LAPORAN TUGAS KECIL I
IF2211 STRATEGI ALGORITMA

Penyelesaian IQ Puzzler dengan Algoritma Brute Force

Disusun oleh:
Boye Mangaratua Ginting
13523127

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2025

Daftar Isi

A. Algoritma Brute Force

Program ini menggunakan algoritma **Brute Force yang dioptimalkan dengan Backtracking** untuk menyusun blok-blok dalam papan permainan **IQ Puzzler**. Penjelasan lengkap mengenai algoritma yang digunakan dalam menyusun blok-blok adalah sebagai berikut :

1. Dilakukan validasi input, dimulai dari validasi papan, jumlah blok, jumlah luas keseluruhan blok yang dimasukkan, blok duplikat, dan kemungkinan-kemungkinan kesalahan input lainnya.
2. Peletakan blok pada papan dilakukan secara berurut, dimulai dari blok pertama (A) dengan konfigurasi awal, kemudian dilanjut blok kedua (B) dan seterusnya.
3. Apabila ada blok yang tidak dapat diletakkan pada bagian paling kiri atas, maka akan digeser ke kanan sebanyak satu satuan hingga tidak dapat diletakkan ke bagian paling kanan yang mungkin. Jika kasus ini terjadi maka akan diletakkan ulang pada bagian paling kiri dan diturunkan satu satuan ke bawah. Kemudian langkah ini diulangi lagi hingga blok terakhir dapat diletakkan atau konfigurasi blok itu tidak dapat diletakkan di manapun.
4. Jika blok terakhir tidak dapat diletakkan maka konfigurasi blok diubah dengan dirotasikan 90 derajat searah jarum jam dan kembali ke langkah 2. Jika masih tidak dapat, dilakukan pencerminan hasil rotasi 90 derajat searah jarum jam. Urutan perubahan konfigurasi blok selanjutnya adalah rotasi 180 derajat, pencerminan rotasi 180 derajat, rotasi 270 derajat searah jarum jam, dan pencerminan 270 derajat searah jarum jam.
5. Jika semua konfigurasi masih tidak dapat untuk meletakkan blok di manapun, maka dilakukan backtracking untuk blok sebelumnya dengan mengulang ke langkah 2 pada blok sebelumnya.
6. Proses akan selesai jika diperoleh satu solusi untuk memenuhi papan dan semua blok sudah digunakan, atau jika semua percobaan sudah dilakukan namun masih belum memenuhi papan dengan menggunakan keseluruhan blok.

Berikut adalah pseudocode dari algoritma di atas :

```
FUNCTION SolvePuzzle(board, blocks):
    VALIDATE input:
        - Ukuran papan harus valid (N > 0, M > 0)
        - Jumlah blok harus sesuai dengan jumlah yang disebutkan
        - Total luas blok harus sama dengan luas papan
        - Tidak boleh ada blok duplikat
        - Setiap blok harus memiliki bentuk yang valid

    CALL RecursiveSolve(board, blocks, 0)

    IF solusi ditemukan:
        PRINT "Solusi ditemukan!"
    ELSE:
```

```
        PRINT "Tidak ada solusi."

FUNCTION RecursiveSolve(board, blocks, index):
    IF index == jumlah blok:
          RETURN isValidBoard(board)   // Jika semua blok sudah diletakkan, cek
apakah papan penuh

    block ← blocks[index]
      orientations ← GenerateAllOrientations(block)   // Buat semua rotasi dan
cermin blok

    FOR each orientation in orientations:
        FOR x from 0 to N:
            FOR y from 0 to M:
                IF canPlace(board, orientation, x, y):
                    placeBlock(board, orientation, x, y)

                    IF RecursiveSolve(board, blocks, index + 1):
                        RETURN true  // Jika solusi ditemukan, hentikan rekursi

                    removeBlock(board, orientation, x, y)  // Backtracking

    RETURN false  // Tidak ada solusi untuk konfigurasi saat ini

FUNCTION GenerateAllOrientations(block):
    orientations ← []

    FOR i from 0 to 3:  // Rotasi 0°, 90°, 180°, 270°
        block ← RotateClockwise(block, 90 * i)
        orientations.append(block)
        orientations.append(FlipHorizontally(block))  // Tambahkan versi cermin

     RETURN unique orientations   // Pastikan tidak ada duplikasi konfigurasi
blok

FUNCTION canPlace(board, block, x, y):
    FOR each cell in block.coordinates:
        newX ← x + cell[0]
        newY ← y + cell[1]
          IF newX out of bounds OR newY out of bounds OR board[newX][newY] !=
'.':
              RETURN false   // Tidak bisa ditempatkan karena bertabrakan atau
keluar dari batas
    RETURN true
```

```
FUNCTION placeBlock(board, block, x, y):
    FOR each cell in block.coordinates:
        newX ← x + cell[0]
        newY ← y + cell[1]
        board[newX][newY] ← block.symbol

FUNCTION removeBlock(board, block, x, y):
    FOR each cell in block.coordinates:
        newX ← x + cell[0]
        newY ← y + cell[1]
        board[newX][newY] ← '.'  // Kembalikan ke kosong

FUNCTION isValidBoard(board):
    FOR each cell in board:
        IF cell == '.':
            RETURN false  // Jika masih ada sel kosong, solusi tidak valid
    RETURN true
```

