

IF2211 Strategi Algoritma
Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force

Laporan Tugas Kecil 1

Disusun untuk memenuhi tugas mata kuliah IF2211 Strategi Algoritma
pada Semester 2 Tahun Akademik 2024/2025



Disusun oleh:
13523135 - Ahmad Syafiq

**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2024**

DAFTAR ISI

DAFTAR ISI.....	2
BAB I.....	3
2.1. Algoritma Brute Force.....	5
2.2. IQ Puzzle Pro.....	5
BAB III.....	6
3.1. Pembuatan Objek Kepingan.....	6
3.2. Pembuatan Objek Papan.....	6
3.3. Algoritma Penyelesaian IQ Puzzle Pro dengan Pendekatan Brute Force.....	6
4.1. Implementasi Program.....	7
4.1.1. Piece.java.....	7
4.1.2. Board.java.....	7
4.1.3. Solver.java.....	8
4.1.4. FileHandler.java.....	9
4.1.5. BoardView.java.....	9
4.1.6. Main.java.....	10
4.2. Kode Sumber.....	10
4.2.1. Piece.java.....	10
4.2.2. Board.java.....	12
4.2.3. Solver.java.....	14
4.2.4. FileHandler.java.....	16
4.2.5. BoardView.java.....	20
4.2.6. Main.java.....	22
BAB V.....	27
5.1. Tampilan Aplikasi.....	27
5.1.1. Halaman Utama.....	27
5.1.2. Halaman Proses.....	28
5.1.3. Halaman Keluaran.....	29
5.2. Pengujian Aplikasi.....	29
BAB VI.....	34

BAB I

DESKRIPSI MASALAH



Gambar 1.1.
Permainan IQ Puzzle Pro

IQ Puzzle Pro adalah permainan teka-teki yang mengharuskan pemain menyusun sejumlah kepingan dengan bentuk unik ke dalam sebuah papan permainan hingga seluruh papan terisi tanpa ada celah kosong atau kepingan yang bertumpuk. Setiap kepingan dapat diputar atau dicerminkan untuk menemukan kombinasi yang tepat agar seluruh papan terisi tanpa ada celah kosong atau kepingan yang bertumpuk.

IQ Puzzle Pro termasuk dalam kategori permainan kombinatorial, yaitu permainan yang melibatkan pemilihan dan penyusunan elemen dalam berbagai kemungkinan kombinasi.

Dalam permainan seperti ini, pemain harus mencari konfigurasi optimal dari sejumlah kemungkinan yang sangat besar. Contoh lain dari permainan kombinatorial meliputi teka-teki tiling, Sudoku, dan permainan strategi seperti catur. Permainan kombinatorial seringkali memiliki ruang solusi yang luas, sehingga metode eksplorasi yang sistematis diperlukan untuk menemukan solusi dalam waktu yang wajar. Algoritma brute force sering menjadi pilihan untuk menyelesaikan permainan kombinatorial seperti IQ Puzzle Pro karena kesederhanaan dan kemampuannya untuk memastikan bahwa solusi ditemukan dengan cara mencoba semua kemungkinan.

Dalam tugas kecil ini, penulis ditugaskan untuk membuat program sederhana dalam bahasa Java yang mengimplementasikan algoritma brute force untuk mencari solusi dalam permainan IQ Puzzler Pro. Algoritma brute force yang diimplementasikan tidak boleh memanfaatkan heuristik.

Program yang dibuat harus bisa menerima input file test case berekstensi .txt yang berisi dimensi papan $N \times M$, banyak blok puzzle P , jenis kasus S , serta bentuk blok puzzle sesuai format berikut:

```
N M P
S
puzzle_1_shape
puzzle_2_shape
...
puzzle_P_shape
```

Kemudian, program menghasilkan output berupa konfigurasi blok puzzle, waktu eksekusi, banyak kasus, serta prompt untuk menyimpan solusi dalam .txt

BAB II

TEORI SINGKAT

2.1. Algoritma Brute Force

Brute force adalah sebuah pendekatan yang langsung (straightforward) untuk memecahkan suatu masalah, biasanya didasarkan pada pernyataan masalah (problem statement) dan definisi konsep yang dilibatkan. Algoritma Brute Force pada dasarnya adalah alur penyelesaian suatu permasalahan dengan cara berpikir yang sederhana, langsung dan dengan cara yang jelas (obvious way) dan tidak membutuhkan suatu pemikiran yang cukup lama untuk dapat menyelesaikan sebuah permasalahan tertentu.

Algoritma brute force kurang mangkus karena membutuhkan banyak komputasi dan waktu yang lama, terutama untuk masalah berskala besar. Oleh karena itu, metode ini lebih cocok digunakan untuk menyelesaikan persoalan yang lebih kecil. Meskipun tidak efisien, algoritma ini tetap memiliki keunggulan karena mampu menyelesaikan hampir semua permasalahan yang ada.

2.2. IQ Puzzle Pro

IQ Puzzler Pro adalah permainan papan yang diproduksi oleh perusahaan Smart Games. Permainan ini terdiri dari sebuah papan dengan bentuk tertentu dan sekumpulan kepingan unik yang harus disusun agar mengisi papan tanpa celah. Setiap kepingan memiliki bentuk yang tidak beraturan, sehingga pemain harus menemukan kombinasi dan orientasi yang tepat agar semua kepingan dapat ditempatkan dengan benar.

Komponen penting dari permainan IQ Puzzler Pro terdiri dari:

1. Board (Papan) – Board merupakan komponen utama yang menjadi tujuan permainan dimana pemain harus mampu mengisi seluruh area papan menggunakan blok-blok yang telah disediakan.
2. 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.

BAB III

METODE PENYELESAIAN

3.1. Pembuatan Objek Kepingan

Fisik kepingan direpresentasikan dengan nilai-nilai true pada matriks dua dimensi berisi boolean. Kepingan dapat dirotasi 90 derajat dan dapat dicerminkan. Kepingan dibuat dengan kelas Piece. Adapun implementasi kelas Piece dapat dilihat pada Bab Implementasi.

3.2. Pembuatan Objek Papan

Papan permainan direpresentasikan dengan matriks dua dimensi berisi char. Petak kosong ditulis dengan '*', petak mati ditulis dengan '-', dan petak berisi ditulis dengan id kepingan yang mengisinya. Suatu kepingan dapat diletakkan di papan dan dicopot dari papan. Papan dibuat dengan kelas Board. Adapun implementasi kelas Board dapat dilihat pada Bab Implementasi.

