Laporan Tugas Kecil 1

NAMA: Lutfi Hakim Yusra

NIM: 13523084

Pendahuluan

Pada mata kuliah Strategi Algoritma IF2211, diberikan tugas membuat sebuah program yang memanfaatkan metode *Brute Force* untuk menyelesaikan sebuah *puzzle.* Mirip dengan permainan *IQ Puzzler Pro*, kita diberikan sebuah *board* dengan panjang dan lebar tertentu, dan kumpulan *block* dengan bentuk-bentuk tertentu. Kumpulan *block* tersebut harus digunakan semua yang tersedia hingga *board* terpenuhi.

Pembahasan Algoritma

Algoritma Brute Force yang digunakan mengikuti langkah-langkah berikut:

1. Inisiasi *board* dan *block*.
2. Mencari titik yang kosong pada board.
3. Cari *block* pertama yang valid (dapat dipasang tanpa menabrak atau *out of bounds*).
4. Mencoba tiap rotasi (beserta inversnya) yang valid dari *block* tersebut.
5. Letakkan balok tersebut.
6. Ulangi pencarian dari *step* 2, hingga *board* penuh, atau tidak ditemukan *block* yang valid.
    a. Jika tidak ditemukan *block* yang valid, *undo* block yang telah diletakkan dan mencoba rotasinya.
    b. Jika rotasinya tidak memenuhi, mencoba balok-balok lainnya yang tersedia.
    c. Jika telah dicoba setiap balok, maka dapat disimpulkan tidak ada solusi.
7. Ketika *board* penuh, maka telah ditemukan solusi dari puzzle.

Struktur Program

Program terdiri dari beberapa kelas utama:

- **Class Board**: Merepresentasikan *board* dalam bentuk matriks, dan juga mengandung fungsionalitas peletakkan *block* dan algoritma solving.

```java
public class Board {  7 usages  pixelatedbus
    public int row;  13 usages
    public int col;  13 usages
    public char[][] matrix;  14 usages
    public int iteration;  4 usages

    // Constructor
    public Board(int rowInput, int colInput) {  1 usage  pixelatedbus
        row = rowInput;
        col = colInput;
        matrix = new char[row][col];

        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                matrix[i][j] = '_';
            }
        }
        iteration = 0;
    }

    // Methods
    public boolean checkFor(char n) {  1 usage  pixelatedbus
        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                if (matrix[i][j] == n) {
                    return true;
                }
            }
        }
        return false;
    }
```

```java
public class Board {  7 usages  pixelatedbus
    public boolean placeBlock(Block block, int x, int y) {  1 usage  pixelatedbus
//                int xCoord = coords[0] + x;
//                int yCoord = coords[1] + y;
//                matrix[xCoord][yCoord] = block.letter;
//            }
//            return true;
//        }
//        iteration++;
//        System.out.println("CHANGING ITERATION");
//        return false;
        for (int i = -1; i < 1; i++){
            for (int j = -1; j < 1; j++){
                if (checkBlock(block,  x: x + i,  y: y + j)){
                    for (int[] coords : block.coordinates){
                        int xCoord = coords[0] + x + i;
                        int yCoord = coords[1] + y + j;
                        if(xCoord >= 0 && xCoord < row && yCoord >= 0 && yCoord < col){
                            matrix[xCoord][yCoord] = block.letter;
                        } else {
                            break;
                        }
                    }
                    return true;
                }
                System.out.println("CHANGING ITERATION");
            }
        }
        return false;
    }
```

```java
public boolean solveBoard(BlockList blockList, int x, int y) {  2 usages  pixelatedbus
    if (isFull()) {
        return blockList.blackList.size() == blockList.blocks.size();
    }
    for (Block[] blocks : blockList.blocks) {
        for (Block currentBlock : blocks) {
            if (!blockList.blackList.contains(currentBlock.letter)) {
                currentBlock.printBlock();
                char currentLetter = currentBlock.letter;
                if (placeBlock(currentBlock, x, y)) {
                    printBoard();
                    System.out.println("CURRENT LETTER: " + currentLetter);
                    int[] nextEmpty = findEmpty();
                    x = nextEmpty[0];
                    y = nextEmpty[1];
                    blockList.blackList.add(currentBlock.letter);
                    if (solveBoard(blockList, x, y)) {
                        return true;
                    } else {
                        System.out.println("REMOVING BLOCK");
                        blockList.blackList.removeLast();
                        removeBlock(currentLetter);
                    }
                }
            }
        }
    }
    iteration++;
    System.out.println("BACKTRACKING");
    return false;
}
```

