LAPORAN TUGAS KECIL I

IF2211 STRATEGI ALGORITMA

Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force

Disusun oleh:

Suthasoma Mahardhika Munthe 13522098

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2024

Daftar Isi

BAGIAN I
ALGORITMA BRUTE FORCE

Algoritma *Brute Force*, adalah algoritma dengan pendekatan yang lempang untuk memecahkan suatu masalah. Algoritma *brute force* adalah algoritma yang sangat sederhana, langsung, dan jelas caranya karena prosesnya *just solve it*. Dalam penyelesaian *Cyberpunk 2077 Breach* Protocol dengan pendekatan *brute* force, algoritma yang digunakan adalah sebagai berikut.

1. Pilih token pada baris paling atas (baris pertama) lalu tambahkan ke dalam buffer. Langkah selanjutnya adalah memilih token berikutnya secara vertikal (melakukan iterasi untuk setiap token yang belum diambil pada kolom yang sama). Lakukan iterasi untuk setiap token pada baris paling atas.
2. Kondisi buffer belum penuh. Jika langkah sebelumnya dilakukan secara vertikal, lakukan iterasi untuk setiap token yang belum diambil secara horizontal. Jika langkah sebelumnya dilakukan secara horizontal, lakukan iterasi untuk setiap token yang belum diambil secara vertikal. Kemudian lakukan pengecekan *reward* dari sekuens yang tersedia. Jika buffer mengandung sekuen, bobot *reward*-nya positif, dan pertama kali ditemukan, inisialisasi solusi dengan buffer tersebut. Jika ditemukan buffer lain dengan bobot *reward* lebih besar dan jumlah token lebih sedikit (optimal), ganti solusi dengan buffer yang baru.
3. Lakukan langkah 2 sampai buffer penuh.
4. Jika solusi ditemukan, solusi akan memuat buffer yang paling optimal. Sebaliknya, solusi kosong jika tidak ditemukan solusi yang optimal.

Setiap kali program menemukan buffer yang mengandung sekuens, bobot *reward* yang bernilai positif akan dibandingkan dengan solusi yang awalnya sudah ditemukan. Jika belum ditemukan maka solusi diinisialisasi dengan buffer tersebut. Syarat penggantian buffer solusi adalah bobot *reward* buffer lebih besar dan jumlah token buffer lebih sedikit dibandingkan solusi yang sebelumnya.

BAB II
SOURCE PROGRAM

Proyek ini diimplementasikan menggunakan Bahasa Java dengan compiler versi 20+. Fungsi atau prosedur yang diimplementasikan dalam program ini adalah sebagai berikut.

1. addTokenNew,
2. isSeqExisted,
3. checkBuffer,
4. getToken,
5. setToken,
6. isTokenValid,
7. isSeqTokenEqual,
8. setVisited,
9. getVisited,
10. isNumeric,
11. readText,
12. randomGenerate,
13. generateSeq,
14. isEqualSeqExist,
15. bruteForce,
16. startSearch,
17. displaySolution,
18. showProperties,
19. saveSolution,
20. readFromFile,
21. readCLI,
22. showHeader,
23. showMainMenu,
24. askSave, dan
25. startGame.

Berikut ini adalah source program proyek ini.

Nama file: Main.java (Driver)

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

import models.*;
import operations.Game;

public class Main {
```

```
    static Game newGame;
    static boolean isPlay;
    static int userInput;
    static Scanner input;

    public static int toInteger (String s){
        return Integer.parseInt(s);
    }

    public static boolean readFromFile() {
        newGame = new Game();
        String path;
        File testFile;
        do {
            printGreen("Masukkan source file: ");
            path = input.nextLine();
            testFile = new File("../test/input/"+path);
            if (!testFile.exists()) {
                printlnRed("System tidak dapat menemukan file!");
            }
        } while (!testFile.exists());
        return newGame.readText("../test/input/"+path);
    }

    public static void readCLI() {
        newGame = new Game();
        int nTOken, bufferSize, row, col, nSequence, seqMaxSize;
        String tokens[];
        String nt;
        do {
            printGreen("Jumlah token unik  : ");
            nt = input.nextLine();
            if (!Game.isNumeric(nt) || Integer.parseInt(nt) < 1) {
                printlnRed("Masukan tidak valid.");
            }
        } while (!Game.isNumeric(nt) || Integer.parseInt(nt) < 1);
        nTOken = Integer.parseInt(nt);
        do {
            printGreen("Token              : ");
            nt = input.nextLine();
            if (!Sequence.isTokenValid(nt, nTOken)) {
                printlnRed("Masukan tidak sesuai.");
            }
        } while (!Sequence.isTokenValid(nt, nTOken));
        tokens = nt.split(" ");
```

