IF2211 - K01

TUGAS KECIL 1 STRATEGI ALGORITMA

Penyelesaian Permainan Queens Linkedin dengan Algoritma Brute Force

Disusun oleh:

Raymond Jonathan Dwi Putra Julianto 13524059

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2026

1 Permasalahan

Queens adalah sebuah permainan logika yang meminta pemain untuk menaruh sebuah poin, pada permainan ini disebut *queen*, untuk diletakkan pada papan yang diberikan. Permainan ini tersedia pada situs jejaring profesional LinkedIn. Tujuan dari permainan ini adalah menempatkan *queen* pada papan yang ada dengan syarat hanya satu *queen* pada tiap baris, kolom, dan daerah warna yang ada. Selain itu, satu *queen* tidak dapat ditempatkan pada *queen* lainnya secara berdekatan dengan jarak 1 kotak. Tujuan dari program ini adalah mencari 1 (satu) solusi valid dari konfigurasi papan yang diberikan atau menyatakan bahwa tidak ada solusi.

2 Pendekatan Solusi

Paradigma yang digunakan dalam menyelesaikan permasalahan ini adalah **Brute-Force**. Paradigma Brute-Force adalah pendekatan penyelesaian suatu masalah dengan mencoba semua kemungkinan percobaan yang ada hingga ketemu solusi yang benar atau optimal. Metode ini mengevaluasi semua kemungkinan yang ada tanpa adanya optimisasi ataupun heuristik yang umumnya memiliki kompleksitas yang tinggi. Hal ini akan menyebabkan program terlalu lama untuk mencari solusi yang ada dan apabila mendapatkan solusi di iterasi awal yang memungkin itu suatu keberuntungan saja.

Pada pencarian solusi dari permainan ini dengan paradigma Brute-Force, algoritma akan mencoba meletakkan satu *queen* pada setiap warna yang ada secara terurut berdasarkan letak koordinat pada papan yang diberikan. Algoritma akan mengecek apakah pada semua penempatan *queen* yang diberikan itu sudah memenuhi atau belum dengan syarat permainan tersebut. Jika terdapat salah satu *queen* yang melanggar peraturan yang ada, algoritma akan mencoba posisi lain dari koordinat warna yang ada.

2.1 Konsep Algoritma

Algoritma akan mencoba membaca masukan dari papan yang ada dan memasukkan data ke program. Karena pada syarat permainan tersebut dijelaskan bahwa tidak diperbolehkan *queen* yang berada pada warna yang sama, maka kita akan memastikan bahwa *queen* tepat satu di satu warna saja dan algoritma hanya mengecek bahwa apakah koordinat sekarang sudah sesuai atau tidak dengan syarat permainan. Hal ini setidaknya menunjukkan algoritma tidak terlalu "bodoh" atau memaksakan untuk menaruh queen di warna yang sama. Program kemudian menyimpan semua data terkait warna yang ada beserta daftar koordinat yang termasuk dalam warna tersebut. Setelah semua data dimasukkan, algoritma akan mencoba semua koordinat yang ada pada setiap warna dan mengecek apakah posisi koordinat dari masing masing *queen* sudah sesuai dengan syarat dari permainan yang ada. Apabila syarat permainan belum terpenuhi, maka algoritma akan mencoba koordinat lain dari daftar koordinat dari warna yang ada. Apabila semua koordinat dari semua warna tidak dapat memenuhi syarat permainan, maka dapat dipastikan bahwa papan yang diberikan tersebut tidak memiliki solusi. Penjelasan dari algoritma ini dapat dibentuk dalam pseudocode sebagai berikut :

```
TYPE Coordinat : < x : integer, y : integer >

TYPE Color : <
    letter : char,
    position : LIST of Coordinat
>

TYPE ListColor : <
    count : integer,
    colors : LIST of Color
>

