

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 memungkinkan 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 :

```
1 TYPE Coordinat : < x : integer, y : integer >
2
3 TYPE Color : <
4     letter : char,
5     position : LIST of Coordinat
6 >
7
8 TYPE ListColor : <
9     count : integer,
10    colors : LIST of Color
11 >
12
```

```

13 TYPE Queen : <
14     queens : integer,
15     position : LIST of Coordinat
16 >
17
18 FUNCTION ListCheck(x1 : integer, y1 : integer, x2 : integer, y2 : integer) ->
    boolean
19 { Mengembalikan true jika posisi aman, false jika melanggar aturan }
20 ALGORITMA
21     { Cek apakah satu baris atau satu kolom }
22     IF (x1 = x2) OR (y1 = y2) THEN
23         RETURN false
24     { Cek apakah berada di kotak sekitar dengan jarak 1 }
25     IF (Absolut(x1 - x2) <= 1) AND (Absolut(y1 - y2) <= 1) THEN
26         RETURN false
27     RETURN true
28
29 FUNCTION CheckQueen(ratu : Queen) -> boolean
30 { Memeriksa apakah semua ratu dalam list 'ratu' aman satu sama lain }
31 KAMUS LOKAL
32     i, j : integer
33     q1, q2 : Coordinat
34 ALGORITMA
35     IF (ratu = null) OR (ratu.position IS EMPTY) THEN
36         RETURN false
37     FOR i <- 0 TO (Panjang(ratu.position) - 1) DO
38         q1 <- ratu.position[i]
39         FOR j <- (i + 1) TO (Panjang(ratu.position) - 1) DO
40             q2 <- ratu.position[j]
41             IF NOT ListCheck(q1.x, q1.y, q2.x, q2.y) THEN
42                 RETURN false
43         RETURN true
44
45 FUNCTION Iteration(colorIdx : integer, papan : ListColor, input/output ratu :
    Queen) -> boolean
46 { Fungsi untuk mencoba kombinasi posisi }
47 KAMUS LOKAL
48     currentColor : Color
49     i : integer
50     success : boolean
51
52 ALGORITMA
53     IF colorIdx >= Panjang(papan.colors) THEN
54         casesTried <- casesTried + 1
55         IF CheckQueen(ratu) THEN
56             RETURN true
57         ELSE
58             RETURN false
59
60     currentColor <- papan.colors[colorIdx]
61     FOR i <- 0 TO (Panjang(currentColor.position) - 1) DO
62         InsertLast(ratu.position, currentColor.position[i])
63         success <- Iteration(colorIdx + 1, papan, ratu)
64         IF success THEN
65             RETURN true
66         DeleteLast(ratu.position)
67     RETURN false
68
69 FUNCTION Solve(papan : ListColor, ratu : Queen) -> Queen
70 { Mengembalikan objek Queen yang berisi solusi posisi, atau NULL jika gagal }
71 KAMUS LOKAL
72     isFound : boolean
73

```

```

74 ALGORITMA
75     IF (papan = null) OR (ratu = null) THEN
76         RETURN null
77
78     casesTried <- 0
79     ratu.position <- CreateNewList()
80     isFound <- Iteration(0, papan, ratu)
81
82     IF isFound THEN
83         RETURN ratu
84     ELSE
85         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

```

1 public static int rows;
2 public static int cols;
3
4     public static class Information{
5         public final int iteration;
6         public final int time;
7         Information(int iteration, int time){
8             this.iteration = iteration;
9             this.time = time;
10        }
11    }
12
13    public static class Coordinat{
14        int x;
15        int y;
16
17        Coordinat(int x, int y){
18            this.x = x;
19            this.y = y;
20        }
21    }
22
23    public static class Color{
24        char letter;
25        List<Coordinat> position;
26
27        Color(char letter) {
28            this.letter = letter;
29            this.position = new ArrayList<>();
30        }
31    }
32
33    public static class ListColor{
34        int count;
35        List<Color> colors;
36
37        ListColor(){
38            this.count = 0;

```

```

39         this.colors = new ArrayList<>();
40     }
41 }
42
43 public static class Queen{
44     int queens;
45     List<Coordinat> position;
46
47     Queen(int queens) {
48         this.queens = queens;
49         this.position = new ArrayList<>();
50     }
51 }
52
53 public static class Map{
54     Boolean Queen;
55     char Letter;
56     Map(char Letter){
57         this.Letter = Letter;
58         this.Queen = false;
59     }
60 }

```

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

