Massachusetts Institute of Technology
6.854J/18.415J: Advanced Algorithms
David Karger

Handout 22
Wednesday, November 14, 2012

# Problem Set 11

**Due: Wednesday, November 21, 2012.**

**Collaboration policy:** collaboration is *strongly encouraged*. However, remember that

1. You must write up your own solutions, independently.
2. You must record the name of every collaborator.
3. You must actually participate in solving all the problems. This is difficult in very large groups, so you should keep your collaboration groups limited to 3 people in a given week.
4. **No bibles. This includes solutions posted to problems in previous years.**

**Problem 1.** Recall the range-tree data structure seen in class. Given a set $P$ of $n$ points in the plane, we can build a range tree to answer queries of the form: which points in $P$ lie in a given rectangle? The query time is $O(\lg^2 n + k)$, where $k$ is the number of points in the rectangle, construction time is $O(n \log n)$, and the space is $O(n \log n)$. The range tree consists of a primary structure of a balanced binary search tree on the $x$-coordinate of the points in $P$. Each internal node $v$ of the primary tree has an associated *secondary structure* of a balanced binary search tree on the $y$-coordinates of the leaf nodes in the subtree rooted at $v$.

In this problem, we consider one possible approach to lowering the space requirements of range trees.

**(a)** Suppose that only the nodes with even depth have an associated *secondary structure*. Show how the query algorithm can be adapted to answer queries correctly.

**(b)** Analyze the storage requirements and query time of this modified data structure.

**(c)** Suppose that only the nodes with depth $0, \lfloor \frac{1}{j} \log n \rfloor, \lfloor \frac{2}{j} \log n \rfloor, \ldots$ have an associated *secondary structure*, where $j \geq 2$ is a constant. Analyze the storage requirements and query time of this data structure. Express the bounds in terms of $n$ and $j$.

**Problem 2.** A problem last week found lines (and polygons) *contained* in a rectangle; here we consider finding lines *crossing* a rectangle. As a starting point, suppose you are given an *interval tree* data structure. This takes $n$ possibly-overlapping intervals on the real line, and builds a size-$n$ data structure that can, in $O(k + \log n)$ time, output the set of all intervals

intersecting with a given query interval (you may optionally design this data structure if you wish). Given such a data structure, show that you can build a size $O(n \log n)$ data structure for the following problem: given $n$ vertical and horizontal segments in the plane, and given a query rectangle, output all the segments that intersect that query rectangle in $O(k + \log^2 n)$ time.

**Problem 3.** Given a set of $n$ points in the plane, we wish to find the closest pair of points. We show how to do so in $O(n \log n)$ time by sweeping a vertical line past the points. At any point in time, we will know the distance $d$ between the closest pair of points behind the sweep line. We maintain a "strip of infinite height and of width $d$ behind the sweep line, and the set of points inside the strip.

**(a)** Argue that when the sweep line encounters a new point $p$, if $p$ is one point in the closest pair behind the sweep line, then the other point in the closest pair is inside the strip—and in fact, in a particular portion of the strip quite close to the new point.

**(b)** Argue that in fact, this portion of the strip can contain only a constant number of points.

**(c)** Develop a data structure to associate with the sweep line so that these candidates for closest pair can be identified quickly ($O(\log n)$ time per event), and show how to maintain this data structure as the line sweeps ($O(\log n)$ time per event). Conclude an $O(n \log n)$ time bound for closest pair.

**(d)** Does the first algorithm above generalize to any higher dimension $k$? What is the time bound as a function of $k$?

**(e)** Alternatively, suppose you construct the Voronoi diagram on the points. Show how the closest pair can then be identified in $O(n)$ time. This gives an alternative $O(n \log n)$ time algorithm in 2 dimensions.

**Problem 4.** Suppose you are given $N$ line segments in the plane, each of which is horizontal or vertical. Show how to compute all $K$ intersections among the line segments using $O(\frac{N}{B} \log_{M/B} \frac{N}{B} + K/B)$ memory transfers in the external-memory model. (**Hint:** Use a sweep-line algorithm and buffer trees.) You may assume that no two horizontal segments intersect and that no two vertical segments intersect.

**Problem 5.** How long did you spend on this problem set (not including the project)? Please answer this question using the Google form that is sent to you via a separate email. This problem is mandatory, and thus counts towards your final grade. It is due by the Monday 2:30pm after the pset due date. You can find the link to the form on the course website.

**Problem 6.** Read through the handout on the final project, which is linked from the course website. Submit a paragraph-long (or so) proposal for the final project detailing the topic and scope of the proposed project. Include citations to relevant papers. If you are collaborating with other students, submit just one copy with all of your names on it. Please limit your groups to 3 students. Email your proposal to Professor Karger at `karger@mit.edu` and the TAs. Put "6.854 project" in the subject line of your email.