Tugas Kecil 1

IF2211 Strategi Algoritma

Penyelesaian **IQ Puzzler Pro** dengan Algoritma Brute Force

Disusun oleh:

Samuel Gerrard Hamonangan Girsang / 13523064

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2025

I. Pendahuluan

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:

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.

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.

II. ALGORITMA

Algoritma yang digunakan pada program ini adalah algoritma ***Brute Force*** yang merupakan algoritma paling sederhana yang mudah dimengerti. Pada kasus ini, IQ Puzzler memiliki 2 komponen utama yaitu Board dan Piece. Board adalah papan tempat permainan dilakukan dan Piece adalah potongan puzzle yang harus diletakkan pada Board. Piece memiliki warna dan bentuk yang beragam.

Dengan algoritma Brute Force, program akan meletakkan satu per satu Piece pada papan. Program akan meletakkan Piece A (atau piece pertama) pada Board mulai dari bagian paling atas kiri. Kemudian Piece B akan diletakkan di sebelah Piece A. Jika bentuk awal dari Piece B tidak bisa diletakkan di sebelah Piece A, maka akan dicoba semua variasi dari Piece B setelah dilakukan rotasi maupun refleksi pada Piece B, hal ini dilakukan dengan Piece seterusnya.

Jika Piece B sama sekali tidak bisa diletakkan di sebelah Piece A dengan semua variasi yang ada. Kita kembali ke Piece A dan mencoba variasi lain dari Piece A. Setelah itu, variasi Piece B yang memungkinkan akan diletakkan pada sebelah Piece A. Jika semua variasi Piece A telah dicoba tetapi tidak memungkinkan untuk meletakkan piece B di sebelahnya, maka piece A bukan Piece yang tepat untuk diletakkan pada pojok kiri atas, maka dari itu kita akan mencoba menggunakan Piece B pada pojok kiri atas.

Semua Piece yang ada akan dicoba satu persatu secara berurutan dan akan dilakukan *backtracking* jika ada kemungkinan salah peletakkan. Hal ini dilakukan sampai puzzle tersebut berhasil dipecahkan atau sama sekali tidak ditemukan sebuah solusi.