```

1 public static void inputColor (ListColor jenis, BufferedReader br) throws
   IOException{
2     if(jenis == null){
3         return;
4     }

```

```

5      List<String> lines = new ArrayList<>();
6      String line = br.readLine();
7
8      while (line != null){
9          line = line.trim();
10         line = line.replaceAll("\\s+", "");
11         if (!line.isEmpty()) {
12             lines.add(line);
13         }
14         line = br.readLine();
15     }
16
17     if (lines.isEmpty()){
18         throw new IOException();
19     }
20
21     rows = lines.size();
22     cols = lines.get(0).length();
23
24     for(int i = 0; i < rows; i++){
25         String lineRow = lines.get(i);
26         if (lineRow.length() != cols) {
27             throw new IOException();
28         }
29
30         for(int j = 0; j < cols; j++){
31             char letter = lineRow.charAt(j);
32             if (!Character.isLetter(letter) && letter != ',') {
33                 throw new IOException();
34             }
35
36             addCells(jenis, letter, i, j);
37         }
38     }
39 }
40
41 private static void addCells(ListColor jenis, char letter, int x, int y) {
42     for(int i = 0; i < jenis.count; i++){
43         if(jenis.colors.get(i).letter == letter){
44             jenis.colors.get(i).position.add(new Coordinat(x, y));
45             return;
46         }
47     }
48     // Untuk count = 0 dan tidak ada dilist saat mendata
49     jenis.count++;
50     Color newColor = new Color(letter);
51     newColor.position.add(new Coordinat(x, y));
52     jenis.colors.add(newColor);
53 }

```

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

```

1 public class GUIController {
2
3     @FXML private TextArea blocksArea;
4     @FXML private Label statusLabel, timeLabel, casesLabel;
5     @FXML private GridPane boardGrid;

```

```

6      @FXML private StackPane boardContainer;
7      @FXML private HBox saveBox;
8      @FXML private TextField saveNameField;
9
10     @FXML
11     public void onLoadClicked(ActionEvent event) {
12         FileChooser chooser = new FileChooser();
13         chooser.getExtensionFilters().add(new FileChooser.ExtensionFilter("Text
files", "*.txt"));
14         File file = chooser.showOpenDialog(boardContainer.getScene().getWindow()
);
15         if (file == null) return;
16
17         try (BufferedReader br = Files.newBufferedReader(file.toPath(),
StandardCharsets.UTF_8)) {
18             String content = Files.readString(file.toPath(), StandardCharsets.
UTF_8);
19             blocksArea.setText(content);
20             boardGrid.getChildren().clear();
21
22             } catch (IOException e) {
23                 e.printStackTrace();
24             }
25     }
26
27     @FXML
28     public void onClearClicked(ActionEvent event) {
29         blocksArea.clear();
30         boardGrid.getChildren().clear();
31         saveBox.setVisible(false);
32         timeLabel.setText("Waktu: -");
33         casesLabel.setText("Iterasi: -");
34         statusLabel.setText("Siap.");
35     }
36
37     @FXML
38     public void onSolveClicked(ActionEvent event) {
39         statusLabel.setText("Sedang mencari solusi...");
40         saveBox.setVisible(false);
41         timeLabel.setText("Waktu: -");
42         casesLabel.setText("Iterasi: -");
43         boardGrid.getChildren().clear();
44
45         Data.ListColor papan = new Data.ListColor();
46         String text = blocksArea.getText();
47         if (text == null || text.trim().isEmpty()) {
48             statusLabel.setText("Board kosong!");
49             return;
50         }
51
52         try (BufferedReader br = new BufferedReader(new StringReader(text))) {
53             Data.rows = 0; Data.cols = 0;
54             Data.inputColor(papan, br);
55             if (papan.count == 0) {
56                 statusLabel.setText("Board kosong!");
57                 return;
58             }
59         } catch (IOException e) {
60             statusLabel.setText("Gagal memproses input.");
61             e.printStackTrace();
62             return;
63         }
64     }

```

