

LAPORAN TUGAS KECIL 1

IF2211 STRATEGI ALGORITMA

Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force



Oleh Ghaisan Zaki Pratama

NIM : 10122078

Semester II Tahun 2024/2025

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2025

Daftar Isi

| | |
|----------------------------------|----|
| Daftar Isi | 2 |
| Algoritma Brute Force | 3 |
| <i>Source Program</i> | 3 |
| <i>Test Case</i> | 14 |
| <i>Pranala Repository</i> | 19 |
| Lampiran | 19 |

Algoritma Brute Force

Pada program ini, algoritma brute force diterapkan untuk mencoba semua kemungkinan penempatan *piece* ke *board*. Setiap *piece* diterapkan refleksi horizontal (terhadap sumbu-y), refleksi vertikal (terhadap sumbu-x), rotasi 90°, rotasi 180°, dan rotasi 270°. Semua hasil tersebut dimasukkan ke dalam suatu *list*. Kemudian, dilakukan *looping* untuk menempatkan *piece* tersebut ke dalam *board*. Penempatan *piece* dilakukan dengan *looping* sepanjang baris dan kolom, jika sesuai maka *piece* ditempatkan di sel-sel yang bersesuaian pada *board* sesuai dengan bentuk *piece*. Setelah itu, dilakukan proses rekursif sampai semua *piece* dapat ditempatkan pada *board* dengan posisi yang tidak saling tumpang tindih. Jika semua *piece* telah ditempatkan, dilakukan proses pengecekan apakah *board* sudah terisi penuh semua selnya. Jika semua sel pada board telah terisi penuh, akan dibuat *board* duplikat yang nantinya akan ditampilkan untuk *output*. Jika dalam proses penempatan suatu *piece* pada *board* sudah tidak memungkinkan (tidak ada kemungkinan penempatan *piece* karena akan tumpang tindih tetapi terdapat *piece* yang belum ditempatkan pada *board*), maka *piece* sebelumnya yang ditempatkan akan dihapus dari *board* dan proses pengisian dilanjutkan kembali dengan mengabaikan bentuk *piece* yang telah dihapus tersebut (dicoba berbagai variasi lain seperti refleksi dan rotasi). Saat solusi sudah ditemukan, maka akan ditampilkan output huruf berwarna disertai waktu penjalanan program (khusus pada bagian algoritma brute force) dan banyak iterasi yang dilakukan.

Source Program

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

public class Main {
    static long iterationCount = 0; // Menghitung jumlah kombinasi yang
dicek
    static char[][] solutionBoard = null; // Menyimpan board solusi jika
ditemukan

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Masukkan path file test case (.txt): ");
        String filePath = input.nextLine();
        File inputFile = new File(filePath);
        String inputName = inputFile.getName();

        String solutionFileName = "test/solusi_" + inputName; //
menghasilkan "test/solusi_testcase.txt"
```

```

        // Deklarasi variabel
        int N, M, P;
        String S;
        char[][] boardTemplate = null;
        List<PuzzlePiece> pieces = new ArrayList<>();

        // Membaca input file
        try (BufferedReader br = new BufferedReader(new
FileReader(filePath))) {
            // Membaca baris pertama (ukuran board dan banyaknya piece)
            String firstLine = br.readLine();
            String[] parts = firstLine.trim().split("\\s+");
            N = Integer.parseInt(parts[0]); // jumlah baris board
            M = Integer.parseInt(parts[1]); // jumlah kolom board
            P = Integer.parseInt(parts[2]); // jumlah puzzle piece
            // Mengecek apakah jumlah piece sesuai
            if (P > 26 || P <= 0) {
                System.out.println("Input tidak sesuai: jumlah pieces
harus 0 < P <= 26");
                return;
            }

            // Membaca baris kedua (jenis konfigurasi
DEFAULT/CUSTOM/PYRAMID)
            S = br.readLine().trim();

