Admin: HW 0 on Crowdmark
due Sept 16 @ 3:30pm

Today: Arithmetic on polynomials

Given $f, g \in \mathbb{C}[x]$, how to compute

- $f(x) + g(x)$
- $f(x)\, g(x)$
- $f(x) / g(x)$.

Measure of complexity: arithmetic ops in $\mathbb{C}$

Addition

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

$$g(x) = g_{n-1} x^{n-1} + \cdots + g_0$$

$$h(x) = f(x) + g(x)$$
$$= (f_{n-1} + g_{n-1}) x^{n-1} + (f_{n-2} + g_{n-2}) x^{n-2} + \cdots + (f_0 + g_0)$$

<u>ADD(f, g):</u>

```
for i ← 0 to n-1
    h_i ← f_i + g_i
h(x) := Σ_i h_i x^i
Output h(x)
```

Cost: $O(n)$ arithmetic ops
Correctness: definition of $f+g$.

Multiplication

$$f(x)\, g(x) = f_n g_n x^{2n} + (f_n g_{n-1} + f_{n-1} g_n) x^{2n-1} + \cdots$$
$$= \sum_{k=0}^{2n} \left(\sum_{i+j=k} f_i g_j \right) x^k$$

Obvious algorithm: $O(n^2)$ ops

Do better? Use a different representation

<u>Lemma (polynomial interpolation)</u>

$f \in \mathbb{C}[x]$, $\deg(f) < n$.

$a_1, \ldots, a_n \in \mathbb{C}$ distinct

Coeffs of f uniquely determined by $f(a_1), \ldots, f(a_n)$.

<u>Proof</u> $f(x) = \sum_{i<n} f_i x^i$

$$f(a_j) = \sum_{i<n} f_i a_j^i$$

Vandermonde matrix

$$\begin{pmatrix} f(a_1) \\ \vdots \\ f(a_n) \end{pmatrix} = \begin{pmatrix} 1 & a_1 & a_1^2 & \cdots & a_1^{n-1} \\ 1 & a_2 & a_2^2 & \cdots & a_2^{n-1} \\ & & \vdots & & \\ 1 & a_n & a_n^2 & \cdots & a_n^{n-1} \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_{n-1} \end{pmatrix}$$

$$\det\left(\nearrow \right) = \prod_{i<j} (a_i - a_j)$$
$$\neq 0$$

$$\Rightarrow \begin{pmatrix} 1 & a_1 & & a_1^{n-1} \\ \vdots & \vdots & \cdots & \\ 1 & a_n & & a_n^{n-1} \end{pmatrix} \text{ invertible}$$

$$\begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & a_1 & a_1^2 & \cdots & a_1^{n-1} \\ 1 & a_2 & a_2^2 & \cdots & a_2^{n-1} \\ & & \vdots & & \\ 1 & a_n & a_n^2 & \cdots & a_n^{n-1} \end{pmatrix}^{-1} \begin{pmatrix} f(a_1) \\ f(a_2) \\ \vdots \\ f(a_n) \end{pmatrix}$$

□

Idea to compute $h(x) := f(x)\, g(x)$

1) Evaluate h at points $a_1, \ldots, a_{2n}$
2) Recover coefficients.

<u>EVAL(f, a):</u>

```
compute a, a^2, ..., a^(n-1)
eval ← 0
for i ← 0 to n-1
    eval ← eval + f_i a^i
output eval
```

$O(n)$ ops → Evaluate f at $a_1, \ldots, a_{2n}$ in $O(n^2)$ ops

<u>Discrete Fourier transform</u>

$\omega := \exp\left(\frac{2\pi i}{n}\right)$ <u>primitive</u> <u>root of unity</u>

primitive: $\omega^k \neq 1$ for $k < n$. root of unity: $\omega^n = 1$

$$\text{DFT}: \mathbb{C}^n \to \mathbb{C}^n$$
$$(f_0, f_1, \ldots, f_{n-1}) \mapsto (f(1), f(\omega), \ldots, f(\omega^{n-1}))$$

FFT: compute DFT in $O(n \log n)$ ops

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

- Compute DFT via divide & conquer
- Inverse DFT: look at the matrix
$$\begin{pmatrix} 1 & \omega^{-1} & \omega^{-2} & \cdots & \omega^{-(n-1)} \\ 1 & \omega^{-2} & (\omega^{-2})^2 & \cdots & \\ \vdots & & & & \\ 1 & & & \ddots & \end{pmatrix}$$
  Same as the DFT, but with ω^{-1} as the root of unity.
- Define polynomial division w/ remainder. State $O(n^2)$ time algo. State $O(n \log n)$ algo (HW)
- Define multipoint evaluation.
- $f(x) = q(x)(x - a) + f(a)$
- Subproduct tree
  - Go up: polynomial multiplication
  - Go down: div with remainder
  - Punchline $O(M(n) \log n)$ ops for multipoint eval
- Remind definition of greatest common divisor (GCD).
- Next time: Euclidean algorithm & polynomial GCDs.