III. SOURCE CODE

1. InOut.java

Kelas ini digunakan untuk mengatur input dan output dari program. Pembacaan input file .txt serta save result dalam file .txt maupun dalam bentuk gambar

```
public class InOut {  3 usages  Henshou
    public static void readInput(String filename) {  1 usage  Henshou
        try {
            BufferedReader br = new BufferedReader(new FileReader(filename));
            String[] firstLine = br.readLine().split( regex: " ");
            int N = Integer.parseInt(firstLine[0]);
            int M = Integer.parseInt(firstLine[1]);
            int P = Integer.parseInt(firstLine[2]);

            br.readLine();

            List<char[][]> shapes = new ArrayList<>();
            char currAlph = '\0';
            List<String> currLine = new ArrayList<>();

            String line;
            while ((line = br.readLine()) != null) {
                if (line.isEmpty()) {
                    continue;
                }
                char firstChar = line.trim().charAt(0);
                if (Character.isLetter(firstChar) && firstChar != currAlph) {
                    if (!currLine.isEmpty()) {
                        char[][] shapeMatrix = new char[currLine.size()][];
                        for (int i = 0; i < currLine.size(); i++) {
                            shapeMatrix[i] = currLine.get(i).toCharArray();
                        }
                        shapes.add(shapeMatrix);
                        currLine.clear();
                    }
                    currAlph = firstChar;
                }
                currLine.add(line);
            }
```

```java
if (!currLine.isEmpty()) {
    char[][] shapeMatrix = new char[currLine.size()][];
    for (int i = 0; i < currLine.size(); i++) {
        shapeMatrix[i] = currLine.get(i).toCharArray();
    }

    shapes.add(shapeMatrix);
}

Piece[] pieces = new Piece[shapes.size()];
for (int i = 0; i < shapes.size(); i++) {
    pieces[i] = matrixToCoordinates(shapes.get(i));
}

long start = System.currentTimeMillis(); // Start timing
Solver solver = new Solver(N, M, pieces);
boolean solvePuzzle = solver.solvePuzzle( startX: 0, startY: 0);
long end = System.currentTimeMillis();
long timeTaken = end - start;

if (solvePuzzle) {
    solver.printBoard();
    Main.setResults(solver.getBoard(), solver.getIteration(), timeTaken);
} else {
    char[][] emptyBoard = new char[N][M];
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < M; j++) {
            emptyBoard[i][j] = ' ';
        }
    }
    Main.setResults(emptyBoard, solver.getIteration(), timeTaken);
    System.out.println("No Solution!");
```

```java
private static Piece matrixToCoordinates(char[][] shape) {  1 usage  Henshou
    char alphabet = '\0';
    ArrayList<int[]> coordinates = new ArrayList<>();

    for (int i = 0; i < shape.length; i++) {
        for (int j = 0; j < shape[i].length; j++) {
            if (Character.isLetter(shape[i][j])) {
                if (alphabet == '\0') {
                    alphabet = shape[i][j];
                }
                coordinates.add(new int[]{i, j});
            }
        }
    }

    return new Piece(alphabet, coordinates);
}

public static void saveResult(String filename, char[][] board, int iterations, long timeTaken) {  1 usage  Henshou
    try (BufferedWriter writer = new BufferedWriter(new FileWriter(filename))) {
        for (char[] row : board) {
            for (char cell : row) {
                writer.write( str: cell + " ");
            }
            writer.newLine();
        }
        writer.write( str: "\nWaktu pencarian: " + timeTaken + " ms\n");
        writer.write( str: "\nBanyak kasus yang ditinjau: " + iterations + "\n");
    } catch (IOException e) {
        System.err.println("Error saving results: " + e.getMessage());
    }
}
```

```java
public static void saveAsImage(JPanel boardPanel, String filePath) {  1 usage  Henshou
    int width = boardPanel.getWidth();
    int height = boardPanel.getHeight();
    BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
    Graphics2D graph = image.createGraphics();
    boardPanel.paint(graph);
    graph.dispose();

    try {
        File file = new File(filePath);
        ImageIO.write(image,  formatName: "png", file);
    } catch (Exception e) {
        JOptionPane.showMessageDialog( parentComponent: null,  message: "Error saving image: " + e.getMessage());
    }
}
```

2. Piece.java
   Kelas ini menghandle object Piece dengan method rotasi maupun refleksi

```java
public class Piece {  23 usages  Henshou
    char alphabet;  5 usages
    ArrayList<int[]> coordinates;  15 usages

    public Piece(char alphabet, ArrayList<int[]> coordinates) {  4 usages
        this.alphabet = alphabet;
        this.coordinates = new ArrayList<>();
        for (int[] point : coordinates) {
            this.coordinates.add(new int[]{point[0], point[1]});
        }
    }

    public void rotatePiece() {  2 usages  Henshou
        ArrayList<int[]> rotated = new ArrayList<>();
        for (int[] point : coordinates) {
            int x = point[0];
            int y = point[1];
            rotated.add(new int[]{-y, x});
        }
        this.coordinates = rotated;
        positivePiece();
    }

    public void mirrorPiece() {  1 usage  Henshou
        ArrayList<int[]> mirrored = new ArrayList<>();
        for (int[] point : coordinates) {
            int x = point[0];
            int y = point[1];
            mirrored.add(new int[]{-x, y});
        }
        this.coordinates = mirrored;
        positivePiece();
    }
```

```java
private void positivePiece() {  2 usages  Henshou
    int minX = Integer.MAX_VALUE;
    int minY = Integer.MAX_VALUE;

    for (int[] coordinate : coordinates) {
        if (coordinate[0] < minX) minX = coordinate[0];
        if (coordinate[1] < minY) minY = coordinate[1];
    }

    ArrayList<int[]> positive = new ArrayList<>();
    for (int[] coordinate : coordinates) {
        positive.add(new int[]{coordinate[0] - minX, coordinate[1] - minY});
    }

    this.coordinates = positive;
}
```

