G22.1170: Fundamental Algorithms I
Problem Set 3
(Due Thursday, April, 5 2007)

The problems in this problem set are about various sorting algorithms. Please consult Chapters 7 & 8 from the book (CLR).

**Problems from Cormen, Leiserson and Rivest:** (pp. 150–151 & 167)
7.5-4 & 7.5-5 *HeapIncreaseKey and Heapdelete*
8.4-4 *Improving the Running time of QuickSort*

**7.5-4** HeapIncreaseKey

Give an $O(\lg n)$-time implementation of the procedure HEAP-INCREASE-KEY($A$, $i$, $k$), which sets $A[i] \leftarrow \max(A[i], k)$ and updates the heap structure appropriately.

**7.5-5** Heapdelete

The operation HEAP-DELETE($A$, $i$) deletes the item in node $i$ from heap $A$. Give an implementation of HEAP-DELETE that runs in $O(\lg n)$ time for an $n$-element heap.

**8.4-4** Improving the Running time of QuickSort

The running time of quicksort can be improved in practice by taking advantage of the fast running time of insertion sort when its input is "nearly" sorted. When quicksort is called on a subarray with fewer than $k$ elements, let it simply return without sorting the subarray. After the top-level call to quicksort returns, run insertion sort on the entire array to finish the sorting process. Argue that this sorting algorithm runs in $O(nk+n\lg(n/k))$ expected time. How should $k$ be picked, both in theory and in practice?

**Problem 3.1** Let $S$ be a set whose elements are drawn from a linearly-ordered universe. If $|S| = n$ then the $\lceil n/2 \rceil^{\text{th}}$ smallest element of $S$ is the *median* of $S$. Design a data structure, called a *Median Heap*, that maintains the set $S$ and supports the following operations:

- Insert$(a, S)$: insert an element $a$ into the set $S$.
- DeleteMedian$(S)$: find the median of $S$ and delete it from $S$.

Your implementation may spend $O(\log n)$ time per each of these operations.

**Problem 3.2** Show that your implementation of the Median Heap is *optimal* in the following sense:

If $\sigma_1$, $\sigma_2$, ..., $\sigma_n$ is a sequence of INSERT and DELETEMEDIAN operations performed then the average complexity of these operations must be

$$T_{\text{avg}}(n) = \frac{\sum_{i=1}^{n} T(\sigma_i)}{n} = \Omega(\log n),$$

where $T(\sigma_i)$ is the time complexity of the operation $\sigma_i$.

**Problem 3.3** Let $S_1$, $S_2$, ..., $S_m$ be a set of sequences of elements, to be read in from $m$ input tapes in the nondecreasing order. $S = \text{MERGEALL}(S_1, S_2, \ldots, S_m)$ is defined to be the sequence consisting of the elements of $S_1$, $S_2$, ..., $S_m$, to be printed on an output tape in the nondecreasing order.

Sketch an algorithm to perform MERGEALL in time

$$O\left(\left(\sum_{i=1}^{m} n_i\right) \log m\right),$$

where $|S_i| = n_i$. Your algorithm must use $O(m)$ space of the Random Access Memory.