Laporan Tugas Kecil I
Mata Kuliah IF 2211 Strategi Algoritma

Disusun oleh:

13524052-Raynard Fausta
raynardfaustasmlone@gmail.com 1324052@std.stei.itb.ac.id

Program Studi
Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2026

Daftar Isi Laporan

Bab I
Pendahuluan

1.1. Latar Belakang

Permainan Queens LinkedIn merupakan permainan logika yang dapat diakses di situs jejaring profesional Linkedin. Permainan ini dimainkan pada sebuah papan yang berukuran n*n(persegi). Untuk menyelesaikan permainan ini, seorang pemain harus mampu menempatkan n buah bidak ratu sedemikian rupa sehingga pada setiap baris, kolom, dan zona berwarna terdapat tepat 1 ratu. Selain itu, dua atau lebih bidak ratu tidak boleh ditempatkan bersebelahan dalam segala arah (horizontal, vertikal, dan diagonal).

Dalam konteks informatika, permainan ini dapat dipandang sebagai sebuah persoalan yang dapat diselesaikan dengan program karena pencarian solusinya melibatkan langkah-langkah yang sistematis dan baku. Terdapat banyak pendekatan komputasi untuk menyelesaikan sebuah permasalah informatika dan salah satunya adalah pendekatan dengan algoritma *brute force*, yang memiliki proses berpikir yang sederhana, pendekatannya bersifat langsung (*straightforward*), dan mempunyai langkah-langkah yang jelas. Oleh karena itu, penulis tertarik mengobservasi penyelesaian permasalahan ini dengan pendekatan algoritma *brute force*.

1.2. Tujuan

Laporan ini bertujuan untuk membuktikan dan menjelaskan bahwa algoritma *brute force* dapat digunakan untuk menyelesaikan permainan Queens LinkedIn dengan menunjukkan solusi untuk setiap kasus uji. Setiap solusi yang disajikan dilengkapi dengan jumlah konfigurasi yang diperiksa, tampilan beberapa konfigurasi yang diperiksa, waktu percobaan, analisis algoritma, serta kesimpulan umum berdasarkan hasil percobaan secara keseluruhan.

Bab II
Landasan Teori

2.1. Algoritma Brute Force

Algoritma *brute force* adalah algoritma yang memiliki proses berpikir yang sederhana dalam penyelesaian masalah, pendekatan yang sifatnya langsung(*straightforward*), dan langkah-langkah penyelesaiannya jelas.

2.2. Exhaustive Search

Exhaustive search tergolong sebagai strategi *brute force* karena teknik ini menelusuri semua kemungkinan untuk menyelesaikan sebuah permasalahan hingga menemukan solusi.

Bab III
Metode Penyelesaian

3.1. Penyelesaian Permasalahan dengan Brute Force

Berikut adalah kode penyelesaian yang disusun dengan pendekatan algoritma *brute force*:

```
    public static char[][] hasilkan_dan_cek_susunan(int mulai, int
jumlah_bidak_tersusun, int ukuran, int[] letak_bidak, char[][] masukan){
        if (jumlah_bidak_tersusun==ukuran){
            konfigurasi_ditinjau++;
```

```
            long now=System.nanoTime();
            if (now-update_terakhir>=interval){

                char[][] calon=new char[masukan.length][];

                for (int i=0; i<masukan.length; i++) {
                    calon[i]=masukan[i].clone();
                }

                for (int i=0; i<ukuran; i++) {

calon[letak_bidak[i]/ukuran][letak_bidak[i]%ukuran]='#';
                }

                preview=calon;
                previewCount=konfigurasi_ditinjau;
                update_terakhir=now;
            }

            if (cek_Quuens(letak_bidak, ukuran, masukan)) {
                char[][] a=masukan;

                for (int i=0; i<ukuran; i++) {
                    a[letak_bidak[i]/ukuran][letak_bidak[i]%ukuran]='#';
                }

                return a;
            }
            return null;
        }

        for (int i=mulai;
i<=(ukuran*ukuran)-ukuran+jumlah_bidak_tersusun; i++) { //misal di
bidang 2*2 kalau masih 0 yang tersusun dia cuman bisa susun sampai kotak
ketiga(2), kalau udah 1 tersusun bisa sampai 3
            letak_bidak[jumlah_bidak_tersusun]=i;
            char[][] hasil=hasilkan_dan_cek_susunan(i+1,
jumlah_bidak_tersusun+1, ukuran, letak_bidak, masukan); //susun bidak
```

```
berikutnya
            if (hasil!=null) {
                return hasil;
            }
        }
        return null;
    }
```

Penyelesaian masalah bersifat rekursi karena jika diperhatikan fungsi akan terus memanggil entitas yang mengandung dirinya sendiri. Fungsi penyelesaian masalah akan menerima masukan berupa konfigurasi awal papan, ukuran papan untuk menentukan jumlah bidak yang harus disusun, serta beberapa variabel seperti mulai, jumlah_bidak_tersusun, dan letak_bidak untuk menyimpan data dari rekursi sebelumnya. Peletakkan pada papan yang seharusnya memiliki 2 elemen, yaitu baris dan kolom, disederhanakan menjadi indeks 0 sampai n*n-1. Perhitungan indeks dimulai dari petak ujung kiri atas bergerak ke ujung kanan atas kemudian bergerak ke petak di bawah petak ujung kiri atas dan seterusnya. Konversinya adalah sebagai berikut **indeks= letak baris * ukuran sisi** + **letak kolom** dengan letak baris dan letak kolom *zero based*. Untuk basis dari rekursi ini adalah apabila jumlah_bidak_tersusun pada sebuah konfigurasi sudah sama dengan ukuran sisi papan dan solusi tersebut benar merupakan solusi untuk permainan Queens LinkedIn.