            // Membuat board
            boardTemplate = new char[N][M];
            if (S.equalsIgnoreCase("CUSTOM")) {
                // Baca N baris untuk konfigurasi custom
                for (int i = 0; i < N; i++) {
                    String boardLine = br.readLine();
                    boardTemplate[i] = boardLine.toCharArray();
                }
            } else {
                // Untuk DEFAULT atau PYRAMID, inisialisasi papan kosong
                (misalkan dengan '.')
                for (int i = 0; i < N; i++) {
                    Arrays.fill(boardTemplate[i], '.');
                }
            }
        }
    }
}

```

```

    }

    // Membaca sisa baris (bentuk-bentuk pieces)
    String line;
    // Selama ada baris yang tersisa
    while ((line = br.readLine()) != null) {
        line = line.trim();
        if (line.isEmpty()) {
            continue; // lewati baris kosong
        }
        // Anggap baris pertama dari sebuah piece
        char id = line.charAt(0); // Huruf yang mewakili piece
        List<String> shapeLines = new ArrayList<>();
        shapeLines.add(line);

        // Untuk menentukan apakah baris berikutnya juga bagian
dari piece yang sama,
        // cek apakah baris itu dimulai dengan huruf yang sama.
        br.mark(1000); // tandai posisi saat ini
        String nextLine = br.readLine();
        while (nextLine != null) {
            nextLine = nextLine.trim();
            if (nextLine.isEmpty()) {
                // Lewati baris kosong dan lanjutkan
                br.mark(1000);
                nextLine = br.readLine();
                continue;
            }
            if (nextLine.charAt(0) == id) {
                shapeLines.add(nextLine);
                br.mark(1000); // tandai posisi setelah membaca
baris ini

                nextLine = br.readLine();
            } else {
                // Baris baru sudah berbeda, maka sudah bagian
dari piece berikutnya.

                // Kembalikan ke posisi sebelum membaca nextLine.
                br.reset();
                break;
            }
        }
    }

```

```

    }

    // Ubah list shapeLines menjadi array String dan buat
PuzzlePiece

    String[] shapeArray = shapeLines.toArray(new String[0]);
    PuzzlePiece piece = new PuzzlePiece(id, shapeArray);
    pieces.add(piece);
}

} catch (Exception e) {
    System.out.println("Input tidak sesuai");
}

// Cek apakah boardTemplate sudah terisi
if (boardTemplate == null) {
    System.out.println("Board template tidak terisi. Periksa file
input.");
    return;
}

// Mulai brute force
long startTime = System.currentTimeMillis();
boolean foundSolution = BruteForce(0, pieces, boardTemplate);
long endTime = System.currentTimeMillis();
long elapsed = endTime - startTime;

if (!foundSolution) {
    System.out.println("Solusi tidak ditemukan.");
} else {
    printBoardColored(solutionBoard);
}

System.out.println("Waktu pencarian: " + elapsed + " ms");
System.out.println("Banyak kasus yang ditinjau: " +
iterationCount);

// Prompt untuk menyimpan solusi ke file .txt
System.out.print("Apakah anda ingin menyimpan solusi? (ya/tidak):
");

String jawab = input.nextLine();
if (jawab.equalsIgnoreCase("ya")) {
    try (PrintWriter out = new PrintWriter(solutionFileName)) {
        if (solutionBoard!=null){
            for (char[] solutionBoard1 : solutionBoard) {

```

```

        for (int j = 0; j < solutionBoard[0].length; j++)
        {
            out.print(solutionBoard1[j]);
        }
        out.println();
    }

    if (solutionBoard==null){
        out.println("Solusi tidak ditemukan");
    }
    out.println("Waktu pencarian: " + elapsed + " ms");
    out.println("Jumlah iterasi yang dicek: " +
iterationCount);
    System.out.println("Solusi telah disimpan ke file " +
solutionFileName);
    } catch (IOException e) {
        System.out.println("Gagal menyimpan file " +
solutionFileName + ": " + e.getMessage());
    }

    } else {
        System.out.println("Solusi tidak disimpan.");
    }
}

// Fungsi rekursif brute force:
// - index: piece ke-berapa yang akan ditempatkan
// - pieces: daftar pieces
// - board: papan saat ini
    public static boolean BruteForce(int index, List<PuzzlePiece>
pieces, char[][] board) {
        iterationCount++;
        // Jika semua piece sudah diproses, cek apakah board penuh
        if (index == pieces.size()) {
            if (isSolution(board)) {
                solutionBoard = cloneBoard(board);
                return true;
            }
            return false;
        }
    }
}

```

