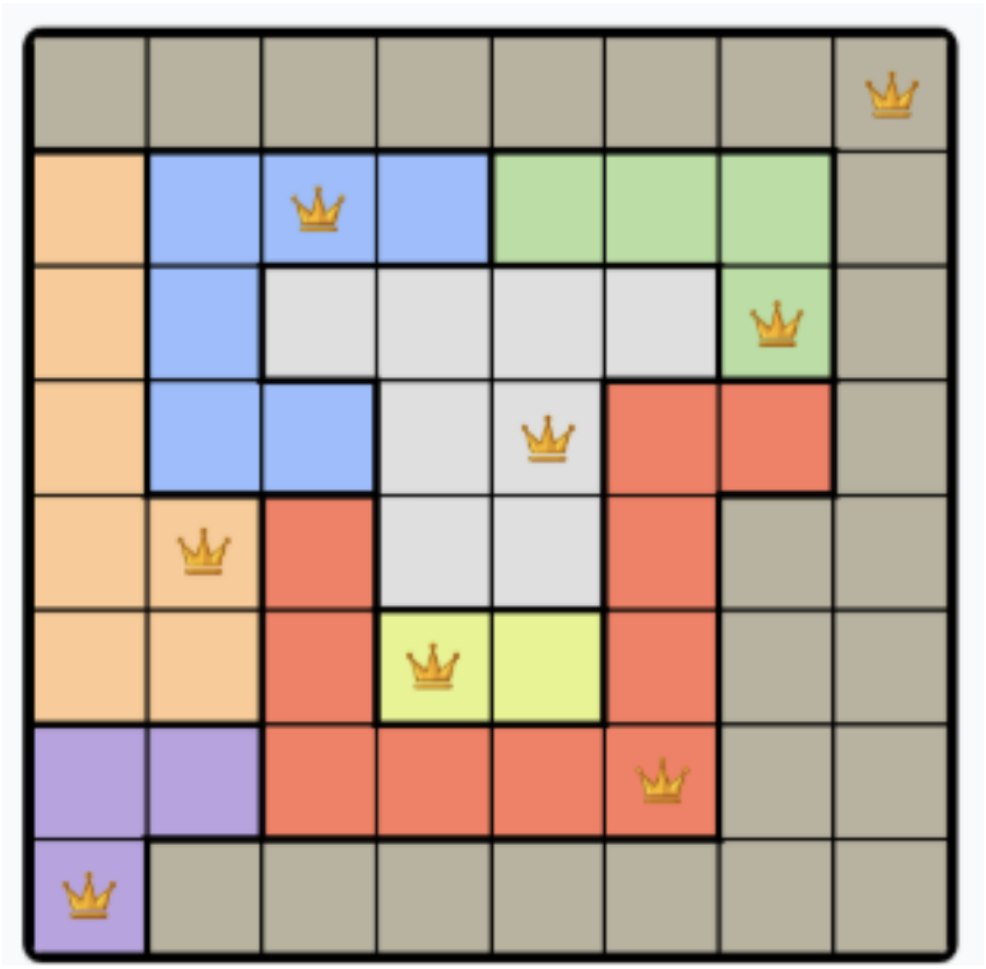
Laporan Tugas Kecil 1

IF2211 Strategi Algoritma

Penyelesaian Permainan ***Queens*** Linkedin

Semester II Tahun 2025/2026



Emilio Justin

13524043

Laboratorium Ilmu dan Rekayasa Komputasi
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

Daftar Isi

Daftar Gambar

Daftar Tabel

Listings

1 Implementasi

Queens adalah gim logika yang tersedia pada situs jejaring profesional LinkedIn. Tujuan dari gim ini 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. Penyelesaian gim ini diprogram dengan bahasa Java, memanfaatkan *toolkit* JavaFX untuk GUI (Graphical User Interface)-nya. Algoritma *brute force* ditekankan untuk menyelesaikan gim ini.

1.1 Brute Force

Algoritma *brute force* yang diimplementasikan bersifat murni, yaitu mengenumerasi seluruh penempatan *queen* yang mungkin berdasarkan aturan permainan. Penulis memanfaatkan *backtracking* tanpa adanya *pruning* sehingga algoritma tetap mengecek seluruh kemungkinan posisi *queen*.

