LAPORAN TUGAS KECIL 1

IF2211 Strategi Algoritma

Disusun oleh:
Zeki Amani - 13524082

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung
Jl. Ganesha 10, Bandung 40132

BAB 1
Pendahuluan

1.1 deskripsi tugas

Queens adalah gim logika yang tersedia pada situs jejaring profesional LinkedIn. Tujuan dari gim ini adalah menempatkan queen pada sebuah papan persegi berwarna sehingga terdapat hanya satu queen pada tiap baris, kolom, dan daerah warna. Selain itu, satu queen tidak dapat ditempatkan bersebelahan dengan queen lainnya, termasuk secara diagonal.

Tujuan tugas anda adalah membuat program yang dapat menemukan satu solusi penempatan queen pada suatu papan berwarna yang diberikan, atau menampilkan bahwa tidak ada solusi yang valid. Program melakukan pencarian solusi menggunakan algoritma brute force.

BAB 2
Pembahasan

2.1 Algoritma

game ini dapat di selesaikan secara bruteforce dengan cara menaruh 1 queen pada masing masing baris. lalu iterasi dilakukan dengan cara memajukan queen terbawah untuk tranverse ke kanan kemudian jika sudah menyentuh batas, maka queen itu kembali ke posisi awal namun sekarang queen diatasnya maju 1 langkah dan hal sama berlaku untuk queen pada baris lebih atasnya lagi apabila queen dibawahnya sudah menyentuh batas kanan. dengan mengikuti aturan iterasi ini, maka akan didapatkan semua kemungkinan posisi queen pada board dengan constrain queen tidak boleh diposisi yang sama dan tidak di baris yang sama. dan tentu saja pada tiap iterasi dilakukan pengecekan apakah kombinasi posisi queen pada iterasi saat ini sudah memenuhi syarat penyelesaian game.

Pengecekan dilakukan dengan cara mengecek untuk setiap queen, apakah dia mengikuti aturan “terdapat hanya satu queen pada tiap baris, kolom, dan daerah warna. Selain itu, satu queen tidak dapat ditempatkan bersebelahan dengan queen lainnya, termasuk secara diagonal”.

Berikut adalah step by step dari algoritma yang digunakan:

1. Persiapan papan: Program membuat representasi papan permainan yang menyimpan informasi warna setiap kotak dan apakah kotak tersebut berisi queen atau tidak. Pada awalnya, semua kotak kosong.
2. Mulai mencoba semua kemungkinan: Program mencoba semua kombinasi posisi queen secara sistematis, dimulai dari kombinasi pertama hingga kombinasi terakhir yang mungkin.
3. Menempatkan queen pada papan: Pada setiap percobaan, program menempatkan satu queen di setiap baris papan. Posisi kolom dari setiap queen berubah-ubah mengikuti pola sistematis, mirip seperti odometer mobil yang bergulir.
4. Memeriksa keabsahan posisi: Setelah semua queen ditempatkan, program mengecek apakah konfigurasi tersebut memenuhi semua aturan permainan:
    i. Setiap kolom hanya boleh berisi satu queen
    ii. Setiap baris hanya boleh berisi satu queen
    iii. Tidak ada dua queen yang bersebelahan secara diagonal
    iv. Setiap daerah warna hanya boleh berisi satu queen
5. Mengambil keputusan: Jika semua aturan terpenuhi, maka solusi telah ditemukan dan program berhenti. Jika tidak, program menghapus semua queen yang baru saja ditempatkan dan mencoba kombinasi berikutnya.
6. Selesai: Program terus mencoba kombinasi baru sampai menemukan solusi yang valid atau sampai semua kemungkinan habis dicoba. Jika tidak ada kombinasi yang valid, program akan melaporkan bahwa tidak ada solusi untuk konfigurasi papan tersebut.

metode ini murni bruteforce karena tiap iterasi peletakkan queen hanya bergantung pada constraint yang diberikan oleh game, maka dari itu metode ini bukan lah algoritma hereustik.

BAB 3
Implementasi

3.1 Struktur Repository

```
tucil1-13524082/
├── src/
│   └── main/
│       └── java/
│           └── tucil1/
│               ├── Bruteforce.java
│               ├── CLI.java
│               ├── GUI.java
│               └── Launcher.java
├── test/
├── target/
├── doc/
├── pom.xml
└── README.md
```

Berikut adalah keterangan untuk tiap folder:

1. Folder src berisi source code program.
2. Folder test berisi input test case untuk pengujian berupa file txt.
3. Folder target berisi executable file

