Polynomials and Fast Fourier Transform (FFT)

Polynomials

$A(x) = \sum_{i=0}^{n-1} a_i x^i$ a polynomial of degree n-1

Evaluate at a point x = b in time ?

Polynomials

$A(x) = \sum_{i=0}^{n-1} a_i x^i$ a polynomial of degree n-1

Evaluate at a point x = b in time O(n): Horner's rule:
Compute a_{n-1} x,

$a_{n-2} + a_{n-1}x^2$,

$a_{n-3} + a_{n-2}\, x + a_{n-1}\, x^3$

…

Each step O(1) operations, multiply by and add coefficient.

There are ≤ n steps. ➔ O(n) time

Summing Polynomials

$\sum_{i=0}^{n-1} a_i x^i$ a polynomial of degree n-1

$\sum_{i=0}^{n-1} b_i x^i$ a polynomial of degree n-1

$\sum_{i=0}^{n-1} c_i x^i$ the sum polynomial of degree n-1

$c_i = a_i + b_i$

Time O(n)

How to multiply polynomials?

$\sum_{i=0}^{n-1} a_i x^i$ a polynomial of degree n-1

$\sum_{i=0}^{n-1} b_i x^i$ a polynomial of degree n-1

$\sum_{i=0}^{2n-2} c_i x^i$ the product polynomial of degree n-1

$c_i = \sum_{j \le i} a_j b_{i-j}$

Trivial algorithm: time $O(n^2)$
FFT gives time $O(n \log n)$

Polynomial representations

Coefficient: $(a_0, a_1, a_2, \ldots a_{n-1})$

Point-value: have points $x_0, x_1, \ldots x_{n-1}$ in mind

Represent polynomials A(X) by pairs

$\{ (x_0, y_0), (x_1, y_1), \ldots \}$ $A(x_i) = y_i$

To multiply in point-value, just need O(n) operations.

Approach to polynomial multiplication:

A, B given as coefficient representation

1) Convert A, B to point-value representation

2) Multiply C = AB in point-value representation

3) Convert C back to coefficient representation

2) done esily in time O(n)

FFT allows to do 1) and 3) in time O(n log n).

Note: For C we need 2n-1 points; we'll just think “n”

From coefficient to point-value:

$$\begin{vmatrix} y_0 \\ y_1 \\ \cdots \\ \cdots \\ \cdots \\ y_{n-1} \end{vmatrix} = \begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{n-1} \end{pmatrix} \begin{vmatrix} a_0 \\ a_1 \\ \cdots \\ \cdots \\ \cdots \\ a_{n-1} \end{vmatrix}$$

From point-value representation, note above matrix is invertible (if points distinct)

Alternatively, Lagrange's formula

We need to evaluate A at points $x_1 \ldots x_n$ in time $O(n \log n)$

Idea: divide and conquer:

$A(x) = A^0(x) + x\, A^1(x)$
where $A^0(x)$ has the even-degree terms, $A^1(x)$ the odd

Example:

$A = a_0 + a_1 x + a_2 x^2 + a_3 x^3$

$A^0(x) = a_0 + a_2 x^2$
$A^1(x) = a_1 x + a_3 x^3$

How is this useful?

We need to evaluate A at points $x_1 \ldots x_n$ in time $O(n \log n)$

Idea: divide and conquer:

$A(x) = A^0(x) + x A^1(x)$
where $A^0(x)$ has the even-degree terms, $A^1(x)$ the odd

Note: $A(x) = A^0(x^2) + x A^1(x^2)$

If my points are $x_1, x_2, x_{n/2}, -x_1, -x_2, -x_{n/2}$

I just need the evaluations of A^0, A^1 at points $x_1^2, x_2^2, x_{n/2}^2$

$T(n) \leq 2\,T(n/2) + O(n)$, with solution $O(n \log n)$. Are we done?