- **Class Block**: Merepresentasikan *block* dalam bentuk kumpulan koordinat relatif dan tipe huruf, dan juga mengandung fungsionalitas rotasi dan inversi *block*.

```java
public class Block {  17 usages  pixelatedbus
    public ArrayList<int[]> coordinates;  11 usages
    public char letter;  11 usages

    public Block(ArrayList<int[]> inputCoordinates, char letterInput){  3 usages  pixelatedbus
        // Construct
        coordinates = inputCoordinates;
        letter = letterInput;
    }

    public Block(ArrayList<String> blockInput) {  1 usage  pixelatedbus
        ArrayList<int[]> coordinatesList = new ArrayList<int[]>();
        char letterBlock = '_'; // temp
        boolean letterSet = false;
        for (int x = 0; x < blockInput.size(); x++) {
            String currentString = blockInput.get(x);
            for (int y = 0; y < currentString.length(); y++) {
                if (currentString.charAt(y) != ' ') {
                    if (!letterSet) {
                        letterBlock = currentString.charAt(y);
                        letterSet = true;
                    }
                    coordinatesList.add(new int[]{x, y});
                }
            }
        }
        coordinates = coordinatesList;
        letter = letterBlock;
    }
```

```java
public Block rotateBlock(){  12 usages  pixelatedbus
    ArrayList<int[]> coordinatesList = new ArrayList<int[]>();


    for (int[] currentCoordinate : coordinates) {
        int newCoordinateX = currentCoordinate[1];
        int newCoordinateY = -currentCoordinate[0];
        int[] newCoordinate = {newCoordinateX, newCoordinateY};
        coordinatesList.add(newCoordinate);
    }

    return new Block(coordinatesList, letter);
}

public Block invertBlock(){  4 usages  pixelatedbus
    ArrayList<int[]> coordinatesList = new ArrayList<int[]>();
    for (int[] currentCoordinate : coordinates) {
        int[] newCoordinate = {currentCoordinate[0], -currentCoordinate[1]};
        coordinatesList.add(newCoordinate);
    }

    return new Block(coordinatesList, letter);
}
```

- **Class BlockList**: Menyimpan daftar semua *block* yang tersedia, beserta variasi rotasi dan inversenya, dan juga menyimpan daftar *block* yang telah digunakan.

```java
public class BlockList {  4 usages  pixelatedbus
    public ArrayList<Block[]> blocks;  5 usages
    public ArrayList<Character> blackList;  5 usages

    public BlockList(){  1 usage  pixelatedbus
        blocks = new ArrayList<Block[]>();
        blackList = new ArrayList<Character>();
    }

    public void setBlocks(Block block){  1 usage  pixelatedbus
        blocks.add(new Block[]{
                block,
                block.rotateBlock().forcePositive(),
                block.rotateBlock().rotateBlock().forcePositive(),
                block.rotateBlock().rotateBlock().rotateBlock().forcePositive(),
                block.invertBlock().forcePositive(),
                block.invertBlock().rotateBlock().forcePositive(),
                block.invertBlock().rotateBlock().rotateBlock().forcePositive(),
                block.invertBlock().rotateBlock().rotateBlock().rotateBlock().forcePositive()
        });
    }
}
```

- **Class GUI**: Menampilkan *Graphical User Interface* dan juga penyimpanan gambar solusi.

```java
public class GUI {  1 usage  pixelatedbus

    public static void mainMenu(){  1 usage  pixelatedbus
        JFrame frame = new JFrame( title: "Tucil 1");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize( width: 800,  height: 600);
        frame.setLocationRelativeTo(null);

        JPanel panel = new JPanel();
        panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));

        JLabel label = new JLabel( text: "Sigma Puzzle Solver");
        label.setFont(new Font( name: "Arial", Font.BOLD,  size: 40));
        label.setAlignmentX(Component.CENTER_ALIGNMENT);
        panel.add(label);

        JButton button1 = new JButton( text: "Solve!");
        JButton button2 = new JButton( text: "Exit");
        ImageIcon gambarHina = new ImageIcon( filename: "../src/citlali.jpg");
        JLabel imageLabel = new JLabel(gambarHina);
        imageLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
        panel.add(imageLabel);


        button1.setAlignmentX(Component.CENTER_ALIGNMENT);
        button2.setAlignmentX(Component.CENTER_ALIGNMENT);

        button2.addActionListener( ActionEvent e -> {
            System.exit( status: 0);
        });
```

