

IF2211 Strategi Algoritma
Laporan Tugas Kecil 1 – Penyelesaian IQ Puzzler Pro dengan
Algoritma Brute Force



Disusun oleh:

I Made Wiweka Putera – 13523160

PROGRAM STUDI TEKNIK INFORMATIKA SEKOLAH
TEKNIK ELEKTRO DAN INFORMATIKA INSTITUT
TEKNOLOGI BANDUNG
JL. GANESA 10, BANDUNG 40132

2025

Daftar Isi

Daftar Isi.....	2
BAB I PENDAHULUAN.....	3
1.1. Deskripsi Masalah.....	3
1.2. Algoritma Brute Force.....	3
BAB II PENERAPAN ALGORITMA BRUTE FORCE.....	4
2.1. Langkah-langkah Penyelesaian IQ Puzzler Pro.....	5
2.2. Pseudocode.....	6
BAB III SOURCE CODE.....	7
3.1. Arsitektur Program.....	7
3.2. Command Line Interface dan Logika Inti (Main.java).....	7
3.3. Graphical User Interface (PuzzleSolverGUI.java).....	20
BAB IV EKSPERIMEN.....	27
4.1. Tangkapan Layar Eksperimen.....	27
LAMPIRAN.....	30

BAB I

PENDAHULUAN

1.1 Deskripsi Masalah

Tugas ini bertujuan untuk mengembangkan program penyelesaian IQ Puzzler Pro berbasis command line interface dan graphical user interface (bonus) menggunakan bahasa pemrograman Java, dengan metode penyelesaian secara brute force murni (tanpa heuristik). Pada permainan ini, papan berukuran $N \times M$ awalnya kosong dan harus diisi seluruhnya dengan P blok puzzle unik (masing-masing diwakili oleh A-Z). Setiap blok dapat dirotasi dan dicerminkan. Program harus membaca file test case (.txt) yang berisi dimensi papan, jumlah blok, jenis konfigurasi (DEFAULT/CUSTOM/PYRAMID), dan bentuk tiap blok. Outputnya adalah tampilan solusi (dengan warna berbeda untuk tiap blok), waktu eksekusi pencarian, jumlah iterasi yang diuji, serta opsi untuk menyimpan solusi ke file .txt.

1.2 Algoritma Brute Force

Algoritma Brute Force adalah pendekatan yang mencoba setiap kemungkinan konfigurasi secara menyeluruh untuk menemukan solusi. Dalam konteks permainan IQ Puzzler Pro, papan yang awalnya kosong dan harus diisi dengan P blok puzzle unik, sehingga algoritma brute force murni akan mengeksplorasi seluruh kombinasi penempatan, rotasi, dan refleksi blok pada papan hingga seluruh papan terisi, atau menampilkan bahwa solusi tidak ditemukan jika tidak ada solusi yang mungkin dari puzzle.

BAB II

PENERAPAN ALGORITMA BRUTE FORCE

2.1 Langkah-langkah Penyelesaian IQ Puzzler Pro

1. Saat program dijalankan, program membaca file test case (.txt) yang dimasukkan oleh pengguna. Program menyimpan informasi N (jumlah baris), M (jumlah kolom), P (jumlah blok), dan bentuk-bentuk blok pada instans PuzzleData.
2. Setelah data selesai dibaca, program menginisialisasi papan kosong berukuran $N \times M$ di mana setiap sel ditandai dengan karakter titik ('.'). Program juga menginisialisasi variabel penghitung jumlah kasus yang ditinjau, yang awalnya bernilai 0.
3. Penyelesaian secara brute force diimplementasikan dengan rekursi backtracking. Algoritma melewati setiap sel papan satu per satu. Pada setiap sel, algoritma mencoba untuk meletakkan blok-blok yang belum diletakkan, dengan memeriksa setiap kemungkinan posisi. Untuk setiap blok yang tersedia, program akan mencoba setiap orientasi, meliputi rotasi 90° , 180° , dan 270° , serta tiap refleksi dan rotasinya.
4. Apabila blok tersebut dapat diletakkan pada posisi dan orientasi tertentu (yaitu, seluruh bagian blok masuk ke dalam batas papan dan tidak bertabrakan dengan blok lain), maka blok tersebut ditempatkan secara sementara di papan. Selanjutnya, algoritma akan melanjutkan ke blok berikutnya dan mengulangi proses yang sama.
5. Apabila pada suatu titik blok berikutnya tidak dapat diletakkan di posisi sel manapun dengan orientasi apapun, maka algoritma melakukan proses backtracking. Algoritma akan menghapus blok terakhir yang berhasil ditempatkan dan kembali ke titik sebelumnya untuk mencoba alternatif penempatan atau orientasi lain dari blok yang dihapus tersebut. Proses ini berlanjut secara rekursif, dengan algoritma berupaya menempuh semua kemungkinan penempatan hingga solusi ditemukan, atau hingga seluruh opsi telah dicoba.
6. Jika seluruh papan berhasil terisi tanpa ada yang bertumpang tindih (seluruh sel terisi blok), maka solusi telah ditemukan. Kemudian, solusi akhir dicetak dan ditampilkan sebagai representasi dari konfigurasi blok yang valid, beserta informasi tambahan seperti waktu pencarian dan jumlah kasus yang ditinjau oleh algoritma.
7. Secara keseluruhan, pendekatan brute force ini memastikan bahwa setiap kemungkinan penempatan dari setiap blok dengan setiap orientasinya akan diperiksa,

sehingga jika solusi ada, algoritma akan menemukannya melalui proses pencarian yang lengkap dengan backtracking apabila terjadi kegagalan penempatan.

