COMPLEXITY ANALYSIS OF THE ALGORITHM

ON FINDING SECOND CLOSEST POINT

Name:Swarnali Saha
ID:1805104
Course No: CSE 204
Course title: Data structure and Algorithm
CSE'18-B, L-2/T-1

Divide and conquer:

A divide and conquer algorithm is a strategy of solving a large problem by breaking the problem into smaller sub-problems using recursive calls, solving the sub-problems, and combining them to get the desired output.

Problem specifications:

Sherlock told Mycraft to choose two houses in Manhattan which are in second minimum distance.
So, he had to find the houses by sorting co-ordinates of the houses and determining the distances of the houses. There are n houses in Manhattan. Each house h_i has a two-dimensional position (x_i,). The distance between two houses (h_i,h_j) is the Euclidean distance of (x_i,y_i) and (x_j,y_j).
Mycraft did this in $O(n^2)$ approach. To reducing the time complexity, we have to do this in O(nlogn) approach.

Algorithm used in this problem:

Here, we used divide and conquer approach to find the appropriate house. Firstly, we take a input as array of house object where we stored co-ordinates of the houses. Then, we sorted it in ascending order to the x-axis. Then we sort the array again according to the y-axis and store it in another array. Then, we pass this pre-sorted array to a recursive function to determine the second closest distance of the houses.
In the recursive function, we actually divide the array into right and left sub array where left array contains the houses which are at left side of the mid point and right array contains the houses which are at right side of the mid point.
Now we have to create an array strip[] that stores all points which are at most second closest distance away from the middle line dividing the two sets. Then, we have to find the 2^{nd} minimum distance from the strip and return it.

Complexity Analysis:

Let's assume the time complexity of the program is T(n). In the recursive function we divide the array in to two sub-arrays and call the function two times recursively. So the time complexity is 2T(n/2). Then we store the value of y array into two sub arrays. As it is a sorted array, it take O(n) times.
We also find the strips in O(n) time. All points in strip[] are sorted according to y co-ordinate. They all have an upper bound on minimum distance as second smallest distance from left and right halves. It's a O(n) method as the inner loop runs much less than n times. For this, it has complexity of O(n) instead of $O(n^2)$.
Then, we find the second minimum distance. Which also takes O(n) times.

So the T(n) is-
T(n) = 2T(n/2) + O(n) + O(n) + O(n)
= 2T(n/2) + O(n)

Now if we use Master's theorem,
we get a=2, b=2, f(n)= O(n)
$n \log_b a = n \log_2 2 = n * 1 = n$

T(n) = O(nLogn)

So total time complexity is **O(nlogn).**