4. Folder doc berisi laporan yaitu document ini.

3.2 Source Code

1. Bruteforce,java
   class ini berisi penerapan dari algoritma inti dari program ini yaitu untuk meng-solve game ini.

```
package tucil1;
public class Bruteforce{
    public static long iterasi = 0;
    public static char[][][] currentmap = null;

    public static boolean cek(int x,int y, char map[][][]){
        //cek sekolom
        for (int i = 0; i < map.length; i++) {
            if(map[i][y][1]=='Y' && i!=x){
                return false;
            }
        }
        //cek serow
        for (int i = 0; i < map.length; i++) {
            if(map[x][i][1]=='Y' && i!=y){
                return false;
            }
        }
        //cek adjacent diagonal
        if(x+1 < map.length && y+1 < map.length &&
map[x+1][y+1][1]=='Y')
            {return false;}
        if(x+1 < map.length && y-1 >= 0 && map[x+1][y-1][1]=='Y')
            {return false;}
        if(x-1 >= 0 && y+1 < map.length && map[x-1][y+1][1]=='Y')
            {return false;}
        if(x-1 >= 0 && y-1 >= 0 && map[x-1][y-1][1]=='Y')
            {return false;}

        //cek warna
        for (int i = 0; i < map.length; i++) {
            for (int j = 0; j < map.length; j++) {
                if(map[i][j][1]=='Y' &&
map[i][j][0]==map[x][y][0] && i!=x && j!=y){
                    return false;
                }
```

```
            }
        }
        return true;
    }
    public static boolean isvalid(char[][][] map) {
        for (int i = 0; i < map.length; i++) {
            for (int j = 0; j < map.length; j++) {
                if(map[i][j][1]!='Y') continue;
                if(!cek(i,j,map)){
                    return false;
                }
            }
        }
        return true;
    }
    public static char[][][] bruteforce(char[][][] map){
        int n=map.length;
        for (long count = 0; count < (long) Math.pow(n, n);
count++) {
            iterasi++;
            long tempCount = count;
            for (int row = 0; row < n; row++) {
                map[row][(int) tempCount % n][1] = 'Y';
                tempCount /= n;
            }
            if (isvalid(map)) {
                return map;
            }
            tempCount = count;
            for (int row = 0; row < n; row++) {
                map[row][(int)(tempCount % n)][1] = 'N';
                tempCount /= n;
            }
        }
        return null;
    }

    public static char[][][] copy(char[][][] map) {
        if (map == null) return null;
        int n = map.length;
        char[][][] copy = new char[n][n][2];
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
```

```
                copy[i][j][0] = map[i][j][0];
                copy[i][j][1] = map[i][j][1];
            }
        }
        return copy;
    }

    public static char[][][] solve(String[] input, int n) {
        char[][][] map = new char[n][n][2];
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                map[i][j][1] = 'N';
                map[i][j][0] = input[i].charAt(j);
            }
        }
        currentmap = map;
        return bruteforce(map);

    }
}
```

