

---

## Homework 4 – Due Friday, February 23, 2007

**Reminders** Collaboration is permitted, but you must write the solutions by yourself without assistance, and be ready to explain them orally to a member of the course staff if asked. You must also identify your collaborators. Getting solutions from outside sources such as the Web or students not enrolled in the class is strictly forbidden.

When you describe an algorithm, remember that clarity is the main goal. Pseudocode is preferred, but an English-language description is also acceptable if it is unambiguous (unless specified otherwise).

**Exercises** These should not be handed in, but the material they cover may appear on exams.

- Quicksort: CLRS Ex. 7.4-1, 7.4-2, 7.4-4, 7.4-5, Problem 7-3.
- Order statistics: CLRS Ex. 9.2-2, 9.2-4, 9.3-5, 9.3-6, 9.3-8
- Abstract data types: CLRS Ex.10.1-6, 10.1-7

### Problems to be handed in

1. (**Median-of-3 Partition**) Recall that in lecture 8 we said that a split generated by PARTITION is *OK* if both parts are of size at least  $\frac{n}{4}$ . Professor Betterchoice proposes to choose the pivot for partitioning more carefully than by picking a random element from the array. She suggests to choose the pivot as the median of a set of 3 elements randomly selected from the array. Assume that all elements in the array are distinct.
  - (a) (review) What is the probability of getting an OK split if the pivot is chosen at random? Explain.
  - (b) Roughly, what is the probability of getting an OK split with the new method? Explain.
  - (c) Let  $I$  be the indicator random variable for getting an OK split using the median-of-3 partition:

$$I = \begin{cases} 1 & \text{if the split is OK} \\ 0 & \text{otherwise} \end{cases}$$

What is the expectation of  $I$ ?

2. (**Largest  $i$  Numbers in Sorted Order**) CLRS Problem 9-1. Give pseudocode for all parts.
3. (**Queue Overflow**) CLRS Exercise 10.1-4

4. (**Glass Jars**) You are doing some stress-testing on various models of glass jars to determine the height from which they can be dropped and still not break. The set up for this experiment, on a particular type of jar, is as follows. You have ladder with  $n$  rungs, and you want to find the highest rung from which you can drop a copy of the jar and not have it break. We call this the *highest safe rung*.
- (a) You decide to try binary search: drop a jar from the middle rung, see if it breaks, and then recurse on the lower or higher half of the ladder, depending on the outcome. In the worse case, how many jars will you need to break to find the answer? How many jar drops do you need?
  - (b) You decided to conserve jars. Give a strategy for finding the highest safe rung by breaking only one jar. How many jar drops do you need in the worst case?
  - (c) Grudgingly, you decide that you can afford breaking 2 jars. Describe a strategy for finding the highest safe rung that requires  $f(n)$  jar drops where  $f(n) = o(n)$ .
  - (d)\* [(extra credit)] Now suppose you have a budget of  $k > 2$  jars, for some given  $k$ . Describe a strategy for finding the highest safe rung using at most  $k$  jars. If  $f_k(n)$  denotes the number of times you need to drop a jar according to your strategy, then the functions  $f_1, f_2, f_3, \dots$  should satisfy  $f_k(n) = o(f_{k-1}(n))$ .