Algoritma utama *brute force* terdapat di prosedur `solveBruteForce(int row, Board papan)` yang alur kerja sederhananya adalah sebagai berikut:

1. Pada awal pemanggilan prosedur (pertama kali), parameter yang di-*assign* adalah 0 dan `papan` yang sudah terisi.
2. Algoritma akan mencoba meletakkan *queen* di setiap kolom dengan adanya *backtracking* di baris `row`, lalu melanjutkan pemanggilan dengan parameter `row` yang lebih dalam.
3. Jika pada saat ini nilai `row` sama dengan baris papan yang sebenarnya, akan dilakukan pengecekan apakah konfigurasi papan yang sudah ditempatkan *queen* valid.
4. Jika valid, atribut `solution[][]` akan di-*update* dengan konfigurasi papan tersebut.

Dikarenakan penulis memberikan opsi untuk mengoptimasi komputasi dengan menyediakan fitur *optimized* maka terdapat trik yang dapat mempercepat komputasi, yaitu dengan memoisasi kolom dan warna yang sudah terpakai, kasus yang bertabrakan dengan hal tersebut akan dilewati dan tidak dicek.

Algoritma murni tanpa optimasi memiliki kompleksitas waktu $O(N^N)$ karena untuk setiap baris, akan dicoba N kolom dari konfigurasi papan sebelumnya.

Algorithm 1 Algoritma Brute Force Queen

```
procedure solveBruteForce(row, Papan)
    if cancelled then
        return
    end if
    if isOptimized = true then
        caseCount ← caseCount + 1
    end if
    if row = Papan.getRow() then
        if isOptimized = true then
            solutionCount ← solutionCount + 1
            for i ← 0 to Papan.getRow() - 1 do
                for j ← 0 to Papan.getCol() - 1 do
                    solution[i][j] ← board[i][j]
                end for
            end for
            return
        else
            caseCount ← caseCount + 1
            if isBoardValid(Papan.getRow() - 1, Papan) then
                solutionCount ← solutionCount + 1
                for i ← 0 to Papan.getRow() - 1 do
                    for j ← 0 to Papan.getCol() - 1 do
                        solution[i][j] ← board[i][j]
                    end for
                end for
            end if
            return
        end if
    end if
    for col ← 0 to Papan.getCol() - 1 do
        if isOptimized = true then
            warna ← Papan.getElmt(row, col) - 'A'
            if usedColumn[col] = 0 and color[warna] = 0 and check8Direction(row, col, Papan) then
                color[warna] ← 1
                board[row][col] ← 1
                usedColumn[col] ← 1
                debugCases(Papan)
                solveBruteForce(row + 1, Papan)
                color[warna] ← 0
                board[row][col] ← 0
                usedColumn[col] ← 0
            end if
        else
            board[row][col] ← 1
            debugCases(Papan)
            solveBruteForce(row + 1, Papan)
            board[row][col] ← 0
        end if
    end for
end procedure
```

1.2 Fitur Pelengkap

Program juga mengimplementasikan beberapa fitur lainnya, antara lain:

1. Graphical User Interface (GUI) : Antarmuka grafis menggunakan JavaFX
2. Live Update : Saat solver berjalan, papan di-*render* ulang setiap langkah dengan delay yang bisa diatur menggunakan slider sehingga pengguna dapat melihat proses pencarian secara *real-time*

3. Output as Text : Solusi bisa disimpan ke file `.txt`
4. Output as Image : Solusi bisa disimpan sebagai file `.png`

2 Source Code

Penulis akan menampilkan *source code* penuh untuk file `Board.java` dan `Solver.java` sedangkan untuk file `Page.java` hanya akan ditampilkan spesifikasi dari setiap fungsi atau prosedur.

2.1 Board.java

```java
package stima.modules;

public class Board {

    /**
     * Atribut
     */
    private char[][] content;
    private int rows, cols;

    /**
     * { Membuat Board kosong berdasarkan masukan jumlah baris dan kolom }
     *
     * @param row
     * @param col
     */
    public Board(int row, int col)
    {
        this.rows = row;
        this.cols = col;
        this.content = new char[row][col];
    }

    /**
     * { Membuat Board dengan mengcopy Board lain }
     *
     * @param Papan
     */
    public Board(Board Papan)
    {
        this.rows = Papan.rows;
        this.cols = Papan.cols;
        this.content = new char[Papan.rows][Papan.cols];
        for (int i = 0; i < this.rows; i++)
        {
            for (int j = 0; j < this.cols; j++)
            {
                this.setElmt(i, j, Papan.content[i][j]);
            }
        }
    }

    /**
     * Selektor baris
     *
     * @return jumlah baris dari Boardnya
     */
    public int getRow()
    {
        return this.rows;
    }

    /**
     * Selektor kolom
     *
     * @return jumlah kolom dari Boardnya
     */
    public int getCol()
    {
        return this.cols;
    }

    /**
     * Getter element
     *
     * @param row
```

```
     * @param col
     * @return elemen yang berada di baris row dan kolom col
     */
    public char getElmt(int row, int col)
    {
        return this.content[row][col];
    }

    /**
     * { Set elemen pada Board dengan value baru }
     *
     * @param row
     * @param col
     * @param value
     */
    public void setElmt(int row, int col, char value)
    {
        this.content[row][col] = value;
    }
}
```

