CS562 Project 1 Skyline

Group member:
Ziyang Chen(U22489933) chenzy@bu.edu
Ci Chu(U76373419) chuci@bu.edu

How the program is designed:

We create a new class called Skyline, which takes a rtree as an input argument and store it inside the class. We can call different methods on a Skyline object to calculate skyline for the rtree, add or delete a point to the rtree and update the skyline. Below is how we implement calculate, add and delete method:

public void calculateSL():

This method is used to calculate skyline. It will be automatically called in the constructor of Skyline Object by default. And user can choose to pass a "false" into the constructor when creating the Object so that skyline will not automatically be calculated.

We firstly initial a Heap pq and a ArrayList S. We calculate the minDist of the root, add the pair of (minDist of root, root) to the pq where pq is order by the first element of the pair. Then in a while loop, we pop out the pair on the heap top and check second element's data type. If the second element is a Leaf object or a NonLeaf object, we will compare it with all the entries in the S, if it is not dominated by any entries in the S, we add it to the pq; If the second element is a Entry Object, we still do the comparison as above, but this time it will be added to S if it's not dominated. The while loop will terminate when the pq is empty and S is the skyline for current rtree and it will be stored to a class variable *private ArrayList<EntryDefault> SkylineList* for further use.

public void add(Point p):

When we add new points in RTree, the points in skyline might be influenced by this updating action. There will be two cases when adding a new point: one is this point is at the above (top right) the skyline, the other one is it will replace some points in skyline. The algorithm we designed is based on these cases. We will loop all the points in skyline and compare to the new inserting point. If the new point at the top right of the current skyline point, the for loop will stop and nothing will be change in skyline list. Otherwise, if the new point at the bottom left of the skyline point, the new point will replace this current point in skyline.

public void delete(Point p):

Before delete the point p, we need to figure out that if p is in skyline points list. If it is in this list, the skyline will have some changes. Firstly, we need find two points in skyline which are adjacent with point p in left and right. Based on these three points, we can have a limited rectangle and the new skyline points will be chosen in this area. We put all the points at this rectangle area and in a priority queue pq with minDist as the standard. Then loop all the points in pq and use the similar method as calculateSL to find all the new skyline points. Finally, add these new points in the skyline points list and delete the point p.

There are some other methods that could be useful, like: *public RTree getRtree()* will return the rtree that is stored in the skyline object; *public ArrayList<EntryDefault> getSL()* will return the skyline for the rtree.

Test case introduction:

We create a SkylineTest class for test purpose. *public void calculatetest(), public void addTest(), public void deleteTest()* can test calculate skyline, add point and update skyline, delete point and update skyline separately. They can output original skyline and updated skyline to the console. And visualizations of the skyline are also generated under path ./target/ for you to check.

How to run the program:

After downloading the program, you need to run *mvn clean install* to build the project and all the test cases will be printed to the console, including the skyline test case. Firstly, you need to add some data in "*src/test/resources/insertpoint.txt*" and "*src/test/resources/deletepoint.txt*". The data form like the form in "*dataset1.txt*".

Then you can also run *rtree/src/test/java/com/github/davidmoten/rtree/SkylineTest.java* to see only the test result for skyline in your IDE. Finally, you will get the three images in */target* folder named "SkylineTree.png", "DeleteSkylineTree.png" and "AddSkylineTree.png" and three list file "SkylineList.txt", "AddSkylineList.txt" and "DeleteSkylineList.txt". The red points in the images are the points on Skyline and the list file record the skyline points with or without other operations (add and delete).