Fungsi penyelesaian dengan algoritma *brute force* dimulai dengan menempatkan bidak ratu pertama di indeks 0(petak ujung kiri atas). Setelah itu program akan memasuki tahap loop dan rekurens secara bersamaan.

Pada tahap loop, akan dicoba konfigurasi lain, yaitu bidak ratu pertama ditempatkan di indeks berikutnya sampai batas(**batas= indeks maksimum - jumlah bidak tersisa yang harus disusun** + **1**, dituliskan pada program sebagai **batas= ukuran sisi * ukuran sisi - jumlah bidak yang harus disusun** + **jumlah bidak yang telah disusun**). Untuk kasus ini, **indeks maksimum** + **1** = **ukuran sisi * ukuran sisi** dan **jumlah bidak tersisa yang harus disusun** = **jumlah bidak yang harus disusun - jumlah bidak yang telah disusun**.

Pada tahap rekurens, fungsi penyelesaian akan dipanggil untuk menyusun kemungkinan penempatan bidak ratu berikutnya yang posisinya minimal harus berada 1 indeks di sebelah kanan indeks bidak ratu sebelumnya.

Pada tahap loop, apabila dari rekurens-rekurens sebelumnya telah mendapatkan solusi yang benar maka fungsi akan berhenti dan mengembalikan solusi untuk konfigurasi papan yang menjadi masukan.

3.2. Optimasi

```
    public static boolean petak_sudah_ada (char[][] papan, int[]
letak_bidak, int idx, int n, int jumlah_bidak_tersusun){
```

```
        for (int i=0; i<jumlah_bidak_tersusun; i++){
            if (letak_bidak[i]%n==idx%n ||
papan[letak_bidak[i]/n][letak_bidak[i]%n]==papan[idx/n][idx%n]) return
true;
        }
        return false;
    }

Penggunaan:
        for (int i=mulai;
i<=(ukuran*ukuran)-ukuran+jumlah_bidak_tersusun; i++) { //misal di
bidang 2*2 kalau masih 0 yang tersusun dia cuman bisa susun sampai kotak
ketiga(2), kalau udah 1 tersusun bisa sampai 3
            if(i>=ukuran*jumlah_bidak_tersusun &&
i<jumlah_bidak_tersusun*ukuran+ukuran && !petak_sudah_ada(masukan,
letak_bidak, i, ukuran, jumlah_bidak_tersusun)) { // optimasi
                letak_bidak[jumlah_bidak_tersusun]=i;
                char[][] hasil=hasilkan_dan_cek_susunan(i+1,
jumlah_bidak_tersusun+1, ukuran, letak_bidak, masukan); //susun bidak
berikutnya
                if (hasil!=null){
                    return hasil;
                }
            }

        }
```

Untuk menangani kasus dengan ukuran yang lebih besar dilakukan sedikit optimasi(telah diperbolehkan sebagai alternatif berdasarkan jawaban QnA Tugas Kecil 1). Salah satu aturan yang berguna untuk optimasi ini adalah tidak boleh ada lebih dari satu bidak ratu yang berada di baris dan/atau kolom yang sama. Oleh karena itu, indeks penempatan bidak berikutnya dapat dibatasi menjadi **jumlah bidak yang telah tersusun * ukuran sisi** ≤ **indeks penempatan** ≤ **jumlah bidak yang telah tersusun * ukuran sisi - 1** dan dipastikan untuk posisi kolomnya **indeks** mod ukuran sisi tidak ada yang sama dengan letak kolom bidak-bidak sebelumnya **.** Selain itu indeks calon penempatan bidak berikutnya juga dipastikan tidak berada di zona yang sama dengan zona yang telah ditempati sebelumnya serta tidak bersebelahan. Pada dasarnya, perbedaan optimasi dan murni pada program ini adalah pada algoritma murni susunan dibentuk

kemudian dicek apakah merupakan solusi, sedangkan pada algoritma yang dioptimasi calonan penempatan langsung diperiksa validitasnya.

3.3. Pemeriksaan Calon Solusi

```
    public static boolean cek_Quuens(int[] letak_bidak, int ukuran,
char[][] masukan){

        for(int k=0; k<ukuran-1; k++) {
            int baris_bidak1=letak_bidak[k]/ukuran;
            int kolom_bidak1=letak_bidak[k]%ukuran;
            for (int l=k+1; l<ukuran; l++) {
                int baris_bidak2=letak_bidak[l]/ukuran;
                int kolom_bidak2=letak_bidak[l]%ukuran;

                if
(masukan[baris_bidak1][kolom_bidak1]==masukan[baris_bidak2][kolom_bidak2
] ||  baris_bidak1==baris_bidak2 || kolom_bidak1==kolom_bidak2 ||
(Math.abs(kolom_bidak1-kolom_bidak2)==1 &&
Math.abs(baris_bidak1-baris_bidak2)==1)) {
                    return false;
                }
            }
        }

        return true;
    }
```

Variabel letak bidang yang dipergunakan untuk menyimpan posisi calon susun bidak digunakan sebagai masukan. Indeks di dalamnya dikonversi menjadi **letak baris** = **indeks** div **ukuran** sisi dan letak kolom = **indeks** mod **ukuran sisi.** Dari indeks paling pertama dilakukan pengecekan terhadap indeks-indeks berikutnya apakah ada yang berada di baris yang sama, kolom yang sama, atau letaknya bersebelahan. Setelah itu, indeks berikutnya juga akan dicek dengan indeks-indeks yang berada di kanannya. Apabila terdapat 2 indeks yang tidak memenuhi maka pengecekan akan otomatis berhenti dengan mengembalikan false atau solusi ditolak. Apabila semua memenuhi maka akan dikembalikan nilai true atau solusi diterima.