Listing 2.1: Board.java – Board Entity Class

2.2 Solver.java

```
package stima.modules;

import java.time.Duration;
import java.time.Instant;
import java.util.function.BiConsumer;

public class Solver {

    /**
     * Atribut
     */
    private int[] color;
    private int[] usedColumn;
    private int[][] board;
    private int[][] solution;
    private Duration executionTime;
    private int caseCount;
    private int solutionCount;
    private int delayMs = 30;
    private boolean isOptimized = false;
    private volatile boolean cancelled = false;
    private BiConsumer<int[][], Board> stepBack = null;

    /**
     * Parent function untuk solve queens dengan inisiasi atribut dan pemanggilan
    prosedur lain
     *
     * @param Papan
     * @return true jika solusi ada, false jika tidak
     */
    public boolean solveQueens(Board Papan)
    {
        color = new int[26];
        usedColumn = new int[Papan.getCol()];
        board = new int[Papan.getRow()][Papan.getCol()];
        solution = new int[Papan.getRow()][Papan.getCol()];
        this.caseCount = 0;
        this.solutionCount = 0;

        Instant start = Instant.now();
        solveBruteForce(0, Papan);
        Instant end = Instant.now();

        this.executionTime = Duration.between(start, end);

        if (solutionCount > 0) {
            for (int i = 0; i < Papan.getRow(); i++) {
                for (int j = 0; j < Papan.getCol(); j++) {
                    board[i][j] = solution[i][j];
                }
```

```
            }
            return true;
        }
        return false;
    }

    /**
     * Algoritma brute force backtracking tanpa heuristik yang mencoba menempatkan satu Queen di setiap baris
     * dengan percobaan untuk menempatkannya di masing-masing kolom, kompleksitasnya O(n^n)
     *
     * Ini ga ada heuristik atau pruning semacamnya sih kalau menurut saya, dengan mengetahui aturan permainan,
     * better brute force untuk setiap baris, coba tempatkan Queen di masing-masing kolom lalu lanjut ke baris selanjutnya
     *
     * @param row
     * @param Papan
     */
    private void solveBruteForce(int row, Board Papan)
    {
        if (cancelled) return;

        if (this.isOptimized == true)
        {
            caseCount++;
        }

        if (row == Papan.getRow())
        {
            if (this.isOptimized == true)
            {
                solutionCount++;
                for (int i = 0; i < Papan.getRow(); i++)
                {
                    for (int j = 0; j < Papan.getCol(); j++)
                    {
                        solution[i][j] = board[i][j];
                    }
                }
                return;
            }
            else
            {
                caseCount++;
                if (isBoardValid(Papan.getRow() - 1, Papan))
                {
                    solutionCount++;
                    for (int i = 0; i < Papan.getRow(); i++)
                    {
                        for (int j = 0; j < Papan.getCol(); j++)
                        {
                            solution[i][j] = board[i][j];
                        }
                    }
                }
                return;
            }

        }

        for (int col = 0; col < Papan.getCol(); col++)
        {
            if (this.isOptimized == true)
            {
                int warna = Papan.getElmt(row, col) - 'A';

                if (usedColumn[col] == 0 && color[warna] == 0 && check8Direction(row, col, Papan))
                {
                    color[warna] = 1;
                    board[row][col] = 1;
                    usedColumn[col] = 1;

                    debugCases(Papan);
```

```
121
122                     solveBruteForce(row+1, Papan);
123
124                     color[warna] = 0;
125                     board[row][col] = 0;
126                     usedColumn[col] = 0;
127                 }
128             }
129             else
130             {
131                 board[row][col] = 1;
132
133                 debugCases(Papan);
134
135                 solveBruteForce(row+1, Papan);
136
137                 board[row][col] = 0;
138             }
139         }
140     }
141
142     /**
143      * { Mengecek 8 arah adjacent apakah ada Queen yang ditempatkan atau tidak }
144      *
145      * @param row
146      * @param col
147      * @param Papan
148      *
149      * @return true jika di ke 8 arah tidak ada queens yang diletakkan, false jika ada
150      */
151     private boolean check8Direction(int row, int col, Board Papan)
152     {
153         for (int i = -1; i <= 1; i++)
154         {
