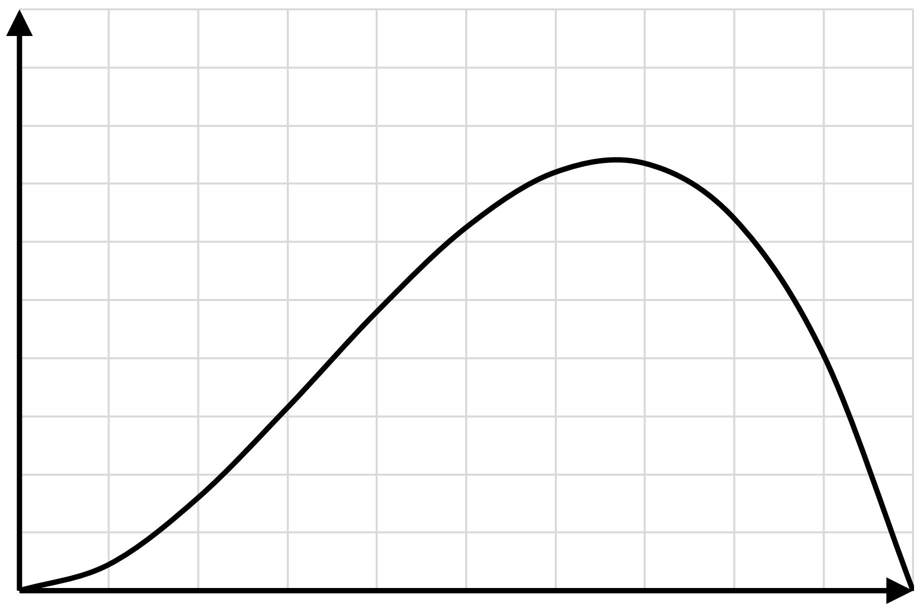
Lecture 4
Polynomial multiplication



Administrative corner

I hope everyone had a nice Labor Day!

Homeworks:

1. Hwk 1 due **tonight**. Good luck!
2. Hwk 2 due next Monday.

New discussion sections:

- Tuesday 3-4pm
- Thursday 10-11am
- Thursday 11am-12pm

Office hours:

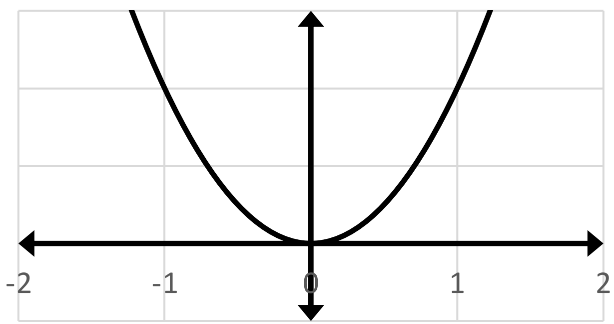
From now on
- using OH queue (oh.cs170.org)
- special system for Hwk Parties

(see weekly post)

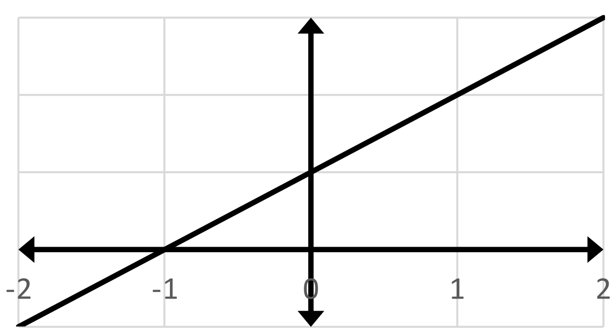
Lectures 1&2: How do you multiply **numbers** quickly?

Lecture 3: How do you multiply **matrices** quickly?

Today: How do you multiply **univariate polynomials** quickly?

