0368.4163 Randomness and Computation March 18, 2009

Lecture 4

Lecturer: Ronitt Rubinfeld *Scribe: Itay Kirshenbaum, Oren Zomer, Margarita Vald*

1 Introduction

Today:

- Another application of hashing
  - But first:
    - Interactive proofs
    - Graph nonisomorphism
    - Public Vs. Private coins
  - Interactive proofs to lower bound set sizes
- Uniform generation of combinatorial objects

2 Interactive Proofs

2.1 Definitions

Definition 1. NP = All decision problems for which proof of "yes" answers can be verified by a polynomial time deterministic Turing Machine

Definition 2. *Interactive Proofs System (IPS)* [Goldwasser Micali Rackoff]

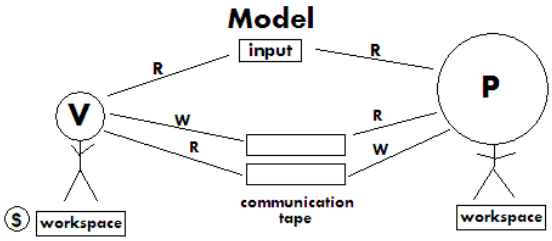

Figure 1: Interactive Proof System Protocol

An *Interactive Proofs System* for language $L \subseteq \{0,1\}^*$ is a protocol for $V(Verifier), P(Prover)$ such that:

- If V, P follow protocol and $x \in L$ then $\Pr_{\text{v's coins}}[\text{V accepts x}] \geq \frac{2}{3}$
- If V follows protocol and $x \notin L$ then $\Pr_{\text{v's coins}}[\text{V accepts x}] \leq \frac{1}{3}$

Definition 3. *Interactive Proofs (IP)* $= \{L | L \text{ has IPS}\}$

Definition 4. *Isomorphism*

Graphs $G = (V_G, E_G)$, $H = (V_H, E_H)$ are said to be isomorphic ($G \cong H$) if $V_H = \pi(V_G)$ such that $(u, v) \in E_G \Leftrightarrow (\pi(u), \pi(v)) \in E_H$

Definition 5. *Graph Isomorphism:* $L_{GI} = \{(G, H) | G \cong H\}$

Definition 6. *Graph Non-Isomorphism:* $L_{GNI} = \{(G, H) | G \ncong H\}$

2.2 Graph Non-Isomorphism

Theorem 7. *$GNI \in IP$ [Goldreich Micali Wigderson]*

Proof. Protocol:

- Input: Graphs G, H (P claims $G \ncong H$)
- V computes:
  - G' a random permutation of G
  - H' a random permutation of H
- Do twice:
  - V flips coin:
    - If Heads: V sends (G, G') to P
    - If Tails: V sends (G, H') to P
  - P sends V $\cong$ or $\ncong$
  - V's response:

V's flip	P's response	V's output
Heads	$\cong$	continue
Tails	$\ncong$	continue
Heads	$\ncong$	FAIL
Tails	$\cong$	FAIL

- Output "accept"

□

Proof. Correctness of Protocol:

- If $(G, H) \in L_{GNI}$, that is $G \ncong H$, unlimited time P can figure out the coin because $G \cong G'$ and $G \ncong H'$, and always answers correctly so $\Pr[V\text{accepts}] = 1 \geq \frac{2}{3}$

- If $(G, H) \notin L_{GNI}$, that is $G \cong H$, P may not follow the protocol, but he can do no better than random guessing (Since the distribution on V's messages is identical when flips Heads/Tails) - Can get V to continue with probability $\frac{1}{2}$ in each loop. More formally:
  - WLOG P is deterministic, that is a fixed function of $(G, \tilde{H})$
  - Some q fraction of permutations $\pi's$ cause $P(G, \pi(G))$ to answer $\ncong$, so (Heads, $\ncong$) will cause failure
  - $1-q$ fraction of permutations $\pi's$ cause $P(G, \pi(G))$ to answer $\cong$, so (Tails, $\cong$) will cause failure
  - H' is a uniformly distributed random permutation of H and $H \cong G$, therefore H' is a uniformly distributed random permutation of G. G', by definition, is a uniformly distributed random permutation of G. Therefore $\tilde{H}$ will be a uniformly distributed random permutation of G.
  - Pr[V fails] $= \frac{1}{2}q + \frac{1}{2}(1-q) = \frac{1}{2}$
  - Enough to run the loop twice to get Pr[Vaccepts] $= \frac{1}{4} \leq \frac{1}{3}$

