Polynomial multiplication problem

- From geometry we now move to algebra, where divide-conquer paradigm would be even more nontrivial.

- A degree $<n$ polynomial is given by n numbers. Say, univariate, integral
$$f(x) = a_0 + a_1 x + \ldots + a_{n-1} x^{n-1} = \sum_{i=0}^{n-1} a_i x^i .$$

- Problem: How do we multiply them?

Input: Given $f = \sum_{i<n} a_i x^i$ & $g = \sum_{i<n} b_i x^i$.

Output: The product $h = f \times g = \sum_{i<2n-1} c_i x^i$

▷ $\forall\, 0 \le j < 2n-1,\ c_j = \sum_{0 \le i \le j} a_i b_{j-i}$ ← $O(j)$ time

⇒ Brute-force algorithm:
$O(n^2)$ time.

- Problem applications: [staple in all areas of CS!]
    - As natural & important as sorting!
    - Signal processing (Discrete Fourier Transform)
    - Data storage & communications (error-correcting codes)
    - Integer multiplication/division.

- Is there a geometry behind polynomials?

    - $f = a_0 + a_1 x$ defines a line.
    - $f = a_0 + a_1 x + a_2 x^2$ " " parabola.
    - $f = \sum_{i<n} a_i x^i$ " " curve of degree $<n$ in $\mathbb{R}^2$.

- How many points define f?

▷ n distinct points suffice to uniquely specify f.

Proof: • Consider the n equations in n unknowns:

$$\sum_{i<n} a_i r_0^i = f(r_0),$$
$$\vdots$$
$$\sum_{i<n} a_i r_{n-1}^i = f(r_{n-1}),$$

where $\{(r_j, f(r_j)) \mid 0 \leq j < n\}$ are the distinct points in the plane.

• The above system is linear & can be rewritten in matrix form:

$$M \rightarrow \begin{pmatrix} 1 & r_0 & r_0^2 & \cdots\cdots & r_0^{n-1} \\ 1 & r_1 & r_1^2 & \cdots\cdots & r_1^{n-1} \\ & \vdots & & & \vdots \\ 1 & r_{n-1} & r_{n-1}^2 & \cdots & r_{n-1}^{n-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} f(r_0) \\ f(r_1) \\ \vdots \\ f(r_{n-1}) \end{pmatrix}.$$

• Exercise: M^{-1} exists. $|M|$ has a simple form.

$\Rightarrow (a_0, a_1, \ldots, a_{n-1})$ exists uniquely.

$\square$

- Thus, we can work with polynomials in the points representation:

  $\{(r_j, f(r_j)) \mid j\}$
- Fix n distinct points $S := \{r_0, r_1, \ldots, r_{n-1}\}$.
- How does it help in computing the product h of f & g?

  - Computing $(r_j, f(r_j) \times g(r_j))$ takes only $O(1)$ time.

  - If we have this for $0 \leq j < 2n-1$, then it uniquely represents $h = f \times g$.

  $\Rightarrow$ Multiplication takes $O(n)$ time in the functional representation.

- Qn: How do we move across the two types of representations?

- Brute-force algorithm: $O(n^2)$ time!

- A better idea is from the times of Gauss; using divide-conquer.

- How to divide? Halving the points??

- Idea 0: We could reduce f to two degree $n/2$ polynomials as:

$$f = \sum_{i<n} a_i x^i = \Big(\sum_{i<\frac{n}{2}} a_i x^i\Big) + \Big(\sum_{\frac{n}{2}\le i<n} a_i x^i\Big)$$

$$= \Big(\sum_{i<n/2} a_i x^i\Big) + x^{n/2}\cdot\Big(\sum_{\frac{n}{2}\le i<n} a_i x^{i-\frac{n}{2}}\Big).$$

• One could recurse this way & try to recover the product $f \times g$.

Karatsuba (1960) analyzed this to get an algorithm of time complexity $O(n^{\lg 3})$.

- Idea 1: Divide the monomials in f into odd & even exponents:

$$f = (a_0 + a_2 x^2 + a_4 x^4 + \ldots\ldots) + (a_1 x + a_3 x^3 + a_5 x^5 + \ldots\ldots)$$

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

- Now to evaluate $f(x)$ on the n points S we the strategy:

$S^2 := \{r_i^2 \mid i\}$

  - evaluate $f_{even}(x)$ on S^2,
  - " $f_{odd}(x)$ " ", &
  - Combine.