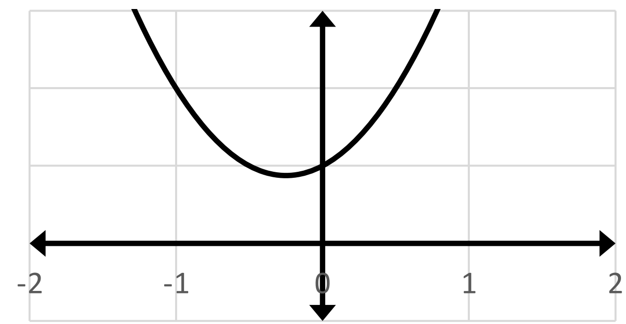


$p(x) = x^2$

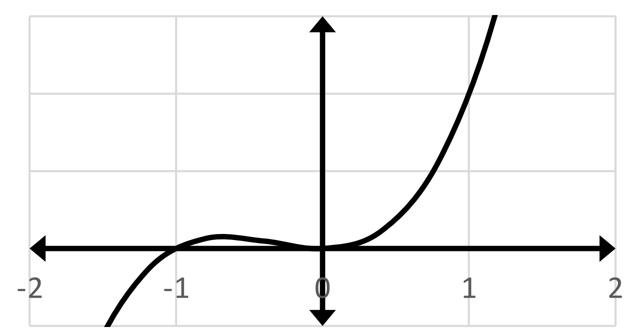


$q(x) = 0.5 + 0.5x$

Addition: $p(x) + q(x) = 0.5 + 0.5x + x^2$



Multiplication: $p(x) \cdot q(x) = 0.5x^2 + 0.5x^3$



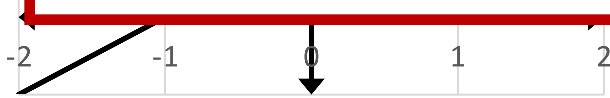
Lectures 1&2: How do you multiply **numbers** quickly?

Warning!

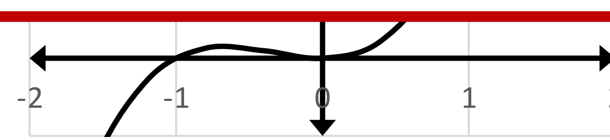
This is a **challenging** lecture.

It contains a lot of different ideas.

Please ask questions!



$q(x) = 0.5 + 0.5x$



Representing polynomials

Let $p(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_{n-1} x^{n-1}$ ⟵ degree n-1

Its **coefficient representation** is the array of real numbers

$$(p_0,\ p_1,\ p_2, \ldots, p_{n-1})$$ ⟵ (the input)

Modeling assumptions

1. Think of $\boldsymbol{n}$ as **large**. (Say, $\boldsymbol{n = 10^{10}}$.)
2. Think of $\boldsymbol{p_0, \ldots, p_{n-1}}$ as **small**. (32-bit float)

   So all arithmetic operations take O(1) time.

Goal: Measure runtime as function of $\boldsymbol{n}$. E.g. $\boldsymbol{T(n) = O(n^2)}$

Task 1: adding polynomials

Input: 1. $p(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_{n-1} x^{n-1}$

2. $q(x) = q_0 + q_1 x + q_2 x^2 + \cdots + q_{n-1} x^{n-1}$

Output: the polynomial $p(x) + q(x)$ (i.e. its coefficients)

Q: How fast can you do this?

$$
\begin{array}{rrcccccccc}
 & p(x) & = & p_0 & + & p_1 x & + & p_2 x^2 & + \cdots + & p_{n-1} x^{n-1} \\
+ & q(x) & = & q_0 & + & q_1 x & + & q_2 x^2 & + \cdots + & q_{n-1} x^{n-1} \\
\hline
\end{array}
$$

$$p(x) + q(x) = (p_0 + q_0) + (p_1 + q_1)x + (p_2 + q_2)x^2 + \cdots + (p_{n-1} + q_{n-1})x^{n-1}$$

A: In $\boldsymbol{O(n)}$ time.

Note: Like adding integers, but simpler. Why? No carries!

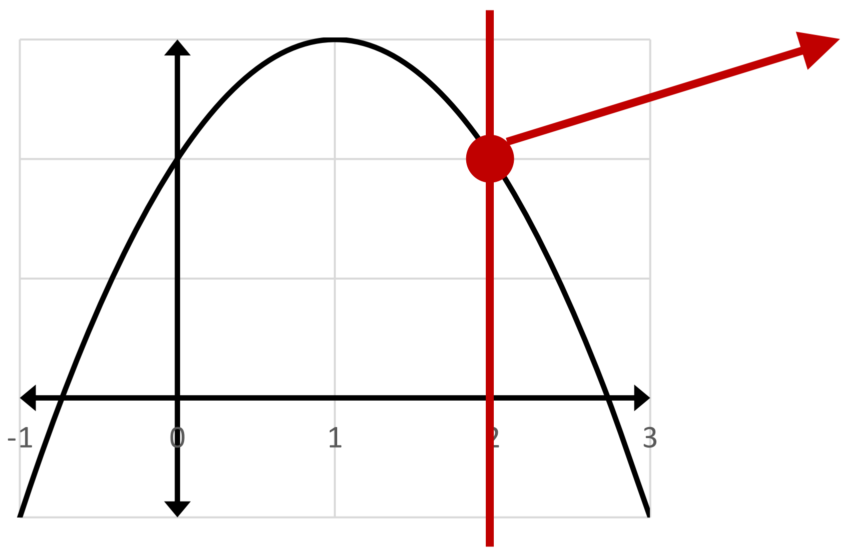
Task 2: evaluating polynomials

Input: 1. $p(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_{n-1} x^{n-1}$

2. a real number $\alpha \in \mathbb{R}$

Output: $p(\alpha) = p_0 + p_1 \alpha + p_2 \alpha^2 + \cdots + p_{n-1} \alpha^{n-1} \in \mathbb{R}$

$\alpha = 2$



$$p(\alpha) = 1 + 2 - 0.5 \cdot 2^2 = 1$$

$p(x) = 1 + x - 0.5x^2$

Q: How fast can you do this?