2.2 Pseudocode

Berikut adalah pseudocode dari algoritma program ini.

```
BEGIN
  // 1. Input and File Validation
  DO
    PROMPT "Masukkan path dari input file: "
    filePath ← nextLine().trim()
    IF filePath is empty OR filePath does not end with ".txt"
  THEN
    PRINT "Masukkan file path yang valid."
    CONTINUE loop
  ENDIF
  f ← new File(filePath)
  IF f does not exist OR f is a directory THEN
    PRINT "Masukkan file path yang valid."
    CONTINUE loop
  ENDIF
  LOOP UNTIL a filePath is valid

  // 2. Read and Parse Test Case Data
  puzzleData ← readPuzzleData(filePath)
  N ← puzzleData.getN() // Number of rows
  M ← puzzleData.getM() // Number of columns
  blocks ← puzzleData.getBlocks()

  // 3. Initialize Board and Counter
  CREATE board[0..N-1][0..M-1] filled with '.'
  kasusCount ← 0

  // 4. Solve the Puzzle via Brute Force Backtracking
  startTime ← currentTimeMillis()
  solutionFound ← solvePuzzle(blocks, index = 0, boardStateList,
writer = null) // writer used for debugging purpose only
  endTime ← currentTimeMillis()
  duration ← endTime - startTime

  // 5. Handle the Outcome
  IF solutionFound THEN
    PRINT "Solusi ditemukan:"
    CALL printColoredBoard()
    PRINT "Waktu pencarian: " + duration + " ms"
    PRINT "Jumlah kasus yang ditinjau: " + kasusCount

    // Convert final board solution to a string for saving
    finalSolution ← boardToString(board)
```

```

// 6. Prompt User to Save the Final Solution
PROMPT "Simpan hasil? (Y/N): "
response ← readLine().trim().toUpperCase()
IF response equals "Y" THEN
    PROMPT "Masukkan full output file path: "
    outputPath ← readLine().trim()
    IF outputPath does not end with ".txt" THEN
        outputPath ← outputPath + ".txt"
    ENDIF
    OPEN file at outputPath for writing
    WRITE finalSolution to file
    PRINT "Hasil disimpan ke " + outputPath
ENDIF
ELSE
    PRINT "Tidak ada solusi yang ditemukan."
    PRINT "Waktu pencarian: " + duration + " ms"
    PRINT "Jumlah kasus yang ditinjau: " + kasusCount
ENDIF
END

```

BAB III

SOURCE CODE

3.1 Arsitektur Program

Program dibagi menjadi dua bagian utama:

- Main.java: Berisi logika inti algoritma penyelesaian puzzle dan antarmuka Command-Line (CLI).
- PuzzleSolverGUI.java: Berisi kode antarmuka grafis (GUI) menggunakan Swing, yang menggunakan logika penyelesaian dari Main.java.

3.2 Command Line Interface dan Logika Inti (Main.java)

Berikut adalah source code dari modul Main.java beserta penjelasannya.

A. Import Statements dan Deklarasi Kelas

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

class PuzzleData {
    private int N, M, P;
    private String S;
    private List<Block> blocks;

    public PuzzleData(int N, int M, int P, String S, List<Block>
blocks) {
        this.N = N;
        this.M = M;
        this.P = P;
        this.S = S;
        this.blocks = blocks;
    }

    public int getN() { return N; }
    public int getM() { return M; }
    public int getP() { return P; }
    public String getS() { return S; }
    public List<Block> getBlocks() { return blocks; }
}

class Block {
    private char identifier;
    private char[][] shape;
```

```

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

public char getIdentifier() { return identifier; }
public char[][] getShape() { return shape; }
}

```

Bagian awal ini bertugas untuk mengimport library yang diperlukan dan mendefinisikan struktur data dasar untuk mewakili data puzzle, yakni PuzzleData yang menyimpan ukuran papan dan daftar blok, serta kelas Block yang menyimpan detail masing-masing blok (identifikasi dan bentuk). Data inilah yang akan digunakan oleh program untuk menginisialisasi papan dan kemudian mencoba menyelesaikan puzzle melalui algoritma brute force backtracking.

B. Deklarasi Variabel Global

```

public class Main {
    public static char[][] board;
    public static long kasusCount = 0;

    public static String resetColor = "\u001B[0m";
    public static Map<Character, Color> shapeColor = new HashMap<>();

    static {
        shapeColor.put('A', new Color(128, 0, 0)); // Maroon
        shapeColor.put('B', new Color(0, 128, 0)); // Green
        shapeColor.put('C', new Color(128, 128, 0)); // Olive
        shapeColor.put('D', new Color(0, 0, 128)); // Navy
        shapeColor.put('E', new Color(128, 0, 128)); // Purple
        shapeColor.put('F', new Color(0, 128, 128)); // Teal
        shapeColor.put('G', new Color(192, 192, 192)); // Silver
        shapeColor.put('H', new Color(128, 128, 128)); // Grey
        shapeColor.put('I', new Color(255, 0, 0)); // Red
        shapeColor.put('J', new Color(0, 255, 0)); // Lime
        shapeColor.put('K', new Color(255, 255, 0)); // Yellow
        shapeColor.put('L', new Color(0, 0, 255)); // Blue
        shapeColor.put('M', new Color(255, 0, 255)); // Fuchsia
        shapeColor.put('N', new Color(0, 255, 255)); // Aqua
        shapeColor.put('O', new Color(255, 255, 255)); // White
        shapeColor.put('P', new Color(0, 0, 0)); // Grey0
        shapeColor.put('Q', new Color(0, 0, 95)); // NavyBlue
        shapeColor.put('R', new Color(0, 0, 135)); // DarkBlue
        shapeColor.put('S', new Color(0, 0, 175)); // Blue3
    }
}

```