We need to evaluate A at points $x_1 \ldots x_n$ in time $O(n \log n)$

Idea: divide and conquer:

$A(x) = A^0(x) + x A^1(x)$
where $A^0(x)$ has the even-degree terms, $A^1(x)$ the dd

Note: $A(x) = A^0(x^2) + x A^1(x^2)$

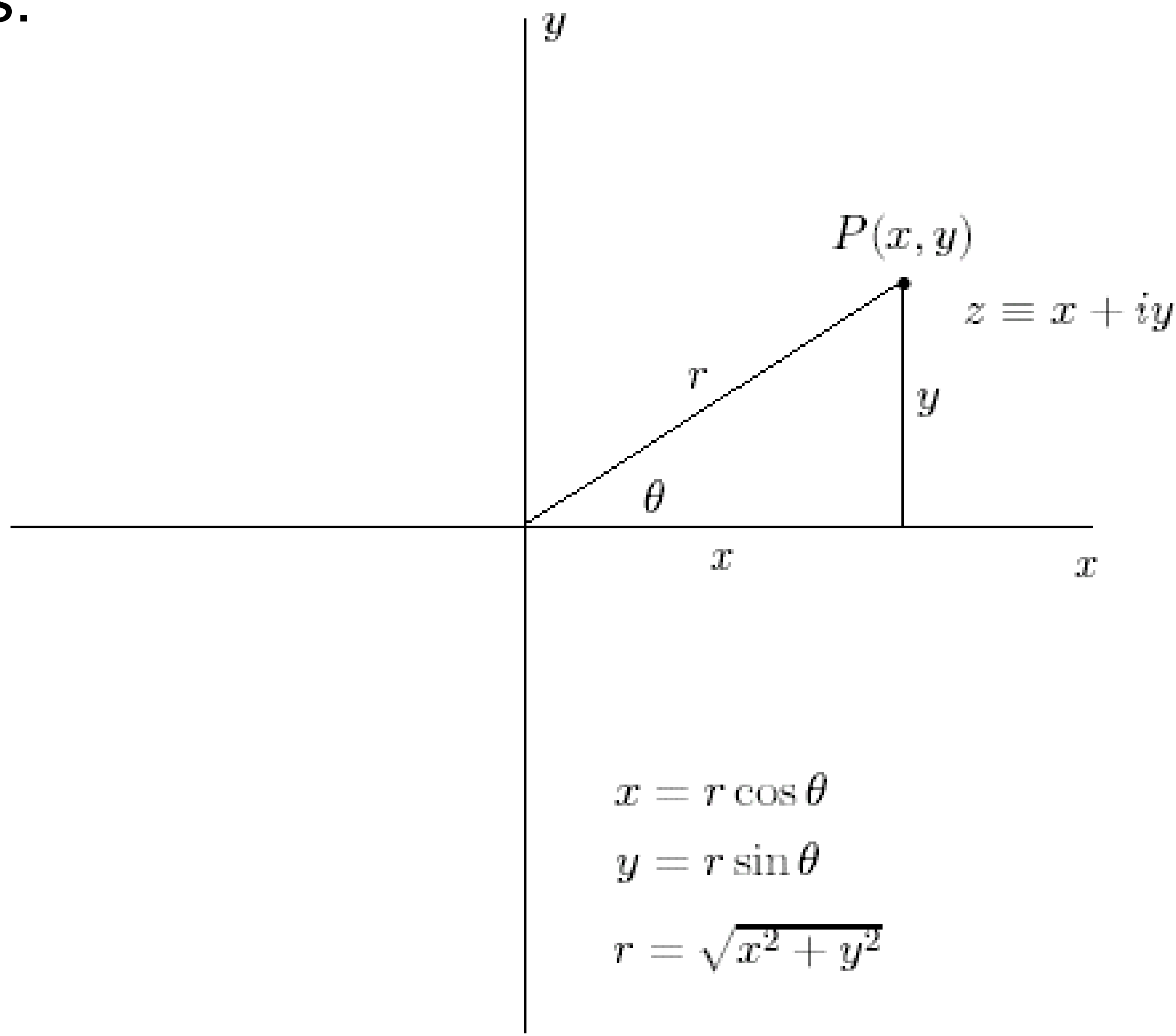
If my points are $x_1, x_2, x_{n/2}, -x_1, -x_2, -x_{n/2}$

I just need the evaluations of A^0, A^1 at points $x_1^2, x_2^2, x_{n/2}^2$

$T(n) \leq 2\,T(n/2) + O(n)$, with solution $O(n \log n)$. Are we done?

