DUE: Wednesday, November 10

1. (8 points) Show that for any \( n \) and any \( k \) such that \( n \geq k \geq 1 \) and for any array \( A[1..n] \) of integers, the \( k^{th} \) largest element of \( A \) can be found in worst-case time in \( O(n + k \log n) \).

2. (20 points) Do Problem 17-2, parts \( a \) and \( b \), on pg. 426 of our text.
CS524
HW#4 SOLUTIONS

1. \textbf{BUILDMAX-HEAP}(A)
\textbf{for} i \leftarrow 1 \textbf{to} k-1 \textbf{do} \textbf{MAX-HEAPIFY}(A,1)
\textbf{return} A[1]

2. \textit{a} Let $k = \lceil \lg (n+1) \rceil$, and there are $O(\lg n)$ arrays. Searching each $A_i$ using \textsc{BinarySearch}, the worst-case search time is $(k-1) + (k-2) + \ldots + 2 + 1 \in O(\lg^2 n)$.

\textit{b} To \textsc{insert} a new element $x$ into this data structure, we make a new (sorted) array $A'_i$ containing only $x$. We then \textsc{merge} this new data structure with the old structure consisting of sorted arrays, analogous to the operation on binomial heaps. To merge two sorted arrays into one sorted array, we use the \textsc{merge} procedure on pg. 29 of our text, which operates in linear time. In the worst case, the data structure contains arrays $A_0, \ldots, A_{k-1}$, and the worst-case execution time is $\Theta(n)$. In general, each element which is \textsc{inserted} into the data structure, gets \textsc{merged} and costs an operation each time it moves from an $A_i$ to an $A_{i+1}$. Thus, it moves $O(k)$ or $O(\lg n)$ times. Thus, when an element is \textsc{inserted}, $\$k$ is spent, $\$1$ to pay for the \textsc{insertion}, and $\$(k-1)$ to pay for future \textsc{merges}. 