6.842 Randomness and Computation February 23, 2022

Lecture 8

Lecturer: Ronitt Rubinfeld *Scribe: Shih-Yu Wang*

In this lecture, we will look at more applications of pairwise independence, specifically we will present the following:

(1) Reducing confidence error using pairwise independence.

(2) Interactive proofs - public vs. private coins

1 Using Pairwise Independence to Improve Confidence

Recall that for a language $L \in \mathbf{RP}$, there exists a polynomial time algorithm $\mathcal{A}$ such that

- if $x \in L$, then $\Pr[\mathcal{A}(x, r) = \text{“accept”}] \geq 1/2$
- if $x \notin L$, then $\Pr[\mathcal{A}(x, r) = \text{“accept”}] = 0$

To improve the confidence error $\frac{1}{2}$, one naive way is to repeat the algorithm k times using new random bits and output “accept” if any of the results is “accept”; otherwise output “reject”. This reduces the confidence error to 2^{-k} for $x \in L$. However, the drawback is that if the algorithm $\mathcal{A}$ uses r random bits, then we need $O(rk)$ random bits to do k repetitions. Thus, we introduce a technique that allows us to use less random bits at the expense of a slower running time utilizing pairwise independence random strings.

2-Point Sampling Assume that there exists a family of pairwise independent function $\mathcal{H} = \{h : [2^{k+2}] \to \{0,1\}^r\}$, such that we can use $O(k + r)$ random bits to randomly pick a function $h \in \mathcal{H}$ and h evaluates in $\text{poly}(k, r)$ time. Consider the following sampling algorithm to improve confidence error for a given algorithm $\mathcal{A}$ for a language $L \in \mathbf{RP}$:

2-Point Sampling

```
1 : h ←R H
2 : for i = 1 ··· 2^{k+2} do
3 :     r_i ← h(i)
4 :     if A(x, r_i) = accept
5 :         return accept
6 : endfor
7 : return reject
```

Since the only step that requires randomness is picking h uniformly at random from $\mathcal{H}$, it is clear that we only need $O(k + r)$ bits. In terms of running time, suppose that $\mathcal{A}$ runs in time $T(n)$, then the above algorithm runs in time $O(2^k \cdot T(n))$ (which is slower than $O(k \cdot T(n))$ if we do naive repetition).

Now we show that pairwise independent random strings can also efficiently improve the confidence error. Before analyzing the confidence error of the above algorithm, we first recall two useful lemmas:

Lemma 1 (Chebyshev's Inequality)**.** *X is a random variable with expectation $\mu = \mathbb{E}[X]$, then*

$$\Pr\left[|x - \mu| \geq \varepsilon\right] \leq \frac{\text{Var}[x]}{\varepsilon^2}$$

However, it might not be easy to calculate the variance of the sum of dependent random variables. Fortunately, we have the following inequality for pairwise independent random variables.

Lemma 2 (Pairwise Independent Tail Inequality)**.** *Let $X_1, X_2, \ldots, X_t$ be pairwise independent random variables that takes value in the range $[0, 1]$. Let $X = \frac{\sum_i X_i}{t}$ and $\mu = \mathbb{E}[X]$, then*

$$\Pr[|X - \mu| \geq \varepsilon] \leq \frac{1}{t\varepsilon^2}$$

Now we are ready for the analysis of the confidence error.

- If $x \notin L$, then the algorithm runs to the end and rejects with probability 1.
- If $x \in L$, let $\delta(r_i)$ be the indicator variable defined as

$$\delta(r_i) = \begin{cases} 0, & \text{if } \mathcal{A}(r_i, x) = \text{“reject”} \\ 1, & \text{if } \mathcal{A}(r_i, x) = \text{“accept”} \end{cases}$$

  Let $Y = \sum_{i=1}^{2^{k+2}} \delta(r_i)/2^{k+2}$ be the average of all indicator variables. If any of the variable is 1, then the sampling algorithm outputs “accept”, which is correct. Therefore,

$$\text{Sampling algorithm outputs incorrect answer} \iff Y = 0$$

  The original algorithm $\mathcal{A}$ guarantees that $\mathbb{E}[\delta(r_i)] \geq 1/2$, so

$$\mathbb{E}[Y] \geq 1/2$$

  Using Pairwise Independent Tail Inequality, we have

$$\mathbb{E}[Y = 0] \leq \mathbb{E}[|Y - \mathbb{E}[Y]| \geq \mathbb{E}[Y]] \leq \frac{1}{2^{k+2}\mathbb{E}[Y]^2} = 2^{-k}$$

Therefore, it has the same confidence error as repeating the algorithm k times, with less random bits but slower running time.

