Devoir d'Algorithmique

Magistère Informatique 1ère Année

Octobre 2010

Treaps

A treap is a binary tree in which every node has both a search key and a priority such that it is a binary search tree with respect to the search keys and a (min-)heap with respect to the priorities. (Tree + Heap = Treap !). Throughout this homework, we assume that all the search keys and priorities are distinct.

Question 1 Suppose we are given a set $\mathcal{T}$ of $\mid \mathcal{T} \mid$ distinct (search key value, priority) pairs. How many treaps are possible on set $\mathcal{T}$?

Question 2 Prove that a treap is exactly the binary search tree that results by inserting the nodes one at a time into an initially empty tree, in order of increasing priority, using the usual insertion algorithm.

Question 3 Write pseudo-codes for (or describe in English/French with enough details how to) search, insert, delete, split and merge such that worst case complexity is linear. (Hint : You may want to use rotations !)

Question 4 Suppose the priorities and independently and uniformly distributed in $[0, 1]$. (Such a treap is called a randomized treap.) What is the expected depth of a node ? Hence deduce the expected running time of the operations in Question 3.

Question 5 What is the exact expected number of leaves in an n-node treap ? (Hint : What is the probability that in an n-node treap, the node with k^{th} smallest search key is a leaf ?)

Question 6 Consider an n-node treap T. Let us denote the node with the i-th smallest search key by x_i. Let i, j, k be integers such that $1 \leq i \leq j \leq k \leq n$.

(a) What is the exact probability that x_j is a common ancestor of x_i and x_k ?

(b) What is the exact expected length of the unique path from x_i to x_k in T ?

Question 7 The left spine of a binary search tree is the path from the root to the node with the smallest key. Symmetrically the right spine is the path from the root to the largest key. The length of a spine is the number of nodes it contains.

(a) Consider the treap $\mathcal{T}$ immediately after x is inserted (using your algorithm in Question 3). Let C be the length of the right spine of the left subtree of x. Let D be the length of the left spine of the right subtree of x. Prove that the total number of rotations that were performed during the insertion of x is equal to $C + D$.

We define indicator random variables

$$X_{i,k} = I\{x_i \text{ is in the right spine of the left subtree of } x_k\}.$$

(b) Show that $X_{i,k} = 1$ if and only if $x_i \cdot prio > x_k \cdot prio$, $x_i \cdot key < x_k \cdot key$, and, for every j such that $i < j < k$, we have $x_i \cdot prio < x_j \cdot prio$.

(c) Show that $\Pr\{X_i, k = 1\} = \frac{1}{k-i+1} \cdot \frac{1}{k-i}$.

(d) Show that the expected length of the right spine of x_k is $1 - \frac{1}{k}$.

(e) Use a symmetry argument to show that the expected length of the right spine of x_k is $1 - \frac{1}{n-k+1}$.

(f) Conclude the expected number of rotations performed when inserting a node into a treap is less than 2.

Question 8 Let's analyze the number of random bits needed to implement the operations of a treap. Suppose we pick a priority p_i at random from the unit interval. Then the binary representation of each p_i can be generated as a potentially infinite series of bits that are the outcome of unbiased coin flips. The idea is to generate only as many bits in this sequence as is necessary for resolving comparisons between different priorities. Suppose we have only generated some prefixes of the binary representations of the priorities of the elements in the treap $\mathcal{T}$. Now, while inserting an item y, we compare its priority $y.prio$ to other's priorities to determine how y should be rotated. While comparing $y.prio$ to some p_i, if the current partial binary representation can resolve the comparison, then we are done. Otherwise, we have the same partial binary representation up to the length of the shorter of the two, and we keep generating more bits for each until they differ.

(a) Compute a tight upper bound on the expected number of coin flips or random bits needed for a single priority comparison. (Note that during insertion, every time we decide whether or not to perform a rotation, we perform a priority comparison. We are interested in the number of bits generated in such a single comparison.)

(b) Generating bits one at a time like this is probably a bad idea in practice. Give a more practical scheme that generates the priorities in advance, using a smaller number of random bits, given an upper bound n on the treap size. Describe a scheme that works correctly with probability $\geq 1 - n^{-c}$, where c is a prespecified constant.

[*You may refer Exercise 13-4 of Cormen et al. to get many of these questions in French (with better explanations)*]