

Laporan Tugas Kecil Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force

**Tugas Kecil 1 - IF2211 Strategi Algoritma Semester II Tahun
2024/2025**



Disusun oleh

Naufarrel Zhafif Abhista - 13523149

Daftar Isi

Daftar Isi.....	1
1. Pendahuluan.....	2
2. Penjelasan Algoritma Brute Force.....	3
2.1. Struktur Program.....	3
2.2. Kelas Board.....	3
2.3. Kelas Block.....	5
2.4. Kelas Solver.....	6
2.5. Kompleksitas Algoritma.....	8
3. Source Code.....	9
3.1. Struktur Proyek.....	9
3.2. Main.java.....	10
3.2.1. Graphical User Interface (GUI).....	10
3.2.2. Memilih Mode.....	10
3.2.3. Meminta Masukan File.....	10
3.2.4. Menampilkan dan Menyimpan Hasil.....	10
3.2.5. Penanganan Kesalahan.....	10
3.3. Kode Lainnya.....	17
4. Uji Coba.....	17
4.1. Tangkapan Layar Uji Coba.....	18
5. Pranala Repository.....	24
6. Lampiran.....	25

1. Pendahuluan

Algoritma **brute force** merupakan salah satu strategi penyelesaian masalah dalam algoritma. Dalam algoritma ini, program didekati secara lempeng/lurus (*straightforward*). Pemecahannya juga dikatakan sederhana, dengan cara yang cukup *obvious*.

Tugas kecil pertama yang diberikan oleh **IF2211 Strategi Algoritma** bertujuan untuk mengimplementasikan dan menganalisis algoritma brute force dalam menyelesaikan suatu permasalahan. Dengan meninjau kasus nyata (yakni sebuah permainan), diharapkan algoritma brute force dapat tergambarkan secara jelas. Laporan ini disusun untuk menjawab persoalan **IQ Puzzler Pro**.

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 (blok puzzle) yang telah tersedia. Komponen penting dari permainan IQ Puzzler Pro terdiri dari:

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



Gambar 1. Contoh Fisik IQ Puzzler Pro

Permainan dimulai dengan papan yang kosong. Pemain dapat meletakkan blok puzzle sedemikian sehingga tidak ada blok yang bertumpang tindih (kecuali dalam kasus 3D). Setiap blok puzzle dapat dirotasikan maupun dicerminkan. Puzzle dinyatakan selesai jika dan hanya jika papan terisi penuh dan seluruh blok puzzle berhasil diletakkan.

2. Penjelasan Algoritma Brute Force

Algoritma Brute Force digunakan dalam tugas ini untuk menyelesaikan permasalahan penyusunan kepingan puzzle pada papan permainan. Algoritma ini bekerja dengan mencoba semua kemungkinan penempatan kepingan pada papan hingga menemukan solusi yang valid.

Pada laporan ini, seperti perintah spesifikasi, yang akan dibahas spesifik adalah algoritma dari brute force. Untuk komponen pelengkap, seperti *export* solusi, validasi input, dan sebagainya, akan dilampirkan dalam *source code* dan juga Pranala Repositori.

2.1. Struktur Program

Pohon dari *source code*

```
src
├── Block.java
├── Board.java
├── Data.java
├── Main.java
├── Handler.java
└── Solver.java
```

- Program secara keseluruhan diselesaikan dijalankan dari kelas GUI.
- Algoritma *bruteforce* berpusat pada Solver.java.
- Bab selanjutnya akan membahas potongan dari Solver.java (dan pelengkapannya, bila diperlukan)

2.2. Kelas Board

- Mengandung tiga *method* penting yang mendukung Solver.java:
 - void removeBlock(Block block, int x, int y)
 - boolean canPlaceBlock(Block block, int x, int y)
 - void placeBlock(Block block, int x, int y)
- void removeBlock(Block block, int x, int y)
Menghapus penempatan block dengan mereplace board yang “diduduki” dengan *whitespace*.
- boolean canPlaceBlock(Block block, int x, int y)
Melakukan pengecekan penempatan blok, dengan x dan y adalah koordinat penempatan blok untuk ujung kiri atas. Bila: keluar dari Board, atau menemukan *whitespace*, mengembalikan nilai false

- void placeBlock(Block block, int x, int y)
Melakukan penempatan blok, dengan x dan y adalah koordinat penempatan blok untuk ujung kiri atas.

