Fast Fourier Transform

Yixiong Gao

May 17th, 2022

Polynomial Multiplication

- Degree-d polynomial : $A(x) = a_0 + a_1x + a_2x^2 + \cdots + a_dx^d$

- Polynomial multiplication :
  - Consider $A(x) = \sum_{i=0}^{d} a_ix^i$, $B(x) = \sum_{i=0}^{d} b_ix^i$
  - Their product $C(x) = A(x) \cdot B(x) = \sum_{i=0}^{2d} c_ix^i$
  - Where $c_k = a_0b_k + a_1b_{k-1} + \cdots + a_kb_0 = \sum_{i=0}^{k} a_ib_{k-i}$

- Compute c_k takes $O(k)$ steps;
- Compute all coefficients require $\sum_{k=0}^{2d+1} O(k) = O(d^2)$ time.

Representation of polynomials

- **Fact** : degree-d polynomial $A(x)$ is uniquely characterized by its values at ***any*** d + 1 distinct points $(x_i, A(x_i))$.
- $$\begin{cases} a_0 + a_1x_0 + a_2x_0^2 + \cdots + a_dx_0^d = A(x_0) \\ a_0 + a_1x_1 + a_2x_1^2 + \cdots + a_dx_1^d = A(x_1) \\ \vdots \\ a_0 + a_1x_d + a_2x_d^2 + \cdots + a_dx_d^d = A(x_d) \end{cases} \Leftrightarrow \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^d \\ 1 & x_1 & x_1^2 & \cdots & x_1^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_d & x_d^2 & \cdots & x_d^d \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_d \end{bmatrix} = \begin{bmatrix} A(x_0) \\ A(x_1) \\ \vdots \\ A(x_d) \end{bmatrix}$$

- The coefficient matrix is a Vandermonde matrix.
- Since $x_0, x_1, \dots, x_d$ are pairwise distinct, the solution is unique.

Multiply in the value representation

- What's the time complexity of multiplying in value representation?
- $C(x_i) = A(x_i)B(x_i)$, only requires $O(d)$ time.

- Consider Multiply in this method:
  - Evaluation: translate $A(x), B(x)$ from coefficients to values at the chosen points $x_0, x_1, \ldots, x_{2d}$
  - Multiplication: multiply in the value representation $C(x_i) = A(x_i)B(x_i)$
  - Interpolation: translate $C(x)$ back to coefficients

Multiplying Process

Figure 2.5 Polynomial multiplication

Input: Coefficients of two polynomials, $A(x)$ and $B(x)$, of degree d
Output: Their product $C = A \cdot B$

Selection

Pick some points $x_0, x_1, \ldots, x_{n-1}$, where $\boxed{n \geq 2d+1}$

Evaluation

Compute $A(x_0), A(x_1), \ldots, A(x_{n-1})$ and $B(x_0), B(x_1), \ldots, B(x_{n-1})$

Multiplication

Compute $C(x_k) = A(x_k)B(x_k)$ for all $k = 0, \ldots, n-1$

Interpolation

Recover $C(x) = c_0 + c_1 x + \cdots + c_{2d} x^{2d}$

Idea : how to choose points

- If we need to pick $2n$ points $(2n \geq 2d + 1)$
- Try to choose them to be positive-negative pairs :$\pm x_1, \pm x_2, \ldots \pm x_n$
- $\begin{cases} A(x_i) = \boxed{a_0} + \boxed{a_1 x_i} + \boxed{a_2 x_i^2} + \cdots + a_d x_i^d. \\ A(-x_i) = \boxed{a_0} - \boxed{a_1 x_i} + \boxed{a_2 x_i^2} - \cdots + \boxed{(-1)^d} a_d x_i^d \end{cases}$
- Split $A(x)$ into its odd and even powers: $A(x) = A_e(x^2) + xA_o(x^2)$
- $\begin{cases} A_e(x) = a_0 + a_2 x + a_4 x^2 + \cdots \\ A_o(x) = a_1 + a_3 x + a_5 x^2 + \cdots \end{cases} \Leftrightarrow \begin{cases} A(x_i) = A_e(x_i^2) + x_i A_o(x_i^2) \\ A(-x_i) = A_e(x_i^2) - x_i A_o(x_i^2) \end{cases}$

