CS 170 Homework 2

Due **Monday 2/6/2023, 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 Decimal to Binary

Given the n-digit decimal representation of a number, converting it into binary in the natural way takes $O(n^2)$ steps. Give a divide and conquer algorithm to do the conversion and show that it does not take much more time than (i.e. within a log factor of) Karatsuba's algorithm for integer multiplication.

Just state the main idea behind your algorithm and its runtime analysis; no proof of correctness is needed as long as your main idea is clear.

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) Show that $\{1, 2, 4, 3\}$ are the 4^{th} roots of unity (modulo 5), i.e., solutions to $z^4 = 1 \pmod 5$. Also show that $1 + \omega + \omega^2 + \omega^3 = 0 \pmod 5$ holds for $\omega = 2$, the primitive 4^{th} root of unity (mod 5).

(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 the primitive root as $\omega = 2$ in modulo 5 instead of $\omega = i$ 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 $2x + 3x^2$ 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.

4 Triple sum

We are given an array $A[0..n-1]$ with n elements, where each element of A is an integer in the range $0 \le A[i] \le n$ (the elements are not necessarily distinct). We would like to know if there exist indices i, j, k (not necessarily distinct) such that

$$A[i] + A[j] + A[k] = n$$

Design an $\mathcal{O}(n \log n)$ time algorithm for this problem. Note that you do not need to actually return the indices; just yes or no is enough. *Hint: Information can be encoded using exponents! If you're stuck, refer to problem 3 on the discussion worksheet.*

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

where $m \ge n$ and $k \le n$.

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 brute-force 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

For this week's homework, we'll be going over the Fast Fourier Transform (FFT) algorithm in a python jupyter notebook called `FFT.ipynb`. There are two ways you can access the notebook and complete the problems:

1. Click here and navigate to the `HW2` folder if you prefer to complete this question on Berkeley DataHub.
2. Run

   ```
   git clone https://github.com/Berkeley-CS170/cs170-coding-notebooks-sp23
   ```

   in your computer's terminal (and navigate to the `HW2` folder) if you prefer to complete it locally.

Notes:

- *Submission Instructions:* Please download your completed `FFT.ipynb` file and submit it to the gradescope assignment titled "Homework 2 Coding Portion".
- *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.