CS 761 : Randomized Algorithms , Spring 2018 , Waterloo

Lecture 8 : Polynomial identity testing

We have seen that simple algebraic ideas used in hashing have various applications.

Today we will see other simple algebraic ideas that are useful in designing fast and parallel algorithms, and next time we will apply these ideas to the design of network coding.

Polynomial identity testing

We have seen the string equality example in L01 where single variable polynomial identity testing is useful.

Now, we see a multivariate generalization, that is an important tool in designing randomized algorithms.

We are given a multivariate polynomial $P(x_1, \ldots, x_n)$ and the objective is to determine if $P(x_1, x_2, \ldots, x_n) \equiv 0$, i.e. the polynomial is symbolically equal to the zero polynomial.

Of course, if the polynomial is given explicitly (e.g. $P(x_1, x_2, x_3) = x_1 x_2^2 x_3^3 + 4x_1^6 x_3 + x_2 x_3^2$), then this problem is straightforward, but the polynomial can also be given compactly (e.g. $P(x_1, x_2, x_3) = (x_1 - x_2^2)(x_3^3 + x_2^4)\ldots(x_1^2 + x_3)$, or $P(x_1, x_2, x_3) = \det\begin{pmatrix} x_1 & x_3 & x_2 + x_4^2 \\ 0 & x_2 - x_3 & 1 \\ x_2^3 & x_1 & x_4^2 x_2 x_3 \end{pmatrix}$) and the problem becomes difficult.

In general, there is no known deterministic polynomial time algorithm for this problem.

On the other hand, there is a simple randomized algorithm for this problem, by generalizing the idea that there are not many roots in a low degree polynomial.

In the following, when we talk about finite fields, you can just think of modular arithmetic over prime.

Lemma (Schwartz-Zippel Lemma) Let $Q(x_1, \ldots, x_n) \in \mathbb{F}[x_1, \ldots, x_n]$ be a multivariate polynomial of total degree d.

Fix any finite set $S \subseteq \mathbb{F}$, and let $r_1, \ldots, r_n$ be chosen independenty and uniformly randomly from S.

Then, $\Pr[Q(r_1, \ldots, r_n) = 0] \leq \frac{d}{|S|}$ if Q is not identically equal to zero.

(The degree of any term is the sum of the exponents, and the total degree is the maximum degree.)

proof We prove by induction. The base case (single variate polynomial) $n = 1$ is discussed before.

To apply induction, we group the terms of $Q(x_1, \ldots, x_n)$ based on the degree of x_1 to obtain

$$Q(x_1, \ldots, x_n) = \sum_{i=0}^{k} x_1^i \, Q_i(x_2, \ldots, x_n),$$

where k is the largest exponent of x_1 and each $Q_i(x_2, \ldots, x_n) \not\equiv 0$.

We condition on the event that $x_2 = r_2$, $x_3 = r_3, \ldots, x_n = r_n$.

Then $\Pr[\quad] \leq \frac{d-k}{|S|}$

Since $Q_k(x_2,\dots,x_n) \not\equiv 0$ and it has $n-1$ variables, by induction, $\Pr(Q_k(x_2,\dots,x_n)=0) \le \frac{d-k}{|S|}$ as the total degree of Q_k is at most $d-k$ (since x_1 is of degree k in the term $x_1^k Q_k(x_2,\dots,x_n)$).

Now, assuming $Q_k(r_2,\dots,r_n) \neq 0$, then $g(x_1) = Q(x_1, r_2, \dots, r_n)$ is a non-zero degree k single variate polynomial.

So, by the base case, $\Pr(g(r_1)=0) \le \frac{k}{|S|}$.

Combining, we have $\Pr(Q(r_1,\dots,r_n)=0)$

$= \Pr(Q(r_1,\dots,r_n)=0 \mid Q_k(r_2,\dots,r_n)=0) \cdot \Pr(Q_k(r_2,\dots,r_n)=0) + \Pr(Q(r_1,\dots,r_n)=0 \mid Q_k(r_2,\dots,r_n)\neq 0) \cdot \Pr(Q_k(r_2,\dots,r_n)\neq 0)$

$\le 1 \cdot \frac{d-k}{|S|} + \frac{k}{|S|} \cdot 1 = \frac{d}{|S|}$. □

Determinant testing

To check whether the determinant of an $n \times n$ matrix is identically equal to zero, we can pick a large enough prime field of size $\Theta(\text{poly}(n))$. This will ensure that with high probability that the determinant is still nonzero in this field (if originally it is nonzero). Then, we can just substitute random field elements in the variables and compute the numerical value of the determinant in polynomial time. By Schwarz-Zippel, if the determinant is nonzero, then the (success) probability that the numerical value is nonzero is very high.

