d-Dimensional Kd-Trees

s.marzieh.abdolahi

Oct 2014

- A data structure to support range queries in IR^{d}.
- Preprocessing time: O(n log n)
- Space complexity: O(n)
- Query time: $O(n^{1-1/d} + k)$.

CONSTRUCTION OF DD KD-TREES

- The construction algorithm is similar to 2-dim. case.
- At the root, we split the set of points into two subsets of same size by a hyperplane ⊥ x1-axis.
- At the children of the root, the partition is based on the second coordinate: x2-coordinate

- So on so forth ... until at depth d -1, we partition on the last coordinate: x_d-coordinate.
- At depth d, we start all over again by partitioning on first coordinate.
- The recursion stops until there is only one point left, which is stored as a leaf.

CONSTRUCTION TIME AND SPACE OF DD KD-TREES

- Since a dD kd-tree is a binary tree with n leaves, the storage is O(n), and construction time is O(n log n).(assuming d is a constant.)
- More precisely,
  - Storage = O(d . n) since we store d x_i-lists.
  - Construction time = O(d . n log n) since we need to compute d x_i-lists for each node.

- Query routine
  - visits those nodes whose regions are properly intersected by R;
  - traverses subtrees that are rooted at nodes whose regions are fully contained in R.
- It can be shown that query time = O($n^{1-1/d} + k$).