Problem Set 7

Due: November 9, 1999.

NO COLLABORATION IS ALLOWED ON THIS PROBLEM SET.

Problem 1. Consider an ordinary binary search tree augmented by adding to each node x the field $size[x]$, giving the number of keys stored in the subtree rooted at x. Let α be a constant in the range $1/2 \leq \alpha < 1$. We say that a given node x is *α-balanced* if

$$\text{size}[\text{left}[x]] \leq \alpha \text{size}[x]$$

and

$$\text{size}[\text{right}[x]] \leq \alpha \text{size}[x]$$

The tree as a whole or a subtree of it is α-balanced if every node in the tree or subtree respectively is α-balanced.

(a) A 1/2-balanced tree is, in some sense, as balanced as it can be. Given a node x in an arbitrary binary search tree, show how to rebuild the subtree rooted at x so that it becomes 1/2-balanced. You algorithm should run in time $O(size[x])$, and it can use $O(\text{size}[x])$ auxiliary storage.

(b) Show that performing a search in an n node α-balanced binary search tree takes $O(\log n)$ time for any constant α.

(c) Assume that the constant α is strictly greater than 1/2. Suppose that INSERT and DELETE are implemented as usual for an unbalanced n-node binary search tree, except that after every such operation, if any node in the tree is no longer α-balanced, then the subtree rooted at the highest such node is "rebuilt" so that it becomes 1/2 balanced. Show that insert and delete take $O(\log n)$ amortized time per operation. (It might be useful to define a potential function that is a sum of potentials over the nodes.)

Problem 2. An arc is upward critical for a min-cost flow problem if increasing its cost changes the cost of the optimum solution; downward critical of decreasing its cost decreases the optimum cost.

(a) Must every min-cost flow problem with nonzero flow have an upward critical arc?

(b) Give an algorithm for finding all upward and downward critical arcs efficiently.

Problem 3. Suppose you are given a maximum flow problem, but are willing to settle for finding a flow within a multiplicative factor of $1-\epsilon$ times the optimum for some small ϵ.

(a) Give a simple algorithm for approximating the maximum flow value to within a multiplicative factor of m in $O(m \log n)$ time (that is, find a value which is at least $1/m$ and no more than m times the flow value).

(b) Suppose you round all edge capacities down to the nearest integer. By how much can you decrease the maximum flow value?

(c) Using the previous two parts, find a way to find a $(1-\epsilon)$ times maximum flow in $O(mn \log(m/\epsilon))$ time.

* **Problem 4.** Suppose you are given $p \leq n$ max-flow problems that are related in the following way: the only arcs that change capacity are the arcs out of the source, and they each change linearly. That is, with each problem p there is an associated parameter α_p such that the capacity $u(s,x) = u_0(s,x) + \alpha_p u_1(s,x)$, where u_0 and u_1 are the same for all of the problems and always non-negative.

You can obviously solve these problems with p invocations of a maximum flow subroutine, but we'd like to do better. Suppose the problems are arranged such that $\alpha_1 \leq \alpha_2 \leq \ldots \leq \alpha_p$. Consider the following algorithm:

1. run Push-Relabel with FIFO selection on the first problem p_1, giving the maximum flow f_1 and leaving behind final distance labels d_1
2. for $i = 1$ to $p-1$

   using f_i and d_i as starting points, saturate arcs out of the source for p_{i+1} and run Push-Relabel with FIFO selection on p_{i+1} (giving f_{i+1} and d_{i+1}).

It's not obvious that this even works, much less runs fast; your job is to show both:

(a) Show that f_i is a feasible flow for p_{i+1}.

(b) Show that the final distance labels d_i are valid for p_{i+1} as soon as we saturate arcs out of the source.

(c) Show that overall the algorithm performs only $O(n^2)$ relabels, $O(nm)$ saturating pushes, and $O(n^3)$ non-saturating pushes, so in terms of worst-case asymptotic bounds, the time to solve all of the problems is the same as the time to solve one of them.

Problem 5. You work for the Short-Term Capital Management company and start the day with D dollars. Your goal is to convert them to Yen through a series of currency trades involving assorted currencies, so as to maximize the amount of Yen you end up with. You are given a list of pending orders: client i is willing to convert up to u_i units of currency a_i into currency b_i at a rate of r_i (that is, he will give you r_i units of currency b_i for each unit of currency a_i). Assume that going around any directed cycle of trades, $\prod r_i < 1$—that is, there is no opportunity to make a profit by arbitrage.

(a) Formulate a linear programming formulation for maximizing the amount of Yen you have at the end of trading.

(b) Show that it is possible to carry out trades to achieve the objective of the linear program, without ever borrowing currency (**hint:** there is a sense in which your solution can be made acyclic).

(c) Show that there is a sequence of trades that will let you end the day with the optimum amount of Yen and no other currency except dollars.

****Problem 6.** For the simplex method, we ignored the issue of finding a starting vertex. Suppose you have a box that solves $\min\{cx \mid Ax = b, x \geq 0\}$ if given a vertex of the polytope. Suppose you have such a problem but no vertex. Show how two calls to the box can be used to solve the problem. (Hint: devise a new linear program with an obvious vertex, whose optimal solutions are vertices of the original linear program.)