- **Class InputOutput**: Mengelola input .txt dari pengguna dan menyimpan solusi dalam output.txt.

```java
public class InputOutput {  2 usages  pixelatedbus
    public static void readInput(String filename, Board board, BlockList blockList) {  1 usage  pixelatedbus
        try {
            BufferedReader reader = new BufferedReader(new FileReader(filename));
            String firstLine = reader.readLine();
            String[] firstLineArray = firstLine.split( regex: " ");
            int row = Integer.parseInt(firstLineArray[0]);
            int col = Integer.parseInt(firstLineArray[1]);

            board.row = row;
            board.col = col;
            board.matrix = new char[row][col];
            for (int i = 0; i < row; i++){
                for (int j = 0; j < col; j++){
                    board.matrix[i][j] = '_';
                }
            }

            int shapeCount = Integer.parseInt(firstLineArray[2]);
            reader.readLine(); //SKIP LINE (CHECK IF EQUAL TO DEFAULT, IF NOT, PRINT UNABLE)
            String currentLine = reader.readLine();

            for (int i = 0; i < shapeCount; i++) {
                if (currentLine == null) {
                    break;
                }
                char currentChar = currentLine.charAt(currentLine.length() - 1);
                ArrayList<String> currentBlock = new ArrayList<>();
                char testChar = currentChar;
                while (testChar == currentChar) {
                    currentBlock.add(currentLine);
                    currentLine = reader.readLine();
                    if (currentLine == null) {
                        break;
```

```
                }
                System.out.println(currentLine);
                testChar = currentLine.charAt(currentLine.length() - 1);
            }
            blockList.setBlocks(new Block(currentBlock));
        }
        reader.close();

    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}

public static void saveBoard(Board board) {  1 usage  pixelatedbus
    String filename = "../test/output.txt";
    try {
        FileWriter writer = new FileWriter(filename);
        for (int i = 0; i < board.row; i++) {
            for (int j = 0; j < board.col; j++) {
                writer.write(board.matrix[i][j]);
            }
            writer.write( str: "\n");
        }
        writer.close();
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}

}
```

Hasil Program

MAIN MENU

TEST CASE

1.

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

2.

```
4 4 5
DEFAULT
A
AA
B
BB
C
CC
DDDD
E
EE
```

3.

```
4 4 5
DEFAULT
A
AA
AAA
B
BB
BBB
C
CC
CCC
DD
E
EE
```

4.

```
6 6 8
DEFAULT
A
AA
AAA
B
BB
BBB
C
CC
CCC
DD
DDD
E
EE
F
FFFF
G
HHHH
```

5.

