HW 1

Course: CS280A Fall 2000

October 11, 2000

1 Exercises:

1. Show how two 2-3 trees T_1 and T_2 such that for all keys x, y if $x \in T_1$ and $y \in T_2$ then $x < y$ can be merged into a single 2-3 tree T in $O(logn)$ operations, where n is the sum of the keys in the two trees.

   Show how an interval of keys can be deleted from a 2-3 tree in $O(logn)$ operations.

2. Let $A[1..n]$ be an array of real numbers. Design algorithms to perform any sequence of the following two operations:

   $Add(i, y)$: Add the value of y to the ith number.

   $Partial\_sum(i)$: return the sum of the first i numbers $\sum_1{}^i A[i]$.

   Each operation should take $O(logn)$ steps. You can use one more array of size n as work space.

   Extend the above to support insertion and deletion. Each element now has a *key* and a *value*. An element is accessed by its key. The addition operation applies to the values (but the elements are accessed by their keys). The $Partial\_sum$ operation is different

   $Partial\_sum(y)$: return the sum of all the elements currently in the set whose value is less than y, $\sum_{x_i < y} x_i$.

   The worst-case running time should still be $O(nlogn)$ for any sequence of $O(n)$ operations.

3. The **Josephus problem** is defined as follows. Suppose that n people are arranged in a circle and that we are given a positive integer $m < n$. Beginning with a designated first person we proceed around the circle, removing every mth person. After each person is removed, counting continues around the circle that remains. This process continues until all the n people have been removed. The order in which the people are removed from the circle defines (n, m)**-Josephus permutation** of the integers $1, 2, ..., n$. Describe an $O(nlogn)$ algorithm that, given the integers n and m, outputs the (n, m)-Josephus permutation.

4. Consider n chords on a circle, each defined by its endpoints. Describe an $O(nlogn)$ algorithm for determining the number of pair of chords that intersect inside the circle.

5. Write an $O(n)$ nonrecursive procedure that, given an n-node binary tree, prints out the key of each node. Use no more than constant extra space outside the tree itself and do not modify the tree, even temporary during the procedure.