CSE 521: Design and Analysis of Algorithms I **Spring 2016**

Lecture 5: Identity Testing & Isolation Lemma

Lecturer: Shayan Oveis Gharan *April 11th* *Scribe: Robbie Weber*

5.1 Introduction

Our focus in this lecture is the Schwartz-Zippel Lemma, which gives a randomized algorithm for polynomial identity testing – that is telling whether a given polynomial is the 0 polynomial or a non-trivial polynomial.We will discuss two applications: matrix multiplication testing and perfect matchings.

The formal statement of the lemma is as follows:

Theorem 5.1 (Schwartz-Zippel Lemma)**.** *Let $p(x_1, x_2, \ldots, x_n)$ be a non-zero polynomial of degree d, and let S be a finite set of numbers. If we assign each x_i an element of S independently and uniformly at random then $\mathbb{P}[p(x_1, x_2, \ldots x_n) = 0] \leq \frac{d}{|S|}$.*

To use this lemma, we first design a polynomial, p, such that whether p is the 0-polynomial or non-zero is meaningful to us. To differentiate those cases, we choose a set S, set the x_i randomly and evaluate p. If p evaluates to something non-zero, then it must be a non-zero polynomial. Conversely if p evaluates to 0, by the lemma, we conclude that it it most likely because p is in fact the 0 polynomial.

Before proving the lemma, we will discuss a simpler version and application of Matrix Multiplication Testing.

5.2 Matrix Multiplication Testing

Suppose we are given three $n \times n$ matrices, A, B, C. Our goal is to determine whether $AB = C$. One possible (non-randomized) approach is to simply multiply A and B check if every entry matches C. This would take $O(n^\omega)$ time (where ω is the matrix-multiplication exponent. The best known bound is approximately 2.373.)

We desire a faster (randomized) algorithm. Consider choosing a random vector x. If $AB = C$ then $ABx = Cx$. On the other hand, we will show that if $AB \neq C$ then for a random x, $ABx \neq Cx$ with some good probability.

First, we note that this is faster than just multiplying A and B. We can calculate ABx by first multiplying Bx and then multiplying A by the result, which requires only $O(n^2)$ time.

Rather than using the Schwartz-Zippel Lemma to finish this proof, we will show directly that if $AB \neq C$ then $\mathbb{P}[ABx = Cx]$ is small. Our precise claim is the following

Claim 5.2. *Let $S = \{0, 1, \ldots, k\}$ Let x be a vector whose entries are independently chosen, uniformly random elements of S. If $AB \neq C$, then $\mathbb{P}[ABx = Cx] \leq \frac{1}{k+1}$*

Our proof will exploit the **Principle of Deferred Decisions**, which is typically used in analyzing randomized algorithm. Here is the high level overview of the technique. Say, we have a set of independently chosen random variables $X_1, \ldots, X_n$. We can first fix a few these variables arbitrarily, say $X_1 = x_1, \ldots, X_j = x_j$. Then, we can use the fact that $X_{j+1}, \ldots, X_n$ are chosen independently to say that, conditioned on $X_1 = x_1, \ldots, X_j = x_j$, the probability of a bad event is very very small.

Proof. Let $a_{i,j}$ be the $(i,j)^{\text{th}}$ entry of the product AB and $c_{i,j}$ the $(i,j)^{\text{th}}$ entry of C. We also define a_i to be the i^{th} row of AB, with c_i to be the i^{th} row of C.

Suppose $AB \neq C$, then there are indices i, j such that $a_{i,j} \neq c_{i,j}$. We wish to show $\mathbb{P}[\langle a_i, x\rangle = \langle c_i, x\rangle] \leq \frac{1}{k+1}$. By the Principle of Deferred Decisions, fix every entry of x except for x_j. Rearranging the formula $\langle a_i, x\rangle = \langle c_i, x\rangle$, we see that $\langle a_i, x\rangle = \langle c_i, x\rangle$ only if

$$x_j = \frac{\sum\limits_{k \neq j, 1 \leq k \leq n} (c_{i,k} - a_{i,k}) x_k}{a_{i,j} - c_{i,j}}$$

Since $a_{i,j} \neq c_{i,j}$, the RHS is a well-defined fraction. We have fixed all the x_i except x_j, thus the right hand side is just a number. Since x_j is set uniformly from the elements of S, the probability that x_j is equal to the RHS is at most $\frac{1}{k+1}$. □

As our last comment on this problem, we remark that we could have used the Schwartz-Zippel Lemma to prove Claim 5.2. $(AB - C)x$ is a degree-one polynomial in the variables $x_1, \ldots, x_n$. Choosing the same set S and applying the lemma, we see that if $AB \neq C$ then $(AB - C)x = 0$ with probability at most $\frac{1}{k+1}$.

