CSE 531: Computational Complexity I **Winter 2016**

Lecture 13: The Complexity of Counting

Feb 17, 2016

Lecturer: Paul Beame *Scribe: Paul Beame*

We first begin with a bit of unfinished business.

1 Proof of Lauteman's Lemma

In the proof of $\mathsf{BPP} \subseteq \Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}$ in last class, we used the following lemma we need to prove. It allows us to distinguish between very small and very large sets.

Lemma 1.1. *Let $S \subseteq \{0,1\}^m$.*

1. *Let $k = \lceil \frac{m}{n} + 1 \rceil$. If $|S|/2^m \geq 1 - 2^{-n}$ then $\exists u_1, \ldots, u_k \in \{0,1\}^m$ such that*

$$\bigcup_{i=1}^{k} (S \oplus u_i) = \{0,1\}^m.$$

2. *If $k < 2^m$ and $|S|/2^m \leq 2^{-n}$ then $\forall u_1, \ldots, u_k \in \{0,1\}^m$ such that*

$$\bigcup_{i=1}^{k} (S \oplus u_i) \neq \{0,1\}^m.$$

Proof. Part 2 is easier to prove first: Since $|S|/2^m \leq 2^{-n}$ and $|S \oplus u_i| = |S|$,
we have $|\bigcup_{i=1}^{k}(S \oplus u_i)| \leq k|S| \leq k2^{m-n} < 2^m$ since $k < 2^n$ which implies that $\bigcup_{i=1}^{k}(S \oplus u_i) \neq \{0,1\}^m$.

For part 1, we use the probabilistic method: In particular, we show that if we choose $u_1, \ldots, u_k$ uniformly and independently from $\{0,1\}^m$, then the *probability* that there is some r in $\{0,1\}^m$ that is not in $\bigcup_{i=1}^{k}(S \oplus u_i)$ is < 1. Hence, there must be some contribution due to a choice of $u_1, \ldots, u_k$ where all elements of $\{0,1\}^m$ are covered, and that choice proves the Lemma.

Fix $r \in \{0,1\}^m$. We have $r \in S \oplus u_i$ if and only if $u_i \in S \oplus r$. Since u_i is uniformly chosen, and $|S \oplus r| = |S| \geq 2^m(1 - 2^{-n})$, we have

$$\mathbb{P}_{u_i}[r \notin S \oplus u_i] \leq 2^{-n}.$$

Therefore, since the choices of the u_i are independent and $k = \lceil m/n + 1 \rceil$,

$$\mathbb{P}_{u_1,\ldots,u_k}[r \notin \bigcup_{i=1}^{k}(S \oplus u_i)] \leq 2^{-kn} \leq 2^{-m-n}.$$

Therefore, by a union bound,

$$\mathbb{P}_{u_1,\ldots,u_k}[\exists r \in \{0,1\}^m.\ r \notin \bigcup_{i=1}^{k}(S \oplus u_i)] \leq 2^m \cdot 2^{-m-n} = 2^{-n} < 1,$$

which is what we wanted to show. □

2 Functions and the Complexity of Counting

Thus far, though we have used function computation as part of the notions of reduction, all the definitions of complexity classes we have used have been of languages or, equivalently, Boolean functions.

Definition 2.1. *Define* $\mathsf{FDTIME}(\mathsf{T}(\mathsf{n}))$ *to be the set of functions* $f : \{0,1\}^* \to \{0,1\}^*$ *(or, alternatively,* $f : \{0,1\}^* \to \mathbb{N}$*) that are computable by a TM* M *with running time* $O(T(n))$.

Then $\mathsf{FP} = \bigcup_{\mathsf{k}} \mathsf{FDTIME}(\mathsf{n}^{\mathsf{k}})$.

We can similarly define $\mathsf{FDSPACE}(\mathsf{T}(\mathsf{n}))$ (recalling that the output tape is not included in the space bound) and $\mathsf{FPSPACE} = \bigcup_{\mathsf{k}} \mathsf{FDSPACE}(\mathsf{n}^{\mathsf{k}})$.

