CAS CS 535: Complexity Theory

Lecturer: Mark Bun — Fall 2023

Lecture Notes 18:

Proof of Valiant-Vazirani, Counting

Reading.

- Arora-Barak § 17.1-17.3.1, 17.4.1

Last time: BPP $\subseteq$ PH, Promise Problems, Randomized Reductions

Recall the Unambiguous Satisfiability problem:

$$\begin{aligned}\mathsf{USAT}_Y &= \{\varphi \mid \text{CNF } \varphi \text{ has exactly one satisfying assignment}\}\\ \mathsf{USAT}_N &= \{\varphi \mid \text{CNF } \varphi \text{ is unsatisfiable}\}.\end{aligned}$$

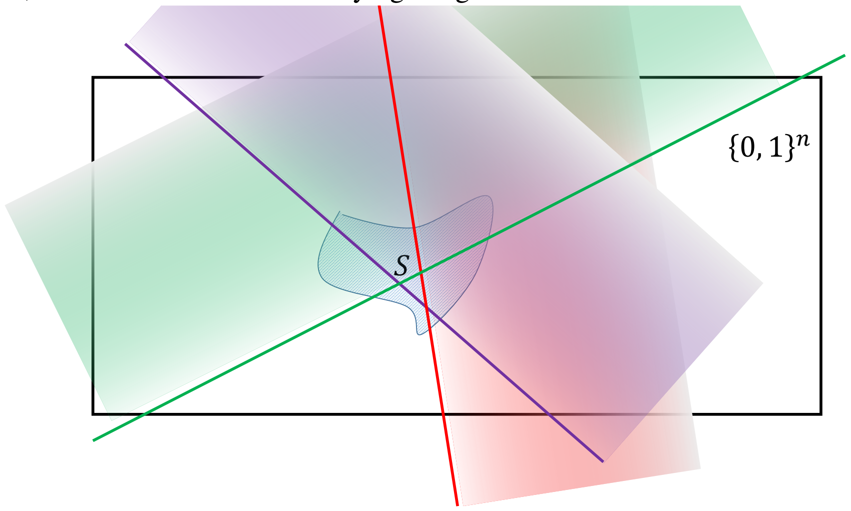
Theorem 1 (Valiant-Vazirani)**.** *There exists a poly-time randomized algorithm A mapping n-variable CNF formulas to n-variable CNF formulas such that*

$$\begin{aligned}\varphi \in \mathsf{SAT} &\implies \Pr[A(\varphi) \in \mathsf{USAT}_Y] \geq \frac{1}{8n}\\ \varphi \notin \mathsf{SAT} &\implies \Pr[A(\varphi) \in \mathsf{USAT}_N] = 1.\end{aligned}$$

1 Proof of Valiant-Vazirani

Proof idea. The reduction A will add randomly chosen additional constraints to φ to eliminate all but 1 satisfying assignment (if one exists).

In pictures. Suppose we have some additional information about our input formula φ: Either it's unsatisfiable, or that the number of satisfying assignments is between 2^{k-1} and 2^k.



Let S be the set of satisfying assignments to φ.

We'll add new constraints $C_1, \dots, C_k$ which each (almost) independently eliminate a random half of $\{0,1\}^n$ as possible satisfying assignments. The expected size of the portion of S that survives is $2^{-k} \cdot |S| \in [1/2, 1]$. We'll show that $|S| = 1$ with constant probability.

One remaining issue: How do we know how to set k? We don't know! We'll just guess it at random from $1, \dots, n$, which will be correct with probability $1/n$.

1.1 Some Technical Tools

The details of the proof actually go somewhat differently than the sketch outlined above, in that instead of adding constraints one at a time, we'll "do it all in one go" and sample a single random constraint that eliminates all but a 2^{-k} fraction of satisfying assignments.

The first idea we might try to do this is to let $h : \{0,1\}^n \to \{0,1\}^k$ be a uniformly random "hash" function. Then the constraint "$h(x) = 0^k$" is indeed satisfied by a 2^{-k} fraction of points in S. The problem is that a truly random function is very complex to describe and evaluate; the truth table takes $k \cdot 2^n$ bits to write down, and Shannon's counting argument shows that almost all such functions cannot be described by circuits substantially smaller than this.