```

65     Thread worker = new Thread(() -> {
66         Solve.onUpdateGUI = (snapshotRatu) -> {
67             javafx.application.Platform.runLater(() -> {
68                 generateBoard(papan, snapshotRatu);
69                 statusLabel.setText("Mencoba kemungkinan...");
70             });
71         };
72
73         Data.Queen ratu = new Data.Queen(0);
74         Data.Queen hasil = Solve.Solve(papan, ratu);
75         Data.Information info = Solve.getInfo();
76
77         javafx.application.Platform.runLater(() -> {
78             Solve.onUpdateGUI = null;
79
80             if (info != null) {
81                 timeLabel.setText("Waktu: " + info.time + " ms");
82                 casesLabel.setText("Iterasi: " + info.iteration);
83             }
84
85             if (hasil != null) {
86                 statusLabel.setText("Solusi Ditemukan!");
87                 saveBox.setVisible(true);
88                 generateBoard(papan, hasil);
89             } else {
90                 statusLabel.setText("Solusi Tidak Ditemukan.");
91                 generateBoard(papan, new Data.Queen(0));
92             }
93         });
94     });
95
96     worker.setDaemon(true);
97     worker.start();
98 }
99
100 @FXML
101 public void onSaveTxtClicked(ActionEvent event) {
102     String fileName = saveNameField.getText();
103     if (fileName == null || fileName.trim().isEmpty()) {
104         statusLabel.setText("Nama file tidak boleh kosong!");
105         return;
106     }
107     Save.saveTxt(fileName.trim());
108     statusLabel.setText("Disimpan: " + fileName + ".txt");
109 }
110
111 @FXML
112 public void onSaveImgClicked(ActionEvent event) {
113     String fileName = saveNameField.getText();
114     if (fileName == null || fileName.trim().isEmpty()) {
115         statusLabel.setText("Nama file tidak boleh kosong!");
116         return;
117     }
118     Save.saveImg(fileName.trim());
119     statusLabel.setText("Disimpan: " + fileName + ".png");
120 }
121
122 public void generateBoard(Data.ListColor papan, Data.Queen ratu) {
123     boardGrid.getChildren().clear();
124     Data.Map = new Data.Cell[Data.rows][Data.cols];
125
126     for (int i = 0; i < papan.colors.size(); i++) {
127         Data.Color c = papan.colors.get(i);

```



```

128         for (int j = 0; j < c.position.size(); j++) {
129             Data.Coordinat coord = c.position.get(j);
130             if (coord.x < Data.rows && coord.y < Data.cols) {
131                 Data.Map[coord.x][coord.y] = new Data.Cell(c.letter);
132             }
133         }
134     }
135
136     if (ratu != null && ratu.position != null) {
137         for (int k = 0; k < ratu.position.size(); k++) {
138             Data.Coordinat qPos = ratu.position.get(k);
139             if (qPos.x < Data.rows && qPos.y < Data.cols && Data.Map[qPos.x][
qPos.y] != null){
140                 Data.Map[qPos.x][qPos.y].Queen = true;
141             }
142         }
143     }
144
145     for (int i = 0; i < Data.rows; i++) {
146         for (int j = 0; j < Data.cols; j++) {
147             StackPane cell = new StackPane();
148             cell.setPrefSize(40, 40);
149             Data.Cell cellData = Data.Map[i][j];
150             String colorHex = "#FFFFFF";
151
152             if (cellData != null) {
153                 colorHex = Data.getColorHex(cellData.Letter);
154                 cell.setStyle("-fx-background-color: " + colorHex + "; -fx-
border-color: rgba(0,0,0,0.1);");
155                 if (cellData.Queen) {
156                     Label queenLabel = new Label(" ");
157                     queenLabel.setStyle("-fx-font-size: 24px; -fx-text-fill:
#000000;");
158                     cell.getChildren().add(queenLabel);
159                 }
160                 } else {
161                     cell.setStyle("-fx-background-color: #ffffff; -fx-border-
color: rgba(0,0,0,0.1);");
162                 }
163                 boardGrid.add(cell, j, i);
164             }
165         }
166     }
167 }

```

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.

