Complexity & Algorithms, Fall 2026

End of lower bounds

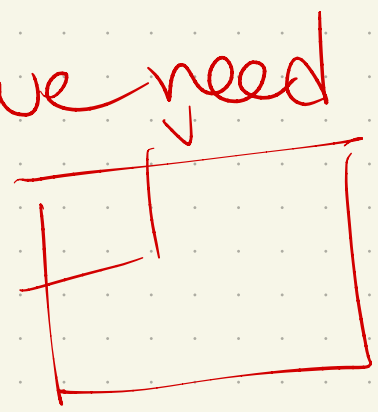
FFTs

Lower bounds

Let's consider another problem:
Given the adjacency matrix for a graph, how hard is connectivity?

Previously: we saw algorithm.
DFS or BFS: $O(V+E)$ → $E \leq \binom{V}{2}$
↳ checks all edges

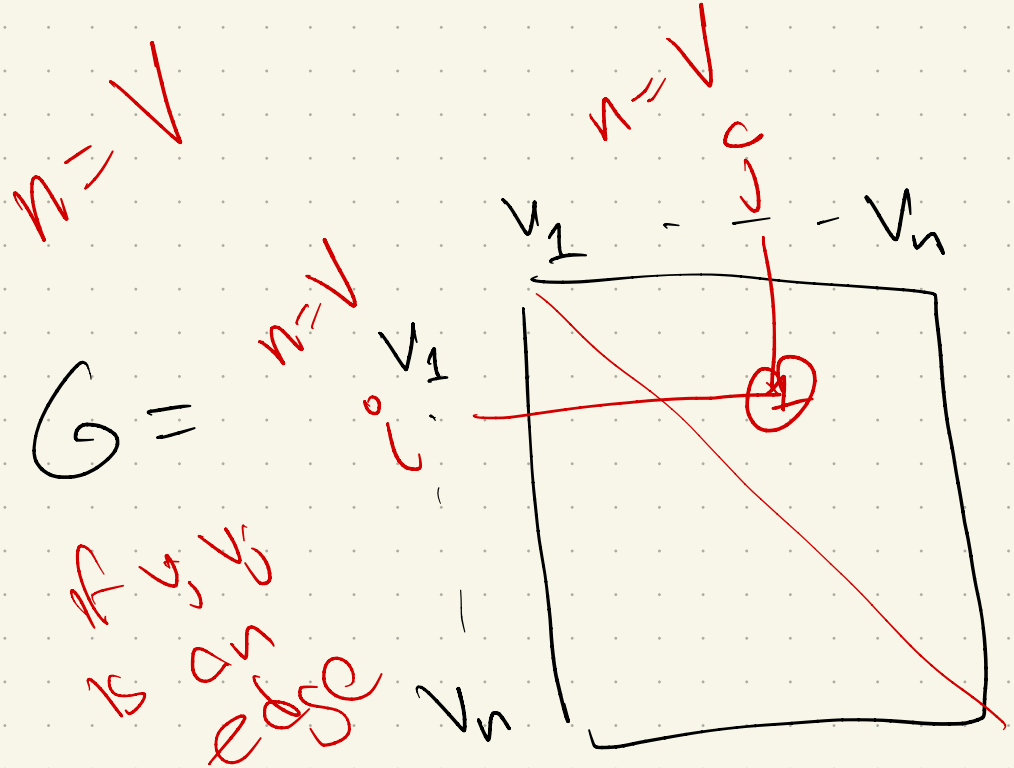
Here, a different notion:
How many edges do we need to check?



Want worst runtime of the best algorithm; which I'll call $D(n)$:

$$D(n) = \min_{A} \max_{G} T(A, G)$$

where $T(A, G)$ = # of bits of G's adj. matrix that A uses



Note:

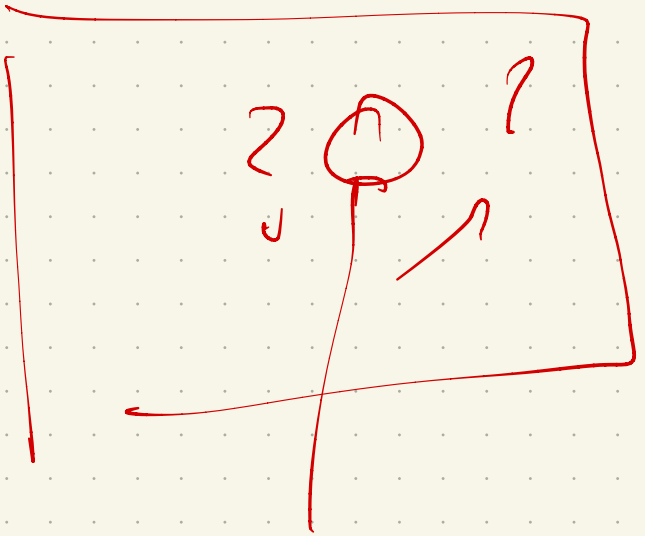
DFS / BFS check all $\binom{n}{2}$ entries.

