Laporan Tugas Kecil 1 IF2211 Strategi Algoritma

Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force

Semester II Tahun 2024/2025

Jessica Allen - 13523059

A. IQ Puzzler Pro

IQ Puzzler Pro adalah permainan papan yang diproduksi oleh perusahaan Smart Games. Tujuan dari permainan ini adalah pemain harus dapat mengisi seluruh papan dengan blok puzzle yang tersedia. Komponen penting dari permainan IQ Puzzler Pro terdiri dari Board, yang merupakan komponen utama yang menjadi tujuan permainan dimana pemain harus mampu mengisi seluruh area papan menggunakan blok-blok yang telah disediakan. Permainan ini juga terdiri atas Blok, yang merupakan komponen yang digunakan pemain untuk mengisi papan kosong hingga terisi penuh. Setiap blok memiliki bentuk yang unik dan semua blok harus digunakan untuk menyelesaikan puzzle.

Permainan dimulai dengan papan yang kosong. Pemain dapat meletakkan blok puzzle sedemikian sehingga tidak ada blok yang bertumpang tindih (kecuali dalam kasus 3D). Setiap blok puzzle dapat dirotasikan maupun dicerminkan. Puzzle dinyatakan selesai jika dan hanya jika papan terisi penuh dan seluruh blok puzzle berhasil diletakkan.

B. Algoritma yang Digunakan

Untuk menyelesaikan IQ Puzzler Pro pada Tugas Kecil kali ini, diwajibkan untuk menggunakan algoritma Brute Force. Untuk program yang saya buat, saya menggunakan sistem brute force dengan pertama-tama mencari segala jenis kemungkinan posisi (versi hasil transformasi) dari blok-blok yang ada. Baik dibalik secara horizontal, ataupun diputar. Dalam hal ini, per-*piece* saya putarkan setiap 90 derajat sekali. Karena perputaran ini *full circle*, jadi *piece* tidak perlu untuk dibalik secara vertikal lagi.

Setelah itu, satu per satu *piece* akan dicoba untuk diletakkan pada setiap baris dan kolom yang memungkinkan. Jika *piece* tidak dapat ditaruh (tidak valid), maka akan dicoba hasil transformasi lainnya (versi) dari *piece* tersebut. Jika *piece* selanjutnya tidak dapat ditaruh secara valid pada *piece* setelah itu, maka akan dicoba kembali dari *piece* sebelumnya dengan posisi yang berbeda, kemudian dicoba lagi *piece* selanjutnya. Hal ini akan diulang terus hingga papan terisi penuh dengan semua *piece*. Setelah itu, program akan mengeluarkan output bentuk penyelesaian puzzle jika menemukan solusinya, dan memunculkan pesan jika gagal untuk menemukan solusinya. Secara garis besar, seperti itulah langkah-langkah algoritma *brute force* yang saya pikirkan dan terapkan dalam program ini.

Saya membagi kelas program menjadi Piece, IQPuzzlerSolver, InputOutput, dan Main. Piece merupakan kelas yang mengatur *piece-piece* blok pada permainan. IQPuzzlerSolver mengatur algoritma untuk menyelesaikan permainan. InputOutput merupakan kelas untuk membuka serta membaca file input, serta menyimpan file output. Terakhir, kelas Main adalah tempat disatukannya semua kelas tersebut agar program dapat dijalankan.