```
        do {
            printGreen("Ukuran buffer      : ");
            nt = input.nextLine();
            if (!Game.isNumeric(nt) || Integer.parseInt(nt) < 1) {
                printlnRed("Masukan tidak valid.");
            }
        } while (!Game.isNumeric(nt) || Integer.parseInt(nt) < 1);
        bufferSize = Integer.parseInt(nt);
        String dimension[];
        do {
            printGreen("Ukuran matrix (w,h): ");
            nt = input.nextLine();
            dimension = nt.split(" ");
            if (dimension.length != 2 || !Game.isNumeric(dimension[0])
|| !Game.isNumeric(dimension[1]) || toInteger(dimension[0]) < 0 ||
toInteger(dimension[1]) < 0) {
                printlnRed("Masukan tidak valid.");
            }
        } while (dimension.length != 2 || !Game.isNumeric(dimension[0])
|| !Game.isNumeric(dimension[1]) || toInteger(dimension[0]) < 0 ||
toInteger(dimension[1]) < 0);
        col = toInteger(dimension[0]);
        row = toInteger(dimension[1]);
        do {
            printGreen("Jumlah sekuens     : ");
            nt = input.nextLine();
            if (!Game.isNumeric(nt) || Integer.parseInt(nt) < 2) {
                printlnRed("Masukan tidak valid.");
            }
        } while (!Game.isNumeric(nt) || Integer.parseInt(nt) < 2);
        nSequence = Integer.parseInt(nt);
        do {
            printGreen("Ukuran maks sekuen : ");
            nt = input.nextLine();
            if (!Game.isNumeric(nt) || Integer.parseInt(nt) < 2) {
                printlnRed("Masukan tidak valid.");
            }
        } while (!Game.isNumeric(nt) || Integer.parseInt(nt) < 2);
        seqMaxSize = Integer.parseInt(nt);
        newGame.randomGenerate(nTOken, tokens, bufferSize, row, col, nSequence,
seqMaxSize);
        System.out.println();
        newGame.showProperties();
    }
```

```
    public static void printlnGreen(String s) {
        String rest = "\u001B[0m";
        String green = "\u001B[32m";

        System.out.println(green + s + rest);
    }

    public static void printGreen(String s) {
        String rest = "\u001B[0m";
        String green = "\u001B[32m";

        System.out.print(green + s + rest);
    }

    public static void printlnYellow(String s) {
        String rest = "\u001B[0m";
        String yellow = "\u001B[33m";

        System.out.println(yellow + s + rest);
    }

    public static void printlnRed(String s) {
        String rest = "\u001B[0m";
        String red = "\u001B[31m";

        System.out.println(red + s + rest);
    }

    public static void showHeader() {
        String header = "";
        try {
            File inputFile = new File("../src/header.txt");
            Scanner readFile = new Scanner(inputFile);
            String line;
            while (readFile.hasNextLine() && (line = readFile.nextLine()) != null) {
                header += line+"\n";
            }
            readFile.close();
        } catch (FileNotFoundException e){
            System.out.println("An error occurred.");
            e.printStackTrace();
        }
        printlnYellow(header);
        printlnGreen("Selamat datang!!");
        printlnGreen("");
```

```
    }

    public static void showMainMenu() {
        printlnGreen("Menu:");
        printlnYellow("1. Masukan berupa file.");
        printlnYellow("2. Masukan random.");
        printlnYellow("0. Keluar.");
    }

    public static void askInput() {
        String userInp;
        do {
            printGreen("Masukkan pilihan: ");
            userInp = input.nextLine();
            if (!Game.isNumeric(userInp) || Integer.parseInt(userInp) < 0 ||
Integer.parseInt(userInp) > 2) {
                printlnRed("Masukan tidak sesuai!");
            }
        } while (!Game.isNumeric(userInp) || Integer.parseInt(userInp) < 0 ||
Integer.parseInt(userInp) > 2);
        userInput = toInteger(userInp);
    }