5.3 Proof of Schwartz-Zippel

We now return to the general statement of the Schwartz-Zippel Lemma, and prove it.

We will induct on n, the number of variables in the polynomial.

Base Case: $n = 1$
p is a degree-d univariate polynomial, so by the Fundamental Theorem of Algebra, p has at most d roots. Thus even if all d roots are in S, the probability of p evaluating to 0 is at most $\frac{d}{|S|}$.

Inductive Case:
Let p be a degree-d polynomial in variables $x_1, x_2, \ldots, x_n$. Let k be the largest power of x_n that appears in any monomial of p. Then we can write

$$p = x_n^k \cdot q(x_1, \ldots, x_{n-1}) + r(x_1, \ldots, x_n),$$

where r includes terms in which x_n is raised to a power at most $k-1$. The polynomial q has degree-$(d-k)$ in $n-1$ variables, so by the inductive hypothesis, q evaluates to 0 with probability at most $\frac{d-k}{|S|}$.

To continue our analysis, we again use the principle of deferred decisions, by setting $x_1, x_2, \ldots, x_{n-1}$. Suppose $q(x_1, \ldots, x_{n-1}) \neq 0$. Then, since $x_1, \ldots, x_{n-1}$ have been fixed, p is a univariate degree k polynomial in x_n. Again by the Fundamental Theorem of Algebra, we can conclude that p evaluates to 0 with probability at most $\frac{k}{|S|}$. Therefore,

$$\mathbb{P}\left[p(x_1, \ldots, x_n) = 0 | q(x_1, \ldots, x_{n-1}) = 0, x_1, \ldots, x_{n-1}\right] \leq \frac{k}{|S|},$$

so

$$\mathbb{P}\left[p(x_1, \ldots, x_n) = 0 | q(x_1, \ldots, x_{n-1}) = 0\right] \leq \frac{k}{|S|}.$$

We can now calculate the probability that p evaluates to 0 by conditioning on whether q evaluates to 0:

$$\begin{aligned}\mathbb{P}[p(x_1,\ldots,x_n)=0] &= \mathbb{P}[p(x_i)=0 \mid q(x_i)=0]\cdot\mathbb{P}[q(x_i)=0]+\mathbb{P}[p(x_i)=0|q(x_i)\neq 0]\cdot\mathbb{P}[q(x_i)\neq 0]\\ &\leq 1\cdot\frac{d-k}{|S|}+\frac{k}{|S|}\cdot 1=\frac{d}{|S|}\end{aligned}$$

completing the proof.

5.4 Perfect Matchings

We now turn to our second application. A graph is called *bipartite* if its vertices can be divided into two sets X and Y (called partite sets) such that every edge has one endpoint in X and the other in Y. We denote a bipartite graph G by (X, Y, E). A *matching* is a subset of the edges of G such that no vertex has more than one incident edge. A *perfect matching* is a matching where every vertex has an incident edge in the matching. In other words, a perfect matching M is a subset of edges such that every vertex of G is adjacent to exactly one edge of M

Finding perfect matchings in graphs is a classical combinatorial problem. In bipartite graphs, the problem can be solved in $O\left(|E|\sqrt{|V|}\right)$ time [HK73]. In this section, we will describe a randomized algorithm for this problem. While the algorithm is not as fast as the classical Hopcroft-Karp algorithm on a single processor, our randomized algorithm will be highly parallelizable. In addition the algorithm is significantly simpler to implement.

Theorem 5.3. *There is a randomized $O(n^\omega)$ algorithm to test whether a bipartite graph has a perfect matching. Here, ω is the matrix multiplication constant, i.e., it is the smallest constant where one can multiply any two given matrices in time $O(n^\omega)$.*

Our algorithm is based off of the **Tutte matrix** of G. Let $X = \{x_1,\ldots,x_n\}$ and $Y = \{y_1,\ldots,y_n\}$. The Tutte matrix of G has rows indexed by elements of X and columns indexed by elements of Y. The (i, j) entry is 1 if there is an edge between x_i and y_j and 0 otherwise.

We will use a variation of this matrix, Z, where the (i, j) entry is a variable z_{ij} if there is an edge between x_i and y_j and 0 otherwise.