```

```
TYPE Queen : <
    queens : integer,
    position : LIST of Coordinat
>

FUNCTION ListCheck(x1 : integer, y1 : integer, x2 : integer, y2 : integer) ->
    boolean
{ Mengembalikan true jika posisi aman, false jika melanggar aturan }
ALGORITMA
    { Cek apakah satu baris atau satu kolom }
    IF (x1 = x2) OR (y1 = y2) THEN
        RETURN false
    { Cek apakah berada di kotak sekitar dengan jarak 1 }
    IF (Absolut(x1 - x2) <= 1) AND (Absolut(y1 - y2) <= 1) THEN
        RETURN false
    RETURN true

FUNCTION CheckQueen(ratu : Queen) -> boolean
{ Memeriksa apakah semua ratu dalam list 'ratu' aman satu sama lain }
KAMUS LOKAL
    i, j : integer
    q1, q2 : Coordinat
ALGORITMA
    IF (ratu = null) OR (ratu.position IS EMPTY) THEN
        RETURN false
    FOR i <- 0 TO (Panjang(ratu.position) - 1) DO
        q1 <- ratu.position[i]
        FOR j <- (i + 1) TO (Panjang(ratu.position) - 1) DO
            q2 <- ratu.position[j]
            IF NOT ListCheck(q1.x, q1.y, q2.x, q2.y) THEN
                RETURN false
    RETURN true

FUNCTION Iteration(colorIdx : integer, papan : ListColor, input/output ratu :
    Queen) -> boolean
{ Fungsi untuk mencoba kombinasi posisi }
KAMUS LOKAL
    currentColor : Color
    i : integer
    success : boolean

ALGORITMA
    IF colorIdx >= Panjang(papan.colors) THEN
        casesTried <- casesTried + 1
        IF CheckQueen(ratu) THEN
            RETURN true
        ELSE
            RETURN false

    currentColor <- papan.colors[colorIdx]
    FOR i <- 0 TO (Panjang(currentColor.position) - 1) DO
        InsertLast(ratu.position, currentColor.position[i])
        success <- Iteration(colorIdx + 1, papan, ratu)
        IF success THEN
            RETURN true
        DeleteLast(ratu.position)
    RETURN false

FUNCTION Solve(papan : ListColor, ratu : Queen) -> Queen
{ Mengembalikan objek Queen yang berisi solusi posisi, atau NULL jika gagal }
KAMUS LOKAL
    isFound : boolean

```

```
ALGORITMA
    IF (papan = null) OR (ratu = null) THEN
        RETURN null

    casesTried <- 0
    ratu.position <- CreateNewList()
    isFound <- Iteration(0, papan, ratu)

    IF isFound THEN
        RETURN ratu
    ELSE
        RETURN null
```

Listing 1: Pseudocode Algoritma Brute Force (Backtracking)

3 Implementasi

Program dikembangkan menggunakan bahasa pemrograman **Java** dengan antarmuka grafis **JavaFX**. Program ini juga menggunakan *framework* tambahan, yaitu **Maven**, untuk mempermudah penggunaan **JavaFX**. Struktur program dibagi menjadi beberapa kelas berikut:

3.1 Struktur Data

```
public static int rows;
public static int cols;

    public static class Information{
        public final int iteration;
        public final int time;
        Information(int iteration, int time){
            this.iteration = iteration;
            this.time = time;
        }
    }

    public static class Coordinat{
        int x;
        int y;

        Coordinat(int x, int y){
            this.x = x;
            this.y = y;
        }
    }

    public static class Color{
        char letter;
        List<Coordinat> position;

        Color(char letter) {
            this.letter = letter;
            this.position = new ArrayList<>();
        }
    }

    public static class ListColor{
        int count;
        List<Color> colors;

        ListColor(){
            this.count = 0;
```

```
            this.colors = new ArrayList<>();
        }
    }

    public static class Queen{
        int queens;
        List<Coordinat> position;

        Queen(int queens) {
            this.queens = queens;
            this.position = new ArrayList<>();
        }
    }

    public static class Map{
        Boolean Queen;
        char Letter;
        Map(char Letter){
            this.Letter = Letter;
            this.Queen = false;
        }
    }
```

Listing 2: Kode Struktur Data

Struktur data dibuat dengan terdapat enam data class dan dua data primitif yang memiliki kegunaannya masing-masing.

1. int rows : data primitif yang menyimpan sebuah integer untuk informasi terkait ukuran baris peta.
2. int cols : data primitif yang menyimpan sebuah integer untuk informasi terkait ukuran kolom peta.
3. class Coordinat : class yang berisikan dua data interger yang digunakan dalam menyimpan informasi dari koordinat pada peta.
4. class Color : class yang berisikan data character dan juga class Coordinat yang digunakan dalam menyimpan informasi dari jenis warna tersebut dan juga daftar koordinat yang termasuk dalam warna tersebut.
5. class ListColor : class yang berisikan data integer dan juga class Color yang digunakan dalam menyimpan informasi jumlah warna yang ada dan juga daftar class color yang ada pada peta.
6. class Queen : class yang berisikan data integer dan juga class Coordinat yang digunakan dalam menyimpan informasi jumlah Queen yang ada dan juga daftar koordinat ratu yang ada pada peta.
7. class Map : class yang berisikan data boolean dan character yang digunakan untuk menyimpan informasi pada *cell* setiap matriks pada peta tentang apakah sedang ditempati ratu dan warna pada *cell* tersebut