The solution is to not choose a h truly uniformly at random, but in a manner that is "just random enough." Specifically, we'll draw h from a *pairwise independent* family of hash functions.

Definition 2. Let $k \le n$. A set of functions $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^k\}$ is <u>pairwise independent</u> if for all $x \neq x' \in \{0,1\}^n$ and all $y, y' \in \{0,1\}^k$, we have

$$\Pr_{h \leftarrow \mathcal{H}}[h(x) = y \wedge h(x') = y'] = \Pr_{h \leftarrow \mathcal{H}}[h(x) = y] \cdot \Pr_{h \leftarrow \mathcal{H}}[h(x') = y'] = 2^{-k} \cdot 2^{-k} = 2^{-2k}.$$

That is, a function h drawn from a pairwise independent family "looks random" as long as you restrict your attention to a single pair of inputs.

The upshot of pairwise independence is that we have constructions of such families that are easy to sample from and easy to compute.

Example 3. Let $k \le n$ and define

$$\mathcal{H} = \{h_{A,b} \mid A \in \{0,1\}^{n \times k}, b \in \{0,1\}^k\}$$

where $h_{A,b} : \{0,1\}^n \to \{0,1\}^k$ is defined by $h_{A,b}(x) = Ax + b$, where the arithmetic is done mod 2. This is a pairwise independent hash family. (Proof left as an exercise.)

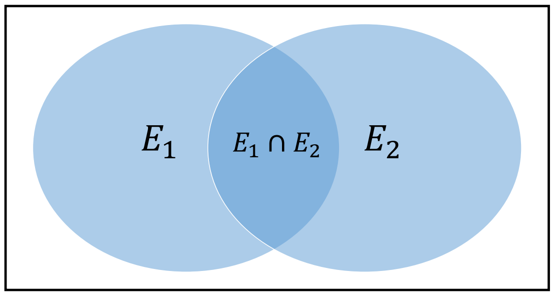
Observe that sampling $h \leftarrow \mathcal{H}$ takes time $\mathrm{poly}(n)$, and each $h \in \mathcal{H}$ can be described by a $\mathrm{poly}(n)$-size circuit (in fact, with each output bit describable by a CNF), and hence can be evaluated in polynomial time.

The other tool we need is the Inclusion-Exclusion Principle. To motivate this, let's first recall

Union Bound: $\Pr\left[\bigcup_{i=1}^n E_i\right] \le \sum_{i=1}^n \Pr[E_i]$

Inclusion-Exclusion: $\Pr\left[\bigcup_{i=1}^n E_i\right] \ge \sum_{i=1}^n \Pr[E_i] - \sum_{1 \le i < j \le n} \Pr[E_i \cap E_j]$.

Proof by picture: In the case of 2 sets, we actually have equality: $\Pr[E_1 \cup E_2] = \Pr[E_1] + \Pr[E_2] - \Pr[E_1 \cap E_2]$.



The general case follows by induction, with the ordinary union bound facilitating the inductive step.

1.2 Back to Valiant-Vazirani

The proof relies on the following key lemma.

Lemma 4 (Isolation Lemma)**.** *If $\mathcal{H}$ is a pairwise independent family of functions of the form $h : \{0,1\}^n \to \{0,1\}^k$, and $S \subseteq \{0,1\}^n$ satisfies $2^{k-2} \le |S| \le 2^{k-1}$, then*

$$\Pr_{h \leftarrow \mathcal{H}}[\textit{There is a unique } x \in S \textit{ s.t. } h(x) = 0^k] \ge \frac{1}{8}.$$

Proof. We write the probability we're interested in as

$$\sum_{x\in S} \Pr[h(x) = 0^k \wedge (\forall x' \in S \setminus \{x\}, h(x') \neq 0^k)]$$

$$\begin{aligned}
&= \sum_{x\in S} \Pr[h(x) = 0^k] - \Pr[h(x) = 0^k \wedge (\exists x' \in S \setminus \{x\}, h(x') = 0^k)] \\
&\ge \sum_{x\in S} \left(2^{-k} - \sum_{x'\in S\setminus\{x\}} \Pr[h(x) = 0^k \wedge h(x') = 0^k]\right) && \text{(union bound)} \\
&\ge \sum_{x\in S} \left(2^{-k} - \sum_{x'\in S\setminus\{x\}} 2^{-2k}\right) && \text{(pairwise independence)} \\
&= |S| \left(2^{-k} - (|S| - 1)2^{-2k}\right) \\
&\ge 2^{k-2}(2^{-k} - 2^{k-1}2^{-2k}) \\
&= 2^{k-2}2^{-k-1} = 2^{-3} = 1/8.
\end{aligned}$$