Kode dari ketiga *method*

```
public void removeBlock(Block block, int x, int y) {
    char[][] shape = block.getShape();
    for (int i = 0; i < shape.length; i++) {
        for (int j = 0; j < shape[i].length; j++) {
            if (shape[i][j] != ' ') {
                grid[x + i][y + j] = ' ';
            }
        }
    }
}

public boolean canPlaceBlock(Block block, int x, int y) {
    char[][] shape = block.getShape();
    int blockRows = shape.length;
    int blockCols = shape[0].length;

    if (x + blockRows > rows || y + blockCols > cols) return false;

    for (int i = 0; i < blockRows; i++) {
        for (int j = 0; j < blockCols; j++) {
            if (shape[i][j] != ' ' && grid[x + i][y + j] != ' ') {
                return false;
            }
        }
    }
    return true;
}

public void placeBlock(Block block, int x, int y) {
    char[][] shape = block.getShape();
    for (int i = 0; i < shape.length; i++) {
        for (int j = 0; j < shape[i].length; j++) {
            if (shape[i][j] != ' ') {
                grid[x + i][y + j] = shape[i][j];
            }
        }
    }
}
```

```
}
```

2.3. Kelas Block

- Mengandung dua *method* penting yang mendukung Solver.java:
 - Block rotate()
 - Block mirror()
- Block rotate()
Mengubah orientasi Block 90 derajat *searah jarum jam*.
- Block mirror()
Mencerminkan Block.
- Kedua metode ini diperlukan karena akan ada delapan kemungkinan (empat rotasi dikalikan dua pencerminan) penempatan Block dalam Board.

Kode dari kedua *method*

```
public Block rotate() {
    int rows = shape.length;
    int cols = shape[0].length;
    char[][] rotated = new char[cols][rows];

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            rotated[j][rows - 1 - i] = shape[i][j];
        }
    }

    return new Block(rotated);
}

public Block mirror() {
    int rows = shape.length;
    int cols = shape[0].length;
    char[][] mirrored = new char[rows][cols];

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            mirrored[i][cols - 1 - j] = shape[i][j];
        }
    }

    return new Block(mirrored);
}
```

```
}
```

2.4. Kelas Solver

- Dalam kelas solver, terdapat inisiasi tiga atribut untuk memudahkan penyimpanan kondisi:
 - Board board akan menyimpan kondisi papan (Baik selesai maupun tidak selesai).
 - Block[] blocks akan menyimpan Block yang diterima program dari pengguna dalam bentuk List of Block.
 - int visited akan menyimpan jumlah kemungkinan yang telah dikunjungi program dalam menyelesaikan Puzzler.

Kode Atribut solver (beserta konstruktornya)

```
private final Board board;  
private final Block[] blocks;  
public int visited = 0;  
  
public Solver(Board board, Block[] blocks) {  
    this.board = board;  
    this.blocks = blocks;  
}
```

- Pendekatan penyelesaian kemudian dilakukan secara rekursif dengan *method* boolean solve(int blockIndex).
- **Basis Rekursi: blockIndex == blocks.size()**
- Artinya, semua blok berhasil ditempatkan pada board tanpa terjadinya *backtracking* lebih lanjut.

Potongan Kode

```
if (blockIndex >= blocks.length) {  
    return true;  
}
```

- **Kasus:**
 - Dilakukan dua kali: **Normal** dan **dicerminkan horizontal**.
 - Dicoba dalam **empat kemungkinan rotasi** (0°, -90°, -180°, -270°)(Karena searah jarum jam).

