

LAPORAN TUGAS KECIL 1
Penyelesaian IQ Puzzle Pro Solver dengan Algoritma Brute Force
Semester II Tahun 2024/2025



Disusun oleh:
Farrel Athalla Putra
13523118

**Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2024**

DAFTAR ISI

DAFTAR ISI.....	1
BAB 1 DESKRIPSI MASALAH.....	2
BAB 2 ALGORITMA PROGRAM.....	3
BAB 3 SOURCE CODE PROGRAM.....	10
3.1 Repository Program.....	10
3.2 Source Code Program.....	10
3.2.1 Block.java.....	10
3.2.2 Solver.java.....	13
3.2.3 Main.java.....	17
3.2.4 IO.java.....	18
BAB 4 MASUKAN DAN LUARAN PROGRAM.....	24
4.1 Test Case 1.....	24
4.2 Test Case 2.....	25
4.3 Test Case 3.....	26
4.4 Test Case 4.....	28
4.5 Test Case 5.....	29
4.6 Test Case 6.....	30
4.7 Test Case 7.....	31
4.8 Test Case 8.....	32
4.9 Test Case 9.....	33
4.10 Test Case 10.....	33
4.11 Test Case 11.....	34
BAB 5 LAMPIRAN.....	36

BAB 1

DESKRIPSI MASALAH

IQ Puzzler Pro adalah permainan papan yang mengharuskan pemain mengisi seluruh area papan menggunakan blok puzzle yang tersedia tanpa menyisakan ruang kosong. Permainan ini melatih keterampilan berpikir logis, pemahaman ruang, dan kemampuan memecahkan masalah dengan menyusun blok yang memiliki bentuk unik agar papan dapat terisi dengan sempurna.

Komponen utama dalam permainan ini terdiri dari **board (papan permainan)** dan **blok/piece**. *Board* merupakan tempat utama pemain harus menyusun blok hingga seluruh area papan terisi tanpa ada bagian yang tersisa. Blok/*piece* adalah komponen yang memiliki bentuk unik dan dapat diputar atau dicerminkan untuk menemukan susunan yang tepat. Setiap blok harus digunakan dalam penyelesaian puzzle, sehingga pemain perlu menemukan cara optimal untuk menempatkan semuanya tanpa bertumpang tindih atau melewati batas papan.

Permainan dimulai dengan papan kosong atau dalam beberapa kasus, dengan beberapa blok yang sudah ditempatkan sebagai petunjuk awal. Pemain harus menyusun sisa blok dengan memperhatikan kemungkinan rotasi dan pencerminan untuk menyesuaikan bentuknya dengan ruang yang tersedia. Tantangan utama dalam permainan ini adalah menemukan kombinasi yang benar sehingga semua blok dapat masuk tanpa menyisakan celah. Jika seluruh papan berhasil terisi penuh dengan aturan yang sesuai, maka puzzle dianggap selesai. Namun, karena jumlah kemungkinan susunan yang sangat banyak, permainan ini dapat menjadi tantangan tersendiri bagi pemain, terutama pada level yang lebih sulit.



Gambar 1. Permainan IQ Puzzler Pro
(Sumber : <https://www.smartgamesusa.com>)

BAB 2

ALGORITMA PROGRAM

Salah satu cara untuk menyelesaikan permainan ini adalah dengan menggunakan algoritma brute force. Berikut adalah langkah-langkah algoritma brute force yang digunakan dalam program penyelesaian IQ Puzzler Pro Solver ini:

1. Membaca dan Memvalidasi Input

Struktur Data Blok

```
import java.util.*;  
  
public class Block {  
    private char label;  
    private char[][] shape;  
    private List<char[][]> variants;  
}
```

Struktur data dari setiap blok terdiri dari label sebagai identitas dari blok, shape yang menyimpan bentuk blok dalam format array 2D, serta variants yang akan menyimpan seluruh variasi dari transformasi pada blok.

```
public static List<Block> readFile(String filename, int[] ukuran,  
List<String> newPapan) throws IOException {  
    List<Block> blocks = new ArrayList<>();  
    Map<Character, Boolean> blockUsed = new HashMap<>();  
  
    ...  
  
    if (width <= 0 || height <= 0 || totalBlocks <= 0) {  
        throw new IllegalArgumentException("Ukuran papan dan jumlah block harus  
        lebih dari 0!");  
    }  
  
    ...  
  
    if (blocks.size() != totalBlocks) {  
        throw new IllegalArgumentException("Total block tidak sesuai!");  
    }  
  
    ...  
  
    if (currentLabel == null || label != currentLabel) {  
        if (currentLabel != null) {  
            blocks.add(new Block(currentLabel, currentBlock));  
            blockUsed.put(currentLabel, true);  
        }  
    }
```

```

...
if (blockUsed.getOrDefault(label, false)) {
    throw new IllegalArgumentException("Terdapat duplikat block " + label +
    ".");
}

```

Program membaca *file input* yang berisi dimensi papan dan daftar blok puzzle. Masukan ini divalidasi untuk memastikan bahwa setiap blok memiliki bentuk yang benar, tidak ada blok yang duplikat, serta papan memiliki ukuran yang sesuai. Jika semua tervalidasi, maka program akan membuat peta dengan ‘X’ sebagai area yang dapat ditempati blok dan ‘.’ sebagai area yang tidak dapat ditempati oleh blok. Blok-blok yang telah dibaca akan dimasukan ke dalam list of Blocks.