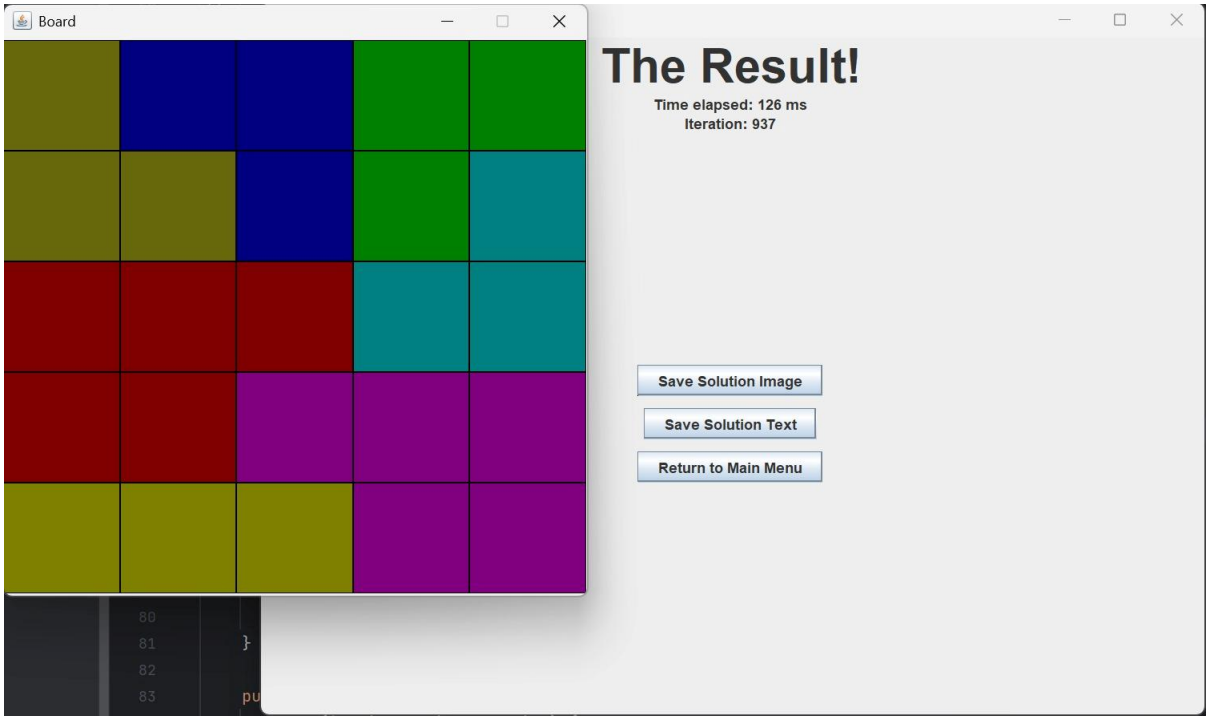
```
4 6 7
DEFAULT
A
AA
AAA
B
BB
B
CC
D
DD
E
EE
 E
FF
GGG
```

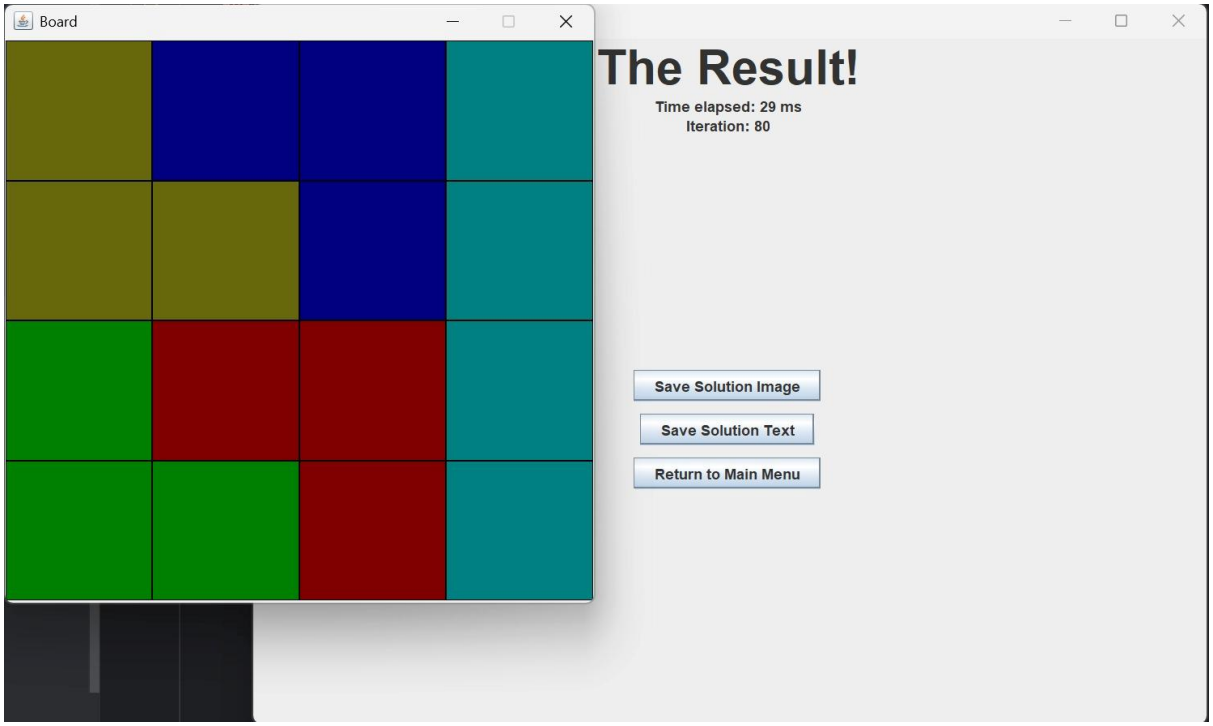
6.