We will be interested in function computation problems involving counting.

Definition 2.2. *Let* $\#\mathsf{P}$ *(usually pronounced "sharp P", or occasionally "number P") be the set of functions* $f : \{0,1\}^* \to \mathbb{N}$ *for which there is a polynomial time TM* M *and a polynomial* p *such that* $f(x) = \#\{y \in \{0,1\}^{p(|x|)} \mid M(x,y) = 1\}$.

$\#\mathsf{P}$ is a complexity class that represents many useful natural problems. In particular it includes many important problems related to probabilistic inference.

Lemma 2.3. $\mathsf{FP} \subseteq \#\mathsf{P} \subseteq \mathsf{FPSPACE}$

Proof. Along the same lines as the arguments for NP and BPP, it is immediate that $\#\mathsf{P} \subseteq \mathsf{FPSPACE}$ by simply trying all possible y and keeping track of a count. To see that $\mathsf{FP} \subseteq \#\mathsf{P}$, observe that we can modify any polynomial-time TM that produces an integer $N = f(x)$ as output, to one that has an auxiliary input y of some polynomial length larger than the number of bits needed to encode N and, instead of outputting N, accepts all strings y with $y < N$. (Here we view bianry strings and natural numbers as interchangeable.) □

A particularly natural problem in $\#\mathsf{P}$ is $\#SAT$, the problem of counting the number of satisfying assignments for a Boolean formula φ. The following is immediate:

Proposition 2.4. *If* $\#SAT \in \mathsf{FP}$ *then* $\mathsf{P} = \mathsf{NP}$.

We can also define counting problems associated with other NP problems. Here, if we have a language $A \in \mathsf{NP}$ we will select a "natural" verifier V_A for A, and define $\#A \in \#\mathsf{P}$ to be the counting problem associated with that natural verifier.

We now see that even a problem $A \in \mathsf{P}$ can have an associated counting problem that is hard.

Define $CYCLE = \{[G] \mid G \text{ is an directed graph containing a directed cycle}\} \in \mathsf{P}$.
This is not a natural NP formulation of the problem since there are cycles of arbitrary length and an NP formulation would require a bounded length witness. One can of course convert this to only ask about cycles of length at most n but if we look at the usual BFS and DFS algorithms, we see that they not only find cycles, they find *simple* cycles, those that do not contain repeated vertices (which are of course of length at most n. So we now think of

$$CYCLE = \{[G] \mid \exists \text{ a simple directed cycle } C \in G\}.$$

With this formulation, we define $\#CYCLE$ by

$$\#CYCLE([G]) = \#\{C \mid C \text{ is a simple directed cycle in directed graph } G\}.$$

Surprisingly, we have:

Theorem 2.5. *If* $\#CYCLE \in \mathsf{FP}$ *then* $\mathsf{P} = \mathsf{NP}$.

Proof. The proof is by reduction from $HAMCYCLE$. We will show that given a directed graph G, we can produce a polynomal-time mapping from G to a directed graph G' such that $\#CYCLE([G'])$ will be larger than some known value depending only on the number n of vertices of G iff G contains a directed Hamiltonian cycle. In particular, this fixed value will be $2^{n^2\lceil \log_2 n \rceil} \geq n^{n^2}$.

The mapping is as follows: Let $k = n\lceil \log_2 n \rceil$. For each edge (u, v) in G, we add $2k$ new intermediate vertices $a_1, \ldots, a_k, b_1, \ldots, b_k$ (different for each (u, v) pair) and directed edges (u, a_1), (u, b_1), (a_k, v), (b_k, v), as well as edges (a_i, a_{i+1}), (a_i, b_{i+1}), (b_i, a_{i+1}), and (b_i, b_{i+1}). The key points are that there are 2^k different paths in this gadget from u to v and the construction is polynomial time.

