BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Department of Computer Science and Engineering

Course No. : CSE 204

Course Title: Data Structures and Algorithms I Sessional

Report on Divide and Conquer

SUBMITTED BY

Name: Ayesha Binte Mostofa

Student ID: 1805062

Department: CSE

Section: B1

In the given problem we are asked to find the second closest pair of houses in a city using divide and conquer algorithm. It is a condition to achieve time complexity of O(n lg n).

Complexity Analysis:

For the problem we first stored the coordinates of the cities in an array. Then we created an array where the cities are arranged by ascending order of x-coordinate of the city. And another array was created where the cities were arranged by ascending order of y-coordinate. This step is necessary to achieve the required time complexity. For the recursive calls we have selected the number of cities in the subarray to as a parameter of the base condition. When the number of cities is less than or equal to 3, we determine the closest and second closest point among them. Otherwise, we continue to divide the array. Like any other divide and conquer algorithm this solution also has three basic steps.

- **Divide**

In this step we divide the existing array into two subarrays by drawing a vertical line. We get two

subarrays two subarrays X(upto mid) and middleX where the houses are sorted by increasing x-coordinate and two subarrays RightY and LeftY where the houses are sorted by increasing y-coordinate. We continue to divide in this way until there are only three houses in the array. Then we run a brute force approach to find the closest and second closest pair of houses.

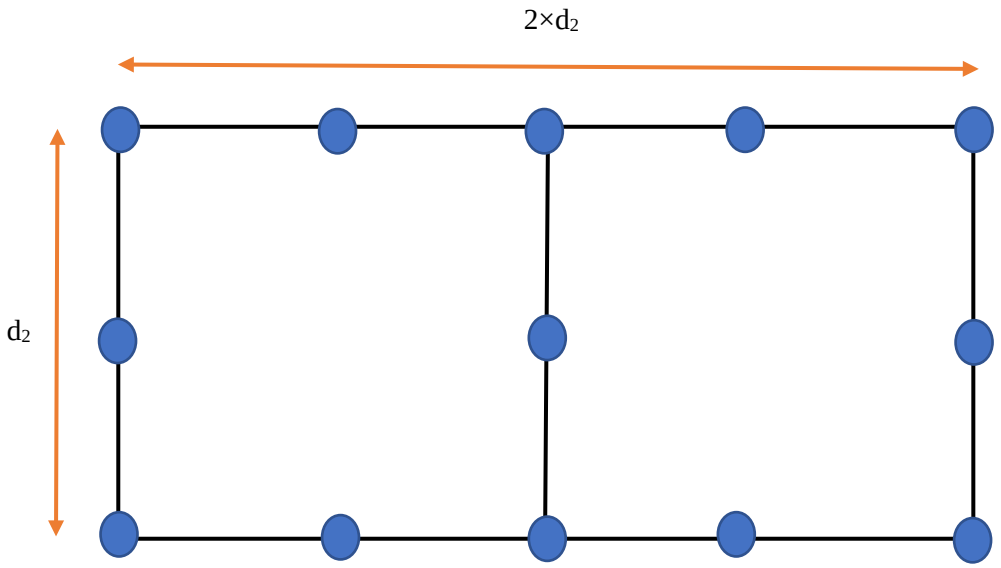
- **Conquer**

After dividing the array into two subarrays we make two recursive call that finds the closest and second closest pair of houses from each of the two subarrays. Now suppose the smallest among these two results is d. Now we compare this distance d with the distance of closest pair of houses. If d is smaller than this distance, the current closest pair becomes second closest pair and we get a new closest pair of cities. Otherwise, If the distance d is smaller than the distance of the second closest pair of houses, we get a new second closest pair of houses. If none of these two conditions is true, we need not make any update to existing closest and second closest pair.

- **Combine**

Now after conquer step we have found the closest and second closest pair of houses from the left and right subarray. But we have not taken the pair that has one house from left array and one from right sub array under consideration. Suppose before this step the distance between the second

closest pair of point is nearestSecondDistance and the distance of the closest pair is nearestDistance. Now we will compare the houses that are at most nearestSecondDistance distance away from the house in both the subarrays that splits the array into two halves. It is obvious that the horizontal and vertical distance between such houses will be less than nearestSecondDistance. We take an array that contains the houses situated inside the strip. This array needs to be sorted by the y-coordinate. But we need not sort this array manually. We can generate an already sorted array from the previously array that was sorted according to y-coordinates. Then we take each house and compare it with other houses in the strip and update our closest and second closest pair of houses accordingly. Here it may seem like we are using brute force approach for solving the problem of strip which may cause a time complexity of $O(n^2)$. But it is not the case here. We can there can be at most 16 points within the nearestSecondDistance × 2nearestSecondDistance rectangle. For nearestSecondDistanec× nearestSecondDistance square, there can be only four houses that is equal to d_2. And within this square there can be only the houses that have a distance of d_1 from the house and this number is 4. So, the nearestSecondDistance × nearestSecondDistance can have at most 8 points.



Similarly, the other half of the rectangle can have at most 8 points. Thus, the rectangle of our concern can have at most 16 points. So, for a point on the strip, it is sufficient to make comparison with the next 15 points.

Calculating Running Time

Now we check if we have achieved the required time complexity. Suppose T'(n) is the running time of the algorithm. We divide the array of houses in two subarrays and make recursive calls on these two subarrays. We create the strip and ultimately update the closest and second closest pair of houses in running time O(n). Presorting the array includes a running time of O(nlg n).

So, the combined running time $T(n) = T'(n) + O(n \lg n)$.

$T'(n) = 2T(n/2) + O(n)$, when $n>3$

$T'(n) = O(1)$, when $n<=3$.

Now we use master theorem to prove that $T'(n) = O(n \lg n)$

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

Now $n^{\log_b a} = n = f(n)$

So, according to master theorem $T'(n) = O(n \lg n)$

From this we can say, $T(n) = O(n \lg n)$, which is our desired running time.