public void copyList_Dyn(ListDyn list) {
        this.nEff = list.nEff;
        this.size = list.size;
        for (int i = 0; i < this.nEff; i++) {
            this.list[i] = list.list[i];
        }
    }
}
```

Nama file: ListListDyn.java

```
import java.util.*;
import java.io.File;
import java.lang.Math;

public class ListListDyn {
    private ListDyn[] buffer;
    private int nEff_list;
    private int size_list;

    public ListListDyn(int size) {
        this.buffer = new ListDyn[size];
        this.nEff_list = 0;
        this.size_list = size;
    }

    public int getnEff_list() {
        return this.nEff_list;
    }

    public int getsize_list() {
        return this.size_list;
    }

    public void setElmt_ListDyn(int index, ListDyn liste) {
        this.buffer[index] = new ListDyn(liste.getnEff());
        // liste.DisplayList_Dyn();
        // System.out.println();
        // for (int i = 0; i < this.buffer[index].getnEff(); i++) {
        //     this.buffer[index].list[i] = liste.list[i];
        // }
        this.buffer[index] = liste;
    }

    public void InsertLast_ListDyn(ListDyn list) {
        if (this.size_list > this.nEff_list) {
            this.setElmt_ListDyn(this.nEff_list, list);
            this.nEff_list += 1;
        } else {
            ListListDyn temp = new ListListDyn(this.buffer.length);
            for (int i = 0; i < this.buffer.length; i++) {
                temp.buffer[i] = this.buffer[i];
            }
            temp.nEff_list = this.nEff_list;
            this.buffer = new ListDyn[temp.nEff_list*2];
            this.nEff_list = temp.nEff_list;
            this.size_list = temp.nEff_list*2;
            for (int i = 0; i < temp.nEff_list; i++) {
                this.buffer[i] = temp.buffer[i];
            }
            this.setElmt_ListDyn(this.nEff_list, list);
            this.nEff_list += 1;
        }
    }

    public Boolean IsIn_ListDyn(ListDyn list) {
        Boolean bool = false;
        for (int i = 0; i < this.nEff_list; i++) {
            if (this.buffer[i] == list) {
                bool = true;
                break;
            }
        }
        return bool;
    }

    public void DeleteLast_ListDyn() {
        this.nEff_list -= 1;
    }

    public void DisplayList_ListDyn() {
        for (int i = 0; i < this.nEff_list; i++) {
            this.buffer[i].DisplayList_Dyn();
            if (i != this.nEff_list-1) {
                System.out.print(",");
            }
            System.out.println();
        }
    }

    public ListDyn getElmt_ListDyn(int n) {
        return this.buffer[n];
    }
}
```

Nama file: ListString.java

```
import java.util.*;
import java.io.File;
import java.lang.Math;

public class ListString {
    private String[] string_content;
    private int nEff_string;
    private int size_string;
    private int prize_string;

    public ListString(int size) {
        this.string_content = new String[size];
        this.nEff_string = 0;
        this.size_string = size;
        this.prize_string = 0;
    }

    public int getnEff_String() {
        return this.nEff_string;
    }

    public int getsize_String() {
        return this.size_string;
    }

    public int getprize_String() {
        return this.prize_string;
    }

    public void setprize_String(int val) {
        this.prize_string = val;
    }

    public void addprize_String(int val) {
```

```java
        this.prize_string += val;
    }

    public String getElmt_String(int index) {
        return this.string_content[index];
    }

    public void setElmt_String (int index, String string) {
        this.string_content[index] = string;
    }

    public void InsertLast_String (String string) {
        if (this.nEff_string < this.size_string) {
            this.setElmt_String(this.nEff_string, string);
            this.nEff_string += 1;
        } else {
            ListString temp = new ListString(this.size_string);
            for (int i = 0; i < this.nEff_string; i++) {
                temp.string_content[i] = this.string_content[i];
            }
            temp.nEff_string = this.nEff_string;
            this.string_content = new String[temp.size_string*2];
            this.nEff_string = temp.nEff_string;
            this.size_string = temp.size_string*2;
            for (int i = 0; i < temp.nEff_string; i++) {
                this.string_content[i] = temp.string_content[i];
            }
            this.string_content[temp.nEff_string] = string;
            this.nEff_string += 1;
        }
    }

    public Boolean IsIn_String(String string) {
        Boolean bool = false;
        for (int i = 0; i < this.nEff_string; i++) {
            if (this.string_content[i] == string) {
                bool = true;
                break;
            }
        }
        return bool;
    }

    public void DeleteLast_String() {
        this.nEff_string -= 1;
    }

    public void DisplayList_String() {
        for (int i = 0; i < this.nEff_string; i++) {
            System.out.print(this.string_content[i]);
            System.out.print(" ");
        }
    }

    public void copyList_String(ListString list) {
        this.nEff_string = list.nEff_string;
        this.size_string = list.size_string;
        for (int i = 0; i < this.nEff_string; i++) {
            this.string_content[i] = list.string_content[i];
        }
    }

    public ListString slice_ListString (ListString list) {
        for (int i = 0; i < list.getnEff_String()-1; i++) {
            list.setElmt_String(i, list.getElmt_String(i+1));
        }
        list.DeleteLast_String();
        return list;
    }

    public Boolean isEqual_String_Beginning(ListString list1, ListString list2) {
        Boolean bool = true;
        if (list1.nEff_string >= list2.nEff_string) {
            for (int i = 0; i < list2.nEff_string; i++) {
                if (list1.getElmt_String(i).equals(list2.getElmt_String(i)) == false) {
                    bool = false;
                    break;
                }
            }
        } else {
            bool = false;
        }
        return bool;
    }

    public Boolean isSubset_String(ListString list1, ListString list2) {
        if (list1.getnEff_String() == 0) {
            return false;
        } else {
            if (isEqual_String_Beginning(list1, list2)) {
```