- The last step takes $O(n)$ time.
  What about the first two calls?

Warning: It seems like we have got to a recurrence of $T(n) = 2\cancel{T(n/2)} + O(n)$.
But, this is not correct as the points (of evaluation) are not reducing; it is only the degree that is halved.

$\Rightarrow$ At the base of the recursion we have n linear polynomials, each to be evaluated on n points.

$\Rightarrow$ $O(n) \times n = O(n^2)$ time!

- Is there an S s.t. $|S| = n$, $|S^2| = n/2$, $|S^4| = n/4$, ..., $|S^{2^\ell}| = n/2^\ell$?

- <u>Simpler question</u>: Suppose $n = 2^\ell$, a 2-power. Can you find a set S s.t. $S^n = 1$?

- <u>Answer</u>: Consider the n-th primitive root of unity $\zeta_n := e^{2\pi i/n} \in \mathbb{C}$ (complex).
$$= \left(\cos\frac{2\pi}{n} + i \sin\frac{2\pi}{n}\right)$$
where $i := \sqrt{-1}$.

- Egs. $\zeta_1 = 1$, $\zeta_2 = -1 = e^{\pi i}$, $\zeta_4 = \sqrt{-1} = e^{\pi i/2}$, $\zeta_8 = \sqrt[4]{-1} = e^{\pi i/4}$, ...

<u>Lemma</u>:
- $S = \{\zeta_n^j \mid 0 \le j < n\}$ are n distinct n-th roots of unity in $\mathbb{C}$.
- S^2 are $n/2$ distinct $(n/2)$-th roots of unity.
- S^4 " $n/4$ " $(n/4)$-th " " ".

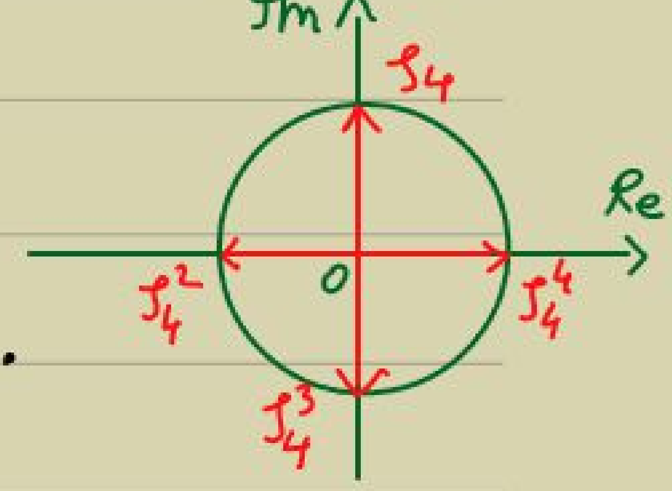
  ⋮

- $S^n = \{1\}$.

<u>Proof</u>: Exercise. $\square$

- These points have a nice geometric meaning.

Eg. $\zeta_4 = \sqrt{-1}$ divides the unit circle into 4 equal parts:



$\Rightarrow$ Square of it gives

$\{\zeta_4^2, \zeta_4^4\} = \{-1, 1\} = \{\zeta_2, \zeta_2^2\}$.

Use $S = \{\zeta_{2n}^j \mid 0 \le j < 2n\}$ in the algorithm

[Fast Fourier Transform, using ζ_{2n}, on $2n$ dimensions]

- While evaluating $f = f_{even}(x^2) + x \cdot f_{odd}(x^2)$ on S, by divide-conquer, now both the degree & the set-size gets halved.

$\Rightarrow$ The recurrence for time is indeed: $T(n) = 2T(n/2) + O(n)$

$\Rightarrow T(n) = O(n \lg n)$.

- Broadly, polynomial multiplication involves:

f, g by coefficients $\xrightarrow{FFT(\zeta_{2n})}$ f, g by evaluations $\Longrightarrow$ $h = f \times g$ by evaluations $\xrightarrow{FFT(\zeta_{2n}^{-1})}$ h by coefficients

Time: $O(n \lg n) + O(n) + O(n \lg n)$

Exercise: Show that the last step can be achieved in a way similar to the first.

Theorem [FFT by Cooley-Tukey '65]: Degree n polynomials can be multiplied, over the complex, in $O(n \lg n)$ time.

Exercise: How do you multiply the complex coefficients in a fast way?

Exercise: How do you multiply over other rings where ζ_n does not exist?