Laporan Tugas Kecil 1

IF2211 - Strategi Algoritma

Disusun Oleh:

Reynard Nathanael
13524103

Bab 1: Algoritma Brute Force

1. Penjelasan Algoritma

Pada pengerjaan tugas kecil ini, program dibuat menggunakan algoritma brute force yang memakai konsep exhaustive search. Jadi secara garis besar, program yang telah dibuat akan menempatkan Queen pada 'board' yang telah dibuat sebelumnya. Queen akan ditempatkan satu per satu yang dilakukan secara bertahap (iterasi) pada setiap kolom, jadi misalnya ketika berada di baris pertama maka program akan mencoba menempatkan Queen pada sebuah kolom lalu pindah ke baris berikutnya untuk melakukan hal serupa. Saat baris dan kolom terakhir sudah diisi dengan Queen, maka proses pengecekan (validasi) lokasi penempatan Queen akan dilakukan. Ketika Queen yang sudah ditempatkan mengalami konflik dengan Queen lainnya, maka program akan mencoba untuk memindahkan posisi Queen pada baris terakhir, dan jika seluruh opsi kolom pada pilihan terakhir juga masih mengakibatkan konflik, akan pindah ke konfigurasi sebelumnya.

2. Langkah-langkah Algoritma

Langkah-langkah Algoritma utamanya berada pada file Board.java, dan dimulai dengan langkah sebagai berikut:

1. User Load file .txt untuk menginisialisasi board. Proses penanganan atau pembacaan file diatasi oleh file FileHandlers.java. Karena algoritma yang digunakan pada program merupakan brute-force exhaustive search, maka lebih baik untuk membuat dua grid, jadi ketika algoritma nanti memerlukan proses kembali ke langkah sebelumnya untuk evaluasi posisi, queen yang sudah ditempatkan akan dihapus dan diganti dengan konten atau input pada posisi yang sama yang diambil dari startGrid.

```
    public Board(char[][] inputGrid){
        this.N = inputGrid.length;
        this.startGrid = inputGrid;
        this.iterations = 0;

        //Duplicate startGrid awal ke finalGrid
        this.finalGrid = new char[N][N];
        for(int i = 0; i < N; i++){
            System.arraycopy(startGrid[i], 0, finalGrid[i], 0,
N);
        }
    }
```

2. Ketika User klik tombol 'Solve' maka program akan memanggil fungsi solveExhaustive() yang menerima parameter row (baris) saat ini. Kemudian program akan mulai menempatkan Queen pada board secara utuh dengan iterasi per kolom hingga

seluruh baris sudah terisi oleh satu Queen. Di setiap kolom, program langsung menempatkan Queen (#) tanpa memeriksa validitasnya terlebih dahulu.

```
public boolean solve(boolean visualizeBrute) {
    this.showVisuals = visualizeBrute;
    this.iterations = 0;
    return solveExhaustive(0);
}
```

```
// Algoritma
private boolean solveExhaustive(int row) {
    iterations++;

    // Basis
    if (row == N) {
        return isCorrect();
    }

    // Rekurens
    for (int col = 0; col < N; col++) {
        finalGrid[row][col] = '#';
        printStep(row, col, "Placing (Brute Force)");

        if (solveExhaustive(row + 1)) {
            return true;
        }

        finalGrid[row][col] = startGrid[row][col];
    }

    return false;
}
```

3. Selanjutnya ketika board sudah terisi penuh (row = N) akan masuk ke tahap validasi. Jika penempatan Queen yang telah dilakukan sebelumnya melanggar peraturan penempatan Queen, maka program akan membuang penempatan Queen tersebut dan menggeser posisi Queen pada baris terakhir, sehingga seluruh kemungkinan kombinasi posisi telah diperiksa pada baris tersebut sebelum berpindah atau solusi ditemukan. Proses ini dilakukan oleh isCorrect() yang memeriksa keseluruhan finalGrid. Dimana posisi Queen

pada board akan dicek dengan memanggil checkQueen() untuk memastikan posisinya aman.

```
private boolean isCorrect() {
    for (int i = 0; i < N; i++) {
        int colFound = -1;
        for (int j = 0; j < N; j++) {
            if (finalGrid[i][j] == '#') {
                colFound = j;
                break;
            }
        }

        if (colFound != -1) {
            if (!checkQueen(i, colFound)) {
                return false;
            }
        }
    }

    return true;
}
```

4. Validasi kemudian dilakukan oleh checkQueen() untuk memastikan bahwa penempatan Queen tidak melanggar peraturan yang sudah ada, yaitu tidak boleh berada pada baris dan kolom yang sama dengan Queen lain, tidak boleh berada berdekatan satu sama lain karena harus ada jarak minimal satu tile diagonal, dan setiap area hanya diperbolehkan ada satu Queen.

```
// Validator
private boolean checkQueen(int row, int col) {

    // Cek Kolom
    for (int i = 0; i < row; i++){
        if (finalGrid[i][col] == '#') {
            return false;
        }
    }

    if (row > 0) {
```

```
            // Cek Diagonal Kiri Atas
            if (col > 0 && finalGrid[row-1][col-1] == '#') {
                return false;
            }

            // Cek Diagonal Kanan Atas
             if (col < N - 1 && finalGrid[row-1][col+1] == '#')
{
                return false;
            }
        }

        // Cek Area
        char currentArea = startGrid[row][col];
        for (int r = 0; r < row; r++) {
            for (int k = 0; k < N; k++) {
                 if (finalGrid[r][k] == '#' && startGrid[r][k]
== currentArea) {
                    return false;
                }
            }
        }

        return true;
    }
```

Bab 2: Source Code Program

Karena program menggunakan bahasa Java, maka proses pembuatan GUI dilakukan menggunakan JavaFX. Berikut adalah isi file yang berada pada folder src:

1. **App.java**

```
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;
import java.io.IOException;

public class App extends Application {

    @Override
    public void start(Stage stage) throws IOException {
                                    FXMLLoader    loader    =    new
FXMLLoader(getClass().getResource("Queen.fxml"));
        Parent root = loader.load();

        Scene scene = new Scene(root);
        stage.setTitle("Tucil1 STIMA");
        stage.setScene(scene);
        stage.show();
    }

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

2. **Board.java**