```

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

```

```

3      "#800080", "#00FFFF", "#FF00FF", "#A52A2A", "#808080",
4      "#FFC0CB", "#4B0082", "#800000", "#008080", "#000080",
5      "#808000", "#C0C0C0", "#FFD700", "#FA8072", "#40E0D0"
6  };
7
8  public static String getColorHex(char c) {
9      int charValue = (int) c - 'A';
10     int index = (charValue * 7) % STANDARD_COLORS.length;
11     if (index < 0){
12         index += STANDARD_COLORS.length;
13     }
14     return STANDARD_COLORS[index];
15 }

```

Listing 5: Fungsi Bantuan Menentukan Warna

3.4 Pencarian Solusi

```

1 public class Solve{
2     private static long casesTried = 0;
3     private static Data.Information info = null;
4     public static Consumer<Data.Queen> onUpdateGUI = null;
5
6     public static Data.Information getInfo() {
7         return info;
8     }
9
10    public static Data.Queen Solve(Data.ListColor papan, Data.Queen ratu){
11        if(papan == null || ratu == null){
12            return null;
13        }
14        casesTried = 0;
15        info = null;
16        ratu.position = new ArrayList<>();
17        long startTime = System.nanoTime();
18        if(iteration(0, papan, ratu)){
19            long elapsedMs = (System.nanoTime() - startTime) / 1_000_000L;
20            int iter = casesTried > Integer.MAX_VALUE ? Integer.MAX_VALUE : (int)
21            casesTried;
22            int time = elapsedMs > Integer.MAX_VALUE ? Integer.MAX_VALUE : (int)
23            elapsedMs;
24            info = new Data.Information(iter, time);
25            return ratu;
26        }
27        long elapsedMs = (System.nanoTime() - startTime) / 1_000_000L;
28        int iter = casesTried > Integer.MAX_VALUE ? Integer.MAX_VALUE : (int)
29        casesTried;
30        int time = elapsedMs > Integer.MAX_VALUE ? Integer.MAX_VALUE : (int)
31        elapsedMs;
32        info = new Data.Information(iter, time);
33
34        return null;
35    }
36
37    // Brute-Force algorithm
38    private static boolean iteration(int colorIdx, Data.ListColor papan, Data.
39    Queen ratu) {
40        if (colorIdx >= papan.colors.size()) {
41            casesTried++;
42            if (casesTried % 1000 == 0 && onUpdateGUI != null) { // Update GUI
43
44                Data.Queen snapshot = new Data.Queen(ratu.queens);
45                snapshot.position = new ArrayList<>(ratu.position);

```

```

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

```

1 public class Save {
2     private static final String OUTPUT_DIR = "test" + File.separator;
3
4     public static void saveTxt(String fileName) {
5         if (Data.Map == null || Data.rows == 0 || Data.cols == 0) {
6             System.out.println("Tidak ada data untuk disimpan.");
7             return;
8         }
9
10        try {
11            File directory = new File(OUTPUT_DIR);
12            if (!directory.exists()) {
13                directory.mkdirs();
14            }
15            File file = new File(OUTPUT_DIR + fileName + ".txt");
16            FileWriter writer = new FileWriter(file);
17
18            for (int i = 0; i < Data.rows; i++) {
19                for (int j = 0; j < Data.cols; j++) {
20                    Data.Cell cell = Data.Map[i][j];
21                    if (cell != null) {
22                        if (cell.Queen) {
23                            writer.write("#");
24                        } else {
25                            writer.write(cell.Letter);
26                        }
27                    } else {
28                        writer.write(".");
29                    }
30                }
31                writer.write("\n");
32            }
33            writer.write("\n");
34            Data.Information info = Solve.getInfo();
35            if (info != null) {
36                writer.write("Waktu pencarian: " + info.time + " ms\n");
37                writer.write("Banyak kasus yang ditinjau: " + info.iteration + "
38\n");
39            } else {
40                writer.write("Waktu pencarian: - ms\n");
41                writer.write("Banyak kasus yang ditinjau: -\n");
42            }
43
44            writer.close();
45            System.out.println("File TXT berhasil disimpan ke: " + file.
46getAbsolutePath());
47        } catch (IOException e) {
48            System.out.println("Gagal menyimpan file TXT.");
49            e.printStackTrace();
50        }
51    }
52 }

```

```

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();

```

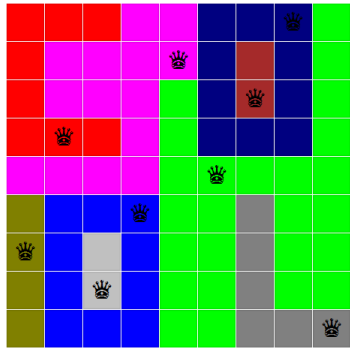
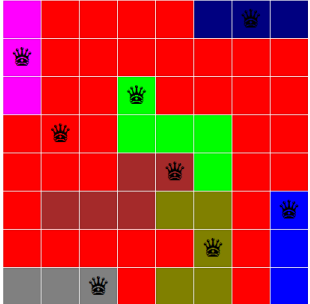
```
106     }  
107 }  
108 }
```