155             for (int j = -1; j <= 1; j++)
156             {
157                 if (i == 0 && j == 0) continue;
158
159                 int newrow = row + i, newcol = col + j;
160                 if (newrow >= 0 && newrow < Papan.getRow() && newcol >= 0 && newcol <
    Papan.getCol() && board[newrow][newcol] == 1) return false;
161             }
162         }
163         return true;
164     }
165
166     /**
167      * { Mengecek apakah Board merupakan solusi yang valid untuk permainan Queens }
168      *
169      * @param row
170      * @param Papan
171      *
172      * @return true jika Board valid, false jika tidak
173      */
174     public boolean isBoardValid(int row, Board Papan)
175     {
176         for (int i = 0; i <= row; i++)
177         {
178             for (int j = 0; j < Papan.getCol(); j++)
179             {
180                 if (board[i][j] == 1)
181                 {
182                     for (int k = 0; k < i; k++)
183                     {
184                         if (board[k][j] == 1) return false;
185                     }
186
187                     int warna = Papan.getElmt(i, j) - 'A';
188                     for (int r = 0; r < i; r++)
189                     {
190                         for (int c = 0; c < Papan.getCol(); c++)
191                         {
192                             if (board[r][c] == 1 && (Papan.getElmt(r, c) - 'A') == warna)
193                             {
194                                 return false;
195                             }
```

```
                        }
                    }

                    if (!check8Direction(i, j, Papan)) return false;
                }
            }
        }
        return true;
    }

    /**
     * Print kondisi papan
     *
     * @param Papan
     */
    public void printBoard(Board Papan)
    {
        for (int i = 0; i < Papan.getRow(); i++)
        {
            for (int j = 0; j < Papan.getCol(); j++)
            {
                System.out.print(board[i][j] == 1 ? '#' : Papan.getElmt(i, j));
            }
            System.out.println();
        }
    }

    /**
     * Live Update Papan
     *
     * @param Papan
     */
    private void debugCases(Board Papan)
    {
        if (stepBack != null) {
            int[][] snapshot = new int[board.length][board[0].length];
            for (int i = 0; i < board.length; i++) {
                System.arraycopy(board[i], 0, snapshot[i], 0, board[i].length);
            }
            stepBack.accept(snapshot, Papan);
        }

        if (stepBack == null) {
            System.out.print("\033[H\033[2J");
            System.out.flush();
            System.out.println("Finding solution...");
            System.out.flush();
            this.printBoard(Papan);
            System.out.flush();
        }

        try {
            Thread.sleep(delayMs);
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
            cancelled = true;
        }
    }

    /**
     * Getter execution time
     *
     * @return
     */
    public Duration getExecutionTime()
    {
        return this.executionTime;
    }

    /**
     * Getter case count
     *
     * @return
     */
    public int getCaseCount()
    {
```

```
272         return this.caseCount;
273     }
274 
275     /**
276      * Getter solution count
277      *
278      * @return
279      */
280     public int getSolutionCount()
281     {
282         return this.solutionCount;
283     }
284 
285     /**
286      * Setter delayMs
287      *
288      * @param delayMs
289      */
290     public void setDelay(int delayMs)
291     {
292         this.delayMs = delayMs;
293     }
294 
295     /**
296      * Setter isOptimized
297      *
298      * @return
299      */
300     public void setOptimized()
301     {
302         this.isOptimized = true;
303     }
304 
305     /**
306      * Getter Board
307      *
308      * @return
309      */
310     public int[][] getBoard()
311     {
312         return this.board;
313     }
314 
315     /**
316      * Set callback yang dipanggil setiap langkah solver
317      *
318      * @param callback menerima (int[][] boardSnapshot, Board papan)
319      */
320     public void setStepBack(BiConsumer<int[][], Board> callback)
321     {
322         this.stepBack = callback;
323     }
324 
325     /**
326      * Cancel solver yang sedang berjalan
327      */
328     public void cancel()
329     {
330         this.cancelled = true;
331     }
332 
333     /**
334      * Cek apakah solver sudah di-cancel
335      */
336     public boolean isCancelled()
337     {
338         return this.cancelled;
339     }
340 }
```

