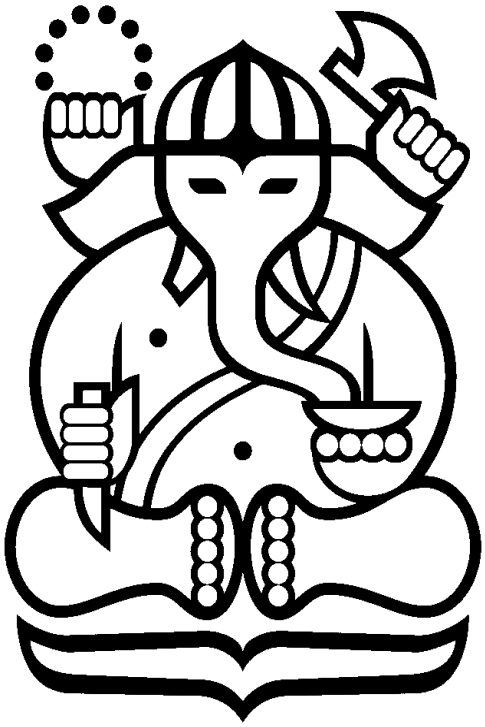
LAPORAN TUGAS KECIL 1
IF2211 STRATEGI ALGORITMA

Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force



Disusun oleh :

13523041 – Hanif Kalyana Aditya

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2025

Daftar Isi

BAB I
DESKRIPSI MASALAH DAN ALGORITMA

1.1 Algoritma Brute Force

Penyelesaian sebuah masalah pada dasarnya dapat dipecahkan oleh cara tertentu. Dalam menyelesaikan masalah tersebut, terdapat sebuah cara atau algoritma yang sangat sering digunakan. Algoritma tersebut adalah Algoritma Brute Force. Algoritma Brute Force adalah salah satu metode pendekatan yang cukup *straightforward* dan dapat memecahkan persoalan dengan sangat sederhana, langsung, serta jelas cara pemecahannya (*obvious way*). Algoritma Brute Force memerlukan komputasi yang besar dan waktu untuk menyelesaikan masalah yang cukup lama sehingga penyelesaian masalahnya menjadi tidak mangkus dan sangkil. Akan tetapi, jika digunakan dalam memecahkan persoalan yang cenderung kecil, penggunaan Algoritma Brute Force dinilai cukup tepat karena kemampuannya untuk menjangkau seluruh kemungkinan dan cara penyelesaiannya yang cukup mudah dipahami. Selain mudah dipahami cara penyelesaiannya dan terlepas dari kelemahannya, algoritma ini ibarat algoritma yang dapat digunakan pada permasalahan apapun jika tidak ada algoritma lain yang lebih baik.

1.2 IQ Puzzler Pro

IQ Puzzler Pro adalah sebuah permainan sederhana yang melibatkan papan dan beberapa blok. Tujuan dari permainan ini adalah menyusun seluruh blok yang tersedia di dalam sebuah papan sehingga papan itu penuh. Bentuk dari blok-blok tersebut juga bervariasi dan papan permainan pada umumnya berbentuk persegi panjang atau kotak (bentuk piramida dan *custom* tidak dibahas). Untuk menyelesaikan permasalahan tersebut, blok digambarkan sebagai suatu titik atau sel di dalam sebuah matriks yang berfungsi sebagai papan. Permainan akan gagal jika blok yang digunakan melebihi batas papan atau saling tumpang tindih antar blok. Selain itu, dalam permainan ini terdapat kemungkinan tidak ada solusi yang dapat disebabkan oleh jumlah blok terlalu banyak atau bentuk blok melebihi batas papan.

1.3 Algoritma Penyelesaian IQ Puzzler Pro dengan Pendekatan Brute Force

Dalam menyelesaikan IQ Puzzler Pro secara algoritmik, penulis menggunakan pendekatan brute force. Berikut langkah penyelesaian permasalahan dalam algoritma brute force :

1. Pengguna memilih menu yaitu antara “Load File” atau “Keluar” dan apabila memilih

"Load File" pengguna akan diharuskan untuk memberi masukan berupa nama file (berekstensi .txt).

