

**CS 610, Spring 2009, Prof. Calvin
Practice Midterm Questions**

1. Recall the Fibonacci sequence, defined by $F_1 = 1$, $F_2 = 2$, and $F_n = F_{n-1} + F_{n-2}$ for $n > 2$. Show by induction that $F_n = \mathcal{O}((7/4)^n)$.

2. Suppose that we use an AVL tree to implement a min-priority queue. Starting with an unordered array of n elements, describe the time needed to build the min-priority queue, the time to extract the minimum, and the time to add a new element to the queue.

3. We can construct a sorting algorithm using a min priority queue as follows: Insert the elements to be sorted into an initially empty priority queue, then repeatedly extract the minimum element until the priority queue is empty. What are the best-case and worst-case times to sort n numbers using this approach if the priority queue is implemented using:

a) an unsorted array?

b) a sorted array?

c) a heap?

4. A variation on the merge sort that we studied is to split a list of length n into k sublists, sort each sublist, and then merge the k sorted sublists into one sorted list. If $k = 2$, this is the merge sort that we discussed in class.

Suppose that you are given a set of k lists with combined length n . Each of the lists is sorted, and you want to process the lists to obtain one sorted list of length n . Describe an efficient algorithm that accomplishes this task and describe its running time.

5. Describe an efficient ordered dictionary structure for storing n elements that have an associated set of $k < n$ keys that come from a total order. Assume that k is much smaller than n . Your structure should perform all the ordered dictionary operations in $O(\lg(n) + s)$ time in the worst case, where s is the number of elements returned. Describe an alternative structure if we relax the worst case bound to $O(\lg(n) + s)$ time on average.

6.a) Let A and B be two sequences of n integers each. Describe an $O(n \lg(n))$ time algorithm for determining if A and B have an element in common.

b) Let A and B be two sequences of n integers each. Given an integer x , describe an $O(n \lg(n))$ time algorithm for determining if there is an integer $a \in A$ and an integer $b \in B$ such that $x = a + b$.

7. Bob has 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 . Unfortunately, the nuts in A all look the same, and the bolts in B all look the same as well. The only kind of comparison that Bob can make is to take a nut-bolt pair (a, b) , such that $a \in A$ and $b \in B$, and test it to see if the threads of a are larger, smaller, or a perfect match for the threads of b . Describe an efficient algorithm for Bob to match up all of his nuts and bolts. What is the running time of this algorithm, in terms of nut-bolt tests that Bob must make?

8. Either give an argument that the following statement is true, or else give an argument that it is false: “For any priority queue that operates using pairwise comparisons of its elements, a sequence of n operations takes time $\Omega(n \lg(n))$ in the worst case.”

9. Suppose that A is an array containing n distinct integers, A_1, A_2, \dots, A_n . Denote the order statistics by $A_{(1)} < A_{(2)} < \dots < A_{(n)}$. Suppose that we want to “approximately” sort A . Specifically, we want to rearrange A so that if $A_i = A_{(j)}$, then in the approximately sorted array $|i - j| \leq k$, for some fixed positive $k < n$. In other words, we want to rearrange A so that each element is no more than k locations away from where it should be in the sorted order.

Describe an efficient algorithm for approximately sorting A , and analyze its average running time if all orderings of A are equally likely.