Tugas Kecil 1 IF 2211

Strategi Algoritma: Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force

Dibuat Oleh:
William Gerald Briandelo - 13222061

Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Bandung
2025

Pseudocode:

```
PROGRAM IQPuzzlerPro

// Deklarasi variabel global
N, M : integer                    // Ukuran board (baris dan kolom)
board : array[0..N-1][0..M-1] of char   // Matriks board, diinisialisasi dengan '.'
pieces : list of Piece            // Kumpulan objek Piece
iterationCount : long integer     // Penghitung jumlah iterasi

Procedure MAIN()
    T : string ← input("Masukkan nama file test case: ")
    CALL readInput(T)

    startTime : long integer ← current time in ms
    solved : boolean ← solve(0)
    endTime : long integer ← current time in ms

    IF solved THEN
        print "Solusi ditemukan:"
        CALL printBoard()
    ELSE
        print "Tidak ada solusi yang ditemukan."
    END IF

    print "Waktu pencarian: " + (endTime - startTime) + " ms"
    print "Banyak kasus yang ditinjau: " + iterationCount

    resp : string ← input("Apakah anda ingin menyimpan solusi? (ya/tidak): ")
    IF (resp equals "ya") THEN
        CALL saveSolution("solusi_" + T)
        print "Solusi telah disimpan ke solusi_" + T
    END IF
End Procedure

Procedure readInput(filename: string)
    Open file dengan nama filename untuk dibaca
    header : string ← readLine(file)
    parts : array of string ← split(header, whitespace)
    N ← convert(parts[0]) ke integer
    M ← convert(parts[1]) ke integer

    board ← 2D array of char berukuran N x M
    FOR i dari 0 sampai N-1 DO
        FOR j dari 0 sampai M-1 DO
            board[i][j] ← '.'
        END FOR
    END FOR
```

```
    readLine(file)  // Lewati baris tambahan jika ada
    allLines : list of string ← []
    WHILE (line : string = readLine(file)) tidak NULL DO
        IF line tidak kosong THEN
            Append trim(line) ke allLines
        END IF
    END WHILE
    Close file

    i : integer ← 0
    WHILE i < length(allLines) DO
        current : string ← allLines[i]
        letter : char ← karakter pertama dari current
        shapeLines : list of string ← [current]
        i ← i + 1
        WHILE i < length(allLines) AND (karakter pertama dari allLines[i] = letter) DO
            Append allLines[i] ke shapeLines
            i ← i + 1
        END WHILE

        maxCols : integer ← 0
        FOR setiap s dalam shapeLines DO
            IF length(s) > maxCols THEN
                maxCols ← length(s)
            END IF
        END FOR
        rows : integer ← length(shapeLines)
        shape : array[0..rows-1][0..maxCols-1] of char

        FOR r dari 0 sampai rows-1 DO
            s : string ← shapeLines[r]
            FOR c dari 0 sampai maxCols-1 DO
                IF c < length(s) THEN
                    shape[r][c] ← s[c]
                ELSE
                    shape[r][c] ← '.'
                END IF
            END FOR
        END FOR

        Buat objek Piece(letter, shape) dan Append ke pieces
    END WHILE
End Procedure

Function solve(index: integer) → boolean
    IF index = length(pieces) THEN
        → isBoardFull()
    END IF

    currentPiece : Piece ← pieces[index]
```

```
    forms : list of 2D array of char ← currentPiece.getTransformations()

    FOR setiap shape dalam forms DO
        FOR r dari 0 sampai N-1 DO
            FOR c dari 0 sampai M-1 DO
                iterationCount ← iterationCount + 1
                IF canPlace(shape, r, c) THEN
                    CALL place(shape, r, c, currentPiece.letter)
                    IF solve(index + 1) → true THEN
                        → true
                    END IF
                    CALL remove(shape, r, c)
                END IF
            END FOR
        END FOR
    END FOR