```

        PuzzlePiece piece = pieces.get(index);
        List<String[]> variants = piece.getVariants();

        // Coba setiap varian piece
        for (String[] variant : variants) {
            int r = variant.length;
            int c = variant[0].length();
            // Coba setiap posisi di board
            for (int row = 0; row <= board.length - r; row++) {
                for (int col = 0; col <= board[0].length - c; col++) {
                    if (BoardUtil.canPlace(board, variant, row, col))
                    {
                        // Tempatkan piece
                        BoardUtil.placePiece(board, variant, row, col,
piece.id);

                        // Lanjut ke piece berikutnya
                        if (BruteForce(index + 1, pieces, board)) {
                            return true;
                        }

                        // Backtrack: hapus piece
                        BoardUtil.removePiece(board, variant, row,
col);
                    }
                }
            }

            // Jika tidak ada yang cocok, return false
            return false;
        }

        // Mengecek apakah board telah terisi penuh (tidak ada sel '.')
        public static boolean isSolution(char[][] board) {
            for (char[] board1 : board) {
                for (int j = 0; j < board[0].length; j++) {
                    if (board1[j] == '.') {
                        return false;
                    }
                }
            }
            return true;
        }
    }
}

```



```

        }
    }
}

return true;
}

// Meng-clone board untuk simulasi penempatan
public static char[][] cloneBoard(char[][] board) {
    int rows = board.length;
    int cols = board[0].length;
    char[][] newBoard = new char[rows][cols];
    for (int i = 0; i < rows; i++) {
        newBoard[i] = board[i].clone();
    }
    return newBoard;
}

// Mencetak board dengan warna menggunakan peta warna 26 huruf (A-Z)
public static void printBoardColored(char[][] board) {
    Map<Character, String> colorMap = BoardUtil.getColorMap();
    String ANSI_RESET = "\u001B[0m";
    for (int i = 0; i < board.length; i++) {
        for (int j = 0; j < board[0].length; j++) {
            char ch = board[i][j];
            if (ch == '.') {
                System.out.print(ch + " ");
            } else {
                String color = colorMap.getOrDefault(ch, "");
                System.out.print(color + ch + ANSI_RESET + " ");
            }
        }
        System.out.println();
    }
}

// Kelas yang merepresentasikan satu piece
class PuzzlePiece {
    char id;          // Misalnya 'A'
    String[] shape;    // Representasi bentuk piece
}

```

```

public PuzzlePiece(char id, String[] shape) {
    this.id = id;
    this.shape = normalizeShape(shape);
}

private String[] normalizeShape(String[] shape) {
    int max = 0;
    for (String s : shape) {
        if (s.length() > max) {
            max = s.length();
        }
    }
    String[] norm = new String[shape.length];
    for (int i = 0; i < shape.length; i++) {
        // Mengisi dengan .
        norm[i] = String.format("%-" + max + "s", shape[i]).replace(' ', '.');
    }
    return norm;
}

// Mengembalikan semua varian piece
public List<String[]> getVariants() {
    List<String[]> variants = new ArrayList<>();
    variants.add(shape);
    String[] mirrorH = mirrorHorizontal(shape);
    variants.add(mirrorH);
    String[] mirrorV = mirrorVertical(shape);
    variants.add(mirrorV);
    String[] rot90 = rotateCounterClockwise(shape);
    variants.add(rot90);
    String[] rot90mirrorH = mirrorHorizontal(rot90);
    variants.add(rot90mirrorH);
    String[] rot90mirrorV = mirrorVertical(rot90);
    variants.add(rot90mirrorV);
    String[] rot180 = rotateCounterClockwise(rot90);
    variants.add(rot180);
    String[] rot180mirrorH = mirrorHorizontal(rot180);
    variants.add(rot180mirrorH);
    String[] rot180mirrorV = mirrorVertical(rot180);

```

```

        variants.add(rot180mirrorV);
        String[] rot270 = rotateCounterClockwise(rot180);
        variants.add(rot270);
        String[] rot270mirrorH = mirrorHorizontal(rot270);
        variants.add(rot270mirrorH);
        String[] rot270mirrorV = mirrorVertical(rot270);
        variants.add(rot270mirrorV);
        return variants;
    }