```java
                return true;
            } else {
                return isSubset_String(slice_ListString(list1), list2);
            }
        }
    }

    public Boolean compare_ListString(ListString list2) {
        // System.out.print(this.prize_string);
        // System.out.print(" ");
        // System.out.print(list2.prize_string);
        // System.out.print(" ");
        if (this.prize_string > list2.prize_string) {
            // System.out.print("TRUE LENGTH");
            return true;
        } else if (this.prize_string == list2.prize_string) {
            if (this.nEff_string <= list2.nEff_string) {
                // System.out.print("TRUE LENGTH");
                return true;
            } else {
                // System.out.print("FALSE LENGTH");
                return false;
            }
        } else {
            // System.out.print("FALSE VALUE");
            return false;
        }
    }
}
```

Nama file: Matrix.java

```java
import java.util.*;
import java.io.File;
import java.lang.Math;

public class Matrix {
    private String[][] content;
    private Point[][] coordinate = new Point[1][1];
    private int row;
    private int col;
    private int[] points;

    public Matrix(int row, int col) {
        this.row = row;
        this.col = col;
        this.content = new String[row][col];
        this.coordinate = new Point[row][col];
        for (int i = 0; i < this.row; i++) {
            for (int j = 0; j < this.col; j++) {
                this.coordinate[i][j] = new Point(i+1,j+1);
            }
        }
    }

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

    public int Col() {
        return this.col;
    }

    public void setRow(int newRow) {
        this.row = newRow;
    }

    public void setCol(int newCol) {
        this.col = newCol;
    }

    public String getElmt(int row, int col) {
        return this.content[row][col];
    }

    public void setElmt(int row, int col, String val) {
        this.content[row][col] = val;
    }

    public int getPoint(int row) {
        return this.points[row];
    }

    public void setPoint(int row, int rep) {
        this.points[row] = rep;
    }

    public void setPointSize(int size) {
        this.points = new int[size];
    }
    public void fillEmpty() {
        for (int i = 0; i < this.row; i++) {
            for (int j = 0; j < this.col; j++) {
                this.content[i][j] = "--";
            }
        }
    }

    public void displayMatrix() {
        for (int i = 0; i < this.row; i++) {
            for (int j = 0; j < this.col; j++) {
                System.out.print(this.content[i][j]);
                System.out.print(" ");
            }
            System.out.println();
        }
    }

    public void displayCoordinate() {
        for (int i = 0; i < this.row; i++) {
            for (int j = 0; j < this.col; j++) {
                this.coordinate[i][j].displayPoint();
            }
            System.out.println();
        }
    }

    public void game_vertical(int col, ListDyn coor_list, ListListDyn token_list) {
        for (int i = 0; i < this.row; i++) {
            if (!coor_list.IsIn_Dyn(this.coordinate[i][col]) && coor_list.getnEff() <
coor_list.getsize()) {
                ListDyn new_list = new ListDyn(coor_list.getsize());
                ListDyn new_list_copy = new ListDyn(new_list.getsize());
                new_list.copyList_Dyn(coor_list);
                new_list.InsertLast_Dyn(this.coordinate[i][col]);
                new_list_copy.copyList_Dyn(new_list);
                token_list.InsertLast_ListDyn(new_list_copy);
                game_horizontal(i, new_list, token_list);
                new_list.DeleteLast_Dyn();
            } else {
                try {
                    if (!coor_list.IsIn_Dyn(this.coordinate[row][i]) && coor_list.getnEff() ==
coor_list.getsize()) {
                        break;
                    }
                } catch (ArrayIndexOutOfBoundsException e) {
                    // Nothing
                }
            }
        }
    }

    public void game_horizontal(int row, ListDyn coor_list, ListListDyn token_list) {
        for (int i = 0; i < this.col; i++) {
            if (!coor_list.IsIn_Dyn(this.coordinate[row][i]) && coor_list.getnEff() <
coor_list.getsize()) {
                ListDyn new_list = new ListDyn(coor_list.getsize());
                ListDyn new_list_copy = new ListDyn(new_list.getsize());
                new_list.copyList_Dyn(coor_list);
                new_list.InsertLast_Dyn(this.coordinate[row][i]);
                new_list_copy.copyList_Dyn(new_list);
                token_list.InsertLast_ListDyn(new_list_copy);
                game_vertical(i, new_list, token_list);
                new_list.DeleteLast_Dyn();
            } else {
                try {
                    if (!coor_list.IsIn_Dyn(this.coordinate[row][i]) && coor_list.getnEff() ==
coor_list.getsize()) {
                        break;
                    }
                } catch (ArrayIndexOutOfBoundsException e) {
                    // Nothing
                }
            }
        }
    }

    public String toString(char[] array) {
        String string = new String(array);
        return string;
    }