2. Selanjutnya adalah *processing file* (apabila file ditemukan). Format masukan file terdiri dari tiga interger pertama yang menyatakan panjang dan lebar papan permainan serta banyaknya blok. Sementara itu, juga terdapat metode/jenis permainan dan dalam hal ini hanya dibahas mengenai jenis permainan *default*. Setelah itu, blok dibaca satu persatu dan jika dalam sebuah baris huruf berganti, blok sebelumnya disimpan dan akan dibaca blok baru. Hasil dari proses ini adalah *array of block*.
3. Kemudian, program akan melanjutkan dengan menyelesaikan persoalan. Penyelesaian ini pada dasarnya dimulai dengan blok pertama dalam *array of block* lalu dicari beberapa kemungkinan bentuk blok dengan cara rotasi dan refleksi (setiap blok memiliki delapan buah transformasi). Lalu blok tersebut diletakkan ke dalam papan dan apabila memungkinkan maka setiap sel dalam papan matrik yang berkoresponden dengan blok akan diubah isinya menjadi huruf dari blok tersebut sebagai representasi blok yang sudah mengisi.
4. Proses kemudian berlanjut untuk setiap blok yang telah ditransformasi dari setiap blok yang tersedia dan proses ini melibatkan metode rekursi. Apabila ditemukan suatu kasus dimana blok tidak dapat diletakkan dalam papan, program akan melakukan *backtracking* dan lanjut untuk variasi blok-blok selanjutnya. Proses rekursi dan *backtracking* ini mengandalkan boolean serta akan terus berlanjut hingga blok terakhir. Akhir dari proses ini adalah boolean "true or false" mengenai kemungkinan adanya solusi untuk kasus yang diberikan
5. Permainan dapat dikatakan terdapat solusi apabila semua blok sudah digunakan dan seluruhnya menempati papan sehingga tidak ada sel kosong di dalam papan. Apabila terdapat solusi, program akan menampilkan kondisi akhir papan yang sudah terselesaikan beserta keterangan berapa banyak kasus yang ditinjau dan waktu yang diperlukan untuk menyelesaikan persoalan.
6. Terakhir, program akan menawarkan untuk melakukan penyimpanan ke dalam file berekstensi .txt.

BAB II
IMPLEMENTASI ALGORITMA DALAM BAHASA JAVA

Dalam pembuatan program ini, penulis menggunakan bahasa pemrograman Java. Struktur dari program ini terbagi menjadi tiga file, yaitu file Main.java, Block.java, dan Solver.java dengan pembagian fungsi sebagai berikut :

2.1 File Main.java

File ini merupakan bagian utama dalam program ini dan berfungsi untuk memulai program, *interface* program, dan menyimpan solusi dari persoalan

2.2 File Block.java

File ini berisi fungsi-fungsi atau hal lain yang berkaitan dengan objek/kelas *block*. Antara lain :

Fungsi	Deskripsi
getAllTransformations()	Membuat seluruh kemungkinan transformasi dari sebuah blok dengan melibatkan fungsi rotasi dan refleksi. Keluaran dari fungsi ini adalah *array of matrix of integer* (List<int[][]>) dengan total delapan transformasi untuk masing-masing blok.
rotateBlock(int[][] shape)	Membuat sebuah blok berotasi 90 derajat searah jarum jam
reflectBlock(int[][] shape)	Membuat sebuah blok terefleksi secara horizontal
getId()	Mengembalikan Id atau (char) huruf dari sebuah blok
getShape()	Mengembalikan bentuk/*shape* (int[][]) dari sebuah blok
getColor(char id)	Mengembalikan warna dari sebuah blok berdasarkan huruf atau Id-nya

2.3 File Solver.java

File ini berisi fungsi-fungsi yang berkaitan dengan *processing file* masukan, penyelesaian persoalan, dan menampilkan solusi dalam bentuk papan matriks.

