`

**Name: Iftekhar E Mahbub Zeeon**

**ID: 1805038**

**Course: CSE204**

**Offline on Divide and Conquer**

# Complexity Analysis:

To find the second closest pair of houses from a set of houses, we had to also track down the closet pair of houses. That's why in the code we used a class ***MinimumPair*** that has both fields closest distance and second closest distance.

The ***secondClosestHouse*** method in **Algorithm.java** file returns the MinimumPair object. This method uses the divide and conquer approach. The method receives four arguments as parameters; two arrays px and py which are x axis and y axis wise sorted respectively, starting index and ending index(inclusive) For sorting, we used merge sort whose complexity is **O(nlogn).**

## Base Part:

At the beginning of the method, from line 7 to 9, there is base part where we check the size of the divided array is below 3 or not. In this part we calculate the minimum distances between 3 or below houses, so time complexity of this part is **O(1)**.

`

## Divide Part:

Recursive method is used for divide part where we are dividing the array from it's mid point two times in line 13 and 14. So the complexity of this part is **2T(n/2).**

## Combine Part:

This part is an external combine part. Only combining the left side and right side would not give us the correct result as there might be an area where left and right might overlap. So we build a middle area array from y axis wise sorted array whose complexity is **O(n)**. And after that, we compare the middle area distances with the left and right side distances. Here after line 67, two for loops are iterating. It might seem the complexity is O(n^2) but it is not. The second for loop is bound to iterate 7 times or less. So ultimately two for loops iterate for maximum cn times where c is the constant. So the complexity here is **O(n).**

So we find the complexity T(n) = 2T(n/2) + O(n).

Using the master theorem, we find the solution **T(n) = O(nlogn).**