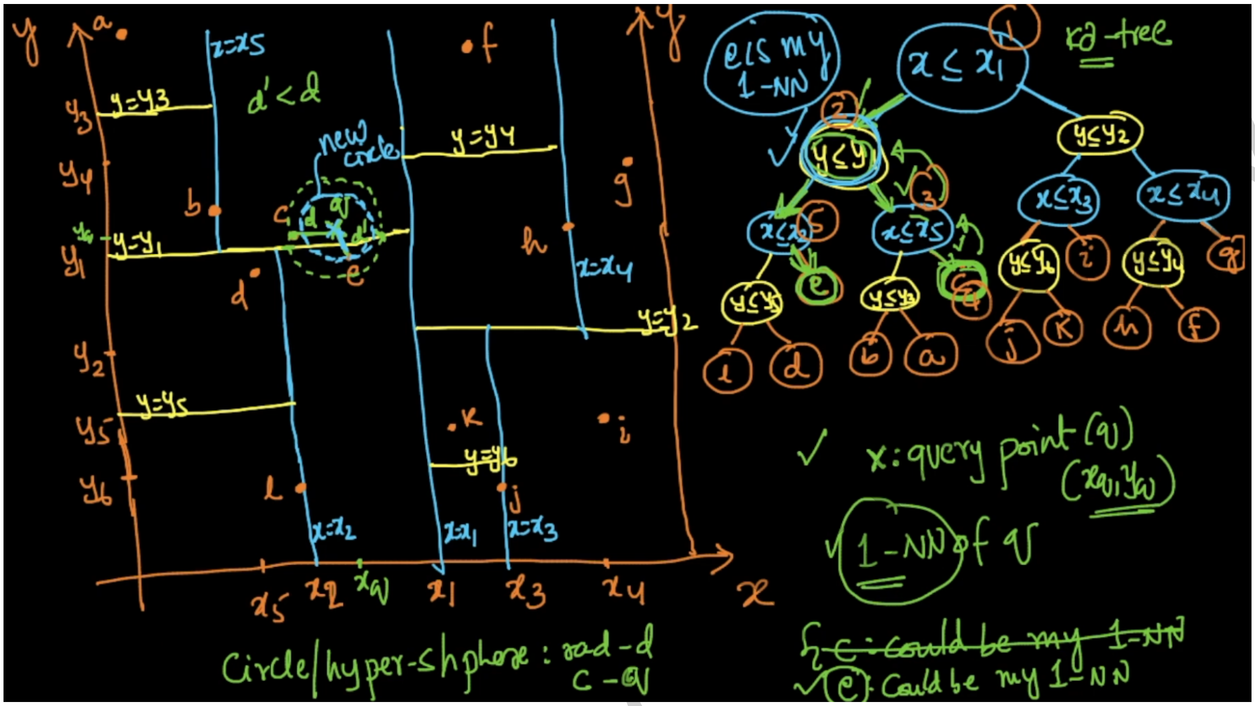
29.23 Finding Nearest Neighbors using KD-Tree



Let us assume we have to find out only the 1 nearest neighbor to a given point (x_q, y_q). In the same way as we did in the previous topic, here instead of using ‘x’, we have to use ‘x_q’ and instead of using ‘y’, we have to use ‘y_q’.

We have to keep traversing the tree in the direction of the results of the conditions at the nodes, until we reach a leaf node. Here the first lead node value we reach is ‘c’. So this tree helps us in finding in which cuboid our point (x_q, y_q) lies in. So we can say that ‘c’ is the 1st nearest neighbor, but we are still not sure.

We shall calculate the distance between ‘c’ and the point (x_q, y_q). Let the distance be ‘d’. As ‘c’ is on the same side as (x_q, y_q), we try to draw a hyperplane with centre at (x_q, y_q) and radius ‘D’.

If we find any point inside this hypersphere, then we can say that point is the 1st nearest neighbor of (x_q, y_q). We looked at only one side of the line $y=y_1$, as ’c’ and (x_q, y_q) are on the same side. We shall now look at the other side of $y=y_1$, to find out the nearest neighbor for (x_q, y_q). For that we need to backtrack till we reach y<=y1, and then as we have already traversed in the right subtree, we then have to traverse in the left subtree, according to the results of condition checking and reach the left node ‘e’. Now we can say ‘e’ also could be our nearest neighbor.

Let us now calculate the distance between (x_q, y_q) and ‘e’ and let this distance be D’.

If D<D', we can ignore 'e' and say that 'c' is the 1st nearest neighbor.
If D'<D, we can ignore 'c' and say that 'e' is the 1st nearest neighbor.

From the figure, it is clear that D'<D, so we can ignore 'c' and declare 'e' as the 1st nearest neighbor.
But again we have to draw a circle/sphere with the centre at (x_q, y_q) and radius D'. Now we'll check if the new circle intersects any other lines other than $y=y_1$. As we don't find any, we can confirm that 'e' is the 1st nearest neighbor.

Note: Number of Comparisons while searching for an element in KD-Tree is
Best Case: O(log(n))
Worst Case: O(n)

It means if there are no intersections of the planes, then the number of comparisons would be O(log(n)). In the presence of intersections, then the number of comparisons would increase.