CS 3510: Design and Analysis of Algorithms
Section A, Spring 2017

Homework 4
Due: Friday 11:55PM, April 7th, 2017

Name: Alan Chiang

Notes:

1. Please don't write a program when you describe an algorithm. You should give an idea of what the algorithm does and then discuss the details of how your algorithm does specific tasks. If you give pseudo-code, you should explain your code.
2. For every algorithm that you are asked to describe, please clearly list the steps of your algorithm and explain its correctness and runtime behavior. We recommend separating the algorithm, correctness argument, and runtime analysis into three paragraphs.
3. You can assume that basic operations on integers which are not described to have a variable length (addition, multiplatication, subtraction, floor, ceiling, absolute value, comparison of less than, greater than, or equals, and anything you can form out of a constant number of these) can be done in constant time. You can also assume that, given i, you can read a_i from a list $a_1, ...a_n$ in constant time.

1. (10 points) Suppose you open a new deck of cards and shuffle it so that the order of the cards is completely random. What is the expected number of cards that are in the same position as they were at the start?

   Answer: Assuming a standard 52 card deck, after shuffling the chance of any particular card being in its original place is 1/52. There are 52 cards in the deck. Therefore, the expected number of cards in their original places is $(52)(1/52) = 1$. In fact, the expected value will remain 1 for any deck of size n, because every card will have a $1/n$ chance to remain in its original position and there are n cards. $(n)(1/n) = 1$.

2. (10 points) Suppose that you want to multiply the two polynomials $2x^2 + x + 1$ and $3x + 2$ using the FFT. Choose an appropriate power of two, find the FFT of the two sequences, multiply the results component-wise, and compute the inverse FFT to get the final result.

Answer: Let $A = 2x^2 + x + 1$ and $B = 3x + 2$. Then AB is degree 3 because of $6x^2$, so $n = 4$, both polynomials must be evaulated at 4 points, and we must use the 4th roots of unity:

$$\begin{bmatrix} 1 \\ i \\ -1 \\ -i \end{bmatrix}$$

Then we plug in the 4th roots of unity and multiply by coefficients to evaluate both polynomials at four points:

$$\begin{bmatrix} 1 & 1 & 1^2 \\ 1 & i & i^2 \\ 1 & -1 & (-1)^2 \\ 1 & -i & (-i)^2 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 4 \\ -1+i \\ 2 \\ -1-i \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1^2 \\ 1 & i & i^2 \\ 1 & -1 & (-1)^2 \\ 1 & -i & (-i)^2 \end{bmatrix} * \begin{bmatrix} 2 \\ 3 \\ 0 \end{bmatrix} = \begin{bmatrix} 5 \\ 2+3i \\ -1 \\ 2-3i \end{bmatrix}$$

Multiplying the results gives us the pointwise evaluation of AB:

$$\begin{bmatrix} 4 \\ -1+i \\ 2 \\ -1-i \end{bmatrix} * \begin{bmatrix} 5 \\ 2+3i \\ -1 \\ 2-3i \end{bmatrix} = \begin{bmatrix} 20 \\ 5-i \\ -2 \\ 5+i \end{bmatrix}$$

To transform these values back into coefficients, we use ω^{-1} for our roots of unity:

$$\begin{bmatrix} 1 \\ -i \\ -1 \\ i \end{bmatrix}$$

Performing the evaluation in reverse:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix} * \begin{bmatrix} 20 \\ 5-i \\ -2 \\ 5+i \end{bmatrix} = \begin{bmatrix} 8 \\ 20 \\ 28 \\ 24 \end{bmatrix}$$

And we divide by the n to get our final answer:

$$\begin{bmatrix} 2 \\ 5 \\ 7 \\ 6 \end{bmatrix}$$

or, in polynomial form: $6x^3 + 7x^2 + 5x + 2$

3. (10 points) Describe an algorithm that, given an n-bit integer, will output the square of that integer in $n\log(n)$ time.

   INPUT: An n-bit integer a

   OUTPUT: a^2

   Argue that your algorithm runs in $O(n\log(n))$ time and that it is correct.