```
public class Board {
    private int N;                  // Ukuran Board
    private char[][] startGrid;     // Grid Area Awal
     private char[][] finalGrid;     // Grid Area Final (Udah ada '#'
sebagai penanda Queen)
    private long iterations;        // Jumlah iterasi yang dilakukan
```

```java
// Visualisasi
private boolean showVisuals;
private BoardListener listener;
private int delay = 50;

// Interface
public interface BoardListener {
    void onStep(char[][] currentGrid);
}

// Konstruktor
public Board(char[][] inputGrid){
    this.N = inputGrid.length;
    this.startGrid = inputGrid;
    this.iterations = 0;

    // Duplicate startGrid awal ke finalGrid
    this.finalGrid = new char[N][N];
    for(int i = 0; i < N; i++){
        System.arraycopy(startGrid[i], 0, finalGrid[i], 0, N);
    }
}

// GUI
public void setListener(BoardListener listener) {
    this.listener = listener;
}

public void setDelay(int delay) {
    this.delay = delay;
}

// Selector
public long getIterations() {
    return iterations;
}

public char[][] getFinalGrid() {
    return finalGrid;
```

```
    }

    // Validator
    private boolean checkQueen(int row, int col) {

        // Cek Kolom
        for (int i = 0; i < row; i++){
            if (finalGrid[i][col] == '#') {
                return false;
            }
        }

        if (row > 0) {
            // Cek Diagonal Kiri Atas
            if (col > 0 && finalGrid[row-1][col-1] == '#') {
                return false;
            }

            // Cek Diagonal Kanan Atas
            if (col < N - 1 && finalGrid[row-1][col+1] == '#') {
                return false;
            }
        }

        // Cek Area
        char currentArea = startGrid[row][col];
        for (int r = 0; r < row; r++) {
            for (int k = 0; k < N; k++) {
                    if (finalGrid[r][k] == '#' && startGrid[r][k] ==
currentArea) {
                        return false;
                    }
            }
        }

        return true;
    }

    private boolean isCorrect() {
        for (int i = 0; i < N; i++) {
```

```
            int colFound = -1;
            for (int j = 0; j < N; j++) {
                if (finalGrid[i][j] == '#') {
                    colFound = j;
                    break;
                }
            }

            if (colFound != -1) {
                if (!checkQueen(i, colFound)) {
                    return false;
                }
            }
        }

        return true;
    }

    // Algoritma
    private boolean solveExhaustive(int row) {
        iterations++;

        // Basis
        if (row == N) {
            return isCorrect();
        }

        // Rekurens
        for (int col = 0; col < N; col++) {
            finalGrid[row][col] = '#';
            printStep(row, col, "Placing (Brute Force)");

            if (solveExhaustive(row + 1)) {
                return true;
            }

            finalGrid[row][col] = startGrid[row][col];
        }

        return false;
```

```
    }

    public boolean solve(boolean visualizeBrute) {
        this.showVisuals = visualizeBrute;
        this.iterations = 0;
        return solveExhaustive(0);
    }

    // Helper
    public void printStep(int curRow, int curCol, String status) {
        if (listener != null) {
            char[][] snapshot = new char[N][N];

            for(int i=0; i<N; i++) {
                snapshot[i] = finalGrid[i].clone();
            }
            listener.onStep(snapshot);
            try {
                if (delay > 0) {
                    Thread.sleep(delay);
                }
            } catch (InterruptedException e) {

            }
            return;
        }

        if (showVisuals) {
            try { Thread.sleep(200); } catch (Exception e) {}

            System.out.println("\n--- Step-" + iterations + " ---");
              System.out.println(status + " at (" + curRow + ", " +
curCol + ")");
            for (int i = 0; i < N; i++) {
                for (int j = 0; j < N; j++) {
                      System.out.print(finalGrid[i][j] == '#' ? "[#]"
: " " + startGrid[i][j] + " ");
                }
                System.out.println();
            }
```

```
        }
    }
}
```

3. **FileHandler.java**

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.util.ArrayList;
import java.util.List;

public class FileHandler {
    public static char[][] readFiles(File file) throws Exception{

        List<String> lines = new ArrayList<>();

        // Read Files
                try (BufferedReader buf = new BufferedReader(new
FileReader(file))) {
            String line;
            while ((line = buf.readLine()) != null) {
                line = line.trim();

                // Ignore Blank
                if (!line.isBlank()){
                    lines.add(line);
                }
            }
        }

        if(lines.isEmpty()){
            throw new Exception("Empty File!");
        }

        int N = lines.size();
        char[][] board = new char[N][N];

        for (int i = 0; i < N; i ++){
```

```
            String curLine = lines.get(i);

            if (curLine.length() != N){
                throw new Exception(
                      "The input on file .txt is not valid! The board
must be square.\n" +
                           "On line-" + (i+1) + " the length is " +
curLine.length() +
                       ", it should be " + N
                );
            }

            if (!curLine.matches("[A-Z]+")){
                 throw new Exception("The input on file .txt contains
characters non-letters or lower case letters!");
            }

            board[i] = curLine.toCharArray();
        }

        return board;
    }
}
```

