Penyelesaian Permainan Queens Linkedin

Tugas Kecil 1 IF2211 Strategi Algoritma

Semester II tahun 2025/2026

Nama: Muhammad Rafi Akbar

NIM: 13524125

1. **Penjelasan Algoritma**

Program Queen Solver adalah sebuah program yang dibuat dengan bahasa Java, dengan Algoritma Brute Force Exhaustive Search, yang mengiterasi semua kemungkinan satu per-satu hingga menemukan jawaban yang benar, Dengan Total Kombinasi N^N iterasi.

Langkah

1. Program menerima input berupa file txt berisi teks A-Z yang membentuk persegi. Program menggunakan **BufferedReader** untuk membaca file baris per baris. Setiap baris yang tidak kosong langsung diubah menjadi array karakter (**char[]**) menggunakan **toCharArray(),** lalu disimpan sementara ke dalam **List<char[]>** sebagai list, karena list bersifat dinamis, dan lalu setelah selesai membaca semua isi file, diubah menjadi **char[][].**
2. Lalu mulailah proses validasi apabila file error/kosong, ukuran tidak persegi, dan menggunakan format yang benar.
3. Setelah semua berhasil terbaca, kita menciptakan class **Point** yang digunakan untuk menyimpan koordinat setiap sel di papan, lalu sel-sel yang punya warna yang sama dikumpulkan bersama menjadi satu region.
4. Setelah membuat region, program akan mulai mencari solusi penempatan Queen, namun sebelum itu, dilakukan penentuan frekuensi live update berdasarkan ukuran board, dan juga inisialisasi **interationCount** dan **StartTime**.
5. Program lalu mulai melakukan pencarian menggunakan **Exhaustive Search,** yang akan mencoba semua kemungkinan kolom dari (0 – N-1). Ketika row = size, maka semua baris sudah terisi, dan dilanjut ke validasi penempatan queen (apakah sudah mengikuti aturan penempatan dengan benar). Melalui fungsi **IsValid**
6. Fungsi IsValid memanggil ketiga helper yaitu **checkcolumn** untuk memastikan hanya ada 1 queen di tiap row / column, **checkcolor** untuk memastikan satu warna hanya memiliki 1 queen, dan **checkdiagonal** untuk memastikan tidak ada queen yang bersebelahan secara diagonal.
7. Program akan menunjukan Board dan juga live update melalui terminal, yang dimana exhaust search yang dilakukan akan berhenti ketika menemukan solusi pertama yang valid (**CurrentSolution = queen**). Lalu kita akan dapat menyimpan jawaban sebagai file txt dengan nama yang kita tentukan.

2. **Implementasi**

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.FileReader;
import java.util.ArrayList;
import java.util.List;
import java.util.HashMap;
import java.util.Map;
```

```java
import java.util.Scanner;

class Point {
    int row,col;
    Point(int row, int col) {
        this.row = row;
        this.col = col;
    }

    @Override
    public String toString() {
        return "("+row+","+col+")";
    }
}

public class queen {

    private static int iterationCount = 0;
    private static long startTime;
    private static long endTime;
    private static Boolean[][] CurrentSolution;

    // baca input file txt
    public static char[][] bacatext(String filepath) {
        List<char[]> Tarray = new ArrayList<>();
        try(BufferedReader reader = new BufferedReader(new FileReader(filepath))) {
            String line;
            while ((line = reader.readLine()) != null) {
                if (!line.trim().isEmpty()) {
                    Tarray.add(line.trim().toCharArray());
                }
            }
        }

        // validasi error/Kosong
        catch (IOException e) {
            System.err.println("Error baca file: " + e.getMessage());
            return null;
        }

        if (Tarray.isEmpty()) {
            System.out.println("File kosong.");
        }

        char[][] map = new char[Tarray.size()][];
        for (int i = 0; i < Tarray.size(); i++) {
            map[i] = Tarray.get(i);
        }
        return map;