3.3. Algoritma Penyelesaian IQ Puzzle Pro dengan Pendekatan Brute Force

Dalam tugas kecil kali ini, penyelesaian permasalahan IQ Puzzle Pro dilakukan dengan pendekatan brute force. Penyelesaian permasalahan dalam algoritma diawali dengan program menerima masukan dan memprosesnya menjadi papan kosong dan daftar kepingan. Kemudian, penyelesaian brute force dilakukan. Adapun langkah penyelesaian permasalahan dalam algoritma adalah sebagai berikut:

1. Program mengambil kepingan pertama dari daftar kepingan. Bentuk kepingan dan pencerminannya disimpan.
2. Program mencoba menaruh kepingan pada kolom pertama dan baris pertama di papan. Jika berhasil, program lanjut memproses kepingan selanjutnya.
3. Jika gagal, program mencoba menaruh pencerminan kepingan di lokasi yang sama. Jika berhasil, program lanjut memproses kepingan selanjutnya.
4. Jika gagal, kepingan dirotasi 90 derajat, kemudian program mengulang langkah 2-4.
5. Jika keempat kombinasi rotasi gagal, program mencoba menaruh kepingan pada kolom selanjutnya dan mengulang langkah 2-5.
6. Jika program belum berhasil menaruh kepingan dan tidak ada kolom selanjutnya, program mencoba menaruh kepingan pada kolom pertama di baris selanjutnya dan mengulang langkah 2-6.
7. Jika kepingan tidak dapat diletakkan dengan cara apapun, kepingan terakhir dicopot dari papan dan peletakannya dianggap gagal sehingga kepingan sebelumnya perlu mengulang langkah 3-6 (4-6 untuk kepingan yang sudah dicerminkan).
8. Jika tidak ada kepingan yang dapat dicopot, program menyatakan bahwa puzzle tidak memiliki penyelesaian.
9. Jika dan hanya jika papan terisi penuh dan semua kepingan terpakai, program menyatakan bahwa puzzle memiliki solusi dan menyimpan keadaan papan.

BAB IV IMPLEMENTASI

4.1. Implementasi Program

Program diimplementasikan dengan bahasa Java dan framework JavaFX dan dibuat dalam dua paket, yaitu model dan view. Paket model berisi logika program, sedangkan paket view mengatasi antarmuka pengguna. Paket model berisi file Piece.java, Board.java, Solver.java, FileHandler.java. Paket view berisi file Main.java dan BoardView.java. Adapun detail terkait tiap file adalah sebagai berikut:

4.1.1. Piece.java

File ini berisi implementasi kepingan puzzle

4.1.1.1. Atribut Piece

Nama	Tipe	Keterangan
id	char	id piece, sesuai file masukan
shape	array of array of boolean	representasi bentuk kepingan
width	integer	lebar kepingan
height	integer	tinggi kepingan

4.1.1.2. Metode Piece

Fungsi	Keterangan
Piece	konstruktor
getId	mengembalikan id kepingan
getShape	mengembalikan shape kepingan
rotate	mengembalikan hasil rotasi kepingan 90 derajat
flip	mengembalikan hasil pencerminan kepingan
isValidPiece	mengecek apakah semua bagian kepingan terhubung
printPiece	menampilkan bentuk kepingan di layar

4.1.2. Board.java

File ini berisi implementasi papan puzzle.

4.1.2.1. Atribut Board

Nama	Tipe	Keterangan
rows	integer	jumlah baris
cols	integer	jumlah kolom

grid	array of array of char	buffer papan <ul style="list-style-type: none"> - cell diisi dengan ID piece - cell kosong direpresentasikan dengan '*' - cell mati direpresentasikan dengan '-'
------	------------------------	---

4.1.2.2. Metode Board

Fungsi	Keterangan
Board	konstruktor
getRows	mengembalikan jumlah baris
getCols	mengembalikan jumlah kolom
getCell	mengembalikan nilai di grid sesuai indeks
canPlacePiece	mengembalikan nilai benar jika piece dapat diletakkan pada papan sesuai indeks
placePiece	menaruh piece pada papan sesuai indeks
removePiece	mencopot piece dari papan sesuai indeks
isBoardFull	mengembalikan nilai benar jika seluruh papan terisi
printBoard	menampilkan kondisi papan ke layar
boardToString	mengembalikan kondisi papan sebagai string
modifyBoard	mengubah bentuk papan untuk kasus custom

4.1.3. Solver.java

File ini berisi implementasi algoritma brute force.

4.1.3.1. Atribut Solver

Nama	Tipe	Keterangan
board	Board	menyimpan keadaan papan puzzle
pieces	array of Piece	menyimpan daftar kepingan puzzle
searchDuration	long	menyimpan waktu pencarian solusi
caseChecked	int	menyimpan jumlah kasus yang ditinjau

4.1.2.2. Metode Solver

Fungsi	Keterangan
Solver	konstruktor
getCaseChecked	mengembalikan jumlah kasus yang ditinjau
getBoard	mengembalikan keadaan papan puzzle

solve	mencari solusi puzzle dengan algoritma brute force
setSearchDuration	mengubah searchDuration
getSearchDuration	mengembalikan searchDuration

4.1.4. FileHandler.java

File ini berisi fungsi-fungsi untuk menangani masukan dan keluaran dalam bentuk file.

Fungsi	Keterangan
readPuzzleFromFile	membaca input .txt dan mengubahnya menjadi Board dan array of Piece untuk Solver
saveToFile	menyimpan output dalam bentuk .txt
saveSceneAsImage	menyimpan output dalam bentuk .png

4.1.5. BoardView.java

File ini berisi fungsi-fungsi untuk menampilkan board pada GUI.

4.1.3.1. Atribut BoardView

Nama	Tipe	Keterangan
board	Board	menyimpan keadaan papan puzzle
colorMap	Map char to Color	menyimpan peta warna untuk setiap kepingan
cells	array of array of Rectangle	petak-petak papan puzzle
grid	GridPane	wadah utama tampilan papan

4.1.2.2. Metode BoardView

Fungsi	Keterangan
BoardView	konstruktor
createBoardView	membuat tampilan papan puzzle
createGrid	membuat dan mengatur tata letak grid untuk papan puzzle
updateBoard	memperbarui tampilan papan puzzle
getColor	menentukan warna piece
generateColorMap	membuat peta warna untuk tiap piece
createColorLegend	membuat legenda yang menjelaskan warna yang digunakan tiap piece
getUsedPieces	mengembalikan daftar id piece yg digunakan

4.1.6. Main.java

File ini digunakan untuk menjalankan aplikasi program. File ini berisi fungsi-fungsi yang berkaitan dengan tampilan program.

4.1.3.1. Variabel

Nama	Tipe	Keterangan
primaryStage	Stage	representasi jendela utama aplikasi
selectedFile	String	path file input
puzzle	Solver	puzzle yang akan diselesaikan
fileLabel	Text	teks feedback pada halaman utama
solveButton	Button	tombol mulai pencarian solusi
customFont	Font	custom font yang digunakan
showSolutionSteps	boolean	flag untuk menentukan apakah langkah pencarian ditampilkan atau tidak
boardView	BoardView	tampilan papan puzzle

4.1.2.2. Fungsi

Fungsi	Keterangan
main	entry point program
start	menampilkan home saat aplikasi dimulai
showHomePage	menampilkan homepage
selectFile	membuka dialog pemilihan file
solvePuzzle	memulai penyelesaian puzzle
showProcessPage	menampilkan page saat proses penyelesaian puzzle
showSolutionPage	menampilkan page output penyelesaian puzzle
saveSolution	menyimpan solusi dalam bentuk .txt
loadFont	memuat custom font
createStyledButton	membuat tombol