4. **GuiController.java**

```
import javafx.application.Platform;
import javafx.embed.swing.SwingFXUtils;
import javafx.fxml.FXML;
import javafx.scene.SnapshotParameters;
import javafx.scene.control.*;
import javafx.scene.image.WritableImage;
import javafx.scene.layout.*;
import javafx.scene.paint.Color;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.scene.text.Text;
import javafx.stage.FileChooser;
import javax.imageio.ImageIO;
```

```java
import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;

public class GuiController implements Board.BoardListener {

    @FXML private MenuItem btnImg;       // Save as Image
    @FXML private MenuItem btnLoad;      // Load File
    @FXML private MenuItem btnSave;      // Save as Text
    @FXML private Button btnSolve;       // Solve

    @FXML private Label context;         // Status
    @FXML private Label iteration;       // Iterations
    @FXML private Label timer;           // Time

    @FXML private CheckBox liveUpdate;   // Live Update
    @FXML private CheckBox showLetters; // Letters
    @FXML private GridPane queenGrid;    // Board

    private Board board;
    private char[][] startGrid;

    private final int TILE_SIZE = 50;
    private final String[] tileColors = {
        "#f5583d", // A
        "#fdad17", // B
        "#f8d409", // C
        "#7cee7c", // D
        "#6fb3ce", // E
        "#eeb1ee", // F
        "#d85e9b", // G
        "#f7eb7e", // H
        "#04ffff", // I
        "#ffadb9", // J
        "#71facc", // K
        "#b3e4eb", // L
        "#ee88ee", // M
        "#f8ceaa", // N
        "#e0e0f1", // O
```

```
        "#04f196", // P
        "#f5dbae", // Q
        "#c49111", // R
        "#f57d52", // S
        "#c4c0c0", // T
        "#91f391", // U
        "#9edada", // V
        "#dda0aa", // W
        "#eed4a4", // X
        "#94bbc9", // Y
        "#b8b5b5"  // Z
    };

    @FXML
    public void initialize() {
        btnSolve.setDisable(true);
        btnImg.setDisable(true);
        btnSave.setDisable(true);

        // Default text
        context.setText("Please Load File");
        iteration.setText("0");
        timer.setText("0 ms");

        // Menu Items
        btnLoad.setOnAction(e -> handleLoad());
        btnImg.setOnAction(e -> handleSaveImage());
        btnSave.setOnAction(e -> handleSaveText());

        // Solve
        btnSolve.setOnAction(e -> handleSolve());

            showLetters.selectedProperty().addListener((obs, oldVal,
newVal) -> {
            if (board != null && board.getFinalGrid() != null) {
                updateView(board.getFinalGrid());
            } else if (startGrid != null) {
                                            char[][]   empty   =   new
char[startGrid.length][startGrid.length];
                updateView(empty);
```

```
            }
        });
    }

    // Event Handlers
    private void handleLoad() {
        FileChooser fc = new FileChooser();
        fc.setTitle("Load file .txt");
                                    fc.getExtensionFilters().add(new
FileChooser.ExtensionFilter("Text Files", "*.txt"));
        File file = fc.showOpenDialog(null);

        if (file != null) {
            try {
                startGrid = FileHandler.readFiles(file);

                btnSolve.setDisable(false);
                btnSave.setDisable(true);
                btnImg.setDisable(true);

                // Reset Label
                timer.setText("0 ms");
                iteration.setText("0");
                context.setText("Ready to Solve");

                                        char[][]  emptyGrid  =  new
char[startGrid.length][startGrid.length];
                updateView(emptyGrid);

            } catch (IOException e) {
                showAlert("Failed to load File: ", e.getMessage());

            } catch (Exception e) {
                        showAlert("Error", "There is an error: " +
e.getMessage());
            }
        }
    }

    private void handleSaveImage() {
```

```
        FileChooser fc = new FileChooser();
         fileLocation(fc, "Save Image (.png)", "PNG Image", "*.png",
"solution.png");
        File file = fc.showSaveDialog(null);

        if (file != null) {
            try {
                        WritableImage image = queenGrid.snapshot(new
SnapshotParameters(), null);
                ImageIO.write(SwingFXUtils.fromFXImage(image, null),
"png", file);
                showAlert("Success", "Image successfully saved!");
            } catch (IOException ex) {
                    showAlert("Error", "Failed to save image: " +
ex.getMessage());
            }
        }
    }

    private void handleSaveText() {
        if (board == null || board.getFinalGrid() == null) {
            showAlert("Error", "No Solution to save!");
            return;
        }

        FileChooser fileChooser = new FileChooser();
            fileLocation(fileChooser, "Save Solution (.txt)", "Text
Files", "*.txt", "solution.txt");

        File file = fileChooser.showSaveDialog(null);

        if (file != null) {
            try (PrintWriter writer = new PrintWriter(file)) {
                char[][] grid = board.getFinalGrid();
                int N = grid.length;

                for (int i = 0; i < N; i++) {
                    for (int j = 0; j < N; j++) {
                        writer.print(grid[i][j]);
                    }
```

```
                    writer.println();
                }

                showAlert("Success", "File successfully saved in:\n"
+ file.getAbsolutePath());

            } catch (Exception ex) {
                    showAlert("Error", "Failed to save file: " +
ex.getMessage());
            }
        }
    }

    private void handleSolve() {
        if (startGrid == null) {
            return;
        }

        btnSolve.setDisable(true);
        btnLoad.setDisable(true);
        context.setText("Solving...");

        board = new Board(startGrid);

        if (liveUpdate.isSelected()) {
            board.setListener(this);
            board.setDelay(1);
        } else {
            board.setListener(null);
            board.setDelay(0);
        }

        new Thread(() -> {
            long startTime = System.currentTimeMillis();
            boolean success = board.solve(false);
            long endTime = System.currentTimeMillis();
            long totalTime = endTime - startTime;
            long totalIter = board.getIterations();

            Platform.runLater(() -> {
```

```
                timer.setText(totalTime + " ms");
                iteration.setText(String.valueOf(totalIter));

                btnLoad.setDisable(false);
                btnSolve.setDisable(false);

                if (success) {
                    context.setText("Solution Found!");
                    btnImg.setDisable(false);
                    btnSave.setDisable(false);
                    updateView(board.getFinalGrid());
                } else {
                    context.setText("No Solution.");
                }
            });
        }).start();
    }

    // Visual
    @Override
    public void onStep(char[][] gridData) {
        long currentIter = board.getIterations();
        if (currentIter > 10 && currentIter % 10000 != 0) {
            return;
        }

        char[][] copy = new char[gridData.length][gridData.length];
                 for(int  i=0;  i<gridData.length;  i++)  copy[i]  =
gridData[i].clone();

        String iterCount = String.valueOf(currentIter);

        Platform.runLater(() -> {
            updateView(copy);
            iteration.setText(iterCount);
        });
    }

    private void updateView(char[][] gridToShow) {
        queenGrid.getChildren().clear();
```

```
        queenGrid.getColumnConstraints().clear();
        queenGrid.getRowConstraints().clear();

        int N = startGrid.length;

        for (int i = 0; i < N; i++) {
            ColumnConstraints colConst = new ColumnConstraints();
            colConst.setPrefWidth(TILE_SIZE);
            queenGrid.getColumnConstraints().add(colConst);

            RowConstraints rowConst = new RowConstraints();
            rowConst.setPrefHeight(TILE_SIZE);
            queenGrid.getRowConstraints().add(rowConst);
        }

        for (int row = 0; row < N; row++) {
            for (int col = 0; col < N; col++) {
                char regionChar = startGrid[row][col];
                Color regionColor = getColor(regionChar);
                String hexColor = toHexString(regionColor);

                StackPane cell = new StackPane();
                cell.setPrefSize(TILE_SIZE, TILE_SIZE);
                  cell.setStyle("-fx-background-color: " + hexColor +
";");

                   double top = (row > 0 && startGrid[row-1][col] ==
regionChar) ? 0.0 : 2.0;
                  double right = (col < N-1 && startGrid[row][col+1]
== regionChar) ? 0.0 : 2.0;
                  double bottom = (row < N-1 && startGrid[row+1][col]
== regionChar) ? 0.0 : 2.0;
                  double left = (col > 0 && startGrid[row][col-1] ==
regionChar) ? 0.0 : 2.0;

                                        cell.setBorder(new  Border(new
BorderStroke(Color.BLACK,                    BorderStrokeStyle.SOLID,
CornerRadii.EMPTY, new BorderWidths(top, right, bottom, left))));

                if (showLetters.isSelected()) {
```

```
                                        Text letterText = new
Text(String.valueOf(regionChar));
                                letterText.setFont(Font.font("Arial",
FontWeight.BOLD, TILE_SIZE * 0.4));
                    letterText.setOpacity(0.3);
                    cell.getChildren().add(letterText);
                }

                if (gridToShow[row][col] == '#') {
                    Text queenText = new Text("♛");
                                queenText.setFont(Font.font("Serif",
FontWeight.BOLD, TILE_SIZE * 0.6));
                    cell.getChildren().add(queenText);
                }
                queenGrid.add(cell, col, row);
            }
        }
    }

    // Helper
     private void fileLocation(FileChooser fc, String title, String
extDesc, String extMap, String defaultName) {
        fc.setTitle(title);
                                    fc.getExtensionFilters().add(new
FileChooser.ExtensionFilter(extDesc, extMap));
        fc.setInitialFileName(defaultName);

        File testDir = new File("test");

        fc.setInitialDirectory(testDir);
    }

    private Color getColor(char c) {
        char upper = Character.toUpperCase(c);
            if (upper < 'A' || upper > 'Z') {
                return Color.WHITE;
            }
            int index = upper - 'A';
            return Color.web(tileColors[index]);
    }
```

```
    private String toHexString(Color c) {
        return String.format("#%02X%02X%02X",
                (int) (c.getRed() * 255), (int) (c.getGreen() * 255),
(int) (c.getBlue() * 255));
    }

    private void showAlert(String title, String msg) {
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle(title);
        alert.setHeaderText(null);
        alert.setContentText(msg);
        alert.show();
    }
}
```