3.2 Pembacaan Input

```
public static void inputColor (ListColor jenis, BufferedReader br) throws
    IOException{
        if(jenis == null){
            return;
        }
```

```
        List<String> lines = new ArrayList<>();
        String line = br.readLine();

        while (line != null){
            line = line.trim();
            line = line.replaceAll("\\s+", "");
            if (!line.isEmpty()) {
                lines.add(line);
            }
            line = br.readLine();
        }

        if (lines.isEmpty()){
            throw new IOException();
        }

        rows = lines.size();
        cols = lines.get(0).length();

        for(int i = 0; i < rows; i++){
            String lineRow = lines.get(i);
            if (lineRow.length() != cols) {
                throw new IOException();
            }

            for(int j = 0; j < cols; j++){
                char letter = lineRow.charAt(j);
                if (!Character.isLetter(letter) && letter != '.') {
                    throw new IOException();
                }

                addCells(jenis, letter, i, j);
            }
        }
    }

    private static void addCells(ListColor jenis, char letter, int x, int y) {
        for(int i = 0; i < jenis.count; i++){
                if(jenis.colors.get(i).letter == letter){
                    jenis.colors.get(i).position.add(new Coordinat(x, y));
                    return;
                }
            }
        // Untuk count = 0 dan tidak ada dilist saat mendata
        jenis.count++;
        Color newColor = new Color(letter);
        newColor.position.add(new Coordinat(x, y));
        jenis.colors.add(newColor);
    }
```

Listing 3: Kode Pembacaan Input

Fungsi inputColor() digunakan untuk memasukkan semua masukan pengguna ke dalam data yang telah dibuat sebelumnya. Fungsi juga menggunakan fungsi bantuan, yaitu addCells() yang berguna untuk menambahkan data pada setiap cell sesuai dengan color yang ada pada data.

3.3 Kontrol GUI

```
public class GUIController {

    @FXML private TextArea blocksArea;
    @FXML private Label statusLabel, timeLabel, casesLabel;
    @FXML private GridPane boardGrid;
```

```
    @FXML private StackPane boardContainer;
    @FXML private HBox saveBox;
    @FXML private TextField saveNameField;

    @FXML
    public void onLoadClicked(ActionEvent event) {
        FileChooser chooser = new FileChooser();
        chooser.getExtensionFilters().add(new FileChooser.ExtensionFilter("Text files", "*.txt"));
        File file = chooser.showOpenDialog(boardContainer.getScene().getWindow());
        if (file == null) return;

        try (BufferedReader br = Files.newBufferedReader(file.toPath(), StandardCharsets.UTF_8)) {
            String content = Files.readString(file.toPath(), StandardCharsets.UTF_8);
            blocksArea.setText(content);
            boardGrid.getChildren().clear();

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @FXML
    public void onClearClicked(ActionEvent event) {
        blocksArea.clear();
        boardGrid.getChildren().clear();
        saveBox.setVisible(false);
        timeLabel.setText("Waktu: -");
        casesLabel.setText("Iterasi: -");
        statusLabel.setText("Siap.");
    }

    @FXML
    public void onSolveClicked(ActionEvent event) {
        statusLabel.setText("Sedang mencari solusi...");
        saveBox.setVisible(false);
        timeLabel.setText("Waktu: -");
        casesLabel.setText("Iterasi: -");
        boardGrid.getChildren().clear();

        Data.ListColor papan = new Data.ListColor();
        String text = blocksArea.getText();
        if (text == null || text.trim().isEmpty()) {
            statusLabel.setText("Board kosong!");
            return;
        }

        try (BufferedReader br = new BufferedReader(new StringReader(text))) {
            Data.rows = 0; Data.cols = 0;
            Data.inputColor(papan, br);
            if (papan.count == 0) {
                statusLabel.setText("Board kosong!");
                return;
            }
        } catch (IOException e) {
            statusLabel.setText("Gagal memproses input.");
            e.printStackTrace();
            return;
        }