Bab IV
Hasil dan Pembahasan

4.1. Analisis Kompleksitas

4.1.1. Brute Force Murni

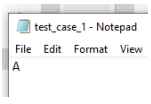
Pada kasus terburuk, penerapan algoritma *brute force* murni menghasilkan n^2Cn kemungkinan kombinasi **T(n)**=$\frac{n^2!}{n!*n!}$dan pada setiap kombinasi harus dilakukan pemeriksaan apakah memenuhi persyaratan Queens LinkedIn sejumlah **T(n)**= $\frac{(n-1)+1}{2} * n$. Oleh karena itu, berdasarkan perhitungan kompleksitas algoritma, ordenya adalah $n^2!*n^2$ tau O($n^2!*n^2$). Orde(n!) sudah merupakan orde paling kompleks dan n^2 mengakibatkan algoritma ini menjadi jauh lebih kompleks lagi. Dengan tambahan n^2, algoritma ini akan memerlukan waktu yang sangat lama dalam melakukan komputasi. Hipotesis ini dapat dibuktikan pada hasil percobaan pada subbab 4.2 menunjukkan bahwa semakin besar ukuran n semakin lama waktu komputasi yang diperlukan.

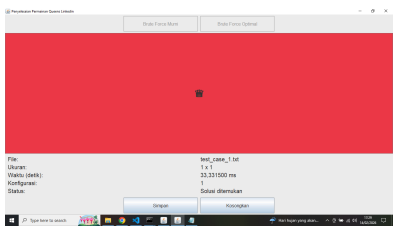
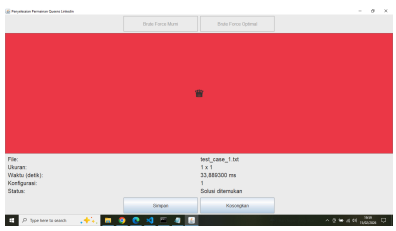
4.1.2. Brute Force Optimal

Pada algoritma yang telah dioptimalkan, orde kompleksitasnya untuk meninjau semua kemungkinan n!, yaitu **T(n)**=n!, karena terdapat calon lokasi penempatan yang tidak ditinjau lebih lanjut jika diketahui sudah melanggar beberapa aturan permainan. Oleh karena itu, T(n) untuk memeriksa apakah calon susunan valid sebagai jawaban juga berkurang signifikan, yaitu **T(n)**=k^2, $k < n$. Pada algoritma ini, terdapat tambahan **T(n)**= m, m sesuai dengan bidak yang tersusun, untuk mengecek validitas calon penempatan. Oleh karena itu, T(n) secara keseluruhan adalah **T(n)**= n!*k^2*m. O(n!*k^2*m) lebih sederhana dibandingkan O(n^2! *n^2). Hal ini dapat dibuktikan pula pada subbab 4.2 ketika input tumbuh menjadi semakin besar.

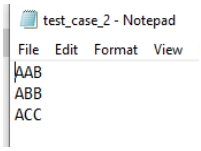
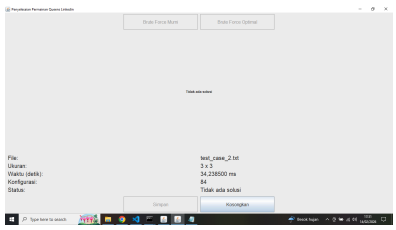
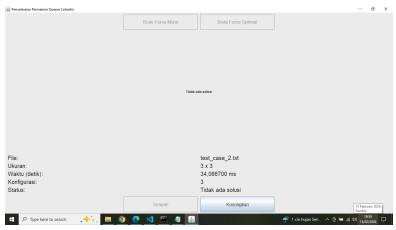
4.2. Hasil Percobaan

4.2.1. Ukuran n=1

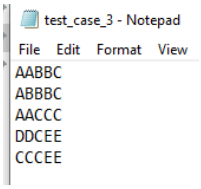
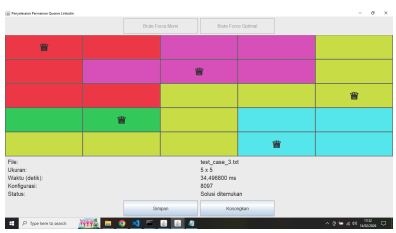
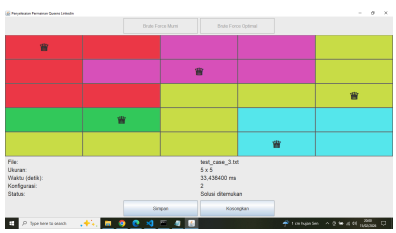
Jenis	Murni	Optimal
Input		

Output		

4.2.2. Ukuran n=3

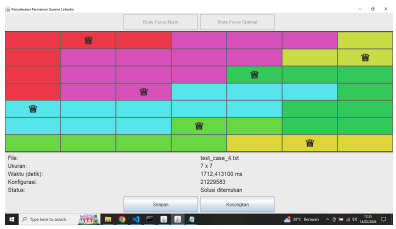
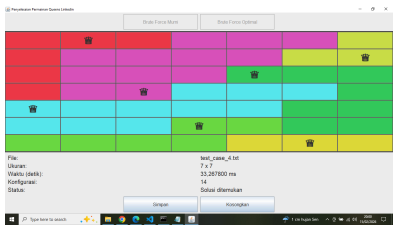
Jenis	Murni	Optimal
Input		
Output		