    → false
End Function

Function isBoardFull() → boolean
    FOR i dari 0 sampai N-1 DO
        FOR j dari 0 sampai M-1 DO
            IF board[i][j] = '.' THEN
                → false
            END IF
        END FOR
    END FOR
    → true
End Function

Function canPlace(shape: array of char, r: integer, c: integer) → boolean
    rows : integer ← jumlah baris dari shape
    cols : integer ← jumlah kolom dari shape
    IF (r + rows > N) OR (c + cols > M) THEN
        → false
    END IF
    FOR i dari 0 sampai rows-1 DO
        FOR j dari 0 sampai cols-1 DO
            IF shape[i][j] ≠ '.' AND board[r + i][c + j] ≠ '.' THEN
                → false
            END IF
        END FOR
    END FOR
    → true
End Function

Procedure place(shape: array of char, r: integer, c: integer, letter: char)
    rows : integer ← jumlah baris dari shape
    cols : integer ← jumlah kolom dari shape
    FOR i dari 0 sampai rows-1 DO
```

```
        FOR j dari 0 sampai cols-1 DO
            IF shape[i][j] ≠ '.' THEN
                board[r + i][c + j] ← letter
            END IF
        END FOR
    END FOR
End Procedure

Procedure remove(shape: array of char, r: integer, c: integer)
    rows : integer ← jumlah baris dari shape
    cols : integer ← jumlah kolom dari shape
    FOR i dari 0 sampai rows-1 DO
        FOR j dari 0 sampai cols-1 DO
            IF shape[i][j] ≠ '.' THEN
                board[r + i][c + j] ← '.'
            END IF
        END FOR
    END FOR
End Procedure

Procedure printBoard()
    FOR i dari 0 sampai N-1 DO
        line : string ← ""
        FOR j dari 0 sampai M-1 DO
            line ← line + board[i][j]
        END FOR
        print line
    END FOR
End Procedure

Procedure saveSolution(filename: string)
    Open file dengan nama filename untuk ditulis
    FOR i dari 0 sampai N-1 DO
        line : string ← ""
        FOR j dari 0 sampai M-1 DO
            line ← line + board[i][j]
        END FOR
        Tulis line ke file
    END FOR
    Close file
End Procedure

Class Piece
    Attributes:
        letter : char               // Karakter identitas piece
        shape : array of char       // 2D array yang merepresentasikan bentuk piece

    Constructor(letter: char, shape: array of char)
        this.letter ← letter
        this.shape ← shape
```

```
    End Constructor

    Function getTransformations() → list of 2D array of char
        list : list of 2D array of char ← []
        Append shape ke list                    // Bentuk asli
        r90 : 2D array of char ← rotate(shape)
        Append r90 ke list
        r180 : 2D array of char ← rotate(r90)
        Append r180 ke list
        r270 : 2D array of char ← rotate(r180)
        Append r270 ke list
        mirrorOriginal : 2D array of char ← mirror(shape)
        Append mirrorOriginal ke list
        → list
    End Function

    Function rotate(mat: array of char) → array of char
        // Rotasi 90° searah jarum jam
        rows : integer ← jumlah baris dari mat
        cols : integer ← jumlah kolom dari mat
        rotated : 2D array of char dengan ukuran [cols][rows]
        FOR i dari 0 sampai rows-1 DO
            FOR j dari 0 sampai cols-1 DO
                rotated[j][rows - 1 - i] ← mat[i][j]
            END FOR
        END FOR
        → rotated
    End Function

    Function mirror(mat: array of char) → array of char
        // Mirror horizontal (cermin secara horizontal)
        rows : integer ← jumlah baris dari mat
        cols : integer ← jumlah kolom dari mat
        mirrored : 2D array of char dengan ukuran [rows][cols]
        FOR i dari 0 sampai rows-1 DO
            FOR j dari 0 sampai cols-1 DO
                mirrored[i][cols - 1 - j] ← mat[i][j]
            END FOR
        END FOR
        → mirrored
    End Function
End Class

END PROGRAM
```

