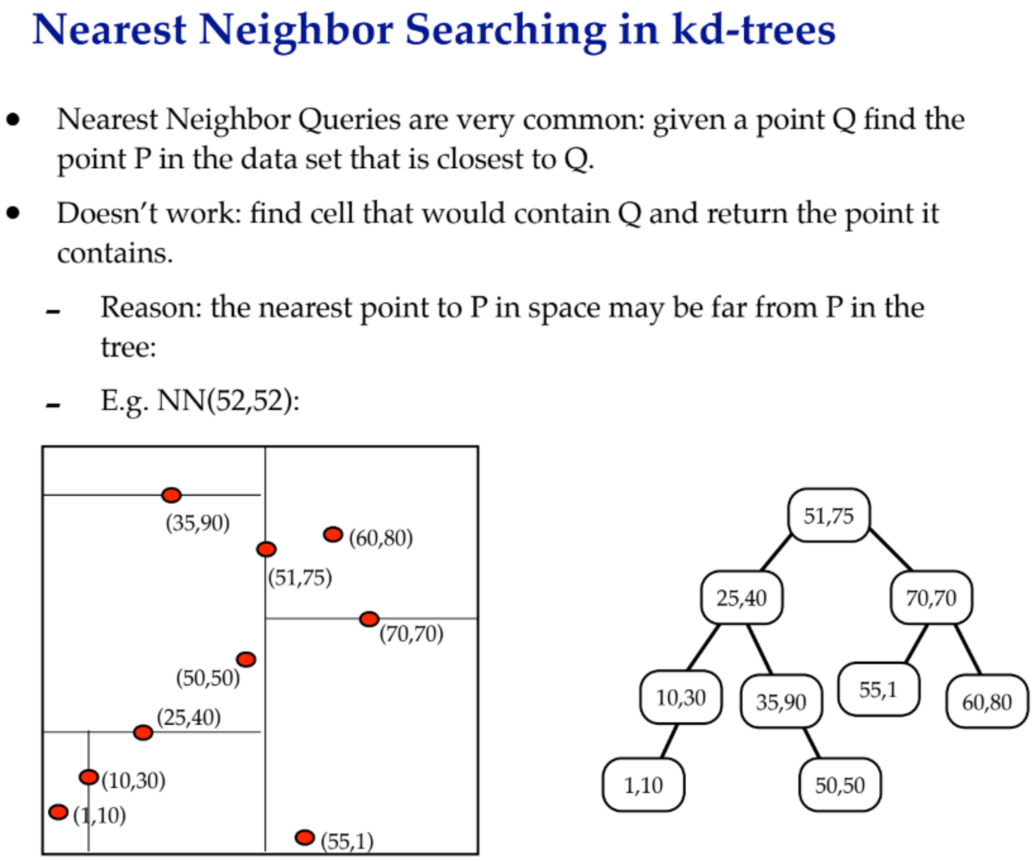
BASIC IDEA:

Below are the snapshots of basic idea of nearest neighbour concept. And after that, I have explained my idea to solve given problem statement.



Nearest Neighbor Searching in kd-trees

- Nearest Neighbor Queries are very common: given a point Q find the point P in the data set that is closest to Q.
- Doesn't work: find cell that would contain Q and return the point it contains.
  - Reason: the nearest point to P in space may be far from P in the tree:
  - E.g. NN(52,52):

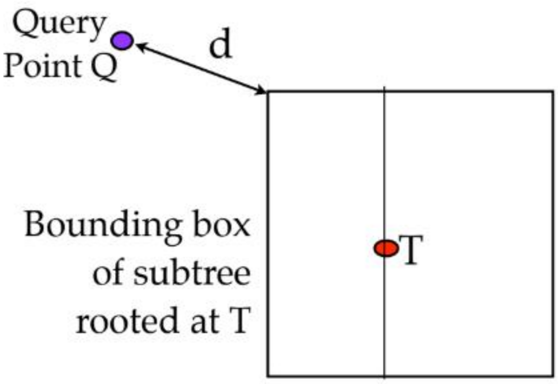
kd-Trees Nearest Neighbor

- Idea: traverse the whole tree, **BUT make two modifications to prune to search space:**

1. Keep variable of closest point C found so far. Prune subtrees once their bounding boxes say that they can't contain any point closer than C

2. Search the subtrees in order that maximizes the chance for pruning

Nearest Neighbor: Ideas, continued



If d > dist(C, Q), then no point in BB(T) can be closer to Q than C. Hence, no reason to search subtree rooted at T.

Update the best point so far, if T is better:
if dist(C, Q) > dist(T.data, Q), C := T.data

Recurse, but start with the subtree "closer" to Q:
First search the subtree that would contain Q if we were inserting Q below T.

IMPLEMENTATION IDEA:

According to the given problem statement, we need to process the algorithm by applying four categories:

1. A1-Hotels
2. A2-Hospitals
3. A3-Police station
4. A4-Restaurants

Also because we need to store the coordinates of these categories, so in my solution, I will use List as my Data Structure. A 2D tree will be created with dimension 1 as latitude and another dimension as longitude.

To build a 2D tree, I will use Binary Search Tree where each level switches with the dimensions specified above. Each node will have a left child and a right child representing a spot from given categories.

In the algorithm, since we have to find k nearest neighbours, We need to keep track of data like accepted points and discarded points, so for this I will use Hashmap data structure. A queue data structure will be used to get nearest point where the top element will represent nearest point. I will loop over the array of point and each element will then be marked either as accepted or rejected. Algo will run while all points are considered for evaluation.

1. Loop over array of points while index not equals size.
2. If point already checked, then exit.
3. Mark each point into rejected or accepted state by comparing current state.
4. Prioritize the queue and update map.
5. Check if size of queue equals 5 and index = last. If yes then terminate.
6. Else add it to discarded list.
7. Increase index.
8. End loop.
9. Print result.