B. Source Code

Program ini menggunakan bahasa Java dan struktur dari kode program ini terdiri dari beberapa file utama:

1. **Main.java**: Program utama

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

public class Main {
    public static void main(String[] args) {
                System.out.println("---------[  Bruteforce  IQ  Puzzler
]---------");
        Scanner scanner = new Scanner(System.in);
          System.out.print("Masukkan nama file test case tanpa .txt (ex :
tc1): ");
        String fileName1 = scanner.nextLine();
        String fileName = ("../test/" + fileName1 +".txt");

        try {
            Solver solver = new Solver(fileName);
            long start = System.currentTimeMillis();
            boolean solved = solver.solve();
            long finish = System.currentTimeMillis();

            if (solved) {
                 System.out.println("Waktu pencarian: " + (finish - start)
+ " ms");
                                System.out.println("Jumlah  iterasi:  "  +
solver.getIterationCount() + " kali");

                System.out.print("Simpan dalam bentuk gambar? (Y/N): ");
                String saveImage = scanner.nextLine();
                if (saveImage.equalsIgnoreCase("Y")) {
                            SaveImage.savePuzzleImage(solver.getBoard(),
"../test/" + fileName1 +".png");
                }
            }
            else {
```

```
                System.out.println("Tidak ada solusi ditemukan.");
            }
        } catch (IOException e) {
            System.err.println("Gagal membaca file: " + e.getMessage());
        }
        scanner.close();
    }
}
```

2. **Solver.java**: Memproses input dan penyelesaian puzzle

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

public class Solver {
    private char[][] board; // Solve puzzle pake matriks
    private List<Block> blocks;
    private int N, M;
    private int iterationCount;

    // Baca input dari file
    private void readInput(String fileName) throws IOException {
        BufferedReader br = new BufferedReader(new FileReader(fileName));

        // Baca N dan M untuk ukuran papan, dan P untuk jumlah blok
        String[] dims = br.readLine().split(" ");
        N = Integer.parseInt(dims[0]);
        M = Integer.parseInt(dims[1]);
        int P = Integer.parseInt(dims[2]);

        // Membaca mode Puzzle: DEFAULT/CUSTOM/PYRAMID
        String type = br.readLine();

        // Inisialisasi papan
        if (N <= 0 || M <= 0){
            System.out.println("Error: Ukuran papan tidak valid");
            System.exit(0);
```

```
        } else {
            board = new char[N][M];
            for (char [] row : board) {
                Arrays.fill(row, '.');
            }
        }


        // Menyimpan blok yang ada
        blocks = new ArrayList<>();
          Set<Character> usedSymbols = new HashSet<>(); // Cek duplikasi
huruf

        // Memulai untuk membaca blok puzzle
        String currentSymbol = null;
        List<String> currentShape = new ArrayList<>();

        String line;
        while ((line = br.readLine()) != null) {
            if (line.isEmpty()) continue; // Lewati baris kosong

            char firstChar = line.charAt(0);

            // Validasi: Pastikan hanya huruf kapital A - Z
              if (!Character.isUpperCase(firstChar) || firstChar < 'A' ||
firstChar > 'Z') {
                    System.out.println("Error: Karakter blok harus berupa
huruf kapital A-Z! Ditemukan: " + firstChar);
                System.exit(0);
            }

            // Jika menemukan blok baru (karakter berubah)
                        if (currentSymbol == null || firstChar !=
currentSymbol.charAt(0)) {
                if (!currentShape.isEmpty()) {
                    // Validasi: Pastikan blok tidak kosong
                    if (!hasValidCharacter(currentShape)) {
                         System.out.println("Error: Blok " + currentSymbol
+ " tidak memiliki bentuk valid!");
                        System.exit(0);
                    }

                         blocks.add(new Block(currentSymbol.charAt(0), new
ArrayList<>(currentShape)));
```

```
                    currentShape.clear();
                }
                currentSymbol = String.valueOf(firstChar);

                // Validasi: Pastikan huruf tidak duplikat
                if (usedSymbols.contains(currentSymbol.charAt(0))) {
                     System.out.println("Error: Blok " + currentSymbol + "
sudah ada! Tidak boleh ada duplikasi.");
                    System.exit(0);
                }
                usedSymbols.add(currentSymbol.charAt(0));
            }
            currentShape.add(line); // Tambahkan baris ke bentuk blok
        }

        // Tambahkan blok terakhir yang belum tersimpan
        if (!currentShape.isEmpty()) {
            if (!hasValidCharacter(currentShape)) {
                    System.out.println("Error: Blok " + currentSymbol + "
tidak memiliki bentuk valid!");
                System.exit(0);
            }
                      blocks.add(new  Block(currentSymbol.charAt(0),  new
ArrayList<>(currentShape)));
        }

        br.close();

        // Validasi: Pastikan jumlah blok sesuai dengan P
        if (blocks.size() != P) {
                    System.out.println("Error:  Jumlah  blok  tidak  sesuai!