Fungsi / Prosedur	Deskripsi
loadFile	Membaca terhadap file masukan dan menyimpannya
addBlock	Menambahkan blok baru ke dalam *array of block*
getColumnLenght	Mendapatkan panjang baris/kolom
getCharFromString	Mendapatkan huruf yang terdapat dalam baris tersebut
solve	Menghubungkan program utama dengan penyelesaian persoalan
placeBlock	Menyelesaikan persoalan dengan metode rekursi
canPlace	Mengecek apakah sebuah blok dapat diletakkan ke dalam papan
place	Meletakkan blok ke dalam papan dengan mengubah isi sel dengan huruf atau Id dari blok tersebut
remove	Menghapus blok yang telah diletakkan ke dalam papan
isLineValid	Mengecek apakah line valid atau tidak
printSolution	Menampilkan papan matriks sebagai solusi persoalan
getCheckedCases	Mendapatkan banyaknya kasus yang ditinjau
getBoard	Mengembalikan papan permainan
getBoardCol	Mengembalikan lebar atau jumlah kolom papan permainan
getBoardRow	Mengembalikan panjang atau jumlah baris papan permainan
getCell	Mengembalikan sel dari matriks papan permainan
isBoardFull	Mengecek apakah seluruh sel dari papan telah terisi

BAB III
SOURCE CODE PROGRAM

3.1 Repository Program

Repository Program dapat diakses melalui tautan *Github* berikut:
https://github.com/hnfadtya/Tucil1_13523041

3.2 Source Code Program

3.2.1 Main.java

```
package src;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

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

        while (true) {
            printHeader();

            System.out.println("[1] Load File");
            System.out.println("[2] Keluar");
            System.out.print("Pilih opsi: ");
            String pilihan = scanner.nextLine();

            if (pilihan.equals("2")) {
                printFooter();
                break;
            } else if (!pilihan.equals("1")) {
                System.out.println("Pilihan tidak valid!\n");
                continue;
            }

            System.out.print("\nMasukkan nama file (dalam folder
    test/): ");
            String filename = scanner.nextLine().trim();
            File inputFile = new File("test/" + filename);

            if (!inputFile.exists()) {
                System.out.println("Error: File '" + filename + "'
    tidak ditemukan!\n");
                continue;
            }
```

```
            System.out.println("\nMemuat file: " + filename);
            long start = System.currentTimeMillis();

            Solver solver = new Solver();
            try {
                solver.loadFile(inputFile.getAbsolutePath());
            } catch (IOException e) {
                System.out.println("Gagal memuat file: " +
e.getMessage());
                continue;
            }

            boolean solution = solver.solve();
            long end = System.currentTimeMillis();
            System.out.printf("Waktu pencarian: " + (end - start) +
"ms\n\n");
            System.out.println("Banyak kasus yang ditinjau: " +
solver.getCheckedCases());

            if (solution && solver.isBoardFull()) {
                System.out.println("\nSolusi ditemukan!");
                solver.printSolution();
                System.out.print("\nApakah anda ingin menyimpan solusi?
(ya/tidak): ");
                String saveSolution = scanner.nextLine().trim();

                if (saveSolution.equals("ya")) {
                    char[][] board = solver.getBoard();
                    int row = solver.getBoardRow();
                    int col = solver.getBoardCol();
                    saveSolutionToFile(board, filename, row, col);
                } else if (!saveSolution.equals("tidak")) {
                    System.out.println("Pilihan tidak valid!\n");
                    // continue;
                }
            } else {
                System.out.println("\nTidak ada solusi yang ditemukan.");
            }

        }

        scanner.close();
    }

    private static void printHeader() {
        System.out.println("=======================================");
        System.out.println("        IQ PUZZLER PRO SOLVER          ");
        System.out.println("=======================================");
    }
```

```
    private static void printFooter() {
        System.out.println("\n=======================================");
        System.out.println("        TERIMA KASIH TELAH COBA!      ");
        System.out.println("=======================================");
    }

    private static void saveSolutionToFile(char[][] board, String
inputFilename, int row, int col) {
        String outputFilename = "test/output/" +
inputFilename.replace(".txt", "_solution.txt");
        File outputFile = new File(outputFilename);

        try {
            outputFile.getParentFile().mkdirs();
            FileWriter writer = new FileWriter(outputFile);

            for (int i = 0; i < row; i++) {
                for (int j = 0; j < col; j++) {
                    writer.write(board[i][j] + " ");
                }
                writer.write("\n");
            }

            writer.close();
            System.out.println("Solusi disimpan ke: " + outputFilename);
        } catch (IOException e) {
            System.out.println("Gagal menyimpan solusi: " +
e.getMessage());
        }
    }
}
```

