CLOSEST PAIR PROBLEM

Course Name: ALGORITHMIC PROBLEM SOLVING
Course code: 17ECSE309

By: Abhishreya Sharma
USN: 01FE15BEC010

1. Introduction

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. The Brute force solution is O(n^2), compute the distance between each pair and return the smallest. We can calculate the smallest distance in O(nLogn) time using Divide and Conquer strategy. Here a O(n x (Logn)^2) approach is discussed

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

2. Algorithm

Following are the detailed steps of a O(n (Logn)^2) algortihm.
Input: An array of n points *P[]*
Output: The smallest distance between two points in the given array.
As a pre-processing step, input array is sorted according to x coordinates.

1) Find the middle point in the sorted array, we can take *P[n/2]* as middle point.
2) Divide the given array in two halves. The first subarray contains points from P[0] to P[n/2]. The second subarray contains points from P[n/2+1] to P[n-1].
3) Recursively find the smallest distances in both subarrays. Let the distances be dl and dr. Find the minimum of dl and dr. Let the minimum be d.
4) From above 3 steps, we have an upper bound d of minimum distance. Now we need to consider the pairs such that one point in pair is from left half and other is from right half. Consider the vertical line passing through passing through P[n/2] and find all points whose x coordinate is closer than d to the middle vertical line. Build an array strip[] of all such points.
5) Sort the array strip[] according to y coordinates. This step is O(nLogn). It can be optimized to O(n) by recursively sorting and merging.
6) Find the smallest distance in strip[]. This is tricky. From first look, it seems to be a O(n^2) step, but it is actually O(n). It can be proved geometrically that for every point in strip, we only need to check at most 7 points after it (note that strip is sorted according to Y coordinate).
7) Finally return the minimum of d and distance calculated in above step (step 6)

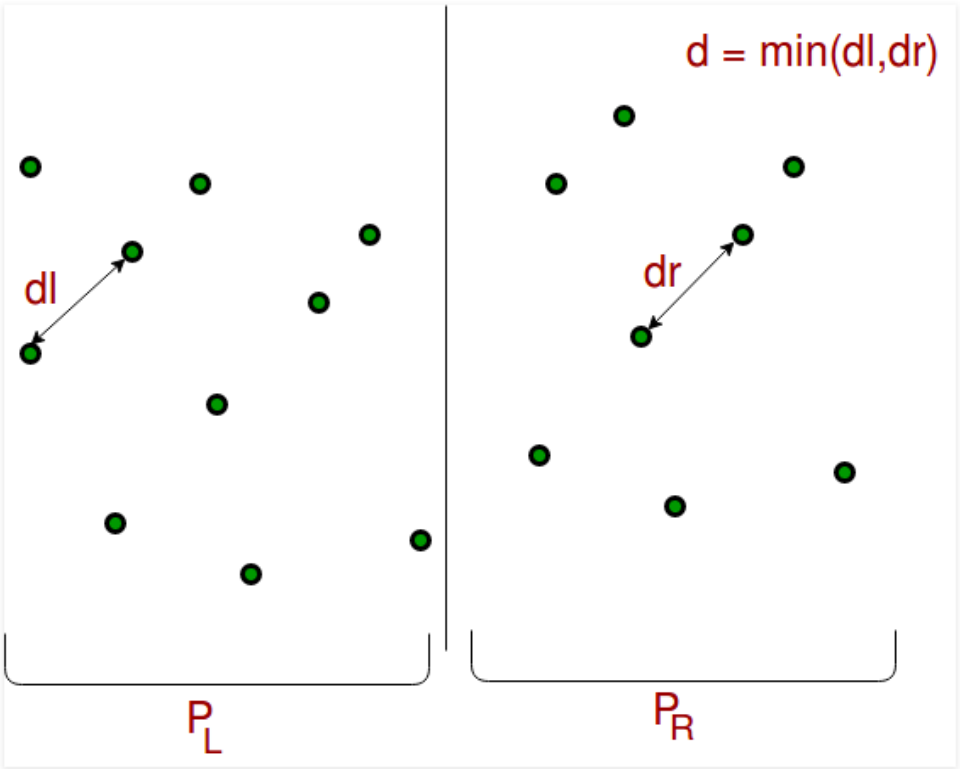

Figure 1 dividing the given array and finding the smallest distance in each sub-space