    public static void askSave(int time) {
        String userInp;
        do {
            printGreen("Apakah ingin menyimpan solusi? (y/n): ");
            userInp = input.nextLine();
            if (userInp == "" || (userInp.charAt(0) != 'y' && userInp.charAt(0) !=
'Y' && userInp.charAt(0) != 'N' && userInp.charAt(0) != 'n')) {
                printlnRed("Masukan tidak sesuai!");
            }
        } while (userInp == "" || (userInp.charAt(0) != 'y' && userInp.charAt(0) !=
'Y' && userInp.charAt(0) != 'N' && userInp.charAt(0) != 'n'));
        if (userInp.charAt(0) == 'Y' || userInp.charAt(0) == 'y') {
            newGame.saveSolution(time);
        } else {
            System.out.println();
        }
    }

    public static void startGame() {
        showHeader();
        isPlay = true;
        while (isPlay) {
```

```
            showMainMenu();
            askInput();
            if (userInput == 1){
                boolean fileValid = readFromFile();
                if (fileValid) {
                    long start = System.currentTimeMillis();
                    newGame.startSearch();
                    long end = System.currentTimeMillis();
                    int diff = (int)(end-start);
                    newGame.displaySolution(diff);
                    askSave(diff);
                } else {
                    printlnRed("\nFile tidak memuat data yang sesuai format!\n");
                }
            } else if (userInput == 2) {
                readCLI();
                long start = System.currentTimeMillis();
                newGame.startSearch();
                long end = System.currentTimeMillis();
                int diff = (int)(end-start);
                newGame.displaySolution(diff);
                askSave(diff);
            } else {
                isPlay = false;
            }
        }
        printlnYellow("\nBye-bye...\n");
    }

    public static void main (String args[]) {
        input = new Scanner(System.in);
        startGame();
        input.close();
    }
}
```

Nama file: Buffer.java

```
package models;

public class Buffer {
    public int position[][];
    public int weight;
    public int length;
    public int maxsize;
    public String buffString;
```

```
    public Buffer() {
    }

    public Buffer(int maxsize) {
        position = new int[maxsize][2];
        length = 0;
        weight = 0;
        this.maxsize = maxsize;
        buffString = "";
    }

    public Buffer(Buffer buf) {
        this(buf.maxsize);
        this.length = buf.length;
        this.weight = buf.weight;
        for (int i = 0; i< this.length; i++){
            this.position[i][0] = buf.position[i][0];
            this.position[i][1] = buf.position[i][1];
        }
        this.buffString = new String(buf.buffString);
    }

    public Buffer addTokenNew(String token, int row, int col){
        Buffer newBuf = new Buffer(this);
        if (newBuf.length != 0) {
            newBuf.buffString += " ";
        }
        newBuf.buffString += token;
        newBuf.position[newBuf.length][0] = row;
        newBuf.position[newBuf.length][1] = col;
        newBuf.length += 1;
        return newBuf;
    }

    public boolean isSeqExisted (Sequence seq) {
        return this.buffString.contains(seq.seqString);
    }

    public int checkBuffer(Sequence seq[]) {
        int len = seq.length;
        weight = 0;
        for (int i = 0; i < len; i++) {
            if (isSeqExisted(seq[i])) {
                weight += seq[i].weight;
```

```
            }
        }
        return weight;
    }
}
```

Nama file: Matrix.java

```
package models;

public class Matrix {
    String content[][];
    public int row;
    public int col;

    public Matrix() {
        content = new String[1000][1000];
        row = 0;
        col = 0;
    }

    public Matrix(int rows, int column) {
        content = new String[rows][column];
        row = rows;
        col = column;
    }

    public Matrix(Matrix mat) {
        this(mat.row, mat.col);
        this.content = mat.content;
    }

    public String getToken(int i, int j) {
        return content[i][j];
    }

    public void setToken(String token, int i, int j) {
        content[i][j] = token;
    }
}
```

Nama file: Sequence.java

```
package models;

import java.util.regex.Matcher;
import java.util.regex.Pattern;
```

```java
public class Sequence {
    public int length;
    public int weight;
    public String seqString;

    public static boolean isTokenValid(String tokens, int length) {
        String regex = "^[a-zA-Z0-9]*$";
        Pattern pattern = Pattern.compile(regex);
        String token[] = tokens.split(" ");
        if (token.length != length){
            return false;
        }
        for (int i = 0; i < token.length; i++) {
            Matcher matcher = pattern.matcher(token[i]);
            if (!matcher.matches() || token[i].length() != 2) {
                return false;
            }
        }
        return true;
    }

    public Sequence(){
        length = 0;
        weight = 0;
    }

    public Sequence(int length, int weight){
        this.length = length;
        this.weight = weight;
        this.seqString = "";
    }

    public Sequence(String token[], int weight){
        this(token.length, weight);
        for (int i = 0; i < length-1; i++){
            seqString += token[i]+" ";
        }
        seqString += token[length-1];
    }

    public boolean isSeqTokenEqual(Sequence second) {
        return this.seqString.equals(second.seqString);
    }
```

