CS 221 Computational Complexity Febraury 22, 2018

Lecture 10

Instructor: Madhu Sudan *Scribe: Daniel Alabi*

1 Goals for Today & Administrativia & Exercises

Today we will talk about

1. Counting problems (eg. #SAT)
2. The class #P
3. The proof sketch for why the Permanent is #P-complete

PSET #3 is out. It's due 8pm next Friday.
Exercises (food for thought) proposed in class:

1. Reduce USAT to SAT. In class, we showed an RP reduction from SAT to USAT.
2. Show that if NP reduces to BPP, then PH collapses.
3. Propostions 2, 3, and 4.

2 Some Motivation for Counting Problems

First, we provide some motivation for why we consider counting problems. In cryptography, there is a notion of one-way permutations for which is it easy to compute $f(x)$ for some function but hard to invert the function. Formally, if we have $f : \{0,1\}^n \to \{0,1\}^n$, then it should be easy to compute $f(x)$ for any $x \in \{0,1\}^n$ but it should be hard (in fact, NP-hard) to find an $x \in \{0,1\}^n$ such that $f(x) = y$ for some given $y \in \{0,1\}^n$. Can we make sure that $f(x) = y$ (given y) is uniquely satisfiable by some x?

We would like to reduce SAT to "uniquely satisfiable instances". Which is harder?

3 Unique SAT (USAT)

First, let's start with a wrong definition of $USAT$:

$$USAT = \{\phi | \phi \text{ has a unique SAT assignment}\}$$

which is harder what we need. Instead, we could set it up as a promise problem: $USAT = (USAT_Y, USAT_N)$

$$USAT_Y = \{\phi | \phi \text{ has a unique SAT assignment}\}$$

and

$$USAT_N = \{\phi | \phi \text{ is not satisfiable}\}$$

It is possible to show that $USAT$ reduces to SAT but is the converse true: $SAT \leq USAT$?

Let $\leq_k$ denote a Karp reduction. It's open if $SAT \leq_k USAT$. The belief is that if $SAT \leq_k USAT$ then the polynomial hierarchy collapses.

First, we define the BP-reduction (and a RP-reduction):

Definition 1. *$A = (A_Y, A_N) \leq_{BP} B = (B_Y, B_N)$ if there exists a function (that runs in probabilistic poly-time) such that*

$$x \in A_Y \Rightarrow \boldsymbol{Pr}[f(x) \in B_Y] \geq c$$

and

$$x \in A_N \Rightarrow \boldsymbol{Pr}[f(x) \notin B_N] \leq s$$

If $c - s \geq \frac{1}{\mathrm{poly}(n)}$, then this is a BP-reduction. If $s = 0$, then this is an RP-reduction.

Proposition 2. *If $A \leq_{BP} B$ and $B \in P$ (or $B \in BPP$) then $A \in BPP$*

Proposition 3. *If $A \leq_{RP} B$ and $B \in RP$ then $A \in RP$*

Proposition 4. *If $A \leq_{RP} B$ and $B \in co - RP$ then $A \in BPP$*

Theorem 5 ([1]Valiant-Vazirani 1986). *$SAT \leq_{RP} USAT = (USAT_Y, USAT_N)$*

We will construct a function f such that

$$\phi \in SAT \Rightarrow f(\phi) \in USAT_Y$$

with probability $\geq \frac{1}{8n}$ [1]

$$\phi \notin SAT \Rightarrow f(\phi) \in USAT_N$$

with probability 1

Proof. First, we define the hash family $\mathcal{H}_{n,k}$:

Definition 6. *$\mathcal{H}_{n,k} \subseteq \{h | h : \{0,1\}^n \to \{0,1\}^k\}$ is pairwise-independent (pwi) if $\forall x \neq x' \in \{0,1\}^n$ and $a, b \in \{0,1\}^k$ we have*

$$\boldsymbol{Pr}_{h \in \mathcal{H}_{n,k}}[h(x) = a \wedge h(x') = b] = \frac{1}{4^k}$$

Proposition 7. *Let $m \in \mathbb{F}_2^{n \times k}$ and $h_m(x) = m \times x + b$ where $b \in \mathbb{F}_2^k$, then $\mathcal{H}_{n,k} = \{h_{m,b} | m, b\}$ is a pwi family.*

Definition 8. *For some $h \in \mathcal{H}_{n,k}$, $\varphi_h(x) = 1$ iff $h(x) = \overline{0}$ (=0 when I forget to put a bar above)*

Given an instance ϕ for SAT, we will use the following (poly-time) algorithm for the reduction:

1. Randomly (and uniformly) pick some $k \in \{2, \dots, n+1\}$
2. Randomly pick a hash function from the hash family $\mathcal{H}_{n,k}$ (defined above).
3. Output $\phi \wedge \varphi_h$

First, let us assume that we know the number of satisfying assignments in ϕ (let's denote this $\#\phi$) and that $\#\phi \in [2^{k-2}, 2^{k-1}]$. Also, let $\phi^{-1}(1) = \{x | \phi(x) = 1\}$.