4.2. Kode Sumber

4.2.1. Piece.java

```
package model;
```

```

import java.util.Stack;

public class Piece {
    private char id;
    private boolean[][] shape;
    private int width, height;

    public Piece(char id, boolean[][] shape) {
        this.id = id;
        this.shape = shape;
        this.width = shape[0].length;
        this.height = shape.length;
    }

    public char getId() {
        return id;
    }

    public boolean[][] getShape() {
        return shape;
    }

    public Piece rotate() {
        boolean[][] result = new boolean[width][height];
        for (int i = 0; i < height; i++) {
            for (int j = 0; j < width; j++) {
                result[j][height - 1 - i] = shape[i][j];
            }
        }
        return new Piece(id, result);
    }

    public Piece flip() {
        boolean[][] result = new boolean[height][width];
        for (int i = 0; i < height; i++) {
            for (int j = 0; j < width; j++) {
                result[i][width - 1 - j] = shape[i][j];
            }
        }
        return new Piece(id, result);
    }

    public boolean isValidPiece() {
        int[][] directions = {{1, 0}, {-1, 0}, {0, 1}, {0, -1}, {1, 1},
{1, -1}, {-1, 1}, {-1, -1}};
        boolean[][] visited = new boolean[height][width];
        int startX = -1, startY = -1;

        for (int i = 0; i < height; i++) {
            for (int j = 0; j < width; j++) {
                if (shape[i][j]) {
                    startX = i;
                    startY = j;
                    break;
                }
            }
            if (startX != -1) break;
        }

        if (startX == -1) return false;
    }

```

```

    int count = 0;
    Stack<int[]> stack = new Stack<>();
    stack.push(new int[]{startX, startY});
    visited[startX][startY] = true;

    while (!stack.isEmpty()) {
        int[] pos = stack.pop();
        count++;

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

            if (newX >= 0 && newX < height && newY >= 0 && newY <
width && shape[newX][newY] && !visited[newX][newY]) {
                stack.push(new int[]{newX, newY});
                visited[newX][newY] = true;
            }
        }
    }

    int total = 0;
    for (boolean[] row : shape) {
        for (boolean cell : row) {
            if (cell) total++;
        }
    }

    return count == total;
}

public void printPiece() {
    for (int i = 0; i < height; i++) {
        for (int j = 0; j < width; j++) {
            if (shape[i][j]) {
                System.out.print(id);
            }
            else {
                System.out.print("x");
            }
        }
        System.out.println();
    }
}
}
}

```

4.2.2. Board.java

```

package model;

import java.util.List;

public class Board {
    private int rows, cols;
    private char[][] grid; //pakai '*' untuk kosong

    public Board(int rows, int cols) {
        this.rows = rows;
        this.cols = cols;
    }
}

```

```

        this.grid = new char[rows][cols];
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                grid[i][j] = '*';
            }
        }
    }

    public Board(Board other) {
        this.rows = other.rows;
        this.cols = other.cols;
        this.grid = new char[rows][cols];

        for (int i = 0; i < rows; i++) {
            System.arraycopy(other.grid[i], 0, this.grid[i], 0, cols);
        }
    }

    public int getRows() {
        return rows;
    }

    public int getCols() {
        return cols;
    }

    public char getCell(int i, int j) {
        return grid[i][j];
    }

    public boolean canPlacePiece(Piece piece, int row, int col) {
        boolean[][] shape = piece.getShape();
        for (int i = 0; i < shape.length; i++) {
            for (int j = 0; j < shape[0].length; j++) {
                if (shape[i][j] && (row + i >= rows || col + j >= cols ||
grid[row + i][col + j] != '*')) {
                    return false;
                }
            }
        }
        return true;
    }

    public void placePiece(Piece piece, int row, int col) {
        boolean[][] shape = piece.getShape();
        for (int i = 0; i < shape.length; i++) {
            for (int j = 0; j < shape[0].length; j++) {
                if (shape[i][j]) {
                    grid[row + i][col + j] = piece.getId();
                }
            }
        }
    }

    public void removePiece(Piece piece, int row, int col) {
        boolean[][] shape = piece.getShape();
        for (int i = 0; i < shape.length; i++) {
            for (int j = 0; j < shape[0].length; j++) {
                if (shape[i][j]) {
                    grid[row + i][col + j] = '*';
                }
            }
        }
    }

```

```

    }
}

public boolean isBoardFull() {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if (grid[i][j] == '*') {
                return false;
            }
        }
    }
    return true;
}

public void printBoard() {
    for (char[] row : grid) {
        for (char cell : row) {
            System.out.print(cell + " ");
        }
        System.out.println();
    }
}

public String boardToString() {
    StringBuilder sb = new StringBuilder();
    for (char[] row : grid) {
        for (char cell : row) {
            sb.append(cell).append(" ");
        }
        sb.append("\n");
    }
    return sb.toString();
}

public void modifyBoard(List<char[]> model) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if (model.get(i)[j] == '.') grid[i][j] = '-';
        }
    }
}
}

```

4.2.3. Solver.java

```

package model;

import java.util.List;
import java.util.function.Consumer;

public class Solver {
    private Board board;
    private List<Piece> pieces;
    private long searchDuration;
    private int caseChecked = 0;

    public Solver(Board board, List<Piece> pieces) {
        this.board = board;
        this.pieces = pieces;
    }
}

```

```

    }

    public int getCaseChecked() {
        return caseChecked;
    }

    public Board getBoard() {
        return board;
    }

    public boolean solve(int idx, Consumer<Board> stateCallback) { //idx
bwt list pieces
        if (idx == pieces.size()) {
            return board.isBoardFull();
        }

        Piece piece = pieces.get(idx);
        Piece flipped_piece = piece.flip();
        for (int row = 0; row < board.getRows(); row++) {
            for (int col = 0; col < board.getCols(); col++) {
                for (int rotation = 0; rotation < 4; rotation++) {
                    if (board.canPlacePiece(piece, row, col)) {
                        board.placePiece(piece, row, col);
                        caseChecked++;
                        if (stateCallback != null) {
                            stateCallback.accept(new Board(board));
                        }
//                        board.printBoard();
//                        System.out.println("\n");
                        if (solve(idx+1, stateCallback)) {
                            return true;
                        }
                        board.removePiece(piece, row, col);
                    }
                    else if (board.canPlacePiece(flipped_piece, row, col))
{
                        board.placePiece(flipped_piece, row, col);
//                        board.printBoard();
//                        System.out.println("\n");
                        if (solve(idx+1, stateCallback)) {
                            return true;
                        }
                        board.removePiece(flipped_piece, row, col);
                    }
                    piece = piece.rotate();
                    flipped_piece = piece.flip();
                }
            }
        }
        return false;
    }

    public void setSearchDuration(long duration) {
        this.searchDuration = duration;
    }

    public long getSearchDuration() {
        return searchDuration;
    }

