15451 Fall 23

The Algorithmic Magic of Polynomials

Danny Sleator

Polynomials

- Polynomial: $p(x) = c_d x^d + c_{d-1} x^{d-1} + \cdots + c_1 x + c_0$
- $(c_d, c_{d-1}, \ldots, c_0)$ completely describes p

Polynomials

- Polynomial: $p(x) = c_d x^d + c_{d-1} x^{d-1} + \cdots + c_1 x + c_0$

- $(c_d, c_{d-1}, \ldots, c_0)$ completely describes p

Assume: adding and multiplying two values in O(1) time

Polynomials

- Polynomial: $p(x) = \ c_d x^d + c_{d-1} x^{d-1} + \cdots + c_1 x + c_0$

- $(c_d, c_{d-1}, \ldots, c_0)$ completely describes p

  - Addition: O(d)
  - Multiplication: O(d log d) using FFT

Polynomials

- Polynomial: $p(x) = c_d x^d + c_{d-1} x^{d-1} + \cdots + c_1 x + c_0$

- $(c_d, c_{d-1}, \ldots, c_0)$ completely describes p

- Addition: O(d)
- Multiplication: O(d log d) using FFT

- Evaluation: ?

Evaluating a Polynomial Quickly

- Polynomial: $p(x) = c_d x^d + c_{d-1} x^{d-1} + \cdots + c_1 x + c_0$

- Evaluate at a point b in time O(d) using Horner's Rule:

- Compute: c_d

  $c_{d-1} + c_d \cdot b$

  $c_{d-2} + c_{d-1} \cdot b + c_d \cdot b^2$

  …

- Each step has O(1) operations – multiply by and add coefficient

Polynomial Degree

- Polynomial: $p(x) = c_d x^d + c_{d-1} x^{d-1} + \cdots + c_1 x + c_0$

- If $c_d \neq 0$, the degree is d

- If A(x) has degree d and B(x) has degree d, then A(x) + B(x) has degree at most d

Why is the degree at most d?

Roots of Polynomials

- A root of a polynomial is a number r for which A(r) = 0
- Fundamental theorem of algebra: a non-zero degree-d polynomial has at most d roots

(Holds for any field)

x^2+1

x^2

x^2-1

Roots of Polynomials

- A root of a polynomial is a number r for which $A(r) = 0$

- Fundamental theorem of algebra: a non-zero degree-d polynomial has at most d roots

  - Implies any distinct degree d polynomials $A(x)$ and $B(x)$ can evaluate to the same value on at most d different values x. Why?

$$A(x) - B(x) = 0$$

Roots of Polynomials

- A root of a polynomial is a number r for which A(r) = 0
- Fundamental theorem of algebra: a non-zero degree-d polynomial has at most d roots

Roots of Polynomials

- A root of a polynomial is a number r for which A(r) = 0

- Fundamental theorem of algebra: a non-zero degree-d polynomial has at most d roots

  - Implies any distinct degree d polynomials A(x) and B(x) can evaluate to the same value on at most d different values x. Why?

  - A(x) – B(x) has degree at most d, so can have at most d roots

Unique Reconstruction Theorem

- Given $(x_0, y_0), \ldots, (x_d, y_d)$ for distinct $x_0, \ldots, x_d$, there exists a polynomial of degree at most d for which $p(x_i) = y_i$ for each i

Unique Reconstruction Theorem

- Given $(x_0, y_0), \ldots, (x_d, y_d)$ for distinct $x_0, \ldots, x_d$, there exists a polynomial of degree at most d for which $p(x_i) = y_i$ for each i

- Define $R_i(x) = \prod_{j \neq i}(x - x_j) \,/\, \prod_{j \neq i}(x_i - x_j)$, which has degree d

Unique Reconstruction Theorem

- Given $(x_0, y_0), \ldots, (x_d, y_d)$ for distinct $x_0, \ldots, x_d$, there exists a polynomial of degree at most d for which $p(x_i) = y_i$ for each i

- Define $R_i(x) = \prod_{j \neq i}(x - x_j) / \prod_{j \neq i}(x_i - x_j)$, which has degree d

