K Dimensional Tree assignment.

Reference used for algorithm: https://en.wikipedia.org/wiki/K-d_tree

With respect to the implementation, I have made a use of a vector to represent the coordinates at each point. This vector along with its original index in the sample data forms the KDPoint structure.

Every node in the KDTree is a KDNode, which can be a leaf node or an internal node. All the KDPoint objects form the leaf nodes while the internal nodes are used to indicate the split axis and the split value. This is to try to build a balanced KDTree. Everytime an internal node is created, an axis is chosen in circular manner (depth % number of dimensions) and a median value for that axis in the concerned subset of points is taken as the split value for the internal node. Standard sort is used to find the median to sort the subset of points according to given axis at each depth to get the median.

To search the nearest point for one given point, the current nearest point is selected when the first leaf node is reached (DFS). The algorithm then unwinds the recursion of the tree, checking if the current node is nearer than the current nearest node. this is implemented as a simple comparison to see whether the distance between the splitting coordinate of the search point and current node is lesser than the distance from the search point to the current best i.e. intersecting the splitting hyperplane with a hypersphere around the search point that has a radius equal to the current nearest distance.

Computational Complexity:

Bulding the tree: $O(n \log^2 n)$ assuming the std::sort takes O(n log n) time

Finding 1 nearest neighbor: balanced *k*-d tree with randomly distributed points takes $\underline{O}(\log n)$

What could have been done better:

I have used a dumb approach to dump the tree and rebuild it. It dumps the leaf nodes and rebuilds it using the same procedure it used for the creating the tree originally. I would use a better serializer and deserializer to do this.

The implementation can be easily extended to find the K nearest points for a given point.