2. Membentuk Variasi Blok

```

private void generateVariants() {
    variants = new ArrayList<>();
    char[][] current = shape;

    for (int i = 0; i < 4; i++) {
        current = rotate(current);
        variants.add(current);
    }

    current = flipHorizontal(shape);
    variants.add(current);
    for (int i = 0; i < 3; i++) {
        current = rotate(current);
        variants.add(current);
    }
}

private char[][] rotate(char[][] matrix) {
    int rows = matrix.length;
    int cols = matrix[0].length;
    char[][] rotated = new char[cols][rows];
    for (int r = 0; r < rows; r++) {
        for (int c = 0; c < cols; c++) {
            rotated[c][rows - 1 - r] = matrix[r][c];
        }
    }
    return rotated;
}

private char[][] flipHorizontal(char[][] matrix) {
    int rows = matrix.length;
    int cols = matrix[0].length;
    char[][] flipped = new char[rows][cols];
    for (int r = 0; r < rows; r++) {
        for (int c = 0; c < cols; c++) {

```

```

        flipped[r][cols - 1 - c] = matrix[r][c];
    }
}
return flipped;
}

```

Setiap blok yang disimpan akan diproses untuk menghasilkan semua kemungkinan variasi transformasi melalui fungsi generateVariants(). Pertama, variants diinisialisasi sebagai ArrayList untuk menyimpan hasil transformasi. Shape awal blok disimpan dalam current, yang setelahnya dirotasi 90 derajat sebanyak 4 kali menggunakan fungsi rotate(), serta pencerminan horizontal menggunakan flipHorizontal() yang dilanjutkan dengan rotasi 90 derajat sebanyak 3 kali untuk mendapatkan seluruh kemungkinan variasi blok. Implementasi fungsi rotate() dan flipHorizontal() dilakukan perhitungan rotasi matriks sederhana.

3. Algoritma Brute Force secara Rekursi

```

public class Solver {
    private char[][] board;
    private List<Block> blocks;
    private long jumlahKasus = 0;

    public Solver(char[][] board, List<Block> blocks) {
        this.board = board;
        this.blocks = blocks;
    }

    public boolean solve() {
        long start = System.currentTimeMillis();
        transformBoard();
        if (bruteForce(0) && isBoardFull()) {
            System.out.println("Solusi ditemukan:");
            printBoard();
            long end = System.currentTimeMillis();
            long totalTime = end - start;
            System.out.println("Waktu pencarian: " + totalTime + " ms");
            System.out.println("Banyak kasus yang ditinjau: " + jumlahKasus);
            return true;
        } else {
            System.out.println("Tidak ada solusi.");
            return false;
        }
    }
}

```

Untuk menyelesaikan persoalan, dibuat class Solver dengan atribut board bertipe matriks of character untuk menyimpan papan, list of Block untuk menyimpan seluruh blok yang dapat digunakan, serta jumlahKasus untuk menyimpan jumlah iterasi. Fungsi solve() digunakan untuk memanggil fungsi bruteForce() dengan masukan indeks nol, yang

berarti dimulai dari blok yang pertama. Sebelum pemanggilan bruteForce(), dilakukan transformBoard() yang bertujuan untuk mengubah seluruh petak pada papan dari ‘X’ menjadi ‘1’ serta dari ‘.’ menjadi ‘0’. Hal ini dilakukan untuk mempermudah pengecekan isBoardFull(), yaitu papan terisi penuh yang menunjukkan terdapat solusi valid, serta menghindari kesalahan deteksi papan kosong karena input dapat berupa huruf A sampai Z yang memungkinkan input huruf X akan terdeteksi sebagai papan kosong. Tujuan dari solve() secara garis besar untuk menentukan apakah persoalan dapat diselesaikan atau tidak.

```

private void transformBoard() {
    for (int r = 0; r < board.length; r++) {
        for (int c = 0; c < board[0].length; c++) {
            if (board[r][c] == 'X') {
                board[r][c] = '1';
            } else {
                board[r][c] = '0';
            }
        }
    }
}

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

```

Secara sederhana, transformBoard() mengubah seluruh papan yang awalnya memiliki format ‘X’ dan ‘.’ menjadi ‘1’ dan ‘0’. Metode isBoardFull() memeriksa seluruh baris dan kolom papan yang memiliki ‘1’ atau masih kosong.

```

private boolean bruteForce(int index) {
    if (index == blocks.size()) {
        return true;
    }
    Block block = blocks.get(index);
    List<char[][]> transformations = block.getVariants();

    for (char[][] shape : transformations) {
        for (int r = 0; r <= board.length - shape.length; r++) {
            for (int c = 0; c <= board[0].length - shape[0].length; c++) {
                jumlahKasus++;
                if (validPlace(shape, r, c)) {
                    placeBlock(shape, r, c, block.getLabel());
                }
            }
        }
    }
}

```

```

        if (bruteForce(index + 1)) return true;
        removeBlock(shape, r, c);
    }
}
}
return false;
}

