CS524
HW#4

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 kth largest element of A can be found in worst-case time in $O(n + k \lg n)$.

2. (20 points) Do Problem *17-2*, parts ***a*** and ***b***, on pg. 426 of our text.

CS524
HW#4 SOLUTIONS

1.	BUILD-MAX-HEAP(A)	$O(n)$
	for $i \leftarrow 1$ **to** k-1 **do** MAX-HEAPIFY(A,1)	$O(k\lg n)$
	return $A[1]$	$O(1)$

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

b To INSERT a new element x into this data structure, we make a new (sorted) array A_0^* containing only x. We then 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 MERGE procedure on pg. 29 of our text, which operates in linear time. In the worst case, the data structure contains arrays $A_0,...,A_{k-1}$, and the worst-case execution time is $\Theta(n)$. In general, each element which is INSERTed into the data structure, gets 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 INSERTed, \$$k$ is spent, \$1 to pay for the INSERTion, and \$(k-1) to pay for future MERGEs.