- $R_i(x_j) = 0$ for $j \neq i$

- $R_i(x_i) = 1$

Unique Reconstruction Theorem

- Given $(x_0, y_0), \ldots, (x_d, y_d)$ for distinct $x_0, \ldots, x_d$, there exists a polynomial of degree at most d for which $p(x_i) = y_i$ for each i

- Define $R_i(x) = \prod_{j \neq i}(x - x_j) / \prod_{j \neq i}(x_i - x_j)$, which has degree d

- $R_i(x_j) = 0$ for $j \neq i$

- $R_i(x_i) = 1$

- $p(x) = \sum_{i=0,\ldots,d} y_i \cdot R_i(x)$ $\quad p(x_i) = y_i \quad \forall i$

Example of Polynomial Reconstruction

- Given pairs (5,1), (6,2), and (7,9), we would like to find a degree-2 polynomial that passes through these points

- $R_0(x) = \frac{(x-6)(x-7)}{(5-6)(5-7)} = \frac{1}{2}(x-6)(x-7)$

- $R_1(x) = \frac{(x-5)(x-7)}{(6-5)(6-7)} = -(x-5)(x-7)$

- $R_2(x) = \frac{(x-5)(x-6)}{(7-5)(7-6)} = \frac{1}{2}(x-5)(x-6)$

- $p(x) = 1 \cdot R_0(x) + \underline{2 \cdot R_1}(x) + \underline{9 \cdot R_2(x)} = 3x^2 - 32x + 86$

Polynomial Reconstruction can be achieved

- in O(d log d) time if roots of unity
- in O(d poly log d) time (for the general case)

see https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.368.9192&rep=rep1&type=pdf

Lecture notes: $O(d^2)$ time

Polynomials For Error Correcting Codes

Error Correcting Codes

- Communication channel may be lossy or noisy
- How can we have reliable communication?

Applications of Error Correcting Code

- Communication, e.g., satellite, wifi
- Storage systems
- QR code
- Lots of applications in cryptography
  - Proof of retrievability
  - Zero-knowledge proofs

A Deletion Channel

(a,b)

d+1 = 2

5, 19, 2, 3, 2

*, 19, *, *, 2

- Alice has d+1 numbers and wants to send them to Bob
- Up to k of the numbers might be replaced with a *

k = 3

- *How can Bob learn Alice's numbers?*

Deletion Channel and Erasure Code

- Alice could repeat each number k+1 times

a, b   aaaa bbbb

Deletion Channel and Erasure Code

- Alice could repeat each number k+1 times
- If k = 3, she sends:

5, 5, 5, 5, 19, 19, 19, 19, 2, 2, 2, 2, 3, 3, 3, 3, 2, 2, 2, 2

- This is (d+1)(k+1) words of communication

Deletion Channel and Erasure Code

- Alice could repeat each number k+1 times

- If k = 3, she sends:

  5, 5, 5, 5, 19, 19, 19, 19, 2, 2, 2, 2, 3, 3, 3, 3, 2, 2, 2, 2

- This is (d+1)(k+1) words of communication

- *Can we get d+k+1 communication?*

Deletion Channel and Erasure Code

- Suppose Alice has $c_d, c_{d-1}, c_{d-2,\ldots,}\ c_0$
- She interprets these as the coefficients of a polynomial P(x):

$$P(x) = \sum_{i=0,\ldots,d} c_i x^i$$

- Alice sends P(0), P(1), P(2), …, P(d+k)

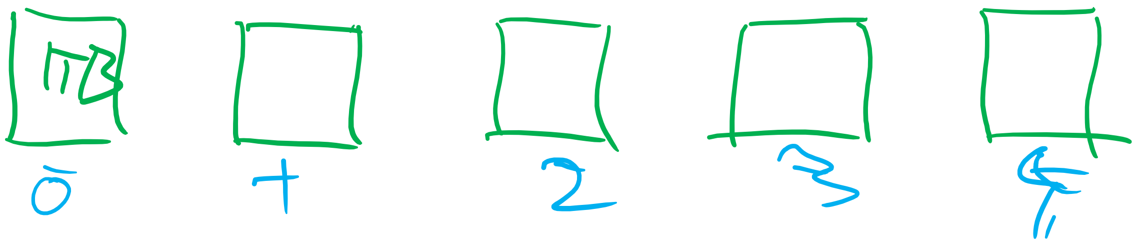
Deletion Channel and Erasure Code

- Suppose Alice has $c_d, c_{d-1}, c_{d-2,\ldots,}\ c_0$
- She interprets these as the coefficients of a polynomial P(x):