5. **Main.java**

```
public class Main {
    public static void main(String[] args) {
        App.main(args);
    }
}
```

6. **Queen.fxml**

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
  Copyright (c) 2015, 2019, Gluon and/or its affiliates.
  All rights reserved. Use is subject to license terms.

  This file is available and licensed under the following license:

  Redistribution and use in source and binary forms, with or without
  modification, are permitted provided that the following conditions
  are met:

  - Redistributions of source code must retain the above copyright
    notice, this list of conditions and the following disclaimer.
```

```
    - Redistributions in binary form must reproduce the above
copyright
      notice, this list of conditions and the following disclaimer in
        the documentation and/or other materials provided with the
distribution.
    - Neither the name of Oracle Corporation nor the names of its
      contributors may be used to endorse or promote products derived
      from this software without specific prior written permission.

      THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
CONTRIBUTORS
    "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
    LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR
      A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT
      OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL,
    SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
    LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
USE,
    DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY
      THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR
TORT
    (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE
USE
      OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH
DAMAGE.
-->

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.CheckBox?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.Menu?>
<?import javafx.scene.control.MenuBar?>
<?import javafx.scene.control.MenuItem?>
<?import javafx.scene.control.SplitPane?>
<?import javafx.scene.layout.AnchorPane?>
```

```
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.RowConstraints?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>

<VBox               prefHeight="600.0"              prefWidth="800.0"
xmlns="http://javafx.com/javafx/25"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="GuiController">
   <children>
      <AnchorPane maxHeight="-1.0" maxWidth="-1.0" prefHeight="-1.0"
prefWidth="-1.0" VBox.vgrow="ALWAYS">
         <children>
                              <MenuBar   AnchorPane.leftAnchor="0.0"
AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">
               <menus>
                  <Menu mnemonicParsing="false" text="File">
                     <items>
                                             <MenuItem  fx:id="btnLoad"
mnemonicParsing="false" text="Load" />
                                             <MenuItem  fx:id="btnSave"
mnemonicParsing="false" text="Save as File" />
                                              <MenuItem  fx:id="btnImg"
mnemonicParsing="false" text="Export" />
                     </items>
                  </Menu>
                  <Menu mnemonicParsing="false" text="Credit">
                     <items>
                                       <MenuItem mnemonicParsing="false"
text="Reynard Nathanael - 13524103" />
                     </items>
                  </Menu>
               </menus>
            </MenuBar>
                  <BorderPane  prefHeight="518.0"  prefWidth="764.0"
style="-fx-background-color:  #254E70;" AnchorPane.bottomAnchor="0.0"
AnchorPane.leftAnchor="0.0"             AnchorPane.rightAnchor="0.0"
AnchorPane.topAnchor="27.0">
               <padding>
```

