

Mar 07, 2008 - Counting Complexity

Lecture note of EECS 395 Winter 2008 (Prof. Fortnow) taken by Bach Ha

Introduction

- NP problems ask if there is a function.
- #P problems ask for the number of solutions.
- Ex: What is the number of satisfying assignments for SAT?
- Reminding: FP is a class of polynomial time computable functions

Formal Definition

- Note that this is not a language class.
- #P is a class of function $\Sigma^* \rightarrow \mathbb{N}$: there is a nondeterministic TM M such that $f(x) =$ the number of accepting computation paths of M on input x .
- Ex: #SAT is the number of satisfying assignments of ϕ
- #SAT \in #P: simply guess satisfying assignments
- f is #P-complete if $f \in \#P$ and $\forall g \in \#P, g \in \text{FP}^f$.
- An oracle for counting function contains an input tape, an output tape, a polynomial time work tape and a special oracle state $q_?$. When it enters $q_?$ state and x is in input tape, then $f(x)$ magically appears on the output tape.
- #SAT is #P-complete: we can use the same proof that SAT is NP-complete with the note that the number of proper tableau is exactly the number of accepting computation paths. Furthermore, using Cook's theorem, satisfying assignment describe a proper tableau.
- The reduction of SAT to 3-SAT doesn't preserve the number of satisfying assignments, so we need to do some clever computation to compute the number of assignment of SAT using those from 3-SAT

Some surprising #P complete problem

- #2-SAT (note that 2-SAT is in P) is NP-complete.
- Bipartite matching problem is in P, but counting is #P-complete
- $\text{Det}(M)$ is FP using Gaussian elimination, but the Permanent(M) is #P-complete
- The number of paths from s to t with at most k edge is in P. (This can be done by consider the adjacency matrix A . And look at A^k and its coordinate). However the number of simple paths in a directed graph G is #P-complete.
- In fact, $\text{PH} \subseteq \text{P}^{\#\text{P}}$ (we will prove this)

Some properties

- Let $f, g \in \#\text{P}$.
- $f + g$ is in #P (run 2 machines parallel)
- $f \cdot g$ is in #P (run 2 machines in sequence)
- #5 in the homework

Counting classes

- NP is a set of language L , such that there is a function $f \in \#\text{P}$ such that $x \in L$ then $f(x) > 0$, or else $f(x) = 0$.
- PP is a set of language L , such that there is a function $f \in \#\text{P}$ and $g \in \text{FP}$ such that if $x \in L$ then $f(x) \geq g(x)$ or else $f(x) < g(x)$.
- Our predefined PP (Probabilistic Polynomial time) is indeed the same.
- Parity P: $\oplus\text{P}$ is a set of language L such that there is a function $f \in \#\text{P}$ such that if $x \in L$ then $f(x)$ is odd, or else even.
- Lemma: $\oplus\text{P}^{\oplus\text{P}} = \oplus\text{P}$.