Programming Paradigms – Haskell Homework

Elisa Marengo Pietro Galliani

1st Semester 2020/21

Download the file `spatial.hs` and solve the exercises in the same file.

In the file two types are declared: a type `Point` as a tuple of two doubles and a type `Rectangle` as a tuple of two points (p_1, p_2). Point p_1 indicates the bottom left coordinate defining the rectangle while p_2 indicates the upper right coordinate defining the rectangle. Also `myDatapoints` is a variable assigned with a list of points which contains all datapoints in the table below. You can use this set of points for testing. You can also modify it (adding, removing, changing points) for testing purposes.

id	x	y
1	2.3	5.4
2	3.4	4.8
3	6.3	9.4
4	7.1	5.4
5	1.1	8.5
6	8.7	3.3
7	9.3	2.3
8	4.6	5.8
9	7.6	4.9
10	2.4	2.8
11	3.9	1.1
12	8.2	2.3
13	4.4	7.2
14	5.5	2.3
15	9.1	9.8
16	9.6	7.1

Table 1: Sample datapoints

This assignment requires you to implement the following functions. For each of them also declare the types as we saw in the course. Types declaration will be part of the evaluation.

1. Write a function `boundingbox` which takes a list of points and a rectangle, and returns all the datapoints contained inside the given rectangle. To test the function use the sample dataset and define some rectangles which should contain different set of points. For instance, you can test the function as follows (anyway, try with different rectangles and check the result).

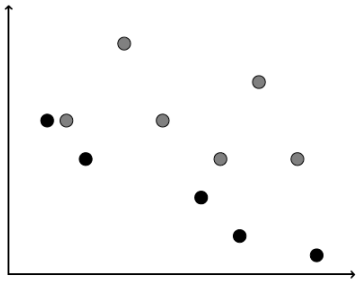

Figure 1: In black the non-dominated points

```
> boundingbox myDatapoints ((1.1,1.1),(8.1,8.1))
> ...
```

2. The Manhattan distance between two points (x_1, y_1) and (x_2, y_2) is defined as $|x_1 - x_2| + |y_1 - y_2|$. Write a function `mindist` which given a point P and a list of points returns the minimal Manhattan distance between P and any point in the list. For example,

   ```
   > mindist (1.0,1.1) [(0,0.1),(2,1.9),(1.1,2.1)]
   1.1
   ```

3. Write a function `nearestneighbors` which returns the `k` nearest neighbors (with respect to the Manhattan distance) for a given point (x, y) in a given list of points. The number of neighbors `k`, the point (x, y) and the list of points are given as input. For example,

   ```
   > nearestneighbors (1.0,1.1) 2 myDatapoints
   [(2.4,2.8),(3.9,1.1)]
   ```

   If `k` is greater than the number of elements in the list, then return all elements of the list.

4. A point $(x1, y1)$ dominates another point $(x2, y2)$ if it both holds that $x1 < x2$ and $y1 < y2$. Therefore, given a point (x_p, y_p) and a list, we can say that the point is *non-dominated* if there does not exist a point (x, y) in the list such that the condition $x < x_p$ and $y < y_p$ is true. Figure 1 represents some points and highlights in black those that are non-dominated.

   Write a function `nondominated` that computes the subset of all non-dominated points of a given set of points. Implementing the solution, you can assume that each point in the list appears only once (i.e., you do not have repeated points in the list).

Deadline: Thu, 10.12.20, 07:00 am (on OLE)