3. PieceList.java
   Kelas ini menghandle object piecelist yaitu sebuah array list of Piece yang menyimpan variasi dari setiap Piece.

```java
public PieceList() { this.pieceList = new ArrayList<>(); }

public Piece[] getVariation(Piece piece) {  1 usage  Henshou
    Piece[] variationArray = new Piece[8];

    Piece temp = new Piece(piece.alphabet, copyCoordinates(piece.coordinates));

    for (int i = 0; i < 4; i++) {
        variationArray[i] = new Piece(temp.alphabet, copyCoordinates(temp.coordinates));
        temp.rotatePiece();
    }

    temp.mirrorPiece();
    for (int i = 4; i < 8; i++) {
        variationArray[i] = new Piece(temp.alphabet, copyCoordinates(temp.coordinates));
        temp.rotatePiece();
    }

    return variationArray;
}

public void makePieceList(Piece piece) { pieceList.add(getVariation(piece)); }

private ArrayList<int[]> copyCoordinates(ArrayList<int[]> original) {  3 usages  Henshou
    ArrayList<int[]> copiedList = new ArrayList<>();
    for (int[] point : original) {
        copiedList.add(new int[]{point[0], point[1]});
    }
    return copiedList;
}
```

4. Solver.java
   Kelas yang berisikan algoritma utama untuk menyelesaikan puzzle

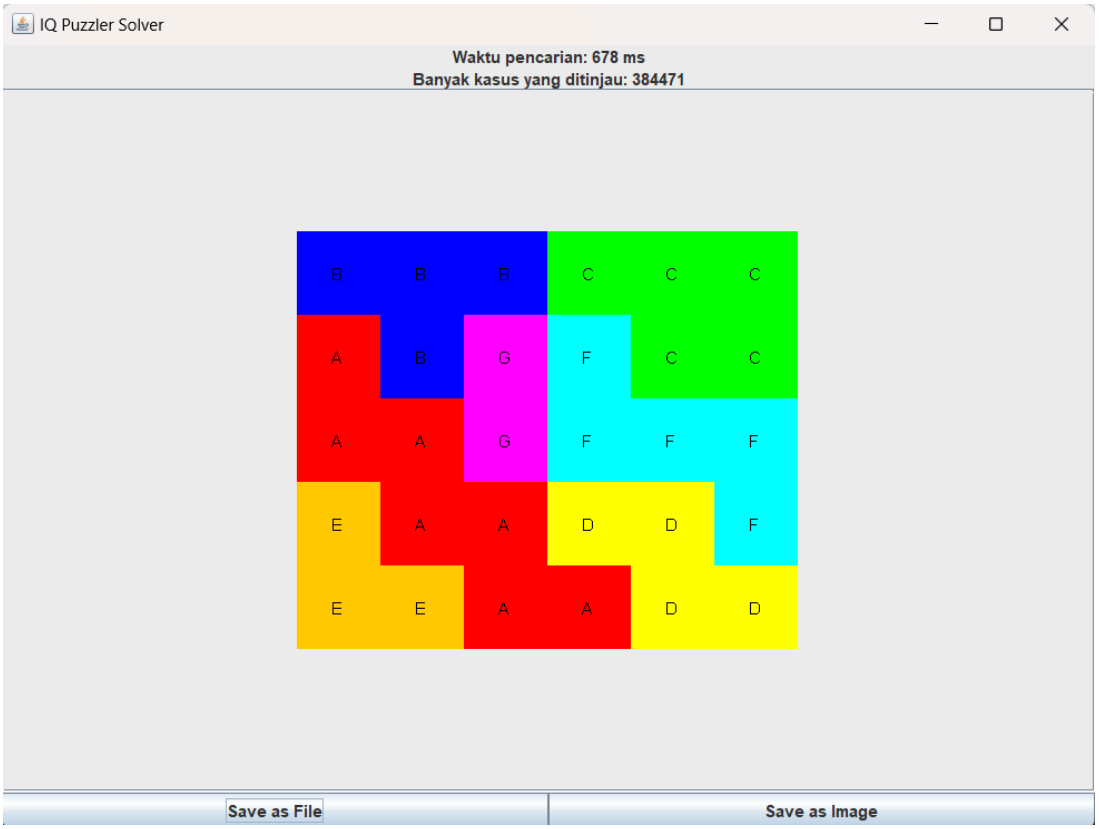
```
    public boolean solvePuzzle(int startX, int startY) {  3 usages   Henshou
//          System.out.println("\nTrying position " + startX + "," + startY);
//          printBoard();

        iteration++;
        if (isSolved()) return true;

        int nextX = startX, nextY = startY + 1;
        if (nextY >= M) {
            nextX = startX + 1;
            nextY = 0;
        }

        if (board[startX][startY] != ' ') {
            if (startX < N - 1 || nextY < M-1) {
                return solvePuzzle(nextX, nextY);
            } else {
                return isSolved();
            }
        }


        for (int i = 0; i < pieces.length; i++) {
            if (pieces[i] == null) continue;

            Piece currentPiece = pieces[i];
            PieceList listPiece = new PieceList();
            listPiece.makePieceList(currentPiece);

            for (Piece[] pieceVariation : listPiece.pieceList) {
                for (Piece availPiece : pieceVariation) {
                    if (canPlace(availPiece, startX, startY)) {
                        placePiece(availPiece, startX, startY);
        for (int i = 0; i < N; i++) {
            System.arraycopy(board[i],  srcPos: 0, copyBoard[i],  destPos: 0, M);
        }
        return copyBoard;
    }
```

```
                    Piece temp = pieces[i];
                    pieces[i] = null;

                    if (startX < N - 1 || nextY < M) {
                        if (solvePuzzle(nextX, nextY)) {
                            return true;
                        }
                    } else {
                        if (isSolved()) {
                            return true;
                        }
                    }

//
                    removePiece(availPiece, startX, startY);
                    pieces[i] = temp;
//
                }
            }
        }
    }

    return false;
}

private boolean canPlace (Piece piece, int startX, int startY) {  1 usage  Henshou
    for (int[] coordinate : piece.coordinates) {
        if (startX + coordinate[0] >= N || startY + coordinate[1] >= M || board[startX + coordinate[0]][startY + coordinate[1]] != ' ')
    }
    return true;
}
```