    public String generateToken() {
        Random r = new Random();
        char[] token = new char[2];
        String upper = "AZB0CYD9EXF1GWV8UHT2SIR7QJP3OKL6M4N5";
        for (int i = 0; i < token.length; i++) {
            token[i] = upper.charAt(r.nextInt(upper.length()));
        }
```

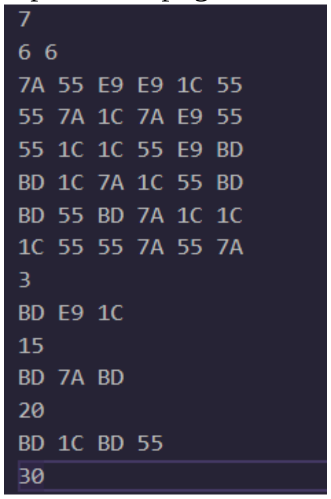
```java
        return toString(token);
    }

    public void generateMatrix(int row, int col, String[] strings) {
        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                this.content[i][j] = strings[(new Random()).nextInt(strings.length)];
            }
        }
    }
}
```

Nama file : Point.java

```java
import java.util.*;
import java.io.File;
import java.lang.Math;

public class Point {
    private int x;
    private int y;

    public Point() {
        this.x = 0;
        this.y = 0;
    }

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public void createPoint(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public int getX() {
        return this.x;
    }

    public int getY() {
        return this.y;
    }

    public void setX(int x) {
        this.x = x;
    }

    public void setY(int y) {
        this.y = y;
    }

    public void displayPoint() {
        System.out.print(this.x);
        System.out.print(",");
        System.out.print(this.y);
        System.out.print(" ");
    }
}
```

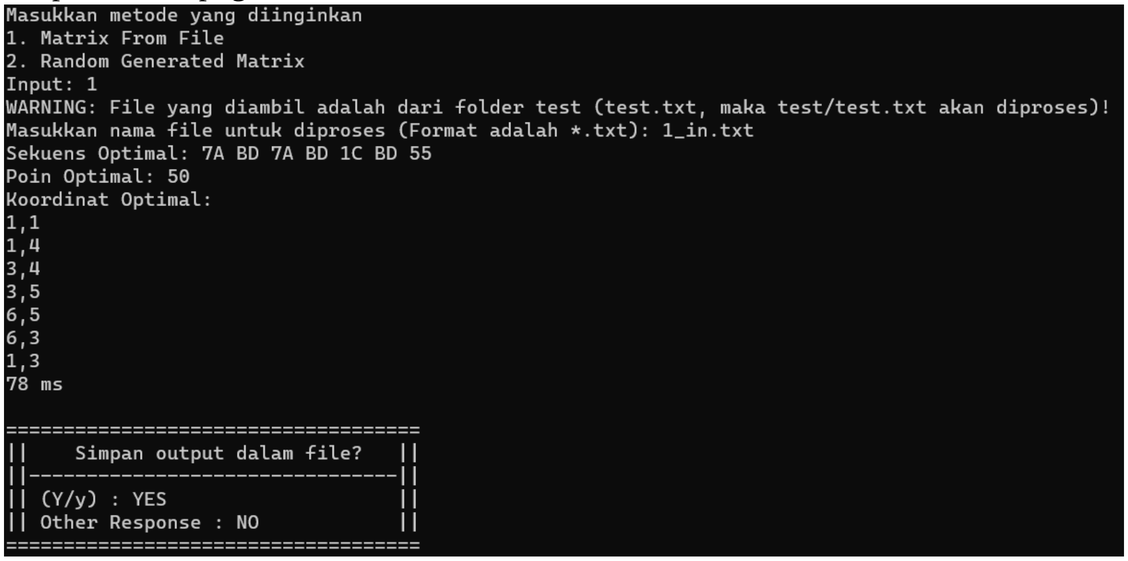
BAB III
SCREENSHOT HASIL TEST

1. Input : 1_in.png



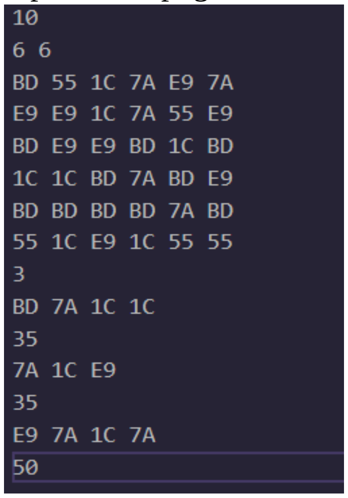
```
7
6 6
7A 55 E9 E9 1C 55
55 7A 1C 7A E9 55
55 1C 1C 55 E9 BD
BD 1C 7A 1C 55 BD
BD 55 BD 7A 1C 1C
1C 55 55 7A 55 7A
3
BD E9 1C
15
BD 7A BD
20
BD 1C BD 55
30
```

Output : 1_out.png



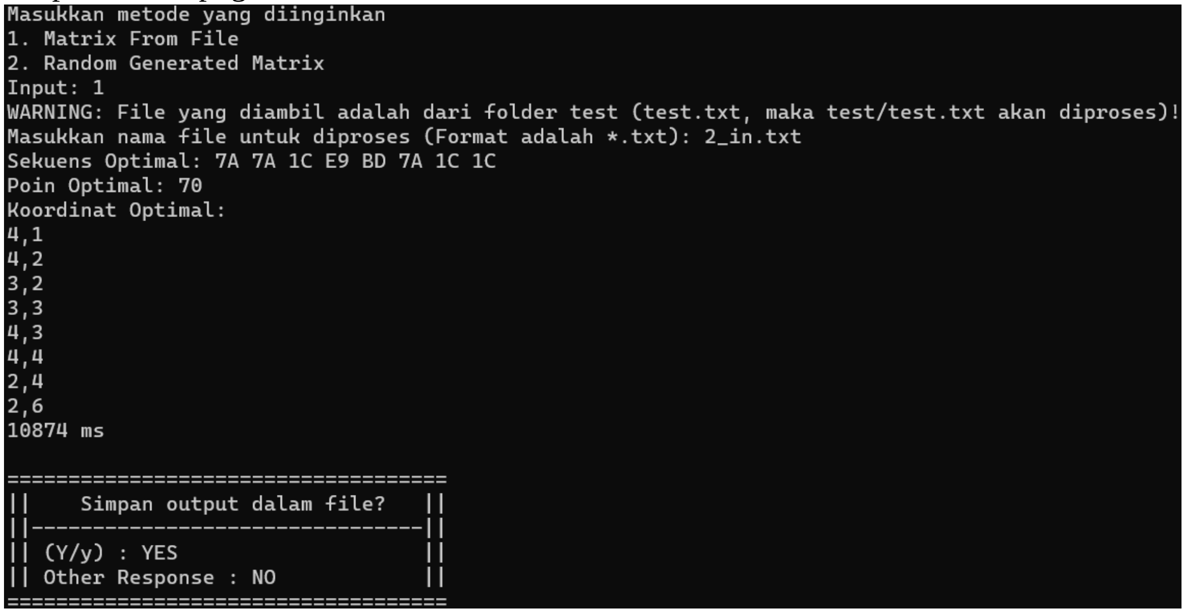
```
Masukkan metode yang diinginkan
1. Matrix From File
2. Random Generated Matrix
Input: 1
WARNING: File yang diambil adalah dari folder test (test.txt, maka test/test.txt akan diproses)!
Masukkan nama file untuk diproses (Format adalah *.txt): 1_in.txt
Sekuens Optimal: 7A BD 7A BD 1C BD 55
Poin Optimal: 50
Koordinat Optimal:
1,1
1,4
3,4
3,5
6,5
6,3
1,3
78 ms

