LAPORAN TUGAS KECIL I
IF2211 STRATEGI ALGORITMA
Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force

Disusun oleh:
Ivant Samuel Silaban
13523129

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

Daftar Isi

Algoritma Brute Force

Algoritma yang digunakan dalam program ini adalah algoritma **Brute Force** dengan bantuan **Backtracking** untuk menyusun blok-blok dalam papan permainan **IQ Puzzle Pro**. Berikut adalah tahapan-tahapannya:

1. Inisialisasi Papan dan Blok
    - Program akan membaca ukuran papan N x M dan daftar blok dari file input
    - Setiap blok direpresentasikan sebagai kumpulan koordinat relatif terhadap titik awalnya.
    - Program juga akan membaca mode permainan DEFAULT, CUSTOM, atau PYRAMID (pada program ini, hanya menyediakan mode DEFAULT) untuk menentukan konfigurasi papan permainan.
    - Dilakukan validasi terhadap input seperti memastikan jumlah total sel blok sesuai dengan ukuran papan, dan validasi-validasi lainnya.
2. Algoritma penyelesaian dengan Brute Force:
    - Ambil blok secara berurutan dalam daftar
    - Membuat semua orientasi pada blok, dimulai dari 4 rotasi dan pencerminan
    - Mencari posisi kosong di papan untuk menempatkan blok dengan mencoba menempatkan blok pada setiap posisi kosong di papan dan memerika apakah blok dapat ditempatkan tanpa keluar batas dan tanpa menimpa blok lain.
    - Rekursi untuk blok selanjutnya
    - Backtracking jika tidak ada posisi yang cocok.

Berikut adalah **pseudocode** untuk proses penyelesaiannya:

```
Function solveRecursively(index):
    If index == jumlah blok:
        Return isValidBoard()  // Jika semua blok sudah ditempatkan, cek
apakah papan penuh

    block ← blok pada index
    orientations ← semua orientasi unik dari block

    For each orient in orientations:
        For each posisi (r, c) di papan:
            If canPlace(orient, r, c):
                placeBlock(orient, r, c, block.symbol)  // Tempatkan blok
pada papan
```

```
                If solveRecursively(index + 1):  // Rekursi untuk blok
selanjutnya
                    Return True  // Jika solusi ditemukan, langsung return

                removeBlock(orient, r, c)  // Backtracking: hapus blok
jika gagal

    Return False  // Jika tidak ada solusi, kembali ke langkah sebelumnya
```

Source Program

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

1. **Main.java**: Program utama untuk menangani input pengguna
2. **PuzzleSolver.java**: Memproses dan menyelesaikan puzzle
3. **Block.java**: Mengelola bentuk dan blok dan menghasilkan orientasi uniknya.
4. **Utils.java**: Utilitas program, yaitu pewarnaan teks di terminal.
5. **PuzzleImageSaver.java** (bonus 1) -> Menyimpan solusi dalam bentuk gambar.

Main.java:

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

public class Main {
    public static void main(String[] args) {
        System.out.println("=== IQ Puzzler Pro Solver ===");
        Scanner scanner = new Scanner(System.in);

        System.out.print("Masukkan nama file test case (.txt): ");
        String fileName = scanner.nextLine();

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

            if (solved) {
                solver.printBoard();
                System.out.println("Waktu pencarian: " + (end - start) + "
ms");
                System.out.println("Banyak iterasi: " +
solver.getIterationCount());

                System.out.print("Ingin menyimpan solusi sebagai gambar?
(ya/tidak): ");
                String saveImage = scanner.nextLine();
                if (saveImage.equalsIgnoreCase("ya")) {
```

```
                    PuzzleImageSaver.savePuzzleImage(solver.getBoard(),
"test/solution.png");
                    System.out.println("Solusi disimpan sebagai gambar di
'test/solution.png'");
                }