Bipartite matching

Given a bipartite graph $G = (U, V; E)$ with $U = \{u_1,\dots,u_n\}$ and $V = \{v_1,\dots,v_n\}$, the bipartite matching problem is to find a maximum cardinality subset of vertex-disjoint edges.

A perfect matching is a matching of n edges.

Theorem (Edmonds) Let A be an $n \times n$ matrix with $A_{ij} = \begin{cases} x_{ij} & \text{if } u_i v_j \in E \\ 0 & \text{otherwise} \end{cases}$ where x_{ij} is a variable.

Then G has a perfect matching if and only if $\det(A) \not\equiv 0$, i.e. $\det(A)$ is a non-zero polynomial.

Proof Recall that $\det(A) = \sum_{\text{permutation } \sigma} \text{sign}(\sigma) \cdot A_{1,\sigma(1)} A_{2,\sigma(2)} \cdots A_{n,\sigma(n)} = \sum_{\text{permutation } \sigma} \text{sign}(\sigma) \prod_{i=1}^{n} A_{i,\sigma(i)}$.

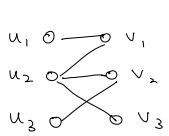
Notice that a permutation defines a bijection between U and V, and so each permutation corresponds to a potential perfect matching between U and V, i.e. $\{u_1 v_{\sigma(1)}, u_2 v_{\sigma(2)}, \dots, u_n v_{\sigma(n)}\}$.

A permutation σ contributes a non-zero term to $\det(A)$ iff $\{u_1 v_{\sigma(1)}, \dots, u_n v_{\sigma(n)}\}$ is a perfect matching in the graph G.

So, if there is no perfect matching in G, then every term is zero and thus $\det(A) \equiv 0$.

On the other hand, if there is a perfect matching in G, then some term is nonzero. Since each term uses a different subset of variables, these terms will not cancel out, and so $\det(A) \neq 0$. □

Pictorically,



$$\begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \end{array}\begin{array}{c} \begin{array}{ccc} v_1 & v_2 & v_3 \end{array} \\ \begin{bmatrix} x_{11} & 0 & 0 \\ x_{21} & x_{22} & x_{23} \\ 0 & x_{32} & 0 \end{bmatrix} \end{array} \qquad \begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \end{array}\begin{array}{c} \begin{array}{ccc} v_1 & v_2 & v_3 \end{array} \\ \begin{bmatrix} 17 & 0 & 0 \\ 33 & 22 & 18 \\ 0 & 61 & 0 \end{bmatrix} \end{array}$$

Randomized algorithm: Using Edmond's theorem and the Schwartz-Zippel lemma, we can obtain a "simple" randomized algorithm to check if a bipartite graph has a perfect matching or not.

The algorithm is simply to pick a large enough field $\mathbb{F}$.

You can simply think of doing modular arithmetic over a large prime p.

Then, for each edge $u_iv_j \in E$, set x_{ij} to be a random element in $\mathbb{F}$ (e.g. $\{0, \cdots, p-1\}$) independently.

Compute $\det(A)$ over the field $\mathbb{F}$. Return YES if and only if $\det(A) \neq 0$.

Analysis: If there is no perfect matching in G, then $\det(A) = 0$ always, no matter what we substitute.

On the other hand, if there is a perfect matching in G, then $\det(A) \neq 0$ by Edmond's theorem.

Note that $\det(A)$ is a multivariate polynomial of total degree n.

By Schwartz-Zippel lemma, if we put in random values into the variables of a non-zero polynomial, the probability that $\det(A) = 0$ is at most $\frac{n}{|\mathbb{F}|}$.

So, if we set a field size to be at least $100n$ (a prime p at least $100n$), then this probability is at most $\frac{1}{100}$. In other words, the algorithm succeeds with probability at least 0.99.

Time complexity: The most difficult step is to compute the determinant.

Since we put in actual values, the determinant can be computed in $O(n^3)$ time by Gaussian elimination.