C. Source Code Program

Piece.java

```
import java.util.*;

public class Piece {
    private char name;
    private char[][] shape;
    private List<char[][]> versions;

    public Piece(char name, List<String> inputShape) {
        this.name = name;
        this.shape = changeToArray(inputShape);

        if (!cekTerhubung()) {
            System.out.println("Piece " + name + " tidak dapat
digunakan karena tidak terhubung.");
            return;
        }

        transform();
    }

    public char getName() {
        return name;
    }

    public char[][] getShape() {
        return shape;
    }

    public List<char[][]> getPieceVersions() {
        return versions;
    }

    private char[][] changeToArray(List<String> inputShape) {
        int rows = inputShape.size();
        int cols = 0;

        for (String row : inputShape) {
            cols = Math.max(cols, row.length());
        }

        char[][] hasil = new char[rows][cols];
        for (int r = 0; r < rows; r++) {
            String row = inputShape.get(r);
            for (int c = 0; c < cols; c++) {
                hasil[r][c] = (c < row.length()) ?
row.charAt(c) : ' ';
            }
        }
        return hasil;
    }
```

```
    private char[][] putar(char[][] matrix) {
        int rows = matrix.length;
        int cols = matrix[0].length;
        char[][] terputar = new char[cols][rows];

        for (int r = 0; r < rows; r++) {
            for (int c = 0; c < cols; c++) {
                terputar[c][rows - 1 - r] = matrix[r][c];
            }
        }
        return terputar;
    }

    private char[][] diBalik(char[][] matrix) {
        int rows = matrix.length;
        int cols = matrix[0].length;
        char[][] terbalik = new char[rows][cols];

        for (int r = 0; r < rows; r++) {
            for (int c = 0; c < cols; c++) {
                terbalik[r][cols - 1 - c] = matrix[r][c];
            }
        }
        return terbalik;
    }

    private boolean cekTerhubung() {
        int rows = shape.length;
        int cols = shape[0].length;
        boolean[][] udahCek = new boolean[rows][cols];

        int[][] arah = {
            {1, 0}, {0, 1}, {-1, 0}, {0, -1}, {1, 1}, {-1, 1},
{1, -1}, {-1, -1}
        };

        int startX = -1, startY = -1;
        for (int r = 0; r < rows; r++) {
            for (int c = 0; c < cols; c++) {
                if (shape[r][c] == name) {
                    startX = r;
                    startY = c;
                    break;
                }
            }
            if (startX != -1) break;
        }

        if (startX == -1) return false;

        Queue<int[]> antrian = new ArrayDeque<>();
```

```
        antrian.add(new int[]{startX, startY});
        udahCek[startX][startY] = true;

        int count = 0, hitung = 0;
        for (char[] row : shape) {
            for (char cell : row) {
                if (cell == name) hitung++;
            }
        }

        while (!antrian.isEmpty()) {
            int[] curr = antrian.poll();
            count++;

            for (int i = 0; i < arah.length; i++) {
                int newX = curr[0] + arah[i][0];
                int newY = curr[1] + arah[i][1];

                if (newX >= 0 && newX < rows && newY >= 0 &&
newY < cols &&
                    !udahCek[newX][newY] && shape[newX][newY]
== name) {
                    udahCek[newX][newY] = true;
                    antrian.add(new int[]{newX, newY});
                }
            }
        }

        return count == hitung;
    }

    public void transform() {
        versions = new ArrayList<>();
        char[][] now = shape;

        for (int i = 0; i < 4; i++) {
            now = putar(now);
            versions.add(now);
        }

        now = diBalik(shape);
        versions.add(now);
        for (int i = 0; i < 3; i++) {
            now = putar(now);
            versions.add(now);
        }
    }
}
```