```

```
        Thread worker = new Thread(() -> {
            Solve.onUpdateGUI = (snapshotRatu) -> {
                javafx.application.Platform.runLater(() -> {
                    generateBoard(papan, snapshotRatu);
                    statusLabel.setText("Mencoba kemungkinan...");
                });
            };

            Data.Queen ratu = new Data.Queen(0);
            Data.Queen hasil = Solve.Solve(papan, ratu);
            Data.Information info = Solve.getInfo();

            javafx.application.Platform.runLater(() -> {
                Solve.onUpdateGUI = null;

                if (info != null) {
                    timeLabel.setText("Waktu: " + info.time + " ms");
                    casesLabel.setText("Iterasi: " + info.iteration);
                }

                if (hasil != null) {
                    statusLabel.setText("Solusi Ditemukan!");
                    saveBox.setVisible(true);
                    generateBoard(papan, hasil);
                } else {
                    statusLabel.setText("Solusi Tidak Ditemukan.");
                    generateBoard(papan, new Data.Queen(0));
                }
            });
        });

        worker.setDaemon(true);
        worker.start();
    }

    @FXML
    public void onSaveTxtClicked(ActionEvent event) {
        String fileName = saveNameField.getText();
        if (fileName == null || fileName.trim().isEmpty()) {
            statusLabel.setText("Nama file tidak boleh kosong!");
            return;
        }
        Save.saveTxt(fileName.trim());
        statusLabel.setText("Disimpan: " + fileName + ".txt");
    }

    @FXML
    public void onSaveImgClicked(ActionEvent event) {
        String fileName = saveNameField.getText();
        if (fileName == null || fileName.trim().isEmpty()) {
            statusLabel.setText("Nama file tidak boleh kosong!");
            return;
        }
        Save.saveImg(fileName.trim());
        statusLabel.setText("Disimpan: " + fileName + ".png");
    }

    public void generateBoard(Data.ListColor papan, Data.Queen ratu) {
        boardGrid.getChildren().clear();
        Data.Map = new Data.Cell[Data.rows][Data.cols];

        for (int i = 0; i < papan.colors.size(); i++) {
            Data.Color c = papan.colors.get(i);
```

```
            for (int j = 0; j < c.position.size(); j++) {
                Data.Coordinat coord = c.position.get(j);
                if (coord.x < Data.rows && coord.y < Data.cols) {
                    Data.Map[coord.x][coord.y] = new Data.Cell(c.letter);
                }
            }
        }

        if (ratu != null && ratu.position != null) {
            for (int k = 0; k < ratu.position.size(); k++) {
                Data.Coordinat qPos = ratu.position.get(k);
                if(qPos.x < Data.rows && qPos.y < Data.cols && Data.Map[qPos.x][
    qPos.y] != null){
                    Data.Map[qPos.x][qPos.y].Queen = true;
                }
            }
        }

        for (int i = 0; i < Data.rows; i++) {
            for (int j = 0; j < Data.cols; j++) {
                StackPane cell = new StackPane();
                cell.setPrefSize(40, 40);
                Data.Cell cellData = Data.Map[i][j];
                String colorHex = "#FFFFFF";

                if (cellData != null) {
                    colorHex = Data.getColorHex(cellData.Letter);
                    cell.setStyle("-fx-background-color: " + colorHex + "; -fx-
    border-color: rgba(0,0,0,0.1);");
                    if (cellData.Queen) {
                        Label queenLabel = new Label("   ");
                        queenLabel.setStyle("-fx-font-size: 24px; -fx-text-fill:
     #000000;");
                        cell.getChildren().add(queenLabel);
                    }
                } else {
                    cell.setStyle("-fx-background-color: #ffffff; -fx-border-
    color: rgba(0,0,0,0.1);");
                }
                boardGrid.add(cell, j, i);
            }
        }
    }
}
```

Listing 4: Kode Kontrol GUI

Kontrol GUI dibagi menjadi beberapa fungsi yang mengatur masing-masing komponen GUI yang ada. Fungsi onLoadClicked() berfungsi untuk menerima dan meminta masukan pengguna berupa file .txt ke dalam program. Fungsi onClearClicked() berfungsi untuk menghapus semua informasi yang pada waktu tersebut dan membuat baru semua data yang ada dan meminta masukan pengguna. Fungsi onSolveClicked() berfungsi untuk melakukan pemanggilan fungsi solve yang telah dibuat dan menghasilkan gambar peta pada layar pengguna. Fungsi onSaveTxtClicked() dan onSaveImgClicked() berfungsi untuk menyimpan hasil solusi yang ditemukan ke dalam bentuk .txt ataupun gambar ke dalam penyimpanan. Fungsi generateBoard() digunakan dalam membuat gambaran peta yang ada dari data yang diberikan fungsi pada waktu tersebut, yang berisikan class ListColor dan juga class Queen. Fungsi generateBoard() dan onSaveImgClicked() memanggil fungsi bantuan getColorHex() pada file lain yang bertugas memetakan semua huruf yang ada menjadi warna-warna yang sesuai dengan huruf tersebut.

```
public static final String[] STANDARD_COLORS = {
        "#FF0000", "#00FF00", "#0000FF", "#FFFF00", "#FFA500",
```

```
        "#800080", "#00FFFF", "#FF00FF", "#A52A2A", "#808080",
        "#FFC0CB", "#4B0082", "#800000", "#008080", "#000080",
        "#808000", "#C0C0C0", "#FFD700", "#FA8072", "#40E0D0"
    };

    public static String getColorHex(char c) {
        int charValue = (int) c - 'A';
        int index = (charValue * 7) % STANDARD_COLORS.length;
        if (index < 0){
            index += STANDARD_COLORS.length;
        }
        return STANDARD_COLORS[index];
    }
```