Diharapkan: " + P + ", tetapi ditemukan: " + blocks.size());
            System.exit(0);
        }
           // Setelah membaca semua blok, hitung jumlah total sel yang
ditempati oleh blok
        int totalBlockCells = 0;
        for (Block block : blocks) {
            totalBlockCells += block.coordinates.size();
        }

        // Validasi: Pastikan total sel blok sama dengan ukuran papan
        int totalBoardCells = N * M;
        if (totalBlockCells < totalBoardCells) {
```

```
            System.out.println("Error: Blok kurang! Total sel blok = " +
totalBlockCells + ", tetapi papan memerlukan " + totalBoardCells);
            System.exit(0);
        } else if (totalBlockCells > totalBoardCells) {
             System.out.println("Error: Blok lebih! Total sel blok = " +
totalBlockCells + ", tetapi papan hanya memiliki " + totalBoardCells);
            System.exit(0);
        }
    }
      // Fungsi untuk memastikan blok memiliki minimal satu huruf yang
valid
    private boolean hasValidCharacter(List<String> shape) {
        for (String row : shape) {
            for (char c : row.toCharArray()) {
                if (Character.isUpperCase(c) && c >= 'A' && c <= 'Z') {
                     return true; // Ditemukan karakter huruf kapital yang
valid
                }
            }
        }
        return false; // Tidak ada huruf kapital dalam blok ini
    }

    // Konstruktor
    public Solver(String filename) throws IOException {
        readInput(filename);
    }

    // Memulai pencarian solusi
    public boolean solve() {
        System.out.println("Memulai pencarian solusi...");
        return recursiveSolve(0);
    }

    // Algoritma Bruteforce
    private boolean recursiveSolve(int index) {
        if (index == blocks.size()) return isValidBoard();
        Block block = blocks.get(index);
        List<Block> orientations = block.generateOrientations();
        for (Block orient : orientations) {
            for (int a = 0; a < N; a++) {
                for (int c = 0; c < M; c++) {
                    if (canPlace(orient, a, c)) {
                        placeBlock(orient, a, c, block.symbol);
```

```
                        clearScreen();
                        printBoard();
                                        try { Thread.sleep(0); } catch
(InterruptedException e) { Thread.currentThread().interrupt(); } //ubah
Thread.sleep untuk perubahan kecepatan animasi

                        if (recursiveSolve(index + 1)){
                            return true;
                        }
                        removeBlock(orient, a, c);
                        clearScreen();
                        printBoard();
                                        try { Thread.sleep(0); } catch
(InterruptedException e) { Thread.currentThread().interrupt(); } //ubah
Thread.sleep untuk perubahan kecepatan animasi
                        iterationCount++;
                    }
                }
            }
        }
        return false; // backtrack
    }

    // Fungsi-fungsi pendukung
     // Mengecek apakah blok bisa ditempatkan dengan mengecek batas papan
dan memastikan tidak menimpa blok lain.
    private boolean canPlace(Block block, int startX, int startY) {
        for (int[] cell : block.coordinates) {
            int x = startX + cell[0];
            int y = startY + cell[1];
             if (x < 0 || x >= N || y < 0 || y >= M || board[x][y] != '.')
{
                return false;
            }
        }
        return true;
    }

    // Menempatkan blok
      private void placeBlock(Block block, int startX, int startY, char
symbol) {
        for (int[] cell : block.coordinates) {
            int x = startX + cell[0];
            int y = startY + cell[1];
```

```
            board[x][y] = symbol;
        }
    }

    // Menghapus blok dari papan untuk backtracking
    private void removeBlock(Block block, int startX, int startY) {
        for (int[] cell : block.coordinates) {
            int x = startX + cell[0];
            int y = startY + cell[1];
            board[x][y] = '.';
        }
    }

       // Validasi akhir papan: memastikan semua sel terisi sebelum
dijadikan solusi
    private boolean isValidBoard() {
        int filledCells = 0;
        for (char[] row : board) {
            for (char cell : row) {
                if (cell != '.') filledCells++;
            }
        }
        return filledCells == (N * M);
    }
        public void printBoard() {
        for (char[] row : board) {
            for (char cell : row) {
                System.out.print(Utility.getColoredChar(cell) + " ");
            }
            System.out.println();
        }
    }

    public void saveSolution(String fileName) throws IOException {
        BufferedWriter bw = new BufferedWriter(new FileWriter(fileName));
        for (char[] row : board) {
            for (char cell : row) {
                bw.write(cell + " ");
            }
            bw.newLine();
        }
        bw.close();
    }
```

```
    public int getIterationCount() {
        return iterationCount;
    }

    public char[][] getBoard() {
        return board;
    }

    private void clearScreen() {
        System.out.print("\033[H\033[2J");
        System.out.flush();
    }
}
```