Algoritma yang digunakan menggunakan metode brute-force, dengan pendekatan back-tracking. Awalnya, program akan membaca input dan menyiapkan *board* (inisialiasasi matrix kosong sesuai dengan input) untuk menempatkan semua *piece* yang diberikan. Dalam setiap langkah yang dijalankan, program akan menempatkan *piece* (dengan semua

kemungkinan transformasinya) pada *board*, jika *piece* berhasil ditempatkan pada *board* (jika dan hanya jika posisi *piece* valid, yaitu tidak ada *piece* yang tumpang tindih dan juga tidak keluar *board*), maka program akan melanjutkan langkahnya secara rekursif. Namun, jika *piece* tidak maka program akan kembali ke langkah sebelumnya dan mencoba langkah alternatif. Program akan terus berjalan hingga *board* berhasil terisi penuh dengan semua *piece* dipakai ataupun ketika semua kemungkinan langkah telah dijalankan.

Source Code:

```
import java.io.*;
import java.util.*;

public class IQPuzzlerPro {
    static int N, M;
    static char[][] board;
    static ArrayList<Piece> pieces = new ArrayList<>();
    static long iterationCount = 0;

    public static void main(String[] args) throws Exception {
        Scanner sc = new Scanner(System.in);
        System.out.print("Masukkan nama file test case: ");
        String filename = sc.nextLine();
        readInput(filename);

        long start = System.currentTimeMillis();
        boolean solved = solve(0);
        long end = System.currentTimeMillis();

        if (solved) {
            System.out.println("\nSolusi ditemukan:");
            printBoard();
        } else {
            System.out.println("\nTidak ada solusi yang ditemukan.");
        }
        System.out.println("Waktu pencarian: " + (end - start) + " ms");
        System.out.println("Banyak kasus yang ditinjau: " +
iterationCount);

        System.out.print("Apakah anda ingin menyimpan solusi?
(ya/tidak): ");
```

```
        String resp = sc.nextLine();
        if(resp.equalsIgnoreCase("ya")) {
            saveSolution("solusi_"+filename);
            System.out.println("Solusi telah disimpan ke
solusi_"+filename);
        }
    }

    static void readInput(String filename) throws Exception {
        BufferedReader br = new BufferedReader(new
FileReader(filename));
        String header = br.readLine();
        String[] parts = header.trim().split("\\s+");
        N = Integer.parseInt(parts[0]);
        M = Integer.parseInt(parts[1]);

        // Inisialisasi board sel kosong dengan '.'
        board = new char[N][M];
        for (int i = 0; i < N; i++) {
            Arrays.fill(board[i], '.');
        }

        br.readLine();

        ArrayList<String> allLines = new ArrayList<>();
        String line;
        while ((line = br.readLine()) != null) {
            if (!line.trim().isEmpty()) {
                allLines.add(line.trim());
            }
        }
        br.close();

        int i = 0;
        while (i < allLines.size()) {
            String current = allLines.get(i);
            char letter = current.charAt(0);
            ArrayList<String> shapeLines = new ArrayList<>();
            shapeLines.add(current);
```

```
            i++;
            while (i < allLines.size() && allLines.get(i).charAt(0) ==
letter) {
                shapeLines.add(allLines.get(i));
                i++;
            }

            int maxCols = 0;
            for (String s : shapeLines) {
                if (s.length() > maxCols) maxCols = s.length();
            }
            int rows = shapeLines.size();
            char[][] shape = new char[rows][maxCols];
            for (int r = 0; r < rows; r++) {
                String s = shapeLines.get(r);
                for (int c = 0; c < maxCols; c++) {
                    if (c < s.length()) {
                        shape[r][c] = s.charAt(c);
                    } else {
                        shape[r][c] = '.';
                    }
                }
            }
            pieces.add(new Piece(letter, shape));
        }
    }

    // Fungsi rekursif backtracking
    // Basis: jika semua piece telah ditempatkan, periksa apakah board
telah terisi penuh
    static boolean solve(int index) {
        if (index == pieces.size()) {
            return isBoardFull();
        }
        Piece current = pieces.get(index);
        ArrayList<char[][]> forms = current.getTransformations();
        for (char[][] shape : forms) {
            for (int r = 0; r < N; r++) {
                for (int c = 0; c < M; c++) {
```

```
                    iterationCount++;
                    if (canPlace(shape, r, c)) {
                        place(shape, r, c, current.letter);
                        if (solve(index + 1)) {
                            return true;
                        }
                        remove(shape, r, c);
                    }
                }
            }
        }
        return false;
    }

    static boolean isBoardFull() {
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < M; j++) {
                if (board[i][j] == '.') {
                    return false;
                }
            }
        }
        return true;
    }

    static boolean canPlace(char[][] shape, int r, int c) {
        int rows = shape.length;
        int cols = shape[0].length;
        if (r + rows > N || c + cols > M) return false;
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                if (shape[i][j] != '.' && board[r + i][c + j] != '.') {
                    return false;
                }
            }
        }
        return true;
    }
```

```
    static void place(char[][] shape, int r, int c, char letter) {
        int rows = shape.length;
        int cols = shape[0].length;
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                if (shape[i][j] != '.') {
                    board[r + i][c + j] = letter;
                }
            }
        }
    }

    static void remove(char[][] shape, int r, int c) {
        int rows = shape.length;
        int cols = shape[0].length;
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                if (shape[i][j] != '.') {
                    board[r + i][c + j] = '.';
                }
            }
        }
    }

    static void printBoard() {
        for (int i = 0; i < N; i++) {
            StringBuilder sb = new StringBuilder();
            for (int j = 0; j < M; j++) {
                sb.append(board[i][j]);
            }
            System.out.println(sb.toString());
        }
    }

    static void saveSolution(String filename) throws Exception {
        PrintWriter pw = new PrintWriter(new FileWriter(filename));
        for (int i = 0; i < N; i++) {
            StringBuilder sb = new StringBuilder();
            for (int j = 0; j < M; j++) {
```

```
                sb.append(board[i][j]);
            }
            pw.println(sb.toString());
        }
        pw.close();
    }

    // Kelas untuk piece blok puzzle
    static class Piece {
        char letter;
        char[][] shape;

        Piece(char letter, char[][] shape) {
            this.letter = letter;
            this.shape = shape;
        }

        // Transformasi blok Original, Rotasi 90°, Rotasi 180°, Rotasi
270°, dan Mirror Horizontal
        ArrayList<char[][]> getTransformations() {
            ArrayList<char[][]> list = new ArrayList<>();
            list.add(shape);

            char[][] r90 = rotate(shape);
            list.add(r90);

            char[][] r180 = rotate(r90);
            list.add(r180);

            char[][] r270 = rotate(r180);
            list.add(r270);

            char[][] mirrorOriginal = mirror(shape);
            list.add(mirrorOriginal);
            return list;
        }

        // Rotasi 90 derajat searah jarum jam
        char[][] rotate(char[][] mat) {
```

```
            int rows = mat.length;
            int cols = mat[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] = mat[i][j];
                }
            }
            return rotated;
        }

        // Mirror horizontal
        char[][] mirror(char[][] mat) {
            int rows = mat.length;
            int cols = mat[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] = mat[i][j];
                }
            }
            return mirrored;
        }
    }
}
```