Listing 5: Fungsi Bantuan Menentukan Warna

3.4 Pencarian Solusi

```
public class Solve{
    private static long casesTried = 0;
    private static Data.Information info = null;
    public static Consumer<Data.Queen> onUpdateGUI = null;

    public static Data.Information getInfo() {
        return info;
    }

    public static Data.Queen Solve(Data.ListColor papan, Data.Queen ratu){
        if(papan == null || ratu == null){
            return null;
        }
        casesTried = 0;
        info = null;
        ratu.position = new ArrayList<>();
        long startTime = System.nanoTime();
        if(iteration(0, papan, ratu)){
            long elapsedMs = (System.nanoTime() - startTime) / 1_000_000L;
            int iter = casesTried > Integer.MAX_VALUE ? Integer.MAX_VALUE : (int
    ) casesTried;
            int time = elapsedMs > Integer.MAX_VALUE ? Integer.MAX_VALUE : (int)
     elapsedMs;
            info = new Data.Information(iter, time);
            return ratu;
        }
        long elapsedMs = (System.nanoTime() - startTime) / 1_000_000L;
        int iter = casesTried > Integer.MAX_VALUE ? Integer.MAX_VALUE : (int)
    casesTried;
        int time = elapsedMs > Integer.MAX_VALUE ? Integer.MAX_VALUE : (int)
    elapsedMs;
        info = new Data.Information(iter, time);

        return null;
    }

    // Brute-Force algorithm
    private static boolean iteration(int colorIdx, Data.ListColor papan, Data.
    Queen ratu) {
        if (colorIdx >= papan.colors.size()) {
            casesTried++;
            if (casesTried % 1000 == 0 && onUpdateGUI != null) { // Update GUI

                Data.Queen snapshot = new Data.Queen(ratu.queens);
                snapshot.position = new ArrayList<>(ratu.position);
```

```
41
42                  onUpdateGUI.accept(snapshot);
43
44                  try { Thread.sleep(1); } catch (InterruptedException e) {}
45              }
46              if (CheckQueen(ratu)) {
47                  return true;
48              }
49              return false;
50          }
51
52          Data.Color currentColor = papan.colors.get(colorIdx);
53
54          for (int i = 0; i < currentColor.position.size(); i++) {
55              ratu.position.add(currentColor.position.get(i));
56
57              boolean success = iteration(colorIdx + 1, papan, ratu);
58
59              if (success){
60                  return true;
61              }
62              ratu.position.remove(ratu.position.size() - 1);
63          }
64          return false;
65      }
66
67      private static boolean CheckQueen(Data.Queen ratu){
68          if(ratu == null || ratu.position == null || ratu.position.isEmpty()){
69          return false;
70          }
71
72          int queenCounts = ratu.position.size();
73          for(int i = 0; i < queenCounts; i++){
74          Data.Coordinat currentQueen = ratu.position.get(i);
75          int x = currentQueen.x;
76          int y = currentQueen.y;
77              for(int j = i + 1; j < queenCounts ;j++){
78                  Data.Coordinat checkQueen = ratu.position.get(j);
79                  int x2 = checkQueen.x;
80                  int y2 = checkQueen.y;
81                  if(!ListCheck(x,y,x2,y2)){
82                      return false;
83                  }
84              }
85          }
86          return true;
87      }
88
89      private static boolean ListCheck(int x1, int y1, int x2, int y2){
90          if(x1 == x2 || y1 == y2){ // terletak baris dan kolom sama
91              return false;
92          }
93          if(Math.abs(x1 - x2) <= 1 && Math.abs(y1 - y2) <= 1){
94              // terletak disekitar dengan jarak satu kotak
95              return false;
96          }
97          return true;
98      }
99  }
```

Listing 6: Kode Pencarian Solusi

Pencarian solusi dilakukan oleh fungsi Solve() yang berguna untuk menyimpan informasi terkait jumlah percobaan yang telah dicari, lama waktu berjalan program, dan memulai percobaan

kombinasi yang ada. Fungsi ini memanggil fungsi iteration() untuk mencoba semua kemungkinan yang menggunakan algoritma yang telah dijelaskan sebelumnya. Selain itu, setiap 1000 kali percobaan yang telah dilakukan akan ditampilkan peta yang ada dan ditampilkan pada GUI. Semua kemungkinan yang ada ini akan dicek menggunakan fungsi CheckQueen() yang mengecek apakah semua letak queen yang ada sudah sesuai dengan syarat permainan dengan bantuan fungsi ListCheck(). Apabila semua percobaan tidak ada yang sesuai dengan syarat, maka akan memberikan informasi ke GUI bahwa tidak ada solusi pada peta tersebut.