3. **Block.java**: Mengelola seluruh kemungkinan konfigurasi blok inputan.

```
import java.util.*;

public class Block {
    public char symbol;
    public List<int[]> coordinates;

    public Block(char symbol, List<String> shape) {
        this.symbol = symbol;
        this.coordinates = parseShape(shape);
    }

    private List<int[]> parseShape(List<String> shape) {
        List<int[]> coords = new ArrayList<>();
        for (int i = 0; i < shape.size(); i++) {
            for (int j = 0; j < shape.get(i).length(); j++) {
                if (shape.get(i).charAt(j) != '.') {
                    coords.add(new int[]{i, j});
                }
            }
        }
        return coords;
    }
    // Generate semua kemungkinan orientasi blok
    public List<Block> generateOrientations() {
        Set<String> seen = new HashSet<>();
        List<Block> orientations = new ArrayList<>();
        char[][] currentShape = toMatrix(this.coordinates);

        for (int i = 0; i < 4; i++) { // 4 kali rotasi
```

```
            String shapeStr = matrixToString(currentShape);

            if (seen.add(shapeStr)) {
                                orientations.add(new  Block(this.symbol,
matrixToShape(currentShape)));

                // Cerminkan dan cek bentuk uniknya
                char[][] reflected = reflect(currentShape);
                shapeStr = matrixToString(reflected);
                if (seen.add(shapeStr)) {
                                orientations.add(new  Block(this.symbol,
matrixToShape(reflected)));
                }
            }

            currentShape = rotate(currentShape);
        }

        return orientations;
    }
    // Konversi koordinat ke matriks
    private char[][] toMatrix(List<int[]> coords) {
        int maxX = 0, maxY = 0;
        for (int[] c : coords) {
            maxX = Math.max(maxX, c[0]);
            maxY = Math.max(maxY, c[1]);
        }

        char[][] shape = new char[maxX + 1][maxY + 1];
        for (char[] row : shape) Arrays.fill(row, '.');

        for (int[] c : coords) {
            shape[c[0]][c[1]] = this.symbol;
        }

        return shape;
    }
    // Konversi matriks ke string
    private String matrixToString(char[][] matrix) {
        StringBuilder sb = new StringBuilder();
        for (char[] row : matrix) {
            sb.append(new String(row)).append("\n");
        }
        return sb.toString();
```

```
    }
    // Konversi matriks ke list bentuk blok
    private List<String> matrixToShape(char[][] matrix) {
        List<String> result = new ArrayList<>();
        for (char[] row : matrix) {
            result.add(new String(row));
        }
        return result;
    }
    // Rotasi 90 derajat searah jarum jam
    private char[][] rotate(char[][] matrix) {
        int rows = matrix.length, cols = matrix[0].length;
        char[][] rotated = new char[cols][rows];

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                rotated[j][rows - i - 1] = matrix[i][j];
            }
        }
        return rotated;
    }
    // Cerminkan blok
    private char[][] reflect(char[][] matrix) {
        int rows = matrix.length, cols = matrix[0].length;
        char[][] reflected = new char[rows][cols];

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

4. **Utility.java**: Utilitas, yakni untuk pewarnaan output pada penyelesaian.

