Notes for Lecture 6

1 Approximate Counting with an NP oracle

We complete the proof of the following result:

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

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

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

Given what we proved in the previous lecture, it only remains to develop an approximate comparison algorithm for $\#CSAT$, that is, an algorithm $\mathtt{a}-\mathtt{comp}$ such that for every circuit C:

- If $\#CSAT(C) \geq 2^{k+1}$ then $\mathtt{a}-\mathtt{comp}(C,k) =$ YES with high probability;
- If $\#CSAT(C) < 2^{k}$ then $\mathtt{a}-\mathtt{comp}(C,k) =$ NO with high probability.

The idea of the proof is to pick a random function $h : \{0,1\}^n \to \{0,1\}^k$, and then consider the number of satisfying assignments for the circuit $C_h(x) := C(x) \wedge (h(x) = \mathbf{0})$. If $\#CSAT(C) \geq 2^{k+1}$ then, on average over the choice of h, $C_h(x)$ has at least two satisfying assignments, but if $\#CSAT(C) \geq 2^{k+1}$ then, on average over the choice of h, $C_h(x)$ has less than one satisfying assignments. Checking if C_h is satisfiable is a test that we would expect to distinguish the two cases.

To make this argument rigorous, we cannot pick the function h uniformly at random among all functions, because then h would be an object requiring an exponential size description, and a description of h (in the form of an evaluation algorithm) has to be part of the circuit C_h. Instead we will pick h from a *pairwise independent* distribution of functions. To improve the distinguishing probability and simplify the analysis, we will work with functions $h : \{0,1\}^n \to \{0,1\}^{k-5}$, and we will treat the case $k \leq 5$ separately.

1.1 Pairwise independent hash functions

Definition 2 *Let H be a distribution over functions of the form $h : \{0,1\}^n \to \{0,1\}^m$. We say that H is a pairwise independent distribution of hash functions if for every two different inputs $x, y \in \{0,1\}^n$ and for every two possible outputs $s, t \in \{0,1\}^m$ we have*

$$\mathop{\mathbb{P}}_{h\in H}[h(x) = s \wedge h(y) = t] = \frac{1}{2^{2m}}$$

Another way to look at the definition is that for every $x \neq y$, when we pick h at random then the random variables $h(x)$ and $h(y)$ are independent and uniformly distributed. In particular, for every $x \neq y$ and for every s, t we have

$$\mathop{\mathbb{P}}_{h}[h(x) = s | h(y) = t] = \mathop{\mathbb{P}}_{h}[h(x) = s]$$

A simple construction of pairwise independent hash functions is as follows: pick a matrix $A \in \{0,1\}^{m \times n}$ and a vector $b \in \{0,1\}^m$ uniformly at random, and then define the function

$$h_{A,b}(x) := Ax + b$$

where the matrix product and vector addition operations are performed over the field $\mathbb{F}_2$. (That is, they are performed modulo 2.)

To see that the pairwise independence property is satisfied, consider any two distinct inputs $x, y \in \{0,1\}^n$ and any two outputs $s, t \in \{0,1\}^m$. If we call $a_1, \ldots, a_m$ the rows of A then we have

$$\mathop{\mathbb{P}}_{A,b}[Ax + b = s \wedge Ay + b = t] = \prod_{i=1}^{m} [a_i^T x + b_i = s_i \wedge a_i^T y + b_i = t_i]$$

because the events $(a_i^T x + b_i = s_i \wedge a_i^T y + b_i = t_i)$ are all mutually independent. The condition $(a_i^T x + b_i = s_i \wedge a_i^T y + b_i = t_i)$ can be equivalently rewritten as

$$a_i^T x - s_i = a_i^T y - t_i \wedge b_i = a_i^T x - s_i$$

and as

$$a_i^T \cdot (x - y) = s_i - t_i \wedge b_i = a_i^T x - s_i$$

and its probability is

$$\begin{aligned} &\mathbb{P}[a_i^T \cdot (x - y) = s_i - t_i \wedge b_i = a_i^T x - s_i] \\ &= \mathbb{P}[a_i^T \cdot (x - y) = s_i - t_i] \cdot \mathbb{P}[b_i = a_i^T x - s_i | a_i^T \cdot (x - y) = s_i - t_i] \\ &= \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4} \end{aligned}$$

Because $x - y$ is a non-zero vector, and so $a_i^T \cdot (x - y)$ is a vector bit, and because b_i is a random bit independent of x, y.