====================================
||    Simpan output dalam file?   ||
||--------------------------------||
|| (Y/y) : YES                    ||
|| Other Response : NO            ||
====================================
```

2. Input : 2_in.png



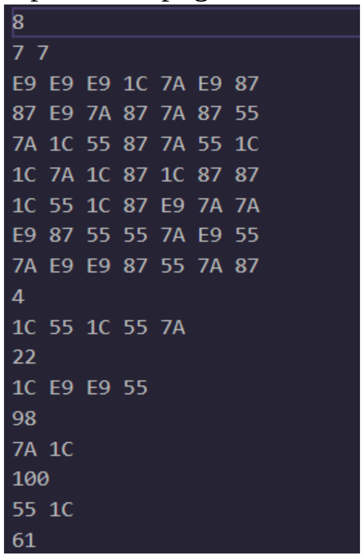
```
10
6 6
BD 55 1C 7A E9 7A
E9 E9 1C 7A 55 E9
BD E9 E9 BD 1C BD
1C 1C BD 7A BD E9
BD BD BD BD 7A BD
55 1C E9 1C 55 55
3
BD 7A 1C 1C
35
7A 1C E9
35
E9 7A 1C 7A
50
```

Output : 2_out.png



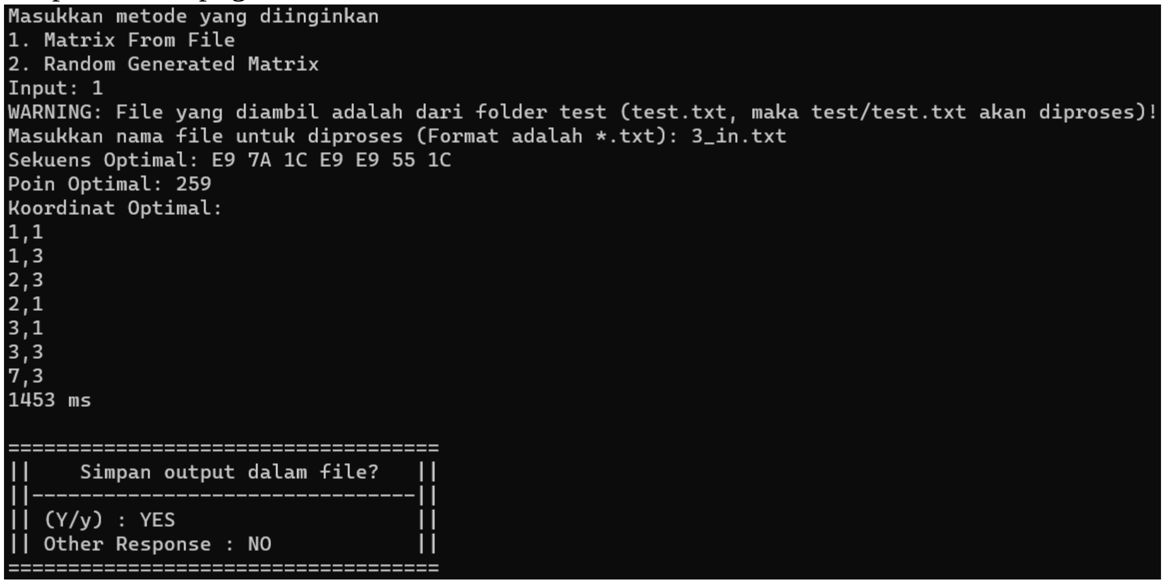
```
Masukkan metode yang diinginkan
1. Matrix From File
2. Random Generated Matrix
Input: 1
WARNING: File yang diambil adalah dari folder test (test.txt, maka test/test.txt akan diproses)!
Masukkan nama file untuk diproses (Format adalah *.txt): 2_in.txt
Sekuens Optimal: 7A 7A 1C E9 BD 7A 1C 1C
Poin Optimal: 70
Koordinat Optimal:
4,1
4,2
3,2
3,3
4,3
4,4
2,4
2,6
10874 ms