Listing 2.2: Solver.java – Algoritma Solver Queens

2.3 Page.java

```
1 package stima.gui;
```

```
2
3  import javafx.animation.FadeTransition;
4  import javafx.animation.PauseTransition;
5  import javafx.animation.SequentialTransition;
6  import javafx.application.Application;
7  import javafx.application.Platform;
8  import javafx.embed.swing.SwingFXUtils;
9  import javafx.geometry.Insets;
10 import javafx.geometry.Pos;
11 import javafx.scene.Scene;
12 import javafx.scene.SnapshotParameters;
13 import javafx.scene.control.*;
14 import javafx.scene.image.WritableImage;
15 import javafx.scene.layout.*;
16 import javafx.scene.paint.Color;
17 import javafx.scene.text.Font;
18 import javafx.scene.text.FontWeight;
19 import javafx.scene.text.Text;
20 import javafx.scene.text.TextAlignment;
21 import javafx.stage.FileChooser;
22 import javafx.util.Duration;
23 import javafx.stage.Stage;
24
25 import stima.modules.Board;
26 import stima.modules.Solver;
27
28 import javax.imageio.ImageIO;
29 import java.io.*;
30 import java.util.ArrayList;
31 import java.util.Scanner;
32
33 public class Page extends Application {
34
35     /**
36      * Atribut
37      */
38     private static final Color[] REGION_COLORS =
39     {
40         Color.web("#ff6d6d"),
41         Color.web("#cdd1ff"),
42         Color.web("#45B7D1"),
43         Color.web("#8785ff"),
44         Color.web("#cfb253"),
45         Color.web("#ff99ff"),
46         Color.web("#8ce4ce"),
47         Color.web("#F7DC6F"),
48         Color.web("#ddd2ff"),
49         Color.web("#ffe2cb"),
50         Color.web("#ffc089"),
51         Color.web("#82E0AA"),
52         Color.web("#F1948A"),
53         Color.web("#AED6F1"),
54         Color.web("#D7BDE2"),
55         Color.web("#ff8bda"),
56         Color.web("#FAD7A0"),
57         Color.web("#afdfff"),
58         Color.web("#f3ff72"),
59         Color.web("#a2ffd5"),
60         Color.web("#cff18f"),
61         Color.web("#c6dffc"),
62         Color.web("#cfffd1"),
63         Color.web("#46d36b"),
64         Color.web("#ffbdbd"),
65         Color.web("#62ffda"),
66     };
67     private Board papan;
68     private Solver solver;
69     private GridPane boardGrid;
70     private Label statusLabel;
71     private Label timeLabel;
72     private Label caseLabel;
73     private CheckBox optimizedCheck;
74     private Slider delaySlider;
75     private Label delayValueLabel;
76     private Button solveBtn;
77     private Button stopBtn;
```

```
78      private Button loadBtn;
79      private Button saveBtn;
80      private Button saveImageBtn;
81      private Thread solverThread;
82      private boolean solving;
83      private StackPane boardContainer;
84
85      @Override
86      public void start(Stage primaryStage);
87
88      /**
89       * Load file test case
90       */
91      private void loadFile(Stage stage);
92
93      /**
94       * Solve Queens Board
95       */
96      private void solve();
97
98      /**
99       * Menghentikan solver yang sedang berjalan
100      */
101     private void stopSolving();
102
103     /**
104      * Render board ke GridPane
105      */
106     private void renderBoard(int[][] solution);
107
108     /**
109      * Save solusi ke file
110      */
111     private void saveSolution(Stage stage);
112
113     /**
114      * Save board sebagai image (PNG)
115      */
116     private void saveAsImage(Stage stage);
117
118     /**
119      * { Styling button }
120      *
121      * @param color
122      *
123      * @return
124      */
125     private String buttonStyle(String color);
126
127     private String toHex(Color color);
128
129     /**
130      * { Menampilkan pop up alert }
131      *
132      * @param title
133      * @param message
134      *
135      * @return
136      */
137     private void showAlert(String title, String message);
138
139     /**
140      * Notifikasi pop up muncul ketika solved
141      */
142     private void showSolvePopUp(String message, String bgColor);
143
144     public static void main(String[] args)
145     {
146         launch(args);
147     }
148 }
```

Listing 2.3: Page.java – Graphical User Interface

3 Pengujian

Penulis menyarankan apabila penguji menggunakan program dan memasukkan input dengan ukuran yang cukup besar, Window GUI program dapat di-*resize* secara menual (tidak responsive).

