CS221: Computational Complexity Prof. Salil Vadhan

Lecture 25: Counting Problems (cont.)

11/22 *Scribe: Grant Schoenebeck*

Contents

1 Counting Matchings

Definition 1 *A* matching *is a set of edges such that no two edges in the set share a vertex. A* perfect matching *is a matching in which every vertex is touched by at least one (and hence exactly one) edge.*

Recall that #**P** contains all problems which amount to counting the number of witnesses to an **NP** relation. Last time, we stated (without proof):

Theorem 2 *Counting the # of perfect matchings,* #PERFECT MATCHINGS*, in a bipartite graph is* #**P***-complete.*

Proof: It is equivalent to computing the permanent of a 0/1 matrix. ■

Now we'll show:

Theorem 3 *Counting the # of matchings (perfect or not) in a bipartite graph G,* #MATCHINGS*, is* #**P***-complete.*

Proof: Clearly it is in #**P** because testing whether a set of a matching is in **P**. We want to show that it is #**P**-hard. We reduce from #PERFECT MATCHINGS via a Cook reduction. That is, given an oracle for #MATCHINGS we can solve #PERFECT MATCHINGS.

Say we are given a bipartite graph G that has n vertices on each side. Let

$$N_i = \text{ \# of matchings in } G \text{ consisting of exactly } i \text{ edges.}$$

So we want to compute N_n, given an oracle for #Matchings. Applying the oracle on G itself, we obtain:

$$\#\textsc{Matchings}(G) = \sum_{i=0}^{n} N_i$$

Now we will make oracle calls on related graphs to obtain additional equations involving the N_i's until we can solve for N_n.

We produce a new graph G_t from G by starting with G and creating $n \cdot t$ new vertices on the right-hand side and connecting t of the new vertices to each vertex on the left side of G_t with a new edge. Now

$$\#\textsc{Matchings}(G_t) = \sum_{i=0}^{n} N_i (t+1)^{n-i}$$

We can see this by first determining which of the old edges (edges from G) we are going to include in our set of edges for the matching, then consider the new edges. So fix a matching on G, then for each unused vertex on the left side, we can use any of the t edges that we added, or we can use none of the t new edges. Thus we have $(t+1)^{(\text{number of unused vertices on the left side})}$ matchings in G_t for each matching in G. This is just $(t+1)^{n-i}$ where i is the number of edges used in the matching in G.

So now we can define the polynomial $P_G(x) = \sum_{i=0}^{n} N_i x^{n-i}$ and then we see

$$\#\textsc{Matchings}(G_t) = P_G(t+1)$$

Lemma 4 *Let $\mathbb{F}$ be a field and $\alpha_0, \alpha_1, \ldots, \alpha_d \in \mathbb{F}$ be distinct and $\beta_o, \beta_1, \ldots, \beta_d \in \mathbb{F}$. Then there exists a unique polynomial of degree $\leq d$ such that $p(\alpha_i) = \beta_i$ for $i = 0, 1, 2, \ldots, d$. Moreover p can be found in polynomial time.*

Proof: Lagrange interpolation formula:

$$p(x) = \sum_{i=0}^{d} \beta_i \prod_{j \neq i} \frac{(x - \alpha_i)}{\alpha_i - \alpha_j}$$

It is easy to check that for the summand term i, the product is 1 if $x = \alpha_i$ and 0 at α_j where $i \neq j$.

The uniqueness comes from the fact that if we had two polynomials that worked we could take their difference and that would still be a polynomial of degree $\leq d$ but would have at least d+1 roots (the d+1 points at which the two polynomials agreed). Which implies that their difference is the zero polynomial, so they were the same after all.

$\blacksquare$

So querying the #MATCHINGS oracle at G_t for $t = 0, \ldots, n$ we recover P_G (by lemma) and we can just look at the first coefficient of P_G, which equals N_n. $\blacksquare$

Theorem 5 (Toda) $\mathbf{PH} \subset \mathbf{P}^{\#\mathbf{P}}$

A final remark is that not *all* nontrivial #**P** problems are hard. For example, counting the number of perfect matchings in *planar graphs* is in **P**.

2 Approximate Counting

So far we've discussed solving counting problems *exactly*. But what often an approximately correct answer is good enough. What does it mean to "approximate"? Two common notions:

1. absolute or additive error – e.g., given a formula φ compute the fraction of assignments that satisfy φ up to $\pm\epsilon$.

