Top k in array Give an $O(n \log k)$ time algorithm to find the $k$ largest elements in an unsorted array $A[1...n]$. Be sure to give at least a basic correctness proof and time analysis. (25 points, 10 points correctness, 15 points efficiency.)

Top k in a heap Give the fastest algorithm you can to find the $k$ smallest elements in an $n$ element min-heap. My best is $O(k \log k)$. Be sure to give at least a basic correctness proof and time analysis. (25 points, 10 points correctness, 15 points efficiency.)

Modifying algorithms Say that, given an $n$-node graph $G$ with non-negative edge weights, a number $1 \leq k \leq n$, and a starting node $s$, you wish to list the $k$ elements closest in shortest path distance to $s$. Give an efficient algorithm for this problem. My best takes time $O(nk)$. (25 points, 10 points correctness, 15 points efficiency.)

Implementation-25 points Implement a naive $O(n^2)$ time sorting algorithm (such as bubble sort) and heap-sort. You can use heaps from a standard library to implement heap-sort. Plot their performance on random arrays of $n$ integers with values between 1 and $n$, for $n = 2^6, 2^8, 2^{10}, 2^{12}, 2^{14}, 2^{16}$. Plot their performance on a log-log scale. Is heap-sort always better than bubble-sort? Why or why not?