    }

    // cek ukuran map nya persegi
    public static boolean validmap(char[][] map) {
        int rows = map.length;
        for(int i = 0; i < rows; i++) {
            if (map[i].length != rows) {
                // panjang beda sama lebar
                return false;
            }
        }
        return true;
    }

    // cek misal ada angka, simbol, dll di map
    public static boolean validformat(char[][] map) {
        for(int i = 0; i < map.length; i++) {
            for (int j = 0; j < map[i].length; j++) {
                char m = map[i][j];
                if (m < 'A' || m > 'Z') {
                    return false;
                }
            }
        }
        return true;
    }
```

```java
    // cek pembagian warna per daerah
    public static Map<Character, List<Point>> checkregion(char[][] map) {
        int size = map.length;
        Map<Character, List<Point>> regions = new HashMap<>();

        for (int i = 0; i < size; i++) {
            for (int j = 0; j < size; j++) {
                char color = map[i][j];

                if(!regions.containsKey(color)) {
                    regions.put(color, new ArrayList<>());
                }

                regions.get(color).add(new Point(i, j));
            }
        }


        if (regions.size() != size) {
            System.err.println("Error:Jumlah Warna Tidak sesuai dengan ukuran board");
            return null;
        }

        return regions;
    }

    // print board
    public static void printboard(char[][] map) {
        System.out.println("\nBoard:");
        for(int i = 0; i < map.length; i++) {
            for (int j = 0; j < map[i].length; j++) {
                System.out.print(map[i][j] + " ");
            }
            System.out.println();
        }
    }

    // Helper Validasi posisi Queens
    public static boolean checkcolumn(Boolean[][] queen, int row, int col) {
        int size = queen.length;
        for (int i = 0; i < size; i++) {
            if (i == row) continue;
            if (queen[i][col]) {
                return false;
            }
        }
        return true;
    }

    public static boolean checkcolor(char[][] map, Map<Character, List<Point>> regions,
Boolean[][] queen, int row, int col) {
        char color = map[row][col];
        List<Point> regionCL = regions.get(color);
        for (Point p : regionCL) {
            if (p.row == row && p.col == col) continue;
            if(queen[p.row][p.col]) {
                return false;
            }
        }
        return true;
    }

    public static boolean checkdiagonal(Boolean[][] queen, int row, int col) {
        int size = queen.length;
        int[] dRow = {-1, -1, -1, 0, 0, 1, 1, 1};
        int[] dCol = {-1, 0, 1, -1, 1, -1, 0, 1};

        for (int i = 0; i < 8; i++){
            int newRow = row + dRow[i];
            int newCol = col + dCol[i];

            if (newRow >= 0 && newRow < size && newCol >= 0 && newCol < size) {
                if (queen[newRow][newCol]) {
                    return false;

                }
```

```
            }
        }
        return true;
    }

    //Backtrack (gajadi dipake)
    public static boolean back(char[][]map, Map<Character, List<Point>> regions, Boolean[][]
queen, int row) {
        int size = map.length;
        if (row == size) {
            return true;
        }
        for (int col = 0; col < size; col++) {
            iterationCount++;
            if (iterationCount % 1000 == 0) {
                System.out.println("Iterasi: " + iterationCount);
            }
            if (isvalid(map, regions, queen, row,col)) {
                queen[row][col] = true;

                if (back(map, regions, queen, row + 1 )) {
                    return true;
                }
                queen[row][col] = false;
            }
        }
        return false;
    }

    public static boolean isvalid(char[][] map, Map<Character, List<Point>> regions,
Boolean[][] queen, int row, int col) {
        return checkcolumn(queen, row, col) && checkcolor(map,regions,queen,row,col) &&
checkdiagonal(queen,row,col);
    }