Potongan Kode

```

Block block = blocks[blockIndex];
for (int flip = 0; flip < 2; flip++) {
    for (int rotation = 0; rotation < 4; rotation++) {
        for (int row = 0; row < board.rows; row++) {
            for (int col = 0; col < board.cols; col++) {
                visited++;
            }
        }
    }
}
(continue..)

```

- Dicoba ditempatkan pada setiap posisi di papan permainan.
- Jika berhasil ditempatkan, fungsi dipanggil secara rekursif untuk kepingan berikutnya (**solve(blockIndex + 1)**).
- Jika solusi tidak ditemukan, fungsi berlanjut menghapus block, kemudian loop dilanjutkan kembali dengan block yang sama, sehingga terjadi **backtracking**.
- Sebelum loop berlanjut, block disetel ulang (baik dirotasi maupun dicerminkan)

- **Method mengembalikan True** bila berhasil menemukan solusi

Potongan Kode

```

if (board.canPlaceBlock(block, row, col)) {
    board.placeBlock(block, row, col);

    // lanjutin blok berikutnya
    if (solve(blockIndex + 1)) {
        return true;
    }
    board.removeBlock(block, row, col); //
Backtrack
}
}
}
block = block.rotate();
}
block = block.mirror();

```

- Terdapat *method* String solve() yang mengikutsertakan tiga kemungkinan:
 - Solusi ada dan papan terisi penuh: Solusi benar-benar ditemukan.
 - Solusi ada, tapi papan tidak penuh: Kombinasi tidak ada.
 - Solusi tidak ada: Tidak ditemukan solusi.
- *method* String solve() juga menghitung waktu yang diperlukan untuk menyelesaikan Puzzle dalam milisekon.

- Dikembalikan sebagai String untuk diolah lebih lanjut di *export* solusi.

Potongan Kode

```
public String solve() {
    long startTime = System.nanoTime();

    boolean isFound = solve(0);

    long endTime = System.nanoTime();
    double time = (endTime - startTime) / 1_000_000.0;

    StringBuilder result = new StringBuilder();

    if (isFound && board.isFull()) {
        result.append("Solution found!\n");
    } else if (!isFound) {
        result.append("No solution found (No combination of
blocks can fit the board)\n");
    } else {
        result.append("No solution found (Board is not
full)\n");
    }

    result.append("Total possibilities visited:
").append(visited).append("\n");
    result.append(String.format("Time taken: %.3f ms\n",
time));

    return result.toString();
}
```

2.5. Kompleksitas Algoritma

Kompleksitas waktu mendekati $O(n!)$. Hal ini karena semua kemungkinan yang ada akan terus dicoba.

3. Source Code

Pada bagian ini, akan diperjelas secara singkat mengenai struktur proyek dan juga modularitas dari setiap kelas.

3.1. Struktur Proyek

Proyek ini memiliki struktur direktori dan file sebagai berikut:

```
Tucil1_13523149
├── bin
│   ├── Block.class
│   ├── Board.class
│   ├── Data.class
│   ├── Handler.class
│   ├── Main.class
│   └── Solver.class
├── doc
├── README.md
├── src
│   ├── Block.java
│   ├── Board.java
│   ├── Data.java
│   ├── Handler.java
│   ├── Main.java
│   └── Solver.java
└── test
    ├── input
    └── output
```

Penjelasan masing-masing direktori dan file:

- **bin/**: Berisi file .class dari kode di src.
- **doc/**: Direktori untuk dokumentasi proyek dalam bentuk .pdf.
- **src/**: Berisi kode sumber utama proyek.
 - **Solver.java**: Implementasi algoritma brute force
 - **Board.java**: Implementasi papan permainan.
 - **Block.java**: Implementasi kepingan puzzle beserta orientasinya.
 - **Handler.java**: Implementasi yang berkaitan dengan I/O
 - **Data.java**: Menyimpan tuple data dari hasil pembacaan Handler.
 - **Main.java**: Alur mulai program dan antarmuka pengguna
- **test/**: Direktori untuk file uji coba dalam bentuk .txt dalam folder input dan hasil dalam bentuk .txt dan .png dalam folder output.
- **README.md**: File readme yang mencakup deskripsi, persyaratan, instalasi, dan cara penggunaan aplikasi.