```

        shapeColor.put('T', new Color(0, 0, 215)); // Blue3
        shapeColor.put('U', new Color(0, 0, 255)); // Blue1
        shapeColor.put('V', new Color(0, 95, 0)); // DarkGreen
        shapeColor.put('W', new Color(0, 95, 95)); // DeepSkyBlue4
        shapeColor.put('X', new Color(0, 95, 135)); // DeepSkyBlue4
        shapeColor.put('Y', new Color(0, 95, 175)); // DeepSkyBlue4
        shapeColor.put('Z', new Color(0, 95, 215)); // DodgerBlue3
    }

```

Bagian ini bertugas mendefinisikan variabel global pada kelas Main yang akan digunakan di seluruh program. Secara singkat:

- Variabel board adalah representasi papan puzzle dalam bentuk array karakter dua dimensi.
- Variabel kasusCount digunakan untuk menghitung jumlah kasus yang ditinjau selama proses pencarian solusi.
- Variabel resetColor adalah string escape untuk mereset warna output (jika menggunakan ANSI color).
- Map shapeColor menyimpan asosiasi antara karakter blok (dari 'A' sampai 'Z') dengan warna tertentu. Inisialisasi map ini dilakukan dalam static block, sehingga setiap karakter dipetakan secara default ke warna yang telah ditentukan, yang nantinya digunakan untuk pewarnaan blok pada tampilan (CLI/GUI).

C. Fungsi main

```

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    String filePath;
    while (true) {
        System.out.print("Masukkan path dari input file: ");
        filePath = scanner.nextLine().trim();
        if (filePath.isEmpty() ||
!filePath.toLowerCase().endsWith(".txt")) {
            System.out.println("Masukkan file path yang valid.");
            continue;
        }
        File f = new File(filePath);
        if (!f.exists() || f.isDirectory()) {
            System.out.println("Masukkan file path yang valid.");
            continue;
        }
        break;
    }
}

```

```

        PuzzleData puzzleData = readPuzzleData(filePath);

        board = new char[puzzleData.getN()][puzzleData.getM()];
        for (int i = 0; i < puzzleData.getN(); i++) {
            for (int j = 0; j < puzzleData.getM(); j++) {
                board[i][j] = '.';
            }
        }

        long startTime = System.currentTimeMillis();
        List<char[][]> boardState = new ArrayList<>();
        boolean solved = solvePuzzle(puzzleData.getBlocks(), 0,
boardState, null);
        long endTime = System.currentTimeMillis();

        if (solved) {
            System.out.println("Solusi ditemukan:");
            System.out.println();
            printColoredBoard();
            System.out.println();
            System.out.println("Waktu pencarian: " + (endTime -
startTime) + " ms");
            System.out.println("Jumlah kasus yang ditinjau: " +
kasusCount);

            System.out.print("Simpan hasil? (Y/N): ");
            String saveChoice =
scanner.nextLine().trim().toUpperCase();
            if (saveChoice.equals("Y")) {
                System.out.print("Masukkan file path output hasil:
");

                String outputPath = scanner.nextLine().trim();
                if (!outputPath.toLowerCase().endsWith(".txt")) {
                    outputPath += ".txt";
                }
                try (PrintWriter pw = new PrintWriter(outputPath)) {
                    pw.print(boardToString(board));
                    System.out.println("Hasil disimpan ke " +
outputPath);
                } catch (Exception ex) {
                    ex.printStackTrace();
                }
            }
            } else {
                System.out.println("Tidak ada solusi yang ditemukan.");
                System.out.println("Waktu pencarian: " + (endTime -
startTime) + " ms");
                System.out.println("Jumlah kasus yang ditinjau: " +
kasusCount);
            }
        }

```

```
        scanner.close();  
    }
```

a. Input File Path:

- Scanner scanner digunakan untuk membaca input.
- Variabel filePath (tipe String) menyimpan input path file.
- Loop while memastikan filePath tidak kosong, berakhiran ".txt", dan menunjuk ke file yang ada (bukan direktori) dengan menggunakan objek File.

b. Pembacaan Data Puzzle:

- Fungsi readPuzzleData(filePath) dipanggil untuk membaca file dan menghasilkan objek PuzzleData.
- Objek ini memuat ukuran papan (N & M), jumlah blok (P), nilai string S, dan daftar blok (List<Block> blocks).

c. Inisialisasi Papan (board):

- Variabel board (tipe char[][] diinisialisasi dengan ukuran puzzleData.getN() x puzzleData.getM().
- Dua nested loop mengisi semua sel papan dengan karakter titik ('.') sebagai penanda kosong.

d. Proses Penyelesaian Puzzle dan Pengukuran Waktu:

- Waktu mulai (startTime) dicatat menggunakan System.currentTimeMillis().
- Fungsi solvePuzzle(...) dipanggil dengan parameter:
 - Daftar blok dari puzzleData.getBlocks(),
 - Indeks awal (0),
 - List kosong boardState,
 - Parameter writer sebagai null.
- Variabel global kasusCount diperbarui untuk mengukur jumlah percobaan penempatan.
- Setelah pemanggilan selesai, waktu selesai (endTime) diambil; durasi pencarian adalah selisih endTime dan startTime.

e. Menampilkan Hasil dan Pilihan Simpan Hasil:

- Jika solvePuzzle mengembalikan true (solusi ditemukan):
 - Pesan "Solusi ditemukan:" dicetak.

- Fungsi `printColoredBoard()` dipanggil untuk menampilkan papan berwarna di konsol.
 - Informasi waktu pencarian dan `kasusCount` dicetak.
 - Program meminta input "Simpan hasil? (Y/N):".
 - Jika pengguna memasukkan "Y", program meminta file output, menambahkan ekstensi ".txt" bila perlu, lalu menulis hasil akhir (dihasilkan oleh `boardToString(board)`) ke file tersebut dengan `PrintWriter`.
 - Jika solusi tidak ditemukan, pesan "Tidak ada solusi yang ditemukan" bersama dengan informasi waktu pencarian dan jumlah kasus ditampilkan.
- f. Penutupan Resource:
- `Scanner scanner` ditutup dengan `scanner.close()` untuk melepaskan resource.

D. Implementasi Konversi Warna ke ANSI dan Pencetakan Papan Berwarna

