Assignment no. 3

due: January 11th, 2006

Exercise 3.1 In some applications one is interested only in the number of points that lie in a range rather than reporting all of them. Such queries are often referred to as *range counting queries*. In this case one would like to avoid paying the $O(k)$ additive term in the query time.
(a) Describe how a 1-dimensional range tree can be adapted such that a range counting query can be performed in $O(\log n)$ time. Prove the query time bound.
(b) Describe how d-dimensional range counting queries can be answered in $O(\log^d n)$ time. Prove the query time bound.
(c) Describe how fractional cascading can be use to improve the query time by a factor $O(\log n)$ for 2- and higher dimensional range counting queries.

Exercise 3.2 Give an example of a set of n points in the plane, and a query rectangle for which the number of nodes of the kd-tree visited is $\Omega(\sqrt{n})$.

Exercise 3.3 The algorithm we saw in class for searching in a kd-tree (where the search is guided by comparing the *region* of a node with the query region) can also be used when querying with ranges other than rectangles. For example, a query is answered correctly if the range is a triangle.
(a) Show that the query time for range queries with triangles is linear in the worst case, even if no points are reported at all. Hint: Choose all the input points to lie on the line $y = x$.
(b) Suppose that a data structure is needed that can answer triangular range queries but only for triangles whose edges are horizontal, vertical or have slope $+1$ or -1. Devise a linear size data structure that answers such queries in $O(n^{3/4} + k)$ time, where k is the number of points to be reported. Hint: Choose 4 coordinate axes in the plane and use a "4-dimensional" kd-tree.
(c) Improve the query time to $O(n^{2/3} + k)$.

Exercise 3.4 Given a star-shaped polygon P with n vertices, show that after $O(n)$ preprocessing time, one can determine whether a query point lies in P in $O(\log n)$ time.

Exercise 3.5 (optional!) Give a randomized algorithm to compute all pairs of intersecting segments in a set of n line segments in expected time $O(n \log n + A)$, where A is the number of intersecting pairs.