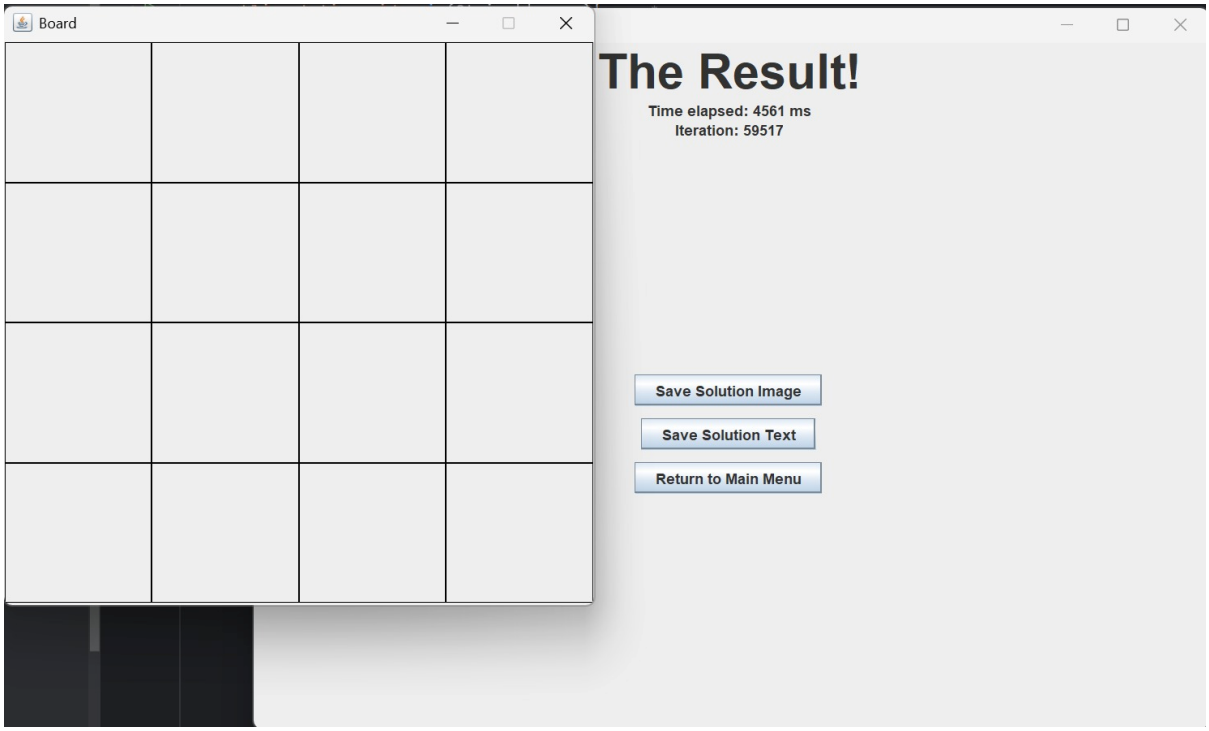
```
4 5 5
DEFAULT
AA
A
AA
B
BBB
B
C
D
D
DD
E
EEEE
```

