Counting problems (#P)

We are going to discuss *counting* versions of NP problems. It will be more convenient for us to use the definition of NP in terms of deterministic polynomial computations with hints. Namely, for an NP-problem A,

$$x \in A \iff (\exists y)\,(|y| < p(|x|) \text{ and } R(x,y)),$$

where R is a polynomial-time computable predicate.

The corresponding counting problem, $\#A$, is the question *how many* hints, or *witnesses,* y, for a given x, satisfy $R(x,y)$. Each NP-problem has its counting counterpart. For example, SAT is the question "does the given Boolean formula have a satisfying assignment?", while #SAT is "how many satisfying assignments does this Boolean formula have?"; HAMPATH is "is there a Hamiltonian path in the given (directed) graph?" and #HAMPATH is "how many Hamiltonian paths does this graph have?" etc. These counting counterparts of NP decision problems form the #P class.

Clearly, the counting problem $\#A$ is at least as hard as the decision problem A: if we can count the number of witnesses, we can then compare it with zero and answer whether there exists at least one witness. In particular, if $\mathsf{P} \neq \mathsf{NP}$ and A is NP-complete, then $\#A$ cannot be solved in polynomial time.

There are, however, situations when $A \in \mathsf{P}$, but $\#A$ is hard (i.e., not polynomial-time solvable, unless $\mathsf{P} = \mathsf{NP}$). In order to establish results of this sort, one develops the theory of #P-completeness, which runs in parallel with the theory of NP-completeness.

Definition. A polynomial-time *counting reduction* of $\#A$ to $\#B$ is a pair of functions f and g, where f maps inputs of $\#A$ to inputs of $\#B$, and $g\colon \mathbb{N} \to \mathbb{N}$, such that

$$\#A(x) = g(\#B(f(x)).$$

Notation: $\#A \leq_c^P \#B$.

Counting reduction is indeed a reduction in the following sense: if $\#B$ is polynomial-time solvable and $\#A \leq_c^P \#B$, then $\#A$ is also polynomial-time solvable.

A specific kind of counting reduction is *parsimonious reduction,* in which g is the identity function (i.e., $\#A(x) = \#B(f(x))$, just as in m-reductions of decision problem).

Definition. A counting problem $\#B$ is #P*-complete,* if for any $\#A \in \#\mathsf{P}$ we have $\#A \leq_c^P \#B$.

The reduction in the proof of Cook–Levin theorem and Tseitin transformations can be performed parsimoniously; thus, we establish the fact that #SAT and #3-SAT are #P-complete.

Using only parsimonious reductions, however, does not give interesting results. The reason is that each parsimonious reduction for counting problems induces an m-reduction on decision problems. Thus, establishing #P-completeness by parsimonious reductions implies NP-completeness for corresponding decision problems (which itself already yields hardness of the counting problem without any reference to the theory of #P-completeness).

More general counting reductions, however, could yield new results. Our first example is #DNF-SAT, the counting problem for satisfying assignments of Boolean formulae in DNF. The corresponding decision problem, DNF-SAT, is polynomial-time decidable. For the counting problem the situation is different, namely, #3-SAT $\leq_c^P$ #DNF-SAT, whence #DNF-SAT is #P-complete.

Indeed, if φ is a 3-CNF, then its negation, $\neg\varphi$, can be easily (polynomial-time) transformed into an equivalent DNF. Next, an assignment is satisfying for φ iff it is not satisfying for $\neg\varphi$, therefore

$$\#\mathrm{SAT}(\varphi) = 2^n - \#\mathrm{SAT}(\neg\varphi),$$

where n is the number of variables (2^n is the total number of assignments). Thus, #3-SAT is reduced to #DNF-SAT by the following counting reduction:

$$f \colon \varphi \mapsto \neg\varphi;$$
$$g \colon k \mapsto 2^n - k.$$

Now, if $\mathsf{P} \neq \mathsf{NP}$, then #DNF-SAT is not polynomial-time solvable. Indeed, polynomial-time solvability of #DNF-SAT, by the aforementioned reduction, would yield that of #3-SAT, and thus of 3-SAT, which is impossible.

The more famous example of #P-complete problem is the *permanent* problem. We consider $n \times n$ Boolean matrices (i.e., constructed 0's and 1's, with operations modulo 2). Recall that the *determinant* of a matrix is

$$\det(a_{ij}) = \sum_{\sigma \in \mathbf{S}_n} (-1)^{\mathrm{sign}(\sigma)} \prod_{i=1}^{n} a_{i\sigma(i)}.$$

(Here $\mathbf{S}_n$ is the set (group) of all permutations of $\{1, \ldots, n\}$.) The determinant can be polynomially computed using, say, Gaussian elimination procedure.

The permanent is defined similarly, but without alternation of signs:

$$\mathrm{perm}(a_{ij}) = \sum_{\sigma \in \mathbf{S}_n} \prod_{i=1}^{n} a_{i\sigma(i)}.$$

Computing the permanent can be seen as a counting problem: for a given matrix (a_{ij}), count the number of permutations $\sigma \in \mathbf{S}_n$ such that $a_{i\sigma(i)} = 1$ for all i. Thus, PERM $\in$ #P. Valiant (1979) showed #P-hardness of PERM, but using a more general notion of Turing reduction; Ben-Dor and Halevi (1993) did it by counting reduction. For proof details, see their paper: A. Ben-Dor, S. Halevi (1993), "Zero-one permanent is #P-complete, a simple proof."

Interestingly enough, #P-completeness of PERM yield that of #2-SAT (Valiant 1979), even in its fragment with only monotone 2-CNF's (without negations). Recall that the 2-SAT decision problem is polynomial. For proof, see L. G. Valiant (1979), "The complexity of enumeration and reliability problems."