1 FFT Intro

We will use \( \omega_n \) to denote the first \( n \)-th root of unity \( \omega_n = e^{2\pi i/n} \). Maybe the most important fact about roots of unity for our purposes is that the squares of the \( 2n \)-th roots of unity are the \( n \)-th roots of unity.

### Fast Fourier Transform!

The Fast Fourier Transform \( \text{FFT}(p, n) \) takes arguments \( n \), some power of 2, and \( p \) is some vector \([p_0, p_1, \ldots, p_{n-1}]\).

Treating \( p \) as a polynomial \( P(x) = p_0 + p_1x + \ldots + p_{n-1}x^{n-1} \), the FFT computes the following matrix multiplication in \( O(n \log n) \) time:

\[
\begin{bmatrix}
P(1) \\
P(\omega_n) \\
P(\omega_n^2) \\
\vdots \\
P(\omega_n^{n-1})
\end{bmatrix} =
\begin{bmatrix}
1 & 1 & 1 & \ldots & 1 \\
1 & \omega_n^1 & \omega_n^2 & \ldots & \omega_n^{(n-1)} \\
1 & \omega_n^2 & \omega_n^4 & \ldots & \omega_n^{2(n-1)} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & \omega_n^{(n-1)} & \omega_n^{2(n-1)} & \ldots & \omega_n^{(n-1)(n-1)}
\end{bmatrix} \begin{bmatrix}
p_0 \\
p_1 \\
p_2 \\
\vdots \\
p_{n-1}
\end{bmatrix}
\]

If we let \( E(x) = p_0 + p_1x + \ldots + p_{n-2}x^{n-2} \) and \( O(x) = p_1 + p_3x + \ldots + p_{n-1}x^{n-2} \), then \( P(x) = E(x^2) + xO(x^2) \), and then \( \text{FFT}(p, n) \) can be expressed as a divide-and-conquer algorithm:

**Solution:** Apologies for the misprint. It is supposed to be:

\[
E(x) = p_0 + p_2x + \ldots + p_{n-2}x^{n/2-1} \\
O(x) = p_1 + p_3x + \ldots + p_{n-1}x^{n/2-1}
\]

1. Compute \( E' = \text{FFT}(E, n/2) \) and \( O' = \text{FFT}(O, n/2) \).
2. For \( i = 0 \ldots n-1 \), assign \( P(\omega_n^i) \leftarrow E((\omega_n^i)^2) + \omega_n^i O((\omega_n^i)^2) \)

(a) Let \( p = [p_0] \). What is \( \text{FFT}(p, 1) \)?

**Solution:** Notice the FFT matrix is just \([1]\), so \( \text{FFT}(p, 1) = [p_0] \).

(b) Use the FFT algorithm to compute \( \text{FFT}([1, 4], 2) \) and \( \text{FFT}([3, 2], 2) \).

**Solution:** \( \text{FFT}([1, 4], 2) = [5, -3] \) and \( \text{FFT}([3, 2], 2) = [5, 1] \).

We show how to compute \( \text{FFT}([1, 4], 2) \), and \( \text{FFT}([3, 2], 2) \) is similar.

First we compute \( \text{FFT}([1, 1]) = [1] = E' \) and \( \text{FFT}([4, 1]) = [4] = O' \) by part (a). Notice that \( E' = [E(1)] = 1 \) and \( O' = [O(1)] = [4] \), so when we need to use these values later they have already been computed in \( E' \) and \( O' \).

Let \( P \) be our result. We wish to compute \( P(\omega_n^0) = P(1) \) and \( P(\omega_n^1) = P(-1) \).

\[
P(1) = E(1) + 1 \cdot O(1) = 1 + 4 = 5 \\
P(-1) = E(1) - 1 \cdot O(1) = 1 - 4 = -3
\]

So our answer is \([5, -3]\).
(c) Use your answers to the previous parts to compute \( \text{FFT}([1, 3, 4, 2], 4) \).

Solution: \( \omega_4 = i \). The following table is good to keep handy:

\[
\begin{array}{c|cccc}
\omega_4^i & 1 & i & -1 & -i \\
\omega_4 & 1 & -1 & 1 & -1
\end{array}
\]

Let \( E' = \text{FFT}([1, 4], 2) = [5, -3] \) and \( O' = \text{FFT}([3, 2], 2) = [5, 1] \). Notice that \( E' = [E(1), E(-1)] = [5, -3] = \) and \( O' = [O(1), O(-1)] = [5, 1] \), so when we need to use these values later they have already been computed in the divide step. Let \( R \) be our result, we wish to compute \( R(1), R(i), R(-1), R(-i) \).

\[
\begin{align*}
R(1) &= E(1) + 1 \cdot O(1) = 5 + 5 = 10 \\
R(i) &= E(-1) + i \cdot O(-1) = -3 + i \\
R(-1) &= E(1) - 1 \cdot O(1) = 5 - 5 = 0 \\
R(-i) &= E(-1) - i \cdot O(-1) = -3 - i
\end{align*}
\]

So our answer \([10, -3 + i, 0, -3 - i]\).