```
private static String boardToString(char[][] board) {
    if (board == null) return "";
    StringBuilder sb = new StringBuilder();
    for (char[] row : board) {
        for (char c : row) {
            sb.append(c);
        }
        sb.append("\n");
    }
    return sb.toString();
}

public static String toAnsi(Color color) {
    if (color.equals(new Color(128, 0, 0))) return "\u001B[31m";
    else if (color.equals(new Color(0, 128, 0))) return
"\u001B[32m";
    else if (color.equals(new Color(128, 128, 0))) return
"\u001B[33m";
    else if (color.equals(new Color(0, 0, 128))) return
"\u001B[34m";
    else if (color.equals(new Color(128, 0, 128))) return
"\u001B[35m";
    else if (color.equals(new Color(0, 128, 128))) return
"\u001B[36m";
    else if (color.equals(new Color(192, 192, 192))) return
"\u001B[37m";
    else if (color.equals(new Color(128, 128, 128))) return
"\u001B[90m";
    else if (color.equals(new Color(255, 0, 0))) return
```

```

"\u001B[91m";
    else if (color.equals(new Color(0, 255, 0))) return
"\u001B[92m";
    else if (color.equals(new Color(255, 255, 0))) return
"\u001B[93m";
    else if (color.equals(new Color(0, 0, 255))) return
"\u001B[94m";
    else if (color.equals(new Color(255, 0, 255))) return
"\u001B[95m";
    else if (color.equals(new Color(0, 255, 255))) return
"\u001B[96m";
    else if (color.equals(new Color(255, 255, 255))) return
"\u001B[97m";
    else if (color.equals(new Color(0, 0, 0))) return
"\u001B[30m";
    else if (color.equals(new Color(0, 0, 95))) return
"\u001B[34m";
    else if (color.equals(new Color(0, 0, 135))) return
"\u001B[34m";
    else if (color.equals(new Color(0, 0, 175))) return
"\u001B[94m";
    else if (color.equals(new Color(0, 0, 215))) return
"\u001B[94m";
    else if (color.equals(new Color(0, 0, 255))) return
"\u001B[94m";
    else if (color.equals(new Color(0, 95, 0))) return
"\u001B[32m";
    else if (color.equals(new Color(0, 95, 95))) return
"\u001B[36m";
    else if (color.equals(new Color(0, 95, 135))) return
"\u001B[34m";
    else if (color.equals(new Color(0, 95, 175))) return
"\u001B[94m";
    else if (color.equals(new Color(0, 95, 215))) return
"\u001B[94m";
    else return "\u001B[0m";
}

public static void printColoredBoard() {
    for (int i = 0; i < board.length; i++) {
        for (int j = 0; j < board[i].length; j++) {
            char cell = board[i][j];
            Color c = Main.shapeColor.getOrDefault(cell,
Color.WHITE);
            String ansiColor = toAnsi(c);
            System.out.print(ansiColor + cell + resetColor);
        }
        System.out.println();
    }
}

```

- a. boardToString:
 - Mengonversi array 2D (papan) menjadi string, dengan tiap baris diakhiri "\n".
- b. toAnsi:
 - Menerima objek Color dan membandingkannya dengan nilai-nilai warna tertentu, lalu mengembalikan kode ANSI yang sesuai untuk mencetak warna di konsol.
- c. printColoredBoard:
 - Mengiterasi setiap sel pada papan. Untuk tiap sel, mengambil warna yang sesuai dari map shapeColor (default Color.WHITE jika tidak ada).
 - Memanggil toAnsi untuk mendapatkan kode ANSI, kemudian mencetak karakter sel bersama kode tersebut dan mengembalikan ke warna default (resetColor).
 - Setelah setiap baris, mencetak newline untuk pemisahan baris.

E. Algoritma Penyelesaian Puzzle dengan Brute Force Backtracking

```
public static boolean solvePuzzle(List<Block> blocks, int index,
List<char[][]> boardState, PrintWriter writer) {
    if (isBoardFilled()) {
        if (index < blocks.size()) {
            writer.println("Jumlah blok melebihi kapasitas
papan.");
            writer.println();
            return false;
        }
        // printBoardStateToFile(copyBoard(board), writer);
        // writer.println();
        // writer.flush();
        return true;
    }
    if (index == blocks.size()) {
        return false;
    }

    Block currentBlock = blocks.get(index);
    for (int row = 0; row < board.length; row++) {
        for (int col = 0; col < board[0].length; col++) {
            for (char[][] orientation :
generateOrientations(currentBlock.getShape())) {
                if (validPlace(orientation, row, col)) {
                    placeBlock(orientation, row, col,
currentBlock.getIdentifier());
                    kasusCount++;
                }
            }
        }
    }
}
```

```

        if (solvePuzzle(blocks, index + 1,
boardState, writer)) {
            return true;
        }
        removeBlock(orientation, row, col);
    }
}
}
}
return false;
}

private static boolean isBoardFilled() {
    for (int i = 0; i < board.length; i++) {
        for (int j = 0; j < board[i].length; j++) {
            if (board[i][j] == '.') {
                return false;
            }
        }
    }
    return true;
}

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

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

```

```

    }

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

    private static List<char[][]> generateOrientations(char[][]
shape) {
        List<char[][]> orientations = new ArrayList<>();

        // Original shape rotations
        char[][] rot90 = rotate90(shape);
        char[][] rot180 = rotate90(rot90);
        char[][] rot270 = rotate90(rot180);

        orientations.add(shape);
        orientations.add(rot90);
        orientations.add(rot180);
        orientations.add(rot270);