```

Metode bruteForce() digunakan untuk menyelesaikan puzzle dengan pedekatan brute force, yaitu mencoba semua kemungkinan peletakkan blok yang ada. Fungsi bruteForce() menerima masukan index, yang menandakan urutan dari blok yang akan diproses. Pertama, fungsi memeriksa apakah semua blok telah digunakan. Jika sudah, maka algoritma selesai dan puzzle berhasil diselesaikan. Jika masih ada blok yang dapat digunakan, fungsi mengambil blok dari daftar blocks yang terletak pada index, lalu mengambil seluruh variasi blok tersebut ke dalam list of array 2D transformations. Selanjutnya, fungsi mencoba meletakkan setiap variasi blok pada semua posisi yang memungkinkan di papan. Proses ini dilakukan dengan mencoba setiap baris (r) dan kolom (c) yang memungkinkan berdasarkan ukuran papan dan ukuran blok. Fungsi validPlace() akan memeriksa apakah posisi tersebut valid. Jika valid, blok ditempatkan pada papan menggunakan placeBlock(), lalu fungsi bruteForce() memanggil dirinya secara rekursif (bruteForce(index+1)) untuk mencoba meletakkan blok selanjutnya. Jika setelah meletakkan blok tidak ditemukan solusi untuk blok berikutnya, maka blok tersebut dihapus dengan menggunakan removeBlock(), dan proses pencarian dilanjutkan dengan mencoba posisi lain atau variasi lainnya. Jika semua kemungkinan telah dicoba dan tidak ada solusi yang ditemukan, algoritma mengembalikan false, yang berarti permainan saat ini tidak memiliki solusi yang valid.

```

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

```

Fungsi validPlace() digunakan untuk memeriksa apakah sebuah blok dapat ditempatkan pada posisi tertentu di papan. Fungsi ini menerima tiga parameter: shape yang merupakan bentuk blok dalam array 2D, serta row dan col yang menunjukkan posisi blok di papan. Fungsi ini melakukan iterasi pada setiap elemen dalam shape. Jika elemen yang diperiksa

bukan karakter kosong (' '), maka posisi tersebut harus cocok dengan karakter '1' pada papan. Jika seluruh elemen blok dapat diletakkan dengan sesuai, fungsi akan mengembalikan true.

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

Metode placeBlock() digunakan untuk menempatkan sebuah blok pada papan. Metode ini menerima empat parameter: shape, yaitu bentuk blok dalam array 2D; row dan col, yang menunjukkan posisi awal di papan tempat blok akan diletakkan; serta label, yaitu karakter yang mewakili blok tersebut. Fungsi ini melakukan iterasi pada setiap elemen dalam shape. Jika elemen yang diperiksa bukan karakter kosong (' '), maka elemen tersebut akan ditempatkan di papan dengan mengganti nilai pada posisi (row + r, col + c) menjadi label. Dengan cara ini, blok akan mengisi area papan sesuai dengan bentuknya tanpa mengubah area yang kosong.

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

Metode removeBlock() digunakan untuk menghapus blok yang telah ditempatkan di papan dengan menggantinya kembali ke karakter default, yaitu '1'. Metode ini menerima tiga parameter: shape, yaitu bentuk blok dalam array 2D; row dan col, yang menunjukkan posisi blok di papan. Metode ini melakukan iterasi pada setiap elemen dalam shape, dan jika elemen tersebut bukan karakter kosong (' '), maka elemen yang sesuai pada papan akan dikembalikan menjadi '1'.

4. Menyimpan dan Menampilkan Solusi

```
private void printBoard() {
    String[] colors = { "\u001B[31m", "\u001B[32m", ... };
    for (char[] row : board) {
```

```

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

public static void savePapanToTxt(char[][] board, String filename) {
    for (char[] row : board) {
        StringBuilder rowString = new StringBuilder();
        for (char cell : row) {
            rowString.append(cell == '0' ? " " : cell);
        }
        writer.write(rowString.toString());
        writer.newLine();
    }
}

public static void savePapanToImage(char[][] board, String filename) {
    int cellSize = 50;
    int fontSize = 30;
    int width = board[0].length * cellSize;
    int height = board.length * cellSize;
    BufferedImage image = new BufferedImage(width, height,
    BufferedImage.TYPE_INT_ARGB);
    Graphics2D g2d = image.createGraphics();

    g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
    RenderingHints.VALUE_ANTIALIAS_ON);
    g2d.dispose();
}

```

Jika semua blok berhasil digunakan dan papan terisi penuh, solusi dianggap valid. Program akan menampilkan hasil solusi di terminal menggunakan warna untuk membedakan setiap blok. Program akan mencetak jumlah kasus yang diuji serta waktu yang dibutuhkan untuk menemukan solusi. Pengguna juga diberikan opsi untuk menyimpan hasil sebagai file teks atau gambar.

Dengan mengikuti langkah-langkah ini, program dapat menemukan solusi dengan mencoba semua kemungkinan penempatan blok hingga menemukan kombinasi yang benar atau menyatakan bahwa tidak ada solusi yang valid.