$$P(x) = \sum_{i=0,\ldots,d} c_i x^i$$

- Alice sends P(0), P(1), P(2), …, P(d+k)
- Bob gets at least d+1 of these numbers. By the unique reconstruction theorem, he recovers P(x), and hence $c_d, c_{d-1}, c_{d-2,\ldots,}\ c_0$

Application of Erasure Code: Redundant Arrays of Independent Disks – RAID



store 3TB

if 3 are good, can recover all.

10^{12} bytes

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \text{ first byte } \Longrightarrow \underline{a_0 \cdots a_4}$$

General Error Correction

- Now the adversary can replace up to k numbers with other numbers

General Error Correction

- Now the adversary can replace up to k numbers with other numbers

- If Alice wants to send Bob a single number x, how many times does she need to copy it?
  - 2k+1, to ensure the majority symbol is correct

General Error Correction

- Now the adversary can replace up to k numbers with other numbers

- If Alice wants to send Bob a single number x, how many times does she need to copy it?
  - 2k+1, to ensure the majority symbol is correct

Repetition code: (d+1) (2k+1)

General Error Correction

- Now the adversary can replace up to k numbers with other numbers

- If Alice wants to send Bob a single number x, how many times does she need to copy it?
  - 2k+1, to ensure the majority symbol is correct

Repetition code: (d+1) (2k+1)

Can we achieve d + 2k + 1?

General Error Correction

- Now the adversary can replace up to k numbers with other numbers

- Now Alice has $c_d, c_{d-1}, c_{d-2,\ldots,}\ c_0$, which she writes as a polynomial $P(x) = \sum_{i=0,\ldots,d} c_i x^i$

- Suppose Alice sends P(0), P(1), …, P(r). How large does r need to be?

General Error Correction

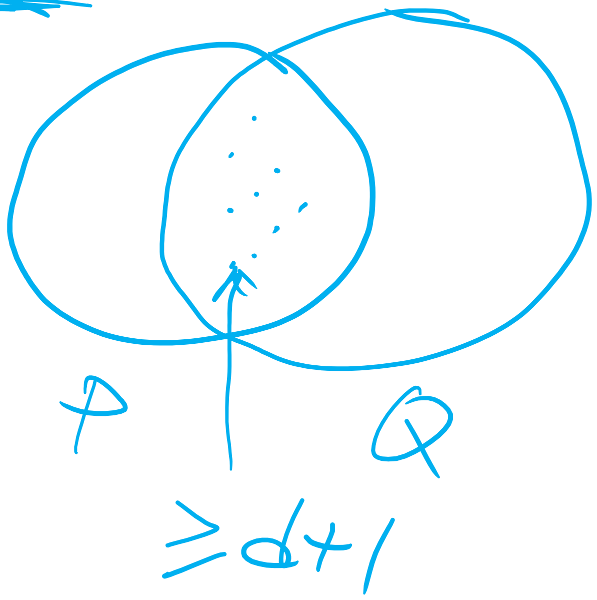
- Now the adversary can replace up to k numbers with other numbers

- Now Alice has $c_d, c_{d-1}, c_{d-2,\ldots,} c_0$, which she writes as a polynomial $P(x) = \sum_{i=0,\ldots,d} c_i x^i$

- Suppose Alice sends P(0), P(1), …, P(r). How large does r need to be?
  - d+2k+1 points is enough, so r = d+2k

Claim: suppose P and Q are consistent with all but k points, then P = Q

d+1+2k points



$d+1+2k$

Try removing all possible k subsets

Naive algorithm for reconstruction: brute force search for a set of d + k + 1 points that are "internally consistent"

Efficient Algorithm for General Error Correction

- But how to find $P(x)$ given k corruptions to $P(0)$, $P(1)$, …, $P(d+2k)$?