2 Interactive Proof Systems

Interactive proof is a new type of proof system where the verifier can not only read the proof from the prover, but also interact/ask questions. Formally, we have a prover and a verifier, they both share an input string, and can communicate with each other. The verifier is allowed to use private random bits (unknown to the prover) and runs in polynomial time, while the prover has unbounded time and space. Without loss of generality, we assume that the prover is deterministic (this result is not trivial and the proof is outside of the scope).

Definition 3 (Goldwasser-Micali-Rackoff)**.** *An **Interactive Proof System** for a language L is a protocol with verifier V and prover P such that*

- *if $x \in L$ and both verifier and prover follow the protocol, $\Pr_{V\text{'s coins}}[\mathcal{A}(x) = \text{“accept”}] \geq 2/3$*
- *if $x \notin L$ and the verifier follows the protocol, $\Pr_{V\text{'s coins}}[\mathcal{A}(x) = \text{“reject”}] \geq 2/3$*

Definition 4. IP *is the class of languages that has an interactive proof system.*

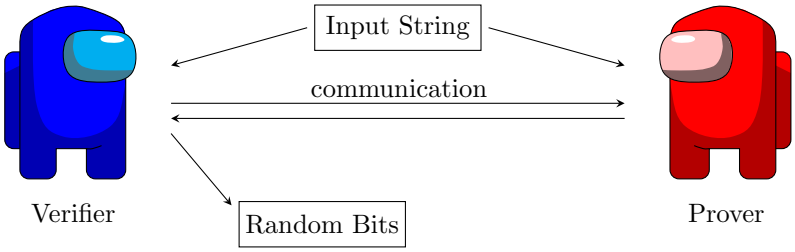

Figure 1: Interactive Proof Model

It is easy to see that **NP** $\subseteq IP$ since the problems in **NP** has a polynomial-time verifiable certificate, and the prover can sends it to the verifier without further interaction. Surprisingly, this proof system is so powerful that it is proven that **IP** = **PSPACE**. Here we give an example of an interactive proof system for graph non-isomorphism.

Graph isomorphism (GI) is a well-known problem in **NP** that it remains unknown whether GI $\in$ **P**, though we know that it is in quasi-**P**. In the complement of GI, called graph non-isomorphism ($\overline{\text{GI}}$), the prover needs to prove that two given graphs are not isomorphic. It's unknown whether $\overline{\text{GI}} \in$ **NP**, but we will show that $\overline{\text{GI}} \in$ **IP** by giving the following simple interactive proof system:

Given two graphs G_0, G_1,

1. The verifier picks a random bit $c \in \{0, 1\}$.
2. The verifier sends a random permutation $\pi \in S_n$ of the graph G_c
3. The prover tries to determine the bit c and sends its answer $c' \in \{0, 1\}$.
4. The verifier accepts if $c = c'$.

The above protocol is illustrated in the following Figure 2.

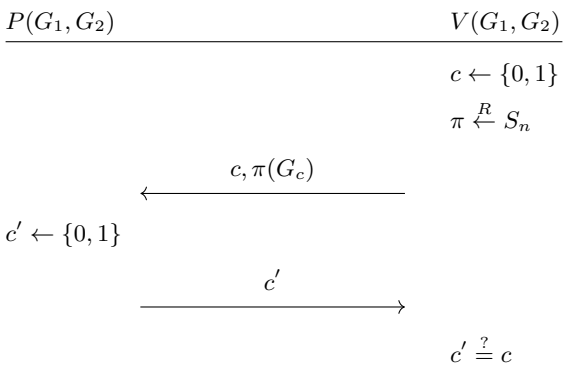

Figure 2: Interactive proof system of Graph Non-isomorphism

The above example illustrate why the verifier's random bits have to be private – otherwise the prover knows the value of c. However, it is proven that with clever transformation, every interactive proof can be transformed to a public-random-bit interactive proof.