10 Polynomial Multiplication and Fast Fourier Transform [KT 5.6]

Given two degree-$(n-1)$ univariate polynomials $A(x) = \sum_{j=0}^{n-1} a_j x^j$ and $B(x) = \sum_{j=0}^{n-1} b_j x^j$, it is easy to compute their sum in linear time. However, the naïve way to compute their product

$$C(x) = A(x)B(x) = \sum_{j=1}^{2n-2} \left(\sum_k a_k b_{j-k} \right) x^j$$

takes $\Theta(n^2)$ time.

To compute $C(x)$ faster, we plan to use the following strategy.

1. Pick $(2n-1)$ different locations $x_1, x_2, \ldots, x_{2n-1}$, and evaluate and $A(x_1), A(x_2), \ldots, A(x_{2n-1})$ and $B(x_1), B(x_2), \ldots, B(x_{2n-1})$.
2. Now we know the evaluation of $C(x)$ at locations $x_1, x_2, \ldots, x_{2n-1}$. More specifically, for each j, we have $C(x_j) = A(x_j)B(x_j)$.
3. Use the $(2n-1)$ evaluations of $C(x)$ to recover the coefficients of $C(x) = \sum_{j=0}^{2n-2} c_j x^j$.

Before talking about time complexity, let us first prove the feasibility (correctness) of such strategy. Indeed, the correctness comes from the following fact.

Fact 1 *Given n points $(x_1, y_1), \ldots, (x_n, y_n)$ with distinct x_i's, there exists a unique degree-$(n-1)$ univariate polynomial passing through all these n points.*

Proof: To prove the fact, we first prove that there exists such a polynomial and then show that the polynomial is unique.

Proof of Existence. This is done by interpolation. For each $j = 1, 2, \ldots, n$, we write down the following polynomial

$$\frac{\Pi_{k=1,\ldots,n,k\neq j}(x - x_k)}{\Pi_{k=1,\ldots,n,k\neq j}(x_j - x_k)},$$

and observe that such a degree-$(n-1)$ polynomial takes value 1 at $x = x_j$ and value 0 at $x = x_k$ for all $k \neq j$. Therefore, it is easy to verify the following degree-$(n-1)$ polynomial takes value y_j at $x = x_j$ for all j.

$$\sum_{j=1}^{n} \frac{\Pi_{k=1,\ldots,n,k\neq j}(x - x_k)}{\Pi_{k=1,\ldots,n,k\neq j}(x_j - x_k)} y_j.$$

To prove the uniqueness of such a polynomial, we use the Fundamental Theorem of Algebra.

Theorem 1 *Every non-zero, single-variable, degree-n polynomial with complex coefficients has, counted with multiplicity, exactly n complex roots.*

Proof of Uniqueness. Now let us assume there are two degree-$(n-1)$ polynomials $p(x)$ and $q(x)$ passing through the given n points. Let $r(x) = p(x) - q(x)$. We see that $r(x_j) = 0$ for all $j = 1, 2, ..., n$. In other words, the degree-$(n-1)$ polynomial $r(x)$ has n roots. According to the Fundamental Theorem of Algebra, $r(x)$ must be constantly zero, i.e. $p(x)$ and $q(x)$ are the same polynomial. $\square$

Now we see that the strategy mentioned above is correct. But what about the time complexity? Step 2 takes $O(n)$ time. However, we need a fast algorithm for Step 1 and Step 3.

While a naive implementation of Step 1 and Step 3 takes at least $\Omega(n^2)$ time, we observe that we can choose evaluation locations $x_1, \ldots, x_{2n-1}$ at our will. We hope to choose a few locations those are easier for fast evaluation. Here is our new task.

New task. Pick n evaluation locations $x_1, \ldots, x_n$ and design a fast algorithm to evaluate any degree-$(n-1)$ polynomial $A(x) = \sum_{j=0}^{n-1} a_j x^j$ at these n locations. Let us assume n is a power of 2 (i.e. $n = 2^m$ for some integer m) but we can always meet such a condition by padding 0's to the higher power terms of the input polynomial.

Indeed, the n locations we pick will be the n n-th roots of unity.

Fact 2 *There are exactly n n-th root of unity, and they are $e^{2\pi i \cdot \frac{1}{n}}, e^{2\pi i \cdot \frac{2}{n}}, \ldots, e^{2\pi i \cdot \frac{n}{n}} = 1$.*

Definition 1 *An n-th root of unity x is primitive if $x^k \neq 1$ for all $0 < k < n$.*

Fact 3 *$e^{2\pi i \cdot \frac{j}{n}}$ is a primitive n-th root of unity if any only if $gcd(j, n) = 1$.*

We will let $\omega = e^{2\pi i \cdot \frac{1}{n}}$ and evaluate the polynomials at n locations $\omega, \omega^2, \ldots, \omega^n = 1$.