A method for lower bounds:

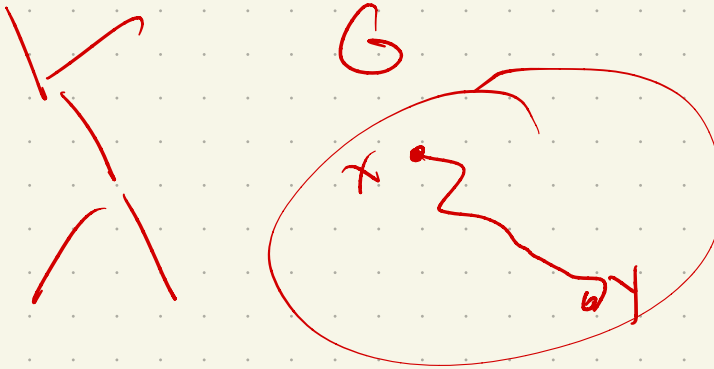
Consider a small "proof" or "certificate."

Here, what could prove G is connected?

a tree that touches every vertex:

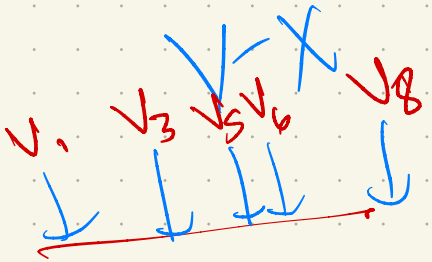


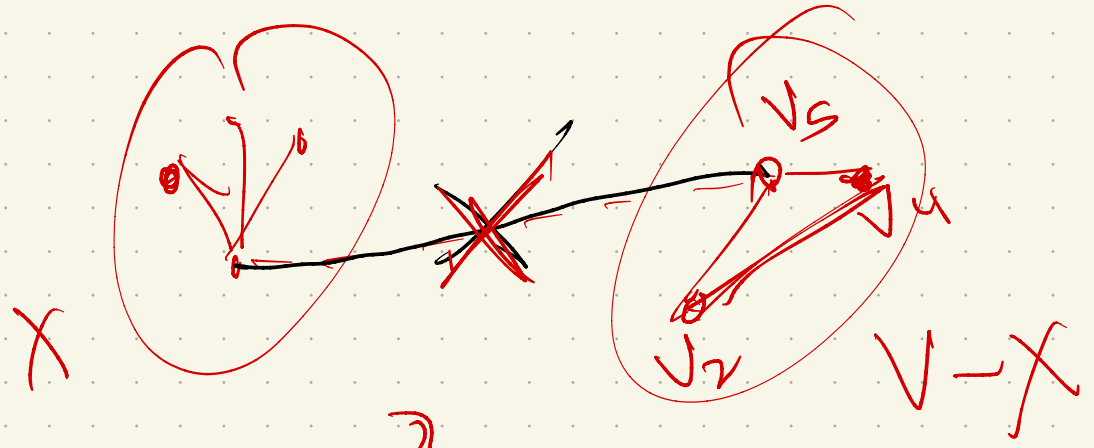
With n-1 queries I could prove connected



What could prove G is not Connected?

A cut:





v_2 →	0	0	1	0	0	0	0
X v_4 →	0	0	0	0	0	0	0
v_7 —	0	0	0	0	0	0	0

no edges from X to V - X

How many edges to prove?

