CS 161 (Stanford, Fall 2025) Section 3

1 Expectation

1. *True or False:* Expected runtime averages the runtime over the outcomes of random events within the algorithm for a random input.
2. I have an algorithm that takes positive integers (n, i) where $1 \leq i \leq n$. The algorithm rolls a n-sided die repeatedly until the die returns any value $\leq i$. What is the expected runtime in n? Worst-case runtime? Rigorous proof not necessary :)

2 Light Bulbs and Sockets

You are given a collection of n differently sized light bulbs that have to be fit into n sockets in a dark room. You are guaranteed that there is exactly one appropriately-sized socket for each light bulb and vice versa; however, there is no way to compare two bulbs together or two sockets together as you are in the dark and can barely see! You can try and fit a light bulb into a chosen socket, from which you can determine whether the light bulb's base is too large, too small, or is an exact fit for the socket.

Suggest a (possibly randomized) algorithm to match each light bulb to its matching socket. Your algorithm should run strictly faster than quadratic time in expectation. Give an upper bound on the worst-case runtime, then prove your algorithm's correctness and expected runtime.

3 Batch Statistics

Design an algorithm which takes as input array A consisting of n possibly very large integers as well as an array R that contains k ranks $r_0, ..., r_k$, which are integers in the range $\{1, ..., n\}$. (You may assume that $k < n$.) The algorithm should output an array B which contains the r_j-th smallest of the n integers, for every j in $1, ..., k$. So if an $r_j = 3$ in input array R, then we want to return the 3rd smallest element in the input array A as part of the output.

Input: A which is an unsorted array of n unbounded distinct integers; R which is an unsorted array of k distinct ranks.

Example:

- Input: $A = [11, 19, 13, 14, 16, 18, 17, 12, 15]$; R = [3, 7]
- Output: $[17, 13]$

- Explanation: 17 is the 7th smallest element of A and 13 is the 3rd smallest of A. [13, 17] is also an acceptable output.

Hint: we are looking for an $O(nlogk)$ runtime algorithm.

4 Sorting with Low Adaptivity

Sometimes, the steps of an algorithm don't depend on one another - this happens frequently in real world settings. When we have such an "independence" between steps, we can use parallelization to speed up the algorithm! (Outside this question we won't spend much time discussing this issue in this class.)

1. We say that a comparison-based sorting algorithm is *non-adaptive* if it commits in advance to the pairs of elements that it will compare. In other words, when choosing a comparison at step k, the algorithm does NOT rely on information from previous comparisons at steps $(1, ..., k-1)$. Prove that any non-adaptive sorting algorithm requires $\Omega(n^2)$ comparisons.

2. Now we define a "stage" of operations as an arbitrary-sized set of operations that all occur simultaneously. We say that a comparison-based sorting algorithm has *adaptivity* t if it can be executed in $t+1$ stages, where the pairs to be compared in the i-th stage only depend on outcome of comparisons in stages $1, \ldots, i-1$ (but not on other comparisons in the i-th stage). For example, non-adaptive algorithms have 0 adaptivity.

   What is the adaptivity of the MergeSort algorithm?

3. Give a (possibly randomized) algorithm with worst-case adaptivity 1 (aka 2 stages) and expected number of comparisons $n^{3/2}$.

   Hint - you may find the following derivations useful:

   (1) Suppose $k + \ell \leq n$. If I sample two sets of size k, ℓ at random (it's enough that one of them is random!) out of n elements, the probability that they do not intersect is given by:

$$\begin{aligned}
\Pr[\text{sets don't intersect}] &= \frac{\#\text{ of ways of picking } k \text{ out of } n-\ell}{\#\text{ of ways of picking } k \text{ out of } n} \\
&= \frac{\binom{n-\ell}{k}}{\binom{n}{k}} \\
&= \frac{(n-\ell)!(n-k)!}{n!(n-k-\ell)!} \\
&\leq \left(\frac{n-\ell}{n}\right)^k \\
&\leq e^{-k\ell/n}.
\end{aligned}$$

(2) If I sum the above expression for many k's, I have:

$$\sum_{k=1}^{k\leq b} e^{-k\ell/n} < \sum_{k=0}^{\infty} e^{-k\ell/n} = \frac{1}{1-e^{-\ell/n}} = O(n/\ell).$$