Input dan Output:
Input 1:

```
5 5 8
DEFAULT
A
AA
B
BB
C
CC
D
DD
EE
EE
E
FF
FF
F
GGG
```

Output 1:

```
Masukkan nama file test case: input1.txt

Solusi ditemukan:
AGGGC
AABCC
EEBBF
EEDFF
EDDFF
Waktu pencarian: 75 ms
Banyak kasus yang ditinjau: 1097882
Apakah anda ingin menyimpan solusi? (ya/tidak): ya
Solusi telah disimpan ke solusi_input1.txt
```

Input 2:

```
4 4 6
DEFAULT
A
AA
BB
C
CC
D
DD
E
EE
FF
```

Output 2:

```
Masukkan nama file test case: input2.txt

Solusi ditemukan:
ABBE
AAEE
CDDF
CCDF
Waktu pencarian: 1 ms
Banyak kasus yang ditinjau: 1893
Apakah anda ingin menyimpan solusi? (ya/tidak): ya
Solusi telah disimpan ke solusi_input2.txt
```

Input 3:

```
5 5 8
DEFAULT
A
A
B
BB
C
CC
D
DD
EE
EE
E
FF
FF
F
GGG
```

Output 3:

```
Masukkan nama file test case: input3.txt

Tidak ada solusi yang ditemukan.
Waktu pencarian: 108057 ms
Banyak kasus yang ditinjau: 4884214625
Apakah anda ingin menyimpan solusi? (ya/tidak): ya
Solusi telah disimpan ke solusi_input3.txt
```

Input 4:

```
4 4 2
DEFAULT
A
AA
BB
```

Output 4:

```
Masukkan nama file test case: input4.txt

Tidak ada solusi yang ditemukan.
Waktu pencarian: 0 ms
Banyak kasus yang ditinjau: 3680
Apakah anda ingin menyimpan solusi? (ya/tidak): ya
Solusi telah disimpan ke solusi_input4.txt
```

Input 5:

```
5 5 9
DEFAULT
A
AA
B
BB
C
CC
D
DD
EE
EE
E
FF
FF
F
GGG
HH
```

Output 5:

```
Masukkan nama file test case: input5.txt

Tidak ada solusi yang ditemukan.
Waktu pencarian: 29684 ms
Banyak kasus yang ditinjau: 1535650125
Apakah anda ingin menyimpan solusi? (ya/tidak): ya
Solusi telah disimpan ke solusi_input5.txt
```