Plus-minus trick

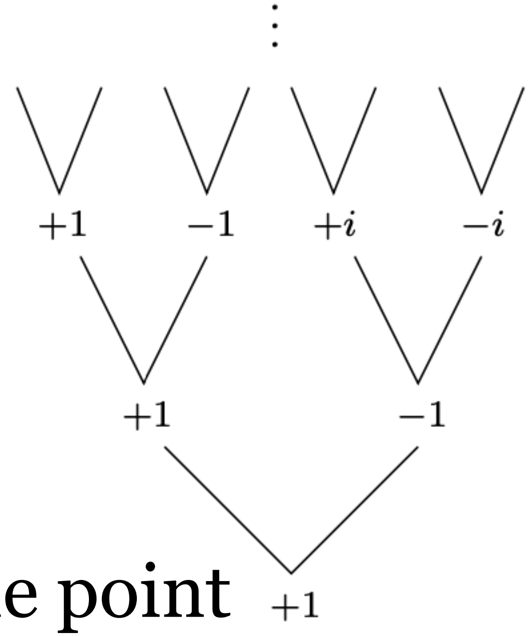
- Compute $A_e(x), A_o(x)$ (**half degree**) at $\boldsymbol{n}$ points $x_1^2, x_2^2, \dots, x_n^2$
- Then we can recover $A(\pm x_1), A(\pm x_2), \dots A(\pm x_n)$ in $O(n)$ times

- If we could compute $A_e(x), A_o(x)$ in this method recursively:
- $T(n) = 2T\left(\frac{n}{2}\right) + O(n) = O(n \log n)$

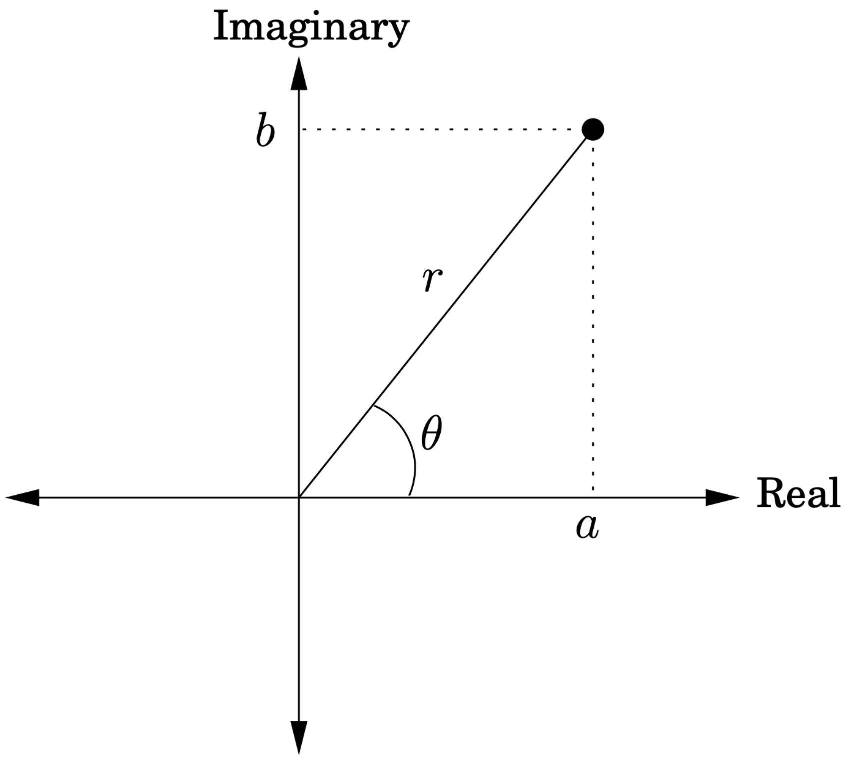
- Problem: If we want to use the same plus-minus trick, we need $x_1^2, x_2^2, \dots, x_n^2$ be themselves plus-minus pairs.
- **Try Complex numbers** !

Reverse engineer

- We can eventually reach the initial set of n points !
- Continuing in this manner
- The third level must consist of square roots of ± 1 :
- The second level must consist of its square roots:
- At the very bottom of the recursion, we have a single point



Review : complex plane



The complex plane

$z = a + bi$ is plotted at position (a, b).

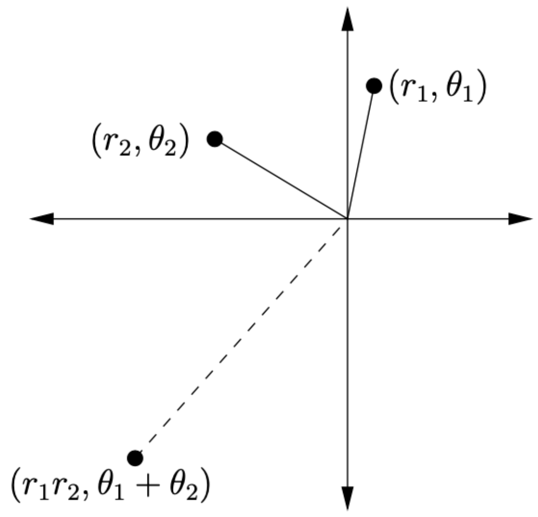
Polar coordinates: rewrite as $z = r(\cos\theta + i\sin\theta) = re^{i\theta}$, denoted (r, θ).