3.2.1. Block.Java

```
package src;

import java.util.*;

public class Block {
    private char id;
    private int[][] shape;
    public static final String RESET = "\u001B[0m";
    public static final 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[90m",
        "\u001B[37m",
        "\u001B[97m",
        "\u001B[41m",
        "\u001B[42m",
        "\u001B[43m",
        "\u001B[45m",
        "\u001B[46m",
        "\u001B[44m",
        "\u001B[100m",
        "\u001B[101m",
        "\u001B[102m",
        "\u001B[103m",
        "\u001B[104m"
    };

    public Block(char id, int[][] shape) {
        this.id = id;
        this.shape = shape;
    }

    public char getId() {
        return id;
    }

    public int[][] getShape() {
        return shape;
    }
```

```
public String getColor(char id) {
    if(id >= 'A' && id <= 'Z') {
        return COLORS[id - 'A'];
    }
    return RESET;
}

public List<int[][]> getAllTransformations() {
    List<int[][]> transformations = new ArrayList<>();
    transformations.add(shape);
    transformations.add(rotateBlock(shape));
    transformations.add(rotateBlock(transformations.get(1)
   ));
    transformations.add(rotateBlock(transformations.get(2)
   ));

    int[][] flipped = reflectBlock(shape);
    transformations.add(flipped);
    transformations.add(rotateBlock(flipped));
    transformations.add(rotateBlock(transformations.get(5)));
    transformations.add(rotateBlock(transformations.get(6)));

    return transformations;
}

private int[][] rotateBlock(int[][] shape) {
    int rows = shape.length;
    int cols = shape[0].length;
    int[][] rotated = new int[cols][rows];

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            rotated[j][rows - 1 - i] = shape[i][j];
        }
    }
    return rotated;
}

private int[][] reflectBlock(int[][] shape) {
    int rows = shape.length;
    int cols = shape[0].length;
    int[][] reflected = new int[rows][cols];

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            reflected[i][cols - 1 - j] = shape[i][j];
        }
    }
    return reflected;
}
```