```

```
}
```

4.2.4. FileHandler.java

```
package model;

import javafx.embed.swing.SwingFXUtils;
import javafx.scene.Scene;
import javafx.scene.image.WritableImage;
import javafx.stage.FileChooser;
import javafx.stage.Stage;

import javax.imageio.ImageIO;
import java.io.*;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;
import java.util.Scanner;

public class FileHandler {
    public static Solver readPuzzleFromFile(String fileName)
    {
        Solver puzzle = null;
        try {
            File file = new File(fileName);
            Scanner sc = new Scanner(file);
            String line = sc.nextLine();
            String[] values = line.trim().split("\\s+");
            if (values.length != 3) {
                System.out.println("Format file tidak valid!\n");
                return null;
            }
            int expected_row = Integer.parseInt(values[0]);
            int expected_col = Integer.parseInt(values[1]);
            Board board = new Board(expected_row, expected_col);
            line = sc.nextLine();
            if (Objects.equals(line, "CUSTOM")) {
                List<char[]> tempMatrix = new ArrayList<>();
                for (int i0 = 0; i0 < expected_row; i0++) {
                    line = sc.nextLine();
                    char[] board_row = line.toCharArray();
                    if (board_row.length != expected_col) {
                        System.out.println("Custom Map tidak valid!");
                        return null;
                    }
                    else {
                        tempMatrix.add(board_row);
                    }
                }
                board.modifyBoard(tempMatrix);
            }
            else if (!Objects.equals(line, "DEFAULT")) {
                return null;
            }

            List<Piece> pieces = new ArrayList<>();
            List<Character> pieces_id = new ArrayList<>();
            int expected_length = Integer.parseInt(values[2]);
        }
    }
}
```



```

char currentId = '-';
List<boolean[]> tempPiece = new ArrayList<>();
int width = -1;
boolean isNewPiece = true;

while (sc.hasNextLine()) {
    line = sc.nextLine();
    char[] chars = line.toCharArray();
    boolean[] row = new boolean[chars.length];
    char current_row_id = '-';
    for (int i = 0; i < chars.length; i++) {
        if (current_row_id != chars[i] && chars[i] != ' ') {
            if (current_row_id == '-') {
                current_row_id = chars[i];
            }
            else {
                System.out.println("Format file tidak
valid!\n");
                return null;
            }
        }
        if (currentId == '-') {
            if (chars[i] != ' ') currentId = chars[i];
        }
        if (chars[i] == currentId) {
            row[i] = true;
        }
        else if (chars[i] == ' ') {
            row[i] = false;
        }
        else {
            if (chars[i] < 'A' || chars[i] > 'Z') {
                System.out.println("Format file tidak
valid!\n");
                return null;
            }
            int height = tempPiece.size();
            boolean[][] piece_shape = new
boolean[height][width];
            for (int i1 = 0; i1 < height; i1++) {
                boolean[] piece_row = tempPiece.get(i1);
                int piece_row_length = piece_row.length;
                for (int j1 = 0; j1 < width; j1++) {
                    if (j1 < piece_row_length) {
                        piece_shape[i1][j1] = piece_row[j1];
                    }
                    else {
                        piece_shape[i1][j1] = false;
                    }
                }
            }
            if (currentId != '-') {
                if (pieces_id.contains(currentId)) {
                    System.out.println("ID piece baru sudah
pernah digunakan!");
                    return null;
                }
                Piece piece = new Piece(currentId,
piece_shape);
                // System.out.println("ID: " + currentId);

```

```

//          System.out.println("Height: " + height);
//          System.out.println("Width: " + width);
//          System.out.println("Shape: ");
//          piece.printPiece();
//          if (!piece.isValidPiece()) {
//              System.out.println("Piece " + currentId +
" tidak terhubung!\n");
//              return null;
//          }
//          pieces.add(piece);
//          pieces_id.add(currentId);
//          }
//          width = -1;
//          currentId = chars[i];
//          row[i] = true;
//          tempPiece.clear();
//          isNewPiece = true;
//          }
//          }
//          if (isNewPiece) {
//              isNewPiece = false;
//              width = chars.length;
//          }
//          else {
//              if (width < chars.length) {
//                  width = chars.length;
//              }
//          }
//          tempPiece.add(row);
//          }
//          sc.close();
//          int height = tempPiece.size();
//          boolean[][] piece_shape = new boolean[height][width];
//          for (int i1 = 0; i1 < height; i1++) {
//              boolean[] piece_row = tempPiece.get(i1);
//              int piece_row_length = piece_row.length;
//              for (int j1 = 0; j1 < width; j1++) {
//                  if (j1 < piece_row_length) {
//                      piece_shape[i1][j1] = piece_row[j1];
//                  }
//                  else {
//                      piece_shape[i1][j1] = false;
//                  }
//              }
//          }
//          if (currentId != ' ') {
//              if (pieces_id.contains(currentId)) {
//                  System.out.println("ID piece baru sudah pernah
digunakan!");
//                  return null;
//              }
//              Piece piece = new Piece(currentId, piece_shape);
//              System.out.println("ID: " + currentId);
//              System.out.println("Height: " + height);
//              System.out.println("Width: " + width);
//              System.out.println("Shape: ");
//              piece.printPiece();
//              if (!piece.isValidPiece()) {
//                  System.out.println("Piece " + currentId + " tidak
terhubung!\n");

```

```

        return null;
    }
    pieces.add(piece);
    pieces_id.add(currentId);
}
if (expected_length != pieces.size()) {
    System.out.println("Jumlah piece tidak sesuai!\nJumlah
piece yang diminta: " + expected_length +
"\nJumlah piece yang diterima: " + pieces.size());
    return null;
}
else {
    puzzle = new Solver(board, pieces);
}
} catch (FileNotFoundException e) {
    System.out.println("Error: File " + fileName + " tidak
ditemukan.");
} catch (IllegalArgumentException e) {
    System.out.println(e.getMessage());
}
return puzzle;
}

public static void saveToFile(String output, Stage stage) {
    FileChooser fileChooser = new FileChooser();
    fileChooser.setTitle("Simpan Teks");
    fileChooser.getExtensionFilters().add(new
FileChooser.ExtensionFilter("Text Files", "*.txt"));
    File file = fileChooser.showSaveDialog(stage);
    if (file != null) {
        try (BufferedWriter writer = new BufferedWriter(new
FileWriter(file))) {
            writer.write(output);
            System.out.println("Output berhasil disimpan ke: " +
file.getAbsolutePath());
        } catch (IOException e) {
            System.err.println("Terjadi kesalahan saat menyimpan file:
" + e.getMessage());
        }
    }
}

public static void saveSceneAsImage(Scene scene, Stage stage) {
    FileChooser fileChooser = new FileChooser();
    fileChooser.setTitle("Simpan Gambar");
    fileChooser.getExtensionFilters().add(new
FileChooser.ExtensionFilter("PNG Image", "*.png"));

    File file = fileChooser.showSaveDialog(stage);
    if (file != null) {
        try {
            WritableImage image = new WritableImage((int)
scene.getWidth(), (int) scene.getHeight());
            scene.snapshot(image);
            ImageIO.write(SwingFXUtils.fromFXImage(image, null),
"png", file);
            System.out.println("Gambar berhasil disimpan: " +
file.getAbsolutePath());
        } catch (IOException e) {

```

```

        System.err.println("Gagal menyimpan gambar: " +
e.getMessage());
    }
}
}
}

