6.842 Randomness and Computation February 20th, 2020

Lecture 5

Lecturer: Ronitt Rubinfeld *Scribe: Eugenia Gleizer*

In today's lecture:

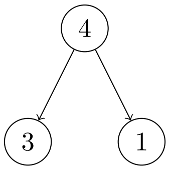
- $\exists$ Exact counter in poly time $\Rightarrow$ $\exists$ Exact uniform generator in poly. time
- Definition of a Language, P, RP and BPP classes
- Derandomization via Enumeration

1 Exact counter $\Rightarrow$ Exact uniform generator

Theorem 1 *If $\exists$ an exact counter algorithm for $F_{b_1 \ldots b_i}$ (from previous lecture), running in polynomial time, then also $\exists$ an exact uniform generator for the same problem, also running in polynomial time*

Notice: We are here stating the theorem for #DNF, as in previous lecture, because it is easier to give a proof for a concrete example. However in reality this theorem holds for all downwards self-reducible problems.

Proof This theorem can be proved via recursion on $b_1 \ldots bi$. We will use the perfect counter to compute $r_0 = F_{b_1 \ldots b_{i-1}0}$ and $r_1 = F_{b_1 \ldots b_{i-1}1}$. After that we can estimate the likelihood of approaching the leaf equal to 1 depending on the choice of the path we made. For example, if $r_0 = 3$ and $r_1 = 1$, it is three times more reasonable for us to set b_i to 0 than to 1.



More precisely, we should go down the left branch of the search tree with probability $\frac{r_0}{r_0+r_1}$ and the right branch with probability $\frac{r_1}{r_0+r_1}$. ■

Claim 2
- *By going down the tree in such a manner we will always reach a satisfying assignment*
- *$P = Pr$[the output assignment will be $b = b_1 \ldots b_n$] $= \frac{F_{b_1}}{F} \times \frac{F_{b_1 b_2}}{F_{b_1}} \times \ldots \times \frac{F_{b_1 b_2 \ldots b_n}}{F_{b_1 b_2 \ldots b_{n-1}}} = \frac{1}{F}$. This follows from fraction reduction and the fact that $F_{b_1 b_2 .. b_n} = 1$. Moreover, we get that:*
- *$P \leq \frac{1}{|F|} \times \frac{(1+\epsilon_0)^n}{\frac{1}{(1+\epsilon_0)^n}} = \frac{1}{|F|} \times (1+\epsilon_0)^{2n}$. This is an extremely generous upper bound, as we are assuming the possibility of independent variation from the norm of numerator and denominator, which is in reality not possible. However, it suffices to show our point if we are to choose $\epsilon_0 \approx (\frac{1}{2n\epsilon})$, giving us that*

$$P \leq \frac{1}{|F|}(1+\epsilon)$$

Theorem 3 (Jerrum, Valiant, Vazirani)

For any downwards self-reducible problem in NP, having polynomial time approximation algorithm to count the number of solutions is equivalent to having a polynomial time algorithm to uniformly generate a solution.

2 Definitions of a language and some complexity classes

Definition 4 *A "language" L is a subset of* $\{0,1\}^*$.

Examples:

- $L_{HP} = \{G | G$ is a graph that has a Hamiltonian path$\}$
- $L_{CNF}= \{\varphi | \varphi$ is satisfiable and in CNF$\}$

Definition 5 *P is a class of languages L with polynomial time deterministic algorithm A s.t.*

$$\forall x \in L \Rightarrow A(x) \text{ accepts}$$
$$\forall x \notin L \Rightarrow A(x) \text{ rejects}$$

Definition 6 *RP is a class of languages L with polynomial time randomized algorithm A s.t.*

$$\forall x \in L \Rightarrow P[A(x) \text{ accepts}] > \frac{1}{2}$$
$$\forall x \notin L \Rightarrow P[A(x) \text{ rejects}] = 1$$

Algorithms deciding languages in RP are said to have *"one-sided error"*

Definition 7 *BPP is a class of languages L with polynomial time randomized algorithm A s.t.*

$$\forall x \in L \Rightarrow P[A(x) \text{ accepts}] > \frac{2}{3}$$
$$\forall x \notin L \Rightarrow P[A(x) \text{ rejects}] > \frac{2}{3}$$

Algorithms deciding languages in BPP are said to have *"two-sided error"*

Notice: Having a language from a RP or BPP classes and using the techniques proved in PS1 - Problem 1, we can come up with an algorithm with any desired accuracy. For RP we simply need to run the algorithm a few times, as done in the homework. For BPP the situation is slightly more complex, as we need to follow the probability of rejections. Though the main idea stays the same and the proof follows from the variation of the *amplification lemma.* Therefore the probability $\frac{2}{3}$ is only one possibility and this definition could have had another figure without loss of generality. More details can be found in *Introduction to the Theory of COmputation* by M.Sipser, page 397.

Main take-away:

- $P \subseteq RP \subseteq BPP$
- P = BPP ??? This remains an open question in complexity theory.

3 Derandomization via Enumeration

We will start with the most simple and intuitive derandomizing algorithm, where we just go through every possible string, get the output for it and then take the majority result appearing. More precisely:

Given: A (a randomized algorithm, running within t(n) time) and **x**(input)

Derandomization algorithm:

- Run A on every possible random string of length r(n), where r(n) $\leq$ runtime of A on input x
- Output majority answer

Behaviour: Assume A has two-sided error.

If x $\in$ L and $\geq \frac{2}{3}$ of random strings cause A to accept $\Rightarrow$ majority answer is 'accept'

If x $\notin$ L and $\geq \frac{2}{3}$ of random strings cause A to reject $\Rightarrow$ majority answer is 'reject'

Runtime $O(t(n) \times 2^{r(n)}) \leq O(t(n) \times 2^{t(n)})$. If t(n) is polynomial in n, then runtime is $O(2^{poly(n)})$

Directly from this follows that BPP $\in$ EXP (= DTIME($\bigcup\limits_c 2^{n^c}$))

End of the lecture