BAB 3

SOURCE CODE PROGRAM

3.1 Repository Program

Repository program dapat diakses melalui tautan *GitHub* berikut:

https://github.com/farrelathalla/Tucil1_13523118

3.2 Source Code Program

3.2.1 Block.java

```
import java.util.*;  
  
public class Block {  
    private char label;  
    private char[][] shape; // Bentuk block dalam array  
    private List<char[][]> variants; // Seluruh hasil transformasi pada block  
  
    public Block(char label, List<String> inputShape) {  
        this.label = label;  
        this.shape = shapeToArray(inputShape);  
  
        // Block tidak terhubung  
        if (!isConnected()) {  
            throw new IllegalArgumentException("Block " + label + " tidak terhubung!");  
        }  
  
        // Generate semua transformasi  
        generateVariants();  
    }  
  
    // Fungsi getter  
    public char getLabel() {  
        return label;  
    }  
  
    public char[][] getShape() {  
        return shape;  
    }  
  
    public List<char[][]> getVariants() {  
        return variants;  
    }  
  
    // Mengubah input ke array 2 dimensi  
    private char[][] shapeToArray(List<String> inputShape) {  
        int rows = inputShape.size();  
        int cols = 0;  
  
        // Mencari row terbesar sebagai patokan ukuran  
        for (String row : inputShape) {  
            cols = Math.max(cols, row.length());  
        }  
    }  
}
```

```

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

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

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

    if (startX == -1) return false; // Blok tidak valid

    // Algoritma BFS
    Queue<int[]> queue = new LinkedList<>();
    queue.add(new int[]{startX, startY});
    visited[startX][startY] = true;

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

    // Cek nyambung
    int[][] directions = {
        {1, 0}, {-1, 0}, {0, 1}, {0, -1}, // Atas, bawah, kiri, kanan
        {1, 1}, {1, -1}, {-1, 1}, {-1, -1} // Diagonal
    };
}

```

```

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

            for (int[] dir : directions) {
                int newX = curr[0] + dir[0];
                int newY = curr[1] + dir[1];

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

        return count == total;
    }

    private void generateVariants() {
        variants = new ArrayList<>();
        char[][] current = shape;

        // Rotasi 90 derajat sebanyak 4 kali
        for (int i = 0; i < 4; i++) {
            current = rotate(current);
            variants.add(current);
        }

        // Flip horizontal dan lakukan rotasi lagi
        current = flipHorizontal(shape);
        variants.add(current);
        for (int i = 0; i < 3; i++) {
            current = rotate(current);
            variants.add(current);
        }
    }

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

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

```

```

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

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

```

3.2.2 Solver.java

```

import java.util.List;

public class Solver {
    private char[][] board;
    private List<Block> blocks;
    private long jumlahKasus = 0;

    public Solver(char[][] board, List<Block> blocks) {
        this.board = board;
        this.blocks = blocks;
    }

    public boolean solve() {
        long start = System.currentTimeMillis(); // Start waktu
        transformBoard();
        // Algoritma Brute Force start dari index 0
        if (bruteForce(0) && isBoardFull()) {
            System.out.println("Solusi ditemukan:");
            printBoard();
            long end = System.currentTimeMillis();
            long totalTime = end - start;

            System.out.println("Waktu pencarian: " + totalTime + " ms");
            System.out.println("Banyak kasus yang ditinjau: " + jumlahKasus);
            return true;
        } else {
            System.out.println("Tidak ada solusi.");
            return false;
        }
    }
}

```

```

private boolean bruteForce(int index) {

    // Semua block digunakan
    if (index == blocks.size()) {
        return true;
    }

    Block block = blocks.get(index); // Mendapatkan block sekarang
    List<char[][]> transformations = block.getVariants(); // Mengambil seluruh variasi block

    // Mencoba meletakkan block pada setiap baris dan kolom yang mungkin
    // Jika tidak bisa, maka lakukan transformasi pada block dan coba kembali
    for (char[][] shape : transformations) {
        for (int r = 0; r <= board.length - shape.length; r++) {
            for (int c = 0; c <= board[0].length - shape[0].length; c++) {
                jumlahKasus++;

                // Jika block dapat diletakkan di papan
                if (validPlace(shape, r, c)) {
                    placeBlock(shape, r, c, block.getLabel());
                    // Jika block selanjutnya tidak dapat diletakkan, maka
                    // hilangkan block sekarang dan ulang proses
                    if (bruteForce(index + 1)) return true;

                    removeBlock(shape, r, c);
                }
            }
        }
    }

    return false;
}

private boolean validPlace(char[][] shape, int row, int col) {
    for (int r = 0; r < shape.length; r++) {
        for (int c = 0; c < shape[0].length; c++) {
            if (shape[r][c] != ' ') {
                // Block hanya dapat diletakkan pada '1'
                if (board[row + r][col + c] != '1') {
                    return false;
                }
            }
        }
    }
    return true;
}

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

```

```

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

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