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

PuzzleSolver.java

```
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 PuzzleSolver {
    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 ukuran papan N x M dan jumlah blok P
        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 tyoe = br.readLine();

        // Inisialisasi papan
        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') {
                throw new IllegalArgumentException("Error: Karakter blok
harus berupa huruf kapital A-Z! Ditemukan: " + firstChar);
            }

            // Jika menemukan blok baru (karakter berubah)
            if (currentSymbol == null || firstChar !=
currentSymbol.charAt(0)) {
                if (!currentShape.isEmpty()) {
```

```
                    // Validasi: Pastikan blok tidak kosong
                    if (!hasValidCharacter(currentShape)) {
                        throw new IllegalArgumentException("Error: Blok "
+ currentSymbol + " tidak memiliki bentuk valid!");
                    }

                    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))) {
                    throw new IllegalArgumentException("Error: Blok " +
currentSymbol + " sudah ada! Tidak boleh ada duplikasi.");
                }
                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)) {
                throw new IllegalArgumentException("Error: Blok " +
currentSymbol + " tidak memiliki bentuk valid!");
            }
            blocks.add(new Block(currentSymbol.charAt(0), new
ArrayList<>(currentShape)));
        }

        br.close();
            // Validasi: Pastikan jumlah total sel blok sama dengan ukuran
papan
        int totalBlockCells = 0;
        for (Block b : blocks) {
            totalBlockCells += b.coordinates.size();
        }
```

```
        if (totalBlockCells != (N * M)) {
            throw new IllegalArgumentException("Error: Total sel blok (" +
totalBlockCells + ") tidak sesuai dengan ukuran papan (" + (N * M) + ")");
        }
        // **Validasi: Pastikan jumlah blok sesuai dengan P**
        if (blocks.size() != P) {
            throw new IllegalArgumentException("Error: Jumlah blok tidak
sesuai! Diharapkan: " + P + ", tetapi ditemukan: " + blocks.size());
        }
    }
    // 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 PuzzleSolver(String filename) throws IOException {
        readInput(filename);
    }

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

    // Algoritma Bruteforce/Backtracking
    private boolean solveRecursively(int index) {
        if (index == blocks.size()) return isValidBoard(); // cek valid

        Block block = blocks.get(index);
        List<Block> orientations = block.generateOrientations();
```

```
        for (Block orient : orientations) {
            for (int r = 0; r < N; r++) {
                for (int c = 0; c < M; c++) {
                    if (canPlace(orient, r, c)) {
                        placeBlock(orient, r, c, block.symbol);
                        clearScreen();
                        printBoard();

                        try { Thread.sleep(200); } catch
(InterruptedException e) { Thread.currentThread().interrupt(); }

                        if (solveRecursively(index + 1)) return true; //
Jika berhasil, langsung return
                        removeBlock(orient, r, c);
                        clearScreen();
                        printBoard();
                        try { Thread.sleep(200); } catch
(InterruptedException e) { Thread.currentThread().interrupt(); }
                        iterationCount++;
                    }
                }
            }
        }
        return false; // backtrack
    }
    // Fungsi-fungsi pendukung untuk algoritma
    // 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 di simbol titik di papan
    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(Utils.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();
    }
}
```

Block.java:

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

public class PuzzleImageSaver {
    private static final int CELL_SIZE = 50;
    private static final int PADDING = 10;

    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();

        // Simpan sebagai file PNG
        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 Color.RED;
            case 'B': return Color.BLUE;
            case 'C': return Color.GREEN;
            case 'D': return Color.ORANGE;
            case 'E': return Color.MAGENTA;
            case 'F': return Color.CYAN;
            case 'G': return Color.PINK;
            case 'H': return new Color(255, 165, 0); // Oranye terang
            case 'I': return new Color(128, 0, 128); // Ungu
            case 'J': return new Color(0, 255, 255); // Aqua
            case 'K': return new Color(255, 215, 0); // Emas
            case 'L': return new Color(139, 69, 19); // Coklat
            case 'M': return new Color(255, 0, 255); // Fuchsia
            case 'N': return new Color(0, 128, 128); // Teal
            case 'O': return new Color(128, 128, 0); // Olive
            case 'P': return new Color(0, 0, 128);   // Navy
            case 'Q': return new Color(178, 34, 34); // Merah gelap
            case 'R': return new Color(70, 130, 180); // Baja Biru
            case 'S': return new Color(218, 112, 214); // Orchid
            case 'T': return new Color(240, 230, 140); // Khaki
            case 'U': return new Color(154, 205, 50);  // Hijau Kuning
            case 'V': return new Color(255, 99, 71);   // Tomato
            case 'W': return new Color(173, 216, 230); // Biru Muda
            case 'X': return new Color(199, 21, 133);  // Deep Pink
            case 'Y': return new Color(0, 191, 255);   // Deep Sky Blue
            case 'Z': return new Color(144, 238, 144); // Hijau Muda
            default: return Color.LIGHT_GRAY; // Untuk simbol yang tidak
dikenali
        }
    }
}
```

Utils.java

