Problem 1.

Suppose that you are given the problem of returning in sorted order the $k$ smallest elements in an array of size $n$, where $k$ is much smaller than $\log(n)$, but much larger than 1.

a. Describe how selection sort, insertion sort, mergesort, and heapsort can be adapted to this problem. Your description need not give the pseudo-code for the modified algorithms; it is enough simply to describe what changes can be made, as long as your description is clear. You may use the recursive version of mergesort.

b. Find the worst-case running times of these algorithms as functions of $k$ and $n$.

Problem 2.

(Modified from Siegel 5.2)

a. Write a recursive procedure AddHeap(X,H,J) that does the following: H and J are minHeaps; you may assume for simplicity that they are implemented as explicitly linked structures. X is a new element. AddHeap destructively combines X, H, and J into a single minHeap.

b. Analyze the running time of the algorithm in (a).

Problem 3.

(CLR&S 7.5-6)

Give an $O(n \cdot \log(k))$ time algorithm to merge $k$ sorted lists into one sorted list, where $n$ is the total number of elements in all the input lists. (Hint: Use a heap for $k$-way merging.)

Problem 4.

(Siegel 5.24) Design an efficient algorithm to determine whether two unsorted sets of $m$ and $n$ integers are disjoint. Assume that $m < n$. Full credit will be given for an algorithm that runs in time $O(n \log m)$; half credit will be given for an algorithm that runs in time $O(n \log n + m \log m)$ (which is the same thing as $O(n \log n)$.)