There is also an $O(n^\omega)$-time fast matrix multiplication algorithm with $\omega \approx 2.37$.

Theoretically, this is the fastest known algorithm for bipartite matching when the graph is dense.

Non-bipartite matching (optional)

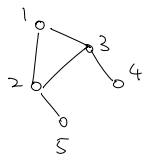
There is also an algebraic formulation for matching in general graphs.

Given a graph $G = (V, E)$, we consider the following $|V| \times |V|$ matrix A such that

for each edge $e = ij$, we have $A_{ij} = x_e$ and $A_{ji} = -x_e$ and other entries to be zero.

$$\begin{array}{c} \\ 1 \\ \vdots \end{array}\begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & -x_{12} & x_{13} & 0 & 0 \\ \end{bmatrix} \end{array}$$

for each edge $e=ij$, we have $A_{ij} = x_e$ and $A_{ji} = -x_e$ and other entries to be zero.



$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & -x_{12} & x_{13} & 0 & 0 \\ x_{12} & 0 & x_{23} & 0 & -x_{25} \\ -x_{13} & -x_{23} & 0 & x_{34} & 0 \\ 0 & 0 & -x_{34} & 0 & 0 \\ 0 & x_{25} & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Theorem (Tutte) G has a perfect matching if and only if $\det(A) \neq 0$.

We won't give a formal proof here. The idea is that each term in the determinant is corresponding to a collection of vertex disjoint cycles whose union covers the whole vertex set. The key point is that those collections with odd cycles won't show up, because they come in pairs (by reversing an odd cycle) and they cancel each other (because of the skew-symmetry of the signed matrix). So, all terms in the determinant correspond to even cycle cover, and they can be decomposed into two perfect matchings, and one can check that they won't cancel out each other.

Remark: Many combinatorial optimization problems have an algebraic formulation.

Fast algorithms: We briefly mention how to find a perfect matching quickly using this approach. For simplicity, we only consider the bipartite matching problem.

A straightforward way is to check for each edge ij, whether $G-\{i,j\}$ has a perfect matching or not: if yes, we found one edge and recurse; if not, we delete ij and try another edge.

It will take $O(m \cdot n^\omega)$ time, where $\omega \approx 2.373$ is the matrix multiplication exponent, and this is too slow.

The next observation is that we can read off whether an edge is in a perfect matching by looking at the inverse. This is because $(A^{-1})_{i,j} = C_{j,i}/\det(A)$ where $C_{j,i}$ is the (j,i)-th cofactor of A. Note that $C_{j,i} = \pm \det(A_{V-j, V-i})$ which is nonzero iff there is a perfect matching in $G-\{i,j\}$. So, an edge ij is in a perfect matching iff $(A^{-1})_{ij} \neq 0$.

This leads to an $O(n^{\omega+1})$-time algorithm for finding a perfect matching, by identifying an edge, updating the matrix and recomputing the inverse again (so as to identify another edge, so on).

The next observation is that we don't need to recompute the inverse in $O(n^\omega)$ time, because it is just a "rank-one update" in the matrix when we fix an edge to the matching, and its inverse can be "updated" in $O(n^2)$ time by the Sherman-Morrison-Woodbury formula.

So, this gives us an $O(n^3)$ algorithm.

Mucha-Sankowski and later Harvey showed that one can find a perfect matching in $O(n^\omega)$ time, by using a divide and conquer method in applying the Sherman-Morrison-Woodbury formula.

See the theses of Piotr Sankowski and Nick Harvey for various applications of this algebraic approach.

Parallel Algorithm and Isolation Lemma

Another advantage of this algebraic approach is to design fast parallel algorithms.

The main reason is that many algebraic problems (such as computing determinant) can be computed efficiently in parallel using a divide-and-conquer approach.

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

Combining this theorem with the algebraic matching algorithm, there is an efficient parallel algorithm to check whether a graph has a perfect matching or not (YES or NO).

We will explain below how to return a perfect matching when there is one.

The difficulty in designing parallel algorithm is to coordinate the processors to search for the same matching.

There are graphs with exponentially many matchings.

The isolation lemma provides a surprising and elegant way to resolve this issue.

Lemma (Isolation Lemma) Given a set system on a ground set of n elements, if we assign a weight independently and uniformly randomly from $\{1, 2, \ldots, 2n\}$, then the probability that there is a unique minimum weight set is at least $\frac{1}{2}$.