3.1 Test 1: Papan berukuran 9×9

Tabel 3.1: Input Test 1

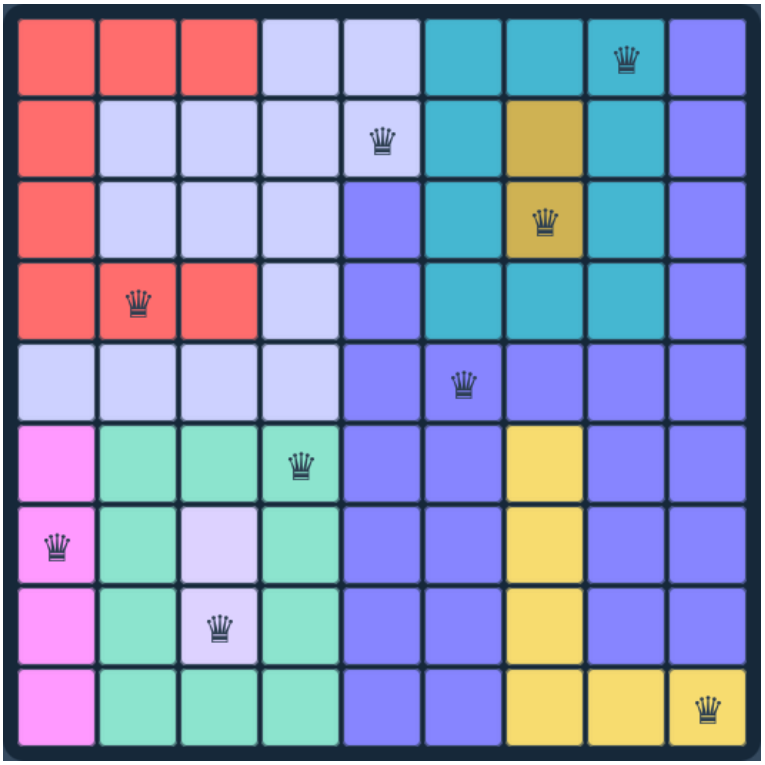
AAABBCCCD ABBBBCECD ABBBDCECD AAABDCCCD BBBBDDDDD FGGGDDHDD FGIGDDHDD FGIGDDHDD FGGGDDHHH

Output sebagai `.txt`

```
AAABBCC#D
ABBB#CECD
ABBBDC#CD
A#ABDCCCD
BBBBD#DDD
FGG#DDHDD
#GIGDDHDD
FG#GDDHDD
FGGGDDHH#
```

Listing 3.1: "Test 1 Output"

Output sebagai gambar



Gambar 3.1.1: Output GUI — Test 1

3.2 Test 2: Papan berukuran 8×8

Tabel 3.2: Input Test 2

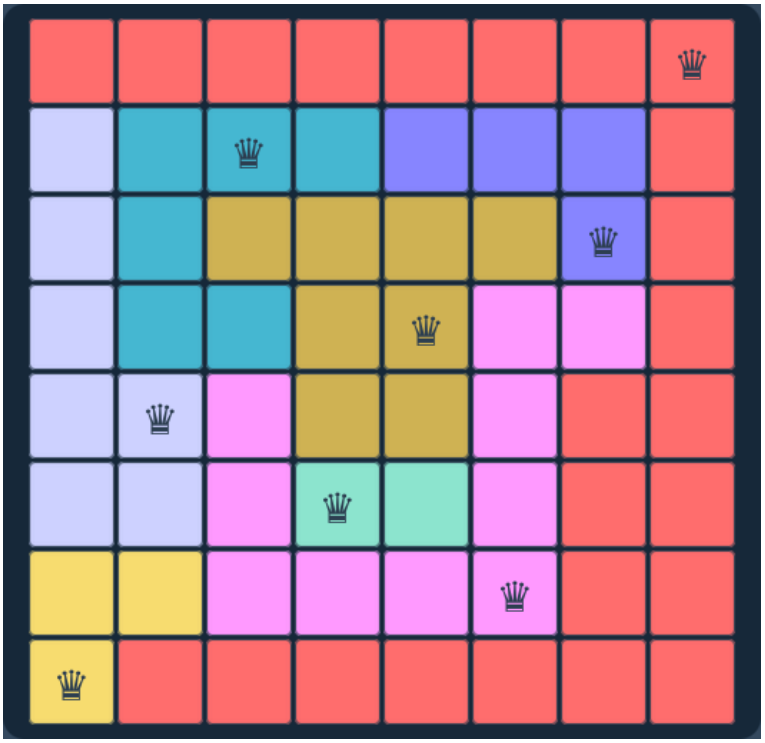
AAAAAAAA BCCCDDDA BCEEEEDA BCCEEFFA BBFEEFAA BBFGGFAA HHFFFFAA HAAAAAAA

Output sebagai `.txt`

```
AAAAAAA#
BC#CDDDA
BCEEEE#A
BCCE#FFA
B#FEEFAA
BBF#GFAA
HHFFF#AA
#AAAAAAA
```

