CS 170 Homework 2

Due **Saturday 2/8/2025, at 10:00 pm (grace period until 11:59pm)**

1 Study Group (1 point)

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

2 Median of Medians (12 points)

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) (2 points) Let us see an example of QuickSelect in action. Suppose you always pick the first element as the pivot. Compute QuickSelect(A, 6) for the following array:

$A =$

12	78	13	1	97	45	48	26	85	100	78

List the all the subarrays that quickselect recurses on, as well as the index k – the k smallest element of the current subarray which the algorithm returns.

For example, quickselect begins on the entire array [12, 78, 13, 1, 97, 45, 48, 26, 85, 100, 78] and $k = 6$.

(b) (2 points) Consider the array

$A =$

1	2	3	...	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

(c) (3 points) 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.

(d) (5 points) In this problem, we will show that the worst-case runtime of DeterministicSelect(A, k) using the 'Median of Medians' strategy is $\mathcal{O}(n)$.

(i) Find a recurrence relation for the time complexity of the algorithm, $T(n)$.

(ii) Use the recurrence relation to show that, for some sufficently large $c \geq 0$, the inequality $T(n) \leq c \cdot n$ always holds.

Hint: Using the Master theorem will likely not work here.

3 The Resistance (12 points)

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.

(a) (2 points) If there is one spy ($s = 1$), come up with a strategy that identifies the spy in $O(\log(n))$ missions. You do not need to prove that your strategy works.

(b) (2 points) In the general case, when there are s spies, consider evenly splitting the n players into x disjoint groups (containing $\approx n/x$ players each), and send each group on a mission. At least how many of these x missions must succeed, in terms of x and s?

(c) (8 points) Come up with a strategy that identifies all the spies in $O(s \log(n/s))$ missions.

Hint: What should you set x to be in part (b) to ensure that you can reduce your problem by a factor of at least $1/2$?

(i) Describe your strategy.

(ii) Prove that your strategy works.

(iii) Analyze the number of missions your strategy requires in the worst case.

4 Poker (13 points)

You are playing poker with n other friends. Some of your friends are "truthers," meaning that they always tell the truth, while everyone else is a "bluffer," meaning that they sometimes tell the truth but also sometimes bluff (lie). You do not know who is a truther or a bluffer, 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 that is a truther.

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

Your strategy should work regardless of whether bluffers lie or tell the truth.

(a) (3 points) For a given player x, devise a strategy that returns whether or not x is a truther using $O(n)$ queries. Just an informal description of your test and a brief explanation of why it works is needed.

(b) (10 points) Devise a divide and conquer algorithm that finds a truther 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 property must hold for at least one of the two groups?*

  (i) Describe your strategy.

  (ii) Prove that your strategy works.

  (iii) Analyze the number of queries that your strategy requires in the worst case.

(c) (**Extra credit**, 3 points) Can you give a $O(n)$ query algorithm? Describe your strategy and explain why it works.

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

5 Coding Question: Closest Pair (9 points)

For this weeks coding questions, you'll solve the Closest Pair problem by implementing the divide-and-conquer algorithm from lecture. There are two ways that you can access the notebook and complete the problems:

1. **On Datahub**: click here and navigate to the `hw02` folder.
2. **On Local Machine**: `git clone` (or if you already cloned it, `git pull`) from the coding homework repo,

   `https://github.com/Berkeley-CS170/cs170-sp25-coding`

   and navigate to the `hw02` folder. Refer to the `README.md` for local setup instructions.

Notes:

- *Submission Instructions:* Please download your completed submission `.zip` file and submit it to the Gradescope assignment titled "Homework 2 Coding Portion".
- *Getting Help:* Conceptual questions are always welcome on edstem and office hours; *note that support for debugging help during OH will be limited.* If you need debugging help first try asking on the public edstem threads. To ensure others can help you, make sure to:
  1. Describe the steps you've taken to debug the issue prior to posting on Ed.
  2. Describe the specific error you're running into.
  3. Include a few small but nontrivial test cases, alongside both the output you expected to receive and your function's actual output.

  If staff tells you to make a private Ed post, make sure to include *all of the above items* plus your full function implementation. If you don't provide them, we will ask you to provide them.
- *Academic Honesty Guideline:* We realize that code for some of the algorithms we ask you to implement may be readily available online, but we strongly encourage you to not directly copy code from these sources. Instead, try to refer to the resources mentioned in the notebook and come up with code yourself. That being said, we **do acknowledge** that there may not be many different ways to code up particular algorithms and that your solution may be similar to other solutions available online.

