ESINF REPORT

Group 118

NUNO BARBOSA – 1210822

KEVIN SOUSA-1180853

RODRIGO FERREIRA - 1191447

JOÃO MARTINS- 1201539

Subject: Estruturas de Informação (ESINF)

Teacher: Nuno Filipe Teixeira Malheiro (NFM)

DECEMBER 2021

Content

Introduction

In the following report, we will present our User Story development as well as the respective Domain Model for the whole project.

First, we decided to use an unorganized KD-tree, as told on the US201, but we used a 2D-tree, so k=2 (k represents the dimension of the tree).

Algorithm Analysis

For each User story, we will explain the code written, as well as the difficulties had while developing.

We also determine the time complexity for each method, according to the Big-O notation

US201 – Importing ports and create a 2D-Tree balanced with port locations

```
public static TwoDTree<Port> loadPortsTwoDTree(String fileName) throws FileNotFoundException {
    TwoDTree<Port> doisDCallSign = new TwoDTree<>();
    File file = new File(fileName);

    Scanner in = new Scanner(file);
    String line = in.nextLine();

    while (in.hasNextLine()) {
        String read[] = in.nextLine().trim().split( regex: ",");
        Port port = new Port(read[0],read[1],Integer.parseInt(read[2]),read[3],Double.parseDouble(read[4]),Double.parseDouble(read[5]));
        doisDCallSign.insert(port);
    }
    return doisDCallSign;
}
```

We imported the list in one different method according to following User Stories where each one of them is useful. This method has a O(log(n)) complexity because we are importing data to an 2D-tree structure.

US102 – find the closest port of a ship given its CallSign, on a certain Date Time

```
    public PortController() throws FileNotFoundException {
    }

    public Port findClosestPortToShip(String callSign, Date year, LocalTime hours) throws FileNotFoundException, ParseException {
        Ship ship = new Ship();
        for(Ship ships : shipMaptest.keySet()) {
            if (ships.getCallSign().equals(callSign)) {
                ship = ships;
            }
        }
        List<ShipDynamicFields> shipDynamicFields = shipMaptest.get(ship);
        ShipDynamicFields finalShip =ShipDynamicFields.checkLastPositionUpgrade(year, hours, shipDynamicFields);
        Node portNode = portstest.findNearestNeighbour(finalShip.getLat(), finalShip.getLon());
        return portNode.getInfo();
    }
}
```

```
public Node findNearestNeighbour(double x, double y) {
    return findNearestNeighbour(root, x, y,root,  divX: true);
}

private Node findNearestNeighbour(Node fromNode, final double x, final double y,Node closestNode, boolean divX) {
    if (fromNode == null) {
        return null;
    }
    double d = Point2D.distanceSq(fromNode.getInfo().getLat(), fromNode.getInfo().getLon(), x, y);
    double closestDist =  Point2D.distanceSq(closestNode.getInfo().getLat(), closestNode.getInfo().getLon(), x, y);
    if (closestDist > d) {
        closestNode.setObject(fromNode);
    }
            double delta = divX ? x - fromNode.getInfo().getLat() : y - fromNode.getInfo().getLon();
            double delta2 = delta * delta;
            Node node1 = delta < 0 ? fromNode.left : fromNode.right;
            Node node2 = delta < 0 ? fromNode.right : fromNode.left;
            findNearestNeighbour(node1,x,y,closestNode,!divX);
            if (delta2 < closestDist) {
                findNearestNeighbour(node2,x,y,closestNode, !divX);
            }
            return closestNode;
```

For this user story, we ask the user the CallSign and the specific date. With this inputs we have to find closest port to the ship . For that we have the function “findShipByCallSign” and we call a method to see the last position of a ship.

In the end, we call a method to find the nearest port of a ship.

The function of find the nearest port of a ship use a 0(log(n)) complexity. And the findClosestPortToShip used a O(n^2).

Domain Model Diagram