```
                    <Insets bottom="20.0" left="20.0" right="20.0"
top="20.0" />
                </padding>
                <center>
                                    <SplitPane  dividerPositions="0.6"
prefHeight="160.0"   prefWidth="200.0"   style="-fx-background-color:
#FBF9FF; -fx-background-radius: 20;" BorderPane.alignment="CENTER">
                    <items>
                         <AnchorPane minHeight="0.0" minWidth="0.0"
prefHeight="64.0">
                          <children>
                                              <VBox alignment="TOP_CENTER"
fillWidth="false"  layoutX="64.0"  layoutY="-63.0"  prefHeight="158.4"
prefWidth="380.0"     spacing="10.0"     AnchorPane.bottomAnchor="0.0"
AnchorPane.leftAnchor="0.0"              AnchorPane.rightAnchor="0.0"
AnchorPane.topAnchor="0.0">
                                     <children>
                                                    <Label  fx:id="context"
alignment="CENTER" text="Board" textAlignment="CENTER">
                                                <font>
                                                          <Font name="Century"
size="18.0" />
                                                </font>
                                            </Label>
                                                  <GridPane id="queenGrid"
fx:id="queenGrid">
                                                <columnConstraints>
                                                          <ColumnConstraints
hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />
                                                          <ColumnConstraints
hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />
                                                </columnConstraints>
                                                <rowConstraints>
                                                               <RowConstraints
minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                                                               <RowConstraints
minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                                                               <RowConstraints
minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                                                </rowConstraints>
```

```
                                    </GridPane>
                                </children>
                                <padding>
                                                <Insets bottom="20.0"
left="20.0" right="20.0" top="20.0" />
                                </padding>
                            </VBox>
                        </children>
                    </AnchorPane>
                      <AnchorPane minHeight="0.0" minWidth="0.0"
prefHeight="158.0">
                        <children>
                             <VBox layoutX="90.0" layoutY="-27.0"
prefHeight="158.4"   prefWidth="252.8"   AnchorPane.bottomAnchor="0.0"
AnchorPane.leftAnchor="0.0"              AnchorPane.rightAnchor="0.0"
AnchorPane.topAnchor="0.0">
                                <children>
                                      <AnchorPane prefHeight="97.0"
prefWidth="301.0">
                                     <children>
                                          <VBox alignment="CENTER"
prefHeight="98.0"   prefWidth="301.0"   AnchorPane.bottomAnchor="0.0"
AnchorPane.leftAnchor="0.0"              AnchorPane.rightAnchor="0.0"
AnchorPane.topAnchor="0.0">
                                              <children>
                                                 <Label text="Timer">
                                                    <font>
                                                                <Font
name="Century" size="16.0" />
                                                    </font>
                                                    <padding>
                                                              <Insets
bottom="10.0" />
                                                    </padding>
                                                 </Label>
                                                     <Label id="timer"
fx:id="timer" text="00:00 ms" />
                                              </children>
                                              <padding>
```

```
                                                                <Insets
bottom="20.0" left="20.0" right="20.0" top="20.0" />
                                                        </padding>
                                                    </VBox>
                                                </children>
                                            </AnchorPane>
                                                <AnchorPane prefHeight="97.0"
prefWidth="301.0">
                                                <children>
                                                     <VBox alignment="CENTER"
prefHeight="98.0"   prefWidth="301.0"   AnchorPane.bottomAnchor="0.0"
AnchorPane.leftAnchor="0.0"              AnchorPane.rightAnchor="0.0"
AnchorPane.topAnchor="0.0">
                                                        <children>
                                                             <Label text="Total
Iterations">
                                                                <font>
                                                                    <Font
name="Century" size="16.0" />
                                                                </font>
                                                                <padding>
                                                                    <Insets
bottom="10.0" />
                                                                </padding>
                                                            </Label>
                                                                <Label
fx:id="iteration" text="0" />
                                                        </children>
                                                        <padding>
                                                                <Insets
bottom="20.0" left="20.0" right="20.0" top="20.0" />
                                                        </padding>
                                                    </VBox>
                                                </children>
                                            </AnchorPane>
                                                <AnchorPane prefHeight="97.0"
prefWidth="301.0">
                                                <children>
                                                        <BorderPane
prefHeight="200.0"   prefWidth="300.0"   AnchorPane.bottomAnchor="0.0"
```

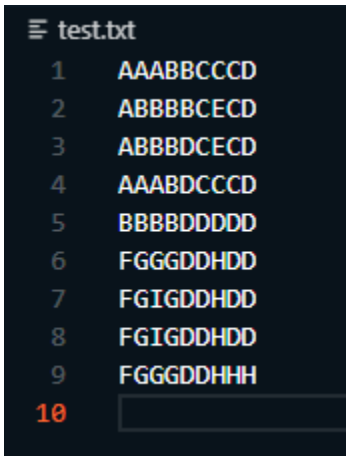
```
AnchorPane.leftAnchor="0.0"              AnchorPane.rightAnchor="0.0"
AnchorPane.topAnchor="0.0">
                                                <center>
                                                                <CheckBox
fx:id="showLetters"     alignment="CENTER"     mnemonicParsing="false"
prefHeight="20.0"      prefWidth="125.0"      text="Show      Letters"
BorderPane.alignment="CENTER" />
                                                </center>
                                                <bottom>
                                                                <CheckBox
fx:id="liveUpdate"     alignment="CENTER"     mnemonicParsing="false"
prefHeight="20.0"    prefWidth="125.0"    text="Show    Live    Update"
BorderPane.alignment="CENTER" />
                                                </bottom>
                                                <padding>
                                                                <Insets
bottom="10.0" top="10.0" />
                                                </padding>
                                                <top>
                                                                 <Label
text="Settings" BorderPane.alignment="CENTER">
                                                                <font>
                                                                 <Font
name="Century" size="16.0" />
                                                                </font>
                                                                <padding>
                                                                <Insets
bottom="10.0" />
                                                                </padding>
                                                            </Label>
                                                </top>
                                            </BorderPane>
                                        </children>
                                    </AnchorPane>
                                    <AnchorPane prefHeight="120.0"
prefWidth="237.0">
                                        <children>
                                            <VBox alignment="CENTER"
layoutX="59.0"  layoutY="-48.0"  prefHeight="152.0"  prefWidth="236.8"
spacing="10.0"                          AnchorPane.bottomAnchor="0.0"
```

