Advanced Algorithms, Fall 2011

Prof. Bernard Moret

Test #3

This is an in-class test. The test is open book, open notes; you can use a laptop, tablet, or PDA, but you must turn off any wireless, phone, or other remote access connections.

The test starts at 13:15 and ends at precisely 16:00. There are four problems, all with the same weight.

Question 1. (Dynamic Programming) Devise and analyze a dynamic programming solution to the following problem. You are given a vector A of n integer values, indexed from 1 to n. You want to find a sequence of indices i_1, i_2, ..., i_k (k is not fixed, but part of the optimization) such that (i) we have $1 \leq i_1 < i_2 < \ldots < i_k \leq n$; (ii) no two of these indices are consecutive; and (iii) the sum $\sum_{j=1}^{k} A[i_j]$ is maximized.

Question 2. (Randomized Incremental Algorithm) Devise and analyze a randomized incremental algorithm to solve the following problem: given a collection of n vertical segments, does there exist a (non-vertical) line that crosses all of them? Your algorithm should work in expected linear time. (Hint: use a result from the solution to Homework 8, Problem 3—attached)

Question 3. (Fingerprinting) A Boolean function can be represented by a directed acyclic graph (dag) with one source and two leaves as follows. One leaf is labelled 0 and the other 1. Each remaining node is labelled with the name of a variable and has exactly two outgoing arcs, one labelled 0 and the other labelled 1. On any path from the source to a leaf, no two nodes can be labelled by the same variable. Given an assignment of truth values to the variables, we can follow a path from the source to one of the leaves by using the arc that corresponds to the value of the variable. For example, the dag shown on the next page represents the function $f(x_1, x_2, x_3) = \overline{x_1 x_2 x_3} + x_1 x_2 + x_2 x_3$

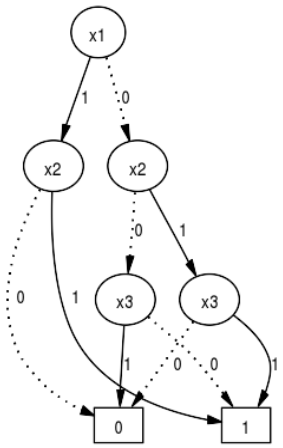
Devise and analyze a randomized algorithm to test whether two such dags represent the same Boolean function. Your algorithm should compute one value for each dag from a set much larger than $\{0, 1\}$; if these two values differ, your algorithm correctly reports different Boolean functions; if they agree, your algorithm reports the same Boolean function, but with some probability of error that you must bound. (Hint: count the number of assignment values.)

Question 4. (Randomized Analysis) Consider the following algorithm for 2-coloring a 2-colorable connected graph. First randomly color the vertices (from the set of two colors). Then, while there remains an edge with two identically colored endpoints, randomly pick one of its endpoints and change its color; otherwise stop.

Analyze the expected number of color changes. Proceed as follows.

1. Set up a Markov chain of $|V| + 1$ states, where each state corresponds to the number of vertices that are correctly colored. (There always are two valid colorings for a connected graph, but for this purpose assume that a fixed reference coloring is chosen—you can always adjust the result by a factor of 2 later.) Define the possible state transitions and derive their probabilities.
2. Compute the expected number of transitions needed to reach the last state of the chain from some arbitrary state, using a tail bound.

The dag for Question 3.



Solution of Problem 3, Homework 8.

Polygons that are locally convex are also known as "star" polygons, since every point in the interior is on a ray originating in the special point. The crucial observation to make is the following: for such a polygon with special point p, that point p lies in the intersection of the halfplanes defined by successive edges of the polygon. And the crucial theorem is that this relationship is an "if and only if:" a polygon is locally convex iff *the intersection of the halfplanes determined by its edges (using left sides and a counterclockwise traversal of the perimeter) is nonempty. So, instead of testing for the existence of p, we will test whether the intersection of these halfplanes is empty, using randomization.*

We use a randomized incremental approach: start with the two consecutive edges (halfplanes) that define the vertex of the polygon with the largest ordinate. We will maintain the highest vertex, v^*, of the intersection polygon; we have just initialized it to be the highest vertex of the input polygon. Now repeatedly add the next, randomly selected, halfplane and update the selection of the highest vertex. If the next halfplane includes v^*, then that halfplane is redundant for the intersection figure and discarded. Otherwise, we must update v^*. Note that the new v^* must lie on the new halfplane's boundary; in fact, it is simply the higher of the two intersection points of that halfplane's boundary with the current convex intersection (if the halfplane boundary happens to be horizontal, then any point along the line segment works, but we can still choose one of the two endpoints). We can find these intersection points by brute force, by simply intersecting each of the halfplanes already processed with the new boundary line. This takes time linear in the number of halfplanes already processed, so the worst-case running time is clearly quadratic, because each successive halfplane added to the collection could modify v^*. However, since we randomized the order of addition, we now want the probability that the next halfplane added will modify v^*. We do this as we handled the smallest enclosing disk problem, by assuming that the ith halfplane was just added and looking at the probability that this addition changed v^*.

Since we assume, as always, that all points are in general position, v^* is determined by the intersection of two boundary lines, from the i such lines (halfplanes) processed so far. Since we randomized the order in which the halfplanes are processed, the probability that the ith halfplane added is one of the two that define the current v^* is bounded by $\frac{2}{i}$, and thus the expected running time of our algorithm is

$$\sum_{i=1}^{n}(1-\frac{2}{i})\cdot\Theta(1)+\frac{2}{i}\cdot\Theta(i)$$

where the first contribution comes from halfplanes that contained the current v^* when added and the second comes from halfplanes that modified v^* and so caused linear work. This is the same sum we encountered with the smallest enclosing disk and is in $\Theta(n)$.