        // Reflect shape & reflected shape rotations
        char[][] reflected = reflect(shape);
        char[][] refRot90 = rotate90(reflected);
        char[][] refRot180 = rotate90(refRot90);
        char[][] refRot270 = rotate90(refRot180);

        orientations.add(reflected);
        orientations.add(refRot90);
        orientations.add(refRot180);
        orientations.add(refRot270);

        return orientations;
    }

    private static char[][] rotate90(char[][] shape) {
        int m = shape.length;
        int n = shape[0].length;
        char[][] rotated = new char[n][m];
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                rotated[j][m - 1 - i] = shape[i][j];
            }
        }
        return rotated;
    }

```



```

    }

    private static char[][] reflect(char[][] shape) {
        int m = shape.length;
        int n = shape[0].length;
        char[][] reflected = new char[m][n];
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                reflected[i][j] = shape[i][n - 1 - j];
            }
        }
        return reflected;
    }
}

```

a. solvePuzzle(...):

- Fungsi rekursif yang berupaya menempatkan blok-blok secara berurutan pada papan.
- Jika papan sudah penuh (dicek oleh isBoardFilled()), kemudian jika jumlah blok yang digunakan kurang dari total blok, hasilnya dianggap salah (kemungkinan terjadi kelebihan blok) dan mengembalikan false; jika tidak, mengembalikan true sebagai solusi.
- Jika indeks blok telah mencapai akhir daftar tanpa mengisi papan, mengembalikan false.
- Untuk setiap sel pada papan, fungsi mencoba setiap orientasi dari blok saat ini (menggunakan generateOrientations()).
- Jika penempatan valid (dicek oleh validPlace()), blok diletakkan dengan placeBlock(), dan jumlah percobaan (kasusCount) diperbarui.
- Selanjutnya, fungsi rekursif dipanggil untuk blok berikutnya. Jika gagal, blok dihapus (dengan removeBlock()) dan alternatif lain dicoba.

b. isBoardFilled():

Mengecek apakah seluruh sel pada papan sudah diisi (tidak ada titik '.' lagi).

c. validPlace(...):

Mengecek apakah suatu orientasi blok dapat ditempatkan di posisi (row, col) tanpa keluar batas papan dan tanpa menumpuk blok lain.

d. placeBlock(...) dan removeBlock(...):

Menempatkan atau menghapus blok pada papan berdasarkan orientasi yang diberikan.

- placeBlock mengisi sel dengan identifier blok.

- removeBlock mengembalikan sel ke kondisi kosong ('.').
- e. generateOrientations(...):
- Menghasilkan semua orientasi valid dari sebuah blok, meliputi:
- Rotasi 90°, 180°, dan 270° dari bentuk asli, serta
 - Bentuk refleksi (cermin) dan rotasinya.
- f. rotate90(...) dan reflect(...):
- Fungsi utilitas untuk:
- rotate90: Menghasilkan array 2D baru yang merupakan hasil rotasi 90° dari bentuk input.
 - reflect: Menghasilkan bentuk cermin dari array 2D sehingga kolom-kolomnya dibalik.

F. Parsing File Test Case

```
public static PuzzleData readPuzzleData(String filePath) {
    int N = 0, M = 0, P = 0;
    String S = "";
    List<Block> blocks = new ArrayList<>();

    try (Scanner fileScanner = new Scanner(new File(filePath))) {
        // 1. Read N M P
        if (fileScanner.hasNextLine()) {
            String[] firstLine =
fileScanner.nextLine().trim().split("\\s+");
            N = Integer.parseInt(firstLine[0]);
            M = Integer.parseInt(firstLine[1]);
            P = Integer.parseInt(firstLine[2]);
        }

        // 2. Read S
        if (fileScanner.hasNextLine()) {
            S = fileScanner.nextLine().trim();
        }

        // 3. Read remaining lines and group by block.
        List<List<String>> groups = new ArrayList<>();
        List<String> currentGroup = new ArrayList<>();
        while (fileScanner.hasNextLine()) {
            String line = fileScanner.nextLine();
            if (line.trim().isEmpty()) {
                continue;
            }
            char firstChar = line.stripLeading().charAt(0);
            if (currentGroup.isEmpty()) {
```

```

        currentGroup.add(line);
    } else {
        char firstCharCurrentGroup =
currentGroup.get(0).stripLeading().charAt(0);
        if (firstChar == firstCharCurrentGroup) {
            currentGroup.add(line);
        } else {
            groups.add(currentGroup);
            currentGroup = new ArrayList<>();
            currentGroup.add(line);
        }
    }
}
if (!currentGroup.isEmpty()) {
    groups.add(currentGroup);
}

for (List<String> group : groups) {
    int maxWidth = 0;
    for (String shapeLine : group) {
        if (shapeLine.length() > maxWidth) {
            maxWidth = shapeLine.length();
        }
    }

    char[][] shapeArray = new
char[group.size()][maxWidth];
    for (int i = 0; i < group.size(); i++) {
        String rowText = group.get(i);
        rowText = rowText.replace(' ', '.');
        while (rowText.length() < maxWidth) {
            rowText += ".";
        }
        shapeArray[i] = rowText.toCharArray();
    }

    char id = group.get(0).stripLeading().charAt(0);
    blocks.add(new Block(id, shapeArray));
}
} catch (FileNotFoundException e) {
    e.printStackTrace();
}