   Answer: If the number is not already in binary, convert it to binary. We can treat each individual digit in the number as a coefficient in an n-degree polynomial, read from left to right such that the most significant binary digit is the coefficient of x^n. We then copy this polynomial, so we have two idential n-degree polynomials whose coefficients correspond to a. We can now run a Fast Fourier Transform on the two polynomials (algorithm exaplained in Problem 2 and in class). Once the FFT is completed, we simply evaluate its value at $x = 2$, which will give us the value of a^2.

   Decimal-to-binary conversion is a linear-time operation. Copying an n-degree polynomial is a linear-time operation. Running FFT is an $O(n\log(n))$ operation. Evaluating a polynomial is also a linear-time operation by Horner's Rule (algorithm explained in class). All operations run in sequence, not simultaneously, so we sum them and take the dominant term, in this case $O(n\log(n))$. Therefore the total time complexity is $O(n\log(n))$.

4. (10 points) Let $\omega_n = e^{(2\pi i/n)}$ denote the n-th root of unity.
In polar coordinates $\omega_n = (1,\ \frac{2\pi}{n})$.

Part (a):

$$\omega_n^1 \times \omega_n^2 \times ... \times \omega_n^n = \underline{\qquad} -1$$

(You can assume n is even.)

$$\omega_n^1 \times \omega_n^2 \times ... \times \omega_n^n = e^{(2\pi i/n)*1} * e^{(2\pi i/n)*2} * e^{(2\pi i/n)*3} * .... * e^{(2\pi i/n)*n} = e^{(2\pi i/n)*(1+2+3+...+n)}$$

The series $(1+2+3+...+n)$ can be expressed as $\frac{n(n+1)}{2}$. So $e^{(2\pi i/n)(1+2+3+...+n)} = e^{(2\pi i/n)(\frac{n(n+1)}{2})}$. Cancelling across the numerators and denominators gives us $e^{(\pi i)(n+1)} = e^{(n\pi i)} * e^{(\pi i)} = (1)(-1) = (-1)$.

So our answer is (-1).

Part (b):
What is $(\omega_n^2)^{-1}$, in other words, what is the multiplicative inverse of $(\omega_n)^2$?
Express your answer in polar coordinates where the angle θ satisfies $0 \leq \theta < 2\pi$.

By definition, the multiplicative inverse of $(\omega_n)^2$ must satisfy the equation $(\omega_n)^2 * (\omega_n^2)^{-1} = 1$. We know that $(\omega_n)^2 = (1,\ \frac{2\pi}{n} * 2) = (1,\ \frac{4\pi}{n})$. Therefore, $(\omega_n^2)^{-1}$ must equal $(1,\ \frac{2\pi}{n} * -2) = (1,\ \frac{-4\pi}{n})$. But since our answer must satisfy the condition "the angle θ satisfies $0 \leq \theta < 2\pi$." we reformat the the inverse such that $(\omega_n^2)^{-1} = (1,\ \frac{2\pi}{n} * -2) = (1,\ \frac{2\pi(n-2)}{n})$. This is possible because, as with all polar values, negative θ values simply mean we are traversing the circle in the clockwise direction instead of the positive θ counterclockwise traversal, so all negative θ values have a positive equivalent.

So our answer is $(1,\ \frac{2\pi(n-2)}{n})$.

Part (c):

$$\omega_n^1 + \omega_n^2 + ... + \omega_n^n = \underline{\qquad}0$$

(You can assume n is even.)

This problem is the equivalent of the geometric series $x + x^2 + ... + x^n$, where $x = \omega_n$. The sum of said series can be expressed as $a(\frac{r^n-1}{r-1})$. In this case, $r = \omega_n$, and because ω_n is an *nth* root of unity, we know that $\omega_n^n = 1$. Therefore the sum of this series is $a(\frac{1-1}{r-1}) = a(\frac{0}{r-1})$, which of course must equal 0 regardless of the other values because there is a zero in the numerator.

So our answer is 0.

Part (d):
The cubes of the 21st roots of unity are the 63rd roots of unity. TRUE or FALSE?

Answer: False. We know this because the 21st roots of unity are expressed as $e^{(2\pi i/21)}$ while the 63rd roots of unity are $e^{(2\pi i/63)}$. Cubing the former value simply gives us: $e^{(2\pi i/21)*3} = e^{(6\pi i/21)}$, which is clearly not equal to $e^{(2\pi i/63)}$.

So our answer is FALSE.