Task 2: evaluating polynomials

Input: polynomial $p(x)$ and $\alpha \in \mathbb{R}$

Output: $p(\alpha)$

Algorithm #1: time $O(n^2)$

redundant work!

$$\begin{array}{c}O(n)\\ \text{adds}\end{array}\left\{\begin{array}{ll} \quad p_0 & 0 \text{ mults}\\ + \; p_1 \cdot \alpha & 1 \text{ mults}\\ + \; p_2 \cdot \alpha \cdot \alpha & 2 \text{ mults}\\ + \; p_3 \cdot \alpha \cdot \alpha \cdot \alpha & 3 \text{ mults}\\ \qquad \dots & \\ + \; p_{n-1} \cdot \alpha \cdot \alpha \cdots \alpha & (n-1) \text{ mults}\end{array}\right.$$

$$\begin{array}{ll} \hline p(\alpha) & O(n^2) \text{ mults}\end{array}$$

Task 2: evaluating polynomials

Input: polynomial $p(x)$ and $\alpha \in \mathbb{R}$

Output: $p(\alpha)$

Algorithm #2: time $O(n)$

Initialize $A = [1, \quad \alpha, \quad \alpha^2, \quad \alpha^3, \quad \ldots, \quad \alpha^{n-1}]$ $\quad O(n)$ mults

Set $A[i] = \alpha \cdot A[i-1]$ for each $i = 1, 2, \ldots$ $\quad$ 1 mult each

$$
\begin{array}{r}
O(n) \\
\text{adds}
\end{array}
\left[
\begin{array}{lll}
 & p_0 \cdot A[0] & \text{1 mult} \\
+ & p_1 \cdot A[1] & \text{1 mult} \\
+ & p_2 \cdot A[2] & \text{1 mult} \\
+ & p_3 \cdot A[3] & \text{1 mult} \\
 & \quad \ldots & \\
+ & p_{n-1} \cdot A[n-1] & \text{1 mult}
\end{array}
\right.
$$

$$p(\alpha) \qquad\qquad O(n) \text{ mults}$$

Task 3: multiplying polynomials

Input: two polynomial $p(x)$ and $q(x)$

Output: $p(x) \cdot q(x)$

Example: $(7 + 5x) \cdot (1 + 3x + 2x^2)$

$$
\begin{array}{llll}
7 & + 21x & + 14x^2 & \\
 & 5x & + 15x^2 & + 10x^3 \\
\hline
7 & + 26x & + 29x^2 & + 10x^3
\end{array}
$$

Q: How fast can you do this?

Task 3: multiplying polynomials

Input: two polynomial $p(x)$ and $q(x)$

Output: $p(x) \cdot q(x)$ ⟵ possibly degree-($2n-2$)

Algorithm

$$(p_0 + p_1 x + \cdots + p_{n-1} x^{n-1}) \cdot (q_0 + q_1 x + \cdots + q_{n-1} x^{n-1})$$

