ICS 261 — Data Structures — Winter 1999 — Midterm

Name:

ID:

1:

2:

3:

4:

total:

1. (20 points)

Any binary search tree can be used as a priority queue: to find the minimum element, simply follow the left child pointers in the search tree until reaching the leftmost node.

Give a reason why one might prefer to use a splay tree as a priority queue, instead of using a Fibonacci heap for the same purpose.

2. (30 points)

Suppose we want to augment a red-black tree so that we can quickly find the kth largest value stored in the tree (where the number k is given as an argument to the query).

(a) What extra information would you need to store at each node of the tree in order to answer these queries efficiently?

(b) Describe how to update the information from your answer to part (a) when the tree undergoes a rotation. (This update should only take constant time per rotation.)

3. (30 points)

For each of the following priority queue data structures, state whether the node copying technique could be used to make the structure fully persistent, and explain why or why not. You may assume that the only operations that need to be made persistent are insert and delete-min.

(a) Binary heaps

(b) Binomial heaps

(c) Fibonacci heaps

4. (20 points)

Suppose we are given a data structure for maintaining a partition of n objects into disjoint sets S_i. Each operation removes at most one item from each set, and forms a new set out of these removed items. Whenever a set's last item is removed by an operation, the set is removed from the data structure. If an operation moves k items to a new set, it takes time $O(k)$.

(a) What is the worst case time per operation? (Your answer should be a function of n only, and should not depend on k.)

(b) Show that the amortized time per operation is $O(\sqrt{n})$, using the potential function

$$\Phi = n^{3/2} - \sum |S_i|^2/\sqrt{n}.$$

Hint: divide the change to Φ into two parts: the increase in Φ caused by removing an element from each set, and the decrease in Φ caused by forming a new k-element set. Show that the first part is $O(\sqrt{n})$, and that whenever the actual time is greater than $O(\sqrt{n})$ it is balanced by an equal or larger decrease in the second part.