In[ut 6:

```
5 5 5
DEFAULT
AAAAA
BBBBB
CCCCC
DDDDD
EEEEE
```

Output 6:

```
Masukkan nama file test case: input6.txt

Solusi ditemukan:
AAAAA
BBBBB
CCCCC
DDDDD
EEEEE
Waktu pencarian: 0 ms
Banyak kasus yang ditinjau: 55
Apakah anda ingin menyimpan solusi? (ya/tidak): ya
Solusi telah disimpan ke solusi_input6.txt
```

Input 7:

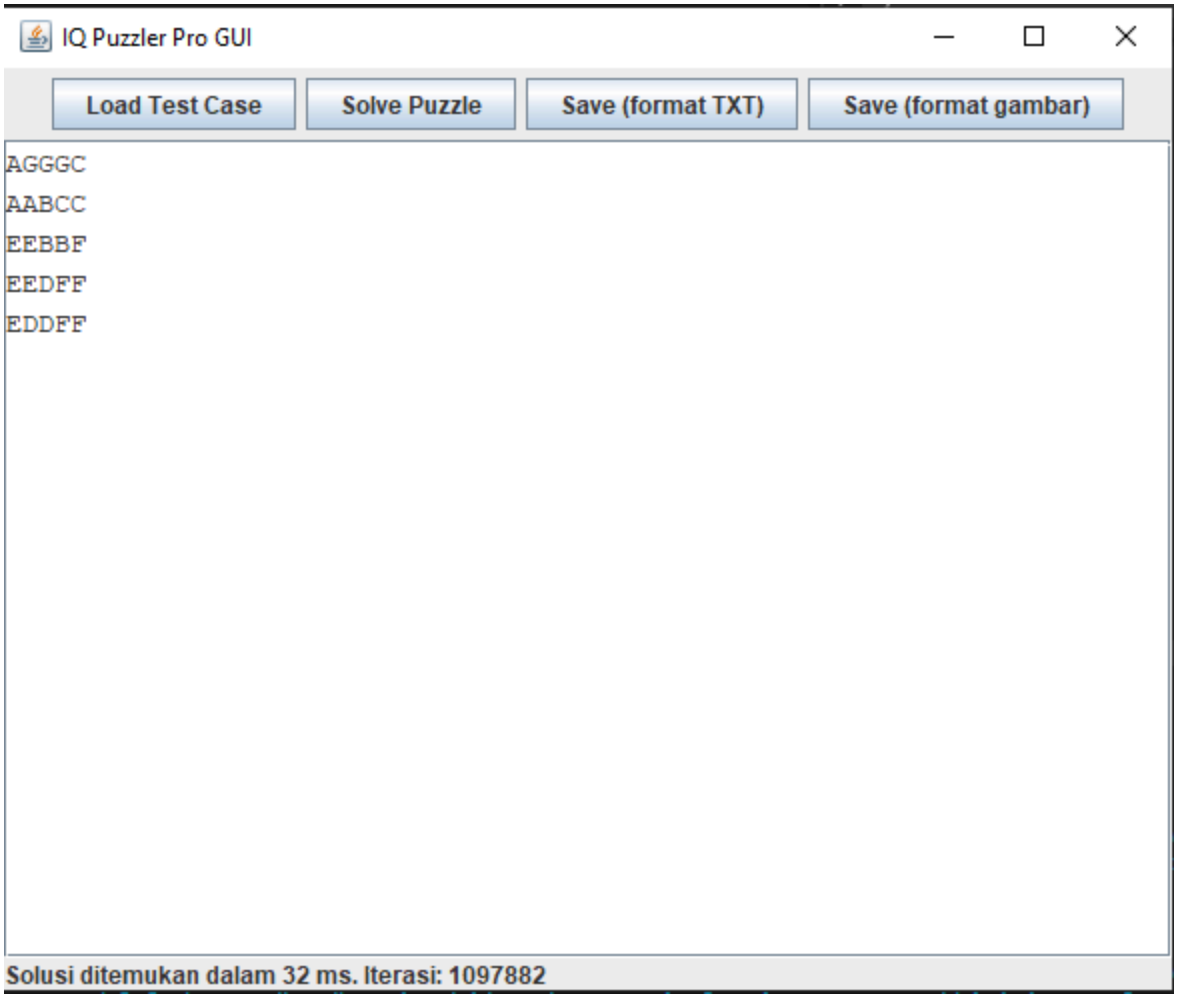
input7.txt

```
4 5 5
DEFAULT
AA
AA
A
BB
BB
CC
C
DD
EEEEE
E
```