4.2.3. Ukuran n=5

Jenis	Murni	Optimal
Input		
Output		

4.2.4. Ukuran n=7

Jenis	Murni	Optimal

Input	test_case_4 - Notepad File Edit Format View AAABBBC ABBBBCC ABBBDDD ABBEEED EEEEEDD EEEFFDD FFFFGGG	
Output		

4.2.5. Ukuran n=9

Jenis	Murni	Optimal
Input	test_case_5 - Notepad File Edit Format View Help AAABBCCCD ABBBBCECD ABBBDCECD AAABDCCCD BBBBDDDDD FGGGDDHDD FGIGDDHDD FGIGDDHDD FGGGDDHHH	
Output		

4.2.6. Ukuran n=10

Jenis	Murni	Optimal
Input	test_case_6 - Notepad File Edit Format View AAABBCCCCD ABBBBCECDD ABBBDCECDD AAABDCCCCD BBBBBDDDDD FFGGGDDHDD FGIGDDHDDJ FGIGDDHDDJ FGGGDDHHHJ FFGGDDDJJJ	
Output		

Bab V
Penutup

5.1. Kesimpulan

Algoritma *brute force* murni dapat digunakan untuk menyelesaikan persoalan Queens LinkedIn, tetapi apabila ukuran papan bertambah semakin besar waktu komputasi yang diperlukan menjadi semakin lama, hal ini dapat terlihat ketika perubahan ukuran dari 5*5 ke 7*7. Bahkan saat ukuran meningkat ke 9*9 dibutuhkan waktu 4 jam lebih untuk menemukan solusi dan meledak hingga 25 jam untuk ukuran 10*10. Akan tetapi, dengan melakukan optimasi algoritma *brute force* dapat menyelesaikan permainan dengan waktu komputasi yang jauh lebih cepat.

5.2. Saran

Meskipun *brute force* selalu dapat menyelesaikan setiap permasalahan komputasi, penerapan *brute force* sebagai solusi utama tidak disarankan karena waktu komputasi akan menjadi sangat besar ketika terjadi peningkatan jumlah input dan umumnya permasalahan dunia nyata tentu melibatkan ukuran input yang besar.

Lampiran

Pranala ke kode sumber:
https://github.com/RayapSunggal/Tucil1_13524052.git

No	Poin	Ya	Tidak
1	Program berhasil di kompilasi 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	✓	

Kode Sumber

Nama File: program.java
Fungsi: Membaca input dan menyimpan output dalam bentuk gambar

```
import java.util.*;
import java.io.*;
import java.nio.charset.StandardCharsets;
import java.awt.Component;
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;
import javax.imageio.ImageIO;

public class program {

    static volatile char[][] preview=null;
    static volatile long previewCount=0;

    static long konfigurasi_ditinjau;
    static long update_terakhir=0;
    static final long interval=5_000L;

    static char[][] baca_masukan_txt(Scanner sc, String nama) {

        if (!nama.endsWith(".txt")){
            if (nama.endsWith(".png") || nama.endsWith(".jgeg") ||
nama.endsWith(".jpg")){
                throw new IllegalArgumentException("Program tidak bisa
membaca gambar sayangnya");
            }
            else{
                throw new IllegalArgumentException("Masukan tidak
valid");
            }
        }

        ArrayList<String> bacaan = new ArrayList<>();

        while (sc.hasNextLine()) {
            String baris=sc.nextLine().trim().replaceAll("\\s+", "");
```

```
            bacaan.add(baris);
        }

        if (bacaan.isEmpty()){
            throw new IllegalArgumentException("Masukan tidak valid");
        }

        int n=bacaan.size();

        for (int i=0;i<n;i++) {
            if (bacaan.get(i).length()!=n) {
                throw new IllegalArgumentException("Masukan tidak
valid");
            }
        }

        char[][] papan = new char[n][n];
        for (int r=0; r<n; r++) {
            papan[r]=bacaan.get(r).toCharArray();
        }

        Set<Character> jumlah_warna = new HashSet<>();

        for (int i=0; i<papan.length; i++) {
            for (int j=0; j<papan[i].length; j++) {
                jumlah_warna.add(papan[i][j]);
            }
        }


        if (jumlah_warna.size()!=papan.length) {
            throw new IllegalArgumentException("Warna dan ukuran papan
tidak sesuai");
        }

        return papan;
    }

    public static boolean cek_Quuens(int[] letak_bidak, int ukuran,
```

```
char[][] masukan){
        konfigurasi_ditinjau++;
        for(int k=0; k<ukuran-1; k++){
            int baris_bidak1=letak_bidak[k]/ukuran;
            int kolom_bidak1=letak_bidak[k]%ukuran;
            for (int l=k+1; l<ukuran; l++){
                int baris_bidak2=letak_bidak[l]/ukuran;
                int kolom_bidak2=letak_bidak[l]%ukuran;

                if
(masukan[baris_bidak1][kolom_bidak1]==masukan[baris_bidak2][kolom_bidak2
] ||  baris_bidak1==baris_bidak2 || kolom_bidak1==kolom_bidak2 ||
(Math.abs(kolom_bidak1-kolom_bidak2)==1 &&
Math.abs(baris_bidak1-baris_bidak2)==1)){
                    return false;
                }
            }
        }

        return true;
    }

    public static char[][] hasilkan_dan_cek_susunan(int mulai, int
jumlah_bidak_tersusun, int ukuran, int[] letak_bidak, char[][] masukan){
        if (jumlah_bidak_tersusun==ukuran){

            long now=System.nanoTime();
            if (now-update_terakhir>=interval){

                char[][] calon=new char[masukan.length][];

                for (int i=0; i<masukan.length; i++) {
                    calon[i]=masukan[i].clone();
                }

                for (int i=0; i<ukuran; i++) {

calon[letak_bidak[i]/ukuran][letak_bidak[i]%ukuran]='#';
                }
```

```
                preview=calon;
                previewCount=konfigurasi_ditinjau;
                update_terakhir=now;
            }

            if (cek_Quuens(letak_bidak, ukuran, masukan)){
                char[][] a=masukan;

                for (int i=0; i<ukuran; i++){
                    a[letak_bidak[i]/ukuran][letak_bidak[i]%ukuran]='#';
                }

                return a;
            }
            return null;
        }

        for (int i=mulai;
i<=(ukuran*ukuran)-ukuran+jumlah_bidak_tersusun; i++){ //misal di bidang
2*2 kalau masih 0 yang tersusun dia cuman bisa susun sampai kotak
ketiga(2), kalau udah 1 tersusun bisa sampai 3
            letak_bidak[jumlah_bidak_tersusun]=i;
            char[][] hasil=hasilkan_dan_cek_susunan(i+1,
jumlah_bidak_tersusun+1, ukuran, letak_bidak, masukan); //susun bidak
berikutnya
            if (hasil!=null){
                return hasil;
            }
        }
        return null;
    }

    public static void simpan_gambar_solusi(Component comp, File outPng)
throws Exception {
        int w=comp.getWidth();
        int h=comp.getHeight();

        if (w<=0 || h<=0) {
```

```
            w=comp.getPreferredSize().width;
            h=comp.getPreferredSize().height;
            if (w<=0) w=800;
            if (h<=0) h=800;
            comp.setSize(w, h);
            comp.doLayout();
        }

        BufferedImage img=new BufferedImage(w, h,
BufferedImage.TYPE_INT_ARGB);
        Graphics2D g2=img.createGraphics();
        comp.paint(g2);
        g2.dispose();

        ImageIO.write(img, "png", outPng);
    }

    public static void simpan_solusi_txt(char[][] papan, char[][]
solusi, File outTxt) throws Exception {
        try (PrintWriter pw=new PrintWriter(outTxt,
StandardCharsets.UTF_8)) {
            int n=papan.length;
            pw.println("Ukuran: " + n + " x " + n);
            pw.println();

            pw.println("Konfigurasi Awal:");
            for (int i=0; i<n; i++) {
                pw.println(new String(papan[i]));
            }
            pw.println();

            pw.println("Solusi (# = Queen):");
            for (int i=0; i<n; i++) {
                pw.println(new String(solusi[i]));
            }
        }
    }

}
```

