IF2211 Strategi Algoritma

Laporan Tugas Kecil 1

oleh

Muhamad Nazih Najmudin   13523144

A. Deskripsi Tugas

IQ Puzzler Pro adalah permainan papan yang diproduksi oleh perusahaan Smart Games. Tujuan dari permainan ini adalah pemain harus dapat mengisi seluruh papan dengan *piece* (*puzzle block*) yang telah tersedia. Komponen penting dari permainan IQ Puzzler Pro terdiri dari:

a. *Board* (Papan) – *Board* merupakan komponen utama yang menjadi tujuan permainan dimana pemain harus mampu mengisi seluruh area papan menggunakan blok-blok yang telah disediakan.

b. *Block/Piece* – Blok adalah komponen yang digunakan pemain untuk mengisi papan kosong hingga terisi penuh. Setiap blok memiliki bentuk yang unik dan semua blok harus digunakan untuk menyelesaikan *puzzle*.

Tugas ini mengharuskan untuk menemukan satu solusi dari permainan IQ Puzzler Pro dengan menggunakan algoritma *brute force*, atau menampilkan bahwa solusi tidak ditemukan jika tidak ada solusi yang mungkin dari *puzzle*.

B. Algoritma *Brute Force* yang digunakan

Pada tugas ini, algoritma yang digunakan untuk menyelesaikan *puzzle* adalah algoritma *brute force* murni tanpa unsur heuristik. Algoritma ini mengharuskan program untuk mengecek semua kemungkinan jawaban dari *puzzle*. Langkah-langkah program dalam mencari jawaban adalah sebagai berikut:

1. Ekstraksi file txt yang diinput oleh user menghasilkan sebuah *board* berukuran *N* x *M* dan *list of block* yang terdiri dari *P* buah *block*. Petak-petak pada *board* yang berjumlah *N* x *M* kedepannya akan disebut sebagai *slot*.
2. Setiap *block* dimanipulasi dengan diputar dan dicerminkan sehingga menghasilkan 8 buah *shape* untuk setiap *block*.

3. Pada program ini, satu siklus algoritma *brute force* merupakan percobaan pencocokkan sebuah *block* yang terdiri dari 8 *shape* untuk setiap *slot* pada *board*. Iterasi dilakukan untuk setiap *slot*, yang pada satu *slot*-nya dicobakan untuk 8 *shape*.
4. Apabila suatu *block* dengan *shape* tertentu berhasil ditempatkan pada suatu *slot*, siklus akan dilanjut oleh *block* selanjutnya secara rekursif.
5. Proses rekursif akan berakhir pada *block* terakhir di *slot* terakhir pada *shape* terakhir, atau *block* terakhir pada suatu *shape* tertentu di *slot* tertentu apabila berhasil ditempatkan. Keberhasilan dinyatakan apabila proses rekursif berakhir dan seluruh *slot* pada *board* tertutup oleh *block*.

Dengan menerapkan algoritma *brute force*, program akan selalu mendapatkan jawaban atas persoalan apapun selama waktu eksekusi dan kebutuhan *resource* komputasinya terpenuhi.

C. Sumber Kode

a. Pranala Repositori

Repository GitHub tugas ini dapat dikunjungi pada laman berikut: https://github.com/nazihstei/Tucil1_13523144/releases/latest

b. Struktur Repositori

```
└───Tucil1_13523144
    │   .classpath
    │   .gitattributes
    │   .gitignore
    │   README.md
    │   run.bat
    │   run.sh
    │
    ├───bin
    ├───doc
    ├───src
    │   │   CLI.java
    │   │   GUI.java
    │   │   Main.java
    │   │
    │   ├───model
    │   │       Block.java
    │   │       Board.java
    │   │
    │   ├───solver
    │   │       Solve.java
    │   │
    │   └───utility
    │           CustomString.java
    │           FileHandler.java
    │           Progress.java
    │
    └───test
            case1.txt
            case2.txt
            case3.txt
            case4.txt
            case5.txt
            case6.txt
            case7.txt
```