```

```

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

    // Mengubah 'X' menjadi '1' dan '.' menjadi '0'
    private void transformBoard() {
        for (int r = 0; r < board.length; r++) {
            for (int c = 0; c < board[0].length; c++) {
                if (board[r][c] == 'X') {
                    board[r][c] = '1';
                } else {
                    board[r][c] = '0';
                }
            }
        }
    }

    // Cek apakah papan penuh
    private boolean isBoardFull() {
        for (int r = 0; r < board.length; r++) {
            for (int c = 0; c < board[0].length; c++) {
                if (board[r][c] == '1') {
                    return false;
                }
            }
        }
        return true;
    }
}

```

3.2.3 Main.java

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

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

        // Masukan nama file input
        System.out.print("Masukan nama file input (e.g., input.txt): ");
        String filename = scanner.nextLine().trim();

        filename = "test/" + filename;

        int[] ukuran = new int[2]; // Width dan height
        List<String> newPapan = new ArrayList<>();

        try {
            // Validasi nama file input
            if (!new File(filename).exists()) {
                throw new FileNotFoundException("File " + filename + " tidak ditemukan.");
            }

            // Membaca input dari file
            List<Block> blocks = IO.readFile(filename, ukuran, newPapan);
            char[][] papan = IO.createPapan(newPapan);

            // Solver puzzle
            Solver solver = new Solver(papan, blocks);
            boolean solved = solver.solve();

            if (solved) {
                while (true) {
                    // Menyimpan hasil di .txt
                    System.out.print("Apakah anda ingin menyimpan jawab dalam .txt? (y/n): ");
                    String saveTxt = scanner.nextLine().trim().toLowerCase();

                    if (saveTxt.equals("y")) {
                        String txtFilename;
                        while (true) {
                            System.out.print("Masukan nama file .txt (e.g., output.txt): ");
                            txtFilename = scanner.nextLine().trim();

                            // Validasi nama output dari txt
                            if (txtFilename.matches("^a-zA-Z0-9_\\-]+\\.txt$")) {
                                break;
                            } else {
                                System.out.println("Nama file harus berakhir dengan .txt, ulangi kembali!");
                            }
                        }
                        IO.savePapanToTxt(papan, txtFilename);
                        break;
                    } else if (saveTxt.equals("n")) {
                        break;
                    } else {
                        System.out.println("Input salah!");
                    }
                }
            }
        }
    }
}
```

```
        while (true) {
            // Menyimpan hasil di .png / .jpg
            System.out.print("Apakah anda ingin menyimpan jawab dalam gambar? (y/n): ");
            String saveImage = scanner.nextLine().trim().toLowerCase();

            if (saveImage.equals("y")) {
                String imageFilename;
                while (true) {
                    System.out.print("Masukan nama file gambar (e.g., output.png, output.jpg): ");
                    imageFilename = scanner.nextLine().trim();

                    // Validasi nama file gambar
                    if (imageFilename.matches("[a-zA-Z0-9_-]+\\.(png|jpg)$")) {
                        break;
                    } else {
                        System.out.println("Nama file harus berakhiri dengan .png atau .jpg, ulangi kembali!");
                    }
                }

                IO.savePapanToImage(papan, imageFilename);
                break;
            } else if (saveImage.equals("n")) {
                break;
            } else {
                System.out.println("Input salah!");
            }
        }
    }
} catch (IOException e) {
    System.out.println("Terjadi kesalahan: " + e.getMessage());
} catch (IllegalArgumentException e) {
    System.out.println("Terjadi kesalahan: " + e.getMessage());
}
}
```

3.2.4 IO.java

```
import java.awt.*;
import java.io.*;
import java.util.*;
import java.awt.image	BufferedImage;
import javax.imageio.ImageIO;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.File;
import java.util.List;

public class IO {
    public static List<Block> readFile(String filename, int[] ukuran, List<String> newPapan) throws IOException {
        List<Block> blocks = new ArrayList<>();
        Map<Character, Boolean> blockUsed = new HashMap<>();

        try (BufferedReader br = new BufferedReader(new FileReader(filename))) {
            String line = br.readLine();

            // Validasi file kosong
            if (line == null) throw new IllegalArgumentException("File kosong.");
            while (line != null) {
                String[] papan = line.split(" ");
                int[] ukur = new int[papan.length];
                for (int i = 0; i < papan.length; i++) {
                    ukur[i] = Integer.parseInt(papan[i]);
                }
                List<Character> baris = new ArrayList<>();
                for (int i = 0; i < ukur[0]; i++) {
                    char c = 'A' + i;
                    if (!blockUsed.containsKey(c)) {
                        baris.add(c);
                        blockUsed.put(c, true);
                    } else {
                        baris.add(' ');
                    }
                }
                blocks.add(new Block(ukur, baris));
                line = br.readLine();
            }
        }
    }
}
```

```

// Baris pertama input (width, height, jumlah blocks)
String[] params = line.split(" ");
if (params.length != 3) throw new IllegalArgumentException("Parameter ukuran tidak lengkap!");

int width = Integer.parseInt(params[0]);
int height = Integer.parseInt(params[1]);
int totalBlocks = Integer.parseInt(params[2]);

// Validasi 0 atau negatif
if (width <= 0 || height <= 0 || totalBlocks <= 0) {
    throw new IllegalArgumentException("Ukuran papan dan jumlah block harus lebih dari 0!");
}

ukuran[0] = width;
ukuran[1] = height;

