Algebraic Techniques

Finger printing

- Suppose we need to compare two very large binary strings x, y (e.g. two copies of a data set, check for consistency)
- Say one party has x (say Alice) and the other party (Bob) has y;

  Complexity $\approx$ # bits communicated
- Clearly any deterministic algorithm needs to send all the bits. (so if $|x|=|y|=n \longrightarrow n$ time needed)
- This is a lower bound for communication complexity
- Can we do better?
- Randomness can help: we see how we can reduce this down to $O(\log n)$ bits.
- In this lecture we work with finite fields $\mathbb{F}$.

  For simplicity we may assume $\mathbb{F}$ is the fild $\mathbb{Z}_p = \{0, 1, 2, \ldots, p-1\}$ for some prime p with operations being modular arithmetic in mod p.
- Suppose $x = a_1 a_2 \cdots a_n$ $\quad y = b_1 b_2 \cdots b_n$

Consider polynomials $A(x) = \sum_{i=1}^{n} a_i x^i$ $\quad B(x) = \sum_{i=1}^{n} b_i x^i$

- Pick a large enough p, say $p \gg n$ and a random $r \in \mathbb{Z}_p$; Alice computes $A(r)$, Bob computes $B(r)$ } compare

- Send $A(r)$ to Bob and compare

  $O(\log p)$ bits say $p \approx \frac{n}{\varepsilon}$

  $\approx O(\frac{\log n}{\varepsilon})$ bits

- Say consistent if $A(r) = B(r)$; o.w. say no

- Clearly if $x = y \longrightarrow$ returns "consistent"

- What is the probability of error?

  (wrong "consistent" when $x \neq y$)

- Must have $A(x) \neq B(x)$ but $A(r) = B(r)$

  $\longrightarrow (A-B)(r) = 0$

- Note A, B, and $A-B$ are degree n polynomials.

  $\longrightarrow$ have at most n roots.

For a random $r \in \mathbb{Z}_p$ to be a root the prob. is $\leq \frac{n}{p} = \varepsilon$

- If we choose $p = n^2 \longrightarrow$ complexity $2 \log n$

  error probability $\frac{1}{n}$

Fingerprinting: Freivalds' Technique

- Suppose we are given matrices A, B, and C and we want to verify if $AB = C$. (all $n \times n$ matrices)
- Fast matrix multiplication takes $O(n^w)$ where $w \approx 2.371\ldots$
- Suppose we pick a random $r \in \{0,1\}^n$

  we check if $ABr = Cr$:

  - Compute $x = Br$: $O(n^2)$ time
  - Compute $y = Ax$: $O(n^2)$ time
  - Compute $z = Cr$: $O(n^2)$ time
- Clearly if $AB = C \longrightarrow y = z$,

  what if $AB \neq C$ but $ABr = Cr$?

  let $D = (AB - C)$. So must have $Dr = 0$ and $D \neq 0$

  if $D \neq 0 \longrightarrow \exists$ a non-zero row, WLOG say $\vec{d}$ is the first row of D and $\vec{d} \neq 0$

$$\sum_{i=1}^{n} d_i \cdot r_i = 0 \longrightarrow r_1 = \frac{-\sum_{i=2}^{n} d_i \cdot r_i}{d_1} \quad (*)$$

  Assuming that all r_i's $(i \geq 2)$ are chosen before r_1

  $\longrightarrow$ there is a unique value $v \in \mathbb{F}$ that is equal to RHS of $(*)$ and the probability that r_1 is equal to that is $\leq \frac{1}{2}$
- We can repeat the process k times to make

error probability $\leq \frac{1}{2^k}$.

