

---

## Problem Set 2 Solutions

**Problem 1.** (a) As in the class, we can construct an adversarial input as follows.

When replying to algorithm's queries on leaves, we always reply such that for any tree, if not all leaves were queried, then the value of the tree is not yet determined. In this case, for each internal node, the algorithm is forced to evaluate each of its children.

For a node  $x$ , call  $val(x)$  the value of  $x$ .

We do this as follows. Suppose the algorithm queries leaf  $x$ , and let  $n_1, n_2, \dots, n_{h-1}, n_h = x$  be the path from root ( $n_1$ ) to  $x$  in the tree. Let  $n_i$  be the lowest node such that there exists a non-examined leaf in  $n_i$ 's subtree ( $x$  being counted as "examined"). Then, if  $n_i$  has 0 children with established value ( $n_{i+1}$  begin considered with non-established value), then we choose  $val(x) = 0$  (in which case,  $val(n_{i+1}) = val(n_{i+2}) = \dots = val(n_h) = 0$ ). If  $n_i$  has 1 child with established value ( $n_{i+1}$  begin considered with non-established value), then we choose  $val(x) = 1$  (in which case,  $val(n_{i+1}) = val(n_{i+2}) = \dots = val(n_h) = 1$ ). Note that the value of the first child of  $n_i$  is 0, and therefore the value of  $n_i$  remains un-established. Also, note that by the definition of  $n_i$ ,  $n_i$  cannot have 2 children with established value ( $n_{i+1}$  being considered with non-established value).

Thus, for each node, the algorithm has to establish the value of all of its 3 children – which means that the algorithm has to query all  $3^h$  leaves.

(b) Consider a node  $x$  in the tree. Let its value be  $val(x)$ . This means that there are two children  $child_1(x)$  and  $child_2(x)$  such that  $val(x) = val(child_1(x)) = val(child_2(x))$ . If, for each  $x$ , we know that these children  $child_1(x)$  and  $child_2(x)$ , then we can only compute the values of these children to compute the value of  $x$ . Thus, a non-deterministic algorithm can just guess  $child_1(x)$  and  $child_2(x)$  and recurse on those 2. If  $val(child_1(x)) = val(child_2(x))$  then we know for sure that  $val(x) = val(child_1(x)) = val(child_2(x))$ . If  $val(child_1(x)) \neq val(child_2(x))$  then we fail on that branch. At least one of the branches of the non-deterministic algorithm will succeed and give the right answer.

The number of leaves queried (in each branch) is given by the recurrence formula:  $T(h) = 2T(h-1)$  (where  $T(0) = 1$ ) and it solves to  $2^h$ , which is  $2^{\log_3 n} = n^{\log_3 2}$ .

(c) Consider a node  $x$  at level  $t$ .  $x$  has at least two children that agree on their values (e.g.,  $child_1(x)$  and  $child_2(x)$ ). If the third child has a different value, then: there is a probability of  $1/3$  that the randomized algorithm will compute the value of only  $child_1(x)$  and  $child_2(x)$ ; and probability of  $2/3$  that the algorithm will compute the value of all three children. If the third child has the same value

with the other 2 children, then the algorithm will compute the value of only 2 of its children.

Thus, if we call  $T(t)$  the number of leaves queried for a node at height  $t$ , then: either  $E[T(t)] = 1/3 \cdot 2E[T(t-1)] + 2/3 \cdot 3E[T(t-1)] = 8/3 \cdot E[T(t-1)]$  (in the first case) or  $E[T(t)] = 2E[T(t-1)]$  (in the second case). Thus,  $E[T(t)] \leq \frac{8}{3}E[T(t-1)]$ . With  $T(0) = 1$ , this solves to  $E[T(t)] = (\frac{8}{3})^t$ .

Concluding, we have that  $E[T(h)] \leq (\frac{8}{3})^{\log_3 n} = n^{\log_3 8/3} \leq n^{.8928}$ .

**Problem 2.** The lower bound is of course  $\Omega(n \log n)$ .

By Yao's minimax principle, it is enough to exhibit a distribution over inputs and prove that any deterministic algorithm will have, in expectation, a runtime of  $\Omega(n \log n)$ .

Coming up with the distribution is the "hardest" part: let's take uniform distribution over inputs. Note that, in the comparison model, an input is a permutation of the ranks of the elements (consider all elements are different). Thus, we have in total  $n!$  different inputs (and outputs), each drawn with probability  $1/n!$ .

Now consider any deterministic sorting algorithm. As in CLR, we can view the algorithm as a binary tree where in each node the algorithm makes a comparison; and, based on the result, branches to left or to right. Clearly, if there are  $n!$  different answers (equivalently inputs), then there should be  $n!$  leaves in the decision tree.

Now we want to compute the expected runtime of this algorithm on uniform distribution over all inputs. In other words, this is exactly the *average* height of the leaves of the decision tree.<sup>1</sup> Let  $h_i$  be the depth of the leaf corresponding to input  $i$  (for  $i = 1 \dots n!$ ). Then, the expected running time is  $\mu = E[h_i]$  (where the expectation is over  $i$ 's). We want to lower-bound the value  $\mu$ .