```
    @Override
    public String toString() {
        return "Sequence{"+seqString+", length:"+Integer.toString(length)+", weight:
"+Integer.toString(weight)+"}";
    }
}
```

Nama file: Visited.java

```
package models;

public class Visited {
    public boolean visited[][];
    public int row;
    public int col;

    public Visited (int row, int col) {
        visited = new boolean[row][col];
        this.row = row;
        this.col = col;
    }

    public Visited (Visited vist) {
        this(vist.row, vist.col);
        for (int i = 0; i < this.row; i++) {
            for (int j = 0; j < this.col; j++) {
                this.setVisited(vist.getVisited(i, j), i, j);
            }
        }
    }

    public void setVisited(boolean val, int row, int col) {
        this.visited[row][col] = val;
    }

    public boolean getVisited(int row, int col) {
        return this.visited[row][col];
    }
}
```

Nama file: Game.java

```
package operations;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
```

```java
import java.util.Random;
import java.util.Scanner;

import models.*;

public class Game {
    public Matrix matriks;
    public Sequence sekuen[];
    public Buffer solution;
    public boolean wasFound;

    public Game() {
        wasFound = false;
    }

    public static boolean isNumeric(String s) {
        if (s == null) {
            return false;
        }
        try {
            Integer.parseInt(s);
        } catch (NumberFormatException nfe) {
            return false;
        }
        return true;
    }

    public boolean readText(String path){
        wasFound = false;
        try {
            File inputFile = new File(path);
            Scanner readFile = new Scanner(inputFile);
            String line;
            if (!readFile.hasNextLine()){
                readFile.close();
                return false;
            }
            line = readFile.nextLine();
            if (!isNumeric(line) || Integer.parseInt(line) < 0){
                readFile.close();
                return false;
            }
            this.solution = new Buffer(Integer.parseInt(line));
            if (!readFile.hasNextLine()){
                readFile.close();
```

```
                return false;
            }
            line = readFile.nextLine();
            String dimension[] = line.split(" ");
            if (dimension.length != 2 || !Game.isNumeric(dimension[0])
|| !Game.isNumeric(dimension[1]) || Integer.parseInt(dimension[0]) < 0 ||
Integer.parseInt(dimension[1]) < 0) {
                readFile.close();
                return false;
            }
            int row = Integer.parseInt(dimension[1]);
            int col = Integer.parseInt(dimension[0]);
            this.matriks = new Matrix(row, col);
            for (int i = 0; i < row; i++){
                if (!readFile.hasNextLine()){
                    readFile.close();
                    return false;
                }
                line = readFile.nextLine();
                String token[] = line.split(" ");
                if (!Sequence.isTokenValid(line, col)){
                    readFile.close();
                    return false;
                }
                for (int j = 0; j < col-1; j++) {
                    matriks.setToken(token[j], i, j);
                }
                String temp = "";
                temp += token[col-1].charAt(0);
                temp += token[col-1].charAt(1);
                matriks.setToken(temp, i, col-1);
            }
            line = readFile.nextLine();
            if (!isNumeric(line) || Integer.parseInt(line) < 0){
                readFile.close();
                return false;
            }
            this.sekuen = new Sequence[Integer.parseInt(line)];
            for (int i = 0; i < this.sekuen.length; i++){
                if (!readFile.hasNextLine()){
                    readFile.close();
                    return false;
                }
                line = readFile.nextLine();
                if (!readFile.hasNextLine()){
```