Clearly, by construction any cycle in G' consists of a cycle in G with each of its edges replaced by one of these new paths. Also by construction, any original cycle in G of length ℓ corresponds to $(2^k)^\ell$ cycles in G'.

There are $< n^\ell$ simple cycles in G, so the total count of cycles in G' based on simple cycles of length at most $n-1$ in G is less than

$$\sum_{\ell=2}^{n-1} n^\ell (2^k)^\ell < 2 \cdot n^{n-1}(2^k)^{n-1}$$

since this is a geometric series with ratio > 2 and is therefore at most twice its largest term.

On the other hand, the contribution from a single Hamiltonian cycle is

$$(2^k)^n = 2^k(2^k)^{n-1} \geq 2^{n \log_2 n}(2^k)^{n-1} = n^n(2^k)^{n-1},$$

which is strictly larger than the total contribution from all shorter cycles. $(2^k)^n = 2^{kn} \geq 2^{n^2 \log_2 n} = n^{n^2}$. □

We now look at the relationship between this counting class of functions and a class of decision problems based on probabilistic algorithms that have success probability barely larger than 1/2. These have success probability that is not "bounded".

Definition 2.6. *Define* PP *(probabilistic polynomial time) to be the set of languages A for which there is a polynomial time TM M and a polynomial p, such that*

$$x \in A \Leftrightarrow \mathbb{P}_{r \in \{0,1\}^{p(|x|)}}[M(x,r) = 1] > 1/2.$$

Note that for PP, unlike BPP, the success probability might be as small as $1/2 + 1/2^{p(|x|)}$ which would not be distinguishable from failure with only a polynomial number of samples.

In relating functions computable in #P to each other and to other complexity classes we need a notion of efficient reduction. Mapping reductions such as $\leq_P$ don't make sense since the answer is not 0 or 1. Even polynomial-time Turing reductions $\leq_P^T$ as we defined them do not quite work since the oracle answer came in the form of the yes or no answer state.

Definition 2.7. *A* function oracle *TM $M^?$ is a TM with a special query tape, special query and answer states, well as a special answer tape. The computation of $M^?$ with oracle f, denoted M^f behaves like an ordinary TM except when it enters the query state. At this point in one step the function f is applied to the contents of the query tape, the value $f(x)$ appears on the answer tape and the state switches to the answer state.*

Based on above definition of function oracles we can define $\mathsf{DTIME^f(T(n))}$, $\mathsf{FDTIME^f(T(n))}$, P^f, FP^f, etc.

Definition 2.8. *Let* C *be a class of decision problems or* FC *be a class of functions. We define $P^{\mathsf{C}} = \bigcup_{A \in \mathsf{C}} P^A$, $P^{\mathsf{FC}} = \bigcup_{f \in \mathsf{FC}} P^f$, and $\mathsf{FP^{FC}} = \bigcup_{\mathsf{f} \in \mathsf{FC}} \mathsf{FP^f}$.*

The following theorem shows that $\#\mathsf{P}$ and PP are natural analogues of each other.

Theorem 2.9. $\mathsf{P}^{\mathsf{PP}} = \mathsf{P}^{\#\mathsf{P}}$.

Proof. (Start)
$\subseteq$: Let A be any PP language. We show that we can compute $A \in \mathsf{P}^{\#\mathsf{P}}$ and hence replace any PP oracle call. Let M be the polynomial-time TM and p be the polynomial bound such that $x \in A \Leftrightarrow \mathbb{P}_{r \in \{0,1\}^{p(|x|)}}[M(x,r) = 1] > 1/2$. The acceptance condition is precisely equivalent to $\#\{r \in \{0,1\}^* \mid M(x,r) = 1\} > 2^{p(|x|)-1}$. The function on the left is obvisouly computable in $\#\mathsf{P}$. The algorithm for A simple compares this quantity to $2^{p(|x|)-1}$ and accepts iff it is larger.
$\supseteq$: Next class. □