```

4.2.5. BoardView.java

```

package view;

import javafx.application.Platform;
import javafx.geometry.Pos;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.paint.Color;
import javafx.scene.shape.Rectangle;
import javafx.scene.text.Text;
import model.Board;

import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;

public class BoardView {
    private Board board;
    private Map<Character, Color> colorMap;
    private Rectangle[][] cells;
    private GridPane grid;

    public BoardView(Board board) {
        this.board = board;
        this.colorMap = generateColorMap();
        this.cells = new Rectangle[board.getRows()][board.getCols()];
    }

    public VBox createBoardView() {
        VBox container = new VBox(15);
        container.setStyle("-fx-alignment: center; -fx-padding: 20px;");

        grid = createGrid();
        HBox gridWrapper = new HBox(grid);
        gridWrapper.setAlignment(Pos.CENTER); // Menjadikan grid berada di
tengah

        HBox legend = createColorLegend();

        container.getChildren().addAll(gridWrapper, legend);
        container.setAlignment(Pos.CENTER); // Memastikan seluruh isi VBox
berada di tengah
        return container;
    }

    private GridPane createGrid() {
        GridPane grid = new GridPane();
        grid.setStyle("-fx-hgap: 2px; -fx-vgap: 2px;");
    }
}

```

```

        for (int i = 0; i < board.getRows(); i++) {
            for (int j = 0; j < board.getCols(); j++) {
                char cell = board.getCell(i, j);
                Rectangle rect = new Rectangle(30, 30);
                rect.setFill(getColor(cell));
                rect.setStroke(Color.BLACK);
                grid.add(rect, j, i);
                cells[i][j] = rect;
            }
        }
        return grid;
    }

    public void updateBoard(Board newBoard) {
        this.board = newBoard;

        Platform.runLater(() -> {
            for (int i = 0; i < board.getRows(); i++) {
                for (int j = 0; j < board.getCols(); j++) {
                    char cell = board.getCell(i, j);
                    cells[i][j].setFill(getColor(cell));
                }
            }
        });
    }

    private Color getColor(char id) {
        if (id == '*') return Color.WHITE;
        if (id == '-') return Color.BLACK;
        return colorMap.getOrDefault(id, Color.GRAY);
    }

    private Map<Character, Color> generateColorMap() {
        Map<Character, Color> map = new HashMap<>();
        Color[] colors = {
            Color.RED, Color.BLUE, Color.GREEN, Color.ORANGE,
            Color.PURPLE, Color.PINK,
            Color.BROWN, Color.CYAN, Color.MAGENTA, Color.YELLOW,
            Color.LIME, Color.TEAL,
            Color.VIOLET, Color.GOLD, Color.SALMON, Color.INDIGO,
            Color.OLIVE, Color.NAVY,
            Color.MAROON, Color.AQUA, Color.CORAL, Color.TURQUOISE,
            Color.TOMATO, Color.KHAKI,
            Color.DARKGREEN, Color.SIENNA
        };

        for (int i = 0; i < 26; i++) {
            map.put((char) ('A' + i), colors[i % colors.length]);
        }
        return map;
    }

    private HBox createColorLegend() {
        HBox legend = new HBox(10);
        legend.setStyle("-fx-alignment: center; -fx-padding: 10px;
            -fx-border-color: black; -fx-border-width: 1px; -fx-background-color:
            lightgray;");

        Set<Character> usedPieces = getUsedPieces();
    }

```

```

        if (usedPieces.isEmpty()) return legend;

        for (char id : usedPieces) {
            Rectangle rect = new Rectangle(20, 20);
            rect.setFill(colorMap.get(id));
            rect.setStroke(Color.BLACK);
            Text label = new Text(String.valueOf(id));

            VBox item = new VBox(5);
            item.setStyle("-fx-alignment: center;");
            item.getChildren().addAll(rect, label);
            legend.getChildren().add(item);
        }

        return legend;
    }

    private Set<Character> getUsedPieces() {
        Set<Character> used = new HashSet<>();
        for (int i = 0; i < board.getRows(); i++) {
            for (int j = 0; j < board.getCols(); j++) {
                char cell = board.getCell(i, j);
                if (cell != '*' && cell != '-') {
                    used.add(cell);
                }
            }
        }
        return used;
    }
}

```

4.2.6. Main.java

```

package view;

import javafx.animation.FadeTransition;
import javafx.application.Application;
import javafx.application.Platform;
import javafx.concurrent.Task;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.CheckBox;
import javafx.scene.layout.*;
import javafx.scene.paint.Color;
import javafx.scene.text.Font;
import javafx.scene.text.Text;
import javafx.stage.FileChooser;
import javafx.stage.Stage;
import javafx.util.Duration;
import model.Board;
import model.FileHandler;
import model.Solver;
import java.io.File;

import static model.FileHandler.saveToFile;

```

```

public class Main extends Application {
    private Stage primaryStage;
    private String selectedFile = null;
    private Solver puzzle;
    private Text fileLabel;
    private Button solveButton;
    private Font customFont;
    private boolean showSolutionSteps = false;
    private BoardView boardView;

    public static void main(String[] args) {
        launch(args);
    }

    @Override
    public void start(Stage primaryStage) {
        this.primaryStage = primaryStage;
        customFont = loadFont("PressStart2P-Regular.ttf", 14);
        showHomePage();
    }

    private void showHomePage() {
        BorderPane layout = new BorderPane();
        layout.setStyle("-fx-background-color: linear-gradient(to bottom,
#1A1A40, #3D3D8B);");

        Text title = new Text("IQ Puzzle Pro Solver");
        title.setFont(loadFont("PressStart2P-Regular.ttf", 24));
        title.setFill(Color.LIGHTYELLOW);

        FadeTransition fadeIn = new FadeTransition(Duration.seconds(2),
title);
        fadeIn.setFromValue(0);
        fadeIn.setToValue(1);
        fadeIn.play();

        VBox titleBox = new VBox(title);
        titleBox.setAlignment(Pos.CENTER);
        titleBox.setPadding(new javafx.geometry.Insets(20));

        fileLabel = new Text("Belum ada file yang dipilih.");
        fileLabel.setFont(customFont);
        fileLabel.setFill(Color.WHITE);
        fileLabel.setStyle("-fx-font-weight: bold;");

        Button selectFileButton = createStyledButton("Pilih File");
        solveButton = createStyledButton("Cari Solusi");
        solveButton.setDisable(true);

        selectFileButton.setOnAction(e -> selectFile());
        solveButton.setOnAction(e -> solvePuzzle());

        CheckBox showStepsCheckBox = new CheckBox("Tunjukkan
Penyelesaian");
        showStepsCheckBox.setFont(customFont);
        showStepsCheckBox.setTextFill(Color.WHITE);
        showStepsCheckBox.setOnAction(e -> showSolutionSteps =
showStepsCheckBox.isSelected());

        VBox centerBox = new VBox(20, fileLabel, selectFileButton,