- ***length*** $r = \sqrt{a^2 + b^2}$.
- ***angle*** $\theta \in [0, 2\pi)$: $\cos\theta = a/r, \sin\theta = b/r$.
- θ can always be reduced modulo 2π.

Examples:

Number	-1	i	$5+5i$
Polar coords	$(1, \pi)$	$(1, \pi/2)$	$(5\sqrt{2}, \pi/4)$

Review : complex multiplying



Multiplying is easy in polar coordinates

Multiply the lengths and add the angles:

$(r_1, \theta_1) \times (r_2, \theta_2) = (r_1 r_2, \theta_1 + \theta_2)$.

For any $z = (r, \theta)$,

- $-z = (r, \theta + \pi)$ since $-1 = (1, \pi)$.
- If z is on the *unit circle* (i.e., $r = 1$), then $z^n = (1, n\theta)$.

$$
\begin{aligned}
& (r_1, \theta_1) \cdot (r_2, \theta_2) \\
&= r_1(\cos\theta_1 + i\sin\theta_1) \cdot r_2(\cos\theta_2 + i\sin\theta_2) \\
&= r_1 r_2[(\cos\theta_1\cos\theta_2 - \sin\theta_1\sin\theta_2) + i\,(\sin\theta_1\cos\theta_2 + \cos\theta_1\sin\theta_2) \\
&= r_1 r_2[\cos(\theta_1 + \theta_2) + i\sin(\theta_1 + \theta_2)] \\
&= (r_1 r_2, \theta_1 + \theta_2)
\end{aligned}
$$

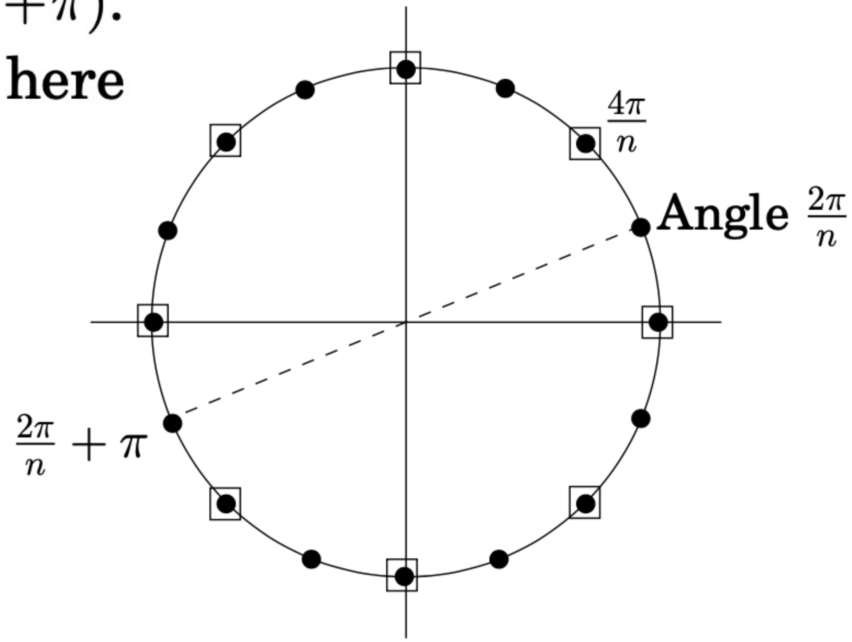
The complex n^{th} roots of unity

Consider n complex solutions to the equation $z^n = 1$

By the multiplication rule: solutions are $z = (1, \theta)$, for θ a multiple of $2\pi/n$ (shown here for $n = 16$).

For even n:

- **These numbers are *plus-minus paired*: $-(1, \theta) = (1, \theta + \pi)$.**
- **Their squares are the $(n/2)$nd roots of unity, shown here with boxes around them.**



The complex n^{th} roots of unity

- The complex numbers $1, \omega, \omega^2, \ldots, \omega^{n-1}$, where $\omega = e^{2\pi i/n}$
- If n is even, then:
  - They are plus-minus paired : $\omega^{\frac{n}{2}+j} = -\omega^j$
  - Squaring them produces the $\left(\frac{n}{2}\right)^{th}$ roots of unity

- If we start with these numbers for some n that is a power of 2
- Then all these sets of numbers for each level are plus-minus paired !
- divide-and-conquer

