Due Date: Thursday, March 1, 2012

1 Binary Search Trees (BST) (20 points)

Consider the following strategy for deleting elements from a BST. When an element in the BST is to be deleted, instead of actually deleting the element and adjusting the tree, just mark the element as deleted by setting a bit. After a certain number of delete operations, do a clean up of complete tree by deleting all the marked nodes and constructing a perfect BST on the unmarked nodes. Describe a linear-time algorithm to reconstruct a perfect BST after deleting half of the elements.

2 Red-Black Trees (20 points)

Describe a data structure that maintains a set S of (ordered) elements and takes $O(\log n)$ time in the worst case to perform each of the following operations:

- Insert (x), Delete(x), as discussed in the class.
- Rank(x): find the number of elements in S that are less than x.
- Range(ℓ, r): find the number of elements in S that lie between ℓ and r, i.e., return $|\{x \in S \mid \ell \leq x \leq r\}|$; the procedure is not allowed to perform subtractions.

3 Amortized Analysis (20 points)

Chicago has many tall buildings, but only some of them have a clear view of Lake Michigan. Suppose we are given an array $A[1..n]$ that stores the height of n buildings on a city block, indexed from west to east. Building i has a good view of Lake Michigan if and only if every building to the east of i is shorter than i. Design a linear-time algorithm that computes which buildings have a good view of Lake Michigan.

4 Shortest Paths [DPV 4.15] (20 points)

Shortest paths between two vertices of a graph are not always unique: sometimes there are two or more different paths with the minimum possible length. Show how to solve the following problem in $O((|V| + |E|) \log |V|)$ time.

Input: An undirected graph $G = (V, E)$; edge lengths $l_e > 0$; starting vertex $s \in V$.

Output: A Boolean array $usp[.]$; for each node u, the entry $usp[u]$ should be true if and only if there is a unique shortest path from s to u. (Note: $usp[s] = true$.)

5 Generalized Shortest Paths [DPV 4.19] (20 points)

In Internet routing, there are delays on lines but also, more significantly, delays at routers. This motivates a generalized shortest-paths problem. Suppose that in addition to having edge lengths(weights) $\{l_e \mid e \in E\}$, a graph also has vertex costs $\{c_v \mid v \in V\}$. Now define the cost of a path to be the sum of its edge lengths plus the costs of all vertices on the path (including the endpoints). Give an efficient algorithm for the following problem.

Input: A directed graph $G = (V, E)$, positive edge lengths l_e, and positive vertex costs c_v, and a starting vertex $s \in V$.

Output: An array $cost[\cdot]$ such that for every vertex u, $cost[u]$ is the least cost of any path from s to u (i.e., the cost of the cheapest path), under the definition above. Notice that $cost[s] = c_s$.