□

3 Public Coins vs. Private Coins

A generalization of the Interactive Proof model is in making the outcome of the coins that the verifier tosses *public*. This small change leads to a model of Proofs called Arthur-Merlin games or "IP public". It is known that the "public" proof model, i.e. Arthur-Merlin games, and the "private" proof model are equivalent.(Sipser and Goldwasser) and anything you can do in IP without access to random bits you can do with access to random bits.

In Arthur-Merlin games the idea is that Arthur (the verifier) flips random coins at each round, but unlike the private coins model, the coins are visible to Merlin (the prover) as well. Merlin then tries to persuade Arthur of the answer. Note that since Merlin is all-powerful, he does not need to see anything but the coin-tosses of Arthur, because once Merlin knows the coin tosses, he can compute what Arthur is going to ask.

Definition 8. A *round* is a communication round, where messages are sent from the prover to the verifier and vice versa in one direction only.

For example IP protocol for GNI is three rounds protocol.

The behavior of the Verifier V (Arthur) can then be described as follows:

if all k rounds are done V decides whether to accept or reject

if less then k rounds are done, V outputs some random bits

This leads to complexity classes $AM(k)$ and $MA(k)$, where k is the number of rounds and the order of letters A and M is determined by who (Arthur or Merlin) starts the conversation.

3.1 Graph Automorphisms

Definition 9. An *automorphism* of a graph $G = (V, E)$ is a permutation π of the vertex set V, such that for any edge $e = (u, v), \pi(e) = (\pi(u), \pi(v))$ is also an edge in E. (That is, it is a graph isomorphism from G to itself). The identity mapping of a graph onto itself is called the *trivial automorphism* of the graph (See for example figures 2 and 3).

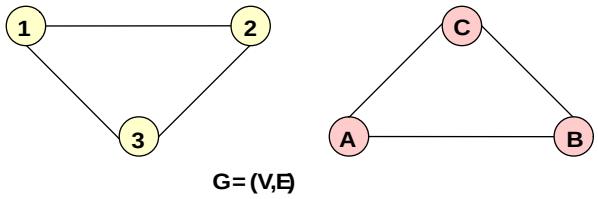

Figure 2: G has six automorphisms $1 \to A, 2 \to B, 3 \to C; 1 \to B, 2 \to A, 3 \to C$ etc...

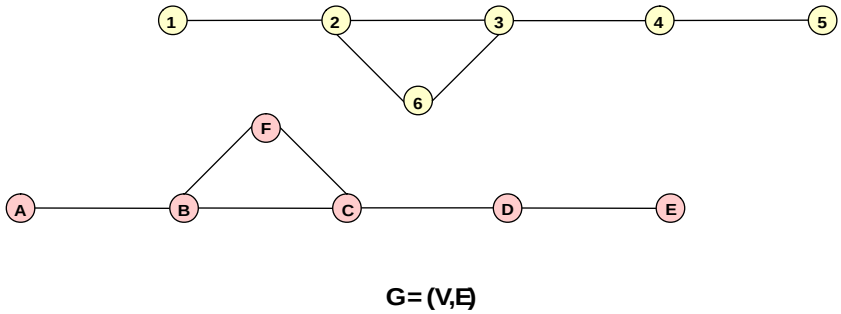

Figure 3: G has only the trivial automorphism $1 \to A, 2 \to B, 3 \to C, 4 \to D, 5 \to E, 6 \to F$

We're about to prove that $GNI \in IP - Public$ but we'll show it only for a special case of graph-nonisomorphism. In this special case, we prove the non-isomorphism of two connected graphs, A and B, each with only the trivial automorphism. Let n denote the number of nodes in both A and B. If they were different, the proof of non-isomorphism would be trivial.