    // Fungsi rotasi counterclockwise 90 derajat.
    // Untuk setiap kolom dari kanan ke kiri di input,
    // ambil karakter dari tiap baris (dari atas ke bawah) untuk membentuk
    baris baru.
    public String[] rotateCounterClockwise(String[] piece) {
        int rows = piece.length;
        int cols = piece[0].length();
        String[] rotated = new String[cols]; // Hasilnya memiliki jumlah
    baris = jumlah kolom input.
        for (int i = 0; i < cols; i++) {
            StringBuilder sb = new StringBuilder();
            for (int j = 0; j < rows; j++) {
                // Ambil karakter dari baris j, kolom (cols - 1 - i)
                sb.append(piece[j].charAt(cols - 1 - i));
            }
            rotated[i] = sb.toString();
        }
        return rotated;
    }

    // Refleksi horizontal: membalik setiap baris (flip terhadap sumbu y)
    public String[] mirrorHorizontal(String[] piece) {
        int rows = piece.length;
        String[] mirrored = new String[rows];
        for (int i = 0; i < rows; i++) {
            // Balikkan string tiap baris
            mirrored[i] = new
    StringBuilder(piece[i]).reverse().toString();
        }
        return mirrored;
    }

```

```

// Refleksi vertical: membalik urutan baris (flip terhadap sumbu x)
public String[] mirrorVertical(String[] piece) {
    int rows = piece.length;
    String[] mirrored = new String[rows];
    for (int i = 0; i < rows; i++) {
        mirrored[i] = piece[rows - 1 - i];
    }
    return mirrored;
}

// Kelas utilitas untuk operasi pada board
class BoardUtil {
    public static boolean canPlace(char[][] board, String[] variant, int
row, int col) {
        int r = variant.length;
        int c = variant[0].length();
        if (row + r > board.length || col + c > board[0].length) {
            return false;
        }
        for (int i = 0; i < r; i++) {
            for (int j = 0; j < c; j++) {
                char ch = variant[i].charAt(j);
                if (ch != '.' && board[row + i][col + j] != '.') {
                    return false;
                }
            }
        }
        return true;
    }

    public static void placePiece(char[][] board, String[] variant, int
row, int col, char id) {
        int r = variant.length;
        int c = variant[0].length();
        for (int i = 0; i < r; i++) {
            for (int j = 0; j < c; j++) {
                if (variant[i].charAt(j) != '.') {
                    board[row + i][col + j] = id;
                }
            }
        }
    }
}

```

```

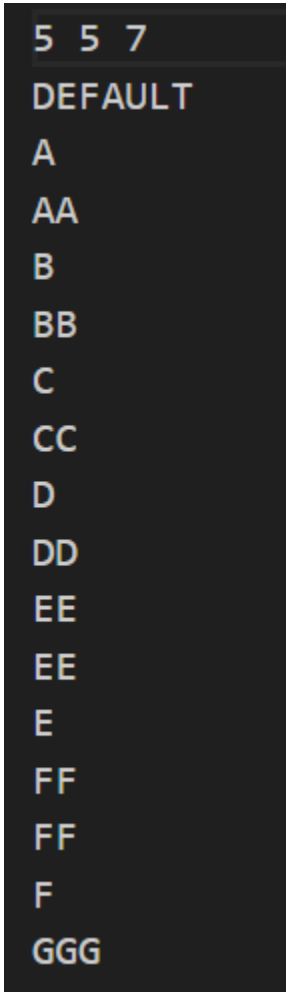
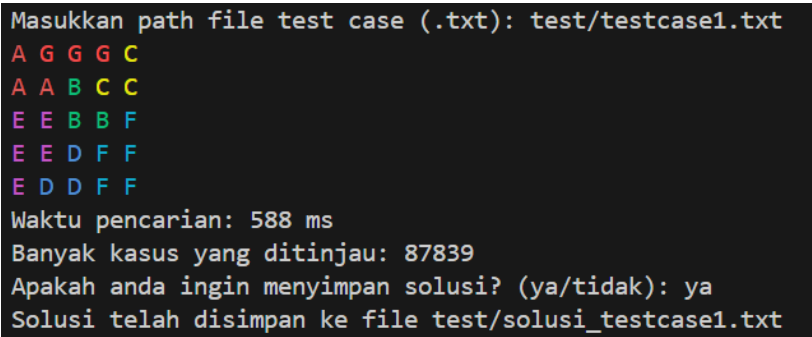
    }
}