$$|X| \cdot |V - X| < \binom{n}{2}$$

Think of this as a function:

$$f : \{0,1\}^{\binom{n}{2}} \longrightarrow \{0,1\}$$

space of 0/1 adj matrices on vertex set of size n

yes/no

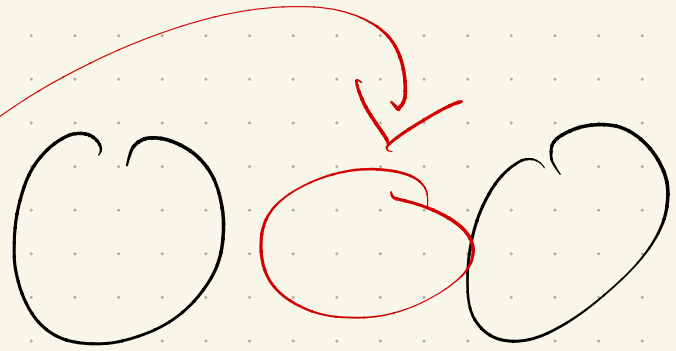
boolean string functions

1-certificate $C_1(n)$:
a subset of bits that forces
$f(x)=1$ "yes" (spanning tree)

Here: $f(n) \geq n-1$

0-certificate $C_0(n)$:
a subset of bits that forces
$f(x)=0$ "no"

Here: cut



x vertices n-x vertices

so $C_0(n) \leq \max\{n(n-x)\}$ maximized when $x=\frac{n}{2}$

$C_0(n) \leq \frac{n}{2} \cdot \frac{n}{2} = \lceil \frac{n^2}{4} \rceil$ x

Let $C(n) = \max\{C_1(n), C_0(n)\}$.

Theorem: $D(n) \geq C(n)$

Why?

If not, I can't tell if answer is yes or no!

Here $D(n) \geq \max\left\{n-1, \frac{n^2}{4}\right\}$

$$\geq \frac{n^2}{4}$$

$$D(n) \leq \binom{n}{2}$$

Theorem: $D(n) = \binom{n}{2}$

Proof: upper bound is easy!

$D(n) \leq \binom{n}{2}$

↳ use DFS or BFS if have all edges

lower bound: "aversary" maintains 2 graphs $Y \subseteq M$

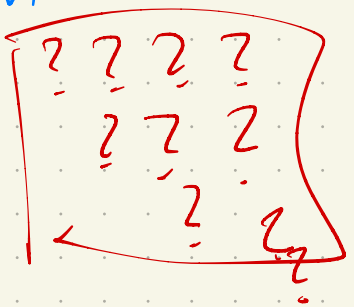
"yes" edges: ones known to exist

"maybe" set: ones might be there

Initially: $Y = \emptyset$

M = all edges $= K_n$

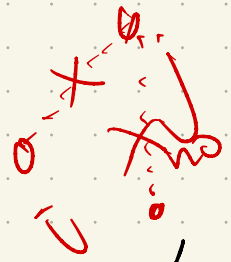
Note: $Y \subseteq M$



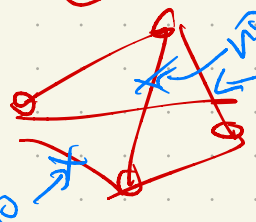
Adversary strategy:

Return FALSE when algorithm checks any edge

↳ unless FALSE would force G to be disconnected



$M = G$

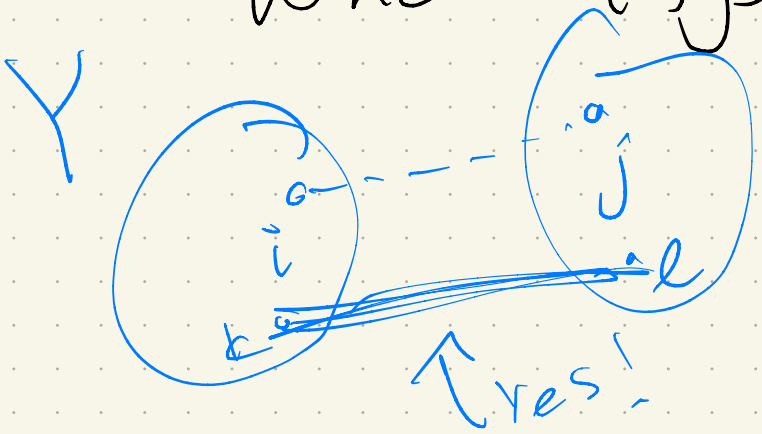


add to Y
say yes

EXAMINE(i, j):
- if $(i, j) \in Y$
  - derisively return TRUE
- else if $(i, j) \notin M$
  - mockingly return FALSE
- else if $M \setminus (i, j)$ is disconnected
  - add (i, j) to Y
  - grudgingly return TRUE
- else
  - remove (i, j) from M
  - sigh and return FALSE