Need points which can be iteratively decomposed in + and -

Complex numbers:



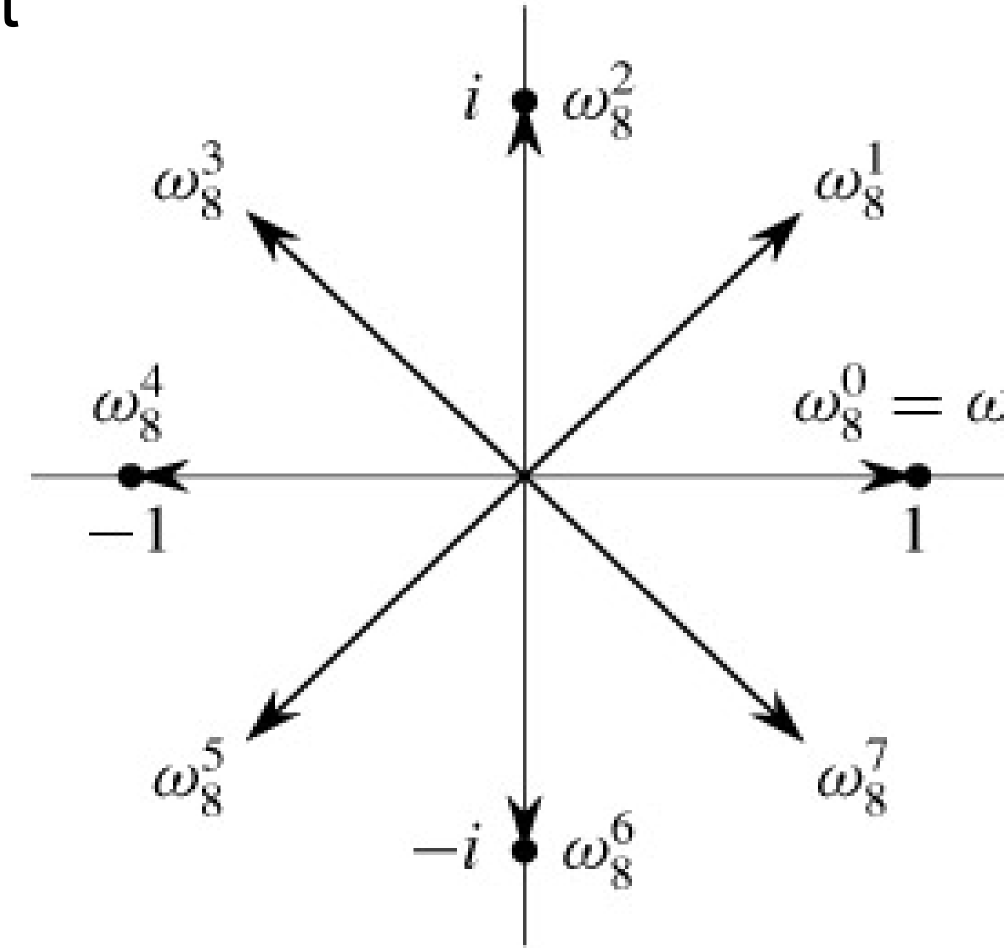
$$x = r\cos\theta$$

$$y = r\sin\theta$$

$$r = \sqrt{x^2 + y^2}$$

ω_n = n-th primitive root of unity

$\omega_n^0, \dots, \omega_n^{n-1}$
n-th roots of unity



We evaluate polynomial A
of degree n-1
at roots of unity
$\omega_n^0, \dots, \omega_n^{n-1}$

Fact: The n squares of the n-th roots of unity are:
first the n/2 n/2-th roots of unity,
then again the n/2 n/2-th roots of unity.

➔ from coefficient to point-value in O(n log n) (complex) steps

It only remains to go from point-value to coefficient represent.

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n & \omega_n^2 & \omega_n^3 & \cdots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \omega_n^6 & \cdots & \omega_n^{2(n-1)} \\ 1 & \omega_n^3 & \omega_n^6 & \omega_n^9 & \cdots & \omega_n^{3(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \omega_n^{3(n-1)} & \cdots & \omega_n^{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{n-1} \end{pmatrix}$$

F

We need to invert F

It only remains to go from point-value to coefficient represent.

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n & \omega_n^2 & \omega_n^3 & \cdots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \omega_n^6 & \cdots & \omega_n^{2(n-1)} \\ 1 & \omega_n^3 & \omega_n^6 & \omega_n^9 & \cdots & \omega_n^{3(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \omega_n^{3(n-1)} & \cdots & \omega_n^{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{n-1} \end{pmatrix}$$

Fact: $(F^{-1})_{j,k} = \omega_n^{-jk} \,/\, n$ Note, $j,k \in \{0,1,\ldots, n-1\}$

To compute inverse, use FFT with ω^{-1} instead of ω, then divide by n.