// Fungsi removePiece untuk backtrack
public static void removePiece(char[][] board, String[] variant, int
row, int col) {
    int r = variant.length;
    int c = variant[0].length();
    for (int i = 0; i < r; i++) {
        for (int j = 0; j < c; j++) {
            if (variant[i].charAt(j) != '.') {
                board[row + i][col + j] = '.';
            }
        }
    }
}

public static Map<Character, String> getColorMap() {
    Map<Character, String> colorMap = new HashMap<>();
    String[] colors = {
        "\u001B[31m", "\u001B[32m", "\u001B[33m", "\u001B[34m",
        "\u001B[35m", "\u001B[36m",
        "\u001B[91m", "\u001B[92m", "\u001B[93m", "\u001B[94m",
        "\u001B[95m", "\u001B[96m",
        "\u001B[31;1m", "\u001B[32;1m", "\u001B[33;1m",
        "\u001B[34;1m", "\u001B[35;1m", "\u001B[36;1m",
        "\u001B[37;1m", "\u001B[90m", "\u001B[91;1m", "\u001B[92;1m",
        "\u001B[93;1m", "\u001B[94;1m",
        "\u001B[95;1m", "\u001B[96;1m"
    };
    for (char ch = 'A'; ch <= 'Z'; ch++) {
        int index = ch - 'A';
        colorMap.put(ch, colors[index]);
    }
    return colorMap;
}
}

```

Test Case

| No | Input | Output |
|----|--------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| 1 |  <p>Gambar 1. Input <i>testcase1.txt</i></p> |  <p>Gambar 2. Output <i>testcase1.txt</i></p> |

2

5 11 8

AA

AA

A

BB

BB

B

CC

DDDD

DDD

E

EEEE

FFFFFFFFFF

FFFFFFFFFF

GGG

GG

GGG

HH

HHH

HH

Gambar 3. Input
testcase2.txt

Masukkan path file test case (.txt): testcase2.txt

A A H C D D D G G G

A H H H E D D D D G G

B H H H E E E G G G

B B F F F F F F F F F

B B F F F F F F F F F

Waktu pencarian: 5017039 ms

Banyak kasus yang ditinjau: 428033762

Apakah anda ingin menyimpan solusi? (ya/tidak): ya

AAHCCDDGGG

AHHHEDDDGG

BHHHEEEGGG

BBFFFFFFFFF

BBFFFFFFFFF

Waktu pencarian: 5017039 ms

Jumlah iterasi yang dicek: 428033762

Solusi telah disimpan ke file test/solusi_testcase2.txt

PS C:\Users\HP\OneDrive\Documents\Stima\Tucil1>

Gambar 4. Output testcase2.txt

3

```
4 5 5
DEFAULT
S
S
S
S
S
T
T
T
T
IIII
MMMM
AAAA
```

Gambar 5. Input
testcase3.txt