In conclusion, we have

$$\mathop{\mathbb{P}}_{A,b}[Ax + b = s \wedge Ay + b = t] = \left(\frac{1}{4}\right)^m$$

as desired.

We will use the following fact about pairwise independent hash functions.

Lemma 3 *Let H be a distribution of pairwise independent hash functions h $\{0,1\}^n \to \{0,1\}^m$, and Let $S \subset \{0,1\}^n$. Then, for every t*

$$\mathop{\mathbb{P}}_{h \in H}\left[\left| |\{a \in S : h(a) = 0\}| - \frac{|S|}{2^m} \right| \geq t\right] \leq \frac{|S|}{t^2 2^m}. \tag{2}$$

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

$$X_i = \begin{cases} 1 \text{ if } h(a_i) = 0 \\ 0 \text{ otherwise.} \end{cases} \tag{3}$$

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

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

$$\mathbb{E}\left[\sum_i X_i\right] = \frac{|S|}{2^m}. \tag{4}$$

Also we calculate the variance

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

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

$$\mathbf{Var}\left[\sum_i X_i\right] = \sum_i \mathbf{Var}[X_i] \leq \frac{|S|}{2^m}. \tag{5}$$

Using Chebyshev's Inequality, we get

$$\begin{aligned} \mathbb{P}\left[\left| |\{a \in S : h(a) = 0\}| - \frac{|S|}{2^m} \right| \geq t\right] &= \mathbb{P}\left[\left| \sum_i X_i - \mathbb{E}[\sum_i X_i] \right| \geq t]\right] \\ &\leq \frac{\mathbf{Var}[\sum_i X_i]}{t^2} \\ &= \frac{|S|}{t^2 2^m} \end{aligned}$$

□

1.2 The algorithm a-comp

We define the algorithm `a-comp` as follows.

- input: C, k
- if $k \leq 5$ then check exactly whether $\#CSAT(C) \geq 2^k$.

- if $k \geq 6$
  - pick h from a set of pairwise independent hash functions $h : \{0,1\}^n \to \{0,1\}^m$, where $m = k - 5$
  - answer YES iff there are more then 48 inputs x to C such that $C(x) = 1$ and $h(x) = \mathbf{0}$.

Notice that the test at the last step can be done with one access to an oracle to **NP** and that the overall algorithm runs in probabilistic polynomial time given an **NP** oracle.

We now analyze the correctness of the algorithm.

Let $S \subseteq \{0,1\}^n$ be the set of inputs x such that $C(x) = 1$. There are 2 cases.

- If $|S| \geq 2^{k+1}$, let $S' \subseteq S$ be an arbitrary subset of S of size exactly 2^{k+1}. Then $|S|/2^m = 64$ and we can use Lemma 3 to estimate the error probability as:

$$\mathop{\mathbb{P}}_{h\sim H}[|\{x \in S : h(x) = 0\}| \leq 48]$$

$$\leq \mathop{\mathbb{P}}_{h\sim H}[|\{x \in S' : h(x) = 0\}| \leq 48]$$

$$= \mathop{\mathbb{P}}_{h\sim H}\left[\frac{|S'|}{2^m} - |\{x \in S : h(x) = 0\}| \geq 16\right]$$

$$\leq \frac{1}{16^2} \cdot \frac{|S|}{2^m} = \frac{1}{4}$$

- If $|S| < 2^k$, then $|S|/2^m < 32$, and the probability of error can be estimated as

$$\mathop{\mathbb{P}}_{h\sim H}[|\{x \in S : h(x) = 0\}| > 48]$$

$$\leq \mathop{\mathbb{P}}_{h\sim H}\left[|\{x \in S : h(x) = 0\}| - \tfrac{|S|}{2^m} \geq 16\right]$$

$$\leq \frac{1}{16^2} \cdot \frac{|S|}{2^m} \leq \frac{1}{8}$$

Therefore, the algorithm will give the correct answer with probability at least $3/4$, which can then be amplified to, say, $1 - 1/4n$ (so that all n invocations of `a-comp` are likely to be correct) by repeating the procedure $O(\log n)$ times and taking the majority answer.