```
                    readFile.close();
                    return false;
                }
                String wght = readFile.nextLine();
                if (!isNumeric(wght)){
                    readFile.close();
                    return false;
                }
                int weight = Integer.parseInt(wght);
                String tokens[] = line.split(" ");
                if (!Sequence.isTokenValid(line, tokens.length)) {
                    readFile.close();
                    return false;
                }
                String clean = "";
                clean += tokens[tokens.length-1].charAt(0);
                clean += tokens[tokens.length-1].charAt(1);
                tokens[tokens.length-1] = clean;
                sekuen[i] = new Sequence(tokens, weight);
            }
            readFile.close();
            return true;
        } catch (FileNotFoundException e){
            System.out.println("An error occurred.");
            e.printStackTrace();
            return false;
        }
    }

    public void randomGenerate(int nToken, String tokens[], int bufferSize, int row,
int col, int nSequence, int seqMaxSize) {
        wasFound = false;
        Random rand = new Random();
        this.matriks = new Matrix(row, col);
        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                this.matriks.setToken(tokens[rand.nextInt(nToken)], i, j);
            }
        }
        this.solution = new Buffer(bufferSize);
        this.sekuen = new Sequence[nSequence];
        for (int i = 0; i < nSequence; i++) {
            do {
                int seqLength = rand.nextInt(seqMaxSize-2+1)+2;
                this.sekuen[i] = generateSeq(tokens, seqLength);
```

```
            } while (isEqualSeqExist(this.sekuen[i], this.sekuen, i));
        }
    }

    public static Sequence generateSeq (String tokens[], int length) {
        Random rand = new Random();
        int nToken = tokens.length;
        Sequence out = new Sequence(length, rand.nextInt(101));
        for (int i = 0; i < length; i++) {
            out.seqString += tokens[rand.nextInt(nToken)];
            if (i != length-1) {
                out.seqString += " ";
            }
        }
        return out;
    }

    public static boolean isEqualSeqExist(Sequence seq, Sequence list[], int offset)
{
        for (int i = 0; i < offset; i++) {
            if (seq.isSeqTokenEqual(list[i])) {
                return true;
            }
        }
        return false;
    }

    public void bruteForce(Matrix matrix, Visited visit, boolean isVertical, Buffer
buff, int curRow, int curCol) {
        if (buff.maxsize > buff.length){
            Visited newVisit = new Visited(visit);
            newVisit.setVisited(true, curRow, curCol);
            Buffer newBuff = buff.addTokenNew(matrix.getToken(curRow, curCol),
curRow, curCol);
            if (newBuff.checkBuffer(sekuen) > 0) {
                if (wasFound) {
                    if (newBuff.weight > solution.weight) {
                        solution = new Buffer(newBuff);
                    } else if (newBuff.weight == solution.weight && newBuff.length <
solution.length) {
                        solution = new Buffer(newBuff);
                    }
                } else {
                    wasFound = true;
                    solution = new Buffer(newBuff);
```

```
                }
            }
            if (isVertical) {
                for (int i = 0; i < newVisit.row; i++){
                    if (!newVisit.getVisited(i, curCol)){
                        bruteForce(matrix, newVisit, !isVertical, newBuff, i,
curCol);
                    }
                }
            } else {
                for (int i = 0; i < newVisit.col; i++){
                    if(!newVisit.getVisited(curRow, i)) {
                        bruteForce(matrix, newVisit,!isVertical, newBuff, curRow, i);
                    }
                }
            }
        }
    }

    public void startSearch() {
        int column = matriks.col;
        Buffer buff = new Buffer(solution);
        Visited visit = new Visited(matriks.row, matriks.col);
        for (int j = 0; j < column; j++) {
            bruteForce(matriks, visit,true, buff, 0, j);
        }
    }

    public void displaySolution(int time) {
        System.out.println("\nSolution:");
        if (wasFound){
            System.err.println(solution.weight);
            String buff = solution.buffString;
            System.out.println(buff);
            for (int i = 0; i < solution.length; i++){
                System.out.println((solution.position[i][1]+1)+",
"+(solution.position[i][0]+1));
            }
        } else {
            System.out.println("0");
            System.out.println("TIdak ada solusi");
        }
        System.out.println("\n"+time+"ms\n");
    }
```

```
    public void showProperties() {
        String temp = "";
        for (int i = 0 ; i < matriks.row; i++){
            for (int j = 0 ; j < matriks.col; j++){
                temp += matriks.getToken(i, j);
                if (j != matriks.col-1){
                    temp+=" ";
                }
            }
            temp += "\n";
        }
        System.out.println("Matriks:");
        System.out.println(temp);
        System.out.println("Sekuens:");
        for (int i = 0; i < sekuen.length; i++){
            String sek = sekuen[i].seqString;
            sek += " bobot: "+sekuen[i].weight;
            System.out.println(sek);
        }
    }

    public void saveSolution(int time) {
        String save = "";
        if (wasFound) {
            save += Integer.toString(solution.weight) +"\n";
            save += solution.buffString;
            save += "\n";
            for (int i = 0; i < solution.length; i++) {
                save += (solution.position[i][1]+1) +", "+(solution.position[i][0]+1)
+ "\n";
            }
            save += "\n" + time + "ms\n";
        } else {
            save += "0\nTidak ada Solusi\n\n"+time+"ms\n";
        }