```

```

solveButton, showStepsCheckBox);
    centerBox.setAlignment(Pos.CENTER);
    centerBox.setPadding(new javafx.geometry.Insets(20));

    layout.setTop(titleBox);
    layout.setCenter(centerBox);
    Scene scene = new Scene(layout, 500, 300);

    primaryStage.setTitle("IQ Puzzle Pro Solver");
    primaryStage.setScene(scene);
    primaryStage.show();
}

private void selectFile() {
    FileChooser fileChooser = new FileChooser();
    fileChooser.setTitle("Pilih File Puzzle");
    fileChooser.getExtensionFilters().add(new
FileChooser.ExtensionFilter("Text Files", "*.txt"));
    File file = fileChooser.showOpenDialog(primaryStage);
    if (file != null) {
        selectedFile = file.getAbsolutePath();
        fileLabel.setText("File dipilih: " + file.getName());
        solveButton.setDisable(false);
    }
}

private void solvePuzzle() {
    if (selectedFile == null) {
        fileLabel.setText("Pilih file terlebih dahulu!");
        return;
    }

    puzzle = FileHandler.readPuzzleFromFile(selectedFile);
    if (puzzle == null) {
        fileLabel.setText("Format file tidak sesuai!");
        return;
    }
    showProcessPage();

    Task<Boolean> solveTask = new Task<>() {
        @Override
        protected Boolean call() {
            long startTime = System.nanoTime();
            boolean found = puzzle.solve(0, showSolutionSteps ?
this::updateBoardState : null);
            long endTime = System.nanoTime();
            long duration = (endTime - startTime) / 1_000_000;
            puzzle.setSearchDuration(duration);
            if (found) updateMessage("Solusi ditemukan!\nWaktu
pencarian: " + duration + " ms\nKasus ditinjau: " +
puzzle.getCaseChecked());
            else updateMessage("Solusi tidak ditemukan!\nWaktu
pencarian: " + duration + " ms\nKasus ditinjau: " +
puzzle.getCaseChecked());
            return found;
        }
    }

    private void updateBoardState(Board board) {
        Platform.runLater(() -> {
            boardView.updateBoard(board);

```



```

        });

        try {
            Thread.sleep(200);
        } catch (InterruptedException ignored) {}
    }
};

solveTask.setOnSucceeded(e -> {
    boolean found = solveTask.getValue();
    showSolutionPage(solveTask.getMessage(), found);
});
new Thread(solveTask).start();
}

private void showProcessPage() {
    BorderPane layout = new BorderPane();
    layout.setStyle("-fx-background-color: #3D3D8B;");

    Text resultText = new Text("Mencari solusi...");
    resultText.setFont(customFont);
    resultText.setFill(Color.LIGHTYELLOW);

    boardView = new BoardView(puzzle.getBoard());
    VBox centerBox = new VBox(20, resultText,
boardView.createBoardView());
    centerBox.setAlignment(Pos.CENTER);

    layout.setCenter(centerBox);
    Scene scene = new Scene(layout, 500, 300);
    primaryStage.setScene(scene);
}

private void showSolutionPage(String message, boolean found) {
    BorderPane layout = new BorderPane();
    layout.setStyle("-fx-background-color: linear-gradient(to bottom,
#1A1A40, #3D3D8B);");

    Text resultText = new Text(message);
    resultText.setFont(customFont);
    resultText.setFill(Color.LIGHTYELLOW);
    System.out.print(message);
    boardView.updateBoard(puzzle.getBoard());

    Button saveButton = createStyledButton("Simpan Teks");
    Button saveImageButton = createStyledButton("Simpan Gambar");
    Button backButton = createStyledButton("Kembali");

    saveButton.setOnAction(e -> saveSolution(found));
    saveImageButton.setOnAction(e ->
FileHandler.saveSceneAsImage(primaryStage.getScene(), primaryStage));
    backButton.setOnAction(e -> showHomePage());

    HBox buttonBox = new HBox(15, saveButton, backButton,
saveImageButton);
    buttonBox.setAlignment(Pos.CENTER);

    VBox centerBox;
    if (found) {
        centerBox = new VBox(20, resultText,

```

```

boardView.createBoardView(), buttonBox);

    }
    else {
        centerBox = new VBox(20, resultText, buttonBox);

    }
    centerBox.setAlignment(Pos.CENTER);
    centerBox.setPadding(new javafx.geometry.Insets(20));
    layout.setCenter(centerBox);

    Scene scene = new Scene(layout, 600, 600);
    primaryStage.setScene(scene);
}

private void saveSolution(boolean found) {
    if (found) {
        saveToFile("Solusi ditemukan!\n\n" +
puzzle.getBoard().boardToString() +
                "\n\nWaktu pencarian: " +
puzzle.getSearchDuration() + " ms\nKasus ditinjau: " +
puzzle.getCaseChecked(),
                primaryStage);
    }
    else {
        saveToFile("Solusi tidak ditemukan!\n\n" + "Waktu pencarian: "
+ puzzle.getSearchDuration() + " ms\nKasus ditinjau: " +
puzzle.getCaseChecked(),
                primaryStage);
    }
}

private Font loadFont(String fontFileName, double size) {
    return
Font.loadFont(getClass().getClassLoader().getResourceAsStream(fontFileNam
e), size);
}

private Button createStyledButton(String text) {
    Button button = new Button(text);
    button.setFont(customFont);
    button.setStyle(
        "-fx-background-color: #FFD700; " +
        "-fx-text-fill: black; " +
        "-fx-font-size: 14px; " +
        "-fx-border-color: white; " +
        "-fx-border-width: 3px; " +
        "-fx-padding: 10px; " +
        "-fx-font-family: 'PressStart2P';"
    );
    return button;
}
}

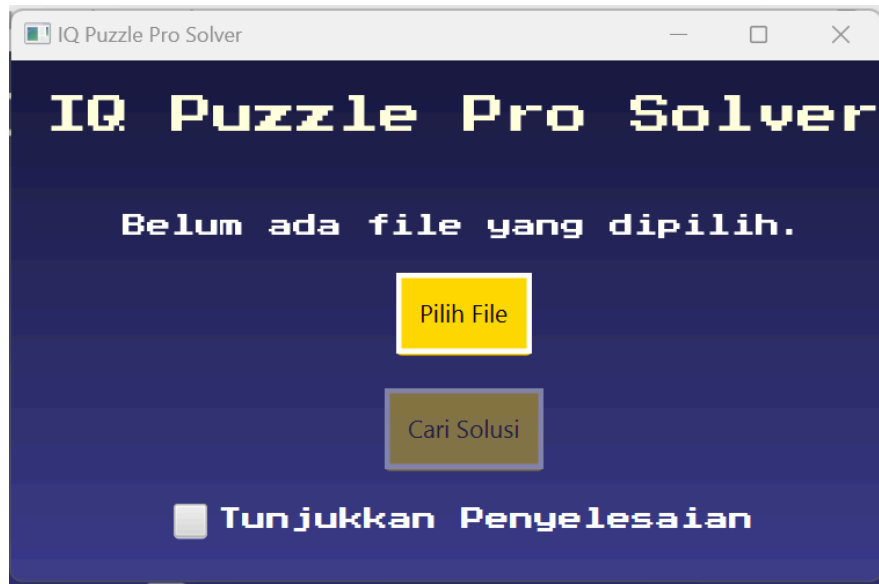
```

