CSN-261 ASSIGNMENT-7

- Name-Kavya Barnwal
- Enrollment No.-18114039
- Course-B.Tech-CSE
- Email: kbarnwal@cs.iitr.ac.in
- Contact:6205125517

===

Github Link

Problem Statement 1:

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.

Point representation:

p i x i , y i)

Line representation:

(x 2 , y 2)

(x 1 , y 1)

Polygon representation:

Sides = 4

<bottom-left, top-right>

{ (x , y) , (x , y) }

Constraints while building the k-d tree:

Sides > 4

anti-clock wise order, starting from bottom-left

{ (x , y), (x , y), ... , (x , y) }

=====================================

Data Structure And Algorithm Used :

Data Structures Used :

K-d Tree ,Linked List, Arrays.

Algorithms Used :

Since there are many possible ways to choose axis-aligned splitting planes, there are many different ways to construct k-d trees. The canonical method of k-d tree construction has the following constraints:

As one moves down the tree, one cycles through the axes used to select the splitting planes. (For example, in a 3-dimensional tree, the root would have an x-aligned plane, the root's children would both have y-aligned planes, the root's grandchildren would all have z-aligned planes, the root's great-grandchildren would all have x-aligned planes, the root's great-great-grandchildren would all have y-aligned planes, and so on.)

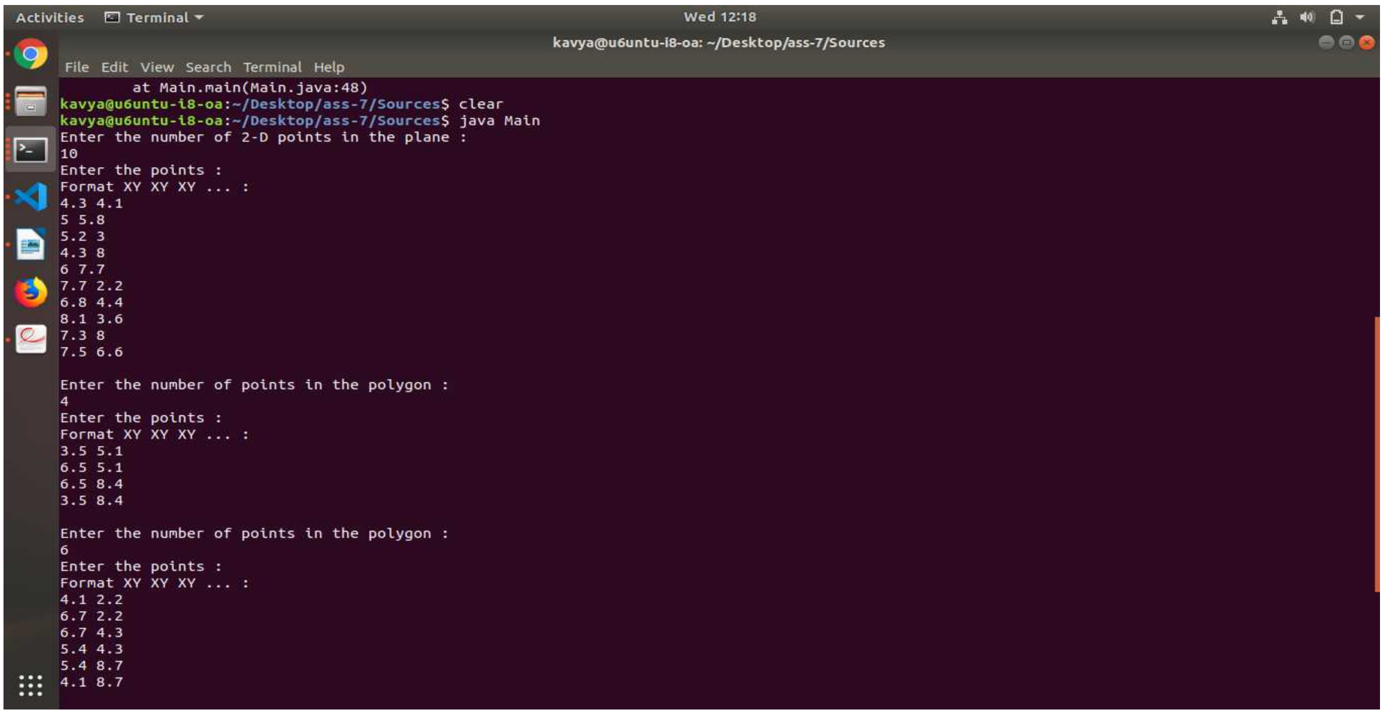
Points are inserted by selecting the median of the points being put into the subtree, with respect to their coordinates in the axis being used to create the splitting plane. (Note the assumption that we feed the entire set of n points into the algorithm up-front.)

