CS 170 Homework 2

Due **Monday 2/5/2024, 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, you must explicitly write "none".

2 Median

Recall the quickselect algorithm used for quickly finding the median of an array. Suppose you ran quickselect on the following array to find it's median.

12	78	13	1	97	45	48	26	85	100	78

1. Suppose you always pick the first element as the pivot. 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 $k = 6$.

2. Specify which elements should be picked as pivots at each step in order to **maximize** the runtime of this algorithm. Write the numerical value of the elements, not their indices.

3. Specify which elements should be picked as pivots at each step in order to **minimize** the runtime of this algorithm. Write the numerical value of the elements, not their indices.

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.

Hint: refer to Q2.

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.

(a) 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 $T(n)$, and try to use induction to show that $T(n) \leq c \cdot n$ for some $c > 0$.

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 Among Us

You are playing a party game with n other friends, who play either as imposters or crewmates. You do not know who is a crewmate and who is a imposter, but all your friends do. There are always more crewmates than there are imposters.

Your goal is to identify one player who is certain to be a crewmate.

Your allowed 'query' operation is as follows: you pick two players as partners. You ask each player if their partner is a crewmate or a imposter. When you do this, a crewmate must tell the truth about the identity of their partner, but a imposter doesn't have to (they may lie or tell the truth about their partner).

Your algorithm should work regardless of the behavior of the imposters.

(a) For a given player x, devise an algorithm that returns whether or not x is a crewmate 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 crewmate 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.

6 Modular Fourier Transform

Fourier transforms (FT) have to deal with computations involving irrational numbers which can be tricky to implement in practice. Motivated by this, in this problem you will demonstrate how to do a Fourier transform in modular arithmetic, using modulo 5 as an example.

(a) There exists $\omega \in \{0, 1, 2, 3, 4\}$ such that $\{\omega^0, \omega^1, \omega^2, \omega^3\}$ the are 4^{th} roots of unity (modulo 5), i.e., solutions to $z^4 = 1 \pmod 5$. When doing the FT in modulo 5, this ω will serve a similar role to the primitive root of unity in our standard FT. Show that $\{1, 2, 3, 4\}$ are the 4^{th} roots of unity (modulo 5), with $\omega = 2$ as the primitive root. Also show that $1 + \omega + \omega^2 + \omega^3 = 0 \pmod 5$ for $\omega = 2$.

(b) Using the FFT, produce the transform of the sequence $(0, 2, 3, 0)$ modulo 5; that is, evaluate the polynomial $2x + 3x^2$ at $\{1, 2, 4, 3\}$ using the recursive FFT algorithm defined in class, but with $\omega = 2$ and in modulo 5 instead of with $\omega = i$ in the complex numbers. Note that all calculations should be performed modulo 5.

Hint: You can verify your calculation by evaluating the polynomial at $\{1, 2, 4, 3\}$ *using the slow method (i.e. DFT).*

(c) Now perform the inverse FFT on the sequence $(0, 1, 4, 0)$, also using the recursive algorithm. Recall that the inverse FFT is the same as the forward FFT, but using ω^{-1} instead of ω, and with an extra multiplication by 4^{-1} for normalization.

(d) Now show how to multiply the polynomials $2x + 3x^2$ and $3 - x$ using the FFT modulo 5. You may use the fact that the FFT of $(3, 4, 0, 0)$ modulo 5 is $(2, 1, 4, 0)$ without doing your own calculation.

7 Coding Question: FFT

For this week's coding questions, you'll implement the **Fast Fourier Transform (FFT)** algorithm and apply it to speed up polynomial multiplication. 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-sp24-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.