Output 7:

```
Masukkan nama file test case: input7.txt

Solusi ditemukan:
EEEEE
AABBE
AABBC
ADDCC
Waktu pencarian: 158 ms
Banyak kasus yang ditinjau: 1995466
Apakah anda ingin menyimpan solusi? (ya/tidak): ya
Solusi telah disimpan ke solusi_input7.txt
```

GUI:



Source Code GUI:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.image.BufferedImage;
import java.io.*;
import java.util.HashMap;
import java.util.Random;
import javax.imageio.ImageIO;

public class IQPuzzlerProGUI extends JFrame {
    private JTextArea boardArea;
    private JButton loadFileButton, solveButton, saveSolutionButton,
saveImageButton;
```

```
    private JLabel statusLabel;
    private File testCaseFile;

    public IQPuzzlerProGUI() {
        setTitle("IQ Puzzler Pro GUI");
        setSize(600, 500);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        initComponents();
    }

    private void initComponents() {
        JPanel topPanel = new JPanel();
        loadFileButton = new JButton("Load Test Case");
        solveButton = new JButton("Solve Puzzle");
        saveSolutionButton = new JButton("Save (format TXT)");
        saveImageButton = new JButton("Save (format gambar)");

        solveButton.setEnabled(false);
        saveSolutionButton.setEnabled(false);
        saveImageButton.setEnabled(false);

        topPanel.add(loadFileButton);
        topPanel.add(solveButton);
        topPanel.add(saveSolutionButton);
        topPanel.add(saveImageButton);
        add(topPanel, BorderLayout.NORTH);

        boardArea = new JTextArea();
        boardArea.setEditable(false);
        boardArea.setFont(new Font("Monospaced", Font.PLAIN, 14));
        JScrollPane scrollPane = new JScrollPane(boardArea);
        add(scrollPane, BorderLayout.CENTER);

        statusLabel = new JLabel("Status: Menunggu file test case");
        add(statusLabel, BorderLayout.SOUTH);

        loadFileButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
```

```
                loadTestCase();
            }
        });

        solveButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                solvePuzzle();
            }
        });

        saveSolutionButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                saveSolution();
            }
        });

        saveImageButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                saveSolutionAsImage();
            }
        });
    }

    private void loadTestCase() {
        JFileChooser fileChooser = new JFileChooser();
        int result = fileChooser.showOpenDialog(this);
        if(result == JFileChooser.APPROVE_OPTION) {
            testCaseFile = fileChooser.getSelectedFile();
            statusLabel.setText("File dipilih: " +
testCaseFile.getName());
            solveButton.setEnabled(true);
        }
    }

    private void solvePuzzle() {
        if(testCaseFile == null) {
            JOptionPane.showMessageDialog(this, "Belum ada file test
case yang dipilih.");
            return;
```

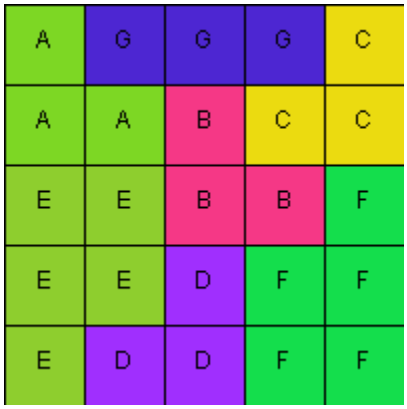
```
        }

        loadFileButton.setEnabled(false);
        solveButton.setEnabled(false);
        saveSolutionButton.setEnabled(false);
        saveImageButton.setEnabled(false);
        statusLabel.setText("Mencari solusi");

