cs3821 Assignment 2

Jennifer z5207930

April 2020

1 Question 3

1.1 Compute P(x)*Q(x)

Given $P(x) = 3x^3 + 5x + 4$ and $Q(x) = 2x^2 + 3x + 2$,

$$\begin{aligned} P(x) * Q(x) &= (3x^3 + 5x + 4)(2x^2 + 3x + 2) \\ &= 3x^3(2x^2 + 3x + 2) + 5x(2x^2 + 3x + 2) + 4(2x^2 + 3x + 2) \\ &= (6x^5 + 9x^4 + 6x^3) + (10x^3 + 15x^2 + 10x) + (8x^2 + 12x + 8) \\ &= 6x^5 + 9x^4 + 10x^3 + 23x^2 + 22x + 8 \end{aligned}$$

1.2 Explanation

Supposed we want to multiply two polynomials of deg n by their coefficients. Let the coefficients for $P(x)$ and $Q(x)$ by vectors $A = [a_0, a_1, ..., a_n]$ and $B = b_0, b_1, ..., b_n$ where the kth entry for $k\epsilon 0, 1, 2, 3, ...$ represents the coefficients of x^k.
A naive method of performing this would be to perform a $O(n^2)$ algorithm to times all the coefficients with each other like in the Discrete Fourier Theorem However, we can apply the Fast Fourier Theorem and the Inverse Fast Fourier Theorem.
First, we want to pad our vectors with n zeros as $R(x)$ would have deg $2n + 1$ and so when we perform an Inverse FFT, the result would correspond to our resulting vector so size $2n + 1$. Now, have vectors A and B

$$A = [a_0, a_1, ..., a_n, 0, 0, ..(n + 1 times)..0]$$
$$B = [b_0, b_1, ..., b_n, 0, 0, ..(n + 1 times)..0]$$

Then we want to apply the FFT onto both A and B. The Fast Fourier Theorem takes advantage of the complex conjugate symmetry where

$$W_N^{k(N-n)} = W_N^{-kn} = conjugate(W_N^{kn}),$$

and the periodicity in complex roots of unity where,

$$W_N^{kn} = W_N^{k(N+n)} = W_N^{(k+n)n},$$

In short, the FFT is a recursive algorithm which continues to split the coefficients of the polynomial in half $logn$ times. This continues until we hit the base case where we deal with $a_0 and a_1$ since they are the first pair of odd and even. For example, in a polynomial of size $N = 8$

$$\begin{aligned} P(x) &= [a_0, a_1, ...a_7, a_8] \\ &= [a_0, a_2, a_4, a_6, a_8], [a_1, a_3, a_5, a_8] \\ &= [a_0, a_2], [a_4, a_6, a_8], [a_1, a_3], [a_5, a_8] \end{aligned}$$

Note how we avoided the case where the length is only 1, as that is the base case and it immediately returns from there. It then computes and simplifies the odd and even power. For example:

$$P(k) = P_{even}[k] + W_N^k P_{odd}[k], \text{ where } W_N^k = e^{-2i*k*n\pi/N}$$

The FFT is complicated but symmetrical in its calculation and allow us to perform the DFT in a much faster $O(nlogn)$. After performing FFT on both A and B, we can produce

$$\begin{aligned} P &= [A_0, A_1, ...A_{2n+1}] \\ Q &= [B_0, B_1, ...B_{2n+1}] \end{aligned}$$

Then, we can calculate the resulting Vector R.

$$R = [C_0, C_1, ...C_{2n+1}], \text{ where } C_k = A_k * B_k$$

Now we have a resulting vector, we need to apply an Inverse FFT in order to extrapolate the coefficients of our resulting polynomial $R(x)$. This is similar to FFT, except backwards as we want to find the original results. Then, we have our resulting vector

$$r = [r_0, r_1, ...r_2n, r_{2n+1}]$$

Therefore our resulting polynomial would be $R(x) = r_{2n}x^{2n} + r_{2n-1}x^{2n-1}.... + r_0$.

1.3 Time Complexity

The time complexity of FFT is $O(nlogn)$.