CIS 313 Intermediate Data Structures
Winter 2010

Assignment 6

due Friday, March 5, 2010

1. Suppose you have k sorted lists containing a total of n elements. Show how to merge these lists into a single sorted list in $O(n\ lgk)$ time. (This is exercise 6.5-9, p 166 in 3rd edition and 6.5-8, p 142 in 2nd.) **[8 points]**

2. Let H be a max-heap storing n keys. Give an efficient algorithm for listing all the keys in H that are greater than or equal to a given value x (which may or may not be in H). The keys do not need to be listed in sorted order. Ideally, your algorithm will run in time $O(k)$, where k is the number of keys returned. **[6 points]**

3. A **min-max** heap H combines properties of both a min-heap and a max-heap. It is defined as a complete binary tree with the following property: let v be a node of H at depth i. Then

   - if i is odd, then the key stored in node v is less than or equal to any other value stored in the subtree of H rooted at v.
   - if i is even, then the key stored in node v is greater than or equal to any other value stored in the subtree of H rooted at v.

   (a) Illustrate a min-max heap by drawing one containing 25 elements.

   (b) Describe how to perform the following operations in a min-max heap, and give the running time of each.

   - findMin()
   - findMax()
   - insert(k)

   [8 points]

4. Exercise 19.2-1, p 518 (in 2nd edition this is 20.2-1, p 488) **[4 points]**

5. From the tree used in the previous problem (before the extractMin), decrease 35 to 6. **[4 points]**

6. Suppose you are given an array S of n elements, each of which is colored red or blue. Give an in-place linear-time ordering method which will list all the blue elements before the red elements. Can you extend this to handle three colors? **[6 points]**

7. Suppose you have an array S of size n, where each element in S represents a different vote for class president, where each vote is given as an integer representing the student ID of the candidate. Without making any assumptions about who is running or how many candidates there are, design an $O(n \lg n)$ algorithm to determine which candidate receives the most votes. **[6 points]**

8. Consider a modification to the previous problem to a situation where we know the number $k < n$ of candidates running. Design an $O(n \lg k)$ algorithm to determine which candidate receives the most votes. **[6 points]**

9. Suppose you have a set A of n nuts and a set B of n bolts, such that each nut in A has a unique matching bolt in B. The only kind of comparison you can make is to take a nut-bolt pair (a, b), where $a \in A$ and $b \in B$, and test to see whether the threads of a are larger, smaller, or a perfect match to the threads of b. Give an efficient algorithm to match up all the nut and bolts. What can you say about the run-time of your algorithm? **[8 points]**

Total: 56 points

Notes:

- *Q1:* You may want to use a heap of size k.
- *Q2:* Prune the search.
- *Q3(a):* Draw it as a tree *and* give the array representation. Pick your own values for the keys in the tree.
- *Q8:* You may want to not sort, but think of a BST like structure.
- *Q9:* Partition, partition, partition.