3.2. Main.java

Main.java merupakan awal mula dari program saat dijalankan pengguna. Secara umum, Main.java bertanggung jawab atas antarmuka pengguna. Fungsionalitas utamanya, mencakup:

3.2.1. *Graphical User Interface (GUI)*

Dalam Main, disertakan kode-kode umum yang bertanggung jawab pada GUI. Kode pada GUI ini menggunakan JavaFX versi 21.0.6.

3.2.2. Memilih Mode

Program dimulai dengan menanyakan cara inisiasi program. Terdapat dua cara, yakni melalui GUI ataupun CLI.

3.2.3. Meminta Masukan File

Pengguna dapat memilih (GUI) atau mengetik (CLI) .txt yang ingin diuji.

3.2.4. Menampilkan dan Menyimpan Hasil

Pengguna akan melihat hasil dari algoritma setelah dijalankan. Pengguna juga dapat menyimpan hasil tersebut ke dalam file .txt untuk dokumentasi atau .png sebagai gambar visualisasi solusi.

3.2.5. Penanganan Kesalahan

Jika file input tidak ditemukan atau tidak valid, program akan menampilkan pesan kesalahan yang sesuai agar pengguna dapat memperbaiki inputnya.

Kode dari Main

```
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.scene.paint.Color;
import javafx.stage.FileChooser;
import javafx.stage.Stage;
```

```

import java.io.File;
import java.util.List;
import java.util.Scanner;

public class Main extends Application {
    private ListView<String> testCaseList;
    private TextArea fileContentArea;
    private Label resultLabel;
    private Canvas canvas;
    private static final int CELL_SIZE = 50;

    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("IQ Puzzler Pro Solver (JavaFX)");

        // Panel kiri (Input)
        VBox leftPanel = new VBox(10);
        leftPanel.setPadding(new Insets(10));

        Label titleLabel = new Label("IQ Puzzler Pro Solver");
        Button uploadButton = new Button("Upload Test Case");
        testCaseList = new ListView<>();
        Button findButton = new Button("Find It!");
        Button saveTxtButton = new Button("Save .txt");
        Button savePngButton = new Button("Save .png");
        fileContentArea = new TextArea();
        fileContentArea.setEditable(false);

        leftPanel.getChildren().addAll(
            titleLabel, uploadButton, testCaseList,
            fileContentArea, findButton, saveTxtButton, savePngButton
        );

        // Panel kanan (Hasil dan File Content)
        VBox rightPanel = new VBox(10);
        rightPanel.setPadding(new Insets(10));
        Label resultTitle = new Label("Results");
        resultLabel = new Label("Searching Time: 0 ms | Cases:
0");

```

```

        canvas = new Canvas(600, 600);
        rightPanel.getChildren().addAll(resultTitle, resultLabel,
canvas);

        // Tombol-tombol (adh ada kurang ga yak)
        uploadButton.setOnAction(e ->
handleUpload(primaryStage));
        testCaseList.setOnMouseClicked(e ->
displaySelectedFile());
        findButton.setOnAction(e -> runSolver());

        HBox root = new HBox(leftPanel, rightPanel);
        Scene scene = new Scene(root, 1000, 600);
        primaryStage.setScene(scene);
        primaryStage.show();

        saveTxtButton.setOnAction(e -> {
            Board board = Handler.getBoard();
            if (board != null) {
                String[] parsedData =
Handler.parseOutput(resultLabel.getText(), board);
                Handler.saveSolutionToTxtGUI(parsedData[0],
parsedData[1], parsedData[2], parsedData[3], primaryStage);
            } else {
                showAlert("Error", "Solusi gagal disimpan!",
Alert.AlertType.ERROR);
            }
        });

        savePngButton.setOnAction(e -> {
            Board board = Handler.getBoard();
            if (board == null) {
                showAlert("Error", "Solusi gagal disimpan!",
Alert.AlertType.ERROR);
                return;
            }
            updateCanvas(board);
            Handler.saveSolutionToImageGUI(canvas, primaryStage);
        });