Remark: This is quite counter-intuitive. A set system could have $\Omega(2^n)$ sets.

On average, there are $\Omega(2^n/(2n^2))$ sets of a given weight, as the maximum weight is $\leq 2n^2$.

But the isolation lemma says with good probability there is only one set of minimum weight.

Proof Let $\mathcal{F}$ be the set system. Let v be an arbitrary element.

Let $\mathcal{F}_v$ be the set of subsets which contain v. Let $\mathcal{F}_{\bar{v}}$ be the set of subsets which don't contain v.

Consider the quantity $\alpha_v = \min_{S \in \mathcal{F}_{\bar{v}}} w(S) - \min_{R \in \mathcal{F}_v} w(R-v)$.

Note that if $\alpha_v < 0$, then v does not belong to any minimum weight set, as there is a

subset S which does not contain v and its weight is smaller than the weight of every subset that contains v (even if the weight of v is not considered yet).

More accurately, by a similar reasoning, if $\alpha_v < w(v)$, then v doesn't belong to any minimum weight set. On the other hand, if $\alpha_v > w(v)$, then every minimum weight set must contain v.

We say an element v is <u>ambigious</u> if $\alpha_v = w(v)$.

Note that α_v is a quantity independent of $w(v)$, and $w(v)$ is chosen uniformly randomly from $\{1, 2, \ldots, 2n\}$.

Therefore, the probability that v is ambiguous is at most $\frac{1}{2n}$.

By union bound, the probability that some element is ambiguous is at most $\frac{1}{2}$.

Finally, observe that if two different sets A, B have the minimum weight, then any element in $A - B \neq \phi$ and $B - A \neq \phi$ must be ambiguous.

Therefore, we conclude that there are two different sets with the minimum weight is at most $\frac{1}{2}$. □

For the matching problem, we assign each edge an independent random weight from $\{1, 2, \ldots, 2m\}$.

By the isolation lemma, with the set system being all possible perfect matchings, there is a unique minimum weight perfect matching with probability at least $\frac{1}{2}$.

So, now our goal is to output this unique minimum weight perfect matching.

<u>Minimum weight perfect matching</u>

For each edge e, we put the value $2^{w(e)}$ to the variable x_e.

Let the unique minimum weight perfect matching have weight $W \leq 2m^2$.

Then, only one term in the determinant has value 2^W.

All other terms in the determinant have value $2^{W'}$ for $W' > W$, an even number times bigger than 2^W.

<u>Computing the minimum weight</u>

We can search for the minimum weight in parallel, using the following observation: $\frac{\det(A)}{2^k} = \begin{cases} \text{even} & \text{if } k < W \\ \text{odd} & \text{if } k = W \\ \text{not divisible} & \text{if } k > W \end{cases}$.

We can find this k in parallel, after computing $\det(A)$ in parallel.

Note that $\det(A)$ can still be computed efficiently, as each entry has at most $2m$ bits, and this is where the isolation lemma is useful in that it does the job by only using small weights.

<u>Finding the minimum weight perfect matching</u>

For each edge e, in parallel, we want to check whether e belongs to the unique minimum perfect matching.

Let $e=uv$, and G_{uv} be the graph obtained from G by deleting the vertices u and v.

Then, note that $e=uv$ is in the unique minimum weight perfect matching in G if and only if the minimum weight perfect in G_{uv} has weight $W-w(uv)$.

This again can be checked in parallel using different processors for different G_{uv}, and so the minimum weight perfect matching can be constructed efficiently in parallel.

Remark: In general, other algorithms based on algebraic formulations can be made parallel.

References: Material in this lecture is from chapter 7 of "Randomized Algorithms" by Motwani and Raghavan.

Open question: Given a graph where each edge is colored red or blue, we would like to determine if there is a perfect matching with exactly k red edges. It's a good exercise to show that there is a randomized polynomial time algorithm for this problem using the algebraic approach, but it has been an open question whether there is a deterministic algorithm for the problem.

Open question: It is a major open question to find a deterministic parallel algorithm for matching. Recently, there is some interesting progress in derandomizing the isolation lemma, which leads to a parallel algorithm using $n^{O(\log n)}$ many processors. Nice question to think about.

Open question: The really big open question is a deterministic algorithm for polynomial identity testing.