        String fileName = "../test/output/solution";
        File testFile = new File(fileName+".txt");
        int idx = 0;
        while (testFile.exists()){
            idx ++;
            testFile = new File(fileName+"_"+idx+".txt");
        }
        if (idx != 0) {
            fileName += "_" + idx + ".txt";
```

```
        } else {
            fileName += ".txt";
        }
        try {
            FileWriter writer = new FileWriter(fileName);
            writer.write(save);
            writer.close();
            System.out.println("\nSolusi berhasil disimpan pada: "+fileName+"\n");
        } catch (Exception e) {
            System.out.println("An error occurred\n");
            e.printStackTrace();
        }
    }
}
```

BAGIAN III
SCREENSHOT HASIL TEST

```
Selamat datang!!

Menu:
1. Masukan berupa file.
2. Masukan random.
0. Keluar.
Masukkan pilihan: 1
Masukkan source file: tc1.txt

Solution:
209
FF FF 7A 1C BD FF G7 BD 1C
3, 1
3, 2
1, 2
1, 7
4, 7
4, 4
5, 4
5, 2
6, 2

5066ms

Apakah ingin menyimpan solusi? (y/n): y
```

test > input > tc1.txt

```
9
7 7
55 1C FF FF 55 FF 1C
7A BD FF FF BD 1C 1C
BD FF 55 55 55 G7 BD
FF 7A BD FF G7 7A G7
7A 55 55 BD 55 7A FF
FF 55 BD 1C BD 7A FF
1C FF 55 BD FF 55 55
11
G7 55 G7
15
G7 BD
1
7A 1C BD FF
31
FF FF
7
FF
48
BD 7A BD 7A
3
BD FF G7
24
G7
36
BD 1C
47
FF 7A
15
1C G7 G7
10
```

Gambar 1. Input dan output dengan file tc1.txt

```
1. Masukan berupa file.
2. Masukan random.
0. Keluar.
Masukkan pilihan: 1
Masukkan source file: tc2.txt

Solution:
937
EE H6 EE EE 73 73
2, 1
2, 3
5, 3
5, 1
3, 1
3, 4

44ms

Apakah ingin menyimpan solusi? (y/n): y

Solusi berhasil disimpan pada: ../test/output/solution_1.txt
```

test > input > tc2.txt

```
7
6 6
73 EE 73 H6 EE 73
G6 73 G6 G6 H6 G6
G6 H6 H6 73 EE H6
G6 EE 73 73 73 73
H6 73 73 H6 G6 EE
EE EE H6 G6 H6 73
3
EE EE 73
164
H6 EE
52
73 73
721
```

Gambar 2.a. Input dan output dengan file tc2.txt

test > output > solution_1.txt