2 Approximate Sampling

So far we have considered the following question: for an **NP**-relation R, given an input x, what is the size of the set $R_x = \{y : (x,y) \in R\}$? A related question is to be able to sample from the uniform distribution over R_x.

Whenever the relation R is "downward self reducible" (a technical condition that we won't define formally), it is possible to prove that there is a probabilistic algorithm running in time polynomial in $|x|$ and $1/\epsilon$ to approximate within $1 + \epsilon$ the value $|R_x|$ if and only if there is a probabilistic algorithm running in time polynomial in $|x|$ and $1/\epsilon$ that samples a distribution ϵ-close to the uniform distribution over R_x.

We show how the above result applies to 3SAT (the general result uses the same proof idea). For a formula ϕ, a variable x and a bit b, let us define by $\phi_{x\leftarrow b}$ the formula obtained by substituting the value b in place of x.[1]

If ϕ is defined over variables $x_1, \ldots, x_n$, it is easy to see that

$$\#\phi = \#\phi_{x\leftarrow 0} + \#\phi_{x\leftarrow 1}$$

Also, if S is the uniform distribution over satisfying assignments for ϕ, we note that

$$\mathop{\mathbb{P}}_{(x_1,\ldots,x_n)\leftarrow S}[x_1 = b] = \frac{\#\phi_{x\leftarrow b}}{\#\phi}$$

Suppose then that we have an efficient sampling algorithm that given ϕ and ϵ generates a distribution ϵ-close to uniform over the satisfying assignments of ϕ.

Let us then ran the sampling algorithm with approximation parameter $\epsilon/2n$ and use it to sample about $\tilde{O}(n^2/\epsilon^2)$ assignments. By computing the fraction of such assignments having $x_1 = 0$ and $x_1 = 1$, we get approximate values p_0, p_1, such that $|p_b - \mathbb{P}_{(x_1,\ldots,x_n)\leftarrow S}[x_1 = b]| \leq \epsilon/n$. Let b be such that $p_b \geq 1/2$, then $\#\phi_{x\leftarrow b}/p_b$ is a good approximation, to within a multiplicative factor $(1 + 2\epsilon/n)$ to $\#\phi$, and we can recurse to compute $\#\phi_{x\leftarrow b}$ to within a $(1 + 2\epsilon/n)^{n-1}$ factor.

Conversely, suppose we have an approximate counting procedure. Then we can approximately compute $p_b = \frac{\#\phi_{x\leftarrow b}}{\#\phi}$, generate a value b for x_1 with probability approximately p_b, and then recurse to generate a random assignment for $\#\phi_{x\leftarrow b}$.

The same equivalence holds, clearly, for 2SAT and, among other problems, for the problem of counting the number of perfect matchings in a bipartite graph. It is known that it is **NP**-hard to perform approximate counting for 2SAT and this result, with the above reduction, implies that approximate sampling is also hard for 2SAT. The problem of approximately sampling a perfect matching has a probabilistic polynomial solution, and the reduction implies that approximately counting the number of perfect matchings in a graph can also be done in probabilistic polynomial time.

The reduction and the results from last section also imply that 3SAT (and any other **NP** relation) has an approximate sampling algorithm that runs in probabilistic polynomial time with an **NP** oracle. With a careful use of the techniques from last week it is indeed possible to get an *exact* sampling algorithm for 3SAT (and any other **NP** relation) running in probabilistic polynomial time with an **NP** oracle. This is essentially best possible, because the approximate sampling requires randomness by its very definition, and generating satisfying assignments for a 3SAT formula requires at least an **NP** oracle.

[1] Specifically, $\phi_{x\leftarrow 1}$ is obtained by removing each occurrence of $\neg x$ from the clauses where it occurs, and removing all the clauses that contain an occurrence of x; the formula $\phi_{x\leftarrow 0}$ is similarly obtained.