3.5 Penyimpanan Hasil

```
public class Save {
    private static final String OUTPUT_DIR = "test" + File.separator;

    public static void saveTxt(String fileName) {
        if (Data.Map == null || Data.rows == 0 || Data.cols == 0) {
            System.out.println("Tidak ada data untuk disimpan.");
            return;
        }

        try {
            File directory = new File(OUTPUT_DIR);
            if (!directory.exists()) {
                directory.mkdirs();
            }
            File file = new File(OUTPUT_DIR + fileName + ".txt");
            FileWriter writer = new FileWriter(file);

            for (int i = 0; i < Data.rows; i++) {
                for (int j = 0; j < Data.cols; j++) {
                    Data.Cell cell = Data.Map[i][j];
                    if (cell != null) {
                        if (cell.Queen) {
                            writer.write("#");
                        } else {
                            writer.write(cell.Letter);
                        }
                    } else {
                        writer.write(".");
                    }
                }
                writer.write("\n");
            }
            writer.write("\n");
            Data.Information info = Solve.getInfo();
            if (info != null) {
                writer.write("Waktu pencarian: " + info.time + " ms\n");
                writer.write("Banyak kasus yang ditinjau: " + info.iteration + "
    \n");
            } else {
                writer.write("Waktu pencarian: - ms\n");
                writer.write("Banyak kasus yang ditinjau: -\n");
            }

            writer.close();
            System.out.println("File TXT berhasil disimpan ke: " + file.
    getAbsolutePath());
        } catch (IOException e) {
            System.out.println("Gagal menyimpan file TXT.");
            e.printStackTrace();
        }
    }
```

```
50
51      public static void saveImg(String fileName) {
52          if (Data.Map == null || Data.rows == 0 || Data.cols == 0) {
53              System.out.println("Tidak ada data matriks untuk digambar.");
54              return;
55          }
56
57          try {
58              File directory = new File(OUTPUT_DIR);
59              if (!directory.exists()) {
60                  directory.mkdirs();
61              }
62
63              int cellSize = 60;
64              int width = Data.cols * cellSize;
65              int height = Data.rows * cellSize;
66              BufferedImage image = new BufferedImage(width, height, BufferedImage
    .TYPE_INT_RGB);
67              Graphics2D g2d = image.createGraphics();
68              g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints
    .VALUE_ANTIALIAS_ON);
69              g2d.setRenderingHint(RenderingHints.KEY_TEXT_ANTIALIASING,
    RenderingHints.VALUE_TEXT_ANTIALIAS_ON);
70
71              for (int i = 0; i < Data.rows; i++) {
72                  for (int j = 0; j < Data.cols; j++) {
73                      Data.Cell cell = Data.Map[i][j];
74                      int x = j * cellSize;
75                      int y = i * cellSize;
76                      if (cell != null) {
77                          String hexColor = Data.getColorHex(cell.Letter);
78                          g2d.setColor(Color.decode(hexColor));
79                          g2d.fillRect(x, y, cellSize, cellSize);
80                          g2d.setColor(new Color(0, 0, 0, 50));
81                          g2d.drawRect(x, y, cellSize, cellSize);
82                          if (cell.Queen) {
83                              g2d.setColor(Color.BLACK);
84                              Font font = new Font("SansSerif", Font.BOLD,
    cellSize / 2 + 10);
85                              g2d.setFont(font);
86                              String symbol = "   ";
87                              FontMetrics metrics = g2d.getFontMetrics(font);
88                              int xText = x + (cellSize - metrics.stringWidth(
    symbol)) / 2;
89                              int yText = y + ((cellSize - metrics.getHeight()) /
    2) + metrics.getAscent();
90                              g2d.drawString(symbol, xText, yText);
91                          }
92                      } else {
93                          g2d.setColor(Color.WHITE);
94                          g2d.fillRect(x, y, cellSize, cellSize);
95                      }
96                  }
97              }
98
99              g2d.dispose();
100             File file = new File(OUTPUT_DIR + fileName + ".png");
101             ImageIO.write(image, "png", file);
102             System.out.println("Gambar PNG (Matriks) disimpan: " + file.
    getAbsolutePath());
103         } catch (IOException e) {
104             System.out.println("Gagal menyimpan file PNG.");
105             e.printStackTrace();
```

```
106             }
107         }
108 }
```