return new PuzzleData(N, M, P, S, blocks);
}
}

```

Fungsi ini membaca file test case dari filePath dengan Scanner. Pertama, ia membaca ukuran papan (N, M, P) dan string S, lalu mengelompokkan baris-baris berikutnya

berdasarkan identifier blok. Untuk setiap grup, fungsi menentukan lebar maksimal, mengganti spasi dengan titik, dan membuat array 2D karakter sebagai bentuk blok. Akhirnya, semua blok dikumpulkan dan dikemas ke dalam objek PuzzleData yang dikembalikan.

3.3 Graphical User Interface (PuzzleSolverGUI.java)

A. GUI

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.*;
import java.util.ArrayList;

public class PuzzleSolverGUI {
    private static BoardPanel boardPanel;
    private static JLabel solveStatusLabel, timeLabel, countLabel;
    private static JButton saveButton;
    private static String finalSolution = "";

    public static void main(String[] args) {
        Main.board = new char[0][0];

        JFrame frame = new JFrame("IQ Puzzler Pro Solver");
        frame.setSize(920, 540);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JPanel panel = new JPanel(new GridBagLayout());
        frame.add(panel);

        boardPanel = new BoardPanel(Main.board);

        placeComponents(panel);

        frame.setVisible(true);
    }

    private static void placeComponents(JPanel panel) {
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(5, 5, 5, 5);
        gbc.anchor = GridBagConstraints.CENTER;

        // Row 0: File Path panel
        JPanel filePathPanel = new JPanel(new
FlowLayout(FlowLayout.CENTER));
        JLabel fileLabel = new JLabel("File Path:");
        JTextField filePathField = new JTextField(20);
```

```

        JButton browseButton = new JButton("Browse");

        browseButton.addActionListener(e -> {
            JFileChooser fc = new JFileChooser();
            // Optional: restrict to .txt files
            fc.setFileFilter(new
javax.swing.filechooser.FileNameExtensionFilter("Text Files",
"txt"));

            int choice = fc.showOpenDialog(panel);
            if (choice == JFileChooser.APPROVE_OPTION) {
                File file = fc.getSelectedFile();
                filePathField.setText(file.getAbsolutePath());
            }
        });

        filePathPanel.add(fileLabel);
        filePathPanel.add(filePathField);
        filePathPanel.add(browseButton);
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.gridwidth = 3;
        gbc.fill = GridBagConstraints.HORIZONTAL;
        panel.add(filePathPanel, gbc);

        // Row 1: Solve Puzzle Button
        JButton solveButton = new JButton("Solve Puzzle");
        gbc.gridx = 0;
        gbc.gridy = 1;
        gbc.gridwidth = 3;
        gbc.fill = GridBagConstraints.NONE;
        panel.add(solveButton, gbc);

        // Row 2: Board Panel
        JPanel boardContainer = new JPanel(new
FlowLayout(FlowLayout.CENTER));
        boardContainer.add(boardPanel);
        gbc.gridx = 0;
        gbc.gridy = 2;
        gbc.gridwidth = 3;
        gbc.fill = GridBagConstraints.NONE;
        panel.add(boardContainer, gbc);

        // Row 3: Information labels
        JPanel infoPanel = new JPanel(new GridLayout(3, 1, 0, 5));
        solveStatusLabel = new JLabel("", SwingConstants.CENTER);
        timeLabel = new JLabel("", SwingConstants.CENTER);
        countLabel = new JLabel("", SwingConstants.CENTER);
        infoPanel.add(solveStatusLabel);
        infoPanel.add(timeLabel);
        infoPanel.add(countLabel);
        gbc.gridx = 0;

```

```

gbc.gridy = 3;
gbc.gridwidth = 3;
gbc.fill = GridBagConstraints.HORIZONTAL;
panel.add(infoPanel, gbc);

// Row 4: Save Results Button
saveButton = new JButton("Simpan Hasil");
saveButton.setVisible(false);
gbc.gridx = 0;
gbc.gridy = 4;
gbc.gridwidth = 3;
gbc.fill = GridBagConstraints.NONE;
panel.add(saveButton, gbc);

// Save button action
saveButton.addActionListener(e -> {
    JFileChooser fc = new JFileChooser();
    fc.setDialogTitle("Simpan Hasil");
    int choice = fc.showSaveDialog(panel);
    if (choice == JFileChooser.APPROVE_OPTION) {
        File file = fc.getSelectedFile();
        if (!file.getName().toLowerCase().endsWith(".txt")) {
            file = new File(file.getAbsolutePath() + ".txt");
        }
        try (PrintWriter pw = new PrintWriter(file)) {
            pw.print(finalSolution);
            JOptionPane.showMessageDialog(panel, "Hasil
disimpan ke " + file.getName());
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
});

// Solve button action
solveButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        solveStatusLabel.setText("");
        timeLabel.setText("");
        countLabel.setText("");
        saveButton.setVisible(false);
        finalSolution = "";

        String filePath = filePathField.getText().trim();
        if (filePath.isEmpty() ||
!filePath.toLowerCase().endsWith(".txt")) {
            solveStatusLabel.setText("Masukkan file path yang
valid.");
            return;
        }
    }
});

```

```

        File f = new File(filePath);
        if (!f.exists() || f.isDirectory()) {
            solveStatusLabel.setText("Masukkan file path yang
valid.");
            return;
        }

        PuzzleData puzzleData =
Main.readPuzzleData(filePath);
        Main.board = new
char[puzzleData.getN()][puzzleData.getM()];
        for (int i = 0; i < puzzleData.getN(); i++) {
            for (int j = 0; j < puzzleData.getM(); j++) {
                Main.board[i][j] = '.';
            }
        }
        Main.kasusCount = 0;

        boardPanel.board = Main.board;
        int cellSize = 40;
        int cellGap = 2;
        int panelWidth = puzzleData.getM() * (cellSize +
cellGap);
        int panelHeight = puzzleData.getN() * (cellSize +
cellGap);
        boardPanel.setPreferredSize(new Dimension(panelWidth,
panelHeight));
        boardPanel.revalidate();
        boardPanel.repaint();

        SwingWorker<Boolean, Void> worker = new
SwingWorker<Boolean, Void>() {
            long startTime = System.currentTimeMillis();
            Timer timer;

            @Override
            protected Boolean doInBackground() throws
Exception {
                timer = new Timer(300, new ActionListener() {
                    public void actionPerformed(ActionEvent
evt) {
                        countLabel.setText("Jumlah kasus yang
ditinjau: " + Main.kasusCount);
                    }
                });
                timer.start();
                boolean solved =
Main.solvePuzzle(puzzleData.getBlocks(), 0, new ArrayList<>(), null);
                if (solved) {
                    finalSolution =