BAB V

PENGUJIAN PROGRAM

5.1. Tampilan Aplikasi

5.1.1. Halaman Utama

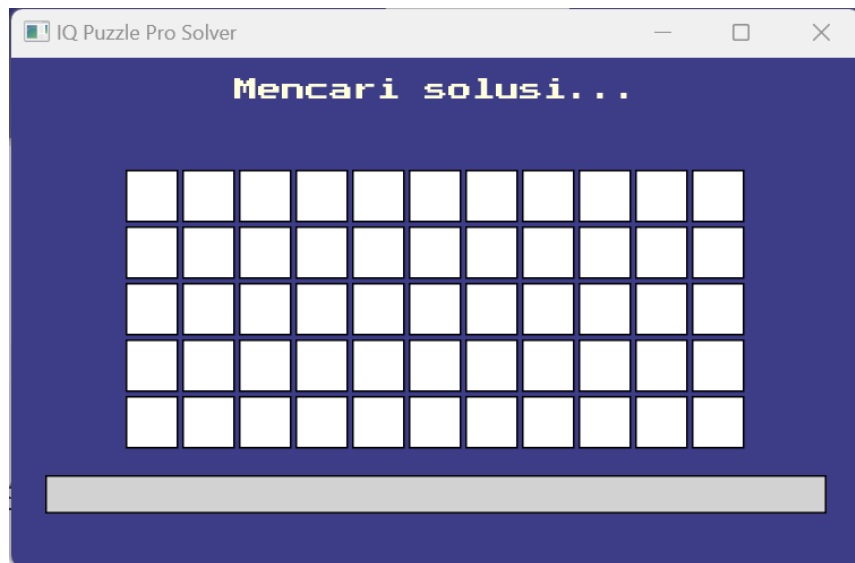


Gambar 5.1.1. Tampilan halaman utama

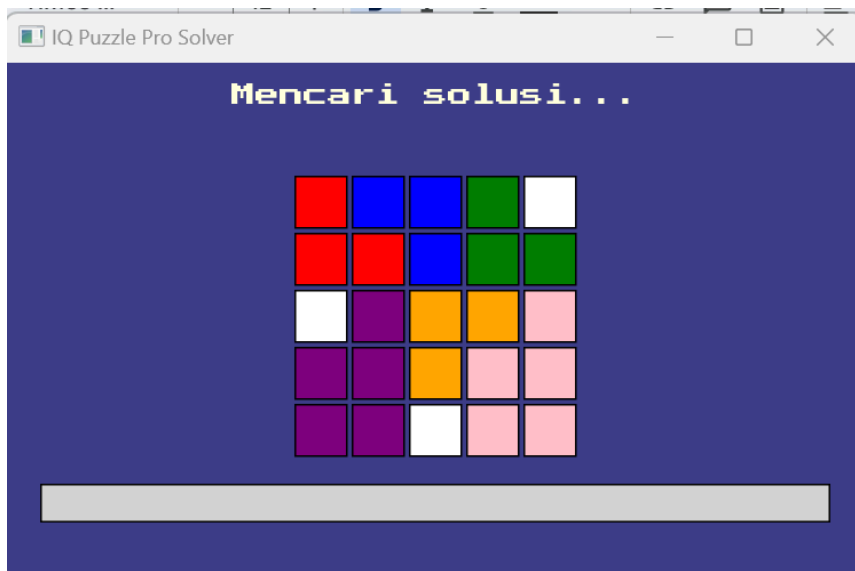


Gambar 5.1.2. Tampilan halaman utama setelah pilih file

5.1.2. Halaman Proses

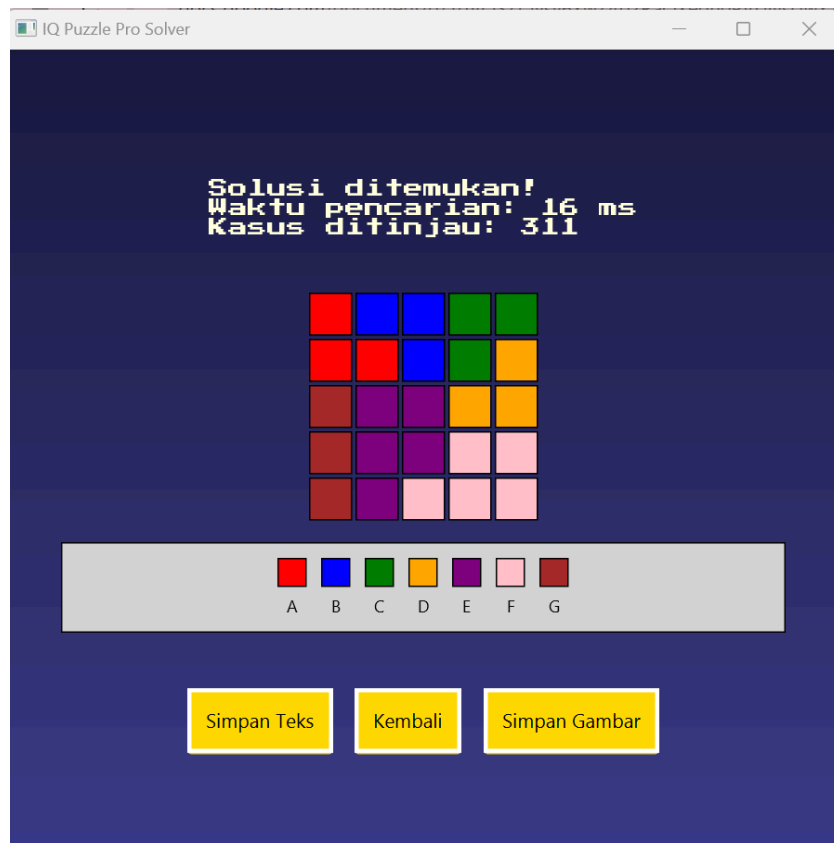


Gambar 5.1.3. Tampilan halaman proses




Gambar 5.1.4. Tampilan halaman proses dengan tunjukkan penyelesaian


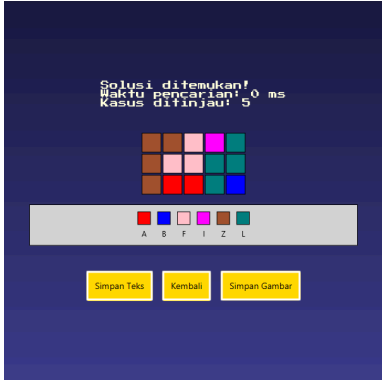
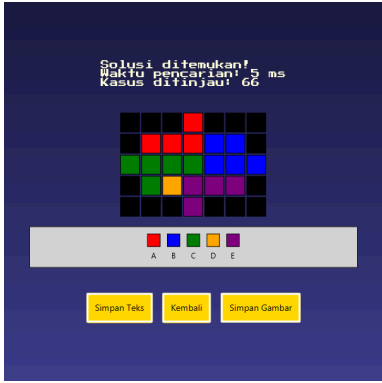
5.1.3. Halaman Keluaran