2. relative error–Compute # up to a fraction of the correct answer. E.g. in the range of $(1 \pm \epsilon) \cdot$ correct answer

Many approximation algorithms are randomized, so we will allow randomized algorithms in our discussion as well. Approximation with additive error can always be achieved in time poly$(n, 1/\varepsilon)$ by random sampling. But this notion of approximation is often useless, e.g. if φ has $< 2^{n/2}$ satisfying assignments because then 0 is a good approximation. Thus we will focus on relative error:

Definition 6 *For* $\alpha \geq 1$*, an* α-approximation algorithm *for* $f : \{0,1\}^* \to \mathbb{R}$ *is a probabilistic algorithm* A *such that for all* x

$$Pr\left[f(x) \leq A(x) \leq \alpha f(x)\right] \geq \frac{2}{3},$$

where the probability is taken over the coin tosses of A*.*

Remarks:

- $\alpha = 1$ corresponds to an exact algorithm, and thus the goal (in algorithm design) is to have α as close to 1 as possible.
- The fact that the error is always above the actually answer is arbitrary and can be switched to be below be dividing the result by α.
- The success probability of $\frac{2}{3}$ can be amplified by running the algorithm many times and taking the median.

Fact 7 *Approximate counting is at least as hard as decision.*

$$f(x) = 0 \Rightarrow \text{ must output } 0$$

$$f(x) > 0 \Rightarrow \text{ must output something} > 0$$

Fact 8 *Many approximate counting algorithms are known for* #**P***-complete problems, e.g.* #MATCHINGS*,* #PERFECT MATCHINGS*, #DNF.*

Fact 9 *For most natural* **NP** *relations (namely, ones which are "self reducible" in an appropriate sense) approximate counting is equivalent to generating witnesses almost uniformly at random, and in fact this is how the above-mentioned algorithms are designed.*

Fact 10 *Approximate counting can always be done in probabilistic polynomial time with an* **NP***-oracle ("***NP***-easy"). So these approximation problems are unlikely to be* #**P***-complete (by Toda).*

There are examples where decision is easy but approximate counting is hard. Let #CYCLES be the problem of counting the number of simple cycles (i.e. cycles without any repeated vertices) in a graph.

Theorem 11 *Approximating* #CYCLES *is* **NP** *hard (for any constant approximation factor).*

Proof: Reduce from HAMILTONIAN CYCLE which is **NP**-complete. Given G with n vertices we want to decide if it has a simple cycle of length n. We use a common technique in which we create a new graph G' that "blows up" the number of simple cycles in G especially long cycles. So if there is a Hamiltonian cycle in G, there will be a lot of cycles in G'. To do this, we replace each edge in G by $t-1$ new vertices and $2 \cdot t$ edges. The first original vertex has two edges going from it to the first new vertex, which has two edges going from it to the second new vertex, which has two edges to the third vertex, and so on. This happens until we run out of new vertices and then we connect the last new vertex to the remaining original vertex with 2 edges.

Now any time a cycle in G uses a particular edge it now as 2^t different ways to traverse the edge in G'. At each new vertex (and at the first original one) the path can take either of the two edges. So a cycle of length k in G yields $\left(2^t\right)^k$ cycles in G'. So if G has an hamiltonian cycle, the new graph will have at least $\geq 2^{t \cdot n}$ cycles (where n is the number of vertices). If G does not have a hamiltonian cycle then each cycle of G is blown up by a factor of at most $2^{t \cdot (n-1)}$. Since G has at most n^n cycles, G' has at most

$$n^n \cdot 2^{t \cdot (n-1)} = 2^{t \cdot n} \cdot \frac{n^n}{2^t}.$$

cycles. So the ratio of the number of cycles in G' if G has a Hamiltonian cycle to the number of cycles in G' where G has no Hamiltonian cycles is at least

$$\frac{2^{t \cdot n}}{2^{t \cdot n} \cdot \frac{n^n}{2^t}} = \frac{2^t}{n^n}$$

Which is arbitrarily large if we let t get large. In particular, if $t = n^2$ (so G' is only polynomially larger than G), then this ratio goes to ∞. So if we could approximate the number of cycles to within a constant factor, we could decide whether G has a Hamiltonian cycle, which is **NP**-complete. ■