$p_0 \cdot (q_0 + q_1 x + \cdots + q_{n-1} x^{n-1})$ ⟵ $O(n)$ time

$p_1 \cdot (q_0 x + q_1 x^2 + \cdots + q_{n-1} x^{n})$ ⟵ $O(n)$ time

$\dots$ $\dots$

$p_{n-1} \cdot (q_0 x^{n-1} + q_1 x^{n} + \cdots + q_{n-1} x^{2n-2})$

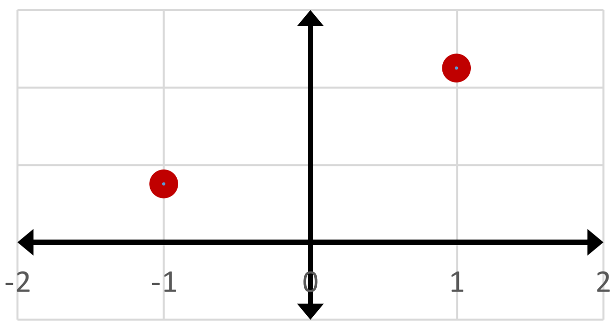
Total: $O(n^2)$ time

Goal of this lecture: improve this to $O(n\log(n))$ time

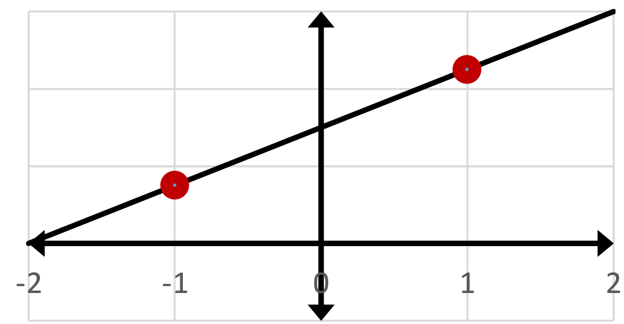
Fact: n points determine a degree-$(n-1)$ polynomial

Examples:

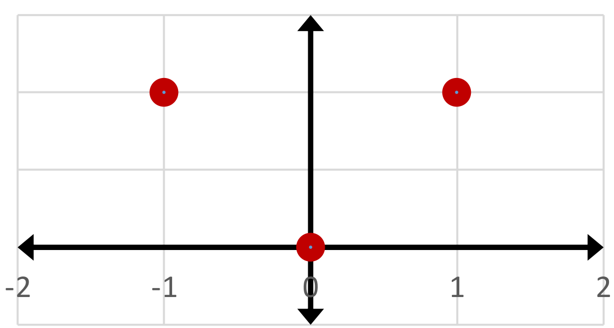
1. $p(x)$ is degree-**1** and we know **2** points



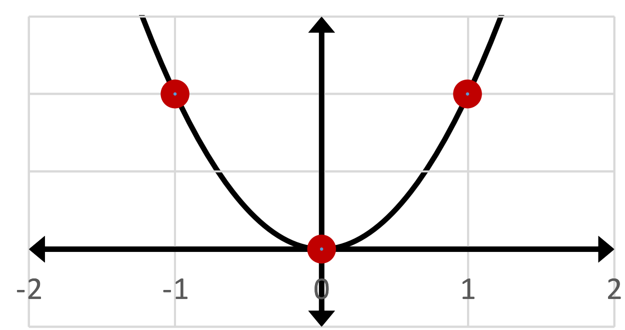
$\Rightarrow p(x) =$



2. $p(x)$ is degree-**2** and we know **3** points



$\Rightarrow p(x) =$



Fact: $\boldsymbol{n}$ points determine a degree-($\boldsymbol{n-1}$) polynomial

Fix points $\boldsymbol{\alpha_1, \alpha_2, \dots, \alpha_m} \in \mathbb{R}$

Let $p(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_{n-1} x^{n-1}$,

where $\boldsymbol{n \leq m}$

Its **value representation** is the array of real numbers

$$(p(\boldsymbol{\alpha_1}), p(\boldsymbol{\alpha_2}), \dots, p(\boldsymbol{\alpha_m}))$$

Note: a "typical" choice of m is $m = O(n)$

Adding and multiplying w/ value rep

Input:
1. $(p(\alpha_1), p(\alpha_2), \dots, p(\alpha_m))$
2. $(q(\alpha_1), q(\alpha_2), \dots, q(\alpha_m))$

degree $(n-1)$, so $m \geq n$

Addition: Output $(p(\alpha_1) + q(\alpha_1), \dots, p(\alpha_m) + q(\alpha_m))$

(value representation of $p + q$)

$O(m)$ time = $O(n)$ time for "typical" m

Multiplication: Output $(p(\alpha_1) \cdot q(\alpha_1), \dots, p(\alpha_m) \cdot q(\alpha_m))$

(value representation of $p \cdot q$)

$O(m)$ time = $O(n)$ time for "typical" m

Note: $p \cdot q$ is degree-$(2n-2)$, so need $m \geq 2n-1$

Multiplication much faster in value representation!

Fast polynomial multiplication algorithm

Coeffs of p and q $\xrightarrow[O(n \log(n))]{\text{Evaluation}}$ Values $p(\alpha_1), \dots, p(\alpha_m)$, $q(\alpha_1), \dots, q(\alpha_m)$

$\downarrow$ $O(m) = O(n)$

Coeffs of $p \cdot q$ $\xleftarrow[O(n \log(n))]{\text{Interpolation}}$ Values $p(\alpha_i) \cdot q(\alpha_i)$

Total: time $O(n \log(n))$?