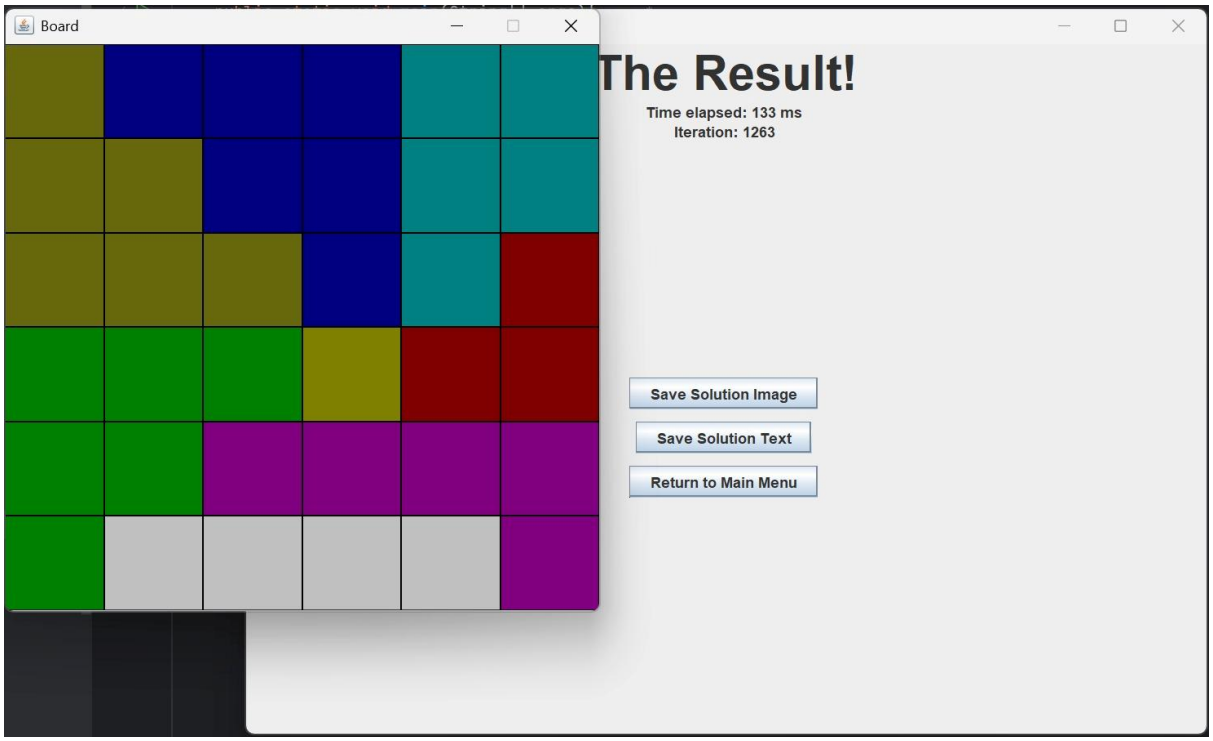
7.