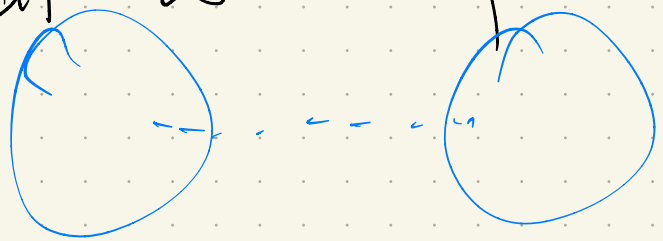
Observation:

- M is always connected. b/c I check
- Y is always acyclic. If adding (i,j) makes cycle, don't add
- When (i,j) is added to Y: $Y \subseteq M$



all edges b/t i's comp & j's comp have been checked

So: Y only connects after $\binom{n}{2}$ queries, at which point $M = Y$.



Before $\left(\frac{n}{2}\right)^{th}$ query:

- Y is disconnected "yes"
- M is connected "maybe"

Both Y & M are consistent with answers so far

$\Rightarrow$ Algorithm can't decide earlier.

□

Must read whole adj. matrix

A string property is a function

$F: \{0,1\}^{n} \longrightarrow \{0,1\}$ yes/no

$\{0,1\}^{\binom{n}{2}}$ ← (n graphs)

A string property is evasive if its deterministic decision tree complexity $D(n)$ is $= n$.

When are things evasive?

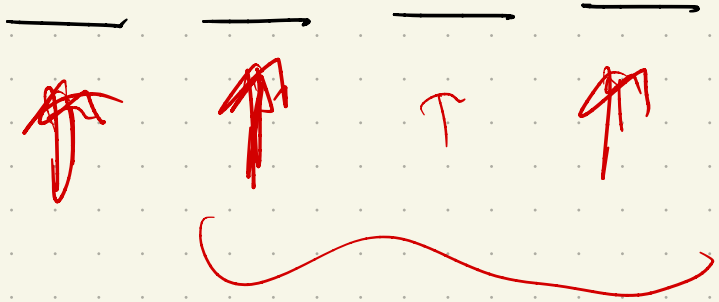
Graph connectivity!

Simple game:

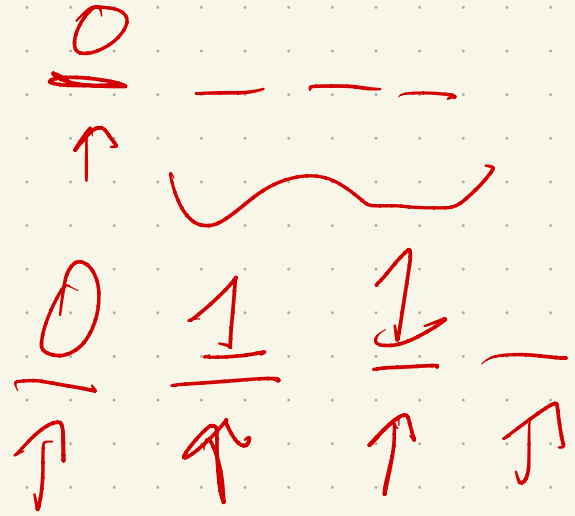
Let $f(s) = 1$ if a boolean string $s \in \{0,1\}^n$ has 3 1's in a row.

$n = 3$: 1 1 1 evasive!

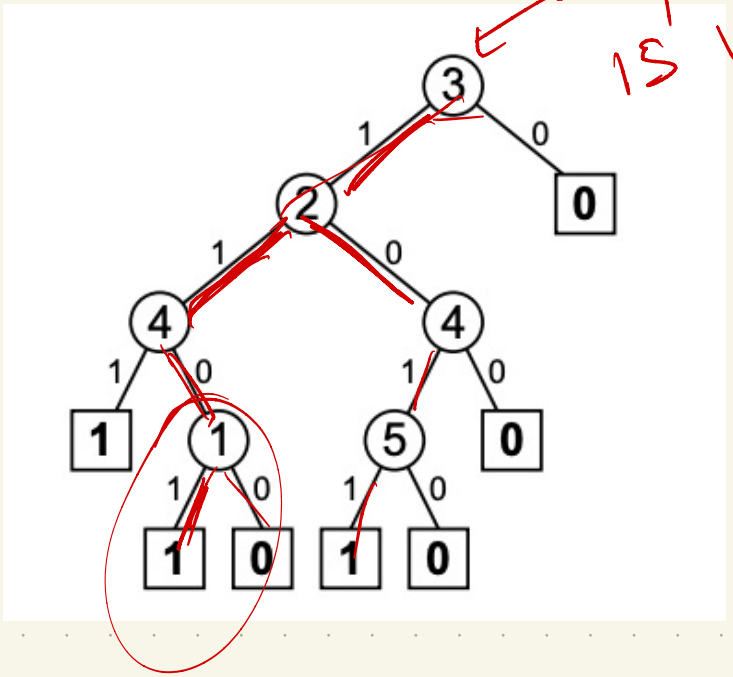
$n = 4$:



4 checks → evasive



When $n=5$, not evasive:



at pos 5, is it 0, or 1?