By Markov, we have that  $\Pr_i(h_i \geq 2\mu) \leq 1/2$ . This means that for at least  $n!/2$  of  $i$ 's (leaves), the depth is  $h_i \leq 2\mu$ . Since all these leaves are leaves in the decision tree, which is a binary tree, we can conclude that there is a binary tree of depth  $\leq 2\mu$  that contains at least  $n!/2$  leaves. However, we know that a binary tree on  $n!/2$  leaves must be of at least depth  $\Omega(\log(n!/2)) = \Omega(n \log n)$ . Therefore  $2\mu \geq \Omega(n \log n)$ , which implies that  $\mu = \Omega(n \log n)$ . Our initial claim is proven.

**Problem 3.** (a) Consider a subsequence of coin flips of length  $\log_2 n + c$ . Such a subsequence has probability of exactly  $(\frac{1}{2})^{\log_2 n + c} = n^{-1}2^{-c}$  that all the coins flips in the considered subsequence are heads. There are in total  $n - \log_2 n - c + 1$  such subsequences in the sequence of length  $n$ , and, therefore, by union bound, there is a probability of  $\leq (n - \log_2 n - c + 1) \cdot n^{-1}2^{-c} \leq n \cdot n^{-1}2^{-c} = 2^{-c}$  that there exists a subsequence of length  $\log_2 n + c$  of heads.

<sup>1</sup>There are numerous ways to show this, including a greedy exchange argument, using the convexity of log, etc. Above analysis is just one example.

(b) Let  $l = \log_2 n - a \log_2 \log_2 n - c$ , where  $a, c$  are constants. Divide the entire sequence into  $n/l$  subsequences of length  $l$ . One such subsequence has a probability  $2^{-l}$  of having all coins heads. Therefore, the probability  $P$  that none of the  $n/l$  subsequences is all heads is  $P \leq (1 - 2^{-l})^{n/l}$ . Let's compute this probability:  $P \leq (1 - 2^{-l})^{n/l} = (1 - \frac{2^c \log_2^a n}{n})^{n/l} \leq (1 - \frac{2^c \log_2^a n}{n})^{n/\log_2 n} \leq \exp[-\frac{2^c \log_2^a n}{\log_2 n}] \leq \exp[-2^c \log_2 n] = n^{-2^c \log_2 e}$  for  $a = 2$ . Since we can choose the constant  $c$  as big as we want, we achieve that whp  $(1 - n^{-\Omega(1)})$ , there will exist a subsequence of heads of length  $l$ .

**Problem 4.** Recall that we are choosing  $s = n^{3/4}$  random samples and taking as “boundaries” the samples  $a$  and  $b$  indexed at  $s/2 - \sqrt{n}$  and  $s/2 + \sqrt{n}$  respectively in the sample. We want to count the number of items between  $a$  and  $b$ . We will instead bound the number of items less than  $a$  and, equivalently, the number greater than  $b$ , then subtract those quantities to find the number in between.

Without loss of generality we can assume that the  $n$  input items are  $1, \dots, n$ . We first show that  $a$  is likely to be at least  $\frac{n}{2} - 2n^{3/4}$ . For this to not be the case, with our  $s$  independent trials, we would have to draw at least  $\frac{s}{2} - \sqrt{n} = \frac{n^{3/4}}{2} - \sqrt{n}$  values less than  $\frac{n}{2} - 2n^{3/4}$ .

Let  $X$  be the number of items we have drawn which are less than  $\frac{n}{2} - 2n^{3/4}$ . The expected value of  $X$  is  $sp = n^{3/4}(\frac{1}{2} - 2n^{-1/4}) = \frac{n^{3/4}}{2} - 2\sqrt{n}$ . The variance of  $X$  is  $sp(1-p) \leq \frac{n^{3/4}}{4}$ , giving a standard deviation of  $\leq \frac{n^{3/8}}{2}$ . Applying Chebyshev's inequality,

$$\Pr\left(\left|X - \frac{n^{3/4}}{2} + 2\sqrt{n}\right| \geq \sqrt{n}\right) \leq \Pr\left(\left|X - \frac{n^{3/4}}{2} + 2\sqrt{n}\right| \geq 2n^{1/8}\sigma_X\right) \leq \frac{1}{4}n^{-1/4}.$$

This implies that we draw at least  $\frac{s}{2} - \sqrt{n} = \frac{n^{3/4}}{2} - \sqrt{n}$  values less than  $\frac{n}{2} - 2n^{3/4}$  (and thus,  $a$  is less than  $\frac{n}{2} - 2n^{3/4}$ ) with probability at most  $\frac{n^{-1/4}}{4}$ . Through a symmetric analysis, we can see that  $b$  is greater than  $\frac{n}{2} + 2n^{3/4}$  with probability at most  $\frac{n^{-1/4}}{4}$ . Thus, by the union bound, there are at most  $4n^{3/4}$  items between  $a$  and  $b$  with probability at least  $1 - O(n^{-1/4})$ .

**Problem 5.** Let  $X = \frac{X_1 + X_2 + \dots + X_n}{n}$ . Suppose for every  $i$ , it holds that  $\text{Var}(X_i) \leq \sigma^2$ .

Thus,  $E[X] = \mu$  and  $\text{Var}(X) = \frac{\text{Var}(X_1) + \dots + \text{Var}(X_n)}{n^2} \leq \frac{n\sigma^2}{n^2} = \frac{\sigma^2}{n}$ . Thus by the Chebyshev's Inequality,

$$\Pr(|X - \mu| > \epsilon) \leq \frac{\sigma^2}{n\epsilon^2},$$

from which our result follows by taking  $n \rightarrow \infty$ .