```
Masukkan path file test case (.txt): test/testcase3.txt
S T I M A
S T I M A
S T I M A
S T I M A
Waktu pencarian: 2 ms
Banyak kasus yang ditinjau: 6
Apakah anda ingin menyimpan solusi? (ya/tidak): ya
Solusi telah disimpan ke file test/solusi testcase3.txt
```

Gambar 6. Output *testcase3.txt*

4

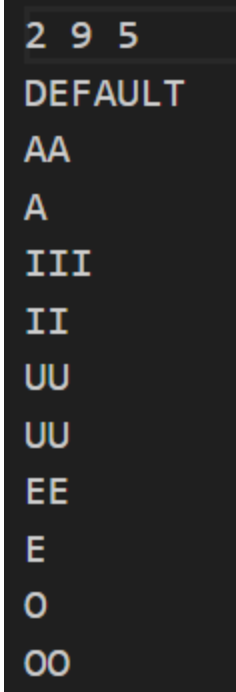
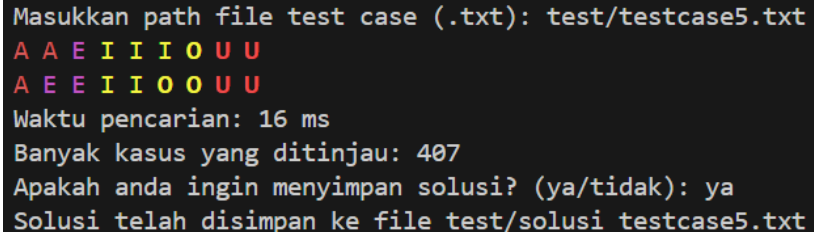
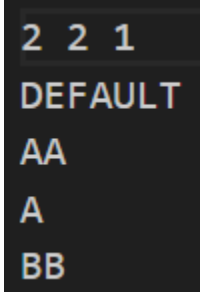
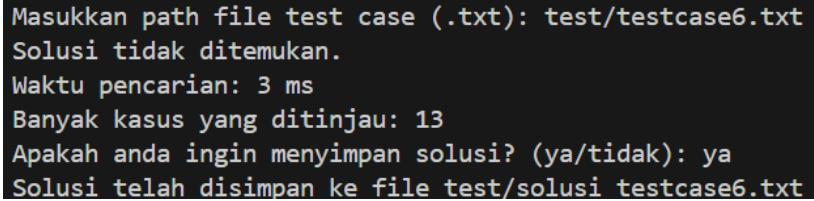
2 13 26
DEFAULT

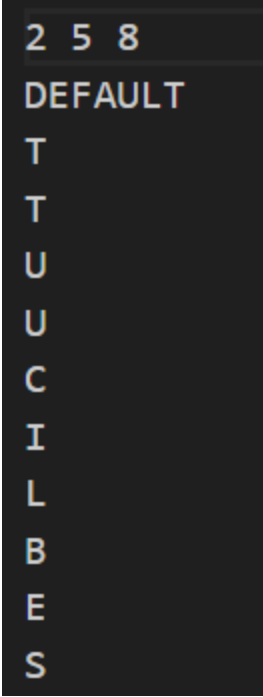
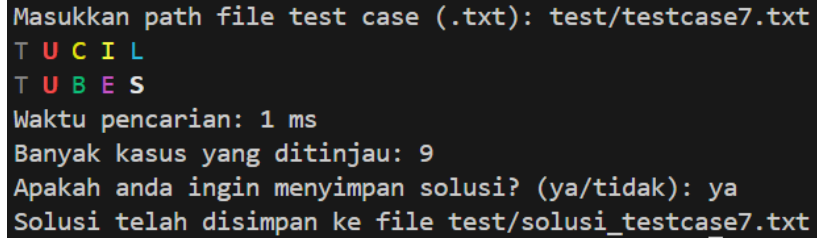
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

Gambar 7. Input
testcase4.txt

```
Masukkan path file test case (.txt): test/testcase4.txt
A B C D E F G H I J K L M
N O P Q R S T U V W X Y Z
Waktu pencarian: 3 ms
Banyak kasus yang ditinjau: 27
Apakah anda ingin menyimpan solusi? (ya/tidak): ya
Solusi telah disimpan ke file test/solusi_testcase4.txt
```

Gambar 8. Output *testcase4.txt*

| | | |
|----------|----------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------|
| <p>5</p> |  <p>Gambar 9. Input <i>testcase5.txt</i></p> |  <p>Gambar 10. Output <i>testcase5.txt</i></p> |
| <p>6</p> |  <p>Gambar 11. Input <i>testcase6.txt</i></p> |  <p>Gambar 12. Output <i>testcase6.txt</i></p> |

| | | |
|---|--------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------|
| 7 |  <p>Gambar 13. Input <i>testcase7.txt</i></p> |  <p>Gambar 7. Output <i>testcase14.txt</i></p> |
|---|--------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------|

Pranala Repository

https://github.com/GhaisanZP/Tucil1_10122078

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 | ✓ | |