====================================
||    Simpan output dalam file?   ||
||--------------------------------||
|| (Y/y) : YES                    ||
|| Other Response : NO            ||
====================================
```

3. Input : 3_in.png



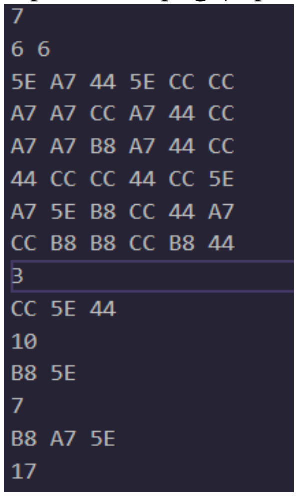
```
8
7 7
E9 E9 E9 1C 7A E9 87
87 E9 7A 87 7A 87 55
7A 1C 55 87 7A 55 1C
1C 7A 1C 87 1C 87 87
1C 55 1C 87 E9 7A 7A
E9 87 55 55 7A E9 55
7A E9 E9 87 55 7A 87
4
1C 55 1C 55 7A
22
1C E9 E9 55
98
7A 1C
100
55 1C
61
```

Output : 3_out.png



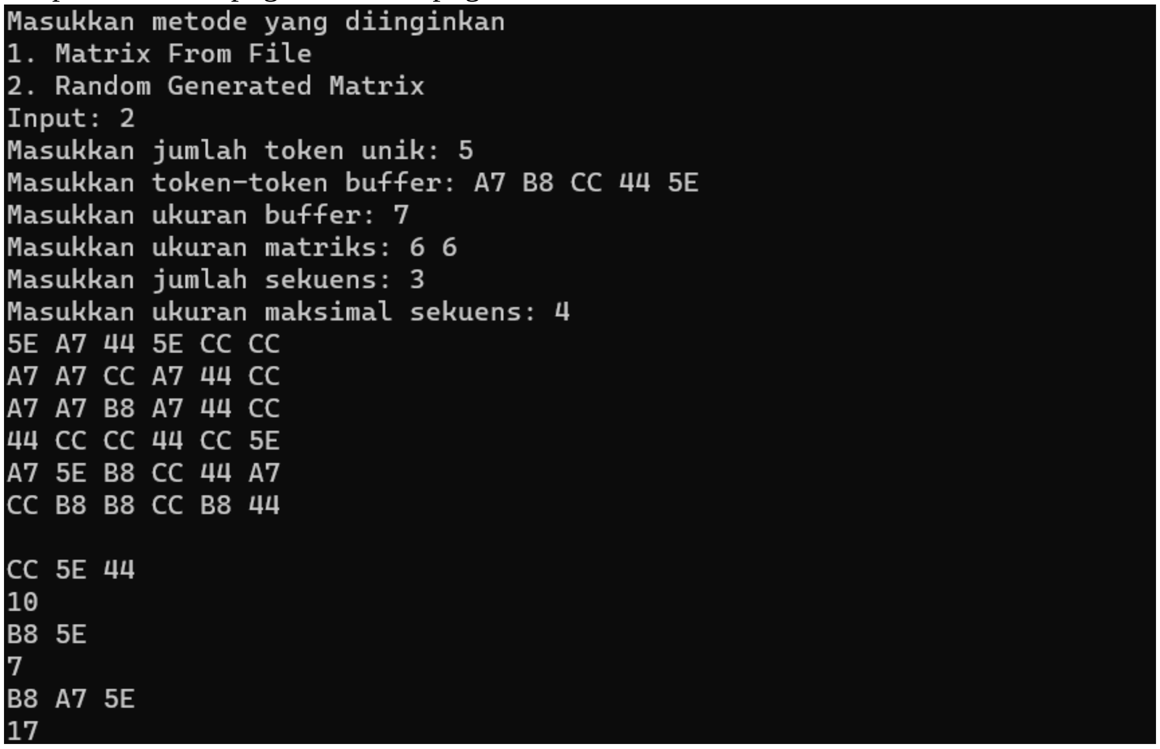
```
Masukkan metode yang diinginkan
1. Matrix From File
2. Random Generated Matrix
Input: 1
WARNING: File yang diambil adalah dari folder test (test.txt, maka test/test.txt akan diproses)!
Masukkan nama file untuk diproses (Format adalah *.txt): 3_in.txt
Sekuens Optimal: E9 7A 1C E9 E9 55 1C
Poin Optimal: 259
Koordinat Optimal:
1,1
1,3
2,3
2,1
3,1
3,3
7,3
1453 ms

