

ECON381 FALL 2024

Semester Project

Question 1: What kind of coordinate system can we use to denote the cells in a pointy top hex grid? If there are alternatives, which one provides the easiest method to compute distances, or perform intersections on ranges as depicted above?

Answer: Coordinates in Mathematics

For example, the location of a table in a building plan may be indicated by an (x, y) point on the drawing.

Question 2: Which data structure is better suited to store the entire map?

Answer: Axial Coordinate is the best option to store the entire map.

Question 3: Which data structure is better suited to store a region defined by the sensor reading? Does it matter if the region is a circle or a ring?

Answer: Trees Data Structure is suitable for this matter.

Similar to a graph, a tree is also a collection of vertices and edges. However, in tree data structure, there can only be one edge between two vertices.

The shape of the region (circle or ring) can affect the choice of data structure for storing and querying sensor readings.

Question 4: Implement with Java the coordinate system, the map, and finding the intersection. Your program should get inputs as

- Number of cells in map, or an indicator of its dimensions (ie. rows, columns, etc)
- Number of cells with radar responses
- Coordinates of cells with radar responses (repeats until the number indicated is satisfied) Then your program should output
- The number of cells in the intersection,
- Their coordinates.

Answer:

```
import java.util.ArrayList;
```

```
import java.util.List;
```

```
class Hex {
```

```
    int q; // Axial x-coordinate
```

```
    int r; // Axial y-coordinate
```

```
    public Hex(int q, int r) {
```

```
        this.q = q;
```

```
        this.r = r;
```

```
    }
```

```
    public int distanceTo(Hex other) {
```

```
        return (Math.abs(this.q - other.q)
```

```
            + Math.abs(this.q + this.r - other.q - other.r)
```

```
            + Math.abs(this.r - other.r)) / 2;
```

```
    }
```

```
    @Override
```

```
    public String toString() {
```

```
        return "Hex(" + q + ", " + r + ");"
```

```
    }
```

```
}
```

```
class HexGrid {
```

```
    public List<Hex> getHexesInRadius(Hex center, int radius) {
```

```
        List<Hex> hexes = new ArrayList<>();
```

```

    for (int dq = -radius; dq <= radius; dq++) {
        for (int dr = Math.max(-radius, -dq - radius); dr <= Math.min(radius, -dq +
radius); dr++) {
            int ds = -dq - dr; // Derived from  $q + r + s = 0$ 
            hexes.add(new Hex(center.q + dq, center.r + dr));
        }
    }
    return hexes;
}

```

```

public List<Hex> getHexesInRing(Hex center, int innerRadius, int outerRadius) {
    List<Hex> result = new ArrayList<>();
    for (int radius = innerRadius; radius <= outerRadius; radius++) {
        List<Hex> ringHexes = getHexesInRadius(center, radius);
        for (Hex hex : ringHexes) {
            if (center.distanceTo(hex) == radius) {
                result.add(hex);
            }
        }
    }
    return result;
}
}

```

```

public class ShotSpotterHex {
    public static void main(String[] args) {
        HexGrid grid = new HexGrid();

        Hex sensorCenter = new Hex(3, 3);
    }
}

```

```
int innerRadius = 1;
```

```
int outerRadius = 1;
```

```
List<Hex> detectedHexes = grid.getHexesInRing(sensorCenter, innerRadius,  
outerRadius);
```

```
System.out.println("Hexes detected within the ring (inner radius " + innerRadius +  
", outer radius " + outerRadius + "):");
```

```
for (Hex hex : detectedHexes) {
```

```
    System.out.println(hex);
```

```
}
```