Nama File: programOptimal.java
Fungsi: Penambahan optimasi, yaitu memeriksa validitas calon posisi sebelum mengevaluasi susunan akhir

```
public class programOptimal {

    static volatile char[][] preview=null;
    static volatile long previewCount=0;

    static long konfigurasi_ditinjau;
    static long update_terakhir=0;
    static final long interval=5_000L;

    public static boolean cek_Quuens(int[] letak_bidak, int ukuran,
char[][] masukan){
        konfigurasi_ditinjau++;
        for(int k=0; k<ukuran-1; k++) {
            int baris_bidak1=letak_bidak[k]/ukuran;
            int kolom_bidak1=letak_bidak[k]%ukuran;
            for (int l=k+1; l<ukuran; l++) {
                int baris_bidak2=letak_bidak[l]/ukuran;
                int kolom_bidak2=letak_bidak[l]%ukuran;

                if
(masukan[baris_bidak1][kolom_bidak1]==masukan[baris_bidak2][kolom_bidak2
] ||  baris_bidak1==baris_bidak2 || kolom_bidak1==kolom_bidak2 ||
(Math.abs(kolom_bidak1-kolom_bidak2)==1 &&
Math.abs(baris_bidak1-baris_bidak2)==1)) {
                    return false;
                }
            }
        }

        return true;
    }

    public static boolean petak_sudah_ada (char[][] papan, int[]
```

```
letak_bidak, int idx, int n, int jumlah_bidak_tersusun){
        for (int i=0; i<jumlah_bidak_tersusun; i++){
            if (letak_bidak[i]%n==idx%n ||
papan[letak_bidak[i]/n][letak_bidak[i]%n]==papan[idx/n][idx%n]) return
true;
        }
        return false;
    }

    public static char[][] hasilkan_dan_cek_susunan(int mulai, int
jumlah_bidak_tersusun, int ukuran, int[] letak_bidak, char[][] masukan){
        if (jumlah_bidak_tersusun==ukuran) {


            long now=System.nanoTime();
            if (now-update_terakhir>=interval) {

                char[][] calon=new char[masukan.length][];

                for (int i=0; i<masukan.length; i++) {
                    calon[i]=masukan[i].clone();
                }

                for (int i=0; i<ukuran; i++) {

calon[letak_bidak[i]/ukuran][letak_bidak[i]%ukuran]='#';
                }

                preview=calon;
                previewCount=konfigurasi_ditinjau;
                update_terakhir=now;
            }

            if (cek_Quuens(letak_bidak, ukuran, masukan)) {
                char[][] a=masukan;

                for (int i=0; i<ukuran; i++) {
                    a[letak_bidak[i]/ukuran][letak_bidak[i]%ukuran]='#';
                }
```

```
                return a;
            }
            return null;
        }

        for (int i=mulai;
i<=(ukuran*ukuran)-ukuran+jumlah_bidak_tersusun; i++) { //misal di
bidang 2*2 kalau masih 0 yang tersusun dia cuman bisa susun sampai kotak
ketiga(2), kalau udah 1 tersusun bisa sampai 3
            if(i>=ukuran*jumlah_bidak_tersusun &&
i<jumlah_bidak_tersusun*ukuran+ukuran && !petak_sudah_ada(masukan,
letak_bidak, i, ukuran, jumlah_bidak_tersusun)) { // optimasi
                letak_bidak[jumlah_bidak_tersusun]=i;
                char[][] hasil=hasilkan_dan_cek_susunan(i+1,
jumlah_bidak_tersusun+1, ukuran, letak_bidak, masukan); //susun bidak
berikutnya
                if (hasil!=null){
                    return hasil;
                }
            }

        }
        return null;
    }
}
```