====================================
||    Simpan output dalam file?   ||
||--------------------------------||
|| (Y/y) : YES                    ||
|| Other Response : NO            ||
====================================
```

4. Input : 4_in.png (Input ini juga digunakan untuk cek kesamaan 2 metode)

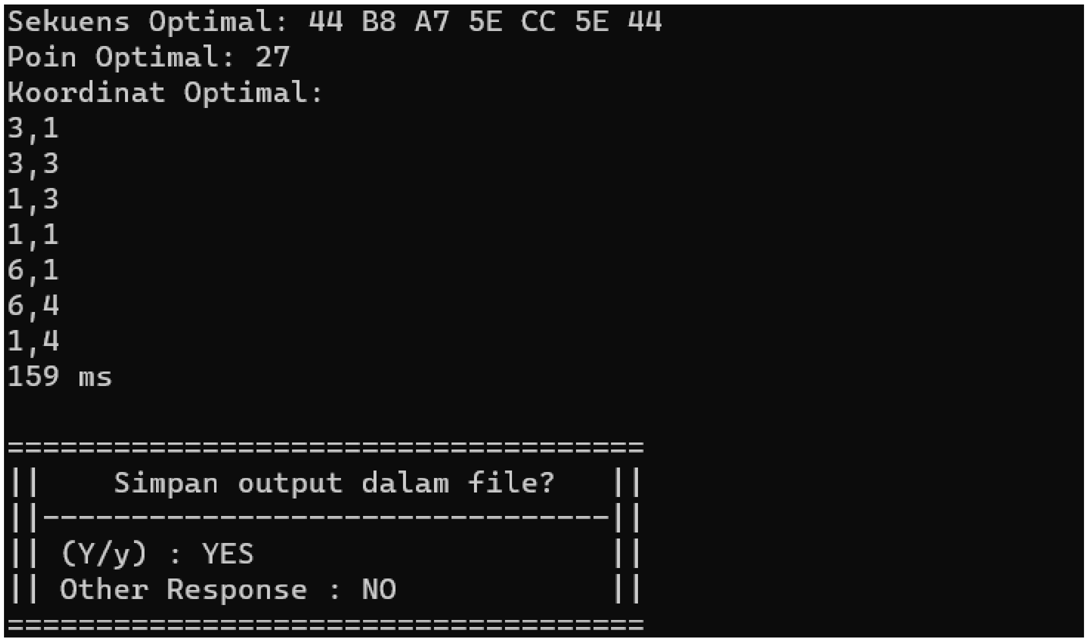


```
7
6 6
5E A7 44 5E CC CC
A7 A7 CC A7 44 CC
A7 A7 B8 A7 44 CC
44 CC CC 44 CC 5E
A7 5E B8 CC 44 A7
CC B8 B8 CC B8 44
3
CC 5E 44
10
B8 5E
7
B8 A7 5E
17
```

Output : 4_out_1.png & 4_out_2.png

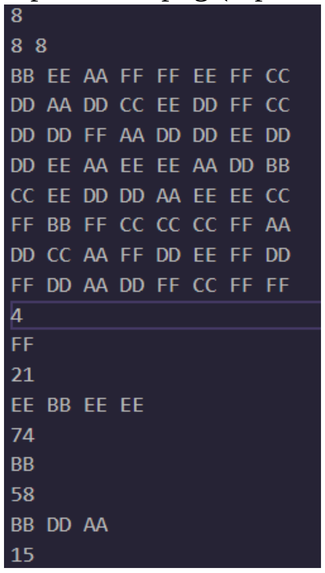


```
Masukkan metode yang diinginkan
1. Matrix From File
2. Random Generated Matrix
Input: 2
Masukkan jumlah token unik: 5
Masukkan token-token buffer: A7 B8 CC 44 5E
Masukkan ukuran buffer: 7
Masukkan ukuran matriks: 6 6
Masukkan jumlah sekuens: 3
Masukkan ukuran maksimal sekuens: 4
5E A7 44 5E CC CC
A7 A7 CC A7 44 CC
A7 A7 B8 A7 44 CC
44 CC CC 44 CC 5E
A7 5E B8 CC 44 A7
CC B8 B8 CC B8 44

CC 5E 44
10
B8 5E
7
B8 A7 5E
17
```



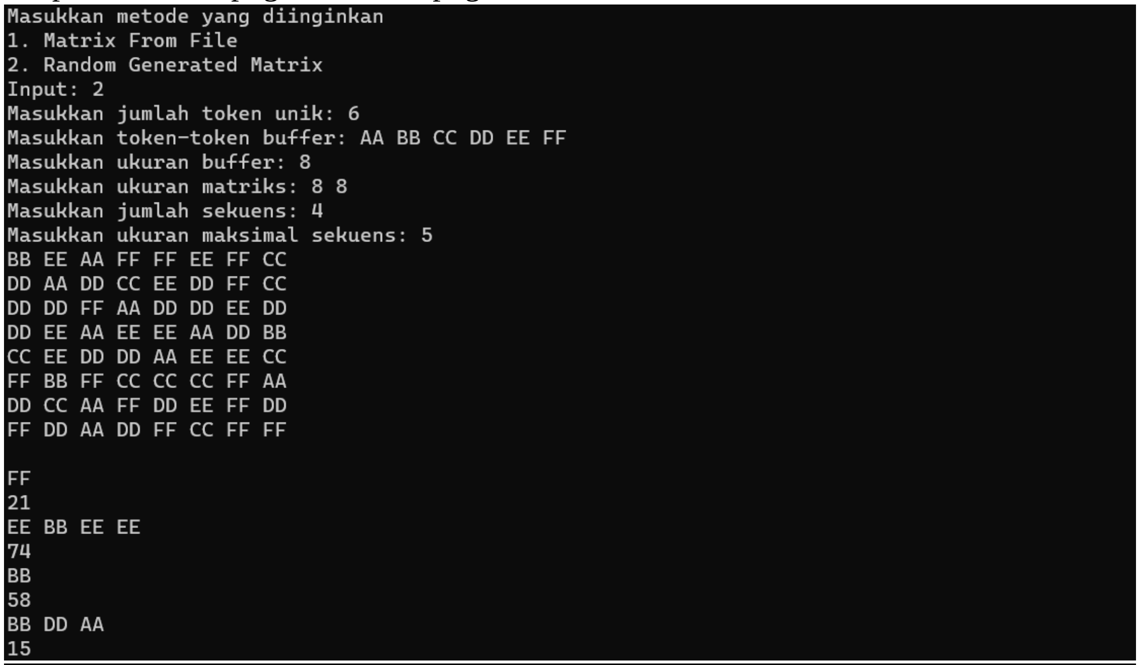
```
Sekuens Optimal: 44 B8 A7 5E CC 5E 44
Poin Optimal: 27
Koordinat Optimal:
3,1
3,3
1,3
1,1
6,1
6,4
1,4
159 ms