2. GUI.java
   class ini berisikan graphical user interface yang dibangun menggunakan javafx

```
package tucil1;

import javafx.application.Application;
import javafx.application.Platform;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.scene.paint.Color;
import javafx.scene.shape.Rectangle;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.scene.text.Text;
import javafx.stage.FileChooser;
import javafx.stage.Stage;
import javafx.animation.Timeline;
import javafx.animation.KeyFrame;
import javafx.util.Duration;
```

```java
import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import java.nio.file.Files;
import java.util.List;

public class GUI extends Application {

    private TextArea inputArea;
    private GridPane boardGrid;
    private Label statusLabel;
    private Label iterationsLabel;
    private Label timeLabel;
    private Button LoadButton;
    private Button solveButton;
    private Button loadFileButton;
    private Button clearButton;
    private Spinner<Integer> sizeSpinner;
    private CheckBox liveUpdateCheckBox;
    private Spinner<Integer> updateIntervalSpinner;
    private int BoardSize = 9;
    private Timeline updateTimeline;
    private long starttime;

    private final Color[] colorlist = {
        Color.web("#FFB6C1"),
        Color.web("#87CEEB"),
        Color.web("#98FB98"),
        Color.web("#FFD700"),
        Color.web("#DDA0DD"),
        Color.web("#F0E68C"),
        Color.web("#FFA07A"),
        Color.web("#E6E6FA"),
        Color.web("#FFE4B5"),
        Color.web("#F08080"),
        Color.web("#AFEEEE"),
        Color.web("#DB7093"),
        Color.web("#FFDAB9"),
        Color.web("#E0BBE4"),
        Color.web("#FFDEAD"),
        Color.web("#D8BFD8"),
        Color.web("#FFB347"),
        Color.web("#B0E0E6"),
```

```
            Color.web("#F5DEB3"),
            Color.web("#FFC0CB"),
            Color.web("#FFFACD"),
            Color.web("#C8A2C8"),
            Color.web("#B19CD9"),
            Color.web("#FFB6D9"),
            Color.web("#ADD8E6"),
            Color.web("#FAFAD2")
    };

    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("LinkedIn Queen Solver");
        BorderPane root = new BorderPane();
        root.setPadding(new Insets(10));
        VBox topSection = createTopSection();
        root.setTop(topSection);
        VBox leftSection = createLeftSection();
        root.setLeft(leftSection);
        VBox centerSection = createCenterSection();
        root.setCenter(centerSection);
        VBox bottomSection = createBottomSection();
        root.setBottom(bottomSection);
        Scene scene = new Scene(root, 1200, 800);
        primaryStage.setScene(scene);
        primaryStage.show();
        initializeBoard(BoardSize);
    }

    private VBox createTopSection() {
        VBox vbox = new VBox(10);
        vbox.setAlignment(Pos.CENTER);
        vbox.setPadding(new Insets(10));

        Text title = new Text("LinkedIn Queen Solver");
        title.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 28));
        title.setFill(Color.web("#2C3E50"));


        vbox.getChildren().addAll(title);
        return vbox;
    }
```

```
    private VBox createLeftSection() {
        VBox vbox = new VBox(15);
        vbox.setPadding(new Insets(10));
        vbox.setPrefWidth(300);

        HBox sizeBox = new HBox(10);
        sizeBox.setAlignment(Pos.CENTER_LEFT);
        Label sizeLabel = new Label("Board Size:");
        sizeLabel.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 12));
        sizeSpinner = new Spinner<>(3, 99, 9);
        sizeSpinner.setEditable(true);
        sizeSpinner.setPrefWidth(80);
        sizeSpinner.valueProperty().addListener((obs, oldVal,
newVal) -> {
            BoardSize = newVal;
            initializeBoard(BoardSize);
            inputArea.clear();
        });
        sizeBox.getChildren().addAll(sizeLabel, sizeSpinner);

        Label liveUpdateLabel = new Label("Live Update
Settings:");
        liveUpdateLabel.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 12));

        liveUpdateCheckBox = new CheckBox("Show Live Updates");
        liveUpdateCheckBox.setSelected(true);
        liveUpdateCheckBox.setFont(Font.font("Times New Roman",
11));

        HBox intervalBox = new HBox(10);
        intervalBox.setAlignment(Pos.CENTER_LEFT);
        Label intervalLabel = new Label("Update Interval (ms):");
        intervalLabel.setFont(Font.font("Times New Roman", 11));
        updateIntervalSpinner = new Spinner<>(100, 10000, 500,
100);
        updateIntervalSpinner.setEditable(true);
        updateIntervalSpinner.setPrefWidth(100);

updateIntervalSpinner.disableProperty().bind(liveUpdateCheckBox.s
electedProperty().not());
```

```
        intervalBox.getChildren().addAll(intervalLabel,
updateIntervalSpinner);

        Label inputLabel = new Label("Bentuk Board:");
        inputLabel.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 12));

        inputArea = new TextArea();
        inputArea.setPromptText("Enter board configuration (e.g.,
AAABBCCCD\r\n" + //
                        "ABBBBCECD\r\n" + //
                        "ABBBDCECD\r\n" + //
                        "AAABDCCCD\r\n" + //
                        "BBBBDDDDD\r\n" + //
                        "FGGGDDHDD\r\n" + //
                        "FGIGDDHDD\r\n" + //
                        "FGIGDDHDD\r\n" + //
                        "FGGGDDHHH\r\n" + //
                        ")\n");
        inputArea.setPrefHeight(200);
        inputArea.setFont(Font.font("Times New Roman", 12));

        LoadButton = new Button("Load");
        LoadButton.setMaxWidth(Double.MAX_VALUE);
        LoadButton.setStyle("-fx-background-color: #3498db;
-fx-text-fill: white; -fx-font-weight: bold;");
        LoadButton.setOnAction(e -> loadConfig());

        loadFileButton = new Button("Load from File");
        loadFileButton.setMaxWidth(Double.MAX_VALUE);
        loadFileButton.setOnAction(e -> loadFromFile());

        solveButton = new Button("Solve");
        solveButton.setMaxWidth(Double.MAX_VALUE);
        solveButton.setStyle("-fx-background-color: #27ae60;
-fx-text-fill: white; -fx-font-weight: bold;");
        solveButton.setOnAction(e -> solve());
        solveButton.setDisable(true);

        clearButton = new Button("Clear");
        clearButton.setMaxWidth(Double.MAX_VALUE);
        clearButton.setOnAction(e -> clearBoard());
```

```
        vbox.getChildren().addAll(
            sizeBox,
            new Separator(),
            liveUpdateLabel,
            liveUpdateCheckBox,
            intervalBox,
            new Separator(),
            inputLabel,
            inputArea,
            LoadButton,
            loadFileButton,
            solveButton,
            clearButton
        );

        return vbox;
    }

    private VBox createCenterSection() {
        VBox vbox = new VBox(10);
        vbox.setAlignment(Pos.CENTER);
        vbox.setPadding(new Insets(10));


        boardGrid = new GridPane();
        boardGrid.setAlignment(Pos.CENTER);
        boardGrid.setHgap(2);
        boardGrid.setVgap(2);
        boardGrid.setStyle("-fx-background-color: #000000;
-fx-padding: 5;");

        vbox.getChildren().addAll(boardGrid);
        return vbox;
    }

    private VBox createBottomSection() {
        VBox vbox = new VBox(5);
        vbox.setPadding(new Insets(10));
        vbox.setAlignment(Pos.CENTER);

        statusLabel = new Label("");
        statusLabel.setFont(Font.font("Times New Roman",
FontWeight.BOLD, 13));
```

```
        statusLabel.setTextFill(Color.web("#2C3E50"));

        iterationsLabel = new Label("Iterations: -");
        iterationsLabel.setFont(Font.font("Times New Roman",
12));

        timeLabel = new Label("Time: -");
        timeLabel.setFont(Font.font("Times New Roman", 12));

        vbox.getChildren().addAll(statusLabel, iterationsLabel,
timeLabel);
        return vbox;
    }

    private void initializeBoard(int size) {
        boardGrid.getChildren().clear();

        double cellSize = Math.min(60, 600.0 / size);

        for (int i = 0; i < size; i++) {
            for (int j = 0; j < size; j++) {
                StackPane cell = new StackPane();
                Rectangle rect = new Rectangle(cellSize,
cellSize);
                rect.setFill(Color.LIGHTGRAY);
                rect.setStroke(Color.DARKGRAY);
                rect.setStrokeWidth(1);

                cell.getChildren().add(rect);
                boardGrid.add(cell, j, i);
            }
        }
    }

    private void visualizeBoard(char[][][] solution) {
        if (solution == null) return;

        boardGrid.getChildren().clear();
        int size = solution.length;
        double cellSize = Math.min(60, 600.0 / size);

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

