COM S 6810 Theory of Computing	February 24, 2009
Lecture 11: Complexity of Counting I	
Instructor: Rafael Pass	*Scribe: Shuang Zhao*

1 Basic Concepts

Given a polynomial-time computable relation R, let L_R be the language defined by

$$L_R := \{x \mid \exists y, (x, y) \in R\}.$$

Then it holds that $L_R \in \mathsf{NP}$. Next we consider a 'harder' problem: compute number of y's satisfying $(x, y) \in R$.

Definition 1. Define the function $f_R(x) : \{0,1\}^* \to \mathbb{N}$ as $f_R(x) = \big|\{y \mid (x, y) \in R\}\big|$. Let

$$\#R := \{(x, k) \mid f_R(x) \geq k\}.$$

Proposition 1. *For all polynomial-time computable R, deciding $\#R$ and computing f_R are Turing-reducible to one another.*

Proof. The reduction from $f_R(x)$ to $\#R$ is obvious. Conversely, if we have the access to an oracle deciding whether $(x, k) \in \#R$, then $f_R(x)$ can be computed by performing a binary search on the value of k, costing *poly*(x) time. ■

Definition 2. $\#\mathsf{P}$ is defined by the class of languages L such that $L = \#R$ for some polynomial-time computable relation R. L is $\#\mathsf{P}-$complete if $L \in \#\mathsf{P}$ and $R \leq_p L$ for all $R \in \#\mathsf{P}$.

For every NP language decided by NTM M, there is a 'generalized' problem in $\#\mathsf{P}$ which computes the number of certificates that make M accepting a given input x. Therefore,

Fact. $\mathsf{NP} \leq \#\mathsf{P} \subseteq \mathsf{PSPACE}$.

Definition 3. We say f is a *parsimonious reduction* from $\#Q$ to $\#R$, if it is polynomial-time computable and for all x, $f_Q(x) = f_R(f(x))$.

Notation 1. If $\#R$ is parsimoniously reducible from Q, we write $\#Q \leq_{par} \#R$.

If f is a parsimonious reduction from $\#Q$ to $\#R$, then $L_Q \leq L_R$, since $x \in L_Q$ iff $f(x) \in L_R$. Conversely, if $\#Q \leq_{par} \#R$, then $(x, k) \in \#Q \Leftrightarrow (f(x), k) \in \#R$.

Theorem 1. #SAT *is* $\#\mathsf{P}-$complete .

Conjecture. $\#L$ is $\#\mathsf{P}-$complete implies that L is NP-complete.

Unfortunately, this conjecture is FALSE:

Theorem 2. *There exists a polynomial-time computable relation R such that $\#R$ is* $\#\mathsf{P}-$complete *but $L_R \in \mathsf{P}$.*

Proof. Define R as follows:

$$(x, y') \in R \;\Leftrightarrow\; y' = 0 \vee (y' = 1y \wedge (x, y) \in R_{SAT}),$$

where $(x, y) \in R_{SAT}$ iff the boolean formula ϕ described by x is satisfied by assigning the values described by y to the variables in ϕ.

It is obvious that $L_R \in \mathsf{P}$ since $(x, 0) \in R$ for all $x \in \{0,1\}^*$, namely $L_R = \{0,1\}^*$. On the other hand, $\#\text{SAT} \leq_{par} \#R$, since $(x, k) \in \#\text{SAT} \Leftrightarrow (x, k+1) \in \#R$. Therefore $\#R$ is $\#\mathsf{P}-$complete . ■

Definition 4. Given an $n \times n$ matrix A, its *permanent* is defined by

$$perm(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} A_{i,\sigma(i)}.$$

Theorem 3 (Valiant)**.** *Computing permanent of 0-1 matrices is* $\#\mathsf{P}-$complete *.*

As shown in Figure 1, given an $n \times n$ 0-1 matrix A, a bipartite graph $G(X, Y, E)$ can be built as follows: $X = \{x_1, x_2, \ldots, x_n\}$, $Y = \{y_1, y_2, \ldots, y_n\}$, $(x_i, y_j) \in E \Leftrightarrow A_{i,j} = 1$.

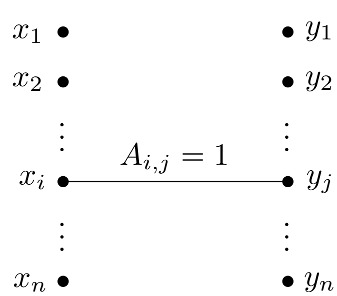

Figure 1: Constructing bipartite graph for 0-1 matrix A

Then it is easy to verify that the permanent of A equals the number of perfect matchings in G. Therefore, counting the number of perfecting matchings in a bipartite graph is also $\#\mathsf{P}-$complete .

2 Approximate Counting

Theorem 4. *Given any polynomial p, there exists a PPT A such that*

$$Pr\left[\#\text{SAT}(\phi)\cdot\left(1-\frac{1}{p(n)}\right)\le A^{\mathsf{NP}}(\phi)\le\#\text{SAT}(\phi)\cdot\left(1+\frac{1}{p(n)}\right)\right]\ge 1-2^{-n}.$$

Basic idea. For all ϕ, if we can find a rough approximation $A'(\phi)$ such that

$$\#\text{SAT}(\phi)\cdot 2^{-i}\le A'(\phi)\le\#\text{SAT}(\phi)\cdot 2^{i}$$

for some constant i, then we are able to obtain a tighter approximation by:

(1) construct ϕ' from ϕ such that $\#\text{SAT}(\phi')=\#\text{SAT}(\phi)^k$ for some k;

(2) output $A'(\phi')^{1/k}$.

Since

$$\#\text{SAT}(\phi)^k\cdot 2^{-i}=\#\text{SAT}(\phi')\cdot 2^{-i}\le A'(\phi')\le\#\text{SAT}(\phi')\cdot 2^{i}=\#\text{SAT}(\phi)^k\cdot 2^{i},$$

it holds that

$$\#\text{SAT}(\phi)\cdot 2^{-i/k}\le A'(\phi')^{1/k}\le\#\text{SAT}(\phi)\cdot 2^{i/k}.$$

For step (1), ϕ' can be constructed by

$$\phi'=\bigwedge_{i=1}^{k}\phi(\vec{x}_i),$$

where $\phi(\vec{x}_i)$ is a copy of ϕ with the variables renamed to $\vec{x}_i$.

Consider GAP-SAT:

$$\Pi_Y=\{(\phi,k)\mid\#\text{SAT}(\phi)\ge 8k\};$$
$$\Pi_N=\{(\phi,k)\mid\#\text{SAT}(\phi)\le k/8\}.$$

Claim. There exists a polynomial-time TM A such that A^O approximates #SAT within factor $8^{1.5}$ where O is an oracle that solves GAP-SAT.

Proof. Let $A(\phi)$ work as follows:

```
i ← 0
while O(φ, 8^i) = 1 do
    i ← i + 1
end while
return 8^(i − 1/2)
```

After exiting the while loop, it holds that $O(\phi, 8^i) \neq 1$ and $O(\phi, 8^{i-1}) = 1$, which implies that $8^{i-2} < \#SAT(\phi) < 8^{i+1}$. Thus

$$8^{-1.5} < \frac{\#SAT(\phi)}{8^{i-\frac{1}{2}}} < 8^{1.5}. \qquad \blacksquare$$

Next lecture we will show how to solve GAP-SAT (with the power of randomness).