$\square$

Proof of Valiant-Vazirani. Let $\mathcal{H}_{n,k}$ denote an efficiently sampleable/computable pairwise independent family of functions $h : \{0,1\}^n \to \{0,1\}^k$. The following randomized algorithm A reduces SAT to USAT.

On input φ:

1. Sample $k \in \{2, \ldots, n+1\}$ uniformly at random

2. Sample $h \leftarrow \mathcal{H}_{n,k}$
3. Output a CNF $\psi(x)$ corresponding to $\varphi(x) \wedge (h(x) = 0^k)$.

By construction, this reduction runs in randomized poly-time. For correctness, we check:

- If $\varphi \notin \mathsf{SAT}$, then ψ is definitely unsatisfiable, so $\psi \in \mathsf{USAT}_N$ with probability 1.
- If $\varphi \in \mathsf{SAT}$, then $\Pr[\psi \in \mathsf{USAT}_Y] \geq \Pr[2^{k-2} \leq |\psi^{-1}(1)| \leq 2^{k-1}] \cdot \frac{1}{8} = \frac{1}{8n}$.

□

2 Counting

We're going to switch gears and talk about a new class of computational problems: Counting problems. Here's a motivating example.

Problem 1. Given a (simple, undirected) graph G, how many spanning trees does G have?

Some basic observations:

- Finding one spanning tree is easy (BFS or DFS)
- Enumerating all spanning trees seems hard. How might you do it?

  Brute force check all subsets of $n-1$ edges? There are $\binom{n(n-1)}{n-1} \approx n^n$ such subsets to check, so naïvely this would take exponential time. Note that the complete graph K_n has n^{n-2} spanning trees, so counting by...actually counting...seems like a bad idea.

For this problem, there turns out to be a beautiful way to count without having to count:

Theorem 5 (Matrix Tree Theorem [Kirchoff)**.** *] Let A_G be the adjacency matrix of G, and for each $i \in [n]$, let d_i be the degree of vertex i. Then the number of spanning trees of G is $\det(\widetilde{L_G})$ where L_G is the "graph Laplacian"*

$$L_G = \begin{pmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & \dots & 0 \\ \vdots & & & \\ 0 & 0 & \dots & d_n \end{pmatrix} - A_G.$$

and $\widetilde{L_G}$ is any matrix obtained by deleting one row and column from L_G.

The determinant can be computed in polynomial time (in fact, in $\mathbf{NC}$), so counting can be done efficiently!

Here's another example.

Problem 2. Given a directed graph G, how many simple cycles does G have?

Finding one cycle is still easy, but as we'll see shortly, counting cycles seems to be much harder than harder than counting spanning trees.

To formalize this, let us first introduce notation for function evaluation problems.

Definition 6. The complexity class $\mathbf{FP}$ consists of all functions $f : \{0,1\}^* \to \mathbb{N}$ such that f can be computed in deterministic polynomial time.

Theorem 7. *If* $\#\mathsf{CYCLE} \in \mathbf{FP}$, *then* $\mathbf{P} = \mathbf{NP}$.

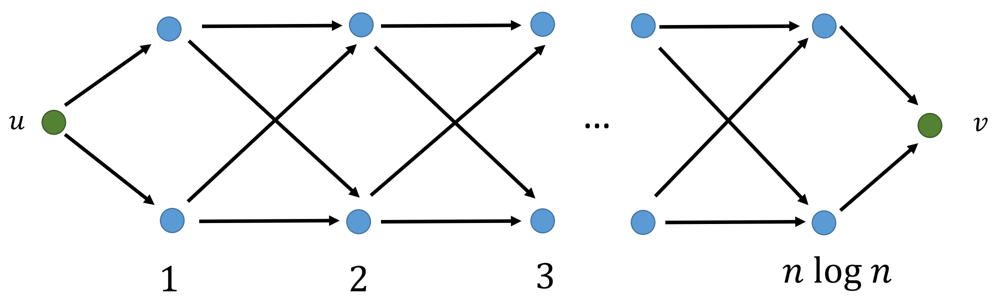
Proof. We will show that if $\#\mathsf{CYCLE} \in \mathbf{FP}$, then the $\mathbf{NP}$-complete problem $\mathsf{HamCycle} \in \mathbf{P}$.