Evaluation takes time $O(m \cdot n) = O(n^2)$

Hope: pick $\alpha_1, \dots, \alpha_m$ cleverly **= using complex numbers!!!**
so that evaluation takes $O(n \log(n))$ time

(same for interpolation)

Outline

1. Complex numbers
2. Polynomial multiplication I: fast evaluation
3. Polynomial multiplication II: fast interpolation
4. The matrix viewpoint
5. Applications

Complex numbers

3-minute break

and close the doors

Complex numbers

$$a + b \cdot i, \qquad i = \sqrt{-1}$$

real imaginary

$$\begin{aligned}(1 + 2 \cdot i) + (3 + 4 \cdot i) &= (1 + 3) + (2 + 4) \cdot i \\ &= 4 + 6 \cdot i\end{aligned}$$

$$\begin{aligned}(1 + 2 \cdot i) \cdot (3 + 4 \cdot i) &= 1 \cdot 3 + 1 \cdot 4i + 2i \cdot 3 + 2i \cdot 4i \\ &= 3 + 4i + 6i + 8i^2 \\ &= 3 + 10i - 8 \\ &= -5 + 10i\end{aligned}$$

Complex plane

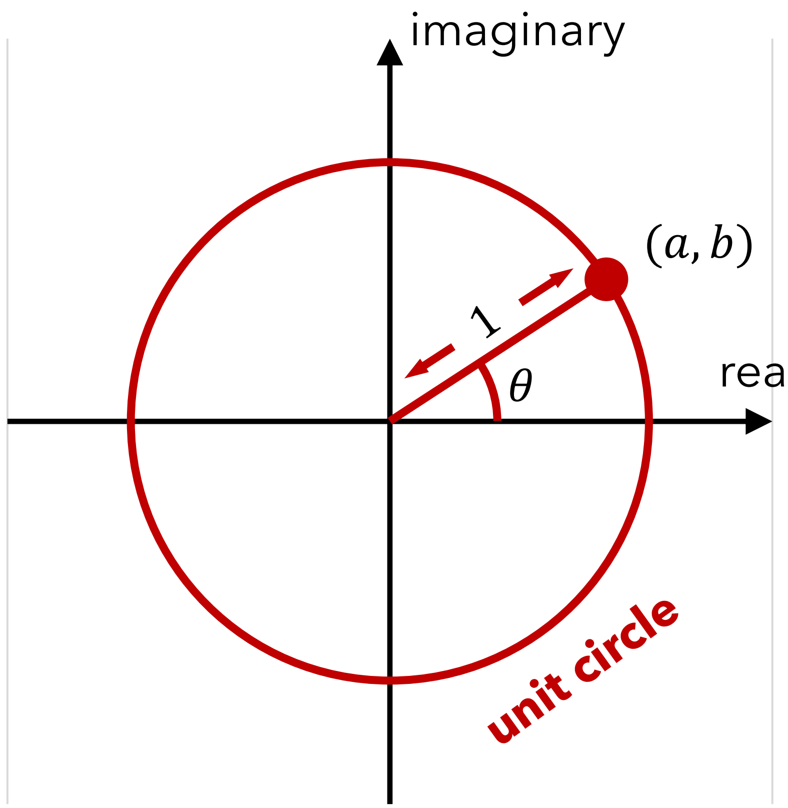
The number $a + b \cdot i$
is (a, b) in the **complex plane**

Polar coordinates

Radius r and angle θ such that

- $a = r \cdot \cos(\theta) = \cos(\theta)$
- $b = r \cdot \sin(\theta) = \sin(\theta)$

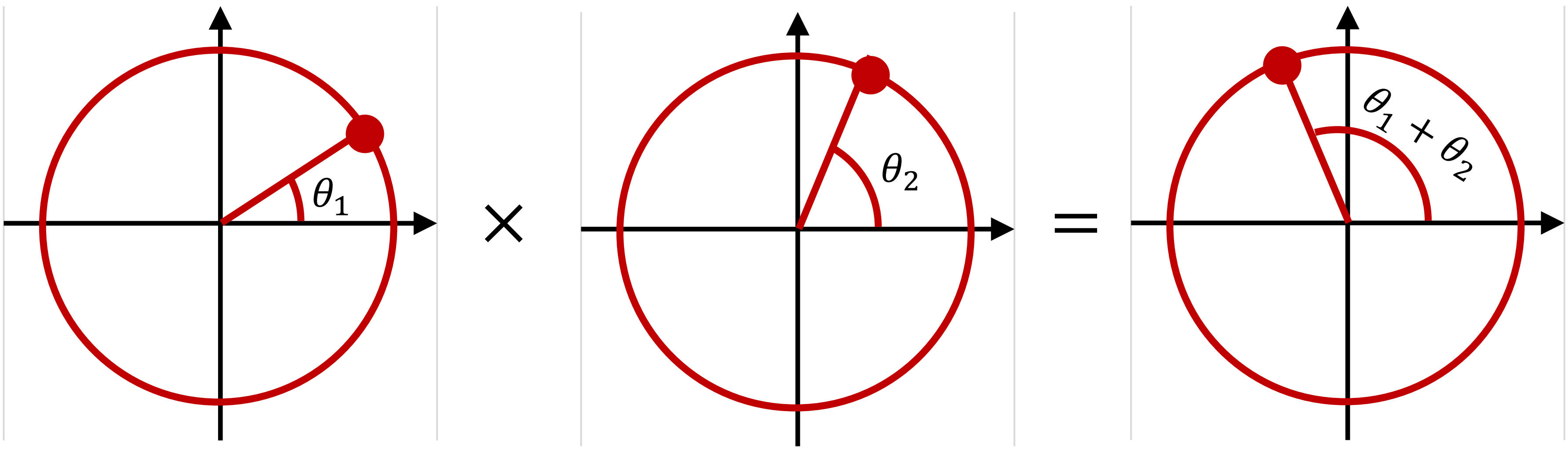
We will only consider
points with $\boldsymbol{r = 1}$ today.