```
                StackPane cell = new StackPane();
                Rectangle rect = new Rectangle(cellSize,
cellSize);

                char colorChar = solution[i][j][0];
                Color cellColor = getcolor(colorChar);
                rect.setFill(cellColor);
                rect.setStroke(Color.DARKGRAY);
                rect.setStrokeWidth(1);

                cell.getChildren().add(rect);

                if (solution[i][j][1] == 'Y') {
                    Text queen = new Text("♕");
                    queen.setFont(Font.font("Times New Roman",
FontWeight.BOLD, cellSize * 0.6));
                    queen.setFill(Color.web("#000000"));
                    queen.setStroke(Color.GOLD);
                    queen.setStrokeWidth(1);
                    cell.getChildren().add(queen);
                }

                boardGrid.add(cell, j, i);
            }
        }
    }

    private Color getcolor(char c) {
        int index = Character.toUpperCase(c) - 'A';
        if (index >= 0 && index < colorlist.length) {
            return colorlist[index];
        }
        return Color.LIGHTGRAY;
    }

    private void visualizeInitialBoard(String[] config, int size)
{
        boardGrid.getChildren().clear();
        double cellSize = Math.min(60, 600.0 / size);

        for (int i = 0; i < size; i++) {
            for (int j = 0; j < size; j++) {
                StackPane cell = new StackPane();
```

```
                Rectangle rect = new Rectangle(cellSize,
cellSize);

                char colorChar = config[i].trim().charAt(j);
                Color cellColor = getcolor(colorChar);
                rect.setFill(cellColor);
                rect.setStroke(Color.DARKGRAY);
                rect.setStrokeWidth(1);

                cell.getChildren().add(rect);

                Text colorLabel = new
Text(String.valueOf(colorChar));
                colorLabel.setFont(Font.font("Times New Roman",
FontWeight.BOLD, cellSize * 0.3));
                colorLabel.setFill(Color.web("#2C3E50"));
                cell.getChildren().add(colorLabel);

                boardGrid.add(cell, j, i);
            }
        }
    }

    private void loadConfig() {
        String input = inputArea.getText().trim();
        if (input.isEmpty()) {
            showAlert("Input Required", "belum ada config
board-nya njir");
            return;
        }

        String[] lines = input.split("\n");
        if (lines.length != BoardSize) {
            showAlert("Invalid Input", "jumlah barisnya salah");
            return;
        }

        for (String line : lines) {
            if (line.trim().length() != BoardSize) {
                showAlert("Invalid Input", "jumlah kolomnnya ada
yang ga bener");
                return;
            }
```

```
        }

        visualizeInitialBoard(lines, BoardSize);
        statusLabel.setText("Ready to solve");
        statusLabel.setTextFill(Color.web("#3498db"));
        solveButton.setDisable(false);
    }

    private void solve() {
        String input = inputArea.getText().trim();
        if (input.isEmpty()) {
            showAlert("Input Required", "belum ada config
board-nya njir");
            return;
        }
        String[] lines = input.split("\n");
        if (lines.length != BoardSize) {
            showAlert("Invalid Input", "jumlah barisnya salah");
            return;
        }
        for (String line : lines) {
            if (line.trim().length() != BoardSize) {
                showAlert("Invalid Input", "jumlah kolomnnya ada
yang ga bener");
                return;
            }
        }
        solveButton.setDisable(true);
        statusLabel.setText("Solving...");
        statusLabel.setTextFill(Color.web("#F39C12"));

        Bruteforce.iterasi = 0;
        Bruteforce.currentmap = null;
        starttime = System.nanoTime();

        if (liveUpdateCheckBox.isSelected()) {
            int intervalMs = updateIntervalSpinner.getValue();
            updateTimeline = new Timeline(new
KeyFrame(Duration.millis((double) intervalMs), e -> {
                long currentIterations = Bruteforce.iterasi;
                double elapsedSeconds = (System.nanoTime() -
starttime) / 1_000_000_000.0;
```

```
iterationsLabel.setText(String.format("Iterations: %,d",
currentIterations));
                timeLabel.setText(String.format("Time: %.2f
seconds", elapsedSeconds));
                if (Bruteforce.currentmap != null) {

visualizeBoard(Bruteforce.copy(Bruteforce.currentmap));
                }
            }));
            updateTimeline.setCycleCount(Timeline.INDEFINITE);
            updateTimeline.play();
        }

        new Thread(() -> {
            char[][][] result =
Bruteforce.solve(lines,BoardSize);
            long endtime = System.nanoTime();
            double timeSeconds = (endtime - starttime) /
1_000_000_000.0;

            Platform.runLater(() -> {
                if (updateTimeline != null) {
                    updateTimeline.stop();
                }

                if (result!=null) {
                    visualizeBoard((char[][][])result);
                    saveSolutionToFile(result);
                    statusLabel.setText("Solution Found!");

statusLabel.setTextFill(Color.web("#27AE60"));

                } else {
                    statusLabel.setText("No Solution Found");

statusLabel.setTextFill(Color.web("#E74C3C"));
                }


iterationsLabel.setText(String.format("Iterations: %,d",
Bruteforce.iterasi));
```

