Indian Institute of Technology Roorkee
Department of Computer Science and Engineering

CSN-261: Data Structures Laboratory
(Autumn 2019-2020)
Lab Assignment-7 (L7)

Name :- Ayush Mangal **Enroll no:- 18114016**
Branch :- CSE **Sub Batch :- O1**

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.

Data Structures used :

- K-d tree
- Array

Algorithms used :

- We are trying to find the number of points in the overlapping region of two orthogonal polygons
- We compare the x coordinates and the y coordinates with the minimum and maximum x coordinates of the overlapping region
- We have used K dimensional tree (KD tree)
- 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.
- 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.

Screenshots:

```
^Cayushtues@Kamikaze:~/L7$ javac problem1.java
ayushtues@Kamikaze:~/L7$ java problem1
Enter the number of points
10
Enter the x and y coordinates of the dot 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
ayushtues@Kamikaze:~/L7$
```

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.

Data Structures used :

- Array
- Segment Tree

Algorithm:

- Segment tree is a binarry tree that stores intervals
- The root stores the entire interval
- The left child stores the left half
- The right child stores the right half
- We can use this to effectively perform operations on ranges
- To find sum of range , find the appropriate range in the segment tree ,and return value of node , where value of node stores the sum of the corresponding interval
- If the range is partially in the node , traverse its left child and right child to find the sum of those partial parts and add them
- To find the maximum , or minimum , its same as sum , just store the max/min of the interval in the nodes , instead of the sum
- To update the range , use lazy propagation algorithm

- If current segment tree node has any pending update, then first add that pending update to current node.
- If the interval represented by current node lies completely in the interval to update, then update the current node and update the lazy[] array for children nodes

- If the interval represented by current node overlaps with the interval to update, then update the nodes as the earlier update function

ScreenShots :

```
ayushtues@Kamikaze:~/L7$ java problem2
Enter the size of the array
6
Enter the elements of the array
1 2 3 4 5 6
Enter 1 for segment tree , and 0 for brute force approach
1
Enter one of the following options (1,2,3,4,5)
1 -> Find Sum in range (Enter range i,j)
2-> Update by adding i in range j to k , enter i , j ,k
3-> Find the maximum in range  (Enter range i,j)
4-> Find the minimum in range  (Enter range i,j)
5-> Terminate
1
Enter initial index
1
Enter final index
3
The answer is
9
The execution time in milli seconds is 6108
```

```
ayushtues@Kamikaze:~/L7$ java problem2
Enter the size of the array
6
Enter the elements of the array
1 2 3 4 5 6
Enter 1 for segment tree , and 0 for brute force approach
0
Enter one of the following options (1,2,3,4,5)
1 -> Find Sum in range (Enter range i,j)
2-> Update by adding i in range j to k , enter i , j ,k
3-> Find the maximum in range  (Enter range i,j)
4-> Find the minimum in range  (Enter range i,j)
5-> Terminate
1
Enter initial index
1
Enter final index
3
The answer is
9
The execution time in milli seconds is 28819
```

```
ayushtues@Kamikaze:~/L7$ java problem2
Enter the size of the array
6
Enter the elements of the array
1 2 3 4 5 6
Enter 1 for segment tree , and 0 for brute force approach
1
Enter one of the following options (1,2,3,4,5)
1 -> Find Sum in range (Enter range i,j)
2-> Update by adding i in range j to k , enter i , j ,k
3-> Find the maximum in range  (Enter range i,j)
4-> Find the minimum in range  (Enter range i,j)
5-> Terminate
3
Enter initial index
1
Enter final index
5
The answer is
6
The execution time in milli seconds is 4325

^Cayushtues@Kamikaze:~/L7$ java problem2
Enter the size of the array
6
Enter the elements of the array
1 2 3 4 5 6
Enter 1 for segment tree , and 0 for brute force approach
0
Enter one of the following options (1,2,3,4,5)
1 -> Find Sum in range (Enter range i,j)
2-> Update by adding i in range j to k , enter i , j ,k
3-> Find the maximum in range  (Enter range i,j)
4-> Find the minimum in range  (Enter range i,j)
5-> Terminate
3
Enter initial index
1
Enter final index
5
The answer is
6
The execution time in milli seconds is 3591
```

```
^[^Cayushtues@Kamikaze:~/L7$ java problem2
Enter the size of the array
6
Enter the elements of the array
1 2 3 4 5 6
Enter 1 for segment tree , and 0 for brute force approach
1
Enter one of the following options (1,2,3,4,5)
1 -> Find Sum in range (Enter range i,j)
2-> Update by adding i in range j to k , enter i , j ,k
3-> Find the maximum in range  (Enter range i,j)
4-> Find the minimum in range  (Enter range i,j)
5-> Terminate
4
Enter initial index
1
Enter final index
3
The answer is
2
The execution time in milli seconds is 3060
```

```
^Cayushtues@Kamikaze:~/L7$ java problem2
Enter the size of the array
6
Enter the elements of the array
1 2 3 4 5 6
Enter 1 for segment tree , and 0 for brute force approach
1
Enter one of the following options (1,2,3,4,5)
1 -> Find Sum in range (Enter range i,j)
2-> Update by adding i in range j to k , enter i , j ,k
3-> Find the maximum in range  (Enter range i,j)
4-> Find the minimum in range  (Enter range i,j)
5-> Terminate
2
Enter value to be added
1
Enter initial index
1
Enter final index
 2
Successfully updated
The execution time in milli seconds is 4390
```

```
^Cayushtues@Kamikaze:~/L7$ java problem2
Enter the size of the array
6
Enter the elements of the array
1 2 3 4 5 6
Enter 1 for segment tree , and 0 for brute force approach
0
Enter one of the following options (1,2,3,4,5)
1 -> Find Sum in range (Enter range i,j)
2-> Update by adding i in range j to k , enter i , j ,k
3-> Find the maximum in range  (Enter range i,j)
4-> Find the minimum in range  (Enter range i,j)
5-> Terminate
2
Enter value to be added
1
Enter initial index
1
Enter final index
 5
Successfully updated
The execution time in milli seconds is 5616
```