Complex plane

Multiplication

Consider two complex numbers on the unit circle.



To multiply, just add the angles.

Roots of unity

Def:

- **unity** = fancy word for the number **1**
- $\boldsymbol{n^{th}}$ roots of unity (**1**) = {solutions to $\boldsymbol{x^n = 1}$}

Examples

$\boldsymbol{2^{nd}}$ roots of unity $= \sqrt{1} = \{\pm 1\}$

$\boldsymbol{4^{th}}$ roots of unity $= \sqrt[4]{1} = \{+1, -1, +i, -i\}$

$$\boldsymbol{i^4 = \left(i^2\right)^2 = (-1)^2 = 1}$$

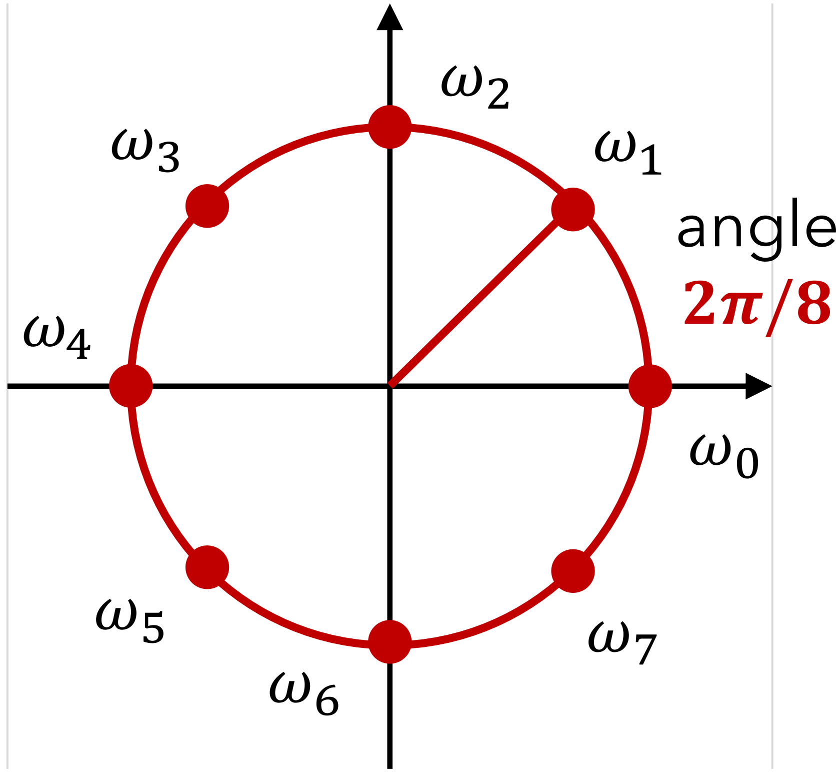
$$\boldsymbol{(-i)^4 = i^4 = 1}$$

$\boldsymbol{8^{th}}$ roots of unity $= \sqrt[8]{1} = \left\{\pm 1, \pm i, \pm\left(\frac{1+i}{\sqrt{2}}\right), \pm\left(\frac{1-i}{\sqrt{2}}\right)\right\}$

Note: Today, will care about $n = 2, 4, 8, 16, \ldots$ roots of unity

Roots of unity

n^{th} roots of unity
$= n$ equally spaced points on unit circle



8^{th} roots of unity

Generator fact:
For all $0 \leq i \leq m-1$,
$$\omega_i = \omega_1^i.$$
In addition, $\omega_1^0 = 1 = \omega_0 = \omega_1^m$.