Efficient Algorithm for General Error Correction

- But how to find P(x) given k corruptions to P(0), P(1), ..., P(d+2k)?
- Suppose Bob receives $r_0, r_1, \ldots, r_{d+2k}$
- Z = {i such that $r_i \neq P(i)$}, and so $|Z| \leq k$
- $E(x) = \prod_{i \in Z}(x - i)$
- $P(x) \cdot E(x) = r_x \cdot E(x)$ for all x = 0, 1, 2, ..., d+2k

Polynomials for Finding Maximum Matchings

Multivariate Polynomials

- $p(x_1, x_2, x_3, x_4) = x_1 x_2^2 x_4 + x_3 x_4^2 + x_1 x_2^2 x_3^2 x_4$
- Degree of monomial $x_1^{i_1} x_2^{i_2} x_3^{i_3} x_4^{i_4}$ is $i_1 + i_2 + i_3 + i_4$
- Degree of p is the maximum degree of any of its monomials

Schwartz-Zippel Lemma for Multivariate Polynomials

- [Schwartz-Zippel] Let $P(X_1, \ldots, X_m)$ be a non-zero, m-variable, degree at most d polynomial, and let S be a subset from the field F. If each X_i is chosen independently in S

$$\Pr[P(X_1, \ldots, X_m) = 0] \leq \frac{d}{|S|}$$

- Sanity check: if m = 1, a non-zero degree-d polynomial has at most d roots

- If |F| > 3d, how can we tell if P is the all zeros polynomial w.pr. 2/3?

Schwartz-Zippel Lemma for Multivariate Polynomials

- [Schwartz-Zippel] Let $P(X_1, \ldots, X_m)$ be a non-zero, m-variable, degree at most d polynomial, and let S be a subset from the field F. If each X_i is chosen independently in S

$$\Pr[P(X_1, \ldots, X_m) = 0] \leq \frac{d}{|S|}$$

- Sanity check: if m = 1, a non-zero degree-d polynomial has at most d roots

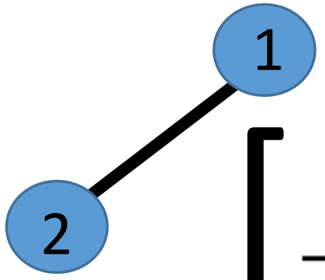
- If |F| > 3d, how can we tell if P is the all zeros polynomial w.pr. 2/3?

- Choose $X_1, \ldots, X_m$ independently from F, and evaluate $P(X_1, \ldots, X_m)$

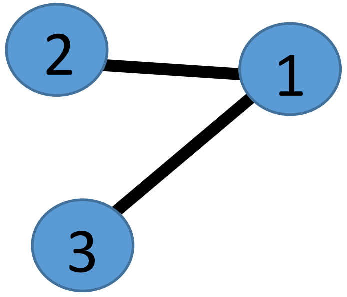
Tutte Matrix

- If G is a graph on vertices $v_1, \ldots, v_n$, the Tutte matrix is a |V| x |V| matrix M(G) with

$$M(G)_{i,j} = \begin{cases} x_{i,j} & \textit{if } \{v_i, v_j\} \in E \textit{ and } i < j \\ -x_{j,i} & \textit{if } \{v_i, v_j\} \in E \textit{ and } i > j \\ 0 & \textit{if } (v_i, v_j) \notin E \end{cases}$$



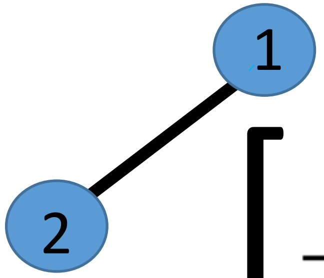
$$\begin{bmatrix} 0 & x_{12} \\ -x_{12} & 0 \end{bmatrix}$$



$$\begin{bmatrix} 0 & x_{12} & x_{13} \\ -x_{12} & 0 & 0 \\ -x_{13} & 0 & 0 \end{bmatrix}$$