```
937
EE H6 EE EE 73 73
2, 1
2, 3
5, 3
5, 1
3, 1
3, 4

44ms

```

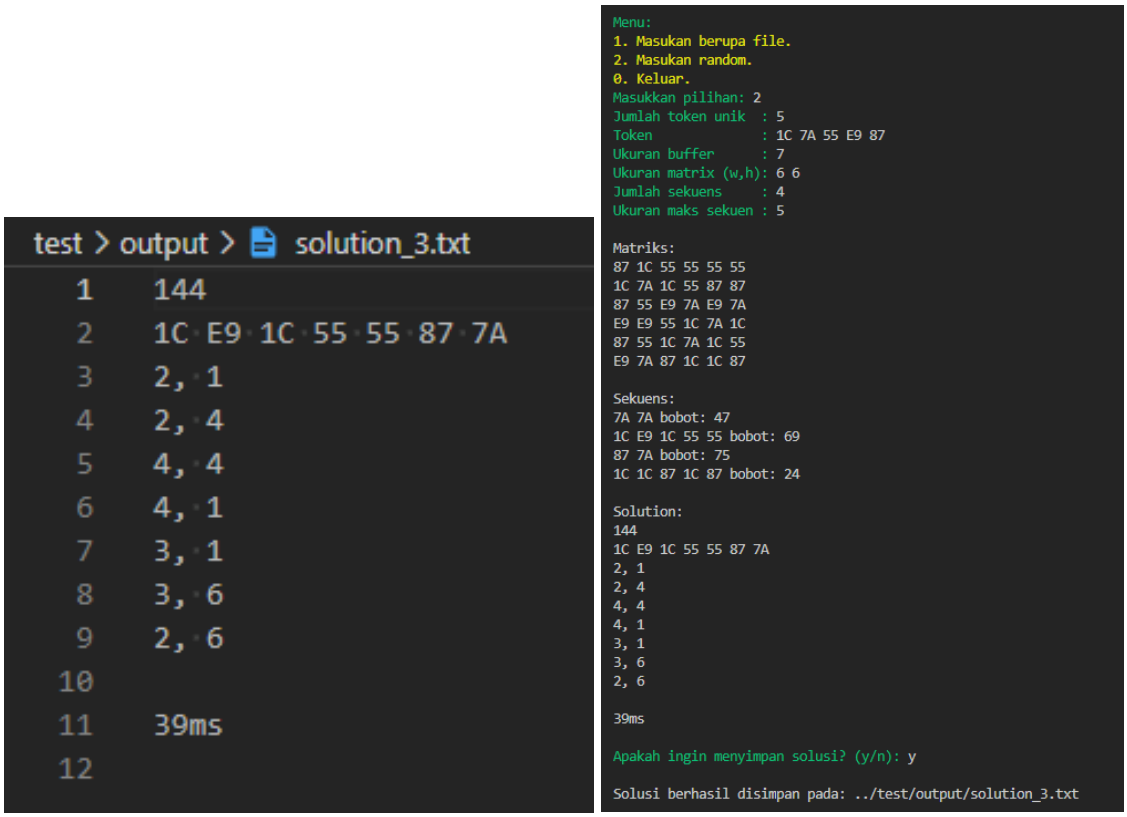
Gambar 2.b. File output hasil penyelesaian tc2.txt

test > input > tc3.txt

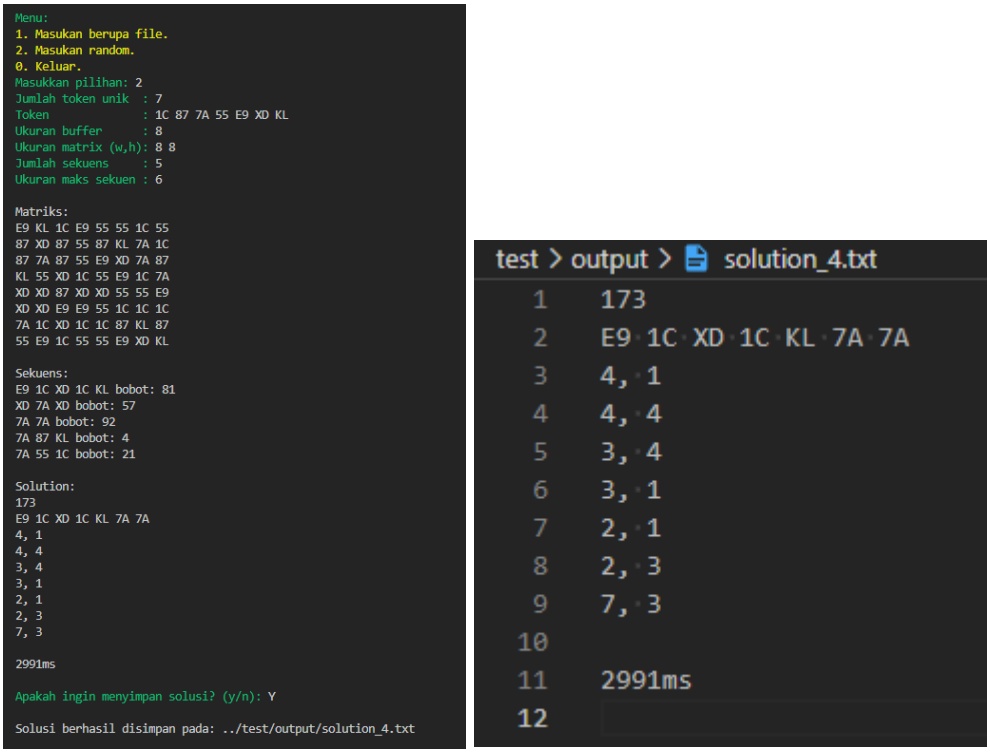
```
8
7 7
1C 1C AS E9 7A XD AS
XD 1C 1C 87 1C AS XD
7A 87 E9 87 7A 87 E9
1C 7A 1C 7A 1C 1C 7A
XD 1C E9 7A BD E9 XD
E9 AS 87 87 XD AS 7A
AS XD BD AS 7A XD 1C
6
7A 87
44
E9 7A 87
60
XD 7A
7
7A 87 7A XD
39
1C BD 1C 87
27
87 AS 1C 87
52
```