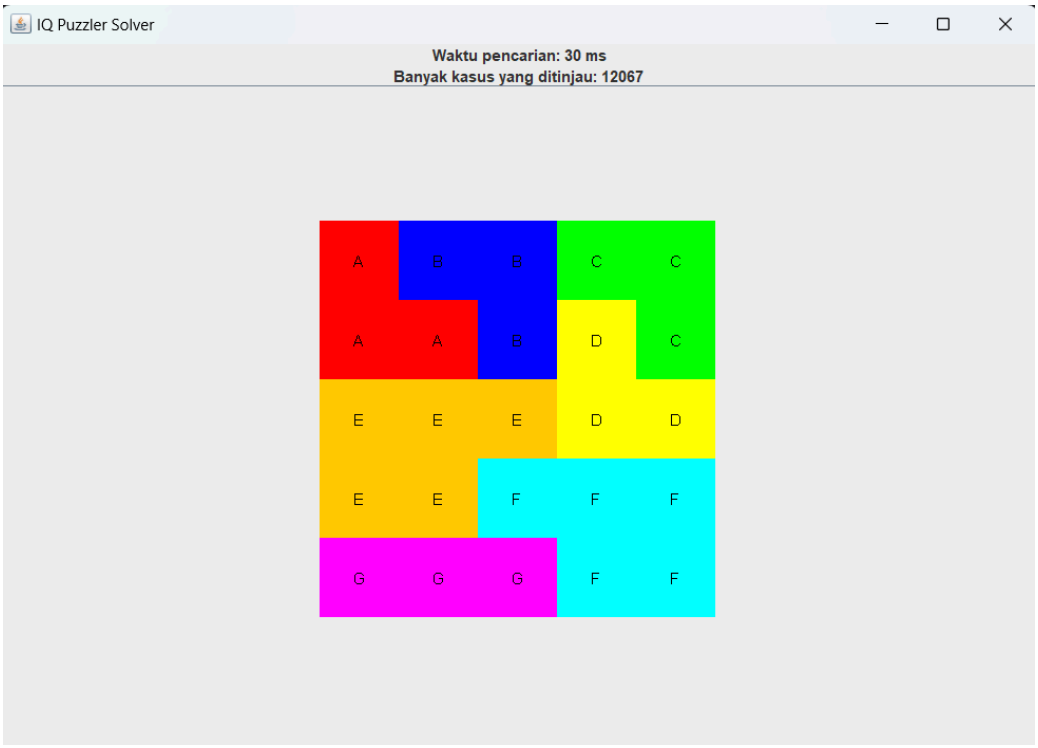
```java
private void placePiece(Piece piece, int startX, int startY) {  1 usage  Henshou
    for (int[] coordinate : piece.coordinates) {
        board[startX + coordinate[0]][startY + coordinate[1]] = piece.alphabet;
    }
}

private void removePiece (Piece piece, int startX, int startY) {  1 usage  Henshou
    for (int[] coordinate : piece.coordinates) {
        board[startX + coordinate[0]][startY + coordinate[1]] = ' ';
    }
}

private boolean isSolved() {  3 usages  Henshou
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < M; j++) {
            if (board[i][j] == ' ') return false;
        }
    }
    return true;
}

public void printBoard() {  1 usage  Henshou
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < M; j++) {
            System.out.print(board[i][j] + " ");
        }
        System.out.println();
    }
}
```