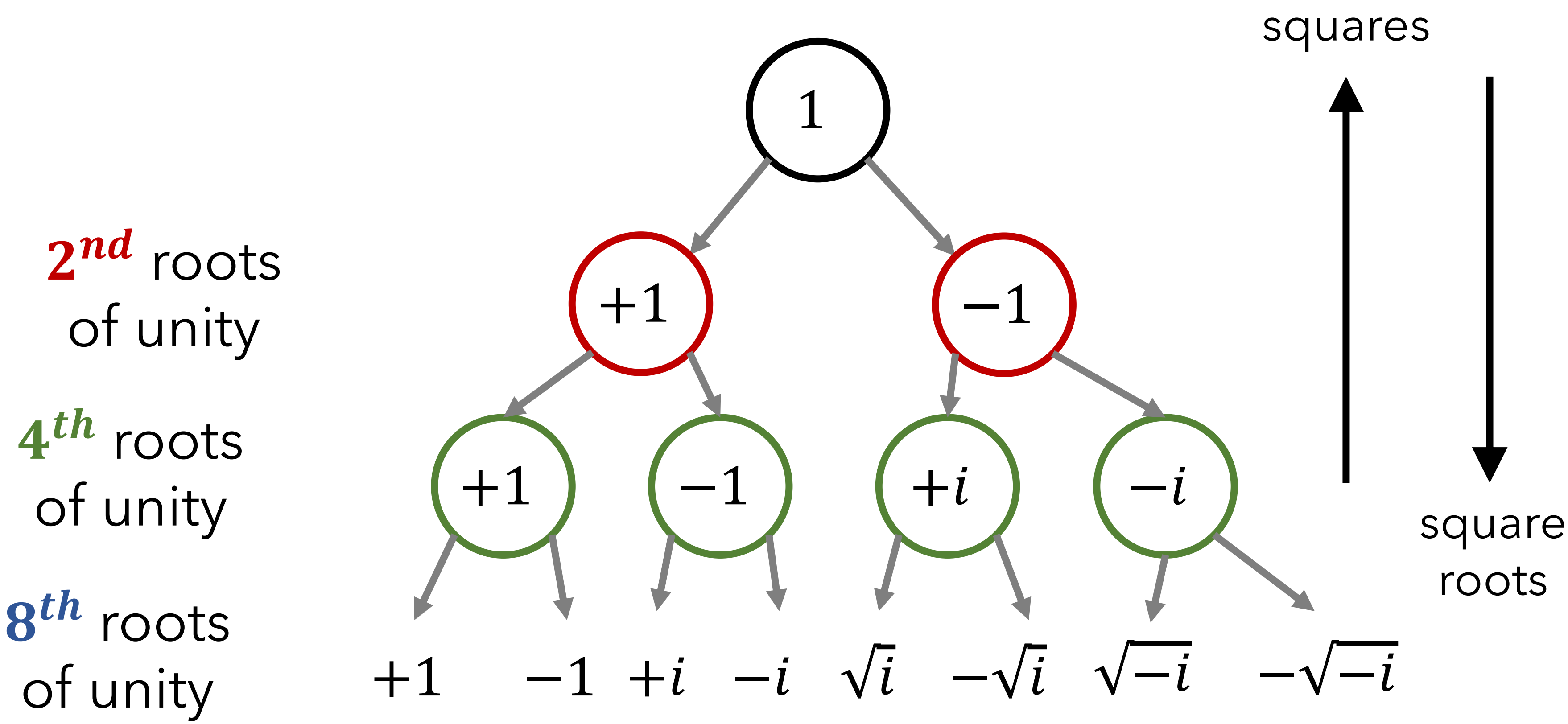
Formula: n^{th} roots of unity

$= \text{angles } 0 \cdot \theta, 1 \cdot \theta, 2 \cdot \theta, 3 \cdot \theta, \ldots,$ where $\theta = (2\pi/n)$,

$= \{\cos(\theta \cdot \ell) + \sin(\theta \cdot \ell) \cdot i \mid \ell = 0, 1, \ldots, n-1\}$

Square roots

Square roots always come in pairs $\pm\sqrt{a}$



Magical Fact: Squares of n^{th} roots = $(n/2)^{th}$ roots

(So squaring **halves** the number of roots)

Complex number takeaways

Generator fact: For all $0 \leq i \leq m-1, \quad \omega_i = \omega_1^i.$

In addition, $\omega_1^0 = 1 = \omega_0 = \omega_1^m.$

Magical Fact: Squares of $\boldsymbol{n^{th}}$ roots = $\boldsymbol{(n/2)^{th}}$ roots

(So squaring **halves** the number of roots)

Not true of most sets of numbers!