Polynomial Identity Testing

- We can extend these ideas for comparing polynomials.
- Given polynomials P, Q, R over $\mathbb{F}$, we want to check if $P(x) \cdot Q(x) = R(x)$ → degree n polynomials
- We can use Fast Fourier Transforms (FFT) to compute multiplication in $O(n \log n)$ time
- Faster (randomized) testing:
  - let $S \subset \mathbb{F}$ and pick (randomly) $r \in S$
  - Compute $P(r)Q(r) - R(r)$: $O(n)$ time
  - Say yes/no if the result is 0
- What is the probability of error?

  if $D(x) = P(x)Q(x) - R(x)$ is not identically zero but $D(r) = 0$ (i.e. r is a root).

  degree of $D(x) \leq 2n \longrightarrow \leq 2n$ roots $\longrightarrow$

  probability of error $\leq \frac{2n}{|S|}$

General setting: suppose we have multi-variable polynomials say $P(x_1 \ldots x_n)$. The degree of a term is the sum of degrees of the var's (e.g. degree of x^2yz^3 is 6)

- We might have an implicit representation of polynomial

- We might have an implicit representation of polynomial

e.g. $p(x_1, x_2, x_3) = \det \begin{bmatrix} 2x_1+x_2 & 1 & x_3^2 \\ 0 & x_2-x_3 & x_1+6x_3 \\ 5x_2^5+x_3 & 12 & x_2^4-x_3 \end{bmatrix}$

- We may also have access to polynomials through a blackbox that can evaluate at a given point.
- There are NO polynomial time deterministic alg's to check polynomial identity testing in these situations!

Theorem (Shwartz-Zippel): let $Q(x_1 \cdots x_n)$ be a multivariate polynomial of total degree d. For $S \subset \mathbb{F}$ let $r_1 \cdots r_n$ be selected u.r. from S. Then

$$\Pr[Q(r_1 \cdots r_n) = 0 \mid Q(x_1 \cdots x_n) \neq 0] \leq \frac{d}{|S|}.$$

Proof: By induction on n (see the references online)

Perfect Matchings

- How can we use algebraic technique to design efficient (parallel) algorithms for detecting & finding perfect matchings?
- Given bipartite graph $G = (U \cup V, E)$, $|U| = |V| = n$

Definition: The Tutte matrix of G is an $n \times n$ matrix M s.t.

$$M_{ij} = \begin{cases} 0 & u_i v_j \notin E \\ x_{ij} & u_i v_j \in E \end{cases}$$

Determinant of M can be used to check existence of matchings in G.

Theorem (Edmond): $Det(M) \neq 0 \iff$ G has a perfect matching.

Proof: Using the definition of determinant:

$$Det(M) = \sum_{\pi \in P_n} (-1)^{Sign(\pi)} \prod M_{i,\pi(i)}$$ where P_n is the set of all permutations of $\{1, \dots, n\}$. There is a one-to-one correspondence between potential perfect matchings in G and permutations of $[n]$

- A permutation π contributes a non-zero term to Det (M) iff $\{u_1 v_{\pi(1)}, u_2 v_{\pi(2)}, \dots, u_n v_{\pi(n)}\}$ is a perfect matching in G.
- So no perfect matching $\longrightarrow$ all terms are zero & Det (M) = 0
- if there is a perfect matching there is a non-zero term & since terms are distinct (different set of variables) no two terms cancel out each other.

Randomized matching: Using this & Schwartz-Zippel for polynomial identity testing we can check if

for polynomial identity testing we can check if G has perfect matchings:

$Det(M)$ is a degree n polynomial. So if we pick our field large enough (say $p \geq 2n$) then if we pick a random $r \in \mathbb{F}$ and check if $Det(M) = 0$ or not we will have

$$Pr[G \text{ has a perfect matching } \& \text{ we accept }] = 1$$

$$Pr[G \text{ has a perfect matching } \& \text{ we reject }] \leq \frac{n}{p} \leq \frac{1}{2}$$

Time complexity: Using Gaussian method we can compute determinant in $O(n^3)$. More efficient algorithms take $O(n^\omega)$ where $\omega \approx 2.371$

How to find a perfect matching? having a "checking" algorithm, we can design an algorithm to "find" one.

for each $e = uv \in E$ check if $G - e$ has a perfect matching.

- if yes delete e from G
- else recurse on $G - e$

Parallel algorithm & isolation lemma

One advantage of algebraic techniques: can be made parallel to run on massive clusters

For e.g. Computing determinant can be done in parallel

Theorem: The determinant of an $n \times n$ matrix can be computed in $O(\log^2 n)$ time using $O(n^{\omega+2})$ processors.

