Algorithms Core 15-750 PROBLEM SET # 2 Spring 1999

DUE: 1:00PM Monday February 15 1999

1 Randomly Balanced Search Trees

In this problem set we shall consider another approach to using randomization to maintain a balanced tree.

We consider the **random insertion model** which assigns a probability to each n-node binary tree T equal to the probability of producing T by inserting the elements by a random permutation of $\{1, \ldots, n\}$ into an initially empty tree(no rotations). A tree is called **random** if it is drawn from all n-nodes trees according to this distribution.

Start by proving the following fact about random trees. Let the **size** of a node in a tree be the number of nodes in its subtree.

Problem 1.1 *Let the sizes of the nodes in a random tree T be $s_1, \ldots, s_n$ show that $Prob(T) = \prod 1/s_k$.*

What is the relationship between $Prob(T)$ and the potential of T used in the Splay tree algorithms?

Consider the following insertion scheme: INSERT(w): While inserting w as a new leaf, at each node v on the list of ancestors of w, flip a coin and mark v with probability $1/(size(v)+1)$. Rotate w up the tree to the highest node marked including a rotation with the highest marked node.

We would like to show that this algorithm generates random trees. That is, the probability that INSERT generates a tree is the same as the random insertion model. We shall do this by taking a tree T and computing the probability that INSERT generates this tree.

We start with a useful combinatorial fact. Let T be a tree of size n with root w and Left and Right subtrees T_L and T_R, respectively. Let $Join(T_L, T_R)$ be the set of all tree obtainable by combining the two trees, i.e., all tree obtained from T by rotating w to a leaf and deleting w. Thus, $Join(T_L, T_R)$ is precisely the maximal set of trees $\hat{T}'$ such that inserting w into $T' \in \hat{T}'$ using INSERT would return T, with some positive probability.

The probability of a distinct set of trees is just the sum of the probabilities of the individual trees.

Prove the following fact:

Problem 1.2 $Prob(T) \cdot n = Prob(T_L) \cdot Prob(T_R) = Prob(Join(T_L, T_R))$

Problem 1.3 *Uses the above two Problems to show that INSERT generates random trees.*

Hint: Let T
w, where w is not necessarily the root, be the set of all tree obtained from T by rotating w to a leaf a deleting w. Write formula which relates the probability of T to that of T
w.

Problem 1.4 *Give a similar algorithm for DELETION. Show that your algorithm generates random trees.*

2 Updates in splay trees

Obtain amortized bounds on the number of rotations done for the following operations on splay trees. Write your answers in terms of the sizes of the relevant trees. (For insertion and deletion, let n be the number of nodes in the tree before the operation. For join, let the numbers of nodes in these trees be n and m respectively.) Do not use asymptotic (big oh notation).

```
Insertion (type 1): Perform insertion in the usual way (the new node
                    has no children).  Splay the new node to the root.

Insertion (type 2): Find the node that would be the parent of the new
                    node in the usual insertion.  Call this node x.
                    Now splay x to the root of the tree.  Now add the
                    new node y above x, to create one of the following
                    trees:

                              y                       y
                             / \                     / \
                            x   B                   A   x
                           /                             \
                          A                               B

                     Where A and B are the subtrees of x after x is
                     splayed to the root.

Join two trees:      Let the two trees be A and B.  If A is empty
                     return B.  Otherwise find the leftmost node in A
                     (call it x) and splay it to the root.  Now attach
                     B as the left child of x.

Delete:              To delete a node, splay it to the root, then join
                     the two subtrees of that node to form a new tree.
```

Hint: You may find it easier to analysis type 2 insertions first.