InputOutput.java

```
import java.io.*;
import java.util.*;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.List;

public class InputOutput {
    public static List<Piece> bacaInput(String fileName, int[]
ukuran, List<String> newBoard) throws IOException {
        List<Piece> pieces = new ArrayList<>();
        Map<Character, Boolean> pieceTerpakai = new
HashMap<>();

        try (BufferedReader reader = new BufferedReader(new
FileReader(fileName))) {
            String line = reader.readLine();

            if (line == null) {
                throw new IllegalArgumentException("File input
tidak valid, tidak memiliki isi.");
            }

            String[] input = line.split(" ");
            if (input.length != 3) {
                throw new IllegalArgumentException("Input
ukuran pada file tidak lengkap/sesuai.");
            }

            int lebar = Integer.parseInt(input[0]);
            int tinggi = Integer.parseInt(input[1]);
            int jumlahPieces = Integer.parseInt(input[2]);

            if (lebar <= 0 || tinggi <= 0 || jumlahPieces <=
0) {
                throw new IllegalArgumentException("Ukuran
board dan jumlah pieces yang ada pada file input harus lebih
besar dari 0.");
            }

            ukuran[0] = lebar;
            ukuran[1] = tinggi;

            String boardType = reader.readLine();
            if (boardType == null) throw new
IllegalArgumentException("File input tidak mengandung jenis
board.");
```

```
            if ("DEFAULT".equals(boardType)) {
                for (int i = 0; i < tinggi; i++) {
                    newBoard.add("1".repeat(lebar));
                }
                List<String> pieceSekarang = new
ArrayList<>();
                Character nameSekarang = null;

                String lineSekarang;
                while ((lineSekarang = reader.readLine()) !=
null) {
                    if (lineSekarang.trim().isEmpty()) {
                        throw new
IllegalArgumentException("Ada baris piece yang kosong pada
file input.");
                    }

                    char name = findName(lineSekarang);
                    if (nameSekarang == null || name !=
nameSekarang) {
                        if (nameSekarang != null) {
                            pieces.add(new Piece(nameSekarang,
pieceSekarang));
                            pieceTerpakai.put(nameSekarang,
true);
                        }

                        if (pieceTerpakai.getOrDefault(name,
false)) {
                            throw new
IllegalArgumentException("Ada dua piece dengan nama " + name +
".");
                        }

                        nameSekarang = name;
                        pieceSekarang = new ArrayList<>();
                    }
                    pieceSekarang.add(lineSekarang);
                }

                if (nameSekarang != null) {
                    pieces.add(new Piece(nameSekarang,
pieceSekarang));
                }

                if (pieces.size() != jumlahPieces) {
                    throw new IllegalArgumentException("Jumlah
piece tidak sesuai dengan input awal.");
                }
                return pieces;
```

```
            } else {
                throw new IllegalArgumentException("Maaf,
program ini hanya mendukung board jenis 'DEFAULT':(");
            }
        }
    }

    private static char findName(String line) {
        for (char c : line.toCharArray()) {
            if (c != ' ') return c;
        }
        throw new IllegalArgumentException("Pieces dalam file
tidak valid.");
    }

    public static char[][] createBoard(List<String> boardRow)
{
        int tinggi = boardRow.size();
        int lebar = boardRow.get(0).length();
        char[][] board = new char[tinggi][lebar];

        for (int r = 0; r < tinggi; r++) {
            for (int c = 0; c < lebar; c++) {
                board[r][c] = boardRow.get(r).charAt(c);
            }
        }
        return board;
    }

    public static void saveBoard(char[][] board, String
fileName) {
        try (BufferedWriter writer = new BufferedWriter(new
FileWriter(fileName))) {
            for (char[] row : board) {
                StringBuilder rowString = new StringBuilder();
                for (char now : row) {
                    if (now == '0') {
                        rowString.append(" ");
                    } else {
                        rowString.append(now);
                    }
                }
                writer.write(rowString.toString());
                writer.newLine();
            }
            System.out.println("Solusi berhasil tersimpan
dengan nama " + fileName + ".");
        } catch (IOException e) {
            System.out.println("Gagal menyimpan dalam bentuk
file." + e.getMessage());
        }
```

```
    }
}
```