Our algorithm is based on the following claim:

Claim 5.4. $\det(Z)$ *is not the zero-polynomial if and only if G has a perfect matching.*

Note that we can test whether the determinant is 0 polynomial using the Schwartz-Zippel Lemma, because the determinant of Z is a degree n polynomial in $|E|$ variables. In particular, all we need to do is to assign values from $\{0, 1, 2, \ldots, n^2\}$ to $z_{i,j}$'s uniformly and independently, and evaluate the determinant of the corresponding matrix. Theorem 5.3 follows from the fact that for any matrix A, $\det(A)$ can be computed in time $O(n^\omega)$.

Proof. Let $G = (X, Y, E)$ be a bipartite graph, where $|X| = |Y| = n$. We use the Leibniz formula for the determinant of a matrix.

$$\det(Z)=\sum_{\sigma\in S_n}\operatorname{sign}(\sigma)\prod_{i=1}^{n}Z_{i,\sigma(i)}$$

Where S_n is the set of all permutations (i.e. one-to-one, onto functions $\sigma : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$). And sign is a function of permutations that outputs ± 1. The exact definition of sign(.) in our context is irrelevant.

Observe that possible perfect matchings are in one-to-one correspondence with permutations (because a perfect matching is a one-to-one, onto mapping from n vertices in X to n vertices in Y). By this formula, if a perfect matching actually exists, then its contribution to the sum is a non-zero monomial, moreover since monomials corresponding to different permutations are supported on different subsets of the variables, they do not cancel on another. Conversely if G has no perfect matching, every permutation has an entry of 0 in its product, so the whole determinant is 0. □

Up until now we have only looked at bipartite graphs. We provide a sketch of how to extend this algorithm to general graphs. We cannot use exactly the same matrix for general graphs as we did in the bipartite case (there is no natural way to divide the vertices into two sets). Instead we will let Z be a skew-symmetric version of the adjacency matrix.

$$Z_{i,j} = \begin{cases} z_{ij} & \text{if } (i,j) \text{ is an edge and } i < j \\ -z_{ji} & \text{if } (i,j) \text{ is an edge and } j > i \\ 0 & \text{ if } (i,j) \text{ is not an edge} \end{cases}.$$

It is a nice exercise to prove that for any graph $G = (V, E)$, $\det(Z) \neq 0$ if and only if G has a perfect matching.

5.5 Finding a Perfect Matching

At this point, we have only described how to determine if a given graph contains a perfect matching. In this part we want to discuss how find the matching itself. A natural first attempt would be to use *self-reducibility*, which is the idea that if we can find a subset of our desired object (here, a set of edges in the perfect matching), we can simplify our original instance to a smaller instance of the same problem.

An algorithm would run as follows:

1. Choose an edge (u, v) from G, and let H be the graph where we remove u and v (and their incident edges) from G.
2. Run our perfect matching decision algorithm on H.
3. If there is a perfect matching in H, recurse on H and find a perfect matching M in H; then, a perfect matching in G is just $M \cup (u, v)$.
4. Otherwise there is no perfect matching in H, it means that the edge (u, v) does not contribute to any perfect matching in G. Permanently remove (u, v) from G (leaving the vertices themselves) and recurse.

In each step we remove one edge, so this new algorithm takes $O(m \cdot n^{\omega})$ time. We remark that there is a faster (sequential) algorithm due to Harvey which finds a perfect matching in time $O(n^{\omega})$ for any general graph [Har09]. Unfortunately, this algorithm is ver sequential.

We now describe a parallel algorithm for this problem, which is much more efficient that even classical combinatorial algorithms if we have a sufficient number of processors.

Theorem 5.5 ([MVV87])**.** *There is a randomized parallel algorithm that finds a perfect matching in $O(\log^2 n)$ time using $O(n^{3.5}m)$ processors.*

Our algorithm relies on the well-known *Isolation Lemma.*

Theorem 5.6 (The Isolation Lemma)**.** *Fix a set S, and let $T_1, T_2, \dots T_k \subseteq 2^{[n]}$. For each $i \in [n]$ independently assign a uniformly random weight from S. Let $w(T_i) = \sum\limits_{x \in T_i} w_x$. Then we have*

$$\mathbb{P}[\textit{there is a unique } T_i \textit{ of minimum weight}] \geq 1 - \frac{n}{|S|}$$

Proof. Let $\mathcal{E}_i$ be the event that $\min\{w(T_j) : i \notin T_j\} = \min\{w(T_j) : i \in T_j\}$. Our proof follows from the following three claims.