_ 1 1 0 _

4 checks suffices

Aanderaa-Karp-Rosenberg Conjecture: (1970s)

Every nontrivial monotone graph property is evasive.

$= \binom{n}{2}$

Known: connectivity

also: acyclic, planarity, bipartiteness, k-colorability, & many more.

[Rivest-Vuillemin]: $D(n) = \Omega(n^2)$

$\geq \frac{n^2}{4}$

$\geq c \cdot n^2$

Blum's Theorem:

For a string function $F: \{0,1\}^n \to \{0,1\}$

let $T(A,s)$ = # a bits in an n-bit input string s that algorithm A must check to compute $F(s)$.

$$D(n) = \min_A \max_{\substack{s \\ \text{length } n}} T(A,s)$$

$$C(n) = \max_s \min_A T(A,s)$$

"certificate"

The theorem:

$\max\{C_1, C_0\} \leq C(n) \leq D(n) \leq C(n)^2$

we worked this already

not so much useful for graphs!

$\rightarrow \left(\frac{n^2}{4}\right)^2$

OK, back to Algorithms!

FFTs: why?

Polynomials abound on computers!

Adding: $A(x) + B(x)$

$=$

$a_{n-1}x^{n-1} + \cdots + a_0$ → $b_{m-1}x^{m-1} + \cdots b_0$

$a_i + b_j$ for each i

$O(n)$ time

Applications: Signal processing!

Scientific applications

Consider $p(x) = \sum_{j=0}^{n-1} a_j \cdot x^j$

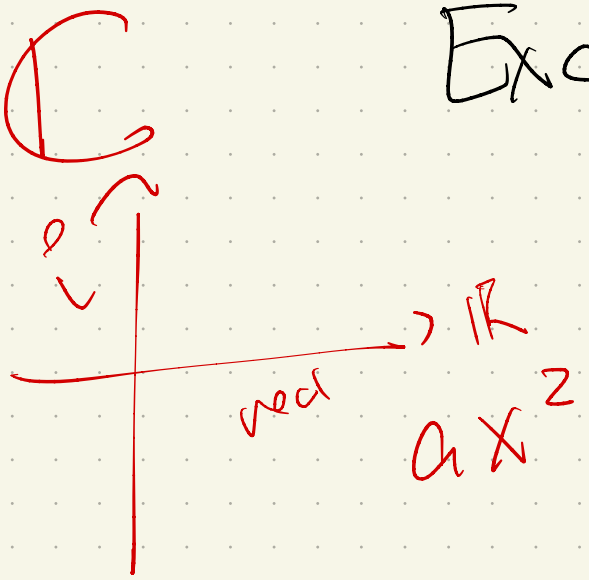
a_j : coefficients in a field

Not always $\mathbb{R}$!

Example: Complex #s $\mathbb{C}$

even if in $\mathbb{R}$, roots can be in $\mathbb{C}$

$a_j = (\underline{a}i + \underline{b})$



$ax^2 + bx + c$

$\hookrightarrow x = \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

Multiplying $C(x) = A(x) \cdot B(x)$

Example: $A(x) = 6x^3 + 7x^2 - 10x + 9$

$B(x) = -2x^3 + 4x - 5$

Setup:

$$\begin{array}{r} 6x^3 + 7x^2 - 10x + 9 \\ \times \quad -2x^3 + 4x - 5 \\ \hline -30x^3 - 35x^2 + 50x - 45 \\ 24x^4 + 28x^3 - 40x^2 + 36x \\ -12x^6 - 14x^5 + 20x^4 - 18x^3 \\ \hline \end{array}$$

In other words:

$$A(x) \cdot B(x) = C(x) = \sum_{j=0}^{2n-2} c_j x^j$$

where $$c_j = \sum_{k=0}^{j} a_k b_{j-k}$$

Question: How to speed up practically?

