Lecture 10

In this lecture, we will prove the theorem concerning the connections between counting problems and the power of randomization.

1 Complexity of counting problems

We will prove the following theorem:

THEOREM 1
For every counting problem $\#A$ in $\#\mathbf{P}$, there is an algorithm C that on input x, compute with high probability a value v such that

$$(1-\epsilon)\#A(x) \leq v \leq (1+\epsilon)\#A(x)$$

in time polynomial in $|x|$ and in $\frac{1}{\epsilon}$, using an oracle for **NP**.

The theorem says that $\#\mathbf{P}$ can be approximate in $\mathbf{BPP}^{\mathbf{NP}}$. We have a remark here that approximating #3SAT is **NP**-hard. Therefore, to compute the value we need at least the power of **NP**, and this theorem states that the power of **NP** and randomization is sufficient.

Another remark is concerning on Toda's theorem which states that $\mathbf{PH} \subseteq \mathbf{P}^{\#\mathbf{P}}$. This implies that #3SAT is Σ_k-hard for every k, i.e., #3SAT lies outside **PH**. Recall that **BPP** lies inside Σ_3, and hence approximating #3SAT can be done in Σ_3. Therefore, approximating #3SAT cannot be equivalent to computing #3SAT exactly, unless the polynomial hierarchy collapses.

We first make some observations so that we can reduce the proof to an easier one.

- It is enough to prove the theorem for #3SAT.

  If we have an approximation algorithm for #3SAT, we can extend it to any #A in $\#\mathbf{P}$ using the parsimonious reduction from #A to #3SAT.

- It is enough to give a polynomial time $O(1)$-approximation for #3SAT.

  Suppose we have an algorithm C and a constant c such that

  $$\frac{1}{c}\#3\mathrm{SAT}(\varphi) \leq C(\varphi) \leq c\#3\mathrm{SAT}(\varphi).$$

  Given φ, we can construct $\varphi^k = \varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_k$. If φ has t satisfying assignments, φ^k has t^k satisfying assignments. Then, giving φ^k to the algorithm we get

  $$\frac{1}{c}t^k \leq C(\varphi^k) \leq ct^k$$
  $$\left(\frac{1}{c}\right)^{1/k} t \leq C(\varphi^k)^{1/k} \leq c^{1/k}t.$$

  If c is a constant and $k = O(\frac{1}{\epsilon})$, $c^{1/k} = 1+\epsilon$.

- For a formula φ that has $O(1)$ satisfying assignments, $\#3\mathrm{SAT}(\varphi)$ can be found in $\mathbf{P}^{\mathbf{NP}}$.

  This can be done by iteratively asking the oracle the questions of the form: "Are there k assignments satisfying this formula?" Notice that these are **NP** questions, because the algorithm can guess these k assignments and check them.

2 An approximate comparison procedure

Consider the following approximate comparison procedure `a-comp` defined as:

$$\texttt{a-comp}(\varphi, t) = \begin{cases} \text{YES} & \text{if } \#3\mathrm{SAT}(\varphi) \geq 2t \\ \text{NO} & \text{if } \#3\mathrm{SAT}(\varphi) \leq t/2 \\ \text{whatever} & \text{otherwise} \end{cases}$$

Given `a-comp`, we can construct an algorithm that 8-approximates #3SAT as follows:

Input: φ

compute:
  `a-comp`$(\varphi, 1)$
  `a-comp`$(\varphi, 2)$
  `a-comp`$(\varphi, 4)$
  $\vdots$
  `a-comp`$(\varphi, 2^n)$

if `a-comp` outputs NO from the first time **then**
  // The value is either 0 or 1.
  // The answer can be checked by one more query to the **NP** oracle.
  Query to the oracle and output an exact value.
else
  Suppose that it outputs YES for $t = 1, \ldots, 2^i$ and NO for $t = 2^{i+1}$
  Output $t = 2^{i+1}$