c. Implementasi Program

i. CLI.java

```java
import model.*;
import solver.*;
import utility.*;

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

import java.time.Instant;
import java.io.FileNotFoundException;
import java.io.PrintStream;
import java.time.Duration;

public class CLI {

    public static void runCLI(boolean heuristic) {

        // IO utility
        Scanner scanner = new Scanner(System.in);

        // Interface
        CustomString.clearTerminal();
        System.out.println("\n[=======[ Welcome To IQ Puzzler ]=======]");
        System.out.println("");
        System.out.println("[Program] Masukkan file txt");
        System.out.print("          >> "); String filepath = scanner.nextLine();
        System.out.println("[Program] Ingin menyimpan solusi Puzzle? (y/n)");
        System.out.print("          >> "); char saveSolution = scanner.nextLine().charAt(0);

        // Load from file
        List<Object> inputData = FileHandler.inputFile(filepath);
        int boardRow = (int) inputData.get(0);
        int boardCol = (int) inputData.get(1);
        int numBlock = (int) inputData.get(2);
        Board.typeBoard boardType = (Board.typeBoard) inputData.get(3);
        List<List<String>> blockStringList = (List<List<String>>) inputData.getLast();
        long totalCombination = Progress.countTotalCombination((boardRow*boardCol), 8, 0,
numBlock-1);

        // Generate Board
        Board board = new Board(boardRow, boardCol, boardType);

        // Generate Blocks
        List<List<Block>> blockList = new ArrayList<>();
        for (List<String> blockString : blockStringList) {
            char blockID = FileHandler.getCharOfBlock(blockString.get(0));
            Block blok = new Block(blockID, blockString);
            blockList.add(blok.getAllShape());
        }

        // Interface
        System.out.println("");
        System.out.println("[Program] Puzzle successfully generated.");
        System.out.println("[Program] Starting to find solution ...");

        // Generate Solution
        Solve solver = new Solve(heuristic);
        Instant timeStart = Instant.now();
        boolean puzzleSolved = solver.checkDefaultSolve(board, blockList, 0, numBlock-1, true,
totalCombination);
        Instant timeEnd = Instant.now();
        long timeExecuted = Duration.between(timeStart, timeEnd).toMillis();
        CustomString.clearTerminal();

        if (saveSolution=='y' || saveSolution=='Y') {
            System.out.println("");
            System.out.println("[Program] Masukkan nama file penyimpanan");
            System.out.print("          >> ");
            String saveFile = scanner.nextLine();
            try {
                FileHandler.OutputRedirector.setOutputToFile(saveFile);
            } catch (FileNotFoundException e) {
                System.out.println("File tidak ditemukan");
                e.getStackTrace();
            }
        }

        // Output
        if (puzzleSolved) {
            if (timeExecuted < 100000) {
                System.out.println("");
                System.out.printf("   ///////\n");
                System.out.printf("  [[^ w ^]]   I can solve this in %d ms\n", timeExecuted);
                System.out.printf("  /ll---ll\\   with %d attempts of combination.\n",
solver.tryCount);
                System.out.printf("  *il###li*\n");
                System.out.println("[======================================================]");
                System.out.println("[=======[       Your Puzzle is Too Easy!       ]=======]");
                System.out.println("[======================================================]");
            } else if (timeExecuted < 10000000) {
                System.out.println("");
                System.out.printf("   ///////\n");
                System.out.printf("  [[. _ .]]   I can solve this in %d ms\n", timeExecuted);
                System.out.printf("  /ll---ll\\   with %d attempts of combination.\n",
solver.tryCount);
                System.out.printf("  *il###li*\n");
                System.out.println("[======================================================]");
                System.out.println("[=======[   Your Puzzle is Quite Difficult...  ]=======]");
                System.out.println("[======================================================]");
            } else {
                System.out.println("");
                System.out.printf("   ///////\n");
                System.out.printf("  [[> x <]]   I can solve this in %d ms\n", timeExecuted);
                System.out.printf("  /ll---ll\\   with %d attempts of combination.\n",
solver.tryCount);
                System.out.printf("  *il###li*\n");
                System.out.println("[======================================================]");
                System.out.println("[=======[    Your Puzzle is EXTREMELY HARD!!!  ]=======]");
                System.out.println("[======================================================]");
            }
            System.out.println("[=======[ This is the Solution of your Puzzle: ]=======]");
            System.out.println("[======================================================]");
            System.out.println("");

            // Solution print here
            board.printBoard();

        } else {
            System.out.println("");
            System.out.printf("   ///////\n");
            System.out.printf("  [[x , x]]   I have try this in %d ms\n", timeExecuted);
            System.out.printf("  /ll---ll\\   with %d attempts of combination.\n",
solver.tryCount);
            System.out.printf("  *il###li*\n");
            System.out.println("[======================================================]");
            System.out.println("[=======[ Have you ever made a Puzzle before?? ]=======]");
            System.out.println("[=======[         Your puzzle is sucks!        ]=======]");
            System.out.println("[=======[      Even Albert Einstein can't      ]=======]");
            System.out.println("[=======[          solve this puzzle.          ]=======]");
            System.out.println("[======================================================]");
            System.out.println("");
        }

        // End Program
        FileHandler.OutputRedirector.setOutputToOriginal();
        if (saveSolution!='Y' || saveSolution!='y') {
            System.exit(0);
        }
        scanner.nextLine();
    }

}
```