// Baris kedua (jenis papan)
String tipePapan = br.readLine();
if (tipePapan == null) throw new IllegalArgumentException("Tidak memiliki jenis papan!");

// Papan DEFAULT
if ("DEFAULT".equals(tipePapan)) {
    // Mengisi papan dengan 'X'
    for (int i = 0; i < height; i++) {
        newPapan.add("X".repeat(width));
    }

    // Membuat blocks baru
    List<String> currentBlock = new ArrayList<>();
    Character currentLabel = null;

    // Validasi baris kosong
    String currentLine;
    while ((currentLine = br.readLine()) != null) {
        if (currentLine.trim().isEmpty()) {
            throw new IllegalArgumentException("Terdapat baris block kosong!");
        }

        // Validasi block duplicate
        char label = findLabel(currentLine);
        // Terdapat block baru
        if (currentLabel == null || label != currentLabel) {
            // Block baru
            if (currentLabel != null) {
                blocks.add(new Block(currentLabel, currentBlock));
                blockUsed.put(currentLabel, true);
            }

            // Block duplikat
            if (blockUsed.getOrDefault(label, false)) {
                throw new IllegalArgumentException("Terdapat duplikat block " + label + ".");
            }

            currentLabel = label;
            currentBlock = new ArrayList<>();
        }

        currentBlock.add(currentLine);
    }
}

```

```

// Masukan block terakhir
if (currentLabel != null) {
    blocks.add(new Block(currentLabel, currentBlock));
}

// Jumlah block tidak sesuai
if (blocks.size() != totalBlocks) {
    throw new IllegalArgumentException("Total block tidak sesuai!");
}
return blocks;

// Papan CUSTOM
} else if ("CUSTOM".equals(tipePapan)) {
    List<String> tempPapan = new ArrayList<>();
    boolean isPapanValid = true;

    // Membaca papan
    for (int i = 0; i < height; i++) {
        String barisPapan = br.readLine();
        if (barisPapan == null) {
            throw new IllegalArgumentException("Papan kosong!");
        }
        tempPapan.add(barisPapan);
    }

    final int widthAwal = width;
    final int heightAwal = height;

    // Validasi papan NxM
    if (tempPapan.size() != heightAwal || tempPapan.stream().anyMatch(lined -> lined.length() != widthAwal || !lined.matches("[X.]+")) {
        isPapanValid = false;
    }

    // Jika papan NxM valid
    if (isPapanValid) {
        newPapan.addAll(tempPapan);

        // Proses blocks
        List<String> currentBlock = new ArrayList<>();
        Character currentLabel = null;

        String currentLine;
        while ((currentline = br.readLine()) != null) {
            if (currentline.trim().isEmpty()) {
                throw new IllegalArgumentException("Terdapat baris block kosong!");
            }

            char label = findLabel(currentline);
            if (currentlabel == null || label != currentLabel) {
                if (currentlabel != null) {
                    blocks.add(new Block(currentLabel, currentBlock));
                    blockUsed.put(currentLabel, true);
                }

                if (blockUsed.getOrDefault(label, false)) {
                    throw new IllegalArgumentException("Terdapat duplikat block " + label + ".");
                }

                currentLabel = label;
                currentBlock = new ArrayList<>();
            }

            currentBlock.add(currentLine);
        }

        if (currentLabel != null) {
            blocks.add(new Block(currentLabel, currentBlock));
        }

        if (blocks.size() != totalBlocks) {
            throw new IllegalArgumentException("Total block tidak sesuai!");
        }
    }
    return blocks;
}

```

```

// Jika NxM gagal, coba MxN
BufferedReader brn = new BufferedReader(new FileReader(filename));
brn.readLine();
brn.readLine();

// Tukar ukuran
int newWidth = heightAwal;
int newHeight = widthAwal;
tempPapan.clear();

for (int i = 0; i < newHeight; i++) {
    String barisPapan = brn.readLine();
    if (barisPapan == null) {
        throw new IllegalArgumentException("Papan kosong!");
    }
    tempPapan.add(barisPapan);
}

// Validasi papan MxN
if (tempPapan.size() != newHeight || tempPapan.stream().anyMatch(lined -> lined.length() != newWidth || !lined.matches("[X.]+")) {
    throw new IllegalArgumentException("Ukuran papan salah!");
}

// Papan MxN benar
newPapan.addAll(tempPapan);
ukuran[0] = newWidth;
ukuran[1] = newHeight;

// Proses blocks
List<String> currentBlock = new ArrayList<>();
Character currentLabel = null;

String currentLine;
while ((currentLine = brn.readLine()) != null) {
    if (currentLine.trim().isEmpty()) {
        throw new IllegalArgumentException("Terdapat baris block kosong!");
    }

    char label = findLabel(currentLine);
    if (currentLabel == null || label != currentLabel) {
        if (currentLabel != null) {
            blocks.add(new Block(currentLabel, currentBlock));
            blockUsed.put(currentLabel, true);
        }

        if (blockUsed.getOrDefault(label, false)) {
            throw new IllegalArgumentException("Terdapat duplikat block " + label + ".");
        }

        currentLabel = label;
        currentBlock = new ArrayList<>();
    }

    currentBlock.add(currentLine);
}

if (currentLabel != null) {
    blocks.add(new Block(currentLabel, currentBlock));
}

if (blocks.size() != totalBlocks) {
    throw new IllegalArgumentException("Total block tidak sesuai!");
}
return blocks;
} else {
    throw new IllegalArgumentException("Jenis papan harus berupa 'DEFAULT' atau 'CUSTOM'!");
}
}