3.2.3. Solver.java

```
package src;

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

public class Solver {
    private int width, height;
    private List<Block> blocks = new ArrayList<>();
    private char[][] board;
    private int checkedCases = 0;
    private boolean solutionFound = false;
    public static final String RESET = "\u001B[0m";

    /* ============== FILE PROCESSING ================*/
    public void loadFile(String filename) throws
IOException {
        BufferedReader br = new BufferedReader(new
FileReader(filename));
        String[] size = br.readLine().split(" ");
        width = Integer.parseInt(size[0]);
        height = Integer.parseInt(size[1]);
        int numBlocks = Integer.parseInt(size[2]);

        br.readLine(); // Abaikan baris DEFAULT

        List<String> shapeLines = new ArrayList<>();
        char blockId = ' ';
        char prevChar = ' ';

        String line;
        while ((line = br.readLine()) != null &&
blocks.size() < numBlocks) { // defaultnya program
berhenti membaca blok ketika jumlahnya sudah sesuai
dengan numBlocks
            line = line.trim();
            if (line.isEmpty()) continue; // baris kosong

            if (prevChar != ' ' && line.charAt(0) !=
prevChar) {
                addBlock(blockId, shapeLines); //
menyimpan blok sebelumnya
                shapeLines.clear();
            }

            if (shapeLines.isEmpty()) {
                blockId = getCharFromString(line);
            }

            shapeLines.add(line);
            System.out.println(shapeLines);
            prevChar = getCharFromString(line);
```

```
        }

        // Simpan blok terakhir dalam file jika masih ada
data
        if (!shapeLines.isEmpty()) {
            addBlock(blockId, shapeLines);
        }

        if (blocks.size() < numBlocks) {
            IOException e = new IOException("Jumlah blok
tidak sesuai");
            throw e;
        }

        br.close();
    }

    private void addBlock(char id, List<String>
shapeLines) {
        int rows = shapeLines.size();
        int cols = getColumnLenght(shapeLines, rows);
        System.out.println(cols);
        int[][] shape = new int[rows][cols];

        for (int r = 0; r < rows; r++) {
            int lineLenght = shapeLines.get(r).length();
            for (int c = 0; c < lineLenght; c++) {
                if (shapeLines.get(r).charAt(c) == id)
{shape[r][c] = 1;} else shape[r][c] = 0;
            }
        }
        // System.out.println(id);
        blocks.add(new Block(id, shape));
    }

    private int getColumnLenght(List<String> shapeLines,
int rows) {
        int columnLenght = shapeLines.get(0).length();
        for (int i = 1; i < rows; i++) {
            if (shapeLines.get(i).length() >
columnLenght) {
                columnLenght =
shapeLines.get(i).length();
            }
        }
        return columnLenght;
    }

    private char getCharFromString(String string) {
        int index = 0;
        while (index < string.length()){
            if (string.charAt(index) != ' ') {
```

```
                return string.charAt(index);
            }
        }
        return ' ';
    }
    private boolean isLineValid(String line, char id) {
        for (int i = 0; i < line.length(); i++) {
            char c = line.charAt(i);
            if (c != ' ' && c != id) {
                return false;
            }
        }
        return true;
    }

    /* ============== PROBLEM SOLVING ==================*/
    public boolean solve() {
        board = new char[height][width];
        for (char[] row : board) Arrays.fill(row, '.');

        return placeBlock(0);
    }

    private boolean placeBlock(int index) {
        if (solutionFound) return true;

        if (index == blocks.size()) { // basis rekursi
            solutionFound = true;
            return true;
        }

        Block block = blocks.get(index);
        List<int[][]> transformations =
block.getAllTransformations();

        for (int[][] shape : transformations) {
            for (int r = 0; r <= height - shape.length;
r++) {
                for (int c = 0; c <= width -
shape[0].length; c++) {
                    checkedCases++;
                    if (canPlace(shape, r, c)) {
                        place(shape, r, c,
block.getId());
                        if (placeBlock(index + 1)) return
true;
                        remove(shape, r, c); // backtrack
jika tidak bisa ditempatkan
                    }
                }
            }
        }
```

```
        return false;
    }

    private boolean canPlace(int[][] shape, int r, int c)
{
        for (int i = 0; i < shape.length; i++) {
            for (int j = 0; j < shape[i].length; j++) {
                if (shape[i][j] == 1 && board[r + i][c +
j] != '.') {

                    return false;
                }
            }
        }
        return true;
    }

    private void place(int[][] shape, int r, int c, char
id) {
        for (int i = 0; i < shape.length; i++) {
            for (int j = 0; j < shape[i].length; j++) {
                if (shape[i][j] == 1) {
                    board[r + i][c + j] = id;
                }
            }
        }
    }

    private void remove(int[][] shape, int r, int c) {
        for (int i = 0; i < shape.length; i++) {
            for (int j = 0; j < shape[i].length; j++) {
                if (shape[i][j] == 1) {
                    board[r + i][c + j] = '.';
                }
            }
        }
    }

    public void printSolution() {
        for (int i = 0; i < board.length; i++) {
            for (int j = 0; j < board[i].length; j++) {
                char cell = board[i][j];
                Block block = blocks.get(cell - 'A');
                String color = block.getColor(cell);
                System.out.print(color + cell + " " +
RESET);
            }
            System.out.println();
        }
    }
```

```
    public int getCheckedCases() {
        return checkedCases;
    }

    /* ============== BOARD ==================*/
    public char[][] getBoard() {
        return board;
    }

    public int getBoardCol() {

        return width;
    }

    public int getBoardRow() {
        return height;
    }

    public char getCell(int r, int c) {
        return board[r][c];
    }

    public boolean isBoardFull() {
        for (int i = 0; i < getBoardCol(); i++) {
            for (int j = 0; j < getBoardRow(); j++) {
                char cell = getCell(i, j);
                if (cell == '.') return false;
            }
        }
        return true;
    }
}
```