Answer: Recursion!

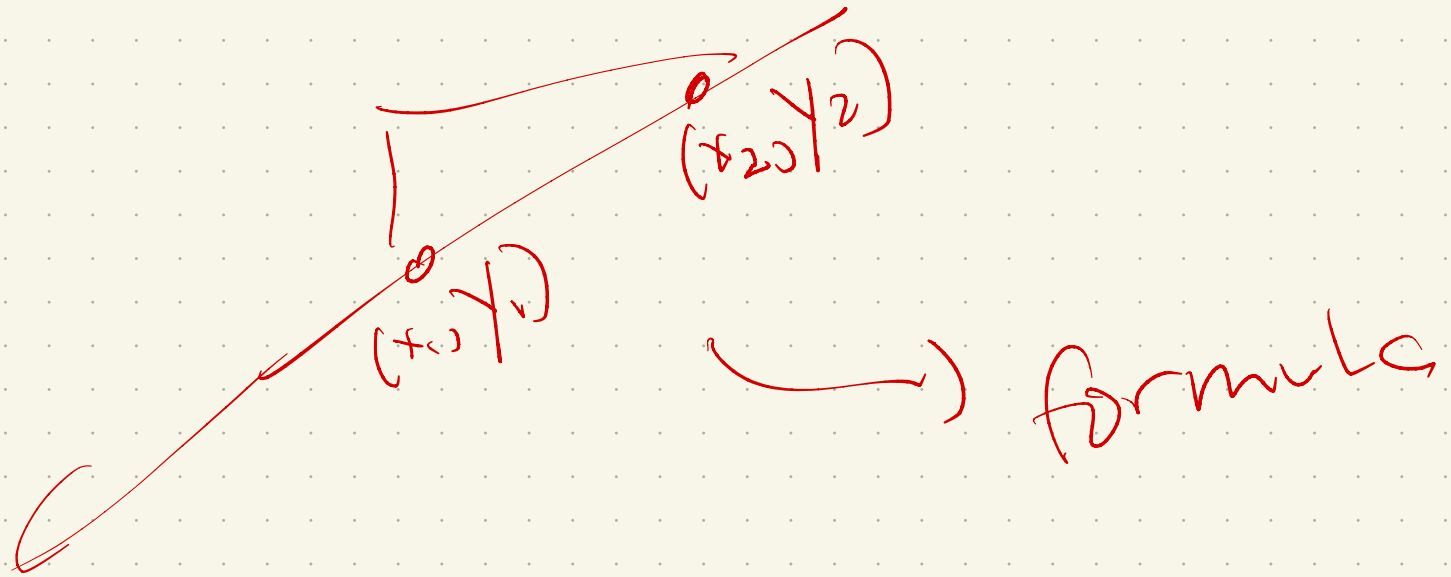
Saw faster mult.

$\hookrightarrow$ ch 1 $\nearrow n^{\log_2 3}$?

Fact: any polynomial of degree n is determined by $n+1$ values, as long as they are distinct

$(x, f(x))$

Ex: degree 1: $f(x) = ax + b$



Degree 2: Fix $(0,1), (1,2), (2,5)$ $\longleftrightarrow$ $x, p(x)$

& find $p(x) = ax^2 + bx + c$

Plug in:

$$p(0) = 1$$

$$\Rightarrow c = 1$$

$$p(1) = 2 = a + b + c$$

$$\Rightarrow a + b = 1$$

So: factoring a polynomial. O's!

Write $p(x) = \prod_{j=1}^{n} (x - r_j)$

Note: • r_j is a root of the polynomial.

• motivates $\mathbb{C}$: r_j won't be in $\mathbb{R}$, even if all coeffs are in $\mathbb{R}$

$P(x) \cdot Q(x) =$ . . .

Unfortunately: two polynomials have different roots!

Vandermonde matrix:

Given n pairs for polynomial: $\{(x_0, y_0), \dots, (x_{n-1}, y_{n-1})\}$

$V(x_0, \dots, x_{n-1})$

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{n-1} \end{pmatrix}$$

x_0 row

~~x_1~~ row

x_{n-1} row

Why?

- Has determinant $\prod_{0 \le j < k \le n-1} (x_k - x_j)$

- It is invertible: V^{-1}

  Goal: get a_i's

- And $V\vec{a} = \vec{y}$

$$\Rightarrow \vec{a} = V^{-1}\vec{y}$$

Next time: Why we care... mult!