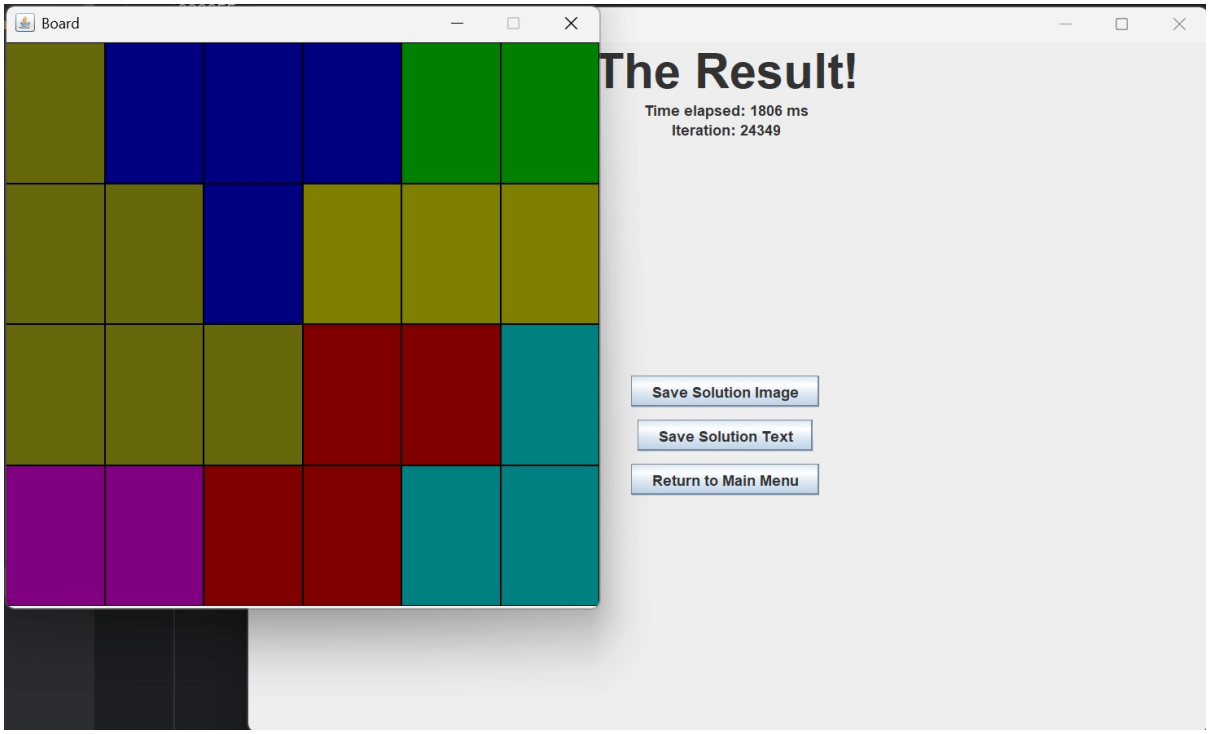
```
3 4 1
DEFAULT
AAAA
AAAA
AAAA
```

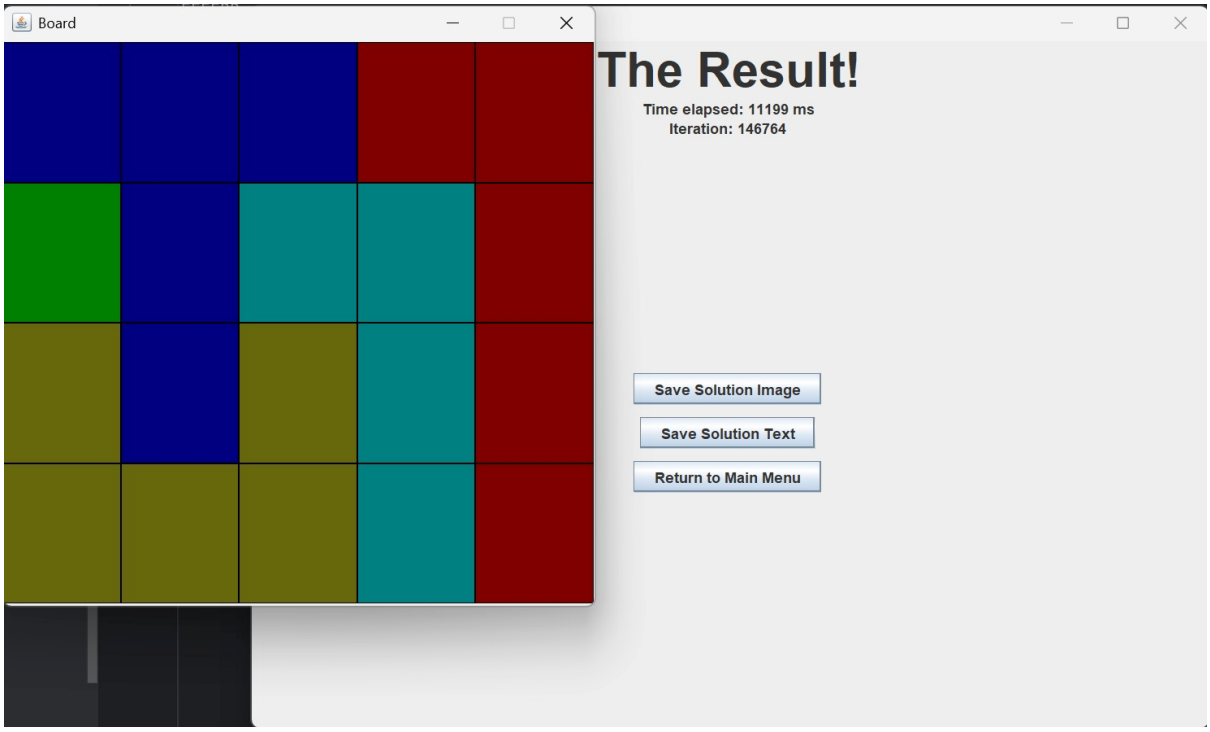

Board
The Result!
Time elapsed: 126 ms
Iteration: 937
Save Solution Image
Save Solution Text
Return to Main Menu