Claim 1: If none of the $\mathcal{E}_i$ occur, then the minimum weight set is unique. We show the contrapositive – suppose that X and Y are distinct minimum-weight sets, then there is some element i such that $i \in X$ but $i \notin Y$, then since X and Y have minimum weight, the event $\mathcal{E}_i$ occurs.

Claim 2: $\mathbb{P}\left[\bigcap\limits_i \neg\mathcal{E}_i\right] = 1 - \mathbb{P}\left[\bigcup\limits_i \mathcal{E}_i\right] \geq 1 - \sum\limits_i \mathbb{P}[\mathcal{E}_i]$. just follows the union bound.

Claim 3: $\mathbb{P}[\mathcal{E}_i] \leq \frac{1}{|S|}$. We use the principle of deferred decision. Fix every w_j except w_i. Let $a = \min\{w(T_j \backslash i) : i \notin T_j\}$ and $b = \min\{w(T_j \backslash i) : i \in T_j\}$. Note that a is the left hand side of the formula for the event $\mathcal{E}_i$ and b is equal to the right hand side of the formula without the contribution of w_i. Thus when we fix w_i, $\mathcal{E}_i$ occurs if and only if $a = b + w_i$. Thus there is at most one possible element of S which we can assign to w_i to cause $\mathcal{E}_i$ to occur, so $\mathcal{E}_i$ occurs with probability at most $\frac{1}{|S|}$.

By Claim 1, there is a unique minimum minimum set if and only if none of the $\mathcal{E}_i$ occur. By Claims 2 and 3 and the union bound this occurs with probability at least $1 - \frac{n}{|S|}$. □

To apply this lemma, our matchings will be the T_i's. Consider the following assignment of weights to edges. For each edge (i, j) assign the weight $2^{w_{i,j}}$. Let Z be a matrix where for each adjacent x_i, y_j, $Z_{i,j} = 2^{w_{i,j}}$ and it is zero otherwise. Consider the maximum w such that 2^w divides $\det(Z)$. Recall that $\det(Z)$ is a sum over all perfect matchings of G, $\det(Z) = \sum_M \pm \prod_{(i,j) \in M} 2^{w_{i,j}}$. Thus if M is the unique minimum weight matching, then every contribution to the determinant from other matchings is a larger power of 2, and 2^w is equal to $\prod_{(i,j) \in M} 2^{w_{ij}}$

Our algorithm utilizes this fact as follows: for each edge (i, j), in parallel, we will calculate $\det(Z'_{i,j})$ where $Z'_{i,j}$ is Z with the i^{th} row and j^{th} column deleted. Then we check if $\frac{\det(Z'_{i,j}) \cdot 2^{w_{i,j}}}{2^w}$ is an odd integer. Note that $\det(Z'_{i,j})$ corresponds to matchings of G with vertices i and j deleted, thus if the minimum weight perfect matching is unique, this expression is odd if and only if the smallest perfect matching in $G \backslash \{i, j\}$ with the addition of the edge (i, j) has the same weight as the minimum weight perfect matching in G, which happens if and only if (i, j) is an edge in the minimum matching of G.

By the Isolation Lemma, and a large enough set S, we can guarantee that the minimum weight perfect matching is unique with high probability, which guarantees our algorithm will find a perfect matching in G.

The determinant of an $n \times n$ matrix can be computed with $O(n^{3.5})$ processors in $O(\log^2 n)$ time [GP85], which completes the proof of Theorem 5.5.

References

[GP85] Z. Galil and V. Pan. “Improved processor bounds for algebraic and combinatorial problems in RNC”. In: *FOCS*. IEEE. 1985, pp. 490–495 (cit. on p. 5-5).

[Har09] N. J. Harvey. “Algebraic algorithms for matching and matroid problems”. In: *SIAM Journal on Computing* 39.2 (2009), pp. 679–702 (cit. on p. 5-4).

[HK73] J. E. Hopcroft and R. M. Karp. “An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs”. In: *SIAM Journal on computing* 2.4 (1973), pp. 225–231 (cit. on p. 5-3).

[MVV87] K. Mulmuley, U. V. Vazirani, and V. V. Vazirani. “Matching is as easy as matrix inversion”. In: *STOC*. ACM. 1987, pp. 345–354 (cit. on p. 5-4).