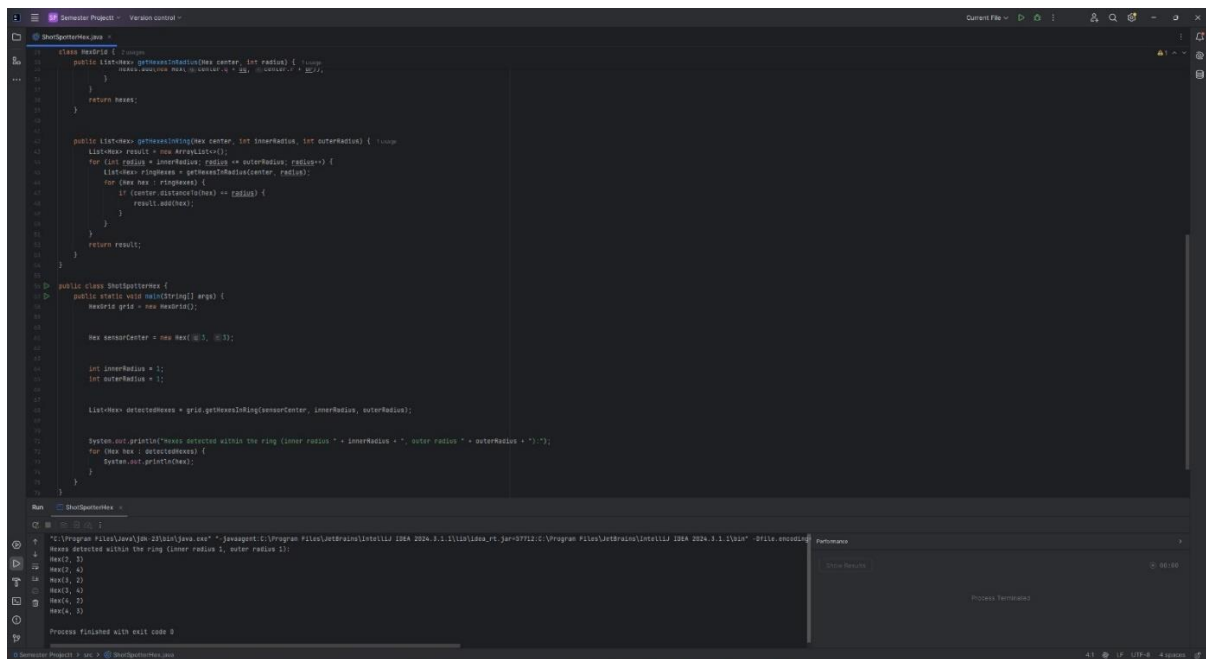
```
}
```

```
}
```

Question 5: Your report should also include a single test case

- A sketch (hand made and photographed is acceptable) which marks the towers and the regions, and the intersection.
- Screen shot of your program working, taking the inputs
- Screen shot of your program working, showing the outputs

Answer:



```
1 class Sensor {
2     public List<Hex> getHexesInRadius(Hex center, int radius) {
3         List<Hex> result = new ArrayList<>();
4         for (int i = 0; i < radius; i++) {
5             for (int j = 0; j < radius; j++) {
6                 Hex hex = new Hex(i, j);
7                 result.add(hex);
8             }
9         }
10        return result;
11    }
12
13    public List<Hex> getHexesInRing(Hex center, int innerRadius, int outerRadius) {
14        List<Hex> result = new ArrayList<>();
15        for (int i = innerRadius; i < outerRadius; i++) {
16            List<Hex> ringHexes = getHexesInRadius(center, i);
17            for (Hex hex : ringHexes) {
18                if (center.contains(hex) == false) {
19                    result.add(hex);
20                }
21            }
22        }
23        return result;
24    }
25
26    public class ShellGame {
27        public static void main(String[] args) {
28            HexGrid grid = new HexGrid();
29
30            Hex sensorCenter = new Hex(0, 0);
31
32            int innerRadius = 1;
33            int outerRadius = 3;
34
35            List<Hex> detectedHexes = grid.getHexesInRing(sensorCenter, innerRadius, outerRadius);
36
37            System.out.println("Hexes detected within the ring (inner radius " + innerRadius + ", outer radius " + outerRadius + ")");
38            for (Hex hex : detectedHexes) {
39                System.out.println(hex);
40            }
41        }
42    }
43}
```

Run

```
1 "C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.1\lib\idea_rt.jar=57712;C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.1\bin" -Dfile.encoding=UTF-8
2 Hexes detected within the ring (inner radius 1, outer radius 3):
3 Hex(1, 0)
4 Hex(1, 1)
5 Hex(1, 2)
6 Hex(2, 0)
7 Hex(2, 1)
8 Hex(2, 2)
9 Hex(3, 0)
10 Hex(3, 1)
11 Hex(3, 2)
12 Process finished with exit code 0
```

Process finished with exit code 0