CS 170 Homework 2

Due **Friday 9/13/204, at 10:00 pm (grace period until 11:59pm)**

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, explicitly write "none".

2 Quantiles

Let A be an array of length n. The boundaries for the k quantiles of A are

$$\{a^{(n/k)}, a^{(2n/k)}, \ldots, a^{((k-1)n/k)}\}$$

where $a^{(\ell)}$ is the ℓ-th smallest element in A.

Devise an algorithm to compute the boundaries of the k quantiles in time $\mathcal{O}(n \log k)$. For convenience, you may assume that k is a power of 2.

Hint: With careful choice of pivot, SELECT*(A, ℓ) gives $a^{(\ell)}$ in guaranteed $\mathcal{O}(n)$ time. We will see this in the next question.*

3 Median of Medians

The QuickSelect(A, k) algorithm for finding the kth smallest element in an unsorted array A picks an arbitrary pivot, then partitions the array into three pieces: the elements less than the pivot, the elements equal to the pivot, and the elements that are greater than the pivot. It is then recursively called on the piece of the array that still contains the kth smallest element.

(a) Consider the array $A = [1, 2, \ldots, n]$ shuffled into some arbitrary order. What is the worst-case runtime of QuickSelect(A, $n/2$) in terms of n? Construct a sequence of 'bad' pivot choices that achieves this worst-case runtime.

The above 'worst case' has a chance of occurring even with randomly-chosen pivots, so the worst-case time for QuickSelect is $\mathcal{O}(n^2)$, even though it achieves $\Theta(n)$ on average.

Based on QuickSelect, let's define a new algorithm DeterministicSelect that deterministically picks a consistently good pivot every time. This pivot-selection strategy is called 'Median of Medians', so that the worst-case runtime of DeterministicSelect(A, k) is $\mathcal{O}(n)$.

Median of Medians

1. Group the array into $\lfloor n/5 \rfloor$ groups of 5 elements each (ignore any leftover elements)
2. Find the median of each group of 5 elements (as each group has a constant 5 elements, finding each individual median is $\mathcal{O}(1)$)
3. Create a new array with only the $\lfloor n/5 \rfloor$ medians, and find the true median of this array using DeterministicSelect.
4. Return this median as the chosen pivot

(b) Let p be the pivot chosen by DeterministicSelect on A. Show that at least $3n/10$ elements in A are less than or equal to p, and that at least $3n/10$ elements are greater than or equal p.

(c) Show that the worst-case runtime of DeterministicSelect(A, k) using the 'Median of Medians' strategy is $\mathcal{O}(n)$.

Hint: Using the Master theorem will likely not work here. Find a recurrence relation for $T(n)$, then show that $T(n) \leq c \cdot n$ for some sufficiently large $c > 0$. You can assume $T(n) \leq c \cdot n$ for small values of n without proof.

4 The Resistance

We are playing a variant of The Resistance, a board game where there are n players, s of which are spies. In this variant, in every round, we choose a subset of players to go on a mission. A mission succeeds if the subset of the players does not contain a spy, but fails if at least one spy goes on the mission. After a mission completes, we only know its outcome and not which of the players on the mission were spies.

Come up with a strategy that identifies all the spies in $O(s \log(n/s))$ missions. **Describe your strategy and analyze the number of missions needed.**

Hint 1: consider evenly splitting the n players into x disjoint groups (containing $\approx n/x$ players each), and send each group on a mission. At most how many of these x missions can fail? What should you set x to be to ensure that you can reduce your problem by a factor of at least $1/2$?

Hint 2: it may help to try a small example like $n = 8$ and $s = 2$ by hand.

5 Poker

You are playing poker with n other friends, who either always tell the truth or sometimes bluff (lie). You do not know who may bluff and who will always tell the truth, but all your friends do. All you know is that there are more people who always tell the truth than people who bluff.

Your goal is to identify one specific player who always tells the truth.

You are allowed to perform a 'query' operation as follows: you pick two players as partners. You ask each player if their partner bluffs or always tells the truth. When you do this, players who tell the truth will tell the truth about the identity of their partner, but a player who bluffs can either lie or tell the truth about their partner.

Your algorithm should work regardless of whether bluffing players lie or tell the truth.

(a) For a given player x, devise an algorithm that returns whether or not x is always telling the truth using $O(n)$ queries. Just an informal description of your test and a brief explanation of why it works is needed.

(b) Show how to find a player who always tells the truth in $O(n \log n)$ queries (where one query is taking two players x and y and asking x to identify y and y to identify x).

Hint: Split the players into two groups, recurse on each group, and use part (a). What invariant must hold for at least one of the two groups?

Give a 3-part solution.

(c) (**Optional, not for credit**) Can you give a $O(n)$ query algorithm?

Hint: Don't be afraid to sometimes 'throw away' a pair of players once you've asked them to identify their partners.

Give a 3-part solution.