```
                timeLabel.setText(String.format("Time: %.6f
seconds", timeSeconds));
                solveButton.setDisable(false);
            });
        }).start();
    }

    private void loadFromFile() {
        FileChooser fileChooser = new FileChooser();
        fileChooser.setTitle("Open Board Configuration");
        fileChooser.getExtensionFilters().add(
            new FileChooser.ExtensionFilter("Text Files",
"*.txt")
        );

        File file =
fileChooser.showOpenDialog(boardGrid.getScene().getWindow());
        if (file != null) {
            try {
                List<String> lines =
Files.readAllLines(file.toPath());
                if (!lines.isEmpty()) {
                    lines.removeIf(line ->
line.trim().isEmpty());
                    if (lines.isEmpty()) {
                        showAlert("Error", "File is empty");
                        return;
                    }
                    int size = lines.get(0).trim().length();
                    for (String line : lines) {
                        if (line.trim().length() != size) {
                            showAlert("Invalid File", "All rows
must have the same length");
                            return;
                        }
                    }
                    sizeSpinner.getValueFactory().setValue(size);
                    StringBuilder sb = new StringBuilder();
                    for (String line : lines) {
                        sb.append(line.trim()).append("\n");
                    }
                    inputArea.setText(sb.toString().trim());
```

```
                    visualizeInitialBoard(lines.toArray(new
String[0]), size);
                    solveButton.setDisable(false);

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

    private void clearBoard() {
        if (updateTimeline != null) {
            updateTimeline.stop();
        }

        inputArea.clear();
        initializeBoard(BoardSize);
        statusLabel.setText("Ready to solve");
        statusLabel.setTextFill(Color.web("#2C3E50"));
        iterationsLabel.setText("Iterations: -");
        timeLabel.setText("Time: -");
        solveButton.setDisable(true);
    }

    private void saveSolutionToFile(char[][][] solution) {
        if (solution == null) return;

        FileChooser fileChooser = new FileChooser();
        fileChooser.setTitle("Save Solution");
        fileChooser.setInitialFileName("solution.txt");
        fileChooser.getExtensionFilters().add(
            new FileChooser.ExtensionFilter("Text Files",
"*.txt")
        );

        File file =
fileChooser.showSaveDialog(boardGrid.getScene().getWindow());
        if (file != null) {
            try (PrintWriter writer = new PrintWriter(file)) {
                int size = solution.length;
                for (int i = 0; i < size; i++) {
```

```
                    for (int j = 0; j < size; j++) {
                        if (solution[i][j][1] == 'Y') {
                            writer.print('#');
                        } else {
                            writer.print(solution[i][j][0]);
                        }
                    }
                    writer.println();
                }

                Alert alert = new
Alert(Alert.AlertType.INFORMATION);
                alert.setTitle("Success");
                alert.setHeaderText(null);
                alert.setContentText("Solution saved to: " +
file.getName());
                alert.showAndWait();

            } catch (IOException e) {
                showAlert("Error", "Failed to save solution: " +
e.getMessage());
            }
        }
    }

    private void showAlert(String title, String message) {
        Alert alert = new Alert(Alert.AlertType.WARNING);
        alert.setTitle(title);
        alert.setHeaderText(null);
        alert.setContentText(message);
        alert.showAndWait();
    }

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

3. Launcher.java
   class ini hanya digunakan sebagai entrypoint untuk maven yang mana akan diarahkan untuk mengeksekusi method main pada class GUI

```
package tucil1;
```