```
AnchorPane.leftAnchor="0.0"              AnchorPane.rightAnchor="0.0"
AnchorPane.topAnchor="0.0">
                                                <children>
                                                                    <Button
id="btnSolve"         fx:id="btnSolve"         mnemonicParsing="false"
prefHeight="41.0" prefWidth="187.0" text="Solve" />
                                                </children>
                                             </VBox>
                                          </children>
                                       </AnchorPane>
                                    </children>
                                 </VBox>
                              </children>
                           </AnchorPane>
                        </items>
                     </SplitPane>
                  </center>
                  <top>
                           <Label  alignment="CENTER"  prefHeight="44.0"
prefWidth="357.0"  style="-fx-font-style: WHITE;" text="Queen's Game"
textAlignment="CENTER"                                textFill="WHITE"
BorderPane.alignment="CENTER">
                        <font>
                           <Font name="Century" size="30.0" />
                        </font>
                        <padding>
                           <Insets bottom="10.0" />
                        </padding>
                     </Label>
                  </top>
               </BorderPane>
            </children>
         </AnchorPane>
      </children>
</VBox>
```

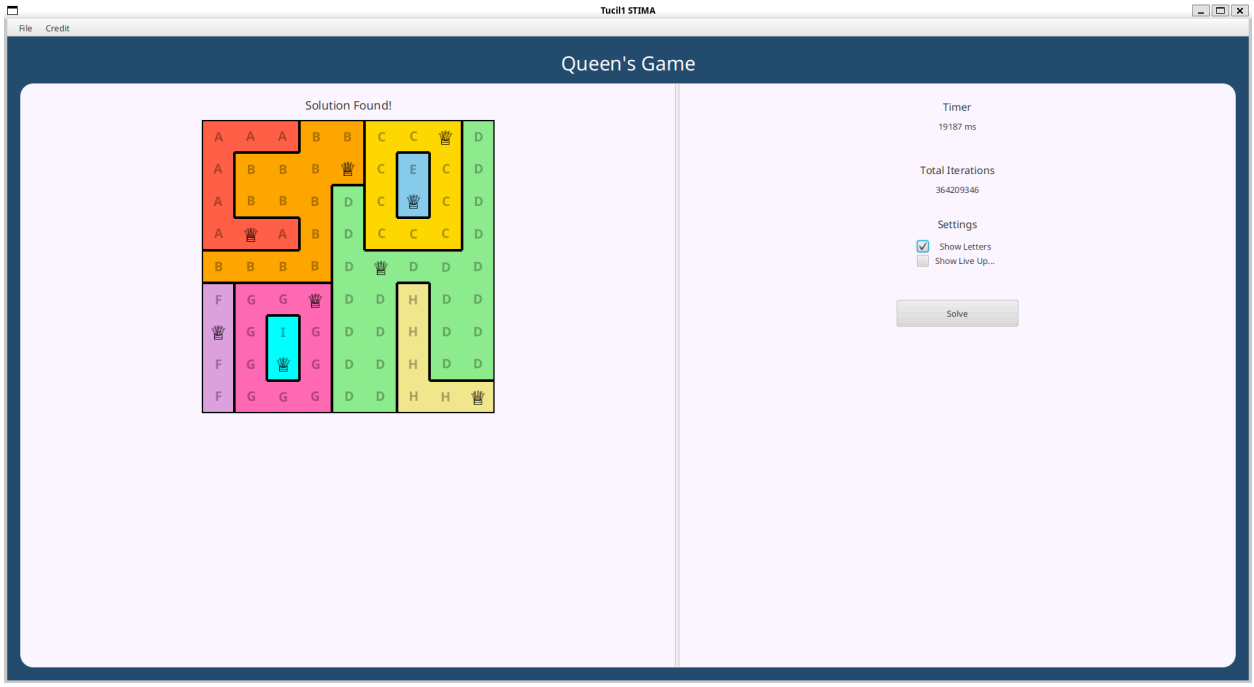
Bab 3: Tangkapan Layar dan Pengujian

1. **File: test.txt**

   Input



   Output

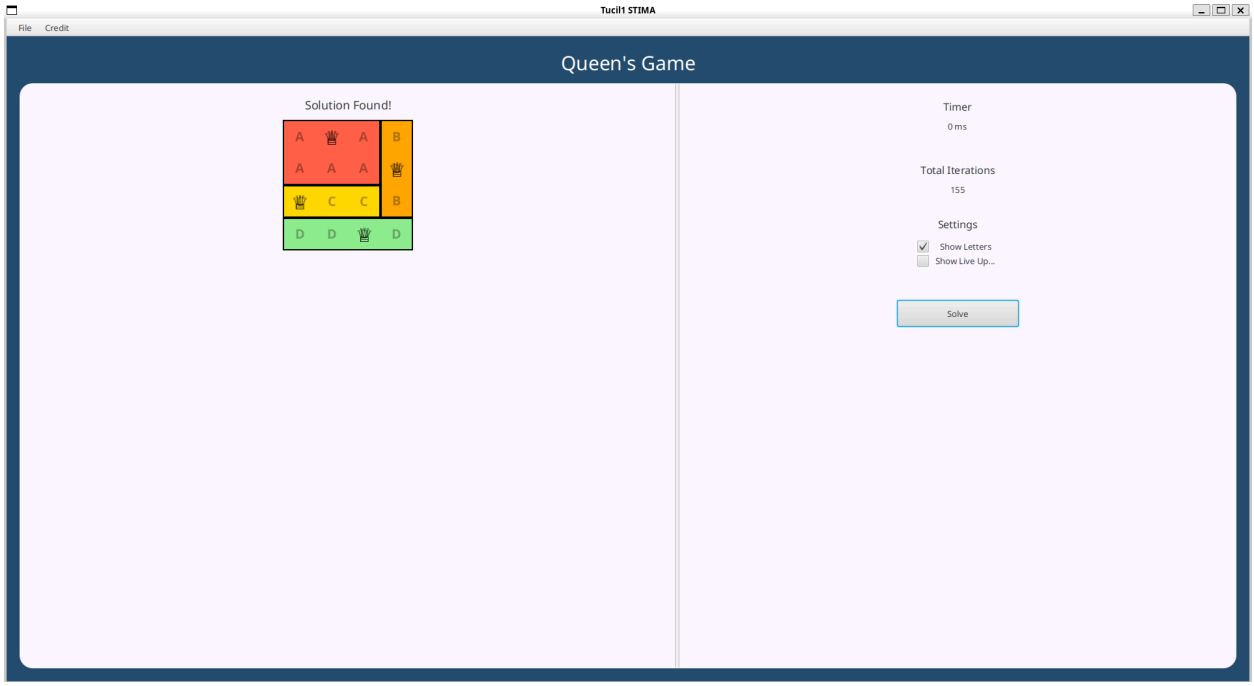


2. **File: test1.txt**

   Input