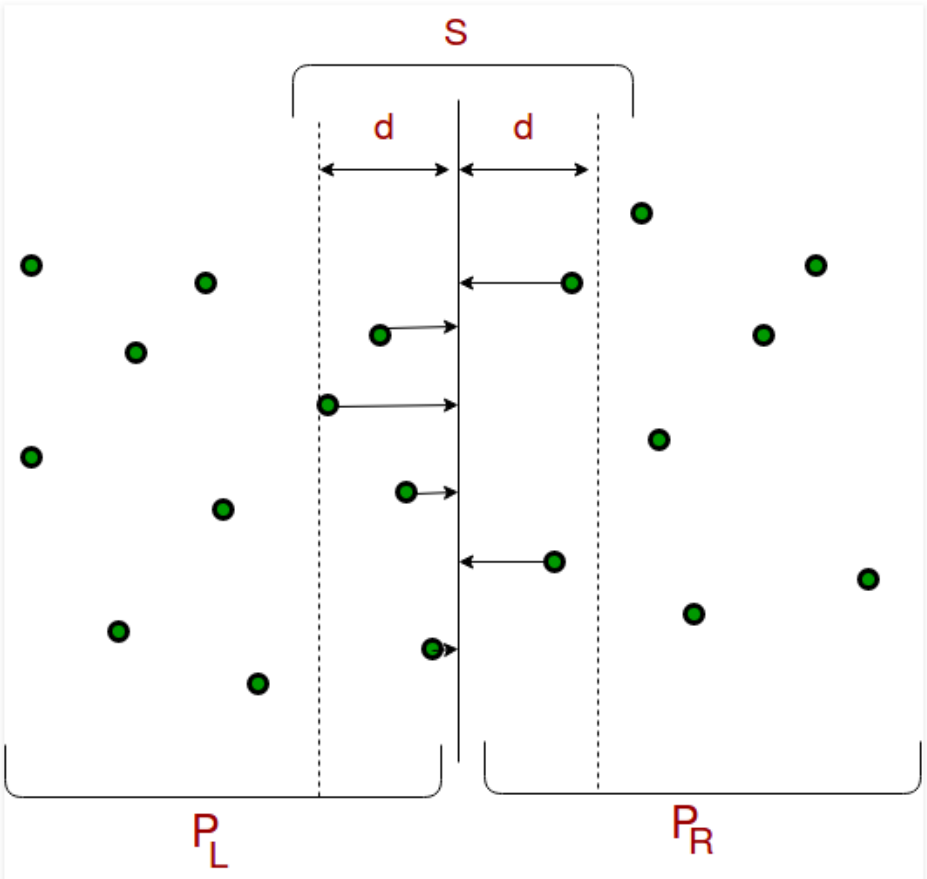

Figure 2 finding all points whose x coordinate is closer than d to the middle vertical line.

Time Complexity Let Time complexity of above algorithm be T(n). Let us assume that we use a O(nLogn) sorting algorithm. The above algorithm divides all points in two sets and recursively calls for two sets. After dividing, it finds the strip in O(n) time, sorts the strip in O(nLogn) time and finally finds the closest points in strip in O(n) time. So T(n) can expressed as follows

T(n) = 2T(n/2) + O(n) + O(nLogn) + O(n)

T(n) = 2T(n/2) + O(nLogn)

T(n) = T(n x Logn x Logn

3. References

https://www.geeksforgeeks.org/closest-pair-of-points/

https://www.geeksforgeeks.org/closest-pair-of-points-onlogn-implementation/