The fast Fourier transform

Figure 2.7 The fast Fourier transform (polynomial formulation)

```
function FFT(A, ω)
Input:  Coefficient representation of a polynomial A(x)
        of degree ≤ n − 1, where n is a power of 2
        ω, an nth root of unity
Output:  Value representation A(ω^0), ..., A(ω^{n−1})

if ω = 1:  return A(1)
express A(x) in the form A_e(x^2) + x A_o(x^2)
call FFT(A_e, ω^2) to evaluate A_e at even powers of ω
call FFT(A_o, ω^2) to evaluate A_o at even powers of ω
for j = 0 to n − 1:
   compute A(ω^j) = A_e(ω^{2j}) + ω^j A_o(ω^{2j})

return A(ω^0), ..., A(ω^{n−1})
```

Multiplying Process

Figure 2.5 Polynomial multiplication

```
Input:  Coefficients of two polynomials, A(x) and B(x), of degree d
Output:  Their product C = A · B
```

Selection

Pick some points $x_0, x_1, \ldots, x_{n-1}$, where $n \geq 2d + 1$

Evaluation

Compute $A(x_0), A(x_1), \ldots, A(x_{n-1})$ and $B(x_0), B(x_1), \ldots, B(x_{n-1})$

Multiplication

Compute $C(x_k) = A(x_k)B(x_k)$ for all $k = 0, \ldots, n-1$

Interpolation

Recover $C(x) = c_0 + c_1 x + \cdots + c_{2d} x^{2d}$

Matrix Reformulation

Evaluation:

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} = \underbrace{\begin{bmatrix} C(x_0) \\ C(x_1) \\ \vdots \\ C(x_n) \end{bmatrix}}_{\text{Target}}$$

$$\Downarrow$$

Interpolation:

$$\underbrace{\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix}}_{\text{Target}} = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix}^{-1} \begin{bmatrix} C(x_0) \\ C(x_1) \\ \vdots \\ C(x_n) \end{bmatrix}$$

Cont’d

$$M_n(\omega) = \left[\begin{array}{ccccc} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-1)} \\ & & \vdots & & \\ 1 & \omega^j & \omega^{2j} & \cdots & \omega^{(n-1)j} \\ & & \vdots & & \\ 1 & \omega^{(n-1)} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)(n-1)} \end{array}\right] \begin{array}{ll} \longleftarrow & \textbf{row for } \omega^0 = 1 \\ \longleftarrow & \omega \\ \longleftarrow & \omega^2 \\ & \vdots \\ \longleftarrow & \omega^j \\ & \vdots \\ \longleftarrow & \omega^{n-1} \end{array}$$

- $M_n(\omega)_{j,k} = \omega^{jk}$
- Inversion formula : $M_n(\omega)^{-1} = \frac{1}{n} M_n(\omega^{-1})$

Proof.

- Denote $M_n(\omega)$ as M, $M_n(\omega^{-1})$ as M' :

$$\frac{1}{n}(MM')_{j,k} = \frac{1}{n}\sum_{i=0}^{n-1} M_{j,i}M'_{i,k} = \frac{1}{n}\sum_{i=0}^{n-1} \omega^{ji}\omega^{-ik} = \frac{1}{n}\sum_{i=0}^{n-1} \omega^{i(j-k)}$$

- If $j = k$, then every element equals to 1 , so $\frac{1}{\mathrm{n}}(MM')_{j,k} = 1$
- Otherwise, it's a geometric series :

$$\frac{1}{n}(MM')_{j,k} = \frac{1}{n}\omega^{j-k}\sum_{i=0}^{n-1}\omega^{i} = \frac{1}{n}\omega^{j-k}\frac{1-\omega^{n}}{1-\omega} = 0$$

- Then $\frac{1}{\mathrm{n}}MM' = I \Rightarrow M_n(\omega)^{-1} = \frac{1}{n}M_n(\omega^{-1})$. ■

Interpolation : FFT

- $$\begin{bmatrix} C(x_0) \\ C(x_1) \\ \vdots \\ C(x_n) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{(n-1)} \\ 1 & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{(n-1)} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)(n-1)} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix}$$
- ⟨values⟩ = FFT (⟨coefficients⟩ , ω)

- $$\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} = \frac{1}{n} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^{-1} & \omega^{-2} & \cdots & \omega^{-(n-1)} \\ 1 & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{-(n-1)} & \omega^{-2(n-1)} & \cdots & \omega^{-(n-1)(n-1)} \end{bmatrix} \begin{bmatrix} C(x_0) \\ C(x_1) \\ \vdots \\ C(x_n) \end{bmatrix}$$
- ⟨coefficients⟩ = $\frac{1}{n}$ FFT (⟨values⟩ , ω^{-1})

Polynomial Multiplication

- Evaluation $A(x), B(x)$ by FFT: $O(n \log n)$
- Compute $C(x) = A(x)B(x)$ in value representation: $O(n)$
- Interpolation $C(x)$ by FFT: $O(n \log n)$
- The time complexity of polynomial multiplying becomes: $O(n \log n) + O(n) + O(n \log n) = O(n \log n)$