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

public static char[][] createPapan(List<String> barisPapan) {
    int height = barisPapan.size();
    int width = barisPapan.get(0).length();
    char[][] papan = new char[height][width];

    for (int r = 0; r < height; r++) {
        for (int c = 0; c < width; c++) {
            papan[r][c] = barisPapan.get(r).charAt(c);
        }
    }
    return papan;
}

```

```

public static void savePapanToTxt(char[][] board, String filename) {
    try (BufferedWriter writer = new BufferedWriter(new FileWriter(filename))) {
        for (char[] row : board) {
            StringBuilder rowString = new StringBuilder();
            for (char cell : row) {
                if (cell == '0') {
                    rowString.append(" ");
                } else {
                    rowString.append(cell);
                }
            }
            writer.write(rowString.toString());
            writer.newLine();
        }
        System.out.println("Solusi " + filename + " berhasil tersimpan.");
    } catch (IOException e) {
        System.out.println("Gagal menyimpan solusi: " + e.getMessage());
    }
}

public static void savePapanToImage(char[][] board, String filename) {
    int cellSize = 50;
    int fontSize = 30;

    int width = board[0].length * cellSize;
    int height = board.length * cellSize;

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

    // Menggambar papan
    g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
    for (int i = 0; i < board.length; i++) {
        for (int j = 0; j < board[i].length; j++) {
            char cell = board[i][j];

            // Memberi warna
            Color color = getColorForBlock(cell);
            g2d.setColor(color);
            g2d.fillRect(j * cellSize, i * cellSize, cellSize, cellSize);

            // Tulis huruf
            if (cell != '0' && cell != ' ') {
                g2d.setFont(new Font("Arial", Font.BOLD, fontSize));
                g2d.setColor(Color.BLACK);

                FontMetrics fm = g2d.getFontMetrics();
                int textWidth = fm.stringWidth(String.valueOf(cell));
                int textHeight = fm.getAscent();

                int textX = j * cellSize + (cellSize - textWidth) / 2;
                int textY = i * cellSize + (cellSize + textHeight) / 2 - 5;

                g2d.drawString(String.valueOf(cell), textX, textY);
            }
        }
    }

    g2d.dispose();

    try {
        ImageIO.write(image, "PNG", new File(filename));
        System.out.println("Solusi " + filename + " berhasil tersimpan.");
    } catch (IOException e) {
        System.out.println("Gagal menyimpan gambar: " + e.getMessage());
    }
}

```

```
private static Color getColorForBlock(char block) {
    int index = (Character.toUpperCase(block) - 'A') % 26;

    // Mapping warna
    switch (index) {
        case 0: return Color.RED;
        case 1: return Color.GREEN;
        case 2: return Color.YELLOW;
        case 3: return Color.BLUE;
        case 4: return Color.MAGENTA;
        case 5: return Color.CYAN;
        case 6: return Color.WHITE;
        case 7: return Color.DARK_GRAY;
        case 8: return Color.PINK;
        case 9: return Color.GREEN.darker();
        case 10: return Color.YELLOW.darker();
        case 11: return Color.BLUE.darker();
        case 12: return Color.MAGENTA.darker();
        case 13: return Color.CYAN.darker();
        case 14: return Color.WHITE;
        case 15: return new Color(255, 0, 0);
        case 16: return new Color(0, 255, 0);
        case 17: return new Color(255, 255, 0);
        case 18: return new Color(0, 0, 255);
        case 19: return new Color(255, 0, 255);
        case 20: return new Color(0, 255, 255);
        case 21: return new Color(255, 255, 255);
        case 22: return new Color(255, 140, 0);
        case 23: return new Color(255, 255, 102);
        case 24: return new Color(135, 206, 235);
        case 25: return new Color(144, 238, 144);
        default: return Color.BLACK;
    }
}
```

BAB 4

MASUKAN DAN LUARAN PROGRAM

4.1 Test Case 1

Isi input1.txt
<pre>5 5 7 DEFAULT A AA B BB C CC D DD EE EE E FF FF F GGG</pre>

Output penyelesaian input1.txt
<pre>Masukan nama file input (e.g., input.txt): input1.txt Solusi ditemukan: AABBD AEBDD EECCF EECFF GGGFF Waktu pencarian: 13 ms Banyak kasus yang ditinjau: 168552 Apakah anda ingin menyimpan jawab dalam .txt? (y/n): █</pre>

4.2 Test Case 2

Isi input2.txt

```
5 7 5
CUSTOM
...X...
.XXXXX.
XXXXXXXX
.XXXXX.
...X...
A
AAA
BB
BBB
CCCC
| C
D
EEE
E|
```

Output penyelesaian input2.txt

```
Masukan nama file input (e.g., input.txt): input2.txt
Solusi ditemukan:
E
AAEBB
DAEEBBB
ACCCC
C