        SwingWorker<Void, Void> worker = new SwingWorker<Void, Void>() {
            private long timeTaken;
            @Override
            protected Void doInBackground() throws Exception {
                IQPuzzlerPro.readInput(testCaseFile.getAbsolutePath());
                long start = System.currentTimeMillis();
                boolean solved = IQPuzzlerPro.solve(0);
                long end = System.currentTimeMillis();
                timeTaken = end - start;
                if(!solved) {
                    boardArea.setText("Tidak ada solusi yang
ditemukan.");
                } else {
                    StringBuilder sb = new StringBuilder();
                    for (int i = 0; i < IQPuzzlerPro.N; i++) {
                        for (int j = 0; j < IQPuzzlerPro.M; j++) {
                            sb.append(IQPuzzlerPro.board[i][j]);
                        }
                        sb.append("\n");
                    }
                    boardArea.setText(sb.toString());
                }
                return null;
            }

            @Override
            protected void done() {
                statusLabel.setText("Solusi ditemukan dalam " +
timeTaken + " ms. Iterasi: " + IQPuzzlerPro.iterationCount);
                loadFileButton.setEnabled(true);
                solveButton.setEnabled(true);
```

```
                saveSolutionButton.setEnabled(true);
                saveImageButton.setEnabled(true);
            }
        };
        worker.execute();
    }

    private void saveSolution() {
        JFileChooser fileChooser = new JFileChooser();
        int result = fileChooser.showSaveDialog(this);
        if(result == JFileChooser.APPROVE_OPTION) {
            File saveFile = fileChooser.getSelectedFile();
            try {
                PrintWriter pw = new PrintWriter(new
FileWriter(saveFile));
                pw.print(boardArea.getText());
                pw.close();
                JOptionPane.showMessageDialog(this, "Solusi telah
disimpan sebagai file teks.");
            } catch(IOException ex) {
                JOptionPane.showMessageDialog(this, "Error saat
menyimpan solusi: " + ex.getMessage());
            }
        }
    }

    private void saveSolutionAsImage() {
        int cellSize = 40;
        int rows = IQPuzzlerPro.N;
        int cols = IQPuzzlerPro.M;
        int imgWidth = cols * cellSize;
        int imgHeight = rows * cellSize;

        BufferedImage image = new BufferedImage(imgWidth, imgHeight,
BufferedImage.TYPE_INT_RGB);
        Graphics2D g2d = image.createGraphics();

        HashMap<Character, Color> colorMap = new HashMap<>();
        Random rand = new Random();
```

```
        g2d.setColor(Color.WHITE);
        g2d.fillRect(0, 0, imgWidth, imgHeight);

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                char letter = IQPuzzlerPro.board[i][j];
                int x = j * cellSize;
                int y = i * cellSize;
                if (letter != '.') {
                    if (!colorMap.containsKey(letter)) {
                        colorMap.put(letter, getRandomColor(rand));
                    }
                    g2d.setColor(colorMap.get(letter));
                    g2d.fillRect(x, y, cellSize, cellSize);
                    g2d.setColor(Color.BLACK);
                    g2d.drawRect(x, y, cellSize, cellSize);
                    g2d.setColor(Color.BLACK);
                    FontMetrics fm = g2d.getFontMetrics();
                    String s = String.valueOf(letter);
                    int textWidth = fm.stringWidth(s);
                    int textHeight = fm.getAscent();
                    g2d.drawString(s, x + (cellSize - textWidth) / 2, y
+ (cellSize + textHeight) / 2 - 4);
                } else {
                    g2d.setColor(Color.LIGHT_GRAY);
                    g2d.drawRect(x, y, cellSize, cellSize);
                }
            }
        }
        g2d.dispose();

        JFileChooser fileChooser = new JFileChooser();
        int result = fileChooser.showSaveDialog(this);
        if(result == JFileChooser.APPROVE_OPTION) {
            File saveFile = fileChooser.getSelectedFile();
            try {
                ImageIO.write(image, "png", saveFile);
                JOptionPane.showMessageDialog(this, "Solusi telah
```

```
disimpan sebagai gambar.");
            } catch(IOException ex) {
                JOptionPane.showMessageDialog(this, "Error saat
menyimpan gambar: " + ex.getMessage());
            }
        }
    }

    private Color getRandomColor(Random rand) {
        float hue = rand.nextFloat();
        float saturation = 0.7f + rand.nextFloat() * 0.3f;
        float brightness = 0.8f + rand.nextFloat() * 0.2f;
        return Color.getHSBColor(hue, saturation, brightness);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                new IQPuzzlerProGUI().setVisible(true);
            }
        });
    }
}
```

Output Gambar:



Lampiran:
https://github.com/Azekhiel/Tucil1_13222061