Example: the set of numbers $\{1,3,5,7\}$

squaring them gives $\{1^2, 3^2, 5^2, 7^2\} = \{1,9,25,49\}$

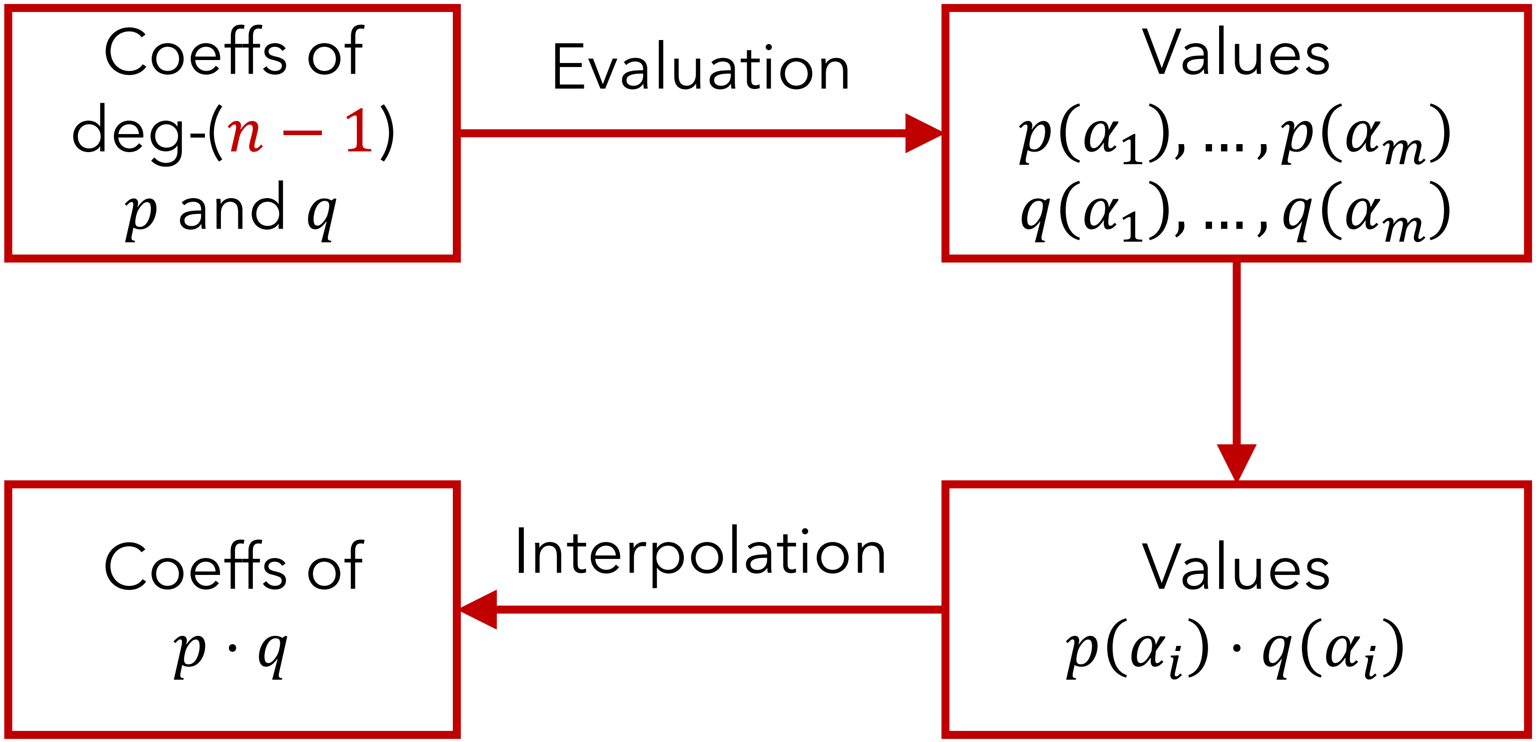
both sets have 4 elements!

Outline

1. Complex numbers
2. Polynomial multiplication I: fast evaluation
3. Polynomial multiplication II: fast interpolation
4. The matrix viewpoint
5. Applications

Returning to polynomial multiplication: Fast evaluation

Fast polynomial multiplication algorithm



Fast polynomial multiplication algorithm

Coeffs of deg-$(n-1)$ p and q	$\xrightarrow{\text{Evaluation}}$	Values $p(\alpha_1), \dots, p(\alpha_m)$ $q(\alpha_1), \dots, q(\alpha_m)$

Recall: $p \cdot\ q$ is degree-$(2n-2)$, so need $m \geq 2n-1$

Let $\boldsymbol{m}$ be first power of $\mathbf{2}$ such that $m \geq 2n-1$

Will evaluate p and q on $\boldsymbol{m^{th}}$ roots of unity

$$\{\omega_0, \omega_1, \dots, \omega_{m-1}\}$$

in time $O(\boldsymbol{m \log(m)}) = O(\boldsymbol{n \log(n)})$.

This is the **Fast Fourier transform**.