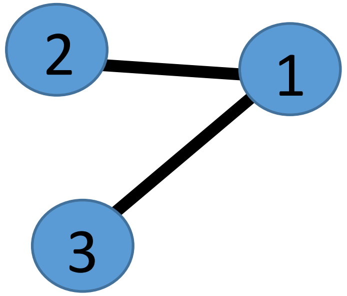
Tutte Determinant Theorem

- [Tutte] A graph has a perfect matching if and only if the determinant of M(G) is not the zero polynomial (a matching is perfect if all nodes are matched)



$$\begin{bmatrix} 0 & x_{12} \\ -x_{12} & 0 \end{bmatrix}$$

$$\det\big(M(G)\big) = x_{12}^2$$

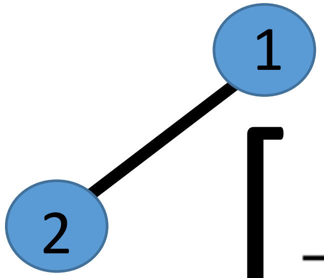


$$\begin{bmatrix} 0 & x_{12} & x_{13} \\ -x_{12} & 0 & 0 \\ -x_{13} & 0 & 0 \end{bmatrix}$$

$$\det\big(M(G)\big) = 0$$

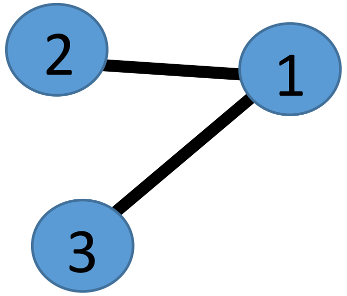
Tutte Determinant Theorem

- [Tutte] A graph has a perfect matching if and only if the determinant of M(G) is not the zero polynomial (a matching is perfect if all nodes are matched)



$$\begin{bmatrix} 0 & x_{12} \\ -x_{12} & 0 \end{bmatrix}$$

$$\det\big(M(G)\big) = x_{12}^2$$



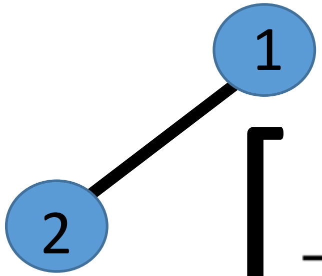
$$\begin{bmatrix} 0 & x_{12} & x_{13} \\ -x_{12} & 0 & 0 \\ -x_{13} & 0 & 0 \end{bmatrix}$$

$$\det\big(M(G)\big) = 0$$

- det(M(G)) is a polynomial of degree at most n, and could have n! terms

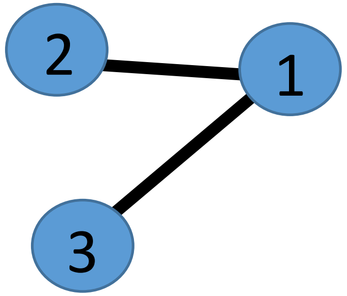
Tutte Determinant Theorem

- [Tutte] A graph has a perfect matching if and only if the determinant of M(G) is not the zero polynomial (a matching is perfect if all nodes are matched)



$$\begin{bmatrix} 0 & x_{12} \\ -x_{12} & 0 \end{bmatrix}$$

$$\det\left(M(G)\right) = x_{12}^2$$



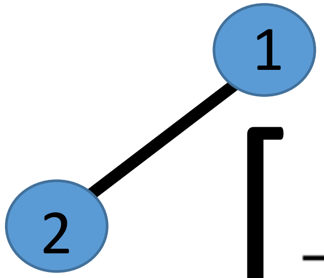
$$\begin{bmatrix} 0 & x_{12} & x_{13} \\ -x_{12} & 0 & 0 \\ -x_{13} & 0 & 0 \end{bmatrix}$$

$$\det\left(M(G)\right) = 0$$

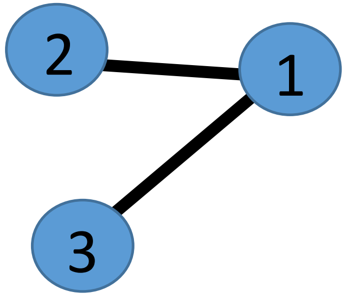
- det(M(G)) is a polynomial of degree at most n, and could have n! terms
- *How can we determine if G has a perfect matching with probability at least 2/3?*

