Hexagonal Grid Radar Project Report

1. Project Overview

This project involves implementing a **hexagonal grid-based radar system** to detect and analyze intersecting regions of radar responses. The goal is to calculate the overlapping regions based on multiple radar inputs and their respective ranges. The hexagonal grid is represented using an axial coordinate system for simplicity and efficiency in distance calculations.

2. Objectives

The main objectives of this project are:

1. Implement a coordinate system for a **pointy-top hexagonal grid**.
2. Develop a mechanism to calculate and store radar response regions based on user input.
3. Calculate the intersection of radar response regions to identify overlapping cells.
4. Output the number and coordinates of intersecting cells.

3. Implementation Details

3.1 Coordinate System

We used the **axial coordinate system** `(q, r)` for the hexagonal grid. This system represents each hex cell with two coordinates:

- `q`: Represents the column.
- `r`: Represents the diagonal row.

To calculate the distance between two hex cells `(q1, r1)` and `(q2, r2)`, the formula used is:

$$\text{distance} = \max(|q1-q2|, |r1-r2|, |(q1+r1)-(q2+r2)|)$$

3.2 Data Structures

1. **Hex Cells**:
    a. Each cell is represented by a `Hex` class, which includes methods for calculating distance and managing neighbors.
    b. To ensure efficient operations, we implemented `equals` and `hashCode` methods.
2. **Radar Regions**:
    a. Each radar region is stored as a `HashSet<Hex>` to allow efficient computation of intersections using set operations.
3. **Grid Representation**:
    a. The entire grid is conceptually infinite but restricted by the range of the radar responses.

3.3 Input and Output

- **Inputs**:
    - Grid dimensions (rows, columns) for reference.
    - Number of radar responses.
    - For each radar: coordinates `(q, r)` and range.
- **Outputs**:
    - The number of intersecting cells.
    - The coordinates of intersecting cells.

4. Code Summary

The project consists of four main classes:

1. **Hex**: Represents a single hexagonal cell with coordinates `(q, r)`.
    a. Includes distance calculation and neighbor generation methods.
2. **SensorRegion**: Handles the generation of radar response regions and intersection calculations.
3. **HexagonalGridProject (Main)**: The main program that integrates the coordinate system, region generation, and intersection calculations.

5. Example Execution

Inputs:

```
Enter grid dimensions (rows and columns):
10 10
Enter the number of radar responses:
2
Enter radar coordinates (q r) and range:
0 0 2
Enter radar coordinates (q r) and range:
2 1 2
```

Outputs:

```
Number of intersecting cells: 7
Intersecting cells: [(1, 0), (0, 1), (0, 0), (2, -1), (1,
1), (-1, 2), (2, 0)]
```

6. Visualization

A sketch was created to represent the grid, radar towers, their regions, and the intersecting cells. This visual clearly shows:

- The **hexagonal grid layout**.
- **Tower 1 (Blue Region)** at `(0, 0)` with a range of `2`.
- **Tower 2 (Red Region)** at `(2, 1)` with a range of `2`.
- The **intersection region (Green)** showing overlapping cells.

7. Challenges and Solutions

1. **Hexagonal Distance Calculations**:
    a. Challenge: Implementing accurate distance calculations.
    b. Solution: Used the axial coordinate system and the maximum difference formula for simplicity.
2. **Intersection Computation**:
    a. Challenge: Efficiently finding common cells between regions.
    b. Solution: Leveraged `HashSet` operations for fast intersection computation.

8. Tools and Technologies

- **Programming Language**: Java
- **IDE**: Eclipse