Problem 6-1 (Fast Priority Queue... Too Fast)  8 points

You receive a sales call from a new start-up called MYPD (which stands for “Manage Your Priorities... Differently”). The MYPD agent tells you that they just developed a ground-breaking comparison-based priority queue. This queue implements \textit{Insert} in time $\log_2(\sqrt{n})$ and \textit{Extract}_\text{max} in time $\sqrt{\log_2 n}$. Explain to the agent that the company can soon be sued by its competitors because either (1) the queue is not comparison-based; or (2) the queue implementation is not correct; or (3) the running time they claim cannot be so good. To put differently, no such comparison-based priority queue can exist.

(Hint: Be very precise with the constant $c$ s.t. $\log_2(n!) \approx cn \log n$.)

Problem 6-2 (Median)  12 points

You are given two sorted arrays $A[1, \ldots, n]$ and $B[1, \ldots, n]$, and you want to find a median of all the $2n$ numbers (i.e., the $n$-th ranked element of the “merge” of $A$ and $B$). Assume $n = 2^k$ and ignore floors and ceilings.

(a) (3 points) Assume you compare $A[{n \over 2}]$ and $B[{n \over 2}]$ of $A$ and $B$ (Please ignore floors and ceiling; say, assume $n$ is a power of 2.) and, say, $A[{n \over 2}] \leq B[{n \over 2}]$. Argue that $\text{Median}(A[1, \ldots, n], B[1, \ldots, n]) = \text{Median}(A[{n \over 2}+1, \ldots, n], B[1, \ldots, {n \over 2}])$.

(b) (6 points) Design an $O(\log n)$ divide-and-conquer algorithm to compute the median of all $2n$ numbers. Do not forget the base case $n = 1$.

Write the exact recurrence equation for the number of comparisons made by your algorithm and solve it exactly (no $O()$-notation). (Hint: Write a recursive procedure $\text{Find-Median}(A, a, B, b, i)$, which finds the median of $A[a, \ldots, a+i-1] \cup B[b, \ldots, b+i-1]$, with the initial call $\text{Find-Median}(A, 1, B, 1, n)$.)

(c) (3 points) Notice that your algorithm is a comparison-based algorithm. This means that for every $n$, its execution can be viewed as a decision-tree, where the internal nodes are the comparisons “$A[\ldots] < B[\ldots]$?” made, tree edges are labeled “yes” and “no”, and the leaves are labeled by the tuple (array-name, index) representing whether the $n$-th largest element comes from $A$ or $B$, and what its index is. Now consider \textit{any} comparison-based algorithm $Z$, which (correctly) computes the $n$-th element of $A \cup B$. How many distinct leaves (i.e., correct answers) should such a decision tree have?

Number of leaves is at least .....................
Use the answer above to show that the running time of any comparison-based algorithm for the problem is at least (give precise answer, without \(O\))

\(\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\ qua
(c) Based on part (b), design an algorithm to help the fingerprint expert find a strategy to find the subset of more than half identical fingerprints, where the number of comparisons is only $O(n)$. (Hint: Try to use the idea of part (b) to reduce the problem into the same kind of problem, but on at most $n/2$ fingerprints, by cleverly "dropping" at least one (and sometimes both!) fingerprint(s) in each pair from part (b).)

(d) (3 points) Write the recurrence equation for the running time of your algorithm and analyze it.

**Problem 6-5 (Free HW Points) 8 points**

Illustrate the operation of Radix-Sort on the following English words: BEAR, FEAR, CLAN, MASK, CLAP, MAKE, RAKE, FAKE, BORE, HAIR, WORD, EXAM, BALD, BOLD, MARK, SLAM.