Lecture 4

Jan 14, 2022

Fast polynomial multiplication

Representing polynomials : $\left[\begin{array}{l}\text{coefficient representation} \\ \text{evaluation representation}\end{array}\right.$

Lemma [Interpolation]

Let $\alpha_1, \ldots, \alpha_n$ be distinct complex numbers.
Then for any polynomial $f(x)$ of $\deg \leq n-1$, f can be recovered uniquely given its evaluations at $\alpha_1, \ldots, \alpha_n$.

Proof:

1. Set up a linear system in the coefficients of f given the evaluations.

- Basic fact
- Worth noting

2. The constraint matrix is the Vandermonde matrix and hence is full rank.

So, the system has a solution.
Moreover, it is unique. $\boxtimes$

Q. Multiply polynomials given as a list of coefficients. Want the output as a list of coefficients.

Toy problem: Multiply polynomials given as a list of evaluations.

Input:
$$\left.\begin{array}{l} \alpha_1, \ldots, \alpha_t \\ f(\alpha_1), f(\alpha_2), \ldots, f(\alpha_t) \\ g(\alpha_1), g(\alpha_2), \ldots, g(\alpha_t) \end{array}\right\} \quad \deg(f), \deg(g) \leq n-1.$$

Output: Evaluation of the product fg at $\alpha_1, \ldots, \alpha_t$.

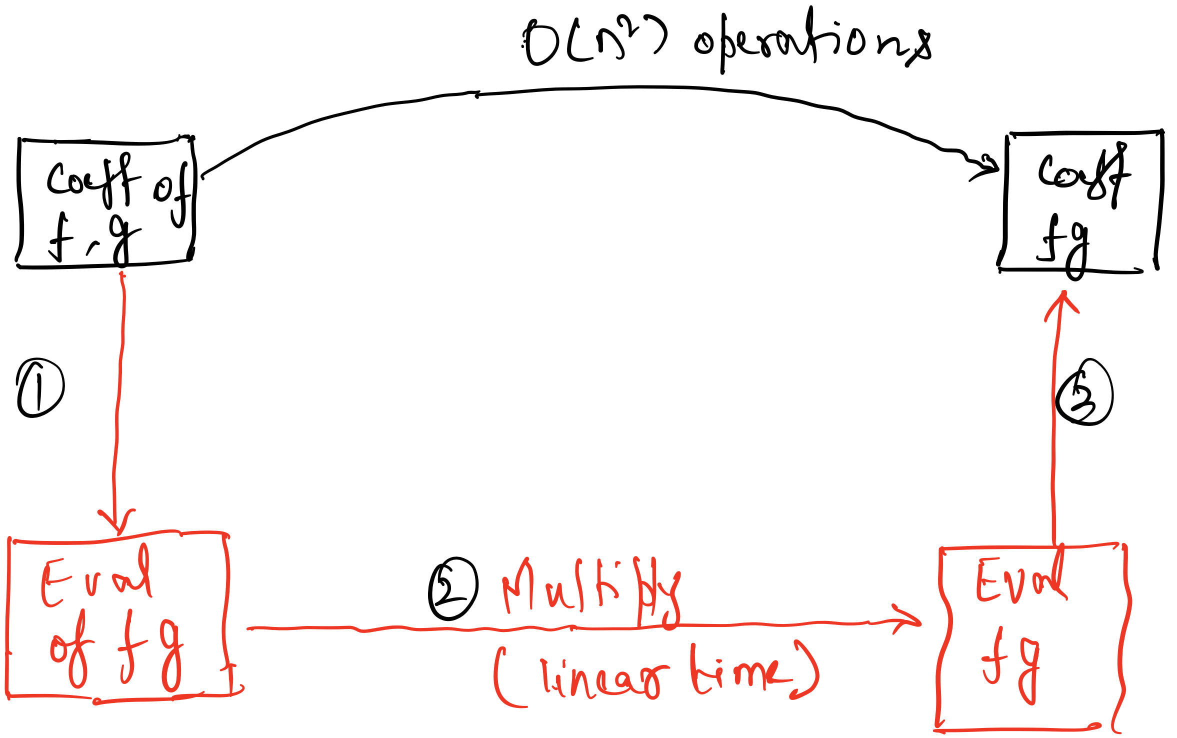
Algo: for $i = 1$ to t

$$\left.\begin{array}{l} fg(\alpha_i) \leftarrow f(\alpha_i) \cdot g(\alpha_i) \\ (f+g)(\alpha_i) \leftarrow f(\alpha_i) + g(\alpha_i) \end{array}\right\}$$

Runtime: $O(t)$ operations - linear time algo.