ii. GUI.java

```
import javax.swing.*;
import javax.swing.filechooser.FileNameExtensionFilter;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.time.Duration;
import java.time.Instant;

import javax.imageio.ImageIO;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import java.util.List;

import model.*;
import solver.*;
import utility.*;

public class GUI {
    private JFrame frame;
    private JPanel boardPanel;
    private JTable infoTable;
    private JButton inputFileButton, runButton;
    private long executionTime;
    private long trialCount;
    private String additionalText;
    private int boardRow, boardCol, numBlock;
    private Board.typeBoard boardType;
    private File selectedFile;
    private Board board;
    private List<List<Block>> blockList;
    private boolean heuristic;
    public GUI(boolean heuristic) {
        this.heuristic = heuristic;
        frame = new JFrame("IQ Puzzler Pro");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(900, 600);
        frame.setLayout(new BorderLayout());

        // Panel kiri (Tombol & Tabel)
        JPanel leftPanel = new JPanel();
        leftPanel.setLayout(new GridLayout(1, 2, 30, 30));
        inputFileButton = new JButton("Input File");
        runButton = new JButton("Run");
        runButton.setEnabled(false); // Run baru aktif setelah file dipilih
        leftPanel.add(inputFileButton);
        leftPanel.add(runButton);

        // Panel kanan (Board)
        JPanel rightPanel = new JPanel(new BorderLayout());
        boardPanel = new JPanel();
        infoTable = new JTable(3, 2);
        infoTable.getColumnModel().getColumn(0).setHeaderValue("Keterangan");
        infoTable.getColumnModel().getColumn(1).setHeaderValue("Nilai");
        infoTable.getTableHeader().repaint(); // Refresh header
        infoTable.setValueAt("Waktu Eksekusi:", 0, 0);
        infoTable.setValueAt("Jumlah Percobaan:", 1, 0);
        infoTable.setValueAt("Info Tambahan:", 2, 0);

        rightPanel.add(boardPanel, BorderLayout.CENTER);
        rightPanel.add(new JScrollPane(infoTable), BorderLayout.WEST);

        // Menambahkan ke frame
        frame.add(rightPanel, BorderLayout.CENTER);
        frame.add(leftPanel, BorderLayout.NORTH);

        // Event Listener
        inputFileButton.addActionListener(e -> loadFile());
        runButton.addActionListener(e -> onRun());

        frame.setVisible(true);
    }

    private void loadFile() {
        JFileChooser fileChooser = new JFileChooser();
        fileChooser.setFileFilter(new FileNameExtensionFilter("Text Files", "txt"));
        int option = fileChooser.showOpenDialog(frame);
        if (option == JFileChooser.APPROVE_OPTION) {
            selectedFile = fileChooser.getSelectedFile();
            JOptionPane.showMessageDialog(frame, "File Loaded: " + selectedFile.getName());

            // Load from file
            List<Object> inputData = FileHandler.inputFile(selectedFile);

            onFileLoaded(inputData);
        }
    }

    private void onFileLoaded(List<Object> data) {
        this.boardRow = (int) data.get(0);
        this.boardCol = (int) data.get(1);
        this.numBlock = (int) data.get(2);
        this.boardType = (Board.typeBoard) data.get(3);
        List<List<String>> blockStringList = (List<List<String>>) data.getLast();

        // Generate Board
        this.board = new Board(boardRow, boardCol, boardType);

        // Generate Blocks
        this.blockList = new ArrayList<>();
        for (List<String> blockString : blockStringList) {
            char blockID = FileHandler.getCharOfBlock(blockString.get(0));
            Block blok = new Block(blockID, blockString);
            this.blockList.add(blok.getAllShape());
        }

        // Mengaktifkan tombol run setelah file dipilih
        runButton.setEnabled(true);
    }

    private void onRun() {
        Solve solver = new Solve(this.heuristic);
        long totalCombination = Progress.countTotalCombination((this.boardRow*this.boardCol), 8,
0, this.numBlock-1);
        Instant timeStart = Instant.now();
        boolean puzzleSolved = solver.checkDefaultSolve(this.board, this.blockList, 0,
this.numBlock-1, true, totalCombination);
        Instant timeEnd = Instant.now();
        long timeExecuted = Duration.between(timeStart, timeEnd).toMillis();

        String resultText;
        if (puzzleSolved) {
            if (timeExecuted < 100000) {
                resultText = "Your Puzzle is Too Easy!";
            } else if (timeExecuted < 10000000) {
                resultText = "Your Puzzle is Quite Difficult...";
            } else {
                resultText = "Your Puzzle is EXTREMELY HARD!!!";
            }
        } else {
            resultText = "Your Puzzle is SUCKS! No one can solve this.";
        }

        setExecutionResult(timeExecuted, solver.tryCount, resultText);

        int saveOption = JOptionPane.showConfirmDialog(frame, "Simpan hasil sebagai gambar?",
"Simpan", JOptionPane.YES_NO_OPTION);
        if (saveOption == JOptionPane.YES_OPTION) {
            saveImage();
        }
        runButton.setEnabled(false);
    }

    public void setExecutionResult(long timeExecuted, long trialCount, String customizeString) {
        this.executionTime = timeExecuted;
        this.trialCount = trialCount;
        this.additionalText = customizeString;

        updateBoardPanel();
        updateTable();
    }

    private void updateBoardPanel() {
        boardPanel.removeAll();
        boardPanel.setLayout(new GridLayout(this.boardRow, this.boardCol));

        Map<Character, Color> colorMap = new HashMap<>();
        for (char[] row : this.board.map) {
            for (char c : row) {
                if (c != ' ' && !colorMap.containsKey(c)) {
                    colorMap.put(c, new Color((int) (Math.random() * 0xFFFFFF)));
                }
            }
        }

        for (char[] row : this.board.map) {
            for (char c : row) {
                JLabel label = new JLabel(String.valueOf(c), SwingConstants.CENTER);
                label.setOpaque(true);
                label.setBorder(BorderFactory.createLineBorder(Color.BLACK));
                label.setBackground(c == ' ' ? Color.WHITE : colorMap.get(c));
                boardPanel.add(label);
            }
        }

        boardPanel.revalidate();
        boardPanel.repaint();
    }

    private void updateTable() {
        infoTable.setValueAt(executionTime + " ms", 0, 1);
        infoTable.setValueAt(trialCount, 1, 1);
        infoTable.setValueAt(additionalText, 2, 1);
    }

    private void saveImage() {
        BufferedImage image = new BufferedImage(boardPanel.getWidth(), boardPanel.getHeight(),
BufferedImage.TYPE_INT_RGB);
        Graphics2D g2d = image.createGraphics();
        boardPanel.paint(g2d);
        g2d.dispose();

        JFileChooser fileChooser = new JFileChooser();
        fileChooser.setFileFilter(new FileNameExtensionFilter("PNG Images", "png"));
        int option = fileChooser.showSaveDialog(frame);

        if (option == JFileChooser.APPROVE_OPTION) {
            File file = fileChooser.getSelectedFile();
            try {
                ImageIO.write(image, "png", new File(file.getAbsolutePath() + ".png"));
                JOptionPane.showMessageDialog(frame, "Gambar disimpan: " + file.getAbsolutePath()
+ ".png");
            } catch (IOException e) {
                JOptionPane.showMessageDialog(frame, "Gagal menyimpan gambar!", "Error",
JOptionPane.ERROR_MESSAGE);
            }
        }
    }

    public static void runGUI(boolean heuristic) {
        SwingUtilities.invokeLater(() -> new GUI(heuristic));
    }
}
```

