Exercises and hand-in

Advanced database technology

March 21, 2005

Today's exercises consider three of the multi-dimensional index structures covered in GUW:

- The grid file.
- The kd-tree.
- The R-tree.

These structures are often used in practice as they are simple and typically work well. The aim of these exercises is to investigate in what situations (i.e., for what kind of data) these index structures behave poorly. For any set of data there are many possible choices for the index (which grid lines to use for grid files, and which rectangles to use for R-trees). We seek data that is bad *no matter* how these choices are made. To simplify, we will consider the 2-dimensional case, and static data structures (no updates).

1. First we consider nearest neighbor queries for a point set that is distributed *almost* exactly along a circle.
   - How will each of the index structures represent such a set? How much internal memory is needed?
   - How will they cope with a query for the nearest neighbor of a point almost exactly at the center of the circle?[1]
2. We then consider range queries on a point set that is distributed almost exactly along a horisontal line.
   - How will each of the index structures represent such a set? How much internal memory is needed?
   - How will they cope with a query for the points in a rectangle that covers all of the points with respect to x-coordinate, but is just above the horizontal line and contains just a tiny fraction of the points?
   - What is the best (worst case) search time we could hope for?[2]

The following exercise is to be handed in at the latest April 11 at 10.00 AM.

Problem 5 from the ADBT exam, June 2004. (See News section on home page.)
Hand in before the lecture on that day, or earlier.

[1] A much better solution in this case is to use the planar point location data structure you saw in the lecture. This is left as a home exercise.

[2] ITU has some of the world's leading researchers in 2-dimensional range searching. They have devised very good worst-case efficient data structures for internal memory.