Listing 3.2: "Test Case 2 Output"

Output sebagai gambar



Gambar 3.2.1: Output GUI — Test 2

3.3 Test 3: Papan berukuran 7×7

Tabel 3.3: Input Test 3

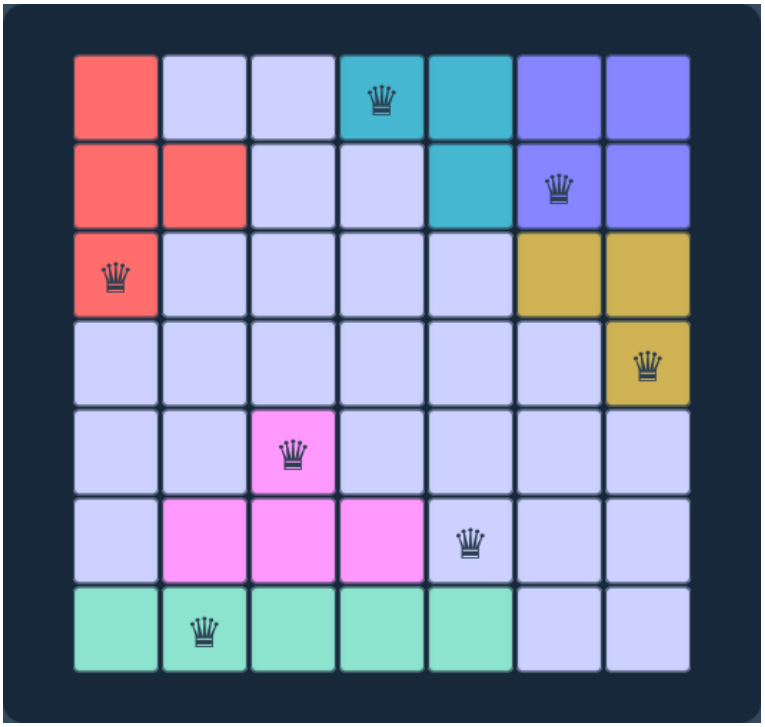
ABBCCDD AABBCDD ABBBBEE BBBBBBE BBFBBBB BFFFBBB GGGGGBB

Output sebagai `.txt`

```
ABB#CDD
AABBC#D
#BBBBEE
BBBBBB#
BB#BBBB
BFFF#BB
G#GGGBB
```

Listing 3.3: "Test Case 3 Output"