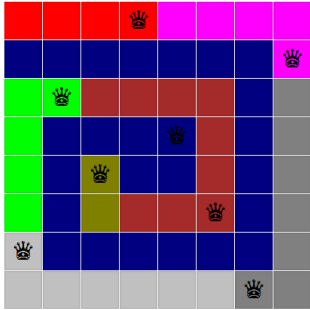
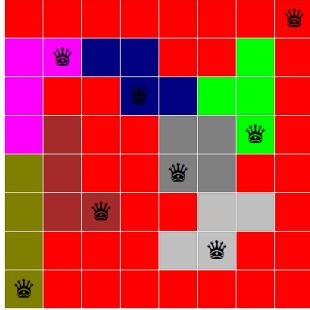
Listing 7: Kode Penyimpanan Hasil


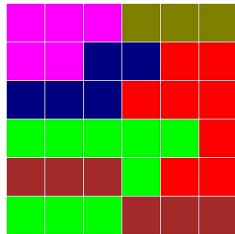
Penyimpanan hasil dari solusi yang didapatkan akan dilakukan oleh fungsi `saveTxt()` dan `saveImg()` yang akan menghasilkan file dalam bentuk `.txt` dan juga image sesuai dengan kebutuhan dari pengguna inginkan.

4 Pengujian

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

Masukan	Keluaran
Test Case 1: Solusi Ditemukan File Input: test1.txt AAABBCCCD ABBBBCECD ABBBDCECD AAABDCCCD BBBBDDDDD FGGGDHDD FGIGDDHDD FGIGDDHDD FGGGDHDDH	<div> Disimpan: test1.png Waktu: 44871 ms Iterasi: 21415356 </div>  <div> Status: Solusi Ditemukan Waktu: 44871 ms Iterasi: 21,415,356 </div>
Test Case 2: Solusi Ditemukan File Input : test2.txt BAAAAACC BAAAAAAA BAADAAAA AAADDDAA AAAEEDAA AEEEEFFAG AAAAAFAG HHHAFFAG	<div> Solusi Ditemukan! Waktu: 400 ms Iterasi: 186816 </div>  <div> Status: Solusi Ditemukan Waktu: 400 ms Iterasi: 186,816 </div>

Masukan	Keluaran
Test Case 3: Solusi Ditemukan File Input : test3.txt AAAABBBB CCCCCCCB DDEEEECH DCCCCECH DCFCCECH DCFEEECH ICCCCCCH IIIIIIHH	<div> Solusi Ditemukan! Waktu: 4201 ms Iterasi: 2144997 </div>  <div> Status: Solusi Ditemukan Waktu: 4201 ms Iterasi: 2,144,997 </div>
Test Case 4: Solusi Ditemukan File Input : test4.txt AAAAAAAA BBCCAADA BAACCDDA BEAAHHDA FEAAHHAA FEEAAIIA FAAAIIAA FAAAAAAA	<div> Solusi Ditemukan! Waktu: 211 ms Iterasi: 121840 </div>  <div> Status: Solusi Ditemukan Waktu: 211 ms Iterasi: 121,840 </div>

Masukan	Keluaran
Test Case 5: Solusi Ditemukan File Input : test5.txt AAAAA BBBBB CCCCC DDDDD EEEEEE FFFFFFF GGGGGGG HHHHHHH IIIIIII AAAAA	<div> Disimpan: Output_test5.png Waktu: 466309 ms Iterasi: 241357969 </div>  <p> Status: Solusi Ditemukan Waktu: 466309 ms Iterasi: 241,357,969 </p>
Test Case 6: Kesalahan Masukan File Input : test6.txt AAAAA BBBBB CC CC DDDDD EEEE	<p>Tidak ada Gambar</p> <p> Status: - Waktu: - ms Iterasi: - </p>
Test Case 7: Tidak Ada Solusi File Input : test7.txt BBBFFF BBCCAA CCCAAA DDDDDA EEEDAA DDDEEE	<div> Solusi Tidak Ditemukan. Waktu: 59 ms Iterasi: 32400 </div>  <p> Status: Tidak Ada Solusi Waktu: 59 ms Iterasi: 32,400 </p>

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