Note: $\deg(f\cdot g)$ could be as large as $2(n-1)$.
So, to recover fg from its evaluations on $\alpha_1, \ldots, \alpha_t$, we need $t \geq 2(n-1)+1$.

High level sketch for a potential algorithm



Need steps ①, ③ to be fast.
$O(n \log n)$ operations.

Focus on step ①

Multipoint Evaluation

Q: Given $f(x) = f_0 + f_1 \cdot x + f_2 \cdot x^2 + \cdots + f_{n-1} x^{n-1}$.

as a list of coefficients.

Obtain evaluations of f at $\boxed{2(n-1)+1}$ distinct points.

↳ $\deg(fg)+1$

Need this to recover fg in step 3.

At what points do we evaluate f?

— Any choice suffices for the plan.

Points are $\alpha_1 \ldots \ldots \alpha_t$. $\boxed{t > 2(n-1)}$

Want: $f(\alpha_1), f(\alpha_2), \ldots, f(\alpha_t)$

Naive: - Iterate from $i=1$ to t.
- Evaluate f at α_i }

Cost: $t \cdot$ cost of single evaluation $= O(tn) = O(n^2)$

$f(\alpha) = f_0 + f_1\alpha + f_2\alpha^2 + \cdots + f_{n-1}\alpha^{n-1}$.

$\left.\begin{array}{l}\text{- Iteratively compute powers of } \alpha \\ \text{- Add stuff up.}\end{array}\right\} O(n)$.

Idea: Come up with a careful choice of $\alpha_1, \ldots, \alpha_t$ for which Multipoint evaluation of f on $\alpha_1, \ldots, \alpha_t$ can be done $O(n \log n)$ operations.

1. Choice of α's.

k^{th} roots of 1

$\beta \in \mathbb{C}$ is a k^{th} root of 1 if $\beta^k = 1$.

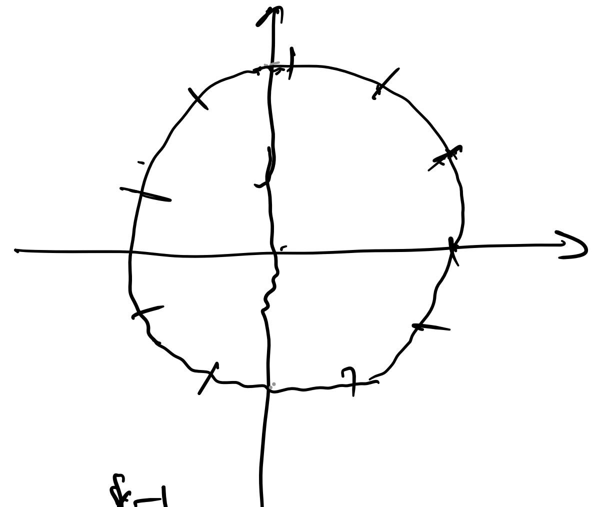
$k=1$: 1

$k=2$: $-1, 1$

$k=3$: $1, \omega_3, \omega_3^2$ $\qquad \omega_3 = e^{i \cdot 2\pi/3}$

$k=4$: $1, -1, i, -i$ $\qquad \omega_4 = e^{i\,2\pi/4} = e^{i\pi/2}$

k : $\{\omega_k^i : i = 0, 1, \dots, k-1\}$ $\qquad \omega_k = e^{i\,2\pi/k}$.



Properties:

(1) For all $k \geq 2$,
sum of all k^{th} roots of 1
equals 0.

$$\left(\sum_{i=0}^{k-1} \omega_k^i = 1 + \omega_k + \omega_k^2 + \dots + \omega_k^{k-1}\right)$$

$$= \frac{\omega_k^k - 1}{\omega_k - 1} = 0.$$

2) $\boxed{k\text{ - even}}$.

Let $S_k = \{1, (\omega_k)^2, (\omega_k^2)^2, \ldots, (\omega_k^{k-1})^2\}$

$= \{1, (\omega_k^2), (\omega_k^2)^2, \ldots, (\omega_k^2)^{k-1}\}$ ✓

Then: 1) $|S_k| = k/2$

2) S_k is the set of all $k/2$ th roots of 1.