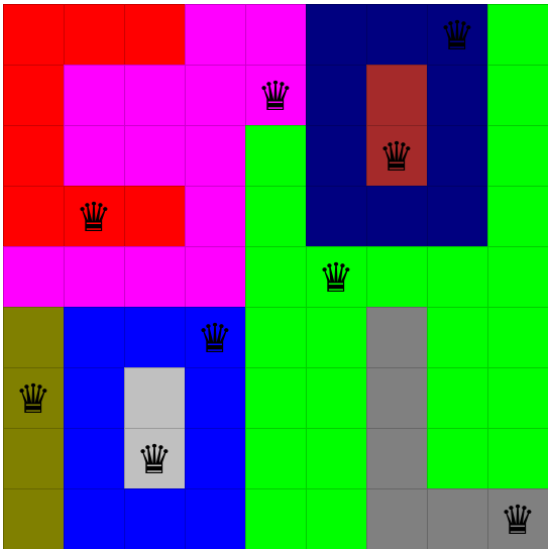
Listing 7: Kode Penyimpanan Hasil

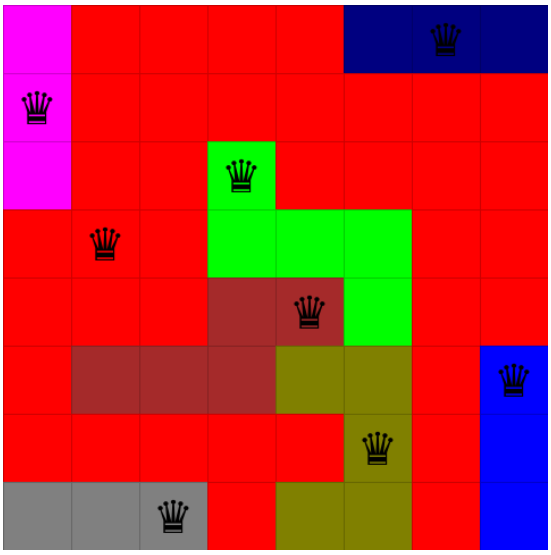
Penyimpanan hasil dari solusi yang didapatkan akan dilakukan oleh fungsi saveTxt() dan saveImg() yang akan menghasilkan file dalam bentuk .txt dan juga image sesuai dengan kebutuhan dari pengguna inginkan.

4 Pengujian

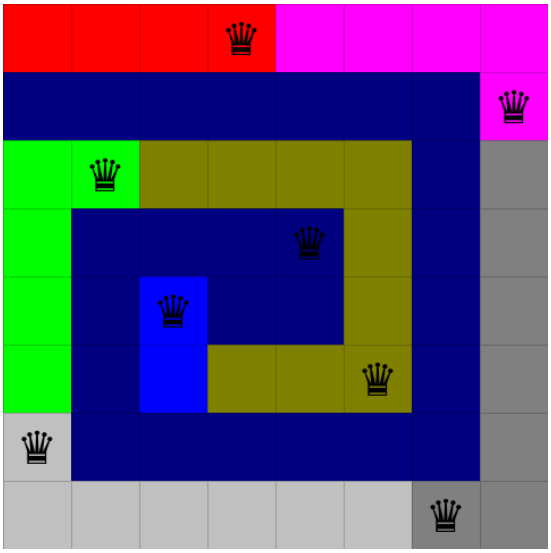
Berikut adalah hasil pengujian program terhadap beberapa kasus uji. Pengujian ini dilakukan pada lingkungan operating system linux pada distro Ubuntu.