Output sebagai gambar



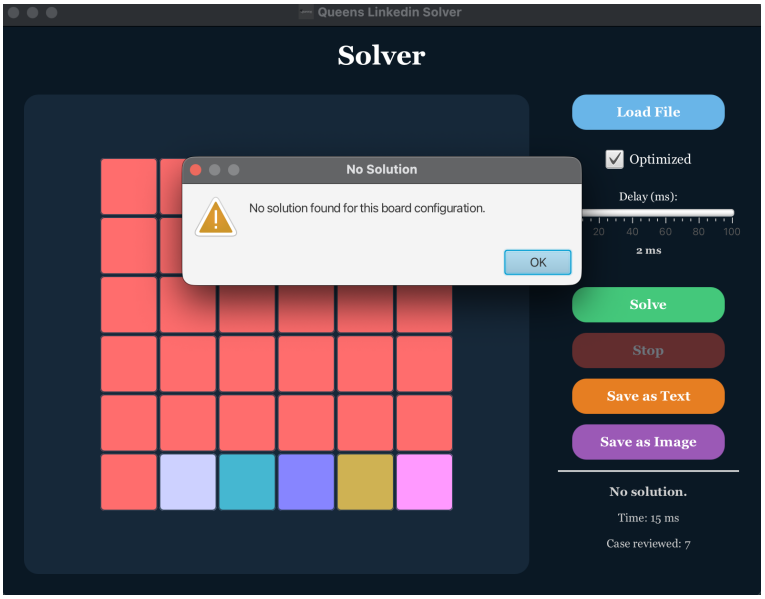
Gambar 3.3.1: Output GUI — Test 3

3.4 Test 4: Papan tidak memiliki solusi

Tabel 3.4: Input Test 4

```
AAAAAA
AAAAAA
AAAAAA
AAAAAA
AAAAAA
ABCDEF
```

Output sebagai gambar



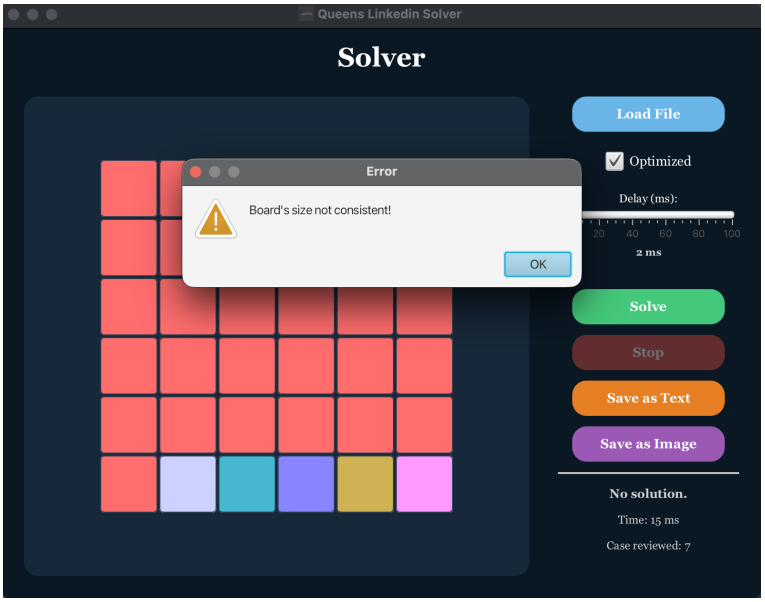
Gambar 3.4.1: Output GUI — Test 4

3.5 Test 5: Input tidak valid

Tabel 3.5: Input Test 5

```
ABBCCDD
AABBCDD
ABBBBEE
BBBBBBE
BBFBBBB
BFFFBBB
GGGG
```

Output sebagai gambar



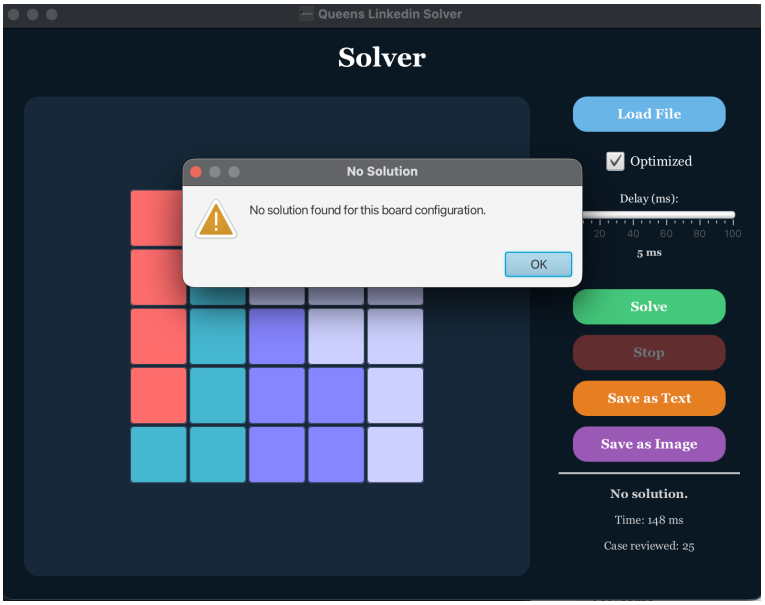
Gambar 3.5.1: Output GUI — Test 5

3.6 Test 6: Papan tidak memiliki solusi

Tabel 3.6: Input Test 6

```
AAABB
ACBBB
ACDBB
ACDDB
CCDDB
```

Output sebagai gambar



Gambar 3.6.1: Output GUI — Test 6

3.7 Test 7: Papan berukuran 5×5

Tabel 3.7: Input Test 7

AAABBC ADDBBC ADDEEC AFFEEC AFFEEC AFFGGC

Output sebagai `.txt`

```
AAABB#
ADD#BC
A#DEEC
AFFE#C
AF#EEC
#FFGGC
```

Listing 3.4: "Test Case 7 Output"

Output sebagai gambar

Gambar 3.7.1: Output GUI — Test 7

4 Lampiran

https://github.com/Valz0504/Tucil1_13524043

Tabel 4.1: Evaluasi

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	✓	
7	Program dapat membaca masukan gambar		✓

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

Emilio Justin