(A Hamiltonian cycle in a digraph is a cycle that visits every vertex exactly once.)

Let G be a digraph (an instance of $\mathsf{HamCycle}$) with n vertices. We'll give a poly-time computable reduction mapping $G \mapsto G'$ such that

$$G \in \mathsf{HamCycle} \implies G' \text{ has } \geq n^{n^2} \text{cycles}$$
$$G \in \mathsf{HamCycle} \implies G' \text{ has } < n^{n^2} \text{cycles}$$

Here is the reduction. We'll construct G' by replacing every edge $u \to v$ in G with the following gadget.



Note: There are thus $2^{n \log n} = n^n$ directed paths from u to v in the new graph G'.

Now let us analyze correctness of the construction.

- $G \in \mathsf{HamCycle} \implies G$ has at least one simple cycle of length n
  $\implies G'$ has at least $(n^n)^n = n^{n^2}$ simple cycles.
- $G \in \mathsf{HamCycle} \implies$ Every cycle in G has length $\leq n-1$
  $\implies G'$ has at most $n^{n-1} \cdot (n^n)^{n-1} = n^{n^2-1} < n^{n^2}$ simple cycles.

$\square$

One more example:

Problem 3. Define the counting problem $\#\mathsf{SAT}$ as follows. Given a Boolean formula φ, how many satisfying assignments does φ have?

This problem is definitely at least has hard as SAT, so unlike $\#\mathsf{CYCLE}$, finding a single satisfying assignment already seems hard. Can we understand how much harder the counting version of the problem is?

3 Class $\#\mathbf{P}$

Definition 8. A function $f : \{0,1\}^* \to \mathbb{N}$ is in $\#\mathbf{P}$ (pronounced "sharp-P") if there exists a polynomial-time deterministic TM M and a polynomial $p : \mathbb{N} \to \mathbb{N}$ such that

$$f(x) = \#\{u \in \{0,1\}^{p(|x|)} \mid M(x,u) = 1\}.$$

That is, for every x, the function $f(x)$ counts the number of witnesses u such that $M(x,u)$ accepts.

Equivalently: A function $f \in \#\mathbf{P}$ if there exists a <u>nondeterministic</u> TM N such that $f(x)$ counts the number of branches on which N accepts on x, i.e., the number of paths from start to accept in the configuration graph $G_{N,x}$.

Examples of problems in $\#\mathbf{P}$:

- $\#$ spanning trees
- $\#$CYCLE
- $\#$SAT
- $\#$CKTSAT: Given a circuit C, how many satisfying assignments does it have?
- $\#$INDSET

Lots of problems in statistical physics, AI, etc. involve <u>sampling</u> from an implicitly defined probability distribution, i.e, $\mu(x) = \frac{p(x)}{\sum_y p(y)}$ where $p(x)$ is efficiently computable, but the normalization constant (a.k.a., the partition function) $\sum_y p(y) \in \#\mathbf{P}$, and is often hard to compute.

3.1 Reductions Between Function/Counting Problems

Definition 9. A *parsimonius reduction* R from f to g is a poly-time computable function such that for all $x \in \{0,1\}^*$, we have

$$f(x) = g(R(x)).$$

Interpretation: If computing g is easy, then computing f is also easy.

Example 10. $\#$CKTSAT parsimoniously reduces to $\#$SAT. The reason is that the reduction from the decision version CKTSAT to SAT preserves the number of satisfying assignments.

Example 11. The Cook-Levin Theorem is parsimonious, so there is a parsimonious reduction from every function $f \in \#\mathbf{P}$ to $\#$SAT.

Theorem 12. $\#$SAT *is $\#\mathbf{P}$-complete under parsimonious reductions.*