Programming Assignment I

Closest-Points Problem

Pseudo-code of the Algorithm

Step 1- Sort the points on x-coordinate and on y-coordinate.

Step 2- Divide them into two halves

2.1) Let the value of the median(when sorted on x-coordnates) be x_mid.

2.2) Split x into x_left and x_right. Points in x_left have the value when x-coordinate <=x_mid and x_right contains the rest of the coordinates.

2.3) Split y into y_left and y_right. Points in y_left have the value when y-coordinate <=x_mid and y_right contains the rest of the coordinates.

2.4) Let min1 and min2 be the two pairs of the form (x1,y1) and (x2,y2) containing the closest pair of points.

Step 3- Recursive calls

3.1) Let d1= distance of the closest pair on input (x_left,x_right) and d2=distance of the closest pair on input (y_left,y_right).

3.2) Compute d=min(d1,d2) and similarly the closest pairs min1 and min2.

3.3) Around x_mid we will consider a band of length d on both sides. Anything outside of the band will have a distance >d.

Step 4- Combine step

4.1) From the vector sorted on the basics of y-coordinate extract all the points for which point.x > x_mid – d and put in a separate vector called band.

4.2) Scan band from top to bottom, comparing each point against a finite number of points(7 points) and computing the distance between them. And if we get a distance smaller than the current distance then update both the distance and the closest pairs. This step will take O(n) time.

Proof of Correctness of the Algorithm

Suppose there exist a closest pair of points in the plane such that one point lies in the left region and one point lies in the right region(left and right region partitioned on the basics of x_mid) and the distance between them is less than distance d.

All we need to show is that the algorithm will consider those two points in the computation and thus will get us the minimum distance between the closest pair of points.

Now take the point which comes first on the basics of y-coordinate values and create a horizontal and a vertical line on both directions. We will get a rectange of length 2d and breadth d.

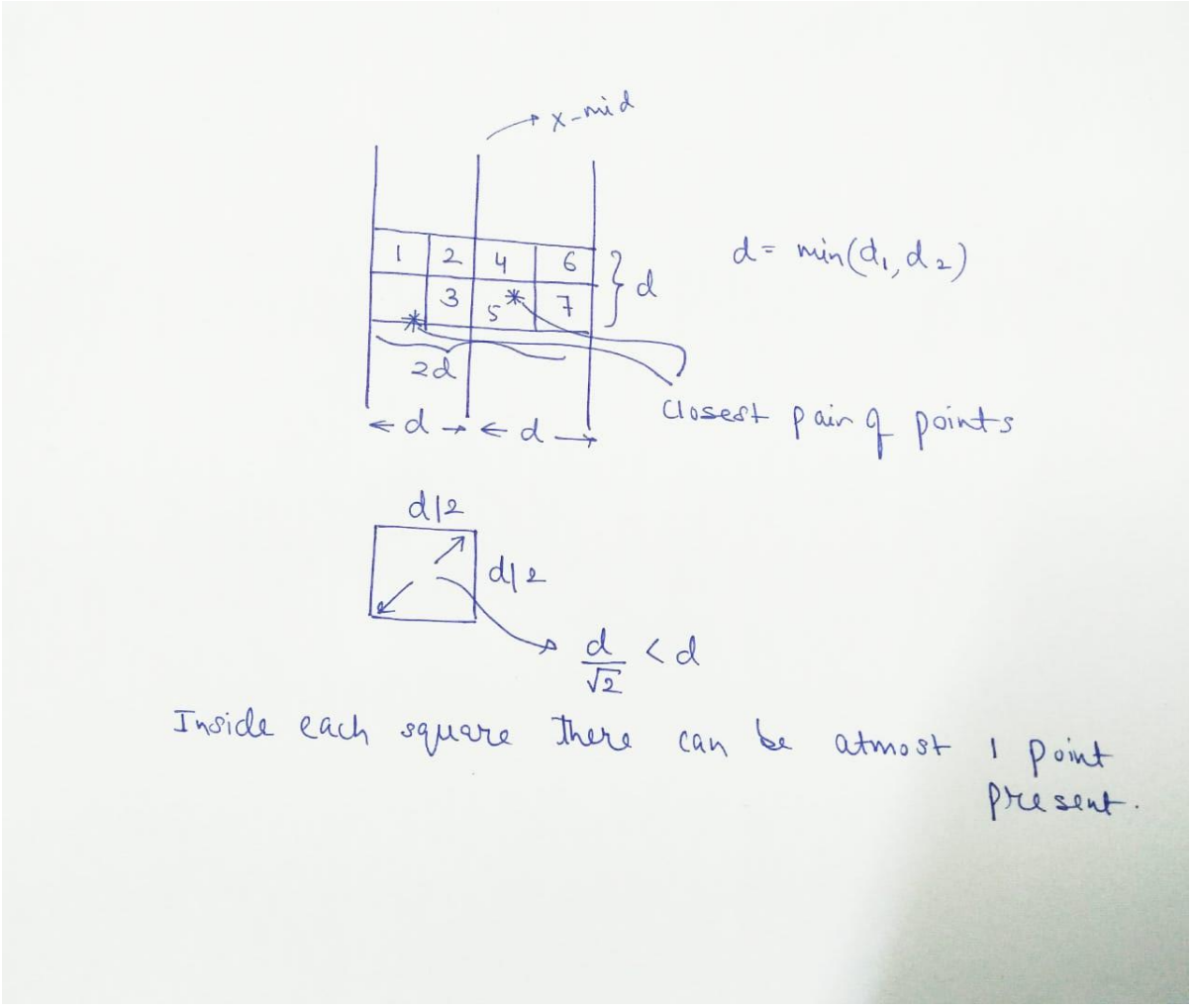
Now the claim is that the closest pair lies inside the rectange, if they are not that means the distance between them is more than d and they are not the closest pair which contradicts our assumption of them being the closest pairs. So, indeed those two pairs would lie inside the rectangle.

To Prove: The number of points inside the rectange excluding the initial point should be atmost 7.

Proof: Lets further break down the rectange into smaller squares, the dimensions of each of the squares is d/2 and the length of the diagonal is d/√2 which is strictly less than d.

Now, inside any of these little squares there can be atmost 1 point. If there are two points present then the distance between those two points will be less than or equal to d/√2(the length of the diagonal) which is strictly less than d which is not allowed. As each of the little squares are either in the left region or the right region and within one of these regions we cannot have pair of points with distance less than d by the definition of d.

So, apart from the point which is in one square we can have atmost 7 other points in 7 sqaures and it's closest pair must lie within these 7 squares. That proves that the algorithm will definitely compare these two pairs and hence will return the minimum distance. This also proves that the combine step will run for atmost O(7*n) times which is O(n) time.



Running time analysis of the Algorithm

Let the time complexity of the above algorithm be T(n). As a preprocessing step we have used mergesort to sort the points on the basics of x-coordinates and y-coordinates respectively which runs in O(nlogn) time.

The sorted points are then fed into the function which divides the set of points into two halves and recursively calls them. After dividing into parts it finds the band in O(n) time. Finally finds the closest points in the band in O(n) time.

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

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

$T(n)=O(nlogn)$