    //exhaust search

    public static boolean solution(char[][] map,Map<Character, List<Point>> regions,
Boolean[][] queen) {
        int size = map.length;
        for (int row = 0; row < size; row++) {
            for (int col = 0; col < size; col++) {
                if (queen[row][col]) {
                    if (!isvalid(map, regions, queen, row, col)) {
                        return false;
                    }
                }
            }
        }
        return true;
    }

    public static boolean exhaustSearch(char[][] map,Map<Character, List<Point>> regions,int[]
placement,int row,int updateFreq) {
        int size = map.length;

        if (row == size) {
            iterationCount++;

            if (iterationCount % updateFreq == 0) {
                clearscreen();
                System.out.println("Live Update:\n");

                Boolean[][] tempQueen = new Boolean[size][size];
                for (int i = 0; i < size; i++)
                    for (int j = 0; j < size; j++)
                        tempQueen[i][j] = false;
                for (int r = 0; r < size; r++)
                    tempQueen[r][placement[r]] = true;

                update(map, tempQueen);

                long duration = System.currentTimeMillis() - startTime;
                System.out.println("\n Iterasi : " + iterationCount);
                System.out.println("Waktu   : " + duration + " ms");

                try {
```

```
                    Thread.sleep(100); // naikin kalo pengen lebih lama
                } catch (InterruptedException e) {
                    Thread.currentThread().interrupt();
                }
            }

            Boolean[][] queen = new Boolean[size][size];
            for (int i = 0; i < size; i++) {
                for (int j = 0; j < size; j++) {
                    queen[i][j] = false;
                }
            }
            for (int r = 0; r < size; r++) {
                queen[r][placement[r]] = true;
            }
            if (solution(map, regions, queen)) {
                CurrentSolution = queen;
                return true;
            }

            return false;
        }

        for (int col = 0; col < size; col++) {
            placement[row] = col;
            if (exhaustSearch(map, regions, placement, row + 1, updateFreq)) {
                return true;
            }
        }

        return false;
    }

    // tunjukin Hasil
    public static void printsolution(char[][] map, Boolean[][] queen) {
        int size = queen.length;
        System.out.println("\nSolusi:");
        for (int i = 0; i < size; i++) {
            for (int j = 0; j < map[i].length; j++) {
                if (queen[i][j]) {
                    System.out.print("# ");
                }
                else {
                    System.out.print(map[i][j] + " ");
                }
            }
            System.out.println();
        }
    }

    public static void printtime() {
        long duration = endTime - startTime;
        System.out.println(" Waktu: " + duration + " ms");
        System.out.println(" Iterasi: " + iterationCount);
    }

    // live update
    public static void clearscreen() {
    System.out.print("\033[H\033[2J");  // Hapus terminal harusnya
    System.out.flush();
    }

    public static void update(char[][] map, Boolean[][] queen) {
        int size = map.length;
        for (int i = 0; i < size; i++) {
            for (int j = 0; j < size; j++) {
                if (queen[i][j] != null && queen[i][j]) {
                    System.out.print("# ");

                }

                else {
                    System.out.print(map[i][j] + " ");
                }
            }
            System.out.println();
        }
```

```
    }

    // save
    public static void savesolution(char[][] map, Boolean[][] queen, String filename) {
        try (java.io.PrintWriter writer = new java.io.PrintWriter(new
java.io.FileWriter(filename))) {
                int size = queen.length;
                for (int i = 0; i < size; i++) {
                    for (int j = 0; j < map[i].length; j++) {
                        if (queen[i][j]) {
                            writer.print('#');
                        } else {
                            writer.print(map[i][j]);
                        }
                    }
                    writer.println();
                }

                writer.println();
                writer.println("Waktu pencarian: " + (endTime - startTime) + " ms");
                writer.println("Banyak kasus: " + iterationCount + " kasus");

                System.out.println("Solusi disimpan ke: " + filename);

            }

            catch (java.io.IOException e) {
                System.err.println("Gagal menyimpan: " + e.getMessage());
            }
   }


    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Masukkan path file: ");
        String filepath = scanner.nextLine();