Lemma 9. *For all $\phi^{-1}(1) \subseteq \{0,1\}^n$,*

$$\boldsymbol{Pr}_{h \in \mathcal{H}}[\exists x \in \phi^{-1}(1) \ s.t.\ h(x) = 0 \wedge \forall y \in \phi^{-1}(1) \setminus \{x\}, h(y) \neq 0] \geq \frac{1}{8}$$

[1] Note that in class, we attempted to show the probability $\geq \frac{1}{4(n+1)}$ but there were some minor mistakes in the proof

Proof. For any fixed $x \in \phi^{-1}(1)$, we have

$$\mathbf{Pr}_{h\in\mathcal{H}}[h(x)=0] = \frac{1}{2^k}$$

For any fixed $y \neq x$,

$$\mathbf{Pr}_{h\in\mathcal{H}}[h(x)=0 \wedge h(y)=0] = \frac{1}{4^k}$$

Then

$$\begin{aligned}
\mathbf{Pr}_{h\in\mathcal{H}}[h(x)=0 \wedge \forall y \in \phi^{-1}(1)\setminus\{x\}, h(y)\neq 0] &\geq \mathbf{Pr}_{h\in\mathcal{H}}[h(x)=0] - \sum_{y\in\phi^{-1}\setminus\{x\}} \mathbf{Pr}_{h\in\mathcal{H}}[h(x)=h(y)=0] \tag{1}\\
&\geq \frac{1}{2^k} - \frac{\#\phi}{4^k} \tag{2}\\
&\geq \frac{1}{2^{k+1}} \tag{3}
\end{aligned}$$

where we used that $\#\phi \leq 2^{k-1}$ As a result,

$$\begin{aligned}
\mathbf{Pr}_{h\in\mathcal{H}}[\exists x \in \phi^{-1}(1) \text{ s.t. } h(x)=0 \wedge \forall y \in \phi^{-1}(1)\setminus\{x\}, h(y)\neq 0] &= \sum_{x\in\phi^{-1}(1)} \mathbf{Pr}_{h\in\mathcal{H}}[h(x)=0 \wedge \forall y \in \phi^{-1}(1)\setminus\{x\}, h(y)\neq 0] \tag{4}\\
&\geq \frac{\#\phi}{2^{k+1}} \geq \frac{1}{8} \tag{5}
\end{aligned}$$

where we have used that $\#\phi \geq 2^{k-2}$ □

But how do we pick k? We just guess k to be one of $\{2,\ldots,n+1\}$ so that with probability $1/n$, we would select the "right" k. As a result, with probability $\geq \frac{1}{8n}$, we will output formula $\phi \wedge \varphi_h$ with a unique assignment. This completes the proof of Theorem 5.

□

We can also show that $SAT \leq_{RP} \oplus - SAT$ and we can amplify the completeness probability (from $\frac{1}{8n}$). On the other hand, for $SAT \leq_{RP} USAT$, we do not know how to amplify the completeness probability.

4 #P

The #P class captures counting problems.

Definition 10. *We say that $f : \{0,1\}^k \to \mathbb{Z}^{\geq 0}$ is a #P-function if there exists some TM $M(\cdot,\cdot)$ running in poly-time such that $\forall x$, $f(x) = \#\{y \text{ s.t. } M(x,y)=1\}$.*

Essentially, we're counting the number of witnesses for any assignment $x \in \{0,1\}^k$.

Consider the following translations from "logic" notation to "arithmetic" notation:

1. $\exists \Rightarrow \sum$
2. $\forall \Rightarrow \prod$
3. $\exists x$ s.t. $\forall j$ clause $C_j(\cdot)$ is satisfiable $\Rightarrow \sum_x \prod_j (\mathbb{1}\,[C_j \text{ is satisfied by } x])$

Examples of #P-complete problems: #SAT, #CLIQUE, #VC (VC = Vertex Cover)
Other #P-complete problems:

1. **Network Reliability** Given a connected graph $G = (V, E)$ with each edge $e \in E$ has failure probability p_e. What is the probability that G remains connected? This involves some counting of the spanning subgraphs of G (which is #P-complete).
2. **Hamiltonian (From Statistical Mechanics)** Let a graph G represent a monomer-dimer system. Each pair of adjacent vertices can be represented by a dimer and all the other vertices can be represented by a monomer. The energy of a configuration M is proportional to $p^{\#dimers}$ where p is some parameter governed by the temperature of the system.
3. **Bayes Net Inference**

The permanent of an $n \times n$ matrix M is:

$$\mathrm{perm}(M) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} M_{i,\sigma(i)}$$

where S_n denotes the set of all permutations of n elements. The permanent can be intepreted combinatorially as follows: if M has entries in $\{0, 1\}$, then it can represent the adjacency matrix of a bipartite graph on $2n$ vertices with n vertices on each "side" of the graph. Let the two sets of vertices be A, B. Then for any two vertices $x_i \in A, y_j \in B$, there is an edge $\{x_i, y_j\}$ iff $M_{i,j} = 1$. $\prod_{i=1}^{n} M_{i,\sigma(i)}$ is 1 iff σ represents a perfect matching. As a result, if M is a $\{0, 1\}$ matrix, perm(M) is the number of perfect matchings in the corresponding graph.

Theorem 11 (Valiant). *Computing the Permanent of* $\{0, 1\}$ *matrices is #P-complete*

Proof Sketch The idea is to show that #VC reduces to computing the Permanent. VC is the subset of vertices that touch every edge. The first key step is to create gadgets for vertices and edges.

For any edges A, B, C, D chosen, we want that to correspond to exactly 1 matching in the gadget. If A, B or C, D is chosen, that should correspond to 1 matching also. But this is a contradiction as no such gadgets exist if edges have positive weight. Now if we use negative weight edges, the permanent is still well-defined but has no combinatorial interpretation. The second key step is to compute the permanent $\mod p$ so that we get all non-negative numbers. [2]

References

[1] Leslie G. Valiant, Vijay V. Vazirani. NP Is as Easy as Detecting Unique Solutions. *STOC*, 458-463 (1985)

[2] See Madhu's hand-written notes for some more details (and a representative diagram).