

CSE 830: Homework #2

Due Oct. 1st 2009, 10:20am

Each problem should be solved on a separate sheet of paper to facilitate grading. Limit the solution of each section to one side of a sheet of paper. Please don't wait until the last minute to look at the problems.

1. Use the partitioning step of quicksort to give an algorithm that finds the *median* element of an unsorted array of n integers in expected $O(n)$ time.
2. Outline an efficient method of solving each of the following problems. Give a tight bound on the worst-case complexity of your methods. You may describe your algorithm in a common programming language, in pseudocode, or in English, but please be sure to be thorough.
 - a. Given an unsorted list of m stocks that you are investing in, and a sorted list of all n stocks available saying if their value has gone up or down today. Determine how many of your stocks have dropped.
 - b. You are given a list of N packages that UPS delivered in a given month. For each package, you are given an unsorted list of which of its M trucks it was transported on to reach its destination. Find the number of packages that were transported on each specific truck. (Note: this one can be a bit tricky if you consider that there may be many more packages than trucks.)
 - c. You have a huge stack of papers that you are carrying from one end of the company to another when a co-worker slams a door causing the first 34 pages to fly off the top. The helpful co-worker aides you in gathering them up, but to your horror you realize he is placing the ones he collected into multiple points in the middle of the once-sorted pile. How can you restore the papers to a sorted state?
3. Suppose that we are given a stack of n elements that we would like to sort, by returning a stack containing the records in sorted order (with the smallest value on top). We are allowed to use only the following operations to manipulate the data:
 - Pop-Push(s_1, s_2) - pop the top item from stack s_1 and push it onto stack s_2 .
 - Compare(s_1, s_2) - test whether the top element of stack s_1 is less than the top element of stack s_2 .
 - a. Give a $\Theta(n^2)$ sorting algorithm using just these operations and three stacks.
 - b. Give an initial stack that cannot be sorted just using these operations and two stacks.
 - c. Give a $\Theta(n^2)$ sorting algorithm using just these operations and two stacks, where you are allowed to temporarily park elements in a constant number of additional registers. (hint: use insertion sort)
4. Devise an algorithm for finding the k smallest elements of an unsorted set of n integers in $O(n + k \lg n)$ time. For partial credit, find an algorithm that takes $O(n + n \lg k)$ time instead.
5. Exercises: 6-2 (Analysis of d -ary heaps) and 8-4 (Water jugs)
6. (*extra credit*) Design a data structure that allows one to search, insert, and delete an integer X in $O(1)$ time (i.e., independent of the total number of integers stored). Assume that X is positive and less than or equal to n , and that there are $m + n$ units of memory available for the symbol table, where m is the maximum number of integers that can be in the table at any one time. (*Hint*: Use two arrays $A[1..n]$ and $B[1..m]$.) You are not allowed to initialize either A or B , as this would take $O(n)$ or $O(m)$ operations. This means that the arrays are full of random garbage to begin with, so you must be careful.