```
test1.txt
1  AAAB
2  AAAB
3
4  CCCB
5
6  DDDD
```

Output

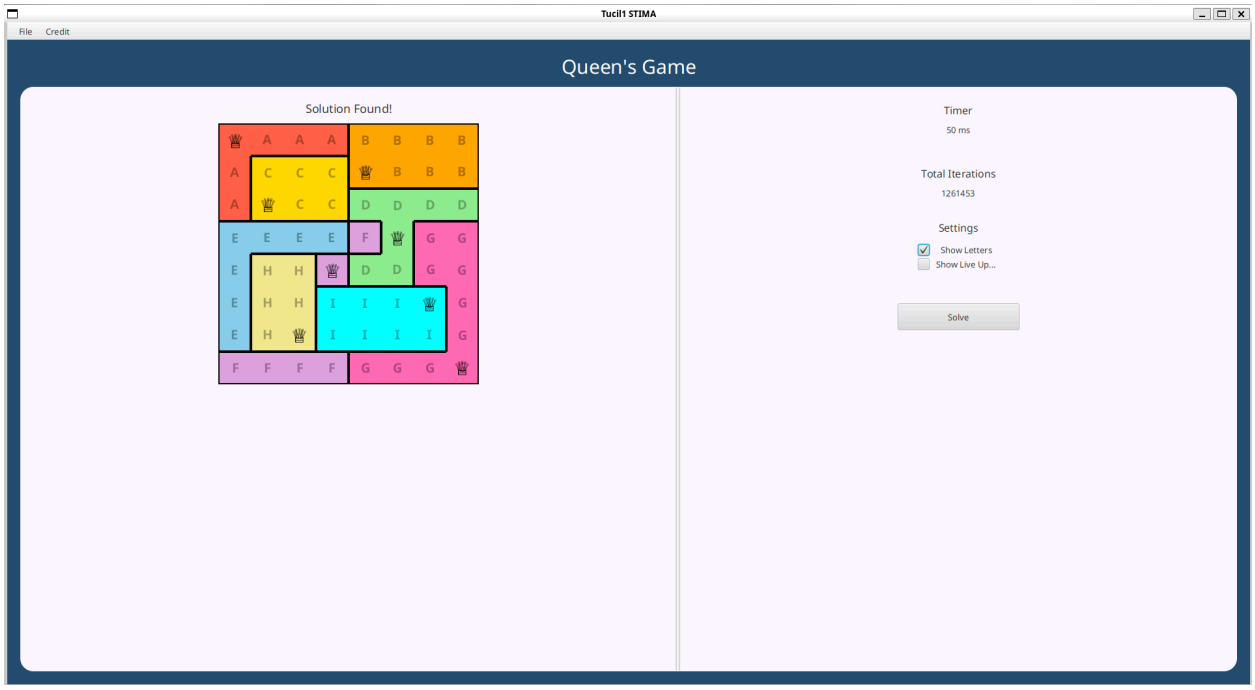


3. **File: test2.txt**

Input

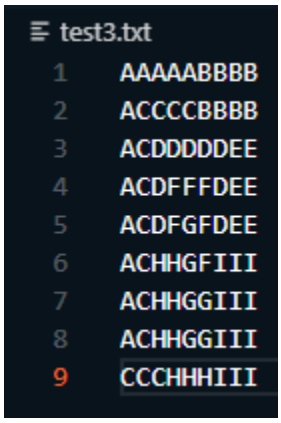
```
test2.txt
1  AAAABBBB
2  ACCCBBBB
3  ACCCDDDD
4  EEEEFDGG
5  EHHFDDGG
6  EHHIIIIG
7  EHHIIIIG
8  FFFFGGGG
```

Output

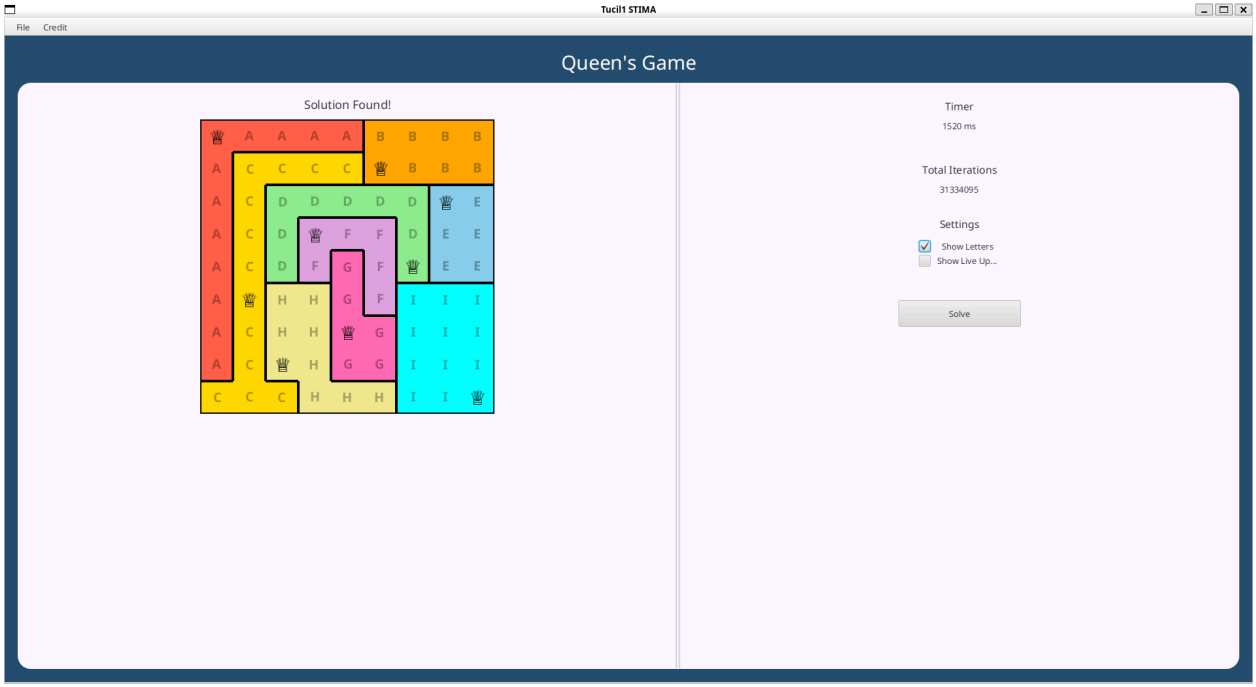


4. **File: test3.txt**

Input

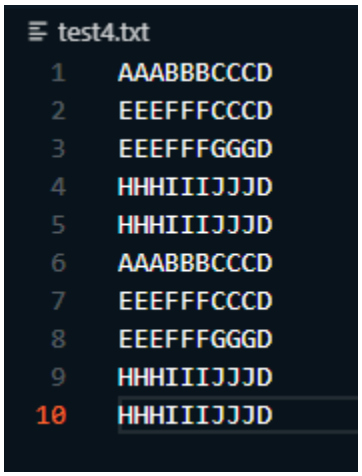


Output

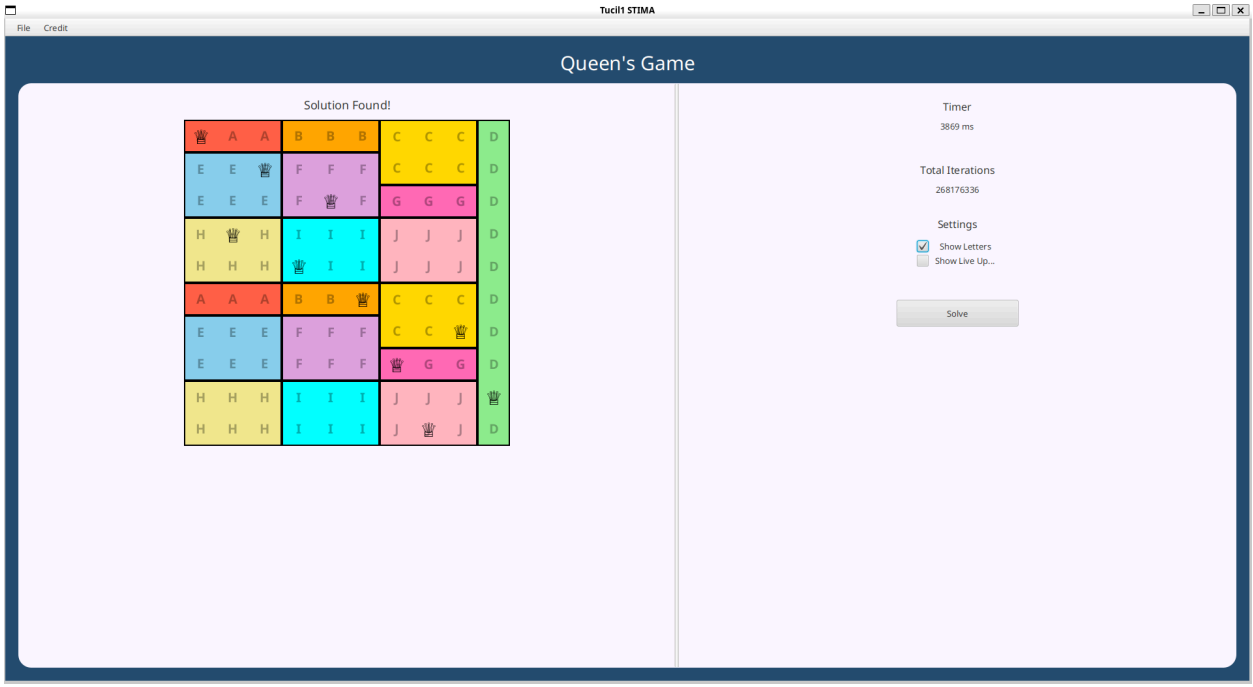


5. **File: test4.txt**

Input

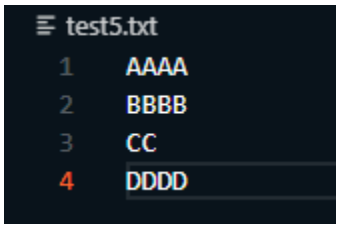


Output

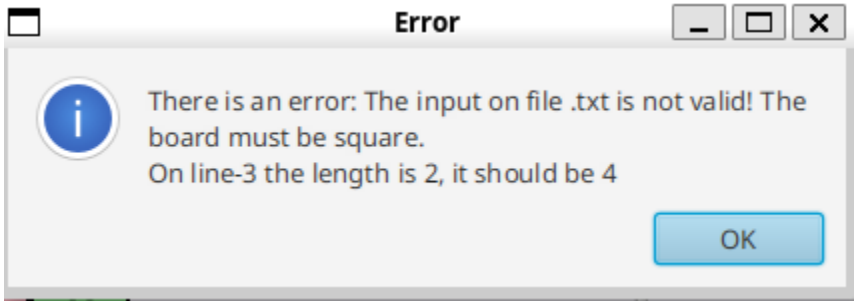


6. File: test5.txt

Input



Output



Bab 4: Referensi

Link Github Repository: https://github.com/Hagon47/Tucil1_13524103

Referensi:

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/02-Algoritma-Brute-Force-(2026)-Bag1.pdf
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/03-Algoritma-Brute-Force-(2026)-Bag2.pdf
https://jenkov.com/tutorials/javafx/fxml.html
https://jenkov.com/tutorials/javafx/gridpane.html
https://docs.oracle.com/javase/8/javafx/api/javafx/scene/paint/Color.html
https://docs.oracle.com/javafx/2/threads/jfxpub-threads.htm
https://code.makery.ch/blog/javafx-2-snapshot-as-png-image/

Bab 5: Lampiran

No	Poin	Ya	Tidak
1	Program berhasil di kompilasi tanpa kesalahan	✓	
2	Program berhasil di jalankan	✓	
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	✓	

PERNYATAAN

Tugas ini disusun sepenuhnya tanpa bantuan kecerdasan buatan (Generative AI), melainkan hasil pemikiran dan analisis mandiri.

Reynard Nathanael