This method leads to a balanced k-d tree, in which each leaf node is approximately the same distance from the root. However, balanced trees are not necessarily optimal for all applications.

Note that it is not required to select the median point. In the case where median points are not selected, there is no guarantee that the tree will be balanced. To avoid coding a complex O(n) median-finding algorithm[3][4] or using an O(n log n) sort such as heapsort or mergesort to sort all n points, a popular practice is to sort a fixed number of randomly selected points, and use the median of those points to serve as the splitting plane. In practice, this technique often results in nicely balanced trees.

Given a list of n points, the following algorithm uses a median-finding sort to construct a balanced k-d tree containing those points.

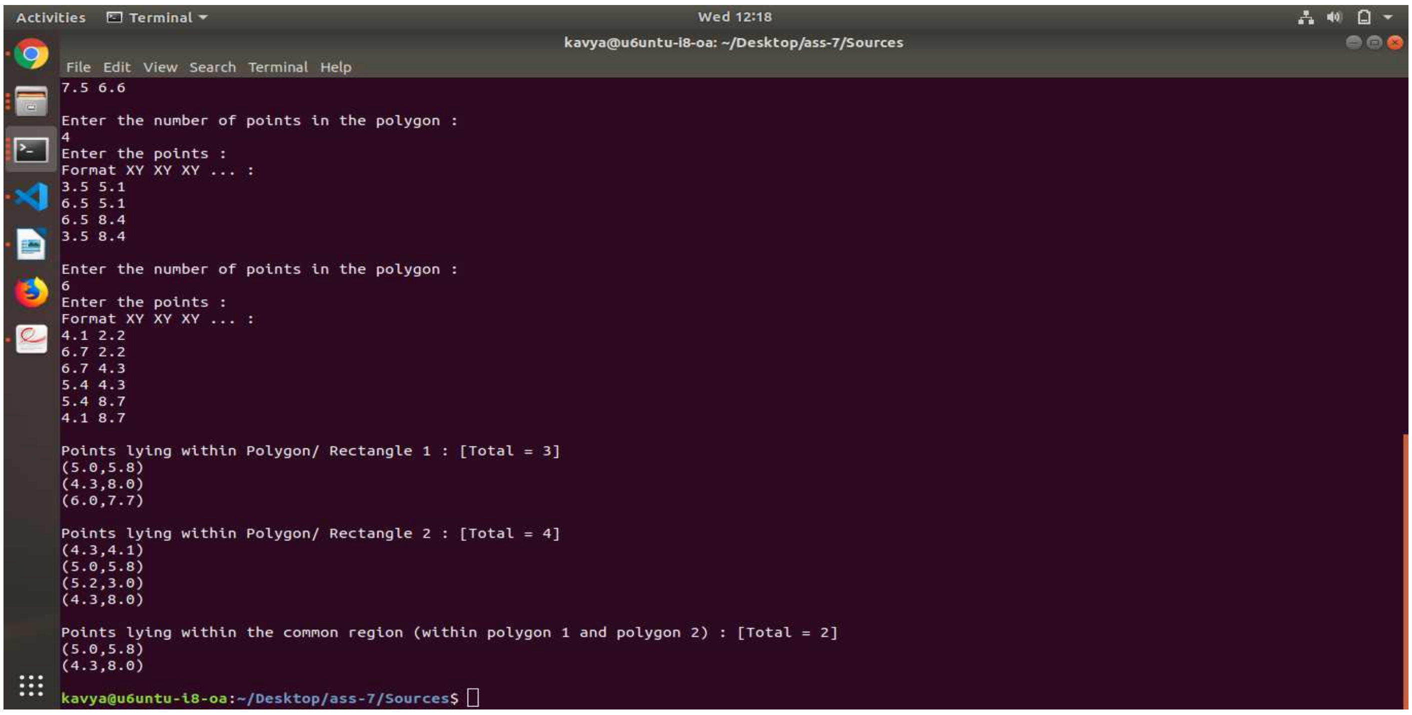
Screenshot of compilation:


Activities
Terminal
Wed 12:18
kavya@u6untu-i8-oa: ~/Desktop/ass-7/Sources
File Edit View Search Terminal Help

```
        at Main.main(Main.java:48)
kavya@u6untu-i8-oa:~/Desktop/ass-7/Sources$ clear
kavya@u6untu-i8-oa:~/Desktop/ass-7/Sources$ java Main
Enter the number of 2-D points in the plane :
10
Enter the points :
Format XY XY XY ... :
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 the number of points in the polygon :
4
Enter the points :
Format XY XY XY ... :
3.5 5.1
6.5 5.1
6.5 8.4
3.5 8.4

Enter the number of points in the polygon :
6
Enter the points :
Format XY XY XY ... :
4.1 2.2
6.7 2.2
6.7 4.3
5.4 4.3
5.4 8.7
4.1 8.7
```


Activities
Terminal
Wed 12:18
kavya@u6untu-i8-oa: ~/Desktop/ass-7/Sources
File Edit View Search Terminal Help

```
7.5 6.6

Enter the number of points in the polygon :
4
Enter the points :
Format XY XY XY ... :
3.5 5.1
6.5 5.1
6.5 8.4
3.5 8.4

Enter the number of points in the polygon :
6
Enter the points :
Format XY XY XY ... :
4.1 2.2
6.7 2.2
6.7 4.3
5.4 4.3
5.4 8.7
4.1 8.7

Points lying within Polygon/ Rectangle 1 : [Total = 3]
(5.0,5.8)
(4.3,8.0)
(6.0,7.7)

Points lying within Polygon/ Rectangle 2 : [Total = 4]
(4.3,4.1)
(5.0,5.8)
(5.2,3.0)
(4.3,8.0)

Points lying within the common region (within polygon 1 and polygon 2) : [Total = 2]
(5.0,5.8)
(4.3,8.0)

kavya@u6untu-i8-oa:~/Desktop/ass-7/Sources$
```

==

Problem Statement 2:

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

execution time of both the methods.

A segment tree is a data structure used for storing information about intervals, range or segments. It

facilitates efficient range querying in O(log n), where n is the size of the given problem.

Example:

Input:

[2, 5, 1, 4, 9, 3]

Index 1 = 3

Index 2 = 5

Output: 3

==

Data Structure And Algorithm Used :

