

CSci 4041, Spring 2007: Homework 1

Due electronically on Friday 2/2 at NOON.

1. [10 pts.]

In general, how many merges will be performed by merge sort in an n -element list? (For the sake of simplicity, assume that n is a power of 2)

2. [10 pts.] (a) Implement (in pseudocode) a data structure which has methods for stack operations (push, pop), as well as an extractMin() method, which removes and returns the smallest element. This structure should be based on the basic heap operations; for example, push is equivalent to insert and pop returns the most recently pushed objects. (b) Explain, in a few sentences, why the operations will run in time $O(\lg n)$.

3. [10 pts.]

(a) For insertion sort, heap sort, and quick sort, give an example of a list that produces the worst case runtime. (b) Which of these algorithms is the least stable in terms of run-time complexity? Give examples and a general explanation to support your choice.

4. [10 pts.]

Intuitively, you could run the insertion sort algorithm “backwards” to randomize a sorted list. Sticking as close to the INSERTION-SORT algorithm in section 2.1 (p. 17) as possible, (a) create pseudocode for the “insertion shuffle” algorithm, and (b) provide a brief inductive proof of correctness. Assume you are given the function rand(i,j) which returns an integer between i and j inclusive. Hint: After each iteration of the outer loop, the contents of the array should be similar to those of figure 2.2, starting at 2.2(f) and going backwards. (But since you are using rand, it will not be exactly the same).

5. [10 pts.]

The ordered pair (x,y) is considered a *swap* of the array A if $x < y$ and $A[x] > A[y]$.

(a) List all the swaps of the array $[3, 4, 7, 5, 1]$.

(b) Of the set of all arrays with values $[1, 2, \dots, n]$, which has the most swaps? How many does it have?

(c) Produce pseudocode for an algorithm based on merge sort which runs in $O(n \lg n)$ time and counts the number of swaps in an input list.

6. [10 pts.]

You are given an algorithm that takes as input an unsorted list, and tells you which of heap sort and merge sort will sort the list fastest, without actually sorting the list. You are considering using this before sorting the list so that you can sort the list most efficiently. What complexity would this algorithm need to have in order to be useful?

7. [10 pts.]

You are setting up a web application in which users can visualize (but not write) long lists of data that could be sorted on multiple keys (for example, a list of customers could be sorted by name, phone number, date of last purchase, etc.). Two possible setups are: 1) User downloads all the data once with a fast sorting algorithm (say $O(n \lg n)$), and 2) User downloads the data that has already been sorted by all possible keys (i.e. if there are m keys in each data element, the user essentially downloads the data m times). In general terms, discuss the tradeoffs between the two different approaches.