Waktu pencarian: 7 ms
Banyak kasus yang ditinjau: 77987
```

4.3 Test Case 3

Isi input3.txt

```
9 9 12
CUSTOM
.XXXX....
XXXXX...
XXXXXXXXX
..XXXXXXXX
..XXXXXX
....XXXX
....XXXX.
A
A
AAA
BB
B
C
CCC
C
DD
DD
D
E
EE
E
FF
F
F
G
G
G
GG
H
HH
HH
I
III
I
J
JJJ
K K
KKK
LL
LLL
```

Output penyelesaian input3.txt

Solusi ditemukan:

AAAL

GABBL

GAEBLLJ

GEEKKLJ

GGEKCCJJD

FKKCCDD

FFFCDDI

HHIII

HHHI

Waktu pencarian: 236398 ms

Banyak kasus yang ditinjau: 41410456894

Apakah anda ingin menyimpan jawab dalam .txt? (y/n): |

4.4 Test Case 4

Isi input4.txt

```
13 2 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 penyelesaian input4.txt

```
Masukan nama file input (e.g., input.txt): input4.txt
Solusi ditemukan:
```

```
ABCDEFGHIJKLM
NOPQRSTUVWXYZ
```

```
Waktu pencarian: 8 ms
Banyak kasus yang ditinjau: 351
```

4.5 Test Case 5

Isi input5.txt

```
11 5 12
DEFAULT
AAA
A
A
| BB
BB
B
C
CC
C
DD
| DD
E
E
E
EE
| FF
FFF
G
GG
HHH
H H
| I
III
| J
JJ
| JJ
KKK
KK
LL
| L
| L
```

Output penyelesaian input5.txt

```
Masukan nama file input (e.g., input.txt): input5.txt
Solusi ditemukan:
```

```
AAABBCCCDEE
KKAIBBCDDFE
KKAITBJDFFE
HKHILJJJFGE
HHHILLLJFGG
```

```
Waktu pencarian: 653 ms
```

```
Banyak kasus yang ditinjau: 122079708
```

```
Apakah anda ingin menyimpan jawab dalam .txt? (y/n): █
```

4.6 Test Case 6

Isi input6.txt
<pre>5 5 6 DEFAULT A AA B BB C CC D DD EE EE E FF FF F GGG</pre>

Output penyelesaian input6.txt
<pre>Masukan nama file input (e.g., input.txt): input6.txt Terjadi kesalahan: Total block tidak sesuai!</pre>

4.7 Test Case 7

Isi input7.txt
<pre>5 5 7 DEFAULT A AA B BB C CC D DD EE EE FF FF F GGG</pre>

Output penyelesaian input7.txt
<pre>Masukan nama file input (e.g., input.txt): input7.txt Terjadi kesalahan: Terdapat baris block kosong!</pre>

4.8 Test Case 8

Isi input8.txt	<pre>0 0 0 DEFAULT A AA B BB C CC D DD EE EE E FF FF F GGG</pre>
----------------	--

Output penyelesaian input8.txt	<pre>Masukan nama file input (e.g., input.txt): input8.txt Terjadi kesalahan: Ukuran papan dan jumlah block harus lebih dari 0!</pre>
--------------------------------	---

4.9 Test Case 9

Isi input10.txt
<pre>5 5 7 DEFAULT A AAA B BB C CC D DD EE EE E FF FF F GGG</pre>

Output penyelesaian input10.txt
<pre>Masukan nama file input (e.g., input.txt): input10.txt Tidak ada solusi.</pre>

4.10 Test Case 10

Isi input11.txt
<pre>3 3 3 DEFAULT A A BBB C Y</pre>

Output penyelesaian input11.txt

Masukan nama file input (e.g., input.txt): input11.txt
Terjadi kesalahan: Block A tidak terhubung!

4.10 Test Case 11

Isi input12.txt

```
15 9 12
CUSTOM
....X...X...X...
..X.X.X.X.X.X.X.
.X.X.X.X.X.X.X.
X.X.X.X.X.X.X.X
.X.X.X.X.X.X.X.
X.X.X.X.X.X.X.X
.X.X.X.X.X.X.X.X
.X.X.X.X.X.X.X.X
..X.X.X.X.X.X.X.
...X...X...X...X...
A
A A
A
B B
B B B
C
C
C
C
D
D D
D
D
:
E
:
E
F
F
G
G
G
G
H
H
H
I
I
I I
I
J
J J
J
J
K K
K
K
L
L
L
L
```

Output penyelesaian input12.txt

Solusi ditemukan:

```
A   B   G  
L A F B G G  
L A F B G C G  
L A E F B C C K  
L E D B C K K  
L J E D C I K K  
J E D D I H H  
J J D I I H  
J   I   H
```

Waktu pencarian: 64378 ms

Banyak kasus yang ditinjau: 11911451780

BAB 5

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 <i>Graphical User Interface</i> (GUI)		✓
6	Program dapat menyimpan solusi dalam bentuk file gambar	✓	
7	Program dapat menyelesaikan kasus konfigurasi <i>custom</i>	✓	
8	Program dapat menyelesaikan kasus konfigurasi Piramida (3D)		✓
9	Program dibuat oleh saya sendiri	✓	