```

```

        loadTestCases();
    }

    private void loadTestCases() {
        List<String> files = Handler.getTestCases();
        testCaseList.getItems().setAll(files);
        if (!files.isEmpty()) {
            testCaseList.getSelectionModel().selectFirst();
            displaySelectedFile();
        }
    }

    private void displaySelectedFile() {
        String selectedFile =
testCaseList.getSelectionModel().getSelectedItem();
        if (selectedFile != null) {

fileContentArea.setText(Handler.readTestCase(selectedFile));
        }
    }

    private void runSolver() {
        String selectedFile =
testCaseList.getSelectionModel().getSelectedItem();
        if (selectedFile != null) {
            String result = Handler.runSolver(selectedFile);

fileContentArea.setText(Handler.readTestCase(selectedFile));
            resultLabel.setText(result);
            updateCanvas(Handler.getBoard());
        } else {
            showAlert("Error", "No test case selected woi!!",
Alert.AlertType.ERROR);
        }
    }

    private void updateCanvas(Board board) {

```

```

    if (board == null) return;

    GraphicsContext gc = canvas.getGraphicsContext2D();
    Color[][] colorGrid = board.getColorGrid();
    char[][] grid = board.grid;
    int rows = board.rows;
    int cols = board.cols;

    double cellWidth = canvas.getWidth() / cols;
    double cellHeight = canvas.getHeight() / rows;

    gc.clearRect(0, 0, canvas.getWidth(),
canvas.getHeight());

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            gc.setFill(colorGrid[i][j]);
            gc.fillRect(j * cellWidth, i * cellHeight, cellWidth,
cellHeight);
            gc.setStroke(Color.BLACK);
            gc.strokeRect(j * cellWidth, i * cellHeight,
cellWidth, cellHeight);

            gc.setFill(Color.BLACK);
            gc.fillText(String.valueOf(grid[i][j]), j * cellWidth
+ cellWidth / 3, i * cellHeight + cellHeight / 2);
        }
    }
}

private void handleUpload(Stage stage) {
    FileChooser fileChooser = new FileChooser();
    fileChooser.getExtensionFilters().add(new
FileChooser.ExtensionFilter("Text Files", "*.txt"));
    File selectedFile = fileChooser.showOpenDialog(stage);

    if (selectedFile != null) {
        boolean success = Handler.handleUpload(selectedFile);
    }
}

```

```

        if (success) {
            loadTestCases();
        } else {
            showAlert("Upload Failed", "Gagal upload :(",
Alert.AlertType.ERROR);
        }
    }
}

private void showAlert(String title, String message,
Alert.AlertType type) {
    Alert alert = new Alert(type);
    alert.setTitle(title);
    alert.setContentText(message);
    alert.showAndWait();
}

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

    while (true) {
        System.out.println("Selamat datang! Pilih opsi
menjalankan program:");
        System.out.println("1. GUI");
        System.out.println("2. CLI");
        System.out.println("3. Keluar");
        System.out.print("Pilihan: ");

        String choicesS = input.next();
        while (!choicesS.equals("1") && !choicesS.equals("2") &&
!choicesS.equals("3")) {
            System.out.println("Pilihan tidak valid!");
            System.out.print("Pilihan: ");
            choicesS = input.next();
        }

        int choice = Integer.parseInt(choicesS);
        if (choice == 1) {
            input.close();

