L7 Report

Name: Priyanshu Garg
Enrollment No.: 18114058
Branch: Computer Science and Engineering
Batch: O3

Problem1:

Given: n 2D points and two orthogonal polygons.
Problem: Find the set of points lie inside the overlapping region (rectangular) of the two given orthogonal
polygons.
Write a program in Java to solve the above problem applying k-d tree data structure.

Data structures:

Program used the concepts of KD-Tree to make a tree, program also used th e arrays, arraylist, array of objects etc.

Algotithm:

Program used th algo as written as in
http://groups.csail.mit.edu/graphics/classes/6.838/S98/meetings/m13/kd.html
If the root node is aligned in planeA, then the left subtree will contain all points whose coordinates in that plane are smaller than that of root node. Similarly, the right subtree will contain all points whose coordinates in that plane are greater-equal to that of root node.
A K-D Tree(also called as K-Dimensional Tree) is a binary search tree where data in each node is a K-Dimensional point in space. In short, it is a space partitioning(details below) data structure for organizing points in a K-Dimensional space.
A non-leaf node in K-D tree divides the space into two parts, called as half-spaces.

Snapshots:

```
priyanshu@Kratos:~/Documents/L7_Priyanshu_Garg_12114058$ java q1
Enter the number of points
10
Enter the x and y coordinates of the points separated by a space
4.3 4.1
5 5.8
5.2 3
4.3 8
6 7.7
7.7 2.2
6.8 4.4
8.1 3.6
7.3 8
7.5 6.6
Enter details for first polygon
Number of sides : 4
Points :
3.5 5.1
6.5 8.4
Enter details for second polygon
Number of sides : 6
Points :
4.1 2.2
6.7 2.2
6.7 4.3
5.4 4.3
5.4 8.7
4.1 8.7
4.3 8.0
5.0 5.8
priyanshu@Kratos:~/Documents/L7_Priyanshu_Garg_12114058$
```

Problem2:

Given n values in an array and two index values, find the result of the following queries
1. minimum value
2. maximum value
3. sum
4. update by adding 4 with each element,
within the given index range using Segment tree. Also implement the brute-force method and compare the

Data structures:

Program used the concepts of segment tree i.e used arrays to store the nodes .

Algorithm:

A Segment Tree can be built usingrecursion (bottom-up approach). Start with the leaves and go up to the root and update the corresponding changes in the nodes that are in the path from leaves to root. Leaves represent a single element. In each step, the data of two children nodes are used to form an internal parent node. Each internal node will represent a union of its children's intervals. Merging may be different for different questions. So, recursion will end up at the root node which will represent the whole array.
For , search the leaf that contains the element to update. This can be done by going to either on the left child or the right child depending on the interval which contains the element. Once the leaf is found, it is updated and again use the bottom-up approach to update the corresponding change in the path from that leaf to the root.

Snapshots:

```
priyanshu@Kratos:~/Documents/L7_Priyanshu_Garg_12114058/Source_Code$ javac SegmentTree.java
priyanshu@Kratos:~/Documents/L7_Priyanshu_Garg_12114058/Source_Code$ java SegmentTree
No. of terms you want to enter
8
32 46 89 14 25 32 45 65
1. minimum value
2. maximum value
3. Sum
4. Update by adding 4 with each element
Enter your option
1
Enter range as left and right (<8)
4 6
14
priyanshu@Kratos:~/Documents/L7_Priyanshu_Garg_12114058/Source_Code$ java SegmentTree
No. of terms you want to enter
8
32 46 89 14 25 32 45 65
1. minimum value
2. maximum value
3. Sum
4. Update by adding 4 with each element
Enter your option
2
Enter range as left and right (<8)
1 7
89
priyanshu@Kratos:~/Documents/L7_Priyanshu_Garg_12114058/Source_Code$ java SegmentTree
No. of terms you want to enter
8
32 46 89 14 25 32 45 65
1. minimum value
2. maximum value
3. Sum
4. Update by adding 4 with each element
Enter your option
3
Enter range as left and right (<8)
2 6
206
priyanshu@Kratos:~/Documents/L7_Priyanshu_Garg_12114058/Source_Code$
```