```
public class Utility {
    private static final String[] COL = {
        "\u001B[38;2;255;0;0m",     // A - Red
        "\u001B[38;2;0;0;255m",     // B - Blue
        "\u001B[38;2;0;205;102m",   // C - Green
        "\u001B[38;2;255;165;0m",   // D - Orange
        "\u001B[38;2;128;0;128m",   // E - Purple
        "\u001B[38;2;64;224;208m", // F - Turquoise
```

```
        "\u001B[38;2;220;20;60m",  // G - Crimson
        "\u001B[38;2;34;139;34m",  // H - Forest Green
        "\u001B[38;2;255;20;147m", // I - Pink
        "\u001B[38;2;0;0;128m",    // J - Navy
        "\u001B[38;2;139;69;19m",  // K - Brown
        "\u001B[38;2;50;205;50m",  // L - Lime
        "\u001B[38;2;255;0;255m",  // M - Magenta
        "\u001B[38;2;0;128;128m",  // N - Teal
        "\u001B[38;2;255;215;0m",  // O - Gold
        "\u001B[38;2;75;0;130m",   // P - Indigo
        "\u001B[38;2;128;0;0m",    // Q - Maroon
        "\u001B[38;2;135;206;235m",// R - Sky Blue
        "\u001B[38;2;255;140;0m",  // S - Dark Orange
        "\u001B[38;2;138;43;226m", // T - Violet
        "\u001B[38;2;128;128;0m",  // U - Olive
        "\u001B[38;2;255;127;80m", // V - Coral
        "\u001B[38;2;0;191;255m",  // W - Deep Sky Blue
        "\u001B[38;2;139;0;0m",    // X - Dark Red
        "\u001B[38;2;0;255;127m",  // Y - Spring Green
        "\u001B[38;2;148;0;211m"   // Z - Dark Violet
    };

    public static String getColoredChar(char c) {
        if (c == '.') return "\u001B[37m.\u001B[0m";
        return COL[c - 'A'] + c + "\u001B[0m";
    }
}
```

5. **SaveImage.java**: Menyimpan solusi dalam bentuk gambar.