BAB IV
MASUKAN DAN LUARAN PROGRAM

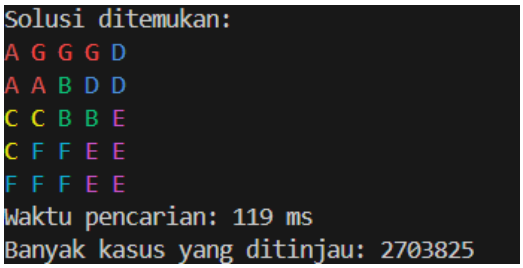
4.1 Input File

a. Kasus yang sesuai spesifikasi

Masukan:

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

Keluaran:



b. Kasus tidak ada solusi

Masukan:

```
5 5 7
DEFAULT
A
A
B
B
C
C
D
D
E
E
E
F
F
F
G
```

Keluaran:

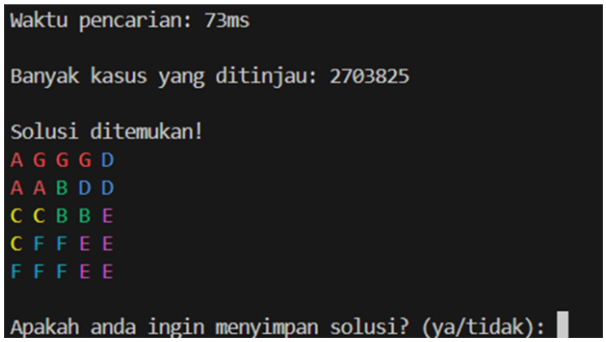
```
Waktu pencarian: 6ms

Banyak kasus yang ditinjau: 38

Tidak ada solusi yang ditemukan.
=====================================
        IQ PUZZLER PRO SOLVER
=====================================
```

c. Blok A pada line pertama hurufnya tidak rata kiri

Masukan:

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

Keluaran:



d. Kasus blok kurang dari masukan di awal (ada enam blok)

Masukan:

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

Keluaran:

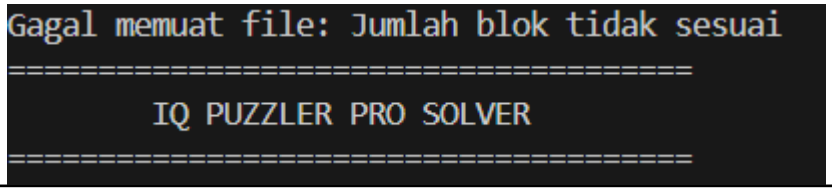
```
Gagal memuat file: Jumlah blok tidak sesuai
```

e. Kasus blok melebihi dari masukan di awal (ada delapan blok)

Masukan:

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

Keluaran:



f. Kasus jumlah blok tidak ada

Masukan:

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

Keluaran:

```
Masukkan nama file (dalam folder test/): testcase6.txt

Memuat file: testcase6.txt
Gagal memuat file: Ukuran papan atau jumlah blok tidak valid
====================================
        IQ PUZZLER PRO SOLVER
------------------------------------
```

g. Kasus blok tidak valid

Masukan:

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

Keluaran:

```
Masukkan nama file (dalam folder test/): testcase7.txt

Memuat file: testcase7.txt
Gagal memuat file: Blok ([AB]) tidak valid
```

BAB V
LAMPIRAN

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	✓	

Referensi

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/02-Algoritma-Brute-Force-(2025)-Bag1.pdf