Nama File: programGUI.java
Fungsi: Tampilan GUI untuk program

```
import java.awt.*;
import javax.swing.*;
import javax.swing.border.LineBorder;
import java.util.ArrayList;
import java.util.Scanner;
import java.io.File;
import java.nio.charset.StandardCharsets;
```

```
public class programGUI {

    private JFrame frame;
    private JPanel boardContainer;
    private JLabel nama_file, ukuran, waktu_komputasi, jumlah_percobaan,
status;
    private JButton uploadButton1, uploadButton2, saveButton,
clearButton;

    private char[][] lastPapan = null;
    private char[][] lastSolusi = null;
    private JPanel lastVisualPanel = null;
    private File lastInputFile = null;

    static final Color[] warna=new Color[] {
        new Color(235, 59, 70),
        new Color(217, 82, 186),
        new Color(200, 224, 72),
        new Color(51, 204, 90),
        new Color(89, 230, 235),
        new Color(107, 217, 69),
        new Color(224, 219, 56),
        new Color(78, 204, 184),
        new Color(75, 235, 168),
        new Color(159, 54, 217),
        new Color(85, 121, 224),
        new Color(204, 155, 65),
        new Color(81, 235, 59),
        new Color(204, 217, 82),
        new Color(224, 72, 180),
        new Color(51, 204, 110),
        new Color(117, 235, 89),
        new Color(174, 69, 217),
        new Color(186, 224, 56),
        new Color(204, 102, 78),
        new Color(235, 188, 75),
        new Color(54, 217, 86),
        new Color(224, 166, 85),
        new Color(65, 137, 204),
```

```
        new Color(59, 138, 235),
        new Color(217, 100, 82)
    };

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new programGUI().start());
    }

    private void start() {
        frame=new JFrame("Penyelesaian Permainan Queens Linkedin ");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(600,600);

        uploadButton1=new JButton("Brute Force Murni");
        uploadButton1.setFont(new Font(Font.SANS_SERIF, Font.PLAIN,
16));
        uploadButton1.setPreferredSize(new Dimension(260,55));

        uploadButton2=new JButton("Brute Force Optimal");
        uploadButton2.setFont(new Font(Font.SANS_SERIF, Font.PLAIN,
16));
        uploadButton2.setPreferredSize(new Dimension(260,55));

        boardContainer=new JPanel(new BorderLayout());

        frame.setLayout(new BorderLayout());

        JPanel topPanel=new JPanel(new
FlowLayout(FlowLayout.CENTER,10,5));
        topPanel.add(uploadButton1);
        topPanel.add(uploadButton2);

        frame.setLayout(new BorderLayout());
        frame.add(topPanel, BorderLayout.NORTH);
        frame.add(boardContainer, BorderLayout.CENTER);

        JPanel statsPanel = new JPanel(new GridLayout(5,2,10,2));

statsPanel.setBorder(BorderFactory.createEmptyBorder(6,10,6,10));
```

```
        statsPanel.add(new JLabel("File:"));
        nama_file=new JLabel("-");
        statsPanel.add(nama_file);

        statsPanel.add(new JLabel("Ukuran:"));
        ukuran=new JLabel("-");
        statsPanel.add(ukuran);

        statsPanel.add(new JLabel("Waktu (detik):"));
        waktu_komputasi=new JLabel("-");
        statsPanel.add(waktu_komputasi);

        statsPanel.add(new JLabel("Konfigurasi:"));
        jumlah_percobaan=new JLabel("-");
        statsPanel.add(jumlah_percobaan);

        statsPanel.add(new JLabel("Status:"));
        status=new JLabel("Pilih file input untuk mulai");
        statsPanel.add(status);

        Font f=new Font(Font.SANS_SERIF,Font.PLAIN,20);

        for (Component comp : statsPanel.getComponents()) {
            if (comp instanceof JLabel) {
                comp.setFont(f);
            }
        }

        saveButton=new JButton("Simpan");
        saveButton.setFont(new Font(Font.SANS_SERIF, Font.PLAIN, 16));
        saveButton.setPreferredSize(new Dimension(260, 55));
        saveButton.setEnabled(false);

        clearButton=new JButton("Kosongkan");
        clearButton.setFont(new Font(Font.SANS_SERIF, Font.PLAIN, 16));
        clearButton.setPreferredSize(new Dimension(260, 55));
        clearButton.setEnabled(false);
```

```
        JPanel bottomButtonsPanel=new JPanel(new
FlowLayout(FlowLayout.CENTER, 10, 5));
        bottomButtonsPanel.add(saveButton);
        bottomButtonsPanel.add(clearButton);

        JPanel bottomContainer=new JPanel();
        bottomContainer.setLayout(new BoxLayout(bottomContainer,
BoxLayout.Y_AXIS));
        bottomContainer.add(statsPanel);
        bottomContainer.add(bottomButtonsPanel);

        frame.add(bottomContainer, BorderLayout.SOUTH);

        uploadButton1.addActionListener(e -> onUpload1());
        uploadButton2.addActionListener(e -> onUpload2());

        saveButton.addActionListener(e -> doSave());
        clearButton.addActionListener(e -> doClear());

        frame.setVisible(true);
    }

    private void onUpload1() {
        JFileChooser chooser=new JFileChooser();
        int res=chooser.showOpenDialog(frame);
        if (res!=JFileChooser.APPROVE_OPTION){
            return;
        }

        File file=chooser.getSelectedFile();
        String nama=file.getName().toLowerCase();

        try (Scanner sc=new Scanner(file, StandardCharsets.UTF_8)) {
            char[][] papan;

            papan=program.baca_masukan_txt(sc, nama);

            uploadButton1.setEnabled(false);
```

```
            uploadButton2.setEnabled(false);

            char[][] copy_papan=new char[papan.length][];
            for (int i=0; i<papan.length; i++){
                copy_papan[i]=papan[i].clone();
            }

            int n=papan.length;
            int[] letak=new int[n];

            program.konfigurasi_ditinjau=0;
            program.preview=null;
            program.previewCount=0;