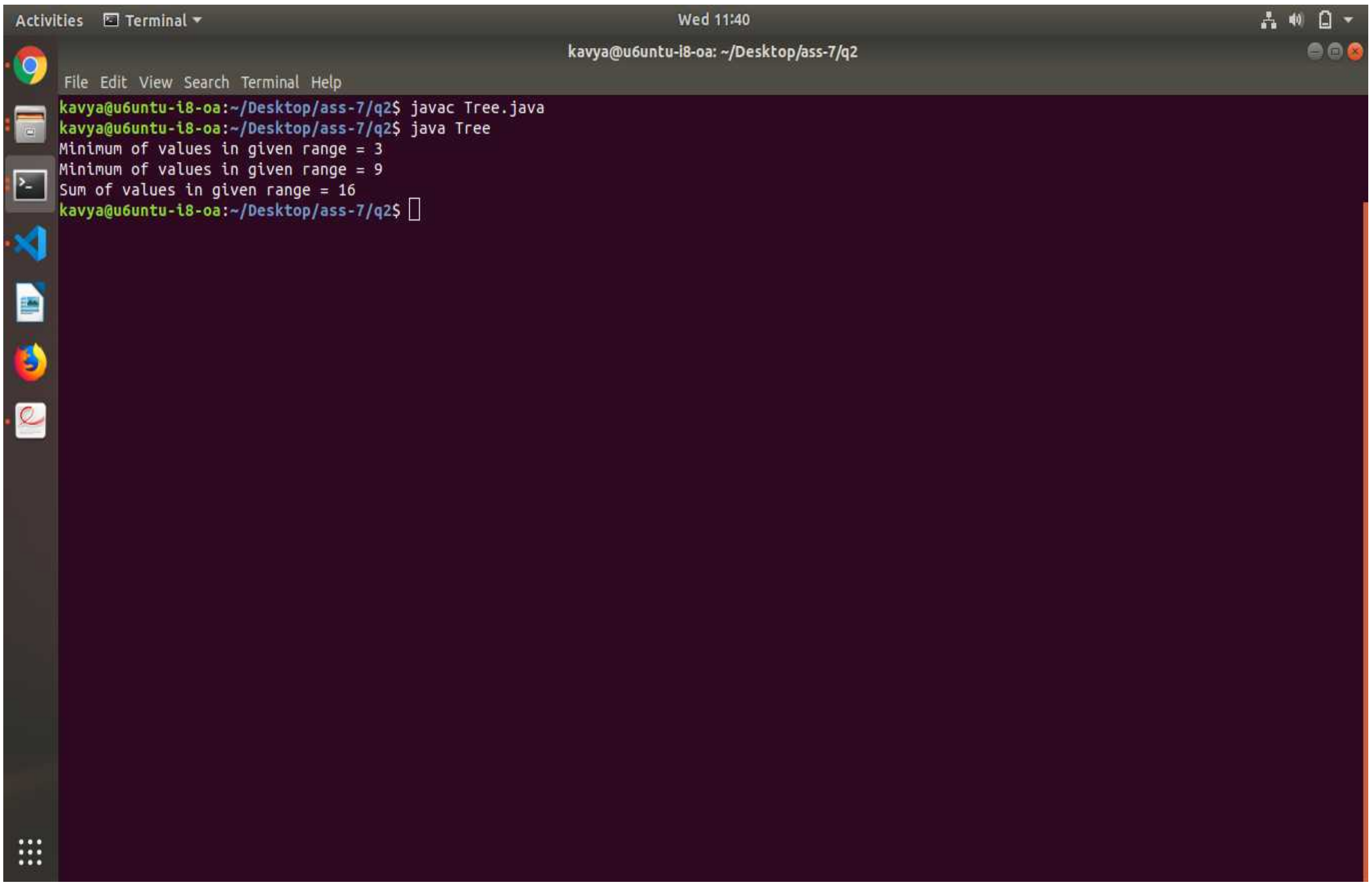
Data Structures Used :

Segment Tree, Linked List, arrays.

Algorithms Used :

The segment tree is also represented as an array. It is not a complete binary tree. It is rather a full binary tree and all levels are filled except possibly the last level.The last level may have gaps between nodes. The dummy values are never accessed and have no use. This is some wastage of space due to simple array representation.Start with a segment arr[0 . . . n-1] and every time divide the current segment into two halves,if it has not yet become a segment of length 1, and then call the same procedure on both halves, and for each such segment, we store the sum or minimum or maximum in the corresponding node.All levels of the constructed segment tree will be completely filled except the last level. Also, the tree will be a full binary tree because we always divide segments in two halves at every level. The update can also be done recursively. Given an index which needs to be updated. Let diff be the value to be added. Start from the root of the segment tree and add diff to all nodes which have given index in their range. If a node doesn't have a given index in its range, don't make any changes to that node.

Screenshot of compilation:


Activities
Terminal
Wed 11:40
kavya@u6untu-i8-oa: ~/Desktop/ass-7/q2
File Edit View Search Terminal Help

```
kavya@u6untu-i8-oa:~/Desktop/ass-7/q2$ javac Tree.java
kavya@u6untu-i8-oa:~/Desktop/ass-7/q2$ java Tree
Minimum of values in given range = 3
Minimum of values in given range = 9
Sum of values in given range = 16
kavya@u6untu-i8-oa:~/Desktop/ass-7/q2$
```

===