====================================
||    Simpan output dalam file?   ||
||--------------------------------||
|| (Y/y) : YES                    ||
|| Other Response : NO            ||
====================================
```

5. Input : 5_in.png (Input ini juga digunakan untuk cek kesamaan 2 metode)

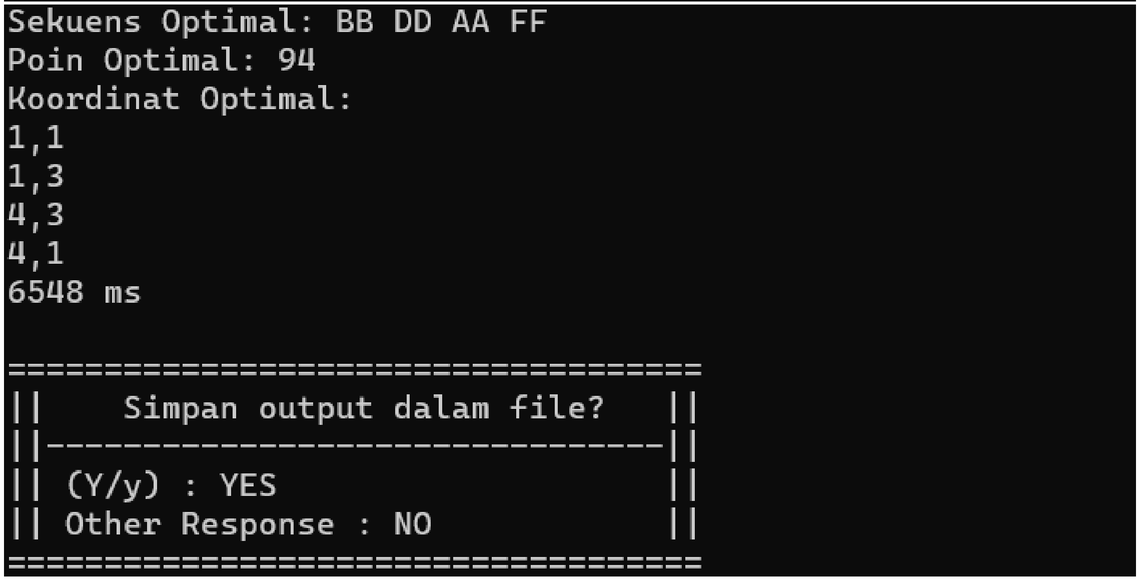


```
8
8 8
BB EE AA FF FF EE FF CC
DD AA DD CC EE DD FF CC
DD DD FF AA DD DD EE DD
DD EE AA EE EE AA DD BB
CC EE DD DD AA EE EE CC
FF BB FF CC CC CC FF AA
DD CC AA FF DD EE FF DD
FF DD AA DD FF CC FF FF
4
FF
21
EE BB EE EE
74
BB
58
BB DD AA
15
```

Output : 5_out_1.png & 5_out_2.png

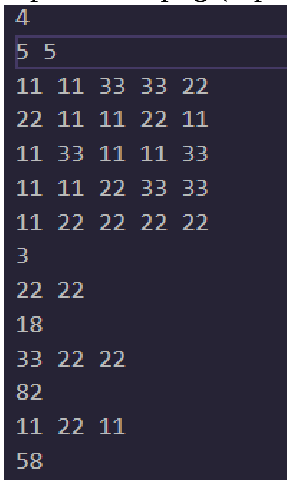


```
Masukkan metode yang diinginkan
1. Matrix From File
2. Random Generated Matrix
Input: 2
Masukkan jumlah token unik: 6
Masukkan token-token buffer: AA BB CC DD EE FF
Masukkan ukuran buffer: 8
Masukkan ukuran matriks: 8 8
Masukkan jumlah sekuens: 4
Masukkan ukuran maksimal sekuens: 5
BB EE AA FF FF EE FF CC
DD AA DD CC EE DD FF CC
DD DD FF AA DD DD EE DD
DD EE AA EE EE AA DD BB
CC EE DD DD AA EE EE CC
FF BB FF CC CC CC FF AA
DD CC AA FF DD EE FF DD
FF DD AA DD FF CC FF FF

