CS 170 Homework 2

Due **Monday 9/12/2022, 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 Werewolves

You are playing a party game with n other friends, who play either as werewolves or villagers. You do not know who is a villager and who is a werewolf, but all your friends do. There are always more villagers than there are werewolves.

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

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

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

(a) Given a single person, devise an algorithm that returns whether or not that person is a villager 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 villager in $O(n \log n)$ queries (where one query is taking two people x and y and asking x to identify y and y to identify x).

There is a linear-time algorithm for this problem, but you cannot use it here, as we would like you to get practice with divide and conquer.

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

Give a 3-part solution.

(c) (**Extra Credit**) Can you give a linear-time algorithm?

Give a 3-part solution.

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

3 The Resistance

We are playing a variant of The Resistance, a board game where there are n players, k 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 no spies are chosen to go on the mission, but fails if at least one spy goes on the mission, and when a mission fails we are not told who the spies are that went on the mission.

Come up with a strategy that identifies all the spies in $O(k \log(n/k))$ missions. Only a main idea and runtime analysis are needed.

4 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 ω are 4^{th} roots of unity (modulo 5), i.e., solutions to $z^4 = 1$. 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). 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. All calculations should be performed modulo 5. *Hint: You can verify your calculation by evaluating the polynomial at the roots of unity using the slow method, if you like.*

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

5 Pattern Matching

Consider the following string matching problem:
Input:

- A string g of length n made of 0s and 1s. Let us call g, the "pattern".
- A string s of length m made of 0s and 1s. Let us call s the "sequence".
- Integer k

Goal: Find the (starting) locations of all length n-substrings of s which match g in at least $n-k$ positions.
Example: Using 0-indexing, if $g = 0111$, $s = 01010110111$, and $k = 1$ your algorithm should output 0,2,4 and 7.

(a) Give a $O(nm)$ time algorithm for this problem.

We will now design an $O(m \log m)$ time algorithm for the problem using FFT. *Pause a moment here to contemplate how strange this is. What does matching strings have to do with roots of unity and complex numbers?*

(b) Devise an FFT based algorithm for the problem that runs in time $O(m \log m)$. Write down the algorithm, prove its correctness and show a runtime bound.

Hint: On the example strings g and s, the first step of the algorithm is to construct the following polynomials

$$0111 \to 1 + x + x^2 - x^3$$
$$01010110111 \to -1 + x - x^2 + x^3 - x^4 + x^5 + x^6 - x^7 + x^8 + x^9 + x^{10}$$

To start, try to think about the case when $k = 0$ (i.e. g matches perfectly), and then work from there.

(c) (Extra Credit) Often times in biology, we would like to locate the existence of a gene in a species' DNA. Of course, due to genetic mutations, there can be many similar but not identical genes that serve the same function, and genes often appear multiple times in one DNA sequence. So a more practical problem is to find all genes in a DNA sequence that are similar to a known gene.

This problem is very similar to the one we solved earlier, the string s is complete sequence and the pattern g is a specific gene. We would like to find all locations in the complete sequence s, where the gene g appears, but for k modifications.

Except in genetics, the strings g and s consist of one of four alphabets $\{A, C, T, G\}$ (not 0s and 1s). Can you devise an $O(m \log m)$ time algorithm for this modified problem?

6 Coding Question

As a new initiative this semester, we're going to have short coding exercises so that you can get a feel of what it's like to implement these amazing algorithms that you learn about in lecture in real code. While the notebook may seem large, the actual code you have to write will usually be very short and direct.

There are two ways you can access the `hw2.ipynb` notebook and complete the problems:

1. Click here if you prefer to complete this question on Berkeley DataHub.
2. Run

```
git clone https://github.com/Berkeley-CS170/cs170-coding-notebooks-fa22.git
```

in your computer's terminal if you prefer to complete it locally.

Notes:

- *Submission Instructions:* Please merge your completed Jupyter Notebook with your written solutions for other questions and submit one merged pdf file to Gradescope.
- *OH/HWP Instructions:* While we will be providing conceptual help on the coding portion of homeworks, OH staff will not look at your code and/or help you debug.
- *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.