CSE548 & AMS542: Analysis of Algorithms, Fall 2017 **Lecturer:** Rezaul A. Chowdhury
Topic: Polynomial Multiplication and Fast Fourier Transform **Date:** 13th September 2017
Scribe: Aditya Bhalchandra Yele

5.1 Point-Value Representation of Polynomials

If you have a polynomial $\mathrm{A}(x)$ of degree bound n, the Point-Value representation of $\mathrm{A}(x)$ is a set of pairs $\{(x_0, y_0), (x_1, y_1), \cdots, (x_n - 1, y_n - 1)\}$ such that for x_k is distinct and $y_k = \mathrm{A}(x_k)$ where $0 \leq k \leq n-1$ These n point value pairs uniquely determine a polynomial

5.1.1 Addition

Suppose we have two polynomials $\mathrm{A}(x)$ and $\mathrm{B}(x)$ with their point value representations

$A : \{(x_0, y_0^a), (x_1, y_1^a), ...(x_n - 1, y_n - 1^a)\}$

$B : \{(x_0, y_0^b), (x_1, y_1^b), ...(x_n - 1, y_n - 1^b)\}.$

Both the representations use the same set of distinct points $\{x_0, x_1, ..., x_n - 1\}$. The polynomial C(x) such that $\mathrm{C}(x) = \mathrm{A}(x) + \mathrm{B}(x)$ can also be calculated by using the point value representation of A and B by adding the y values at corresponding x

$C : \{(x_0, y_0^a + y_0^b), (x_1, y_1^a + y_1^b), ...(x_n - 1, y_n - 1^a + y_n - 1^b)\}$

This takes $\Theta(n)$ time

5.1.2 Multiplication

Point value representation can be used to multiply two polynomials in a way similar to addition. This takes $\Theta(n)$ time.

We use the same equations $A(x)$ and $B(x)$. However, we take $2n - 1$ pairs are used instead of n. This is because to uniquely determine the equation of the resultant polynomial we would require at least $2n - 1$ points, as the degree bound of the resultant polynomial will be 2n.

Thus for two equations $A(x)$ and $B(x)$ with point value representations

$A : \{(x_0, y_0^a), (x_1, y_1^a), ...(x_{2n-1}, y_{2n-1}^a)\}$

$B : \{(x_0, y_0^b), (x_1, y_1^b), ...(x_{2n-1}, y_{2n-1}^b)\}.$

We can calculate $C(x) = A(x) \times B(x)$ as

$C : \{(x_0, y_0^a \times y_0^b), (x_1, y_1^a \times y_1^b), ..., (x_{2n-1}, y_{2n-1}^a \times y_{2n-1}^b)\}$

5.2 Coefficient form to Point Value Form

While multiplication using point value point is cheaper we haven't yet considered the cost of converting polynomials in coefficient form to point value form and vice versa

One way to convert is to choose n points and evaluate the result of the polynomial for each point. For each point this approach takes $\Theta(n)$ and for n points it takes $\Theta(n^2)$

For conversion from point value form to coefficient form we could use Langrange's formula

$$A(x) = \sum_{k=0}^{n-1} \frac{\prod_{j \neq k}(x - x_j)}{\prod_{j \neq k}(x_k - x_j)} y_k$$

This again takes $\Theta(n^2)$ time

5.2.1 Faster Conversion

A polynomial $A(x)$ of degree bound n can be represented in matrix form as follows

$$\begin{bmatrix} A(x_0) \\ A(x_1) \\ \vdots \\ A(x_{n-1}) \end{bmatrix} = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{n-1} \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix}$$

The point value representation of the above polynomial is

$$A : \{(x_0, y_0), (x_1, y_1), ...(x_n - 1, y_n - 1)\}$$

where $\{x_0, x_1, \cdots, x_n - 1\}$ are n distinct points and $y_k =$ A(x_k) where $0 \leq k \leq n - 1$

We have the freedom to choose the values of x. If the points are chosen intelligently the speed of conversion increases. We choose the values of x such that the second half of the values are the negative numbers of the first part. For instance if the degree bound of A(x) is 4 then we can choose x_0, x_1, x_2, x_3 such that

$$x_2 = -x_0$$
$$x_3 = -x_1$$

A(x) can be written as

$$\begin{aligned}
A(x) &= \sum_{l=0}^{n-1} a_l x^l \\
&= \sum_{l=0}^{n/2-1} a_{2l}x^{2l} + \sum_{l=0}^{n/2-1} a_{2l+1}x^{2l+1} \\
&= \sum_{l=0}^{n/2-1} a_{2l}x^{2l} + x\sum_{l=0}^{n/2-1} a_{2l+1}x^{2l} \\
&= A_{even}(x^2) + xA_{odd}(x^2)
\end{aligned}$$

For all n points x we can calculate A(x) as

$$\begin{aligned}
A(x_j) &= A_{even}(x_j^2) + x_jA_{odd}(x_j^2) \\
A(x_{n/2+j}) &= A_{even}(x_j^2) - x_jA_{odd}(x_j^2)
\end{aligned}$$

So to evaluate A(x_j) for all $0 \leq j \leq n/2-1$ we need to recursively evaluate $A_{even}(x_j^2)$ and $A_{odd}(x_j^2)$ and add the result for all j. This takes $\Theta(n \log n)$ time.