```
import java.awt.*;
import java.awt.image.BufferedImage;
import javax.imageio.ImageIO;
import java.io.File;
import java.io.IOException;

public class SaveImage {
    private static final int CELL_SIZE = 50; // Ukuran tiap sel
    private static final int PADDING = 10;   // Padding sekitar board

    public static void savePuzzleImage(char[][] board, String filename) {
        int rows = board.length;
        int cols = board[0].length;
        int width = cols * CELL_SIZE + 2 * PADDING;
        int height = rows * CELL_SIZE + 2 * PADDING;
```

```
                BufferedImage image = new BufferedImage(width, height,
BufferedImage.TYPE_INT_RGB);
        Graphics2D g2d = image.createGraphics();

        // Set warna background putih
        g2d.setColor(Color.WHITE);
        g2d.fillRect(0, 0, width, height);

        // Gambar grid dan blok
        for (int r = 0; r < rows; r++) {
            for (int c = 0; c < cols; c++) {
                char symbol = board[r][c];
                g2d.setColor(getColorForSymbol(symbol));
                     g2d.fillRect(PADDING + c * CELL_SIZE, PADDING + r *
CELL_SIZE, CELL_SIZE, CELL_SIZE);

                // Gambar border
                g2d.setColor(Color.BLACK);
                     g2d.drawRect(PADDING + c * CELL_SIZE, PADDING + r *
CELL_SIZE, CELL_SIZE, CELL_SIZE);

                // Tulis huruf blok
                g2d.setColor(Color.BLACK);
                g2d.setFont(new Font("Arial", Font.BOLD, 20));
                     g2d.drawString(String.valueOf(symbol), PADDING + c *
CELL_SIZE + 20, PADDING + r * CELL_SIZE + 30);
            }
        }

        g2d.dispose();

        try {
            ImageIO.write(image, "png", new File(filename));
                  System.out.println("Gambar solusi disimpan sebagai " +
filename);
        } catch (IOException e) {
                       System.err.println("Gagal menyimpan gambar: " +
e.getMessage());
        }
    }

    // Fungsi untuk memilih warna blok berdasarkan huruf
    private static Color getColorForSymbol(char symbol) {
```

```
        switch (symbol) {
            case 'A': return new Color(255, 0, 0);
            case 'B': return new Color(0, 0, 255);
            case 'C': return new Color(0, 205, 102);
            case 'D': return new Color(255, 165, 0);
            case 'E': return new Color(128, 0, 128);
            case 'F': return new Color(64, 224, 208);
            case 'G': return new Color(220, 20, 60);
            case 'H': return new Color(34, 139, 34);
            case 'I': return new Color(255, 20, 147);
            case 'J': return new Color(0, 0, 128);
            case 'K': return new Color(139, 69, 19);
            case 'L': return new Color(50, 205, 50);
            case 'M': return new Color(255, 0, 255);
            case 'N': return new Color(0, 128, 128);
            case 'O': return new Color(255, 215, 0);
            case 'P': return new Color(75, 0, 130);
            case 'Q': return new Color(128, 0, 0);
            case 'R': return new Color(135, 206, 235);
            case 'S': return new Color(255, 140, 0);
            case 'T': return new Color(138, 43, 226);
            case 'U': return new Color(128, 128, 0);
            case 'V': return new Color(255, 127, 80);
            case 'W': return new Color(0, 191, 255);
            case 'X': return new Color(139, 0, 0);
            case 'Y': return new Color(0, 255, 127);
            case 'Z': return new Color(148, 0, 211);
            default: return Color.GRAY;
        }
    }
}
```

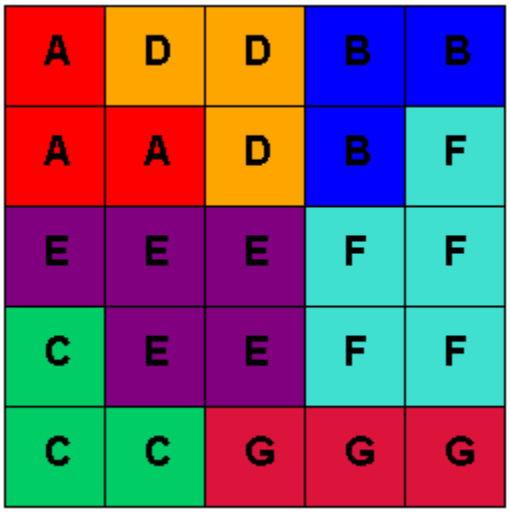
C. Uji Coba Program

1. **Test Case 1 (valid input, terdapat penyelesaian)**
   5 5 7
   DEFAULT
   A
   AA
   BB
   B
   C

CC
D
DD
EE
EE
E
F
FF
FF
GGG