            program.update_terakhir=System.nanoTime();

            status.setText("Memproses (Murni)");
            nama_file.setText(file.getName());
            ukuran.setText(String.format("%d x %d", n, n));
            waktu_komputasi.setText("-");
            jumlah_percobaan.setText("0");

            Timer timer=new Timer(500, ev -> {
                char[][] snap=program.preview;
                if (snap!=null) {
                    boardContainer.removeAll();
                    boardContainer.add(visualisasi_papan(copy_papan,
snap), BorderLayout.CENTER);
                    boardContainer.revalidate();
                    boardContainer.repaint();


jumlah_percobaan.setText(String.valueOf(program.previewCount));
                }
            });
            timer.start();

            SwingWorker<char[][], Void> worker = new SwingWorker<>() {
                long mulai;
```

```
                @Override
                protected char[][] doInBackground() {
                    mulai=System.nanoTime();
                    return program.hasilkan_dan_cek_susunan(0, 0, n,
letak, papan);
                }

                @Override
                protected void done() {
                    timer.stop();

                    try {
                        char[][] solusi=get();
                        long selesai=System.nanoTime();
                        double detik=(selesai-mulai)/1_000_000.0;

                        boardContainer.removeAll();
                        if (solusi==null) {
                            boardContainer.add(new JLabel("Tidak ada
solusi", SwingConstants.CENTER), BorderLayout.CENTER);
                            status.setText("Tidak ada solusi");
                        }
                        else {

boardContainer.add(visualisasi_papan(copy_papan, solusi),
BorderLayout.CENTER);
                            status.setText("Solusi ditemukan");
                        }
                        boardContainer.revalidate();
                        boardContainer.repaint();

                        waktu_komputasi.setText(String.format("%.6f ms",
detik));

jumlah_percobaan.setText(String.valueOf(program.konfigurasi_ditinjau));

                        lastPapan=copy_papan;
                        lastSolusi=solusi;
                        lastInputFile=file;
```

```
                        Component c=(boardContainer.getComponentCount()
> 0) ? boardContainer.getComponent(0) : null;
                        lastVisualPanel=(c instanceof JPanel) ? (JPanel)
c : null;

                        saveButton.setEnabled(lastVisualPanel != null);
                        clearButton.setEnabled(true);

                    }
                    catch (Exception ex) {

JOptionPane.showMessageDialog(frame,ex.toString(),"Error",JOptionPane.ER
ROR_MESSAGE);
                        status.setText("Error");
                    }
                }
            };

            worker.execute();
        }
        catch (IllegalArgumentException ex) {
            JOptionPane.showMessageDialog(frame,ex.getMessage(),"Masukan
tidak valid",JOptionPane.ERROR_MESSAGE);
        }
        catch (Exception ex) {

JOptionPane.showMessageDialog(frame,ex.toString(),"Error",JOptionPane.ER
ROR_MESSAGE);
        }
    }

    private void onUpload2() {
        JFileChooser chooser=new JFileChooser();
        int res=chooser.showOpenDialog(frame);
        if (res!=JFileChooser.APPROVE_OPTION){
            return;
        }
```

```
        File file=chooser.getSelectedFile();

        String nama=file.getName().toLowerCase();

        try (Scanner sc=new Scanner(file, StandardCharsets.UTF_8)) {
            char[][] papan;

            papan=program.baca_masukan_txt(sc, nama);

            uploadButton1.setEnabled(false);
            uploadButton2.setEnabled(false);

            char[][] copy_papan = new char[papan.length][];
            for (int i=0; i<papan.length; i++){
                copy_papan[i] = papan[i].clone();
            }

            int n = papan.length;
            int[] letak = new int[n];

            programOptimal.konfigurasi_ditinjau = 0;
            programOptimal.preview = null;
            programOptimal.previewCount = 0;
            programOptimal.update_terakhir = System.nanoTime();

            status.setText("Memproses (Optimal)");
            nama_file.setText(file.getName());
            ukuran.setText(String.format("%d x %d", n, n));
            waktu_komputasi.setText("-");
            jumlah_percobaan.setText("0");

            Timer timer = new Timer(500, ev -> {
                char[][] snap = programOptimal.preview;
                if (snap != null) {
                    boardContainer.removeAll();
                    boardContainer.add(visualisasi_papan(copy_papan,
snap), BorderLayout.CENTER);
                    boardContainer.revalidate();
                    boardContainer.repaint();
```

```
jumlah_percobaan.setText(String.valueOf(programOptimal.previewCount));
                }
            });
            timer.start();


            SwingWorker<char[][], Void> worker = new SwingWorker<>() {
                long mulai;

                @Override
                protected char[][] doInBackground() {
                    mulai=System.nanoTime();
                    return programOptimal.hasilkan_dan_cek_susunan(0, 0,
n, letak, papan);
                }

                @Override
                protected void done() {
                    timer.stop();

                    try {
                        char[][] solusi=get();
                        long selesai=System.nanoTime();
                        double detik=(selesai-mulai)/1_000_000.0;

                        boardContainer.removeAll();