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

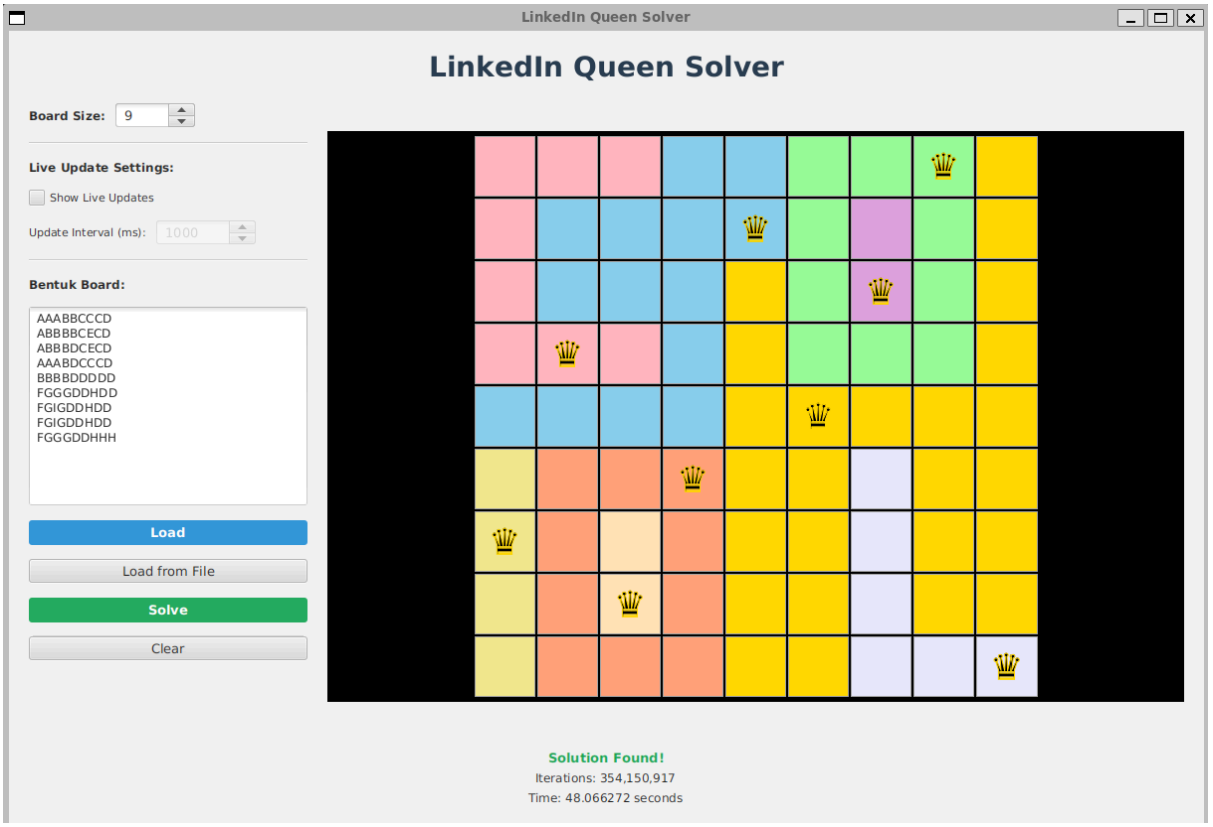
BAB 4
Pengujian

semua test case dapat diakses pada folder test

4.1 test1.txt

input:
AAABBCCCD
ABBBBCECD
ABBBDCECD
AAABDCCCD
BBBBDDDDD
FGGGDDHDD
FGIGDDHDD
FGIGDDHDD
FGGGDDHHH

output:

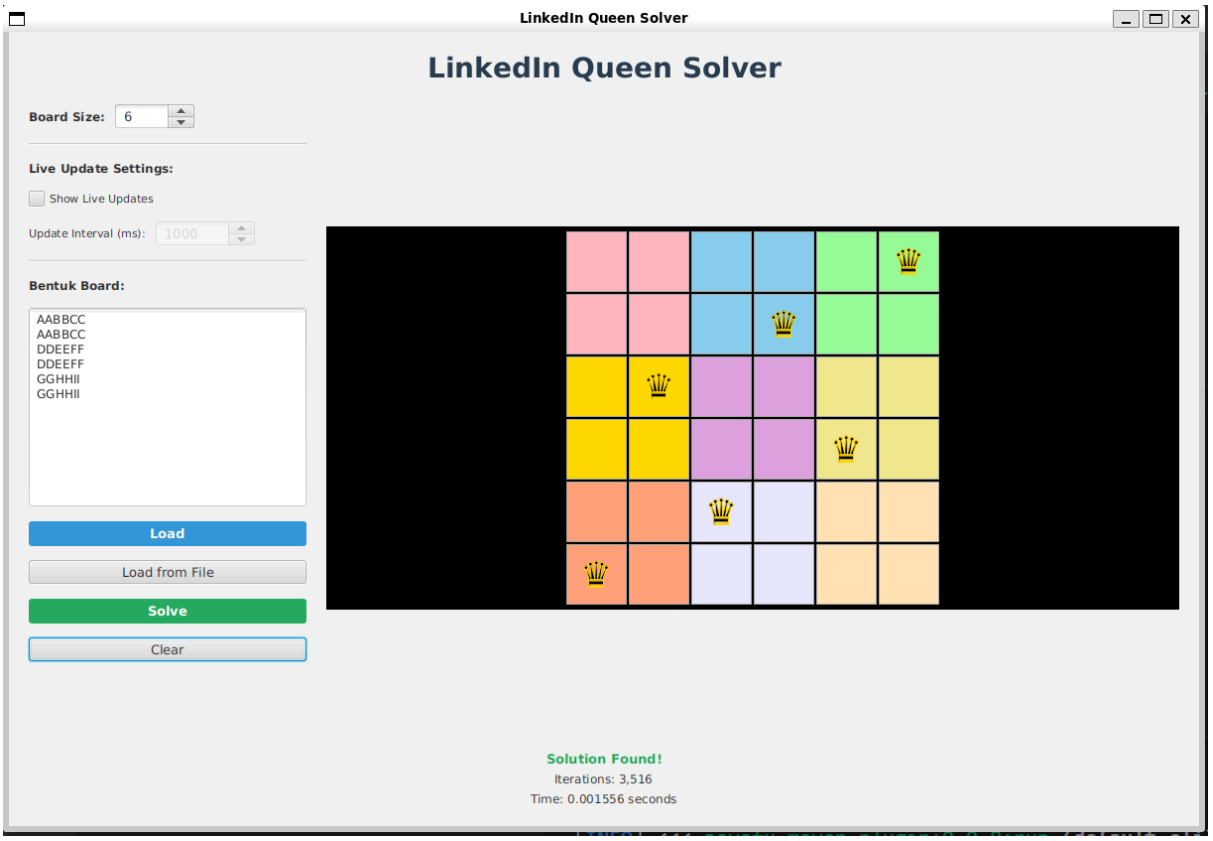


4.2 test2.txt

input:
AABBCC

AABBCC
DDEEFF
DDEEFF
GGHHII
GGHHII

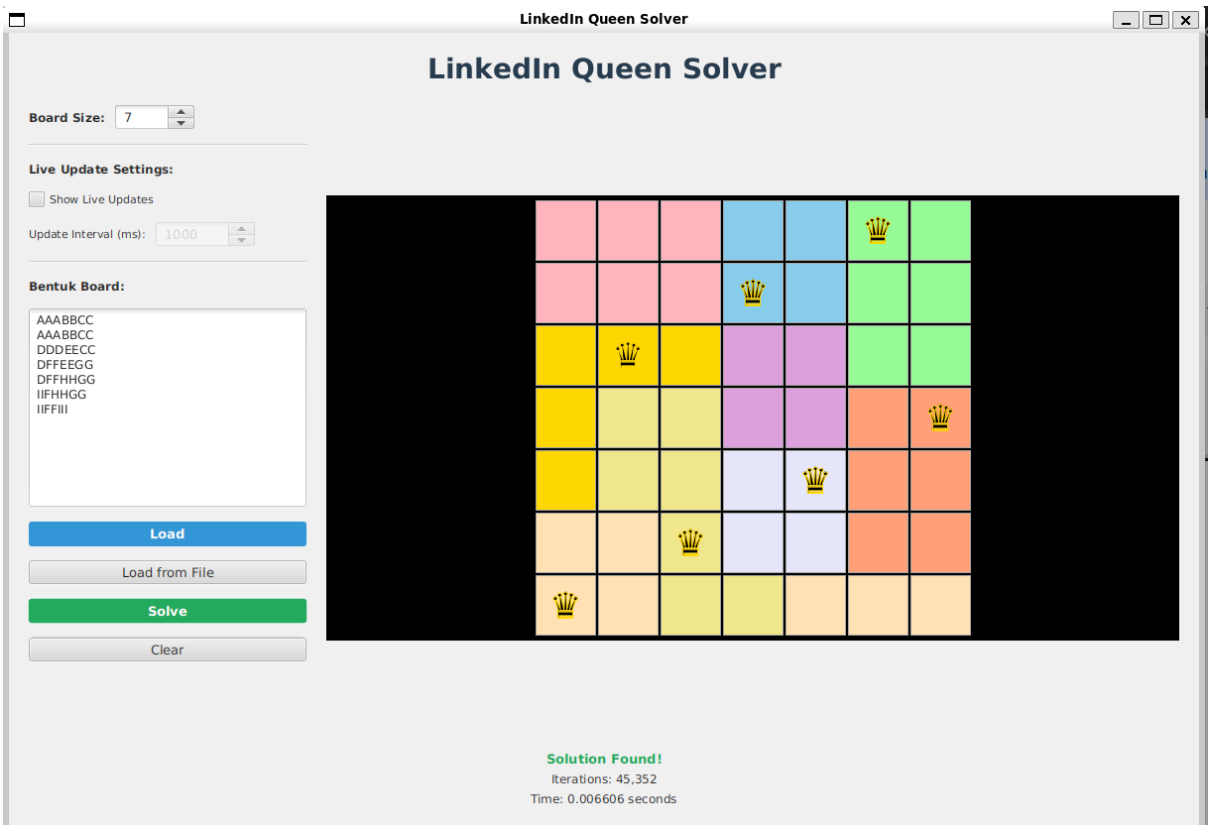
output:



4.3 test3.txt

input:
AAABBCC
AAABBCC
DDDEECC
DFFEEGG
DFFHHGG
IIFHHGG
IIFFIII

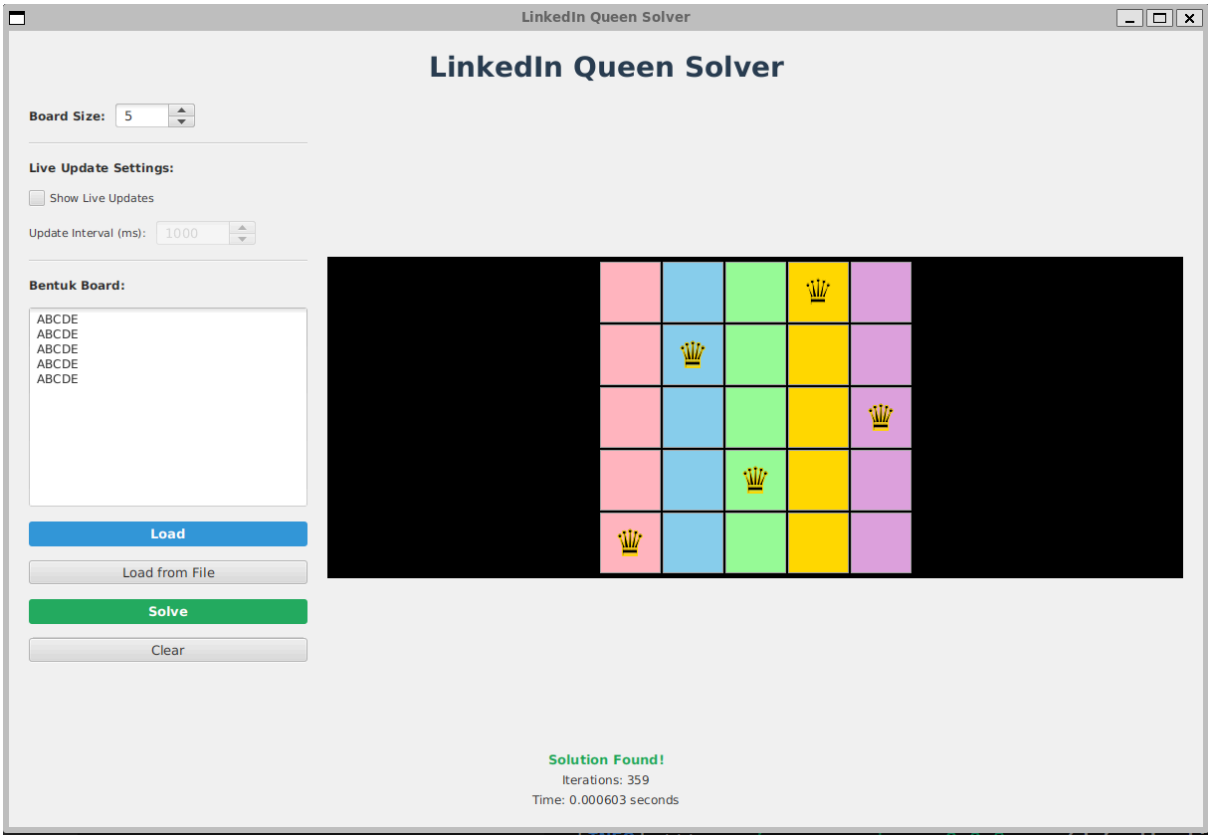
output:



4.4 test4.txt

input:
ABCDE
ABCDE
ABCDE
ABCDE
ABCDE

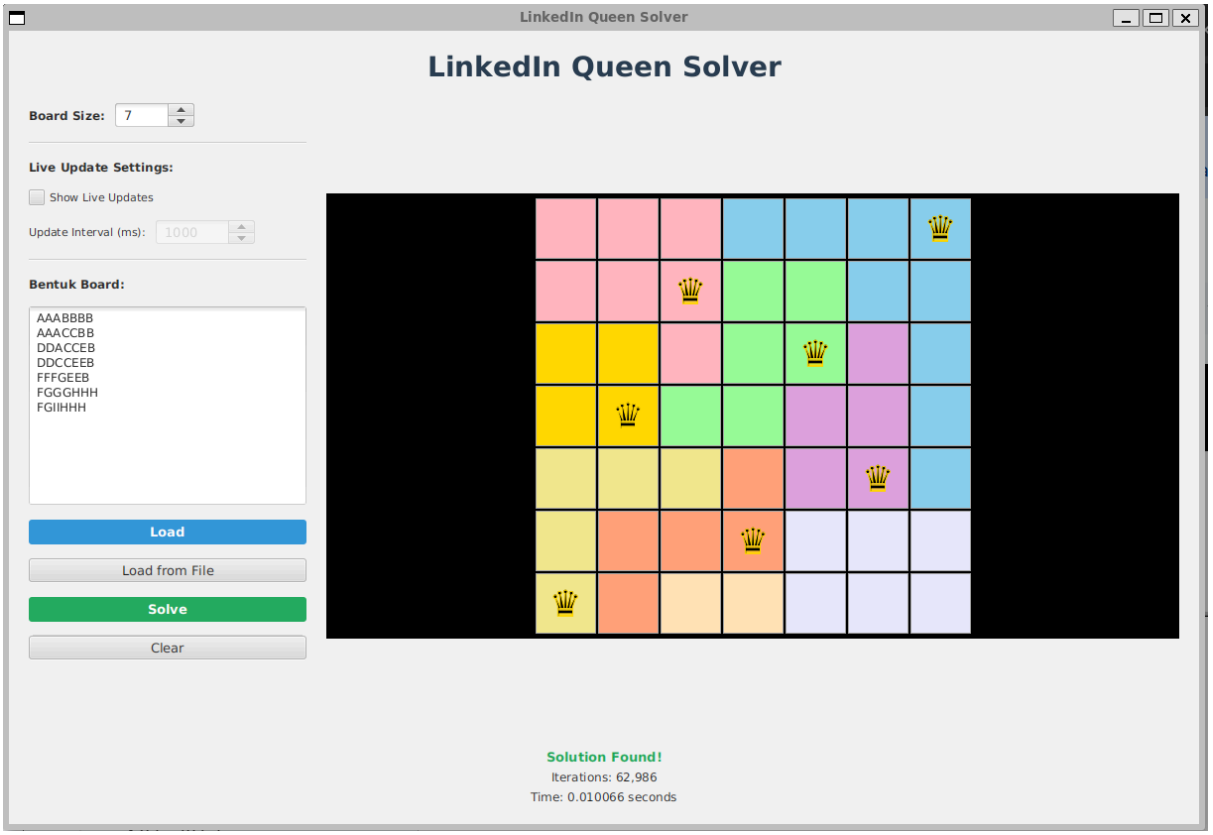
output:



4.5 test5.txt

input:
AAABBBB
AAACCBB
DDACCEB
DDCCEEB
FFFGEEB
FGGGHHH
FGIIHHH

output:


LinkedIn Queen Solver
LinkedIn Queen Solver
Board Size: 7
Live Update Settings:
Show Live Updates
Update Interval (ms): 1000
Bentuk Board:
AAABBBB
AAACCBB
DDACCEB
DDCCEEB
FFFGEEB
FGGGHHH
FGIIHHH
Load
Load from File
Solve
Clear
Solution Found!
Iterations: 62,986
Time: 0.010066 seconds

Lampiran

Github repository: https://github.com/Zekiamani1/Tucil1_13524082

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		✓

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

Zeki Amani