```
public class Utils {
    public static String getColoredChar(char c) {
        if (c == '.') return "\u001B[37m.\u001B[0m";
        int color = 31 + (c - 'A') % 6; // 6 warna berbeda
        return "\u001B[" + color + "m" + c + "\u001B[0m";
    }
}
```

PuzzleImageSaver.java (bonus):

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

public class PuzzleImageSaver {
    private static final int CELL_SIZE = 50;
    private static final int PADDING = 10;

    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();

        // Simpan sebagai file PNG
        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 Color.RED;
            case 'B': return Color.BLUE;
            case 'C': return Color.GREEN;
            case 'D': return Color.ORANGE;
            case 'E': return Color.MAGENTA;
            case 'F': return Color.CYAN;
            case 'G': return Color.PINK;
```

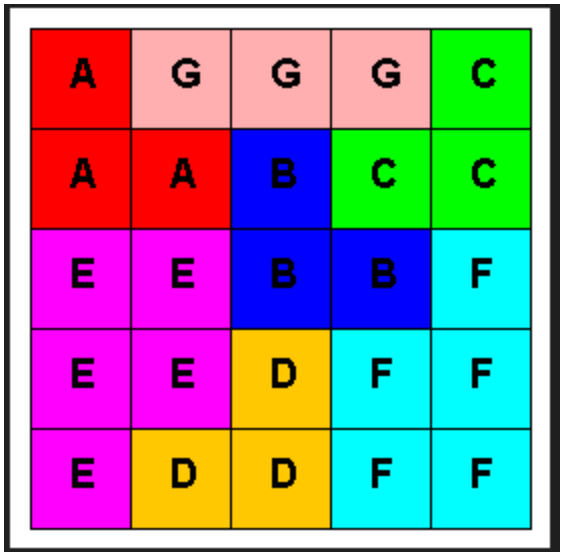
```
            case 'H': return new Color(255, 165, 0);
            case 'I': return new Color(128, 0, 128);
            case 'J': return new Color(0, 255, 255);
            case 'K': return new Color(255, 215, 0);
            case 'L': return new Color(139, 69, 19);
            case 'M': return new Color(255, 0, 255);
            case 'N': return new Color(0, 128, 128);
            case 'O': return new Color(128, 128, 0);
            case 'P': return new Color(0, 0, 128);
            case 'Q': return new Color(178, 34, 34);
            case 'R': return new Color(70, 130, 180);
            case 'S': return new Color(218, 112, 214);
            case 'T': return new Color(240, 230, 140);
            case 'U': return new Color(154, 205, 50);
            case 'V': return new Color(255, 99, 71);
            case 'W': return new Color(173, 216, 230);
            case 'X': return new Color(199, 21, 133);
            case 'Y': return new Color(0, 191, 255);
            case 'Z': return new Color(144, 238, 144);
            default: return Color.LIGHT_GRAY;
        }
    }
}
```

Hasil Tangkapan Layar Input dan Output

1. **Test Case 1 (test case pada spesifikasi)**:
   5 5 7
   DEFAULT
   A
   AA
   B
   BB
   C
   CC
   D
   DD
   EE
   EE
   E
   FF
   FF
   F
   GGG
   Output (jika tidak menggunakan printboard() pada perulangan untuk menampilkan proses brute force):

```
=== IQ Puzzler Pro Solver ===
Masukkan nama file test case (.txt): test/test1.txt
Memulai pencarian solusi...
A G G G C
A A B C C
E E B B F
E E D F F
E D D F F
Waktu pencarian: 59 ms
Banyak iterasi: 7159
Ingin menyimpan solusi sebagai gambar? (ya/tidak): ya
Gambar solusi disimpan sebagai test/solution.png
Solusi disimpan sebagai gambar di 'test/solution.png'
```

   Gambar:



2. **Test Case 2 (karakter tidak valid):**
   5 5 3
   DEFAULT
   a
   aa
   B
   BB
   C
   CC
   Output:

```
=== IQ Puzzler Pro Solver ===
Masukkan nama file test case (.txt): test/test2.txt
Exception in thread "main" java.lang.IllegalArgumentExce
ption: Error: Karakter blok harus berupa huruf kapital A
-Z! Ditemukan: a
        at PuzzleSolver.readInput(PuzzleSolver.java:52)
        at PuzzleSolver.<init>(PuzzleSolver.java:114)
        at Main.main(Main.java:13)
```