We need to show that this algorithm approximates #3SAT with in a factor of 8. If `a-comp` answers NO from the first time, the algorithm outputs the right answer because it checks for the answer explicitly. Now suppose `a-comp` says YES for all $t = 1, 2, \ldots, 2^i$ and says NO for $t = 2^{i+1}$. Since `a-comp`$(\varphi, 2^i)$ outputs YES, $\#3\mathrm{SAT}(\varphi) > \frac{1}{2} \cdot 2^i$, and also since `a-comp`$(\varphi, 2^{i+1})$ outputs NO, $\#3\mathrm{SAT}(\varphi) < 2 \cdot 2^{i+1}$. The algorithm outputs $t = 2^{i+1}$. Hence,

$$\frac{1}{4} \cdot t < \#3\mathrm{SAT}(\varphi) < 2 \cdot t,$$

or

$$2 \cdot \#3\mathrm{SAT}(\varphi) < t < 4 \cdot \#3\mathrm{SAT}(\varphi),$$

and the algorithm outputs the correct answer with in a factor of 8.

Thus, to establish the theorem, it is enough to give a $\mathbf{BPP}^{\mathbf{NP}}$ implementation of the `a-comp`.

3 Constructing `a-comp`

In order to construct the procedure, we will use the property of a special kind of families of hash functions, namely, the families of pairwise independent hash function.

DEFINITION 1 *Let H be a family of functions $h \in H, h : \{0,1\}^n \to \{0,1\}^m$. H is a collection of pairwise independent hash functions if, for every $a, b \in \{0,1\}^n$, $a \neq b$, and for every v_1, v_2 in $\{0,1\}^m$,*

$$\mathbf{Pr}_{h \in H}[h(a) = v_1 \wedge h(b) = v_2] = \frac{1}{2^{2m}},$$

or, equivalently, for a random $h \in H$, $h(a)$ and $h(b)$ are independently and uniformly distributed in $\{0,1\}^m$.

A family of pairwise independent hash functions exists. Furthermore, if we take H to be the set of all functions $h_A : \{0,1\}^n \to \{0,1\}^m$ where $h_A(x) = Ax^T$ and $A \in \{0,1\}^{n \times m}$, then H is a family of pairwise independent hash functions. Notice that it is easy to pick a hash function from this family and it is easy to compute the function.

We also use the Leftover Hash Lemma proved by Impagliazzo, Levin, and Luby in 1989:

THEOREM 2
Let H be a family of pairwise independent hash functions $h : \{0,1\}^n \to \{0,1\}^m$. Let $S \subset 0, 1^n$, $|S| \geq \frac{4 \cdot 2^m}{\epsilon^2}$. Then,

$$\mathbf{Pr}_{h \in H}\left[\left| |\{a \in S : h(a) = 0\}| - \frac{|S|}{2^m} \right| \geq \epsilon \frac{|S|}{2^m} \right] \leq \frac{1}{4}.$$

In practice, we use hash function in a data structure for representing a subset $S \subseteq \{1,0\}^n$ by mapping each member of S to a slot on the smaller set $\{0,1\}^m$ and maintaining, for each slot, a linked list of items mapped to it. For this scheme to work efficiently, a small number of elements from S should be mapped to each slot. This theorem says that the number is about $\frac{|S|}{2^m}$. From this, `a-comp` can be constructed as follows.

input: φ, t

if $t \leq 32$ **then** check exactly whether $\#3\mathrm{SAT}(\varphi) \geq t$.
if $t \geq 32$,
pick h from a set of pairwise independent hash functions $h : \{0,1\}^n \to \{0,1\}^m$,
where m is such that $2^{m+1} \geq \frac{t}{16} \geq 2^m$.
answer YES iff there are more then $\frac{t}{2^m}$ assignments a to φ such that
a satisfies φ and $h(a) = 0$.

Notice that since $\frac{t}{2^m} \leq 32$, the test at the last step can be done with one access to an oracle to **NP**. We will show that the algorithm is in $\mathbf{BPP}^{\mathbf{NP}}$. Let $S \subseteq \{0,1\}^n$ be the set of satisfying assignments for φ. There are 2 cases.