```

```

boardToString(Main.board);
    }
    return solved;
}

@Override
protected void done() {
    if (timer != null) {
        timer.stop();
    }
    long endTime = System.currentTimeMillis();
    try {
        boolean solved = get();
        if (solved) {
            solveStatusLabel.setText("Solusi
ditemukan!");
            saveButton.setVisible(true);
        } else {
            solveStatusLabel.setText("Tidak ada
solusi yang ditemukan.");
        }
        timeLabel.setText("Waktu pencarian: " +
(endTime - startTime) + " ms");
        countLabel.setText("Jumlah kasus yang
ditinjau: " + Main.kasusCount);
        boardPanel.repaint();
    } catch (Exception ex) {
        ex.printStackTrace();
    }
}

};
worker.execute();
});
}

private static String boardToString(char[][] board) {
    if (board == null) return "";
    StringBuilder sb = new StringBuilder();
    for (char[] row : board) {
        for (char c : row) {
            sb.append(c);
        }
        sb.append("\n");
    }
    return sb.toString();
}

static class BoardPanel extends JPanel {
    char[][] board;

```



```

    public BoardPanel(char[][] board) {
        this.board = board;
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        if (board == null) return;
        Graphics2D g2d = (Graphics2D) g;
        int cellSize = 40;
        int arcSize = 15;
        int cellGap = 2;
        for (int i = 0; i < board.length; i++) {
            for (int j = 0; j < board[i].length; j++) {
                char cell = board[i][j];
                Color color = Main.shapeColor.getOrDefault(cell,
Color.WHITE);

                int x = j * (cellSize + cellGap);
                int y = i * (cellSize + cellGap);
                g2d.setColor(color);
                g2d.fillRoundRect(x, y, cellSize, cellSize,
arcSize, arcSize);
                g2d.setColor(Color.WHITE);
                g2d.drawString(String.valueOf(cell), x + cellSize
/ 2 - 5, y + cellSize / 2 + 5);
            }
        }
    }
}

```

a. Frame & Panel Utama:

- JFrame dibuat dengan judul "IQ Puzzler Pro Solver" dan ukuran 920×540.
- JPanel utama menggunakan GridBagLayout untuk mengatur layout komponen.

b. Input File Path:

- Baris 0 menampilkan panel input yang berisi JLabel ("File Path:"), JTextField, dan tombol "Browse" (dibuka JFileChooser untuk memilih file .txt).

c. Solve Puzzle & Informasi:

- Tombol "Solve Puzzle" (baris 1) memulai proses penyelesaian.
- Saat ditekan, program memvalidasi file path, membaca data puzzle (dengan Main.readPuzzleData), menginisialisasi papan (board), dan mengatur ukuran BoardPanel.

- Proses solving dijalankan oleh `SwingWorker`, yang menjalankan metode `solvePuzzle` secara background.
 - Timer di `SwingWorker` memperbarui label jumlah kasus secara periodik.
 - Setelah selesai, status solusi, waktu pencarian, dan jumlah kasus ditampilkan; jika solusi ditemukan, tombol "Simpan Hasil" muncul untuk menyimpan output ke file.

d. `BoardPanel` & Pencetakan Board:

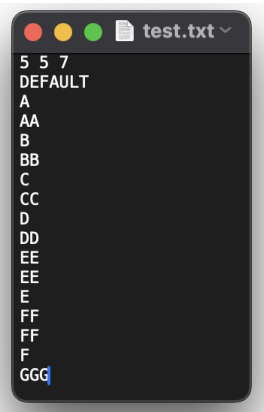
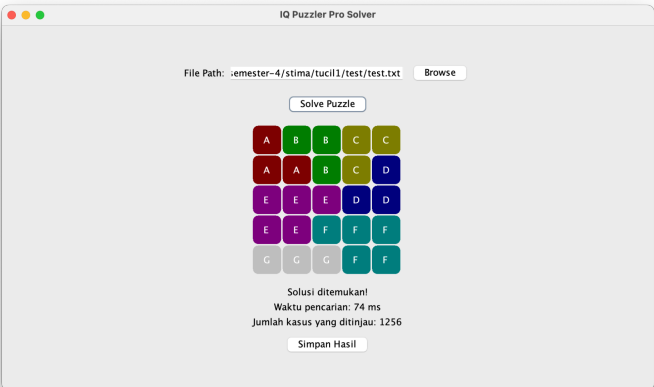
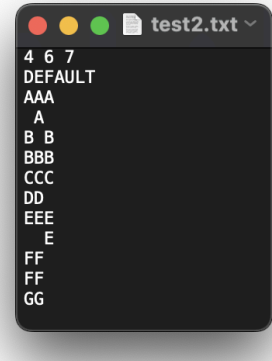
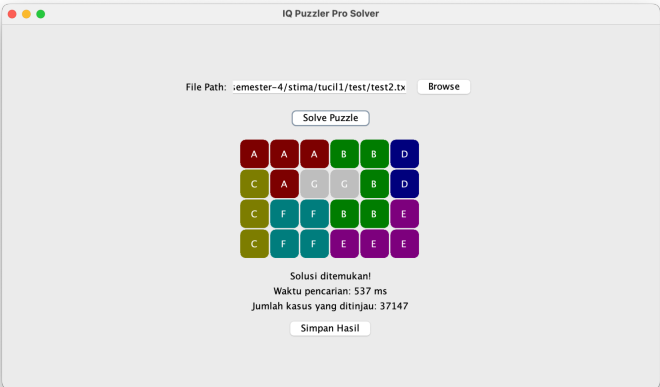
- Kelas `BoardPanel` meng-override metode `paintComponent(Graphics g)` untuk menggambar papan:
 - Setiap sel diiterasi; posisi dihitung berdasarkan `cellSize` dan jarak (`cellGap`).
 - Warna diambil dari `Main.shapeColor` dan digunakan untuk menggambar sel dengan `fillRoundRect`.
 - Identifier sel dicetak di atas blok menggunakan `drawString` dengan offset untuk pemusatan.