3. **Test Case 3 (karakter duplikat):**
   5 5 3
   DEFAULT
   A
   AA
   B
   BB
   C

CC
A
AA
Output:

```
=== IQ Puzzler Pro Solver ===
Masukkan nama file test case (.txt): test/test3.txt
Exception in thread "main" java.lang.IllegalArgumentExce
ption: Error: Blok A sudah ada! Tidak boleh ada duplikas
i.
        at PuzzleSolver.readInput(PuzzleSolver.java:70)
        at PuzzleSolver.<init>(PuzzleSolver.java:114)
        at Main.main(Main.java:13)
```

4. **Test Case 4 (Jumlah blok tidak sesuai dengan inputan jumlah blok di awal):**
   2 3 3
   DEFAULT
   A
   AA
   B
   BB
   Output:

```
=== IQ Puzzler Pro Solver ===
Masukkan nama file test case (.txt): test/test4.txt
Exception in thread "main" java.lang.IllegalArgumentExce
ption: Error: Jumlah blok tidak sesuai! Diharapkan: 3, t
etapi ditemukan: 2
        at PuzzleSolver.readInput(PuzzleSolver.java:97)
        at PuzzleSolver.<init>(PuzzleSolver.java:114)
        at Main.main(Main.java:13)
```

   Note: output sama untuk P <= 0

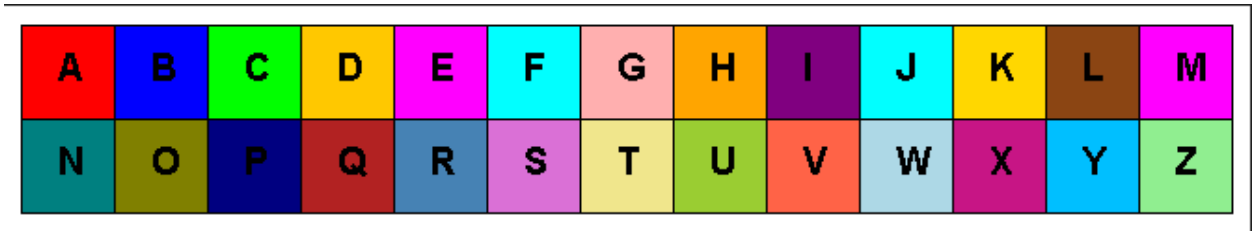
5. **Test Case 5 (Memasukkan semua karakter valid):**
   2 13 26
   DEFAULT
   A
   B
   C
   D
   E
   F
   G

H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

Output:

```
=== IQ Puzzler Pro Solver ===
Masukkan nama file test case (.txt): test/test5.txt
Memulai pencarian solusi...
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: 2 ms
Banyak iterasi: 0
Ingin menyimpan solusi sebagai gambar? (ya/tidak): ya
Gambar solusi disimpan sebagai test/solution.png
Solusi disimpan sebagai gambar di 'test/solution.png'
```

Gambar:



6. **Test Case 6 (Papan tidak terisi penuh karena ukuran blok tidak sesuai dengan ukuran papan):**

5 5 6

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

Output:

```
=== IQ Puzzler Pro Solver ===
Masukkan nama file test case (.txt): test/test6.txt
Exception in thread "main" java.lang.IllegalArgumentExce
ption: Error: Total sel blok (20) tidak sesuai dengan uk
uran papan (25)
        at PuzzleSolver.readInput(PuzzleSolver.java:93)
        at PuzzleSolver.<init>(PuzzleSolver.java:114)
        at Main.main(Main.java:13)
```

7. **Test Case 7 (Ukuran papan tidak sesuai dengan ukuran blok):**

12 1 3
DEFAULT
A
AA
B
BBBB
C
CC
C

Output:

```
=== IQ Puzzler Pro Solver ===
Masukkan nama file test case (.txt): test/test7.txt
Memulai pencarian solusi...
Tidak ada solusi ditemukan.
```

8. **Test Case 8 (Ukuran bernilai negatif):**

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

Output:

```
=== IQ Puzzler Pro Solver ===
Masukkan nama file test case (.txt): test/test8.txt
Exception in thread "main" java.lang.NegativeArraySizeException: -5
        at PuzzleSolver.readInput(PuzzleSolver.java:32)
        at PuzzleSolver.<init>(PuzzleSolver.java:114)
        at Main.main(Main.java:13)
```

Check List dan Link Repository

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: ivant8k/Tucil1_13523129