CS 170 Homework 2

Due **2/07/2022, at 10:00 pm**

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 citizens. You do not know who is a citizen and who is a werewolf, but all your friends do. There are always more citizens than there are werewolves.

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

Your allowed 'query' operation is as follows: you pick two people. You ask each person if their partner is a citizen or a werewolf. When you do this, a citizen 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) Give a way to test if a single player is a citizen 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 citizen 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?

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

3 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 $(0, 3, 2, 0)$ modulo 5; that is, evaluate the polynomial $3x + 2x^2$ 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 $(0, 4, 1, 0)$, 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. Note that although we would normally have to use the 8th roots of unity to do this multiplication, we can get away with just the 4th roots of unity in this case (why?).

4 Protein Matching

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.

To model this problem, let g be a length-n string corresponding to the known gene, and let s be a length-m string corresponding to the full DNA sequence, where $m \geq n$. We would like to solve the following problem: find the (starting) location of all length n-substrings of s which match g in at least $n - k$ positions. For example, using 0-indexing, if $g = ACT$, $s = ACTCTA$, and $k = 1$ your algorithm should output 0 and 2.

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

(b) Let's simplify this problem by assuming g and s are given as a vector of 0's or 1's. Can we use the dot product of g and $s[: len(g)]$ to figure out whether g and $s[: len(g)]$ differ in at most k positions? If so, how? Otherwise, what should we change to make this work?

(c) Let $g' = [0, 1]$ and $s = [1, 0, 0, 0]$. Apply any isomorphisms to the vectors, as specified in your answer from part (b). Now, treating s and g' as polynomials, perform polynomial multiplication. You do not need to use the DFT matrix and inverse DFT matrix for this question.

For some vector g, this polynomial product can be used to find the indices j where g and $s[j : j + n]$ match exactly. What is g? You may use your answer from part (b). *Hint: To answer the last question, you might have to convert g' to some other vector.*

(d) Assume g and s are given as a vector containing only 0's or 1's. Give a $O(m \log m)$ time algorithm that works for any k.

Hint: Represent the strings as vectors, and use FFT/fast polynomial multiplication. Are there some tricks used in the subparts above that we can use for this problem?

You do not need a 3-part solution for each part. Instead, describe the algorithms clearly and give an analysis of the running time.

5 Finding Clusters

We are given a directed graph $G = (V, E)$, where $V = \{1, \ldots, n\}$, i.e. the vertices are integers in the range 1 to n. For every vertex i we would like to compute the value $m(i)$ defined as follows: $m(i)$ is the smallest j such from which you can reach vertex i. (As a convention, we assume that i is reachable from i.)

(a) Show that the values $m(1), \ldots, m(n)$ can be computed in $O(|V| + |E|)$ time.

Please give a 3-part solution to this problem.

(b) Suppose we instead define $m(i)$ to be the smallest j that can be reached from i, instead of the smallest j from which you can reach i. How should you modify your answer to part (a) to work in this case?