                        if (solusi==null) {
                            boardContainer.add(new JLabel("Tidak ada
solusi", SwingConstants.CENTER), BorderLayout.CENTER);
                            status.setText("Tidak ada solusi");
                        }
                        else {

boardContainer.add(visualisasi_papan(copy_papan, solusi),
BorderLayout.CENTER);
                            status.setText("Solusi ditemukan");
                        }
```

```
                        boardContainer.revalidate();
                        boardContainer.repaint();

                        waktu_komputasi.setText(String.format("%.6f ms",
detik));

jumlah_percobaan.setText(String.valueOf(programOptimal.konfigurasi_ditin
jau));

                        lastPapan=copy_papan;
                        lastSolusi=solusi;
                        lastInputFile=file;

                        Component c=(boardContainer.getComponentCount()
> 0) ? boardContainer.getComponent(0) : null;
                        lastVisualPanel=(c instanceof JPanel) ? (JPanel)
c : null;

                        saveButton.setEnabled(lastVisualPanel != null);
                        clearButton.setEnabled(true);

                    }
                    catch (Exception ex) {

JOptionPane.showMessageDialog(frame,ex.toString(),"Error",JOptionPane.ER
ROR_MESSAGE);
                        status.setText("Error");
                    }
                }
            };

            worker.execute();

        }
        catch (IllegalArgumentException ex) {
            JOptionPane.showMessageDialog(frame,ex.getMessage(),"Masukan
tidak valid",JOptionPane.ERROR_MESSAGE);
        }
        catch (Exception ex) {
```

```
JOptionPane.showMessageDialog(frame,ex.toString(),"Error",JOptionPane.ERROR_MESSAGE);
        }
    }

    static char[] daftar_warna(char[][] papan) {
        ArrayList<Character> unik=new ArrayList<>();
        int n=papan.length;

        for (int i=0; i<n; i++) {
            for (int j=0; j<n; j++) {
                char ch=papan[i][j];
                if (!unik.contains(ch)) {
                    unik.add(ch);
                }
            }
        }

        char[] hasil=new char[unik.size()];
        for (int i=0; i<unik.size(); i++) {
            hasil[i]=unik.get(i);
        }

        return hasil;
    }

    static int warnai(char[] arr, char target) {
        for (int i=0; i<arr.length; i++) {
            if (arr[i]==target){
                return i;
            }
        }
        return -1; //ga akan kepakai
    }

    static JPanel visualisasi_papan(char[][] papan, char[][] solusi) {
        int n=papan.length;
```

```
        char[] warnaUnik=daftar_warna(papan);

        JPanel grid=new JPanel(new GridLayout(n,n,1,1));
        grid.setBackground(Color.DARK_GRAY);

        for (int i=0; i<n; i++) {
            for (int j=0; j<n; j++) {
                char ch=papan[i][j];

                JLabel cell=new JLabel("", SwingConstants.CENTER);
                cell.setOpaque(true);
                cell.setBorder(new LineBorder(Color.GRAY, 1));
                cell.setFont(new Font(Font.SANS_SERIF, Font.BOLD, 18));

                int idx=warnai(warnaUnik,ch);
                Color bg=warna[idx];
                cell.setBackground(bg);

                if (solusi[i][j]=='#') {
                    cell.setText("♛");
                    cell.setFont(new
Font(Font.SANS_SERIF,Font.BOLD,36));
                }

                grid.add(cell);
            }
        }

        return grid;
    }

    private void doClear() {
        boardContainer.removeAll();
        boardContainer.revalidate();
        boardContainer.repaint();

        nama_file.setText("-");
        ukuran.setText("-");
        waktu_komputasi.setText("-");
```

```
        jumlah_percobaan.setText("-");
        status.setText("Pilih file input untuk mulai");

        lastPapan=null;
        lastSolusi=null;
        lastVisualPanel=null;
        lastInputFile=null;

        uploadButton1.setEnabled(true);
        uploadButton2.setEnabled(true);
        saveButton.setEnabled(false);
        clearButton.setEnabled(false);
    }

    private void doSave() {
        if (lastVisualPanel==null || lastPapan==null ||
lastSolusi==null) {
            JOptionPane.showMessageDialog(frame, "Tidak ada solusi untuk
disimpan", "Info", JOptionPane.INFORMATION_MESSAGE);
            return;
        }

        JFileChooser chooser=new JFileChooser();
        chooser.setDialogTitle("Pilih folder untuk menyimpan");
        chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);

        int res=chooser.showSaveDialog(frame);
        if (res!=JFileChooser.APPROVE_OPTION) return;

        File dir=chooser.getSelectedFile();

        String baseName=(lastInputFile!=null) ? lastInputFile.getName()
: "hasil";

        int dot=baseName.lastIndexOf('.');
        if (dot!=-1) baseName = baseName.substring(0, dot);

        File outPng=new File(dir, baseName + "_solusi_gambar.png");
        File outTxt=new File(dir, baseName + "_solusi_dokumen.txt");
```

```
        try {
            program.simpan_gambar_solusi(lastVisualPanel, outPng);
            program.simpan_solusi_txt(lastPapan, lastSolusi, outTxt);

            JOptionPane.showMessageDialog(frame,
                    "Tersimpan:\n- " + outPng.getName() + "\n- " +
outTxt.getName(),
                    "Berhasil", JOptionPane.INFORMATION_MESSAGE);

            doClear();

        }
        catch (Exception ex) {
            JOptionPane.showMessageDialog(frame, ex.toString(), "Error",
JOptionPane.ERROR_MESSAGE);
        }
    }

}
```

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

Raynard Fausta