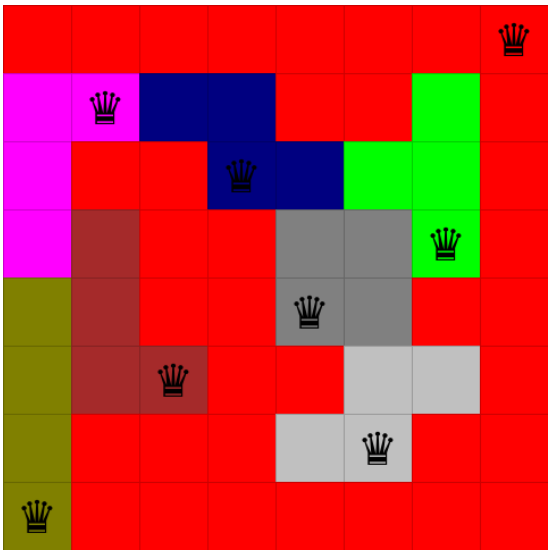
Masukan	Keluaran

Test Case 1: Solusi Ditemukan
File Input: `test1.txt`
 	
`AAABBCCCD`
`ABBBBCECD`
`ABBBDCECD`
`AAABDCCCD`
`BBBBDDDDD`
`FGGGDDHDD`
`FGIGDDHDD`
`FGIGDDHDD`
`FGGGDDHHH`
	
	Status: Solusi Ditemukan
	Waktu: 6715 ms
	Iterasi: 21,415,356

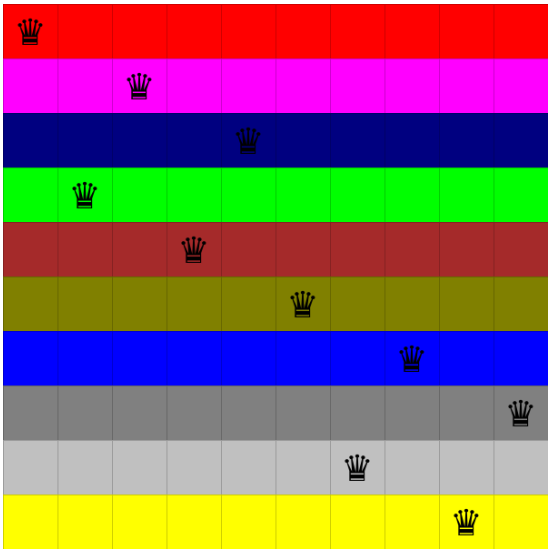
Test Case 2: Solusi Ditemukan
File Input : test2.txt
`BAAAACCC`
`BAAAAAAA`
`BAADAAAA`
`AAADDDAA`
`AAAEEDAA`
`AEEEFFAG`
`AAAAAFAG`
`HHHAFFAG`
	
	Status: Solusi Ditemukan
	Waktu: 60 ms
	Iterasi: 186,816

Masukan	Keluaran

Test Case 3: Solusi Ditemukan
File Input : test3.txt
`AAAABBBB`
`CCCCCCCB`
`DDEEEECH`
`DCCCCECH`
`DCFCCECH`
`DCFEEECH`
`ICCCCCCH`
`IIIIIIHH`
	
	Status: Solusi Ditemukan
	Waktu: 674 ms
	Iterasi: 2,144,997

Test Case 4: Solusi Ditemukan
File Input : test4.txt
`AAAAAAAA`
`BBCCAADA`
`BAACCDDA`
`BEAAHHDA`
`FEAAHHAA`
`FEEAAIIA`
`FAAAIIAA`
`FAAAAAAA`
	
	Status: Solusi Ditemukan
	Waktu: 46 ms
	Iterasi: 121,840

Masukan	Keluaran

Test Case 5: Solusi Ditemukan
File Input : test5.txt
`AAAAAAAAAA`
`BBBBBBBBBB`
`CCCCCCCCCC`
`DDDDDDDDDD`
`EEEEEEEEEE`
`FFFFFFFFFF`
`GGGGGGGGGG`
`HHHHHHHHHH`
`IIIIIIIIII`
`JJJJJJJJJJ`
	
	Status: Solusi Ditemukan
	Waktu: 74802 ms
	Iterasi: 241,357,969

Test Case 6: Kesalahan Masukan
File Input : test6.txt
`AAAAA`
`BBBBB`
`CC CC`
`DDDDD`
`EEEEE`
	Tidak ada Gambar
	Status: -
	Waktu: - ms
	Iterasi: -

Test Case 7: Tidak Ada Solusi
File Input : test7.txt
`BBBFFF`
`BBCCAA`
`CCCAAA`
`DDDDDA`
`EEEDAA`
`DDDEEE`
	
	Status: Tidak Ada Solusi
	Iterasi: 32,400

5 Lampiran

No	Poin	Ya	Tidak
1	Program berhasil dikompilasi tanpa kesalahan	✓	
2	Program berhasil dijalankan	✓	
3	Solusi yang diberikan program benar dan mematuhi aturan permainan	✓	
4	Program dapat membaca masukan berkas .txt serta menyimpan solusi dalam berkas .txt	✓	
5	Program memiliki Graphical User Interface (GUI)	✓	
6	Program dapat menyimpan solusi dalam bentuk file gambar	✓	

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

Raymond Jonathan Dwi Putra Julianto

Tautan Repositori: `https://github.com/Alpaomega1136/Tucil1_13524059`