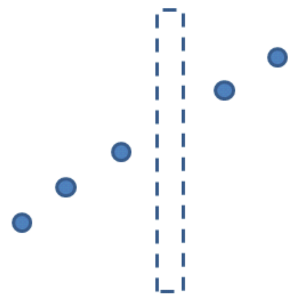
2013 Spring CS504

Homework #1 Solution

5.1

$Q(n) = O(\sqrt{n})$, using either Master theorem or substitution method.

To show that it is also a lower bound, here is an example:



5.3

(a) BUILDKDTREE (P, depth)

Input. A set of points P and the current depth *depth*.

Output. The root of a kd-tree storing P.

1. **if** P contains only one point
2. **then return** a leaf storing this point
3. **else**
4. **then** Split P into two subsets, P_1 and P_2, of roughly the same size by a hyperplane h perpendicular to the x_{depth}-axis.
5. **if** depth is d **then** depth ← 0
6. v_{left} ← BUILDKDTREE(P_1, depth+1)
7. v_{right} ← BUILDKDTREE(P_2, depth+1)
8. Create a node v storing h, make v_{left} the left child of v, and make v_{right} the right child of v.
9. **return** v

Presorting the set of points on $x_1, \ldots, x_d$ – coordinate takes $O(d\,n \log n)$.

So we can find the splitting hyperplane in $O(n)$ time, no changes on the recurrence of 2d case.

We may consider d as constant and the construction time becomes $O(n \log n)$.

(b) We can easily generalize SEARCHKDTREE algorithm to d-dimension. It takes $O(k)$ to process the subroutine REPORTSUBTREE. Now let $T(n)$ as the number of intersected regions in a kd-tree whose root contains a splitting hyperplane perpendicular to the x_i-axis. To write a recurrence, we have to go down d steps in the tree.

$$T(n) = \begin{cases} O(1) & if\ n = 1 \\ 2^{d-1} + 2^{d-1} T\left(\frac{n}{2^d}\right) & if\ n > 1 \end{cases}$$

So $T(n) = O\left(n^{1-\frac{1}{d}}\right)$, and total query time is $O(n^{1-\frac{1}{d}} + k)$.

(c) Each node of the kd-tree uses O(d) storage, so total $O(dn)$.

To presort the points, it takes $O(d\,n \log n)$ time.

No dependence on d of the query time.

5.10

(a) In the construction of range tree, each internal node stores the number of all leaf nodes in the subtree rooted itself.

(b) Construct a d-dimensional range tree using the same idea. Then,

$$Q_d(n) = O(\log n + O(\log n) \cdot Q_{d-1}(n)$$

$$Q_2(n) = O(\log^2 n)$$

So, $Q_d(n) = O(\log^d n)$.

(c) (will be discussed in the class)