FF
21
EE BB EE EE
74
BB
58
BB DD AA
15
```



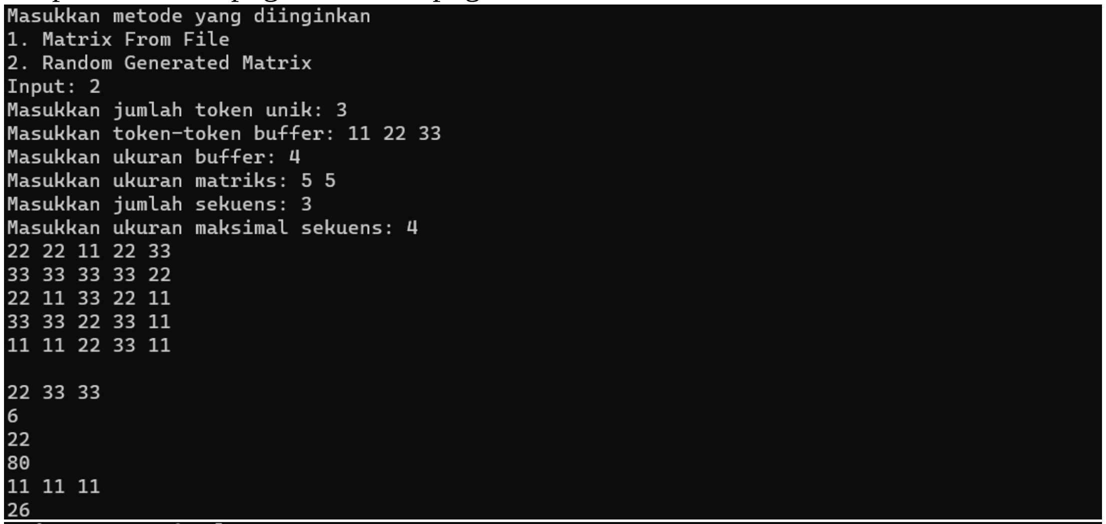
```
Sekuens Optimal: BB DD AA FF
Poin Optimal: 94
Koordinat Optimal:
1,1
1,3
4,3
4,1
6548 ms

====================================
||    Simpan output dalam file?   ||
||--------------------------------||
|| (Y/y) : YES                    ||
|| Other Response : NO            ||
====================================
```

6. Input : 6_in.png (Input ini juga digunakan untuk cek kesamaan 2 metode)

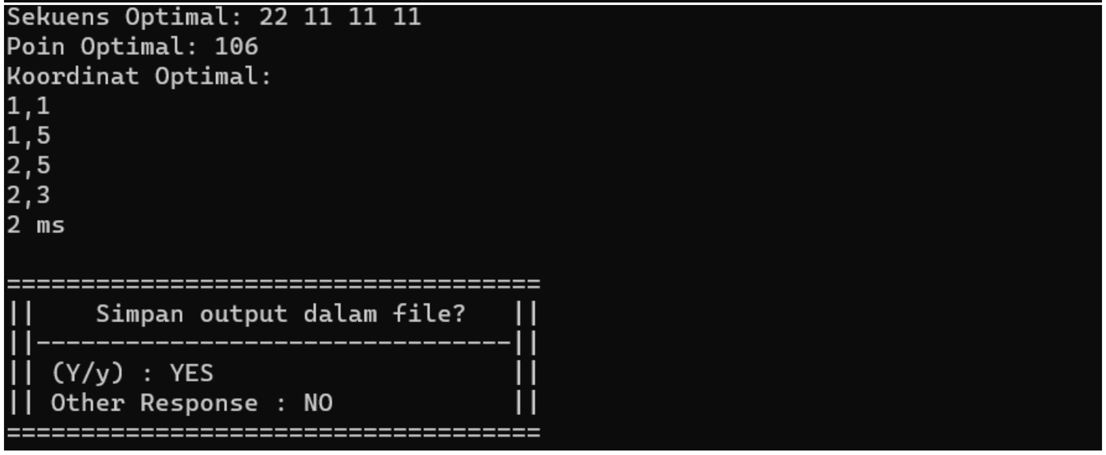


```
4
5 5
11 11 33 33 22
22 11 11 22 11
11 33 11 11 33
11 11 22 33 33
11 22 22 22 22
3
22 22
18
33 22 22
82
11 22 11
58
```

Output : 6_out_1.png & 6_out_2.png



```
Masukkan metode yang diinginkan
1. Matrix From File
2. Random Generated Matrix
Input: 2
Masukkan jumlah token unik: 3
Masukkan token-token buffer: 11 22 33
Masukkan ukuran buffer: 4
Masukkan ukuran matriks: 5 5
Masukkan jumlah sekuens: 3
Masukkan ukuran maksimal sekuens: 4
22 22 11 22 33
33 33 33 33 22
22 11 33 22 11
33 33 22 33 11
11 11 22 33 11

22 33 33
6
22
80
11 11 11
26
```



```
Sekuens Optimal: 22 11 11 11
Poin Optimal: 106
Koordinat Optimal:
1,1
1,5
2,5
2,3
2 ms

======================================
||    Simpan output dalam file?     ||
||--------------------------------||
|| (Y/y) : YES                      ||
|| Other Response : NO              ||
======================================
```

LINK REPOSITORY

https://github.com/scifo04/Tucil_Stima_IF2211_01

CHECKLIST

Poin	Ya	Tidak
1. Program dapat dikompilasi tanpa kesalahan	✓	
2. Program berhasil dijalankan	✓	
3. Program dapat membaca 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		✓