iii. Main.java

```
import java.util.Scanner;
import utility.*;

public class Main {

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

        boolean heuristic = false;
        boolean start = true;
        while (start) {
            CustomString.clearTerminal();
            System.out.println("[=======[ Welcome To IQ Puzzler ]=======]");
            System.out.println("");
            System.out.println("[Program] Enable heuristic for unsolvable puzzle? (y/n)");
            System.out.print("          >> "); char enableHeuristic =
scanner.nextLine().charAt(0);
            switch (enableHeuristic) {
                case 'Y', 'y' -> {heuristic=true; start=false;}
                case 'N', 'n' -> {heuristic=false; start=false;}
                default       -> {;}
            }
        }
        boolean exit = false;
        while (!exit) {
            CustomString.clearTerminal();
            System.out.println("[=======[ Welcome To IQ Puzzler ]=======]");
            System.out.println("");
            System.out.println("[Program] Silahkan pilih menu: ");
            System.out.println("          A. run on CLI");
            System.out.println("          B. run on GUI");
            System.out.println("          C. exit program");
            System.out.println("");
            System.out.print(">> "); char selectedOption = scanner.nextLine().charAt(0);
            switch (selectedOption) {
                case 'A', 'a' -> {CLI.runCLI(heuristic);}
                case 'B', 'b' -> {GUI.runGUI(heuristic);}
                case 'C', 'c' -> {exit = true;}
                default       -> {;}
            }
        }
        CustomString.clearTerminal();
        scanner.close();
    }
}
```

iv. Package model

1. Block.java

```
package model;

import java.util.List;
import java.util.Set;
import java.util.ArrayList;
import java.util.HashSet;

public class Block {

    // Attrubutes
    public char id;
    public boolean[][] shape;
    public int tileCount;

    // Create Block
    public Block(char id, boolean[][] shape) {
        this.id = id;
        this.shape = shape;
        this.tileCount = countActiveTiles(shape);
    }
    public Block(char id, List<String> stringShape) {
        this.id = id;
        this.shape = createShape(id, stringShape);
        this.tileCount = countActiveTiles(this.shape);
    }
    public static boolean[][] createShape(char id, List<String> stringBlock) {
        int row = stringBlock.size();
        int col = 0;
        for (String line : stringBlock) {
            col = Math.max(col, line.length());
        }
        boolean[][] matrix = new boolean[row][col];
        for (int i=0; i<row; i++) {
            for (int j=0; j<col; j++) {
                matrix[i][j] = false;
            }
        }
        for (int i=0; i<stringBlock.size(); i++) {
            for (int j=0; j<stringBlock.get(i).length(); j++) {
                matrix[i][j] = (id == stringBlock.get(i).charAt(j));
            }
        }
        return matrix;
    }

    public List<Block> getAllShape() {
        Set<Block> shapeSet = new HashSet<>();
        Block normalBlock = new Block(this.id, this.shape);
        Block flippedBlock = this.flip();

        for (int i=0; i<4; i++) {
            shapeSet.add(new Block(normalBlock.id, normalBlock.shape));
            shapeSet.add(new Block(flippedBlock.id, flippedBlock.shape));
            normalBlock = normalBlock.rotate();
            flippedBlock = flippedBlock.rotate();
        }

        return new ArrayList<>(shapeSet);
    }

    // Rotation 90 deg counter-clockwise
    public Block rotate() {
        int row = this.shape.length;
        int col = this.shape[0].length;
        boolean[][] newShape = new boolean[col][row];

        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                newShape[j][row - i - 1] = this.shape[i][j];
            }
        }
        return new Block(this.id, newShape);
    }

    // Flip horizontally
    public Block flip() {
        int row = this.shape.length;
        int col = this.shape[0].length;
        boolean[][] newShape = new boolean[row][col];

        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                newShape[i][col - j - 1] = this.shape[i][j];
            }
        }
        return new Block(this.id, newShape);
    }

    // Count how many Tiles
    public static int countActiveTiles(boolean[][] shape) {
        int count = 0;
        for (int i=0; i<shape.length; i++) {
            for (int j=0; j<shape[0].length; j++) {
                if (shape[i][j]) {
                    count++;
                }
            }
        }
        return count;
    }

    // Print Block
    public void printBlock() {
        for (int i=0; i<this.shape.length; i++) {
            for (int j=0; j<this.shape[0].length; j++) {
                if (this.shape[i][j]) {
                    System.out.print(this.id);
                } else {
                    System.out.print(" ");
                }
            }
            System.out.print("\n");
        }
        System.out.print("\n");
    }
}
```

