# 15-451/651 Algorithm Design & Analysis, Spring 2024

# Extra Review Problems

## Lower Bounds

1. **(Multiple choice)**

   (i) Regarding lower bounds for problems in the comparison model, which of the following statements are true and which are false?

   (a) $\exists M$ distinct inputs $\implies$ lower bound of $\lg M$

   (b) $\exists M$ distinct inputs that require distinct outputs $\implies$ lower bound of $\lg M$

   (c) $\exists M$ distinct outputs, each required by some input $\implies$ lower bound of $\lg M$

   (d) $\exists M$ distinct outputs, each possible for some input $\implies$ lower bound of $\lg M$

   (ii) Out of the following choices, what is the largest lower bound on the number of comparisons required to find the second largest element in an array?

   (a) $n + \lg n - 2$

   (b) $n$

   (c) $n - 1$

   (d) $2n - 3$

   (e) $n \lg n$

2. **(Deciding the maximum)** In lecture we used an adversary argument to obtain a lower bound of $n-1$ comparisons for finding the maximum element of a list.

   (a) Suppose we instead used an information-theoretic argument. What lower bound would we obtain for the problem? Is it useful?

   (b) Suppose we instead use decision trees to analyze the problem

   i. How many outputs does the root node of any decision tree for this problem contain?

   ii. How many outputs does each internal node of any decision tree for this problem contain?

   iii. Are the leaves of these decision trees unique?

   iv. How many nodes do each of the possible decision tree have?

   v. What lower bound is implied by the structure of these decision trees?

3. **(Lower bounds via reductions)** In 15-251, you learned how to use *reductions* to show that one problem is just "as hard" as another problem. 251 taught you how to show that a problem is loosely in the same complexity class as another (e.g., NP-hard). This technique is not only applicable to loose complexity classes, but can also be used to establish tight lower bounds!

   Lets do an example. Recall the partitioning algorithm used as a subroutine of Quicksort, which, given an array $A$ and some "pivot" element $A[i]$ of the array, rearranges the elements of $A$ so that all elements that are $\leq A[i]$ come before all elements that are $> A[i]$. What if we generalize this problem to pick $k$ distinct pivot elements, $A[i_1], A[i_2], ..., A[i_k]$? (The pivots do not have to be given in sorted order). The problem then becomes to rearrange the array so that all elements that are at most as large as the smallest pivot come first, followed by those that are larger but at most as large as the second-smallest pivot and so on...

   (a) Give an algorithm that solves $k$-partitioning in $\Theta(n \log k)$ comparisons.

   (b) Show that there can not possibly exist an algorithm that performs $k$-partitioning in fewer than $\Theta(n \log k)$ comparisons. To do so, show that if you assume such an algorithm exists, then it must be possible to sort in fewer than $\Theta(n \log n)$ comparisons.

4. **(Breaking Eggs)** Say I choose a number between 1 and $N$. You want to guess it in as few questions as possible (each time you make an incorrect guess, I'll tell you if it is too high or too low). As we know, the strategy that minimizes the worst-case number of guesses is to do binary search: it takes $\lceil \log_2 N \rceil$ guesses.

   But, what if you are only allowed **one** guess that is too high? Can you still solve the problem in $o(N)$ guesses? [If you were not allowed **any** guesses that are too high, the only option you would have would be to guess 1,2,3,... in order].

   (a) Show how to figure out how many centimeters high you can drop an egg without it breaking, using $2\sqrt{N}$ guesses, when you only have two eggs.

   (b) Can you show an $\Omega(\sqrt{N})$ lower bound for any deterministic algorithm?