        System.out.println("Queen Solver\n");
        System.out.println("Memvalidasi Board...");

        char[][] map = bacatext(filepath);
        if (map == null) {
            scanner.close();
            return;
        }

        if (!validmap(map)) {
            System.err.println("Error: Board tidak berbentuk persegi.");
            scanner.close();
            return;
        }

        if (!validformat(map)) {
            System.err.println("Error: Board mengandung karakter tidak valid.");
            scanner.close();
            return;
        }

        Map<Character, List<Point>> regions = checkregion(map);
        if (regions == null) {
            scanner.close();
            return;
        }

        System.out.println("Board Valid.... menunjukan Board");
        printboard(map);

        System.out.println("\n Menyeesaikan masalah...");

        int size = map.length;
        int updateFreq;
        if (size <= 5) {
            updateFreq = 100;
        } else if (size <= 7) {
            updateFreq = 10000;
        } else {
```

```
            updateFreq = 10000000;
        }

        iterationCount = 0;
        startTime = System.currentTimeMillis();

        int[] placement = new int[size];

        Boolean found = exhaustSearch(map, regions, placement, 0, updateFreq);

        endTime = System.currentTimeMillis();

        if (found) {
        clearscreen();
        printboard(map);
        printsolution(map, CurrentSolution);
        }

        else {
        System.out.println("Tidak ada solusi!");
        }

        printtime();


        if (found) {
            System.out.println("Apakah anda ingin Save Jawaban? (Y/n)");
            String jawab = scanner.nextLine();

            if(jawab.equalsIgnoreCase("Y") || jawab.equalsIgnoreCase("y")) {
                System.out.println("Nama file output:");
                String outputFile = scanner.nextLine();
                savesolution( map, CurrentSolution, outputFile);

            }
        }

        scanner.close();

    }
}
```

3. Test Case

```
Masukkan path file: test/test2.txt
Queen Solver

Memvalidasi Board...
Board Valid.... menunjukan Board
```

```
Board:
A A B B
A A B B
C C D D
C C D D

 Menyeesaikan masalah...
```

```
Live Update:

A # B B
A A # B
# C D D
C C D #

 Iterasi : 100
Waktu   : 1 ms
```

```
Solusi:
A # B B
A A B #
# C D D
C C # D
 Waktu: 103 ms
 Iterasi: 115
Apakah anda ingin Save Jawaban? (Y/n)
n
```

```
Masukkan path file: test/test1.txt
Queen Solver

Memvalidasi Board...
Board Valid.... menunjukan Board

Board:
A A A B B C C C D
A B B B B C E C D
A B B B D C E C D
A A A B D C C C D
B B B B D D D D D
F G G G D D H D D
F G I G D D H D D
F G I G D D H D D
F G G G D D H H H

 Menyeesaikan masalah...
Live Update:
```

```
A A A B B C C # D
A # B B B C E C D
A B B B D C E # D
A A # B D C C C D
B B B B D D D # D
F G G G D D H D #
# G I G D D H D D
F G I G # D H D D
F G G # D D H H H

 Iterasi : 310000000
Waktu   : 96276 ms
Live Update:

A A A B B C C # D
A B B # B C E C D
A B B B D C E C #
A # A B D C C C D
B B # B D D D D D
# G G G D D H D D
F G I G # D H D D
F G # G D D H D D
F G G G # D H H H

 Iterasi : 320000000
Waktu   : 99381 ms
```

```
Solusi:
A A A B B C C # D
A B B B # C E C D
A B B B D C # C D
A # A B D C C C D
B B B B D # D D D
F G G # D D H D D
# G I G D D H D D
F G # G D D H D D
F G G G D D H H #
 Waktu: 100612 ms
 Iterasi: 323741637
Apakah anda ingin Save Jawaban? (Y/n)
Y
Nama file output:
hasil1
Solusi disimpan ke: hasil1
```

4. **Link Repository**

   https://github.com/Fibarrr/Tucil1_13524125

5. Lampiran

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

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