- If $|S| \geq 2t$, by the Leftover Hash Lemma we have:

$$\mathbf{Pr}_{h\in H}\left[\left||\{a \in S : h(a) = 0\}| - \frac{|S|}{2^m}\right| \geq \frac{1}{2}\cdot\frac{|S|}{2^m}\right] \leq \frac{1}{4},$$

(set $\epsilon = \frac{1}{2}$, and $|S| \geq \frac{4\cdot 2^m}{\epsilon^2} = 16\cdot 2^m$, because $|S| > t \geq 16\cdot 2^m$)

$$\mathbf{Pr}_{h\in H}\left[|\{a \in S : h(a) = 0\}| \leq \frac{1}{2}\cdot\frac{|S|}{2^m}\right] \leq \frac{1}{4},$$
$$\mathbf{Pr}_{h\in H}\left[|\{a \in S : h(a) = 0\}| \leq \frac{t}{2^m}\right] \leq \frac{1}{4},$$

which is the failure probability of the algorithm. Hence, the algorithm succeeds with probability at least $3/4$.

- If $|S| \leq t/2$:

Let S' be a superset of S of size $t/2$. We have

$$\begin{aligned}\mathbf{Pr}_{h\in H}[\text{answer YES}] &= \mathbf{Pr}_{h\in H}[|\{a \in S : h(s) = 0\}| > \tfrac{t}{2^m}] \\ &\leq \mathbf{Pr}_{h\in H}[\left|\{a \in S' : h(s) = 0\}\right| > \tfrac{t}{2^m}] \\ &\leq \mathbf{Pr}_{h\in H}\left[\left|\left|\{a \in S' : h(s) = 0\}\right| - \tfrac{|S'|}{2^m}\right| > \tfrac{|S'|}{2^m}\right] \\ &\leq \frac{1}{4}\end{aligned}$$

(by L.H.L. with $\epsilon = 1, |S'| \geq 4\cdot 2^m$, and $t \geq 8\cdot 2^m$.)

Therefore, the algorithm will give the correct answer with probability at least $3/4$.

4 The proof of the Leftover Hash Lemma

We finish the lecture by proving the Leftover Hash Lemma.

PROOF: We will use Chebyshev's Inequality to bound the failure probability. Let $S = \{a_1, \ldots, a_k\}$, and pick a random $h \in H$. We define random variables $X_1, \ldots, X_k$ as

$$X_i = \begin{cases} 1 & \text{if } h(a_i) = 0 \\ 0 & \text{otherwise.}\end{cases}$$

Clearly, $|\{a \in S : h(a) = 0\}| = \sum_i X_i$.

We now calculate the expectations. For each i, $\mathbf{Pr}[X_i = 1] = \frac{1}{2^m}$ and $\mathbf{E}[X_i] = \frac{1}{2^m}$. Hence,

$$\mathbf{E}[\sum_i X_i] = \frac{|S|}{2^m}.$$

Also we calculate the variance

$$\begin{aligned}\mathbf{Var}[X_i] &= \mathbf{E}[X_i^2] - \mathbf{E}[X_i]^2 \\ &\leq \mathbf{E}[X_i^2] \\ &= \mathbf{E}[X_i] = \frac{1}{2^m}.\end{aligned}$$

Because $X_1, \ldots, X_k$ are pairwise independent,

$$\mathbf{Var}\left[\sum_i X_i\right] = \sum_i \mathbf{Var}[X_i] \leq \frac{|S|}{2^m}.$$

Using Chebyshev's Inequality, we get

$$\begin{aligned}\mathbf{Pr}\left[\left||\{a \in S : h(a) = 0\}| - \frac{|S|}{2^m}\right| \geq \epsilon\frac{|S|}{2^m}\right] &= \mathbf{Pr}\left[\left|\sum_i X_i - \mathbf{E}[\sum_i X_i]\right| \geq \epsilon\mathbf{E}[\sum_i X_i]\right] \\ &\leq \frac{\mathbf{Var}[\sum_i X_i]}{\epsilon^2\mathbf{E}[\sum_i X_i]^2} \leq \frac{\frac{|S|}{2^m}}{\epsilon^2\frac{|S|^2}{(2^m)^2}} \\ &= \frac{2^m}{\epsilon^2\,|S|} \leq \frac{1}{4}.\end{aligned}$$

□