Closest Pair of Points | O(nlogn) Implementation

Difficulty Level : Hard • Last Updated : 15 Apr, 2020

We are given an array of n points in the plane, and the problem is to find out the closest pair of points in the array. This problem arises in a number of applications. For example, in air-traffic control, you may want to monitor planes that come too close together, since this may indicate a possible collision. Recall the following formula for distance between two points p and q.

$$||pq|| = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$$

We have discussed a divide and conquer solution for this problem. The time complexity of the implementation provided in the previous post is O(n (Logn)^2). In this post, we discuss implementation with time complexity as O(nLogn).

Following is a recap of the algorithm discussed in the previous post.

1) We sort all points according to x coordinates.

2) Divide all points in two halves.

3) Recursively find the smallest distances in both subarrays.

4) Take the minimum of two smallest distances. Let the minimum be d.

5) Create an array strip[] that stores all points which are at most d distance away from the middle line dividing the two sets.

6) Find the smallest distance in strip[].

7) Return the minimum of d and the smallest distance calculated in above step 6.

The great thing about the above approach is, if the array strip[] is sorted according to y coordinate, then we can find the smallest distance in strip[] in O(n) time. In the implementation discussed in the previous post, strip[] was explicitly sorted in every recursive call that made the time complexity O(n (Logn)^2), assuming that the sorting step takes O(nLogn) time.
In this post, we discuss an implementation where the time complexity is O(nLogn). The idea is to presort all points according to y coordinates. Let the sorted array be Py[]. When we make recursive calls, we need to divide points of Py[] also according to the vertical line. We can do that by simply processing every point and comparing its x coordinate with x coordinate of the middle line.

Following is C++ implementation of O(nLogn) approach.