2. Board.java

```
package model;

import java.util.ArrayList;
import java.util.List;

import utility.CustomString;

public class Board {

    // Attributes
    public int row;
    public int col;
    public typeBoard tipe;
    public char[][] map;

    public static enum typeBoard {
        DEFAULT,
        CUSTOM,
        PYRAMID
    }

    // Create Board
    public Board(int row, int col, typeBoard tipe) {
        this.row = row;
        this.col = col;
        this.tipe = tipe;
        this.map = createMap(row, col);
    }
    public static char[][] createMap(int row, int col) {
        char[][] map = new char[row][col];
        for (int i=0; i<row; i++) {
            for (int j=0; j<col; j++) {
                map[i][j] = ' ';
            }
        }
        return map;
    }

    // Put Block
    public boolean putBlock(Block blok, int idRow, int idCol) {
        int row = blok.shape.length;
        int col = blok.shape[0].length;
        for (int i=0; i<row; i++) {
            for (int j=0; j<col; j++) {
                if (((i+idRow) >= this.row) || ((j+idCol) >= this.col)) {
                    return false;
                }
                if (blok.shape[i][j] && (this.map[i+idRow][j+idCol] != ' ')) {
                    return false;
                }
            }
        }
        for (int i=0; i<row; i++) {
            for (int j=0; j<col; j++) {
                if (blok.shape[i][j]) {
                    this.map[i+idRow][j+idCol] = blok.id;
                }
            }
        }
        return true;
    }

    // Pull out Block
    public boolean pullOutBlock(Block blok, int idRow, int idCol) {
        int row = blok.shape.length;
        int col = blok.shape[0].length;
        for (int i=0; i<row; i++) {
            for (int j=0; j<col; j++) {
                if (blok.shape[i][j]) {
                    this.map[i+idRow][j+idCol] = ' ';
                }
            }
        }
        return true;
    }

    // Check Board
    public boolean isComplete() {
        for (char[] rowMap : this.map) {
            for (char cc : rowMap) {
                if (cc == ' ') {
                    return false;
                }
            }
        }
        return true;
    }

    // Print Board
    public void printBoard() {
        int idxForColor = -1;
        List<Character> charMet = new ArrayList<>();
        for (int i=0; i<this.row; i++) {
            for (int j=0; j<this.col; j++) {
                // No Color
                if (this.map[i][j] == ' ') {
                    System.out.print(". ");
                } else {
                    System.out.print(this.map[i][j] + " ");
                }
            }
            System.out.print("\n");
        }
        System.out.print("\n");
    }
}
```

v. Solve.java

```
package solver;

import model.*;
import utility.Progress;

import java.util.List;

public class Solve {
    public boolean heuristic;
    public long tryCount;
    private long progress;

    // Constructor
    public Solve(boolean heuristic) {
        this.heuristic = heuristic;
        this.tryCount = 0;
        this.progress = 0;
    }

    // DEFAULT solver
    public boolean checkDefaultSolve(Board board, List<List<Block>> blockList, int
idx, int stopIdx, boolean debugProgress, long totalComb) {

        if (this.heuristic) {
            int total = 0;
            for (List<Block> block : blockList) {
                total += block.get(0).tileCount;
            }
            if (!(board.row*board.col == total)) {
                return false;
            }
        }

        boolean isSuccess = false;
        List<Block> blok = blockList.get(idx);

        for (int i=0; i<board.row; i++) {
            for (int j=0; j<board.col; j++) {

                for (Block shape : blok) {

                    if (idx == stopIdx) {
                        this.tryCount++;
                    }

                    boolean isFit = board.putBlock(shape, i, j);
                    if (isFit) {
                        this.progress++;
                        if (idx == stopIdx) {
                            boolean isFull = board.isComplete();
                            if (isFull) {
                                return true;
                            } else {
                                board.pullOutBlock(shape, i, j);
                                return false;
                            }
                        } else {
                            isSuccess = checkDefaultSolve(board, blockList, idx+1,
stopIdx, debugProgress, totalComb);
                        }

                        if (isSuccess) {
                            return isSuccess;
                        } else {
                            board.pullOutBlock(shape, i, j);
                        }

                    } else {
                        this.progress +=
Progress.countTotalCombination((board.row*board.col), 8, idx, stopIdx-1);
                    }

                    if (debugProgress) {
                        Progress.displayProgress(this.progress, totalComb);
                    }

                }
            }
        }
        return isSuccess;
    }
}
```

