

CSE 204

Data Structure & Algorithm I Sessional

A Report on Finding The Second Closest Point

Submitted By

Name: Md. Nazmul Islam Ananto

ID: 1805093

Date of Submission: 2021-06-18

Complexity Analysis:

If we look closer into the method "*findClosestPair*" it takes two arrays of size n who are sorted w.r.t. respectively x coordinate and y coordinate. And then it calls the method that is implemented with divide and conquer approach.

The complexity of sorting is $O(n \log n)$.

Let the method "*findClosestPair*" requires $T(n)$ time. And the combine portion requires $T'(n)$ time. $T(n)$ is the total complexity of my program.

So, the total running time is $T'(n) = T(n) + O(n \log n)$

Complexity of "*divideAndFindSecondClosest*" method:

Base case: If $n \leq 3$ then we consider it to be the base case. In the base condition, at max 3 sorted points are compared with each other. So, it's time complexity is $O(1)$.

Divide Step: In this step, the set of arrays are divided into two sets of arrays along the middle position of x axis. Middle point is calculated in $O(1)$. And the set of sorted points along y axis are divided through a loop which takes $O(n)$ time.

Conquer Step: Here, the method *findClosestPair* is called twice recursively. So, it will cost $2T(n/2)$ time.

Combine Step: When we try to combine the two sets from where we have already found the two pairs of points with closest distance, it is seen that we haven't considered any pair of points where one is from the left set and the other is from the right set. As the second min distance can only be possible between the two subsets at a distance smaller than or equal to the present second closest distance on either side of the vertical division line. So, only those points that are inside that range are taken. An array of the points that are inside that range are taken from the presorted array. So, the time complexity is $O(n)$. Finally, the two pairs of points with closest distance in the region is found and compared with the previously found closest points. This takes $O(n)$ time because, the dataset is already sorted.

So, the total complexity of time becomes

$$T(n) = 2T(n/2) + O(1) + O(1) + O(n) + O(n)$$

$$\Rightarrow T(n) = 2T(n/2) + O(n)$$

Therefore,

$$T(n) = O(1) \quad ; n \leq 5$$

$$T(n) = 2T(n/2) + O(n); \text{ otherwise}$$

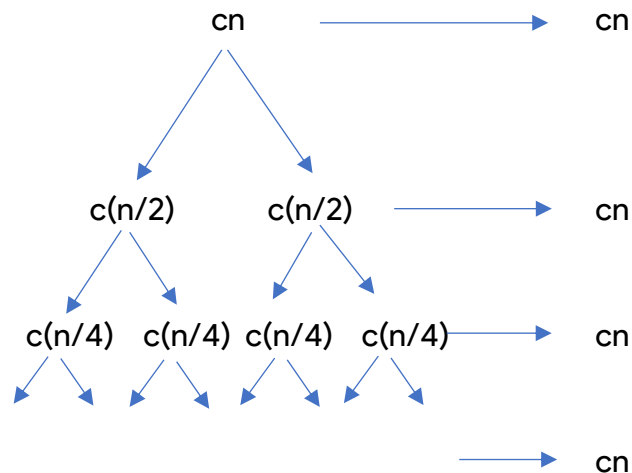
According to master theorem,

Here, $a = 2$, $b = 2$ and $f(n) = O(n)$

And $n^{\log_b a} = n^{\log_2 2} = n^1 = n$

So, we can guess the complexity to be $O(n \log n)$

Proving the complexity $O(n \log n)$ with recursion tree:



Here, the height of the tree is $\log_2 n$

So, total number of levels is $\log_2 n + 1$

Cost of every level is $= cn$

Therefore, total cost is $cn(\log_2 n + 1) = cn \log_2 n + cn$

So the complexity is $O(n \log n)$

Therefore, $T(n) = O(n \log n)$

So, $T'(n) = O(n \log n) + O(n \log n) = O(n \log n)$

Hence, total time complexity of the algorithm is **$O(n \log n)$**