IQPuzzlerSolver.java

```
import java.util.List;

public class IQPuzzlerSolver {
    private char[][] board;
    private List<Piece> pieces;
    private long totalCase = 0;

    public IQPuzzlerSolver(char[][] board, List<Piece> pieces)
{
        this.board = board;
        this.pieces = pieces;
    }

    public boolean solvePuzzle() {
        long start = System.currentTimeMillis();
        if (bruteForce(0) && penuhBoard()) {
            System.out.println("Solusi Puzzle:");
            printBoard();
            long end = System.currentTimeMillis();
            long totalTime = end - start;

            System.out.println("Waktu pencarian: " + totalTime
+ " ms");
            System.out.println();
            System.out.println("Banyak kasus yang ditinjau: "
+ totalCase);
            System.out.println();
            return true;
        } else {
            System.out.println("Maaf, solusi tidak dapat
ditemukan oleh program :(");
            return false;
        }
    }

    private void masukinPiece(char[][] shape, int row, int
col, char label) {
        for (int r = 0; r < shape.length; r++) {
            for (int c = 0; c < shape[0].length; c++) {
                if (shape[r][c] != ' ') {
                    board[row + r][col + c] = label;
                }
            }
        }
```

```
    }

    private void hapusPiece(char[][] shape, int row, int col)
{
        for (int r = 0; r < shape.length; r++) {
            for (int c = 0; c < shape[0].length; c++) {
                if (shape[r][c] != ' ') {
                    board[row + r][col + c] = '1';
                }
            }
        }
    }

    private boolean bruteForce(int index) {
        if (index == pieces.size()) {
            return true;
        }

        Piece piece = pieces.get(index);
        List<char[][]> transformations =
piece.getPieceVersions();

        for (char[][] shape : transformations) {
            for (int r = 0; r <= board.length - shape.length;
r++) {
                for (int c = 0; c <= board[0].length -
shape[0].length; c++) {
                    totalCase++;
                    if (valid(shape, r, c)) {
                        masukinPiece(shape, r, c,
piece.getName());
                        if (bruteForce(index + 1)) return
true;
                        hapusPiece(shape, r, c);
                    }
                }
            }
        }

        return false;
    }

    private boolean valid(char[][] shape, int row, int col) {
        for (int r = 0; r < shape.length; r++) {
            for (int c = 0; c < shape[0].length; c++) {
                if (shape[r][c] != ' ') {
                    if (board[row + r][col + c] != '1') {
                        return false;
                    }
                }
            }
```

```
        }
        return true;
    }

    private boolean penuhBoard() {
        for (int r = 0; r < board.length; r++) {
            for (int c = 0; c < board[0].length; c++) {
                if (board[r][c] == '1') {
                    return false;
                }
            }
        }
        return true;
    }

    private void printBoard() {
        String[] colors = {
            "\u001B[37m", "\u001B[31m", "\u001B[94m",
"\u001B[33m", "\u001B[95m", "\u001B[91m", "\u001B[32m",
"\u001B[93m", "\u001B[34m", "\u001B[96m", "\u001B[92m",
"\u001B[35m", "\u001B[36m", "\u001B[97m", "\u001B[90m",
"\u001B[38;5;208m", "\u001B[105m", "\u001B[38;5;51m",
"\u001B[106m", "\u001B[38;5;226m", "\u001B[101m",
"\u001B[104m", "\u001B[102m", "\u001B[103m", "\u001B[107m",
"\u001B[38;5;50m"
        };

        for (char[] row : board) {
            for (char cell : row) {
                if (cell == '0') {
                    System.out.print(" ");
                } else {
                    int colorIndex =
(Character.toUpperCase(cell) - 'A') % 26;
                    System.out.print(colors[colorIndex] + cell
+ "\u001B[0m");
                }
            }
            System.out.println();
        }
        System.out.println();
    }
}
```

Main.java

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

public class Main {
```

```
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Nama file input (tambahkan .txt):
");
        String fileName = scanner.nextLine().trim();

        fileName = "../test/input/" + fileName;

        int[] ukuran = new int[2];
        List<String> newBoard = new ArrayList<>();

        try {
            if (!new File(fileName).exists()) {
                throw new FileNotFoundException("File dengan
nama " + fileName + " tidak dapat ditemukan. Mohon cek kembali
nama file atau penempatannya.");
            }

            List<Piece> pieces =
InputOutput.bacaInput(fileName, ukuran, newBoard);
            char[][] board =
InputOutput.createBoard(newBoard);
            IQPuzzlerSolver solver = new
IQPuzzlerSolver(board, pieces);
            boolean solved = solver.solvePuzzle();

            if (solved) {
                while (true) {
                    System.out.print("Apakah Anda ingin
menyimpan jawaban Anda dalam file? (y/n): ");
                    String saveTxt =
scanner.nextLine().trim().toLowerCase();

                    if (saveTxt.equals("y")) {
                        String txtFilename;
                        while (true) {
                            System.out.print("Nama file output
(tambahkan .txt): ");
                            txtFilename =
scanner.nextLine().trim();

                            if
(txtFilename.matches("^[a-zA-Z0-9_\\-]+\\.txt$")) {
                                break;
                            } else {
                                System.out.println("Mohon
akhiri nama file dengan .txt.");
                            }
                        }
                        txtFilename = "../test/output/" +
```

```
txtFilename;
                        InputOutput.saveBoard(board,
txtFilename);
                        break;
                    } else if (saveTxt.equals("n")) {
                        break;
                    } else {
                        System.out.println("Mohon input y atau
n.");
                    }
                }

            }
        } catch (IOException e) {
            System.out.println("Mohon maaf, kesalahan telah
terjadi pada program. " + e.getMessage());
        } catch (IllegalArgumentException e) {
            System.out.println("Mohon maaf, kesalahan telah
terjadi pada program. " + e.getMessage());
        }
    }
}
```

D.

E. Tangkapan Layar

1. Test Case 1

Input:

```
5 5 7
DEFAULT
A
AA
B
BB
C
CC
D
DD
EE
EE
E
FF
FF
F
GGG
```

Gambar 1 Input 1 *(Sumber: Arsip Pribadi)*

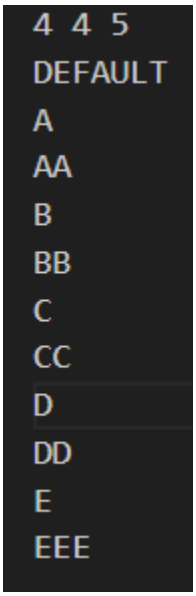
Output:

```
Nama file input (tambahkan .txt): 1.txt
Solusi Puzzle:
AABBD
AEBDD
EECCF
EECFF
GGGFF

Waktu pencarian: 17 ms

Banyak kasus yang ditinjau: 168552

Apakah Anda ingin menyimpan jawaban Anda dalam file? (y/n):
```

Gambar 2 Output 1 *(Sumber: Arsip Pribadi)*

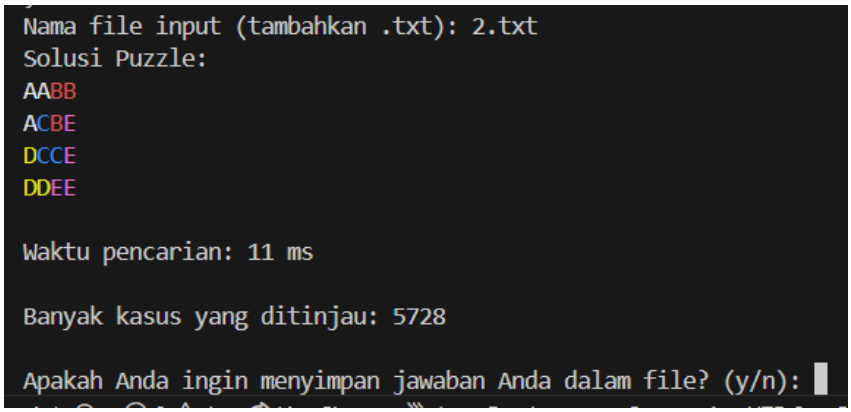
2. Test Case 2

Input:



Gambar 3 Input 2 *(Sumber: Arsip Pribadi)*

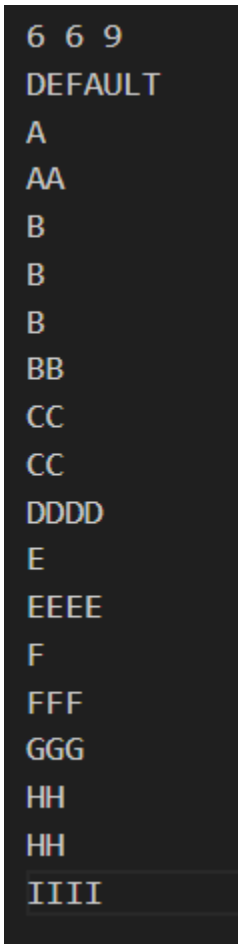
Output:



Gambar 4 Output 2 *(Sumber: Arsip Pribadi)*

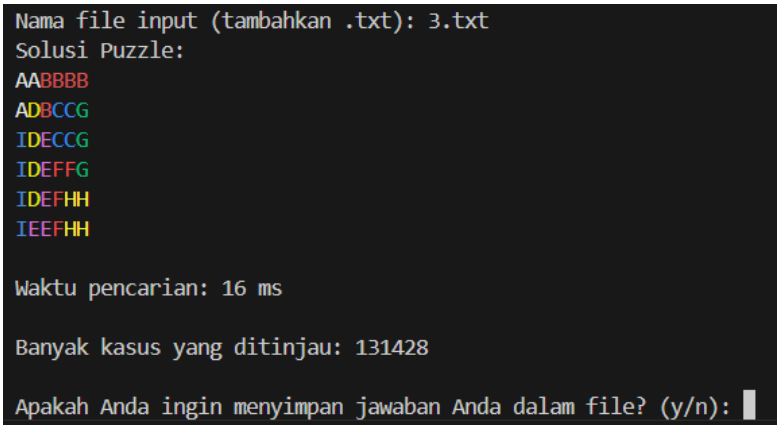
3. Test Case 3

Input:



Gambar 5 Input 3 *(Sumber: Arsip Pribadi)*

Output:



Gambar 6 Output 3 *(Sumber: Arsip Pribadi)*

4. Test Case 4

Input:

```
5 7 9
DEFAULT
AA
AA
B
BBBB
CCCC
D
DD
E
EEE
F
FF
GG
GGG
HH
IIIII
```

Gambar 7 Input 4 *(Sumber: Arsip Pribadi)*

Output:

```
Nama file input (tambahkan .txt): 4.txt
Solusi Puzzle:
AABBC
AABIC
DDBIC
DFBIC
FFGIE
HGGIE
HGGEE

Waktu pencarian: 34 ms

Banyak kasus yang ditinjau: 1660124

Apakah Anda ingin menyimpan jawaban Anda dalam file? (y/n): y
Nama file output (tambahkan .txt): output4.txt
Solusi berhasil tersimpan dengan nama ../test/output/output4.txt.
```

Gambar 8 Output 4 *(Sumber: Arsip Pribadi)*

5. Test Case 5

Input:

```
7 7 8
DEFAULT
A
AAA
AAA
BB
BB
B
B
CC
CCC
D
D
DDD
DDD
E
EEE
FFFF
FFFFF
G
GG
HHH
HHHH
```

Gambar 9 Input 5 *(Sumber: Arsip Pribadi)*

Output:

```
Nama file input (tambahkan .txt): 5.txt
Solusi Puzzle:
AAABBBB
AACCEBB
AACCEEE
HHCGDDD
HHGGDDD
HHFFFFD
HFFFFFD

Waktu pencarian: 21 ms

Banyak kasus yang ditinjau: 526195

Apakah Anda ingin menyimpan jawaban Anda dalam file? (y/n): y
Nama file output (tambahkan .txt): output5.txt
Solusi berhasil tersimpan dengan nama ../test/output/output5.txt.
```

Gambar 10 Output 5 *(Sumber: Arsip Pribadi)*

6. Test Case 6

Input:

```
8 8 8
DEFAULT
AAAAAAAA
AAAAA
B
BB
BBB
CC
CCC
DDDD
DDDD
DDDDDDD
EEE
EEEE
FF
FFF
GG
GG
HHH
HHH
HHH
```

Gambar 11 Input 6 *(Sumber: Arsip Pribadi)*

Output:

```
Nama file input (tambahkan .txt): 6.txt
HHHAADCC
HHHAADGG
BBBAADGG
BBEAADDD
BFEEADDD
FFEEADDD
FFEEADDD

Waktu pencarian: 11982 ms

Banyak kasus yang ditinjau: 2418794331

Apakah Anda ingin menyimpan jawaban Anda dalam file? (y/n): y
Nama file output (tambahkan .txt): output6.txt
Solusi berhasil tersimpan dengan nama ../test/output/output6.txt.
```

Gambar 12 Output 6 *(Sumber: Arsip Pribadi)*

7. Test Case 7

Input:

```
5 5 7
DEFAULT
A
AA
BBB
BB
C
CC
D
DDD
E
EEE
F
G
G
GGG
```

Gambar 13 Input 7 *(Sumber: Arsip Pribadi)*

Output:

```
Nama file input (tambahkan .txt): 7.txt
Solusi Puzzle:
AADBB
AEDBB
GEDDB
GEECC
GGGCF

Waktu pencarian: 90 ms

Banyak kasus yang ditinjau: 5823561

Apakah Anda ingin menyimpan jawaban Anda dalam file? (y/n): y
Nama file output (tambahkan .txt): output7
Mohon akhiri nama file dengan .txt.
Nama file output (tambahkan .txt): output7.txt
Solusi berhasil tersimpan dengan nama ../test/output/output7.txt.
```

Gambar 14 Output 7 *(Sumber: Arsip Pribadi)*

F. Pranala

Repository Tugas Kecil: https://github.com/allen2610/Tucil_13523059

G. Lampiran

No	Poin	Ya	Tidak
1	Program berhasil dikompilasi tanpa kesalahan	✓	
2	Program berhasil dijalankan	✓	
3	Solusi yang diberikan program benar dan mematuhi aturan permainan	✓	
4	Program dapat membaca masukan berkas .txt serta menyimpan solusi dalam berkas .txt	✓	
5	Program memiliki *Graphical User Interface* (GUI)		✓
6	Program dapat menyimpan solusi dalam bentuk file gambar		✓
7	Program dapat menyelesaikan kasus konfigurasi *custom*		✓
8	Program dapat menyelesaikan kasus konfigurasi Piramida (3D)		✓
9	Program dibuat oleh saya sendiri	✓	