vi. Package utility

1. CustomString.java

```
package utility;

import java.io.IOException;

public class CustomString {
    public static final String RESET = "\\u001B[0m"; // Reset ke default

    // Mapping warna
    public static final String[] COLORS = {
        "\\u001B[38;5;127m", "\\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[38;5;208m", "\\u001B[38;5;200m", "\\u001B[38;5;39m",
        "\\u001B[38;5;120m", "\\u001B[38;5;45m", "\\u001B[38;5;183m",
"\\u001B[38;5;220m",
        "\\u001B[38;5;242m", "\\u001B[38;5;130m", "\\u001B[38;5;30m", "\\u001B[30m"
    };
    public static final String[] COLOR_NAMES = {
        "Dark Magenta", "Red", "Green", "Yellow", "Blue", "Purple", "Cyan", "White",
        "Bright Black", "Bright Red", "Bright Green", "Bright Yellow", "Bright Blue",
        "Bright Purple", "Bright Cyan", "Bright White", "Orange", "Pink", "Light Blue",
        "Light Green", "Turquoise", "Lavender", "Gold", "Gray", "Brown", "Teal",
"Black"
    };

    public static void colorPrint(String text, String color) {
        int index = -1;
        // Cari indeks warna yang sesuai
        for (int i = 0; i < COLOR_NAMES.length; i++) {
            if (COLOR_NAMES[i].equalsIgnoreCase(color)) {
                index = i;
                break;
            }
        }
        // Jika warna tidak ditemukan, gunakan default (putih)
        if (index == -1) {
            System.out.print(text);
        } else {
            System.out.print(COLORS[index] + text + RESET);
        }
    }
    public static void colorPrint(String text, int colorIndex) {
        // Jika warna tidak ditemukan, gunakan default (putih)
        if (0<colorIndex || colorIndex>25) {
            System.out.print(text);
        } else {
            System.out.print(COLORS[colorIndex] + text + RESET);
        }
    }

    // Clear terminal
    public static void clearTerminal() {
        try {
            if (System.getProperty("os.name").contains("Windows")) {
                new ProcessBuilder("cmd", "/c", "cls").inheritIO().start().waitFor();
            } else {
                new ProcessBuilder("clear").inheritIO().start().waitFor();
            }
        } catch (IOException | InterruptedException e) {
            e.printStackTrace();
        }
    }

}
```

2. FileHandler.java

```
package utility;

import model.*;

import java.io.*;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;

public class FileHandler {

    // Input File
    public static List<Object> inputFile(String filepath) {
        List<Object> output = new ArrayList<>();

        try {
            int row=0, col=0, blockCount=0;
            Board.typeBoard tipe=Board.typeBoard.DEFAULT;
            List<List<String>> blockList = new ArrayList<>();

            char prevChar = ' ';
            int lineCount = 1;
            int blockListIndex = -1;

            File file = new File(filepath);
            Scanner stream = new Scanner(file);
            while (stream.hasNextLine()) {
                String line = stream.nextLine();
                switch (lineCount) {
                    case 1 -> {
                        List<String> line1 = Arrays.asList(line.split(" "));
                        row = Integer.parseInt(line1.get(0));
                        col = Integer.parseInt(line1.get(1));
                        blockCount = Integer.parseInt(line1.get(2));
                    }
                    case 2 -> {
                        tipe = Board.typeBoard.valueOf(line);
                    }
                    default -> {
                        char currChar = getCharOfBlock(line);
                        if (currChar != prevChar) {
                            List<String> newBlock = new ArrayList<>();
newBlock.add(line);
                            blockList.add(newBlock);
                            blockListIndex++;
                            prevChar = currChar;
                        } else {
                            blockList.get(blockListIndex).add(line);
                        }
                    }
                }
                lineCount++;
            }
            stream.close();
            output.add(row); output.add(col); output.add(blockCount);
            output.add(tipe);
            output.add(blockList);

        } catch (FileNotFoundException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
        }

        return output;
    }
    public static List<Object> inputFile(File file) {
        List<Object> output = new ArrayList<>();

        try {
            int row=0, col=0, blockCount=0;
            Board.typeBoard tipe=Board.typeBoard.DEFAULT;
            List<List<String>> blockList = new ArrayList<>();

            char prevChar = ' ';
            int lineCount = 1;
            int blockListIndex = -1;

            Scanner stream = new Scanner(file);
            while (stream.hasNextLine()) {
                String line = stream.nextLine();
                switch (lineCount) {
                    case 1 -> {
                        List<String> line1 = Arrays.asList(line.split(" "));
                        row = Integer.parseInt(line1.get(0));
                        col = Integer.parseInt(line1.get(1));
                        blockCount = Integer.parseInt(line1.get(2));
                    }
                    case 2 -> {
                        tipe = Board.typeBoard.valueOf(line);
                    }
                    default -> {
                        char currChar = getCharOfBlock(line);
                        if (currChar != prevChar) {
                            List<String> newBlock = new ArrayList<>();
newBlock.add(line);
                            blockList.add(newBlock);
                            blockListIndex++;
                            prevChar = currChar;
                        } else {
                            blockList.get(blockListIndex).add(line);
                        }
                    }
                }
                lineCount++;
            }
            stream.close();
            output.add(row); output.add(col); output.add(blockCount);
            output.add(tipe);
            output.add(blockList);

        } catch (FileNotFoundException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
        }

        return output;
    }
    // Helper untuk mencari char of Block dari stream
    public static char getCharOfBlock(String str) {
        for (char c : str.toCharArray()) {
            if (c != ' ') {
                return c;
            }
        }
        return ' ';
    }

    // Output File
    public class OutputRedirector {
        private static PrintStream originalOut = System.out;
        private static PrintStream fileOut;

        public static void setOutputToFile(String filepath) throws
FileNotFoundException {
            // File output stream
            FileOutputStream fos = new FileOutputStream(filepath);
            fileOut = new PrintStream(new TeeOutputStream(originalOut, fos), false);

            // Set System.out ke output baru (konsol + file)
            System.setOut(fileOut);
        }

        public static void setOutputToOriginal() {
            System.setOut(originalOut);
            System.out.flush();
            if (fileOut != null) {
                try {
                    fileOut.flush();
                    fileOut.close();
                } catch (Exception e) {
                    System.err.println("Gagal menutup output file: " + e.getMessage());
                }
                fileOut = null;
            }
        }

        // Helper class untuk duplikasi output
        private static class TeeOutputStream extends OutputStream {
            private final OutputStream out1, out2;

            public TeeOutputStream(OutputStream out1, OutputStream out2) {
                this.out1 = out1;
                this.out2 = out2;
            }
            @Override
            public void write(int b) throws IOException {
                out1.write(b);
                out2.write(b);
            }
            @Override
            public void flush() throws IOException {
                out1.flush();
                out2.flush();
            }
            @Override
            public void close() throws IOException {
                out1.close();
                out2.close();
            }
        }
    }
}
```