```



```

        launch(args);
        return;
    } else if (choice == 2) {
        Handler handler = new Handler();
        Data data;

        input.nextLine();
        System.out.print("Masukkan nama file: ");
        String fileName = input.nextLine().trim();
        data = handler.handleFile(fileName);
        if (data == null) {
            continue;
        }

        Board board = new Board(data.N, data.M);
        Solver solver = new Solver(board, data.blocks);

        String result = solver.solve();
        System.out.println(result);
        board.printBoard();
        System.out.println("Ingin menyimpan data?");
        System.out.println("1. Simpan sebagai .txt");
        System.out.println("2. Simpan sebagai .png");
        System.out.println("3. Lewati.");
        choicesS = input.next();
        while (!choicesS.equals("1") && !choicesS.equals("2")
&& !choicesS.equals("3")) {
            System.out.println("Pilihan tidak valid!");
            System.out.print("Pilihan: ");
            choicesS = input.next();
        }
        choice = Integer.parseInt(choicesS);
        if (choice == 1) {
            String[] parsedData = Handler.parseOutput(result,
board);
            Handler.saveSolutionToTxtCLI(parsedData[0],
parsedData[1], parsedData[2], parsedData[3]);
        } else if (choice == 2) {
            System.out.println("Maaf, fitur ini hanya bisa

```

```
digunakan dalam GUI karena keterbatasan JavaFX.");  
    }  
    } else if (choice == 3) {  
        input.close();  
        System.exit(0);  
        return;  
    } else {  
        System.out.println("Pilihan tidak valid!");  
    }  
}  
}
```

3.3. Kode Lainnya

Untuk melihat isi kode dari kelas lain, dapat melihat Pranala Repositori yang disediakan di akhir. Secara garis besar,

- **Solver.java:** *Telah dijelaskan*
- **Board.java:** *Telah dijelaskan*
- **Block.java:** *Telah dijelaskan*
- **Handler.java:** Implementasi yang berkaitan dengan I/O. Mengandung fungsionalitas:
 - Membaca input .txt dan mengubahnya ke block[] (List of Block).
 - *Export* luaran sebagai .txt maupun .png
 - Validasi input dalam .txt yang diperlukan
- **Data.java:** Menyimpan tuple data dari hasil pembacaan Handler. Atributnya terdiri dari:

```

    public final int N, M, P;
    public final String S;
    public final Block[] blocks;

```

- **Main.java:** *Telah dijelaskan*

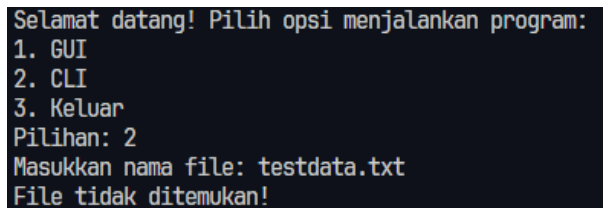
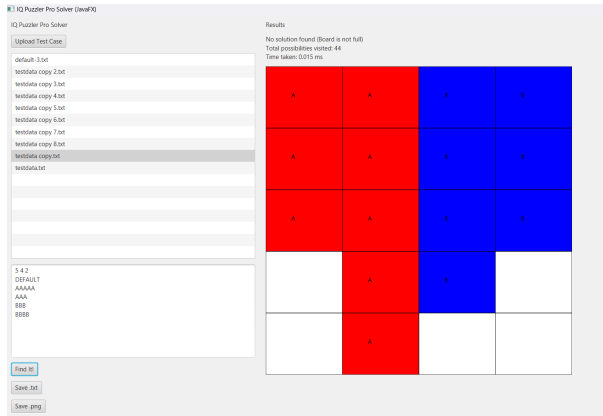
4. Uji Coba