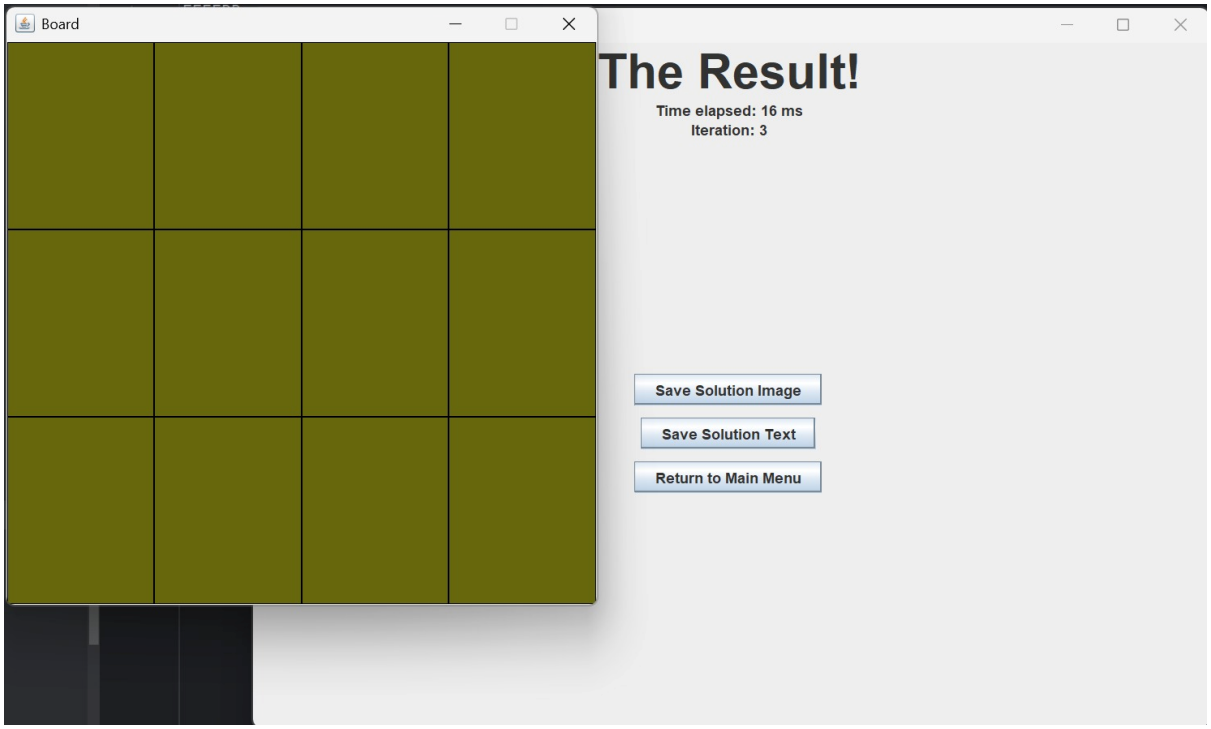

Board
The Result!
Time elapsed: 29 ms
Iteration: 80
Save Solution Image
Save Solution Text
Return to Main Menu


Board
The Result!
Time elapsed: 4561 ms
Iteration: 59517
Save Solution Image
Save Solution Text
Return to Main Menu


Board
The Result!
Time elapsed: 133 ms
Iteration: 1263
Save Solution Image
Save Solution Text
Return to Main Menu


Board
The Result!
Time elapsed: 1806 ms
Iteration: 24349
Save Solution Image
Save Solution Text
Return to Main Menu


Board
The Result!
Time elapsed: 11199 ms
Iteration: 146764
Save Solution Image
Save Solution Text
Return to Main Menu



LAMPIRAN

Link Github: https://github.com/pixelatedbus/Tucil1_13523084/

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 *Graphical User Interface*	V	
6	Program dapat menyimpan solusi dalam bentuk file gambar	V	
7	Program dapat menyelsaikan kasus konfigurasi *custom*		V
8	Program dapat menyelesaikan kasus konfigurasi Piramida (3D)		V
9	Program dibuat oleh saya sendiri	V	