IV. TEST CASE

```
5 6 7
DEFAULT
AA
 AA
  AA
   A
B
BB
B
C
CC
CC
D
DD
 D
EE
E
FF
 F
 FF
GG
```

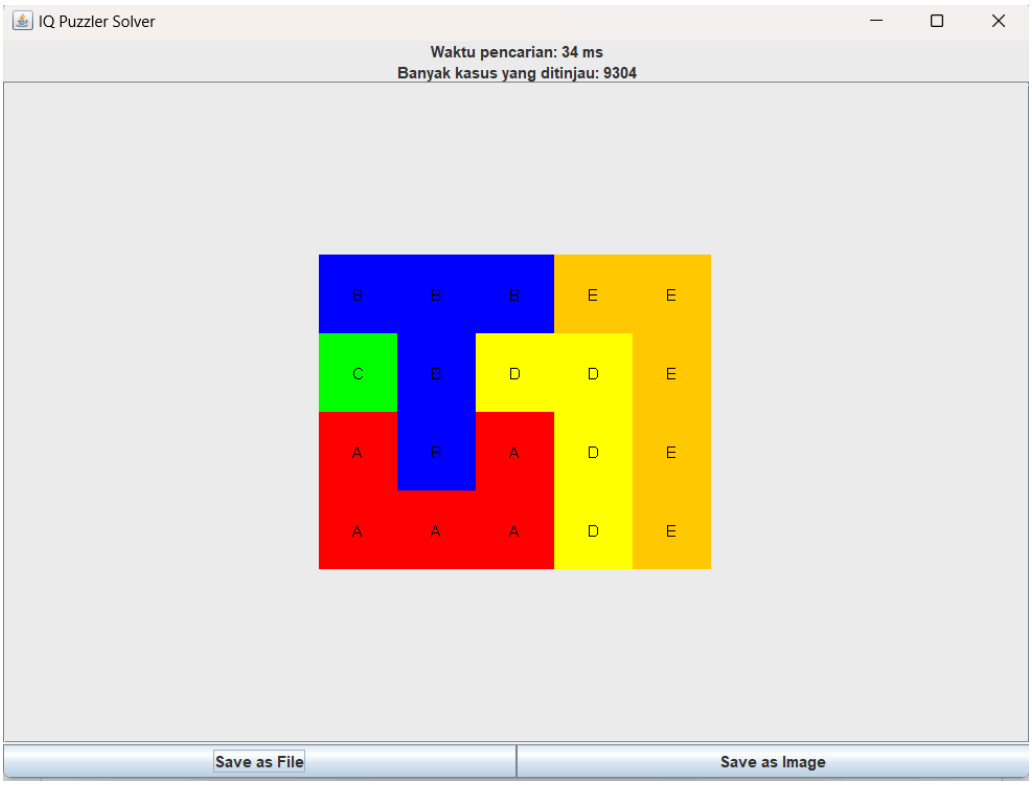


```
5 5 7
DEFAULT
A
AA
B
BB
C
CC
D
DD
EE
EE
E
FF
FF
F
GGG
```