```
---------[ Bruteforce IQ Puzzler ]---------
Masukkan nama file test case tanpa .txt (ex : tc1): tc1
```

```
A D D B B
A A D B F
E E E F F
C E E F F
C C G G G
Waktu pencarian: 14047 ms
Jumlah iterasi: 6848 kali
Simpan dalam bentuk gambar? (Y/N): Y
Gambar solusi disimpan sebagai ../test/tc1.png
```



2. **Test Case 2 (papan tidak valid)**
   -5 -5 7
   DEFAULT
   A
   AA

BB
B
C
CC
D
DD
EE
EE
E
F
FF
FF
GGG

```
---------[ Bruteforce IQ Puzzler ]---------
Masukkan nama file test case tanpa .txt (ex : tc1): tc2
Error: Ukuran papan tidak valid
```

3. **Test Case 3 (jumlah blok tidak sesuai)**

5 5 6
DEFAULT
A
AA
B
BB
C
CC
D
DD
EE
EE
E
FF
FF
F
GGG

```
---------[ Bruteforce IQ Puzzler ]---------
Masukkan nama file test case tanpa .txt (ex : tc1): tc3
Error: Jumlah blok tidak sesuai. Masukan P=6, namun terdapat 7 blok.
```

4. **Test Case 4 (total luas blok berlebih/kurang)**

4 4 7

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

```
---------[ Bruteforce IQ Puzzler ]---------
Masukkan nama file test case tanpa .txt (ex : tc1): tc4
Error: Blok lebih! Total sel blok = 25, tetapi papan hanya memiliki 16
```

5. **Test Case 5 (terdapat karakter tidak valid)**

3 3 3
DEFAULT
AAA
BBB
;;;

```
---------[ Bruteforce IQ Puzzler ]---------
Masukkan nama file test case tanpa .txt (ex : tc1): tc5
Error: Karakter blok ;bukan A-Z.
```

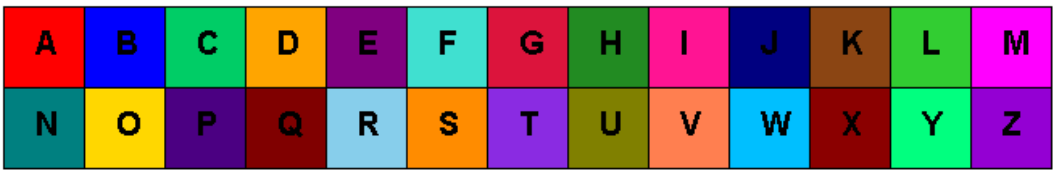
6. **Test Case 6 (valid input, namun tidak ada solusi)**

3 3 3
DEFAULT
A
AA
BBB
CCC

```
. . .
. . .
. . .
Tidak ada solusi ditemukan.
```

7. **Test Case 7 (input seluruh huruf)**
   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

```
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: 36 ms
Jumlah iterasi: 0 kali
Simpan dalam bentuk gambar? (Y/N): Y
Gambar berhasil disimpan pada ../test/tc7.png
```


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

D. 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	✔	

Link Repository Github : https://github.com/jandhiesto/TucilStima1_13523127