In order to evaluate the given polynomial $A(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}$ at point $1, \omega, \ldots, \omega^{n-1}$, we introduce

$$A_{even}(x) = a_0 + a_2 x + a_4 x^2 + \cdot + a_{n-2} x^{(n-2)/2},$$
$$A_{odd}(x) = a_1 + a_3 x + a_5 x^2 + \cdot + a_{n-1} x^{(n-2)/2},$$

and we apply the following identity

$$A(x) = A_{even}(x^2) + x A_{odd}(x^2)$$

to $x = 1, \omega, \omega^2, \ldots, \omega^{n-1}$. We observe that we need to evaluate both degree-$(\frac{n}{2} - 1)$ polynomials A_{even} and A_{odd} at the following n points $x = 1, \omega^2, \ldots, \omega^{2n-2}$. However, since $x^n = 1$, we have

$$\omega^n = 1, \omega^{n+2} = \omega^2, \ldots, \omega^{2n-2} = \omega^{n-2}.$$

Therefore, there are only $n/2$ distinct points for evaluation. And this reduces the problem of evaluating the degree-$(n-1)$ polynomial $A(x)$ at n n-th roots of unity to the two subproblems of evaluating degree-$(n/2-1)$ polynomials at $(n/2)$ $(n/2)$-th roots of unity. Following this idea, we are able to design the Fast Fourier Transform algorithm in $O(n \log n)$ time. Finally, we formally describe the FFT algorithm to evaluate a polynomial at n-th roots of unity as follows.

FFT($n, A(x) = \sum_{j=0}^{n-1} a_j x^j$)

IF $(n = 1)$ RETURN $A(1) = a_0$

Let $\omega = e^{2\pi i \cdot \frac{1}{n}}$

$A_{even}(x) = a_0 + a_2 x + \cdots + a_{n-2} x^{n/2-1}$

$A_{odd}(x) = a_1 + a_3 x + \cdots + a_{n-1} x^{n/2-1}$

Recursively evaluate $A_{even}(\omega^2), A_{even}((\omega^2)^2), \ldots, A_{even}((\omega^2)^{n/2})$

Recursively evaluate $A_{odd}(\omega^2), A_{odd}((\omega^2)^2), \ldots, A_{odd}((\omega^2)^{n/2})$

For each $j = 1, \ldots, n$, evaluate $A(\omega^j) = A_{even}((\omega^2)^j) + \omega^j \cdot A_{odd}((\omega^2)^j)$

Now we discuss the inverse Fast Fourier Transform in order to complete Step 3 of the whole polynomial multiplication algorithm. That is, once we have the evaluations $C(1), \ldots, C(\omega^{n-1})$, we would like to design a fast algorithm to recover the coefficients of $C(x)$ polynomial. In order to do this, we introduce the Fourier Transform matrix and took a matrix view of Fourier Transform. We define the Fourier Transform matrix $M_n(\omega)$ to be an $n \times n$ matrix in the following form.

$$M_n(\omega) = \begin{bmatrix} 1 & 1 & 1 & 1 & \ldots \\ 1 & \omega & \omega^2 & \omega^3 & \ldots \\ 1 & \omega^2 & \omega^4 & \omega^6 & \ldots \\ 1 & \omega^3 & \omega^6 & \omega^9 & \ldots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

That is, the i-th row and j-th column of the matrix is $\omega^{(i-1)(j-1)}$.

We can verify that for any degree-$(n-1)$ polynomial $A(x) = a_0 + a_1 x + \ldots + a_{n-1} x^{n-1}$, we have

$$\begin{pmatrix} A(1) \\ A(\omega) \\ A(\omega^2) \\ \vdots \end{pmatrix} = M_n(\omega) \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \end{pmatrix}$$

and this is the matrix view of Fourier Transform (also what was done by the FFT algorithm). In order to recover the coefficients of a polynomial form its evaluations at n-th roots of unity, we can simply multiply the left hand side of the equation above by the inverse of the Fourier Transform matrix, i.e.,

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \end{pmatrix} = (M_n(\omega))^{-1} \begin{pmatrix} C(1) \\ C(\omega) \\ C(\omega^2) \\ \vdots \end{pmatrix}$$

The following claim shows that $(M_n(\omega))^{-1} = \frac{1}{n} M_n(\omega^{-1})$. Indeed, using this claim and the fact that ω^{-1} is also a primitive n-th root of unity, we see that we only need to invoke the FFT algorithm again, with ω^{-1} and $C(1), C(\omega), C(\omega^2), \ldots$ as input, and get the coefficients of $C(x)$ in $O(n \log n)$ time.

Claim 6 $M_n(\omega)M_n(\omega^{-1}) = nI$.

Proof: Let us directly compute the i-th row and j-th entry of the product matrix. By the definition of matrix multiplication, this value is

$$\sum_{k=1}^{n}(M_n(\omega))_{i,k}(M_n(\omega^{-1}))_{k,j} = \sum_{k=1}^{n}\omega^{(i-1)(k-1)}w^{(-1)(k-1)(j-1)} = \sum_{k=1}^{n}(\omega^{i-j})^{(k-1)}.$$

$\square$

We see this is sum of a geometric series. If $i = j$ (and therefore $\omega^{i-j} = 1$), this sum becomes n, otherwise we have $\omega^{i-j} \neq 1$, and the sum becomes

$$\frac{1-(\omega^{i-j})^n}{1-\omega^{i-j}} = 0,$$

where the equation is because ω^{i-j} is also an n-th root of unity.

To summarize, we have proved that if $i = j$, the diagonal entries are always n, otherwise, the off-diagonal entries are always 0. Therefore the product matrix is nI.