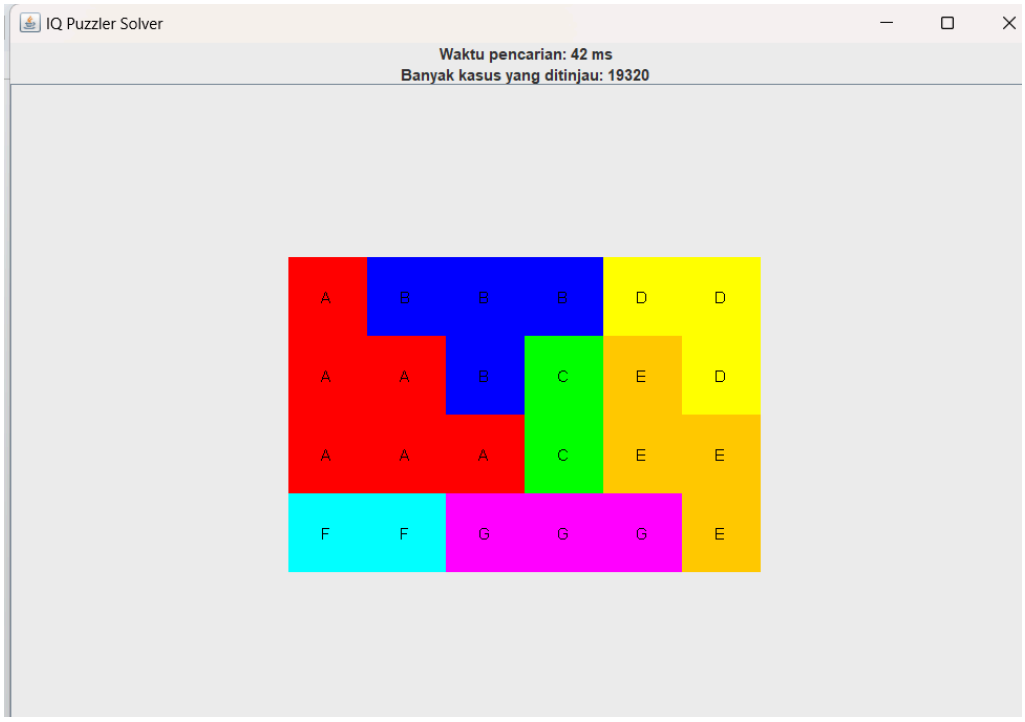



```
2 2 1
DEFAULT
A
A
A
A
```

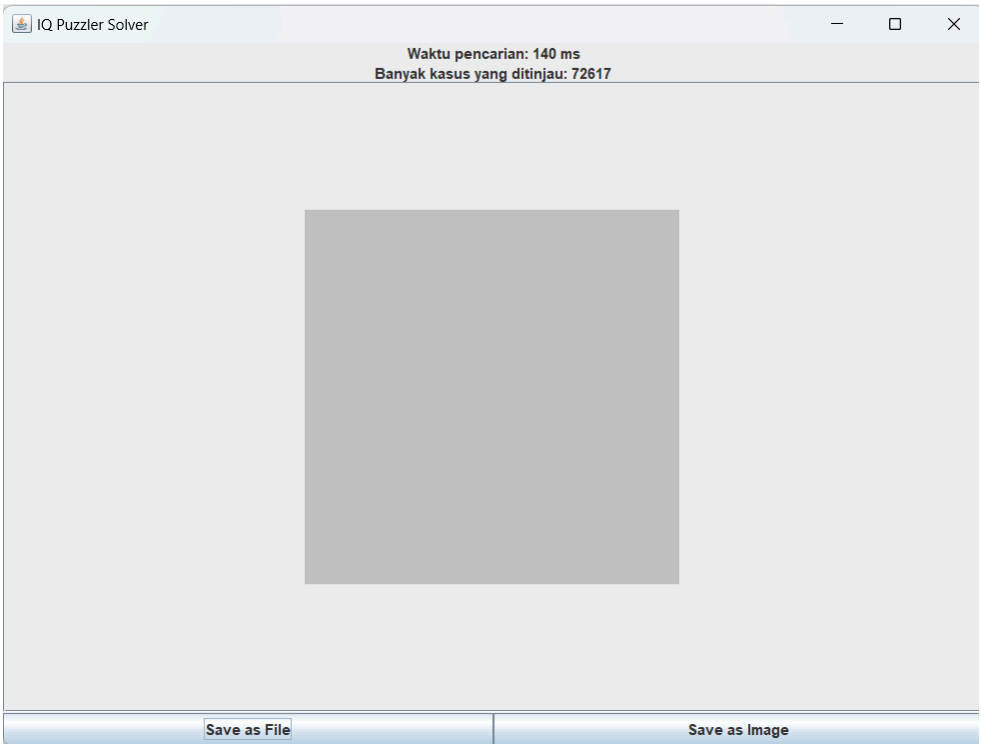
```
4 5 5
DEFAULT
AA
A
AA
B
BBB
B
C
D
D
DD
E
EEEE
```



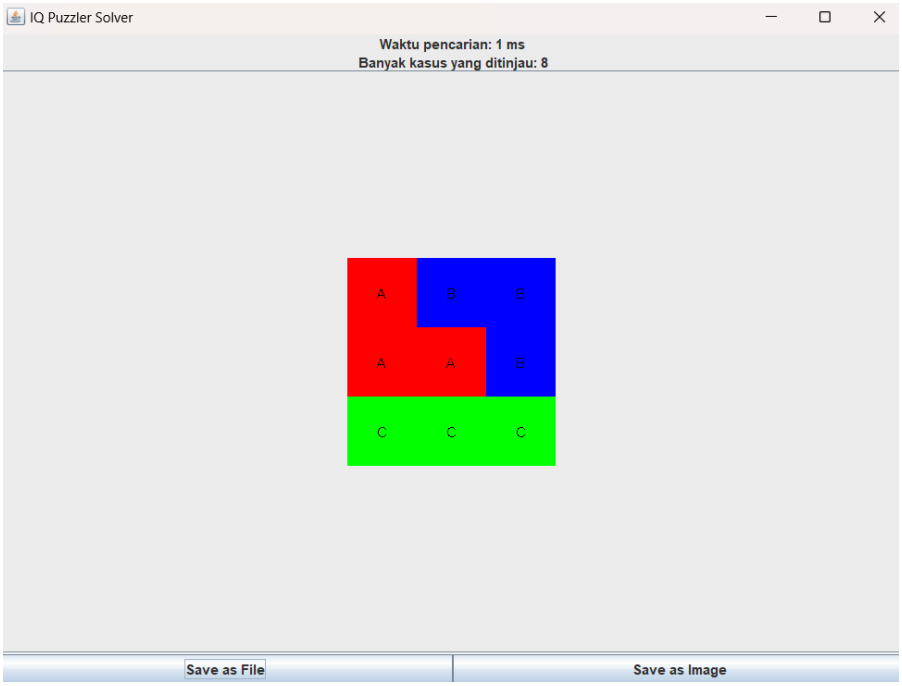

```
4 6 7
DEFAULT
A
AA
AAA
B
BB
B
CC
D
DD
E
EE
 E
FF
GGG
```



```
5 5 6
DEFAULT
A
AA
A
B
BB
BBB
C
CC
C
D
DD
D
E
EE
E
F
FF
F
```




```
3 3 3
DEFAULT
A
AA
B
BB
C
C
C
```



IV. 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 *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	✔	

Link repositori: https://github.com/Henshou/Tucil1_13523064.git
Spesifikasi: Spesifikasi Tugas Kecil 1 Stima 2024/2025