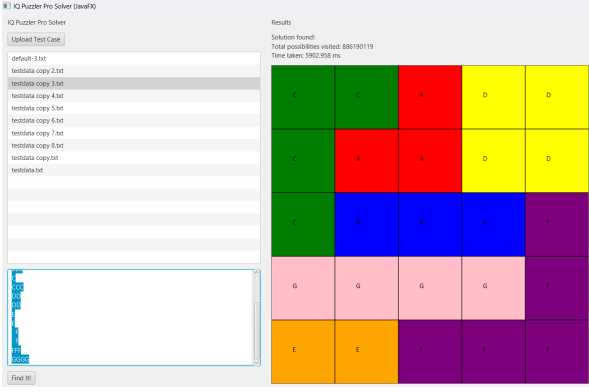
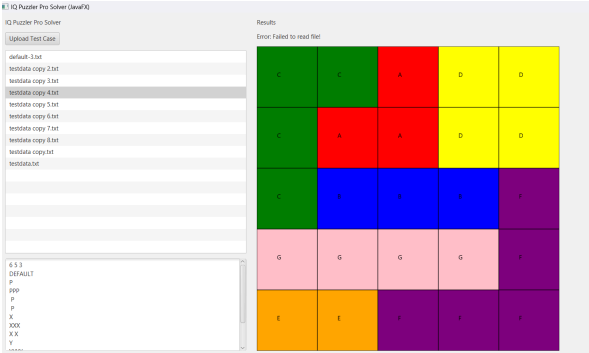
Program diuji coba dengan menggunakan file berformat **.txt**. Strukturnya mengikuti spesifikasi:

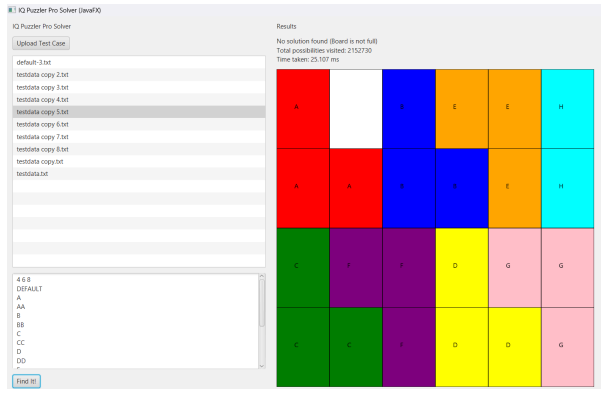
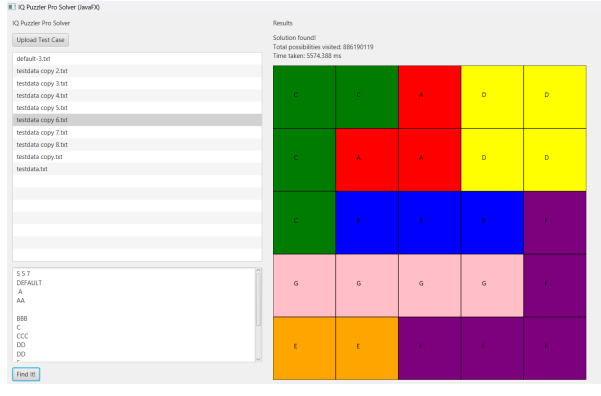
N M P
S
puzzle_1_shape
puzzle_2_shape
...
puzzle_P_shape

Dengan N dan M sebagai dimensi papan ($N \times M$), P sebagai jumlah blok puzzle, S sebagai jenis kasus yang hanya bernilai DEFAULT, serta setiap blok puzzle direpresentasikan oleh P huruf kapital unik (A-Z).

4.1. Tangkapan Layar Uji Coba

(Kosong)	 <p>(Itu messagenya salah, dia memang tidak deteksi kalau filenya kosong isinya)</p>
5 4 2 DEFAULT AAAAA AAA BBB BBBB	

<p>6 5 3 DEFAULT</p>	
<p>5 5 7 DEFAULT A AA BBB C CCC DD DD E E F F FFF GGGG</p>	
<p>6 5 3 DEFAULT P PPP P P X XXX X X Y YYYY Y SSSS S SS</p>	

4 6 8 DEFAULT A AA B BB C CC D DD E EE F FF G GG H H	
5 5 7 DEFAULT A AA BBB C CCC DD DD E E F F FFF GGGG	 <p>(Program bisa handle whitespace diantara blok, kalau ada)</p>

5. Pranala Repository

[Tucill_13523149](https://doi.org/10.24127/tucill.v1i1.13523149)

6. 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	✓	