BAB IV

EKSPERIMEN

4.1 Tangkapan Layar Eksperimen

Berikut adalah tangkapan layar input dan output eksperimen.

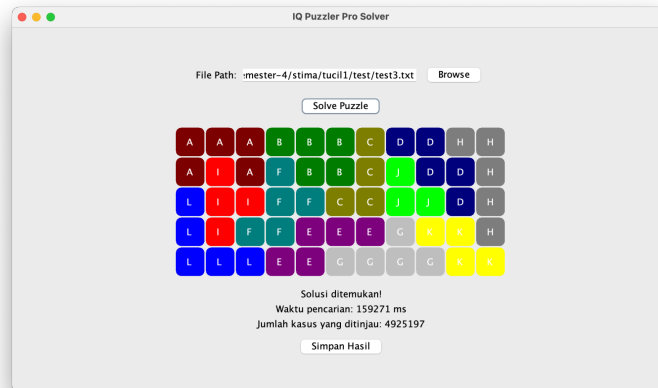
No	Input	Output
1	 <pre> 5 5 7 DEFAULT A AA B BB C CC D DD EE EE E FF FF F GGG </pre>	 <p>File Path: :semester-4/stima/tucil1/test/test.txt <input type="button" value="Browse"/></p> <p><input type="button" value="Solve Puzzle"/></p> <p>Solusi ditemukan! Waktu pencarian: 74 ms Jumlah kasus yang ditinjau: 1256</p> <p><input type="button" value="Simpan Hasil"/></p>
2	 <pre> 4 6 7 AAA A B B BBB CCC DD EEE E FF FF GG </pre>	 <p>File Path: :semester-4/stima/tucil1/test/test2.tx <input type="button" value="Browse"/></p> <p><input type="button" value="Solve Puzzle"/></p> <p>Solusi ditemukan! Waktu pencarian: 537 ms Jumlah kasus yang ditinjau: 37147</p> <p><input type="button" value="Simpan Hasil"/></p>

3

```

test3.txt
5 11 12
DEFAULT
AAA
A A
BBB
BB
CC
C C
C
DD
DD
D
E
EE
E
E
F
FF
FF
G
GGGG
H
HHHH
III
I
J
JJ
K
KK
K
L
L
LLL

```

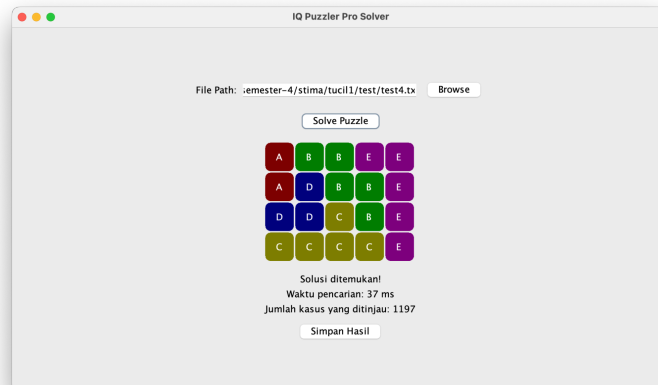


4

```

test4.txt
4 5 5
DEFAULT
AA
BB
BB
B
C
CC
C
C
DD
D
EEEE
E

```



5

```

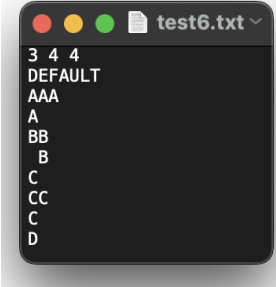
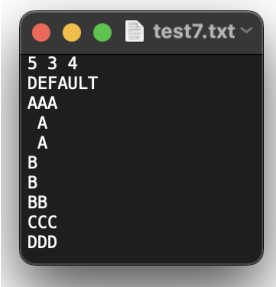
test5.txt
5 7 7
DEFAULT
AAA
AAA
BBBB
B
CC
C
DD
DDD
D
EE
EEEE
FFF
F
G
GGGG

```

Masukkan path dari input file: /Users/Agung/Documents/wiw
ekaputera/ITB/semester-4/stima/tucil1/test/test5.txt
Solusi ditemukan:

AAABBBB
DDAAABC
DDDEEC
FDEEEG
FFFGGG

Waktu pencarian: 103 ms
Jumlah kasus yang ditinjau: 3037
Simpan hasil? (Y/N): y
Masukkan file path output hasil: result5.txt
Hasil disimpan ke result5.txt

6		<p>Masukkan path dari input file: /Users/Agung/Documents/wiw ekaputera/ITB/semester-4/stima/tucil1/test/test6.txt Solusi ditemukan:</p> <p>AAAB ACBB CCCD</p> <p>Waktu pencarian: 0 ms Jumlah kasus yang ditinjau: 4 Simpan hasil? (Y/N): y Masukkan file path output hasil: result6.txt Hasil disimpan ke result6.txt</p>
7		<p>Masukkan path dari input file: /Users/Agung/Documents/wiw ekaputera/ITB/semester-4/stima/tucil1/test/test7.txt Solusi ditemukan:</p> <p>AAA BAC BAC BBC DDD</p> <p>Waktu pencarian: 0 ms Jumlah kasus yang ditinjau: 4 Simpan hasil? (Y/N): y Masukkan file path output hasil: result7.txt Hasil disimpan ke result7.txt</p>

LAMPIRAN

Pranala ke repository yang berisi kode program:

https://github.com/wiwekaputera/Tucil1_13523160

No	Poin	Ya	Tidak
1.	Program berhasil dikompilasi tanpa keasalahan	✓	
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	✓	