Pf: $\omega_k = e^{i\,2\pi/k}$

$\omega_k^2 = e^{i \cdot \frac{2\pi}{k} \cdot 2} = e^{i\left(\frac{2\pi}{k/2}\right)} = \omega_{k/2}$.

Know $\{1, \omega_{k/2}, \omega_{k/2}^2, \ldots, \omega_{k/2}^{k/2-1}\}$ is the set of all $k/2$ th roots of 1.

$S_k = \{1, \omega_{k/2}, \omega_{k/2}^2, \ldots \qquad , \omega_{k/2}^{k-1}\}$

$$= \{ 1, \omega_{k/2}, \omega_{k/2}^2, \ldots, \omega_{k/2}^{k/2-1}, \underbrace{\omega_{k/2}^{k/2}}_{1}, \underbrace{\omega_{k/2}^{k/2+1}}_{\omega_{k/2}}, \ldots \}$$

$$\omega_{k/2}^{k/2+i} = \omega_{k/2}^{k/2} \cdot \omega_{k/2}^{i} = 1 \cdot \omega_{k/2}^{i}$$

$k = 4.$ $\{ 1, -1, i, -i \}$

$$S_4 = \{ 1^2, (-1)^2, i^2, (-i)^2 \}$$

$$= \{ 1, 1, -1, -1 \}$$

$|S_4| = 2$

Takeaways

(1) Set $\omega_k = e^{i 2\pi/k}$, then, $\{ \omega_k^i : i = 0, \ldots, k-1 \}$

are all k^{th} roots of 1.

(2) Sum of these roots equals 0.

(3) Squaring k^{th} roots gives us $k/2^{th}$ roots of 1.
(k = even)

- Choice of $\alpha_1 \dots \alpha_t$ $\quad t \geq 2(n-1)+1$.
- Pick $\alpha_1 \dots \alpha_t$ to be all the t^{th} roots of 1.

[set t to be the power of 2 closest to and larger than $2(n-1)+1$]

- $\{ \omega_t^i : i = 0, \dots, t-1 \}$

Q: Evaluate f on $\{ \omega_t^i : i = 0, \dots t-1 \}$ in nearly linear time. ($O(n \log n)$).

Discrete Fourier transform

$$f(w_t) = f_0 + f_1 w_t + f_2 w_t^2 + \cdots + f_{n-1} \omega_t^{n-1}$$

$$w_t = e^{i 2\pi / t}$$

Fast Fourier Transform

$n-1-\text{even}$

$$f = f_0 + f_1 x + f_2 x^2 + \cdots + f_{n-1} x^{n-1}$$

$$f_{even} := f_0 + f_2 x + f_4 x^2 + \cdots + f_{n-1} x^{\frac{n-1}{2}}.$$

$$f_{odd} := f_1 + f_3 x + f_5 x^2 + \cdots + f_{n-2} x^{\frac{n-3}{2}}$$

$$\deg(f_{even}), \deg(f_{odd}) \leq \frac{n-1}{2}.$$

Claim 1: $f(x) = f_{even}(x^2) + x \cdot f_{odd}(x^2)$.

$$f(w_t^i) = f_{even}\left((w_t^2)^i\right) + w_t^i \cdot f_{odd}\left((w_t^2)^i\right)$$

Conclusion: To evaluate f on $\{w_t^i : i=0,\ldots,t-1\}$ it suffices to evaluate f_{odd} and f_{even} on $S_t = \{w_t^{2i} : i=0,\ldots,t-1\}$.

But from prev discussion:

S_t = set of all $t/2$ th roots of 1

$$= \{w_{t/2}^i : i=0,\ldots,t/2-1\}.$$

Algo FFT

(1) Base case - $\deg(f) = 1$ ⟶ trivial

(2) construct f_{even}, f_{odd}

3) Recursively evaluate f_{even}, f_{odd} on all $t/2^{th}$ roots of 1.

(4) Recover f on $\{w_t^i : i \in \{0, \dots t-1\}\}$ via claim 1.

$[t = O(n)]$

Running time:

$$T(n) \leq 2T(n/2) + O(n)$$

$$T(n) \leq O(n \log n)$$ ✓

Correctness.

- Induction on deg.
-