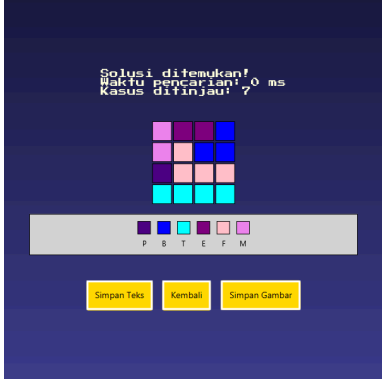
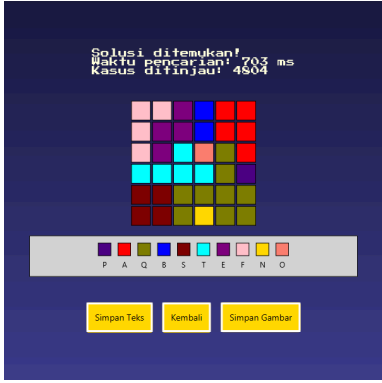


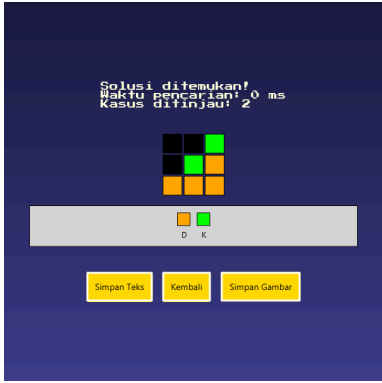


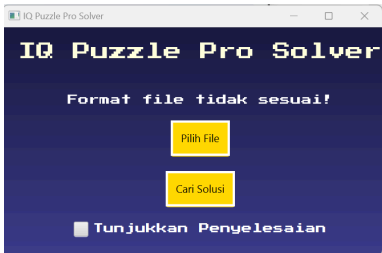
Gambar 5.1.5. Tampilan halaman keluaran

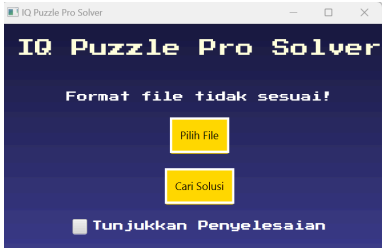
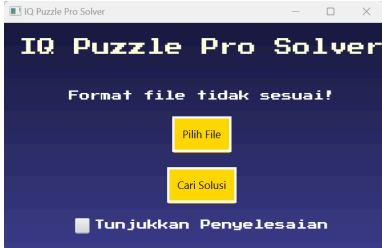

5.2. Pengujian Aplikasi

Masukan	Keluaran .txt	Keluaran .png	Keluaran Terminal
5 5 7 DEFAULT A AA B BB C CC D DD EE EE E FF FF F GGG	Solusi ditemukan! A B B C C A A B C D G E E D D G E E F F G E F F F Waktu pencarian: 13 ms Kasus ditinjau: 311		Solusi ditemukan! Waktu pencarian: 13 ms Kasus ditinjau: 311

3 10 8 DEFAULT A A AA BB BB B B B B C DD D E E E FF GGG G G GGG H	Solusi ditemukan! A B B B B B B G G G A B B C D D F G H G A A E E E D F G G G Waktu pencarian: 26191 ms Kasus ditinjau: 380714		Solusi ditemukan! Waktu pencarian: 26191 ms Kasus ditinjau: 380714
3 5 6 DEFAULT ZZ Z Z F FF I L LL L A A B	Solusi ditemukan! Z Z F I L Z F F L L Z A A L B Waktu pencarian: 0 ms Kasus ditinjau: 5		Solusi ditemukan! Waktu pencarian: 0 ms Kasus ditinjau: 5
5 7 5 CUSTOM ...X... .XXXXX. XXXXXXXX .XXXXX. ...X... A AAA BB BBB CCCC C D EEE	Solusi ditemukan! - - - A - - - - A A A B B - C C C C B B B - C D E E E - - - - E - - - Waktu pencarian: 5 ms Kasus ditinjau: 66		Solusi ditemukan! Waktu pencarian: 5 ms Kasus ditinjau: 66

E			
4 4 6 DEFAULT M M EE B BB FFF F T T T T P	Solusi ditemukan! M E E B M F B B P F F F T T T T Waktu pencarian: 0 ms Kasus ditinjau: 7		Solusi ditemukan! Waktu pencarian: 0 ms Kasus ditinjau: 7
6 6 10 DEFAULT FF F F E EE E B B AA AA A O P T TT T T Q Q QQQQ QQ Q SS SS N	Solusi ditemukan! F F E B A A F E E B A A F E T O Q A T T T T Q P S S Q Q Q Q S S Q N Q Q Waktu pencarian: 703 ms Kasus ditinjau: 4804		Solusi ditemukan! Waktu pencarian: 703 ms Kasus ditinjau: 4804

3 3 2 CUSTOM ..X .XX XXX K K D D DD	Solusi ditemukan! - - K - K D D D D Waktu pencarian: 0 ms Kasus ditinjau: 2		Solusi ditemukan! Waktu pencarian: 0 ms Kasus ditinjau: 2
3 3 2 DEFAULT A AA BBB B B	Solusi tidak ditemukan! Waktu pencarian: 0 ms Kasus ditinjau: 24		Solusi tidak ditemukan! Waktu pencarian: 0 ms Kasus ditinjau: 24
3 3 2 CUSTOM ..X .XX XXX K K DD D DD	Solusi tidak ditemukan! Waktu pencarian: 0 ms Kasus ditinjau: 8		Solusi tidak ditemukan! Waktu pencarian: 0 ms Kasus ditinjau: 8
5 5 7 CUSTOM A AA B BB C CC D DD EE EE E FF FF F GGG	-		Custom Map tidak valid!

5 11 12 DEFAULT CC C CC W WW WW L LLLL Z ZZ Z Z C C CCC TTT T M MMMM G GG GG QQ QQ BB B PP PPP N	-		ID piece baru sudah pernah digunakan!
	-		File Kosong!
Mau bikin SIM bayar polisi Ketilang di jalan bayar polisi Touring motor gede bayar polisi Angkot mau ngetem bayar polisi	-		Format file tidak valid!

Aduh aduh ku tak punya uang Untuk bisa bayar polisi			
---	--	--	--

LAMPIRAN

Lampiran Checklist Spesifikasi Program

Tabel 5.1. Tabel *checklist* spesifikasi program

No	Poin	Ya	Tidak
1	Program berhasil dikompilasi tanpa kesalahan	v	
2	Program berhasil dijalankan	v	
3	Solusi yang diberikan program benar dan mematuhi aturan permainan	v	
4	Program dapat membaca masukan berkas .txt serta menyimpan solusi dalam berkas .txt	v	
5	Program memiliki <i>Graphical User Interface</i> (GUI)	v	
6	Program dapat menyimpan solusi dalam bentuk file gambar	v	
7	Program dapat menyelesaikan kasus konfigurasi <i>custom</i>	v	
8	Program dapat menyelesaikan kasus konfigurasi Piramida (3D)		v
9	Program dibuat oleh saya sendiri	v	

Lampiran Repository Program

Repository program dapat dibuka lewat tautan berikut:

https://github.com/iammadsfq/Tucil1_13523135