So we can have a fast parallel algorithm to detect if G has a perfect matching or not.

How to find a perfect matching in parallel?

if each processor deletes one edge $e=uv$ and check if $G-e$ has a perfect matching?

how do you coordinate different processors to search for the same matching?

Lemma (Isolation lemma): let $X = \{x_1 \dots x_m\}$ be a set of elements and $F = \{S_1 \dots S_k\}$ a family of subsets of X (all distinct). Suppose $w: X \to \{1 \dots 2m\}$ is a positive integer function chosen uniformly at random for each element. Then

$$\Pr[\text{min\_weight set in } F \text{ is unique}] \geq \frac{1}{2}$$

Remark: This lemma seems counter-intuitive.

Note that k can be very large, say $k = 2^m$.

then weight of each set is in the range $\{1, \ldots, 2m^2\}$, so we expect to see $\frac{2^m}{2m^2}$ sets of each weight in $\{1, \ldots, 2m^2\}$!!

Proof: Fix an element x_i and let:

Y_i^+: the set of all sets in F containing x_i

Y_i^-: the " " " " " " Not containing x_i

$$\alpha_i : \min_{S \in Y_i^-} w(S) - \min_{S' \in Y_i^+} w(S' - \{x_i\})$$

Note that if $\alpha_i < w(x_i)$ then x_i does not belong to any min-value set: Since there is a $S \in Y_i^-$ whose weight is smaller than $w(S' - \{x_i\})$ for any $S' \in Y_i^+ \longrightarrow w(S) < w(S' - \{x_i\}) + w(x_i)$

$$= w(S')$$

- if $\alpha_i > w(x_i)$ then every min-value set must contain x_i.

We say x_i is ambiguous if $\alpha_i = w(x_i)$

- Note: α_i is independent of $w(x_i)$ and

- Note: α_i is independent of $w(e_i)$ and $w(x_i)$ is chosen uniformly from $\{1 \ldots 2m\}$

$\longrightarrow \Pr[x_i \text{ is ambiguous}] \leq \frac{1}{2m}$

Union bound: $\Pr[\text{some } x_i \text{ is ambiguous}] \leq \frac{1}{2}$

if there is no ambiguous element $\longrightarrow$ min_value set is unique.

Matching using isolation lemma:

let $X = E$ and set the edge weights randomly from $\{1, \ldots, 2m\}$, then with prob. $\geq \frac{1}{2}$ the min-weight perfect matching is unique.

How to find it?

In the Tutte matrix let $x_{ij} = 2^{w_{ij}}$ where w_{ij} is the weight of $u_i v_j$ and let B be this (modified) Tutte matrix.

Lemma: if there is a unique min_weight perfect matching in G with weights W then:

i) $\text{Det}(B) \neq 0$ and

ii) the largest power of 2 dividing $\text{Det}(B)$ is 2^W

(1) The largest power of 2 dividing Det(B) is 2^W

Proof: other than the min-weight perfect matchy all others have weight $\geq W+1$. So the terms in Det(B) are of the form $\{\pm 2^W, \pm 2^{W+1}, \pm 2^{W+2}, \ldots\}$

Taking a factor of 2^W, since the min-value is unique we get a sum of the form $2^W(1 + a_1 \cdot 2 + a_2 \cdot 2^2 + \ldots)$ where a_i's are integer. The sum inside ( ) is odd and $\neq 0$

$\longrightarrow$ Det(B) $\neq 0$ and 2^W is the largest divider. ▣

Parallel algorithm for matching:

- Pick w_{ij} uniformly at random from $\{1 \ldots 2m\}$
- Compute the Tutte matrix B and compute Det(B) and find largest W s.t. $2^W \mid$ Det(B)
- if Det(B) $= 0 \longrightarrow$ no perfect matching
- For each $u_i v_j \in E$ do in parallel
  - Compute Det(B_{ij}) where B_{ij} is B after deleting row i & column j
  - if Det(B_{ij}) $\neq 0$ find the largest $2^{W_{ij}} \mid$ Det(B_{ij})
  - if $W_{ij} + w_{ij} = W$ then output $u_i v_j$