(d) Describe how to multiply two polynomials \( p(x), q(x) \) in coefficient form of degree at most \( d \).

Solution: The idea is to take the FFT of both \( p \) and \( q \), multiply the evaluations, and then take the inverse FFT. Note that \( p \cdot q \) has degree at most \( 2d \), which means we need to pick \( n \) as the smallest power of 2 greater than \( 2d \), call this \( 2^k \). We can zero-pad both polynomials so they have degree \( 2^k - 1 \).

Then \( M = \text{FFT}(p, 2^k) \cdot \text{FFT}(q, 2^k) \) (with multiplication elementwise) computes \( pq(\omega_{2^k}^i) \) for all \( i = 0, \ldots, 2^k - 1 \).

We take the inverse FFT of \( M \) to get back to \( p \cdot q \) in coefficient form.

# Cubed Fourier

(a) Cubing the 9th roots of unity gives the 3rd roots of unity. Next to each of the third roots below, write down the corresponding 9th roots which cube to it. The first has been filled for you. We will use \( \omega_9 \) to represent the primitive 9th root of unity, and \( \omega_3 \) to represent the primitive 3rd root.

\[
\begin{align*}
\omega_3^0 & \ : \ \omega_9^0, \\
\omega_3^1 & \ : \ , \\
\omega_3^2 & \ : \ ,
\end{align*}
\]

(b) You want to run FFT on a degree-8 polynomial, but you don’t like having to pad it with 0s to make the (degree+1) a power of 2. Instead, you realize that 9 is a power of 3, and you decide to work directly with 9th roots of unity and use the fact proven in part (a). Say that your polynomial looks like \( P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_8 x^8 \). Describe a way to split \( P(x) \) into three pieces so that you can make an FFT-like divide-and-conquer algorithm.

Solution:

(a) \( \omega_9^0 : \omega_9^0, \omega_9^3, \omega_9^6 \)

\( \omega_9^1 : \omega_9^1, \omega_9^4, \omega_9^7 \)

\( \omega_9^2 : \omega_9^2, \omega_9^5, \omega_9^8 \)
(b) Let \( P(x) = P_1(x^3) + xP_2(x^3) + x^2P_3(x^3) \)
where \( P_1(x^3) = a_0 + a_3x^3 + a_6x^6. \)
and \( P_2(x^3) = a_1 + a_4x^3 + a_7x^6. \)
and \( P_3(x^3) = a_2 + a_5x^3 + a_8x^6. \)
3 Cartesian Sum

Let \( A \) and \( B \) be two sets of integers in the range 0 to 10. The \emph{Cartesian sum} of \( A \) and \( B \) is defined as

\[
A + B = \{ a + b \mid a \in A, b \in B \}
\]

i.e. all sums of an element from \( A \) and an element with \( B \). For example, \( \{1,3\} + \{2,4\} = \{3,5,7\} \).

Note that the values of \( A + B \) are integers in the range 0 to 20. Design an algorithm that finds the elements of \( A + B \) in \( O(n \log n) \) time, which additionally tells you for each \( c \in A + B \), \emph{how many} pairs \( a \in A, b \in B \) there are such that \( a + b = c \).

\textit{Hint:} Notice that \((x^1 + x^3) \cdot (x^2 + x^4) = x^3 + 2x^5 + x^7\)

\textbf{Solution:}

Define two polynomials \( P \) and \( Q \) as follows:

\[
P = \sum_{a \in A} x^a, \quad Q = \sum_{b \in B} x^b
\]

Notice that \( P \) and \( Q \) both have degree at most 10n.

Let \( R(x) = P(x) \cdot Q(x) = r_0 + r_1x + \ldots + r_{20n}x^{20n} \). The formula for \( r_k \) is given by:

\[
r_k = \sum_{j=0}^{k} p_j q_{k-j}
\]

Note that \( p_j q_{k-j} \) is either 0 or 1, and it is 1 iff \( j \in A, k-j \in B \). So \( r_k \) counts the number of \( a \in A, b \in B \) such that \( a + b = k \). So by simply reading the coefficients of the result polynomial, we get the answer.

We can compute the polynomial multiplication in \( O(n \log n) \) time using the FFT, so we are finished.