5.2.1.1 Recursive Evaluation

We need to recursively evaluate $A_{even}(x^2)$ and $A_{odd}(x^2)$

Let us consider an example of polynomial of degree bound 4. Here A$(x) = a_0 + a_1x + a_2x^2 + a_3x^3$ We have to choose 4 distinct points to calculate A(x)

$$\begin{bmatrix} A(x_0) \\ A(x_1) \\ A(x_2) \\ A(x_3) \end{bmatrix} = \begin{bmatrix} 1 & x_0 & x_0^2 & x_0^3 \\ 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Let's set x_2 and x_3 such that $x_2 = -x_0 and x_3 = -x_1$ We now have to evaluate only two polynomials, each can be divided into even and odd halves. Let us consider the even parts the process is similar for odd parts.We evaluate using the above $A_{even}(x^2)$

$A_{even}(x_0^2) = a_0 + x_0^2$
$A_{even}(x_1^2) = a_1 + x_1^2$

To evaluate this set $x_1^2 = -x_0^2$. Let's say we choose x_0 to be 1 then x_1^2 will be -1. Thus

$$\begin{aligned}
x_0 &= 1 \\
x_1 &= i \\
x_2 &= -1 \\
x_3 &= -i
\end{aligned}$$

For a polynomial of degree bound 8 we will again divide and evaluate with $A(x_{even}^2)$ which will be a polynomial of degree bound 4. The values returned by it will be the squares of the values we want for a polynomial of degree bound 8

That is for degree bound 8 we will get the same values $1, i, -1, -i$ but they will be the squares of the x_0, x_1, x_2, x_3 of the polynomial of degree bound 8. So for a polynomial of degree bound 8

$$\begin{aligned}
x_0 &= \sqrt{1} = 1 \\
x_1 &= \sqrt{i} = \frac{1}{\sqrt{2}}(1+i) \\
x_2 &= \sqrt{-1} = i \\
x_3 &= \sqrt{-i} = \frac{1}{\sqrt{2}}(1-i)
\end{aligned}$$

5.2.1.2 Roots of Unity

Using the power series for $cos(\alpha)$, $sin(\alpha)$, e^{α} we can derive Eulers formula which says,

For any real number α

$$cos(\alpha) + isin(\alpha) = e^{i\alpha}$$

Also for any real numbers α and p

$$\begin{aligned}
(cos(\alpha) + isin(\alpha))^p &= (e^{i\alpha})^p \\
&= e^{i(p\alpha)} \\
&= cos(p\alpha) + isin(p\alpha)
\end{aligned}$$

Let ω_n be the n^{th} root of unity

$$\begin{aligned}
\omega_n &= e^{(2\pi i/n)} \\
&= cos(2\pi/n) + isin(2\pi/n)
\end{aligned}$$

Using the above results it is possible to find the value of k^{th} root of unity from $0 \leq k \leq n-1$, after which it repeats.

$$\begin{aligned}
1^{1/n} &= cos(2\pi k/n) + isin(2\pi k/n) \\
&= (\omega_n)^k
\end{aligned}$$

Thus n^{th} roots of unity are given by $1, \omega_n, (\omega_n)^2, \cdots, (\omega_n)^{n-1}$

5.2.1.3 Fast Fourier Transform

For a polynomial of degree n we need to perform evaluations at most $\log n$ times We then choose $x_0 = 1 = \omega_n^0$ and set $x_j = \omega_n^j$ for $1 \leq j \leq n-1$

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n & (\omega_n)^2 & \cdots & (\omega_n)^{n-1} \\ 1 & \omega_n^2 & (\omega_n^2)^2 & \cdots & (\omega_n^2)^{n-1} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & \omega_n^{n-1} & (\omega_n^{n-1})^2 & \cdots & (\omega_n^{n-1})^{n-1} \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \vdots \\ a_{n-1} \end{bmatrix}$$

The vector y is called Discrete Fourier Transform and this method of computing is called Fast Fourier Transform