3. Progress.java

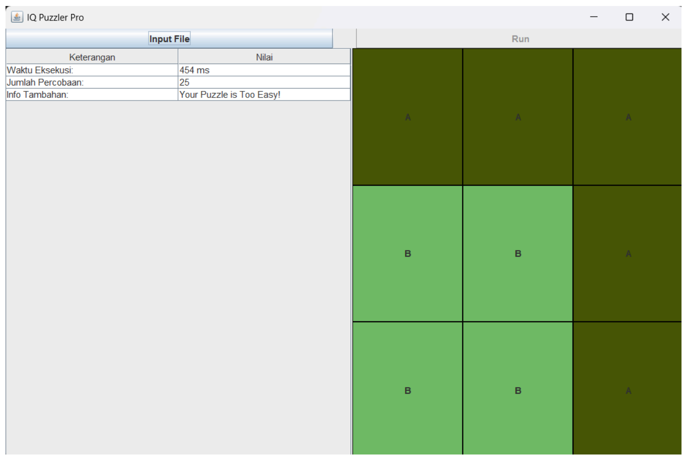
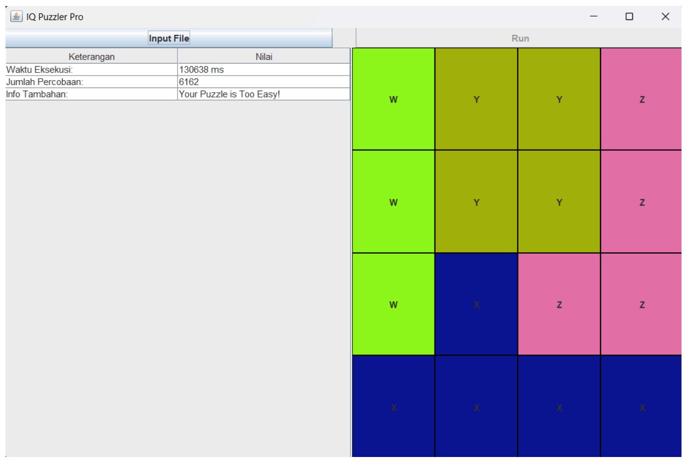
```
package utility;
```

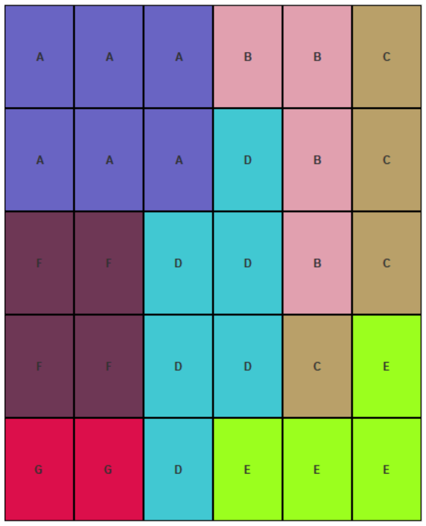
```
public class Progress {
    public static long countTotalCombination(int loop1, int loop2, int startIdx, int
stopIdx) {
        return (long) Math.pow((loop1 * loop2), (stopIdx-startIdx+1));
    }

    public static void displayProgress(long current, long total) {
        int percentage = (int) (((double) current / total) * 100);
        CustomString.clearTerminal();
        System.out.println("");
        System.out.println("[=======[ Welcome To IQ Puzzler ]=======]");
        System.out.println("");
        System.out.printf("[Program] Progress   : %d\n", current);
        System.out.printf("[Program] Total      : %d\n", total);
        System.out.printf("[Program] Percentage : %d %%\n", percentage);
        System.out.println("");
        System.out.println("[======================================]");
    }
}
```

D. Pengujian

Terdapat 7 kasus uji untuk memastikan program dapat berjalan dengan baik. Kasus uji dan hasil pengujian tertera pada tabel berikut:

Tabel 1. Pengujian Program

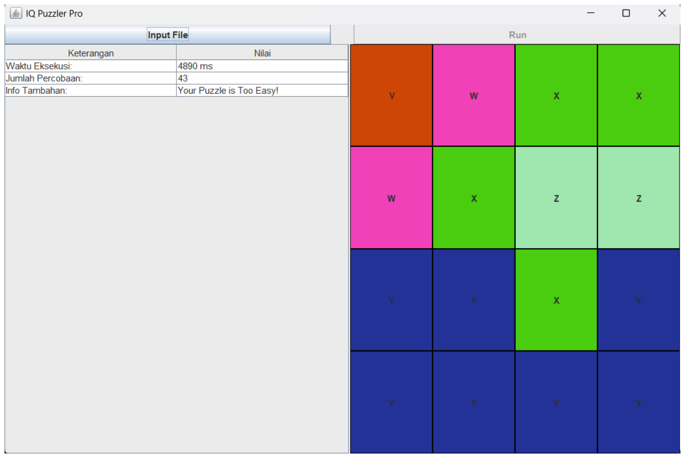
No	Indikator	Kasus	Hasil
1	*Puzzle* dapat diselesaikan (sederhana) Menggunakan GUI	3 3 2 DEFAULT AAA A A BB BB	 IQ Puzzler Pro Input File Run A A A B B A B B A Keterangan: Nilai Waktu Eksekusi: 454 ms Jumlah Percobaan: 25 Info Tambahan: Your Puzzle is Too Easy!
2	*Puzzle* dapat diselesaikan (sedang) Menggunakan GUI	4 4 4 DEFAULT WWW XXXX X YY YY Z ZZZ	 IQ Puzzler Pro Input File Run W Y Y Z W Y Y Z W X Z Z X X X X Keterangan: Nilai Waktu Eksekusi: 130638 ms Jumlah Percobaan: 6162 Info Tambahan: Your Puzzle is Too Easy!

3	*Puzzle* dapat diselesaikan (kompleks)
	
	Menggunakan GUI
	
	Disimpan dalam bentuk gambar PNG
		`5 6 7`
		`DEFAULT`
		`AAA`
		`AAA`
		`  B`
		`BBB`
		`    C`
		`CCC`
		`DDD`
		` DDD`
		`EEE`
		`E`
		`FF`
		`FF`
		`GG`
			 A A A B B C
			A A A D B C
			F F D D B C
			F F D D C E
			G G D E E E
			`ini adalah hasil gambar yang disimpan.`

4	Jumlah petak pada total *block* < jumlah *slot* pada *board*
	
	Menggunakan CLI
		`4 4 2`
		`DEFAULT`
		`XX`
		`X`
		`YY`
			`  ///////`
			` [[x , x]]   I have try this in 9261 ms`
			` /ll---ll\   with 192 attempts of combination.`
			` *il###li*`
			`[=======================================================]`
			`[=======[ Have you ever made a Puzzle before?? ]=======]`
			`[=======[         Your puzzle is sucks!         ]=======]`
			`[=======[      Even Albert Einstein can't       ]=======]`
			`[=======[          solve this puzzle.           ]=======]`
			`[=======================================================]`

5	Jumlah petak pada total *block* > jumlah *slot* pada *board*
	
	Menggunakan CLI
		`3 3 3`
		`DEFAULT`
		`AAA`
		`AAA`
		`BB`
		`BB`
		`C`
		`C`
		`C`
			`  ///////`
			` [[x , x]]   I have try this in 82059 ms`
			` /ll---ll\   with 0 attempts of combination.`
			` *il###li*`
			`[=======================================================]`
			`[=======[ Have you ever made a Puzzle before?? ]=======]`
			`[=======[         Your puzzle is sucks!         ]=======]`
			`[=======[      Even Albert Einstein can't       ]=======]`
			`[=======[          solve this puzzle.           ]=======]`
			`[=======================================================]`

6	Jumlah petak pada total *block* = jumlah *slot* pada *board*, tetapi *puzzle* tidak dapat diselesaikan
	
	Menggunakan CLI
		`2 6 3`
		`DEFAULT`
		`XX`
		`XX`
		`YY`
		`YY`
		`ZZZZ`
			`  ///////`
			` [[x , x]]   I have try this in 270400 ms`
			` /ll---ll\   with 4096 attempts of combination.`
			` *il###li*`
			`[=======================================================]`
			`[=======[ Have you ever made a Puzzle before?? ]=======]`
			`[=======[         Your puzzle is sucks!         ]=======]`
			`[=======[      Even Albert Einstein can't       ]=======]`
			`[=======[          solve this puzzle.           ]=======]`
			`[=======================================================]`

7	Eksplorasi bentuk-bentuk *block* yang aneh dan unik
	
	Menggunakan GUI
		`4 4 5`
		`DEFAULT`
		`V`
		`W`
		`  W`
		`    X`
		`X X`
		`  X`
		`Y YY`
		`YYYY`
		`  ZZ`
			 IQ Puzzler Pro
			Input File — Run
			
			Keterangan | Nilai
			Waktu Eksekusi: | 4890 ms
			Jumlah Percobaan: | 43
			Info Tambahan: | Your Puzzle is Too Easy!
			
			V W X X
			W X Z Z
			Y Y X Y
			Y Y Y Y

E. Lampiran

a. Tabel Pemenuhan Spesifikasi

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	😎	