Tutte Determinant Theorem

- [Tutte] A graph has a perfect matching if and only if the determinant of M(G) is not the zero polynomial (a matching is perfect if all nodes are matched)



$$\begin{bmatrix} 0 & x_{12} \\ -x_{12} & 0 \end{bmatrix}$$



$$\begin{bmatrix} 0 & x_{12} & x_{13} \\ -x_{12} & 0 & 0 \\ -x_{13} & 0 & 0 \end{bmatrix}$$

$$\det\big(M(G)\big) = x_{12}^2$$

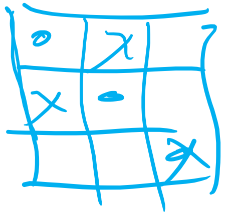
$$\det\big(M(G)\big) = 0$$

- $\det(M(G))$ is a polynomial of degree at most n, and could have n! terms
- *How can we determine if G has a perfect matching with probability at least 2/3?*
- Choose a field F with |F| > 3n, randomly fill in the $x_{i,j}$ values, and compute determinant!

Finding a Perfect Matching

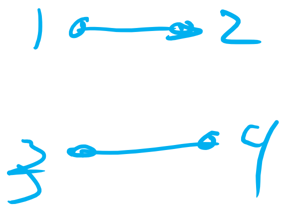
- We can quickly determine if G has a perfect matching

- Can reduce the error probability to $1/n^3$, say, by choosing $|F| = n^4$

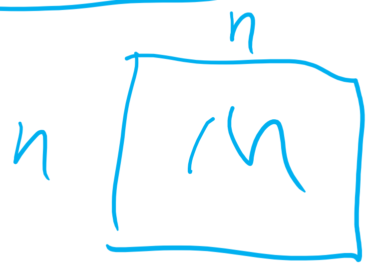


$$\det(M) = \sum_{\text{Perms } \pi} \text{sign}(\pi) \cdot \prod_{1 \le i \le n} M_{i\pi(i)}$$

$$x_{12}(-x_{12})x_{34}(-x_{34})$$



$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & & x_{12} & \cdot & \cdot \\ 2 & -x_{12} & & & \cdot \\ 3 & \cdot & \cdot & & x_{34} \\ 4 & \cdot & \cdot & -x_{34} & \end{array}$$



Finding a Perfect Matching

- We can quickly determine if G has a perfect matching
- Can reduce the error probability to $1/n^3$, say, by choosing $|F| = n^4$
- But how to output the edges in the perfect matching?

Finding a Perfect Matching

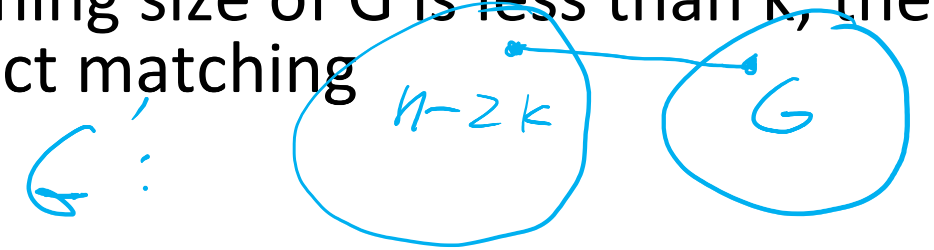
- We can quickly determine if G has a perfect matching
- Can reduce the error probability to $1/n^3$, say, by choosing $|F| = n^4$
- But how to output the edges in the perfect matching?
- For each edge e,
  - Remove e and see if there is still a perfect matching
  - If there is no perfect matching, put e back in G, otherwise discard e
- At the end, will be left with exactly n/2 edges in a perfect matching

Finding a Maximum Matching

- Can we find a maximum matching if we can find a perfect matching?

Finding a Maximum Matching

- Can we find a maximum matching if we can find a perfect matching?
- Given a graph G, connect n-2k new nodes to every node in G
- If G has a matching of size at least k, then this new graph has a perfect matching
- If the maximum matching size of G is less than k, then this new graph does not have a perfect matching



- Binary search on k