Skyline of R-tree

Yuang Liu (U99473611) Jun Xiao (U85900288)

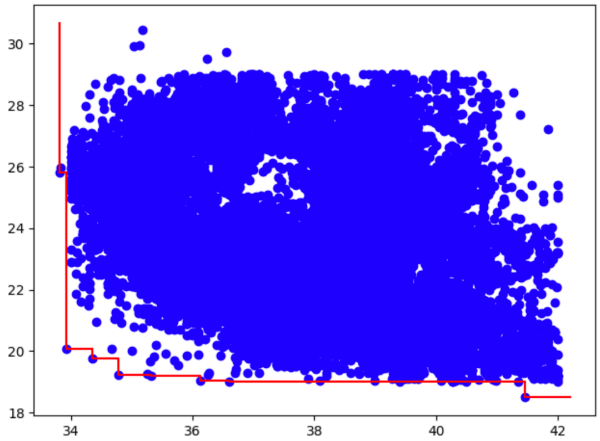
Dependency:

- This project is based on RTree2 (*https://github.com/davidmoten/rtree2*)
- Please download the RTree2 and make these directories YuangLiu.JunXiao.CS562Project under this path: rtree2/main/java/. Then put the Skyline.java and Start.java files under the 'YuangLiu.JunXiao.CS562Project' directory. And put the dataset1.txt file the same level with the rtree2.

How to run:

- The Start.java is the entrance of the program. Running this program will automatically calculate and output the skyline. The skyline of dataset1 has been visualized by python.
- The general work flow of the program is shown as below:

```
The skyline is:
33.82 25.82
33.92 20.08
34.36 19.77
34.78 19.24
35.32 19.2
36.13 19.05
36.6 19.0
41.47 18.52
```




```
Input 1 for add point, 2 for delete point,
3 for quit: 1
Input the x value and y value for the point
(separate by whitespace): 34.36 19.7
Insert: (34.36, 19.7) into rtree.
Insert: (34.36, 19.7) into skyline.
Delete these point(s) from skyline:
(34.36, 19.77)
The skyline is:
33.82 25.82
33.92 20.08
34.36 19.7
34.78 19.24
35.32 19.2
36.13 19.05
36.6 19.0
41.47 18.52

Input 1 for add point, 2 for delete point,
3 for quit: 2
```

```
Input the x value and y value for the point
(separate by whitespace): 5 5
No such a point in rtree!
Input 1 for add point, 2 for delete point,
3 for quit: 2
Input the x value and y value for the point
(separate by whitespace): 34.36 19.7
Delete: (34.36, 19.7) from rtree.
Delete: (34.36 19.7) from skyline.
The skyline is:
33.82 25.82
33.92 20.08
34.36 19.77
34.78 19.24
35.32 19.2
36.13 19.05
36.6 19.0
41.47 18.52

Input 1 for add point, 2 for delete point,
3 for quit: 3
```

Details of implementation:

- Generate skyline:
  We implemented the algorithm from the paper using the function `skylineAlg()`. The heap mentioned in the paper is implemented by `PriorityQueue<HasGeometry>`, where `HasGeometry` contains information of all nodes. The `PriorityQueue` in Java is

a heap, where we only need to provide a comparator. The comparator sorts the queue by the sum of x and y values of bottom-left corner of the MBR or sum of x and y of a point.

- Dominate:

  We check whether a point or an MBR is dominated by current skyline with the function `notDominate(cur)`, which will return False if not dominated and return True otherwise. We use following conditions for the judgement:

```
(n.geometry().mbr().x1() > e.geometry().x() &&
n.geometry().mbr().y1() >= e.geometry().y()) ||
(n.geometry().mbr().x1() >= e.geometry().x() &&
n.geometry().mbr().y1() > e.geometry().y())
```

  Where n is the point and e be any point from the skyline.

- Insertion:

  We first added the point (x, y) into the r-tree. If the new point is not dominated by the current skyline, we add it into the skyline then check whether any points in skyline are dominated by the new point. We implement this for every skyline point (x', y') as following: if both x'>x and y'>y, or x'=x but y'>y, or y'=y but x'>x, then the skyline point is dominated by the new point. After knowing which points in current skyline are dominated by the new point, remove those points and finish the insertion.

- Deletion:

  First check whether the deleted point (x, y) is in the r-tree or not. If exits, delete it from the r-tree. If the point is not in current skyline, then done. Otherwise, we call the function `skylineAlg()` based on the updated skyline. The reason why we can do the updates by calling that function is that we only focus on the areas that are not dominated by any point in the skyline, which are the area used to be dominated by the deleted point.