```
Menu:
1. Masukan berupa file.
2. Masukan random.
0. Keluar.
Masukkan pilihan: 1
Masukkan source file: tc3.txt

Solution:
156
AS E9 7A 87 AS 1C 87
3, 1
3, 5
4, 5
4, 6
2, 6
2, 2
4, 2

860ms

Apakah ingin menyimpan solusi? (y/n): y

Solusi berhasil disimpan pada: ../test/output/solution_2.txt
```

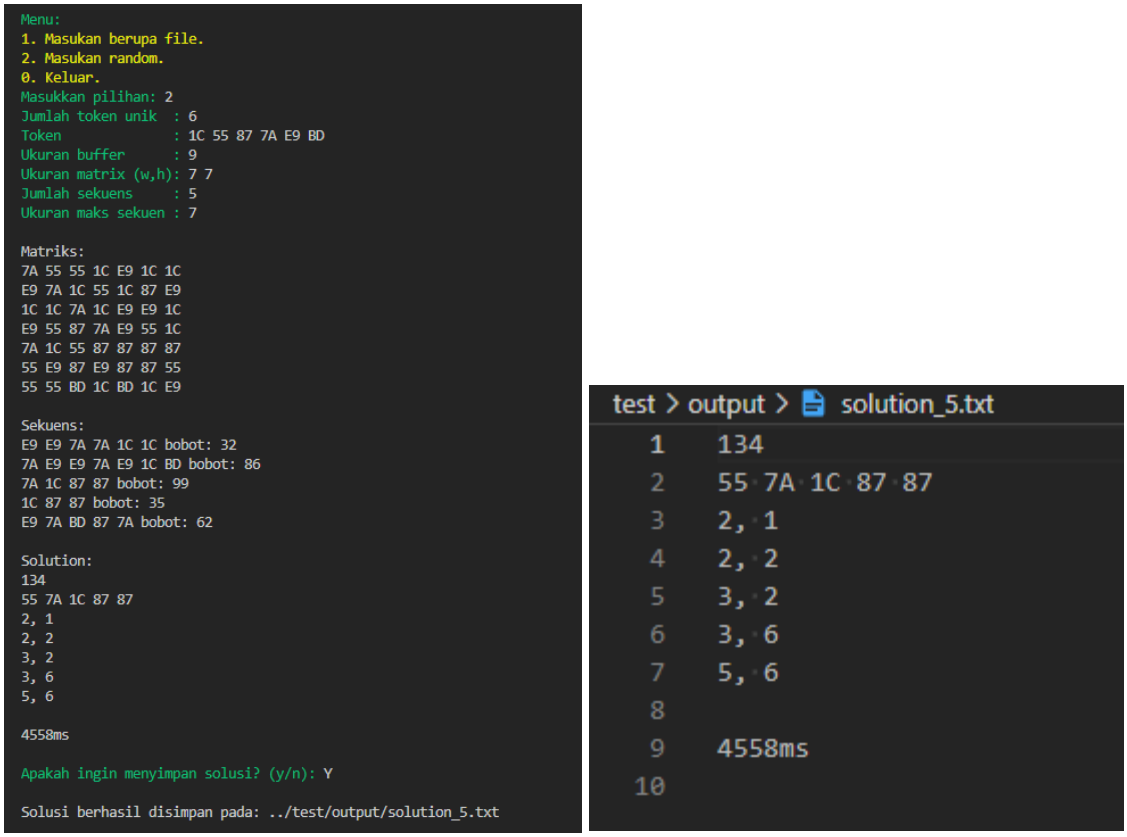
Gambar 3. Input dan output tc3.txt



Gambar 4. Input dan output test *generate* random pertama



Gambar 5. Input dan output test *generate* random kedua.



Gambar 6. Input dan output test *generate* random ketiga.

LINK REPOSITORY

https://github.com/sotul04/Tucil1_13522098

CHECKLIST

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		✓