If G is a graph, then let $[G]$ denote the set of graphs isomorphic to G. All of the graphs in $[G]$ have the same set of vertices, and all are obtained by permuting the vertices of G. However, a pair of vertices might be adjacent in one graph in $[G]$, but not in another.

For a graph G, the size of $[G]$ is $n!/|Aut(G)|$ where n is the number of vertices in G and $|Aut(G)|$ is the number of automorphisms of G. Since the graphs A and B have only one, trivial automorphism, $|[A]| = |[B]| = n!$ (See figure 4)

Define U to be the set of graphs obtained by uniting $[A]$ and $[B]$. As consequences, we obtain two fact that we will use extensively:

- If A and B are not isomorphic, then $|U| = |[A] \cup [B]| = 2n!$ and we refer it as **"large"**.

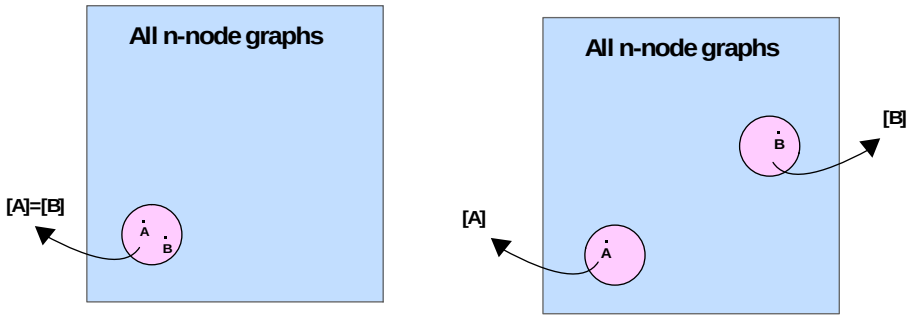

Figure 4: The set of all permutation on A and B. The graph on the right shows the case where A and B are not isomorphic. On the other hand the graph on the left shows the case where A and B are isomorphic.

- If A and B are isomorphic, then $|U| = |[A] \cup [B]| = n!$ and we refer it as **"small"**.

3.2 IP-Public Protocol

Theorem 10. *$GNI \in IP - Public$*

The overall idea is that the prover convinces the verifier that $|U| = |[A] \cup [B]|$ is big— more like $2n!$ than $n!$. By the preceding facts, this implies that A and B are not isomorphic.

The present problem is to determine whether a set is "a tiny fraction" or "half a tiny fraction" of a universe. In this case, the universe D_n is defined to be the set of all n node graphs and let m be the number of bits needed to describe a n-node graph ($m = n^2$). It is easy to see that $[A] \cup [B]$ is at most "a tiny fraction" of this universe D_n.

To see why this is a problem, consider the following procedure for proving graph non-isomorphism.

1. The verifier V uses public coins to generate a random n node graph $G \in D_n$ and send it to V.

2. If $G \in [A] \cup [B] = U$, then the prover P returns a proof of this fact in the form of a mapping between the vertices of G and the vertices of A or B.

3. The verifier looks at how often the prover returns a valid proof. If A and B are not isomorphic, then $[A] \cup [B]$ is twice as big as if they were isomorphic. This means that the prover can return a valid proof twice as often. The verifier detects this and concludes that A and B are not isomorphic. Recall that $\frac{\#successes}{\#tries} \leq \frac{|U|}{\#graphs}$

Note that $\frac{|U|}{\#graphs}$ is exponentially small. Hence, the flaw in this procedure is that a random graph $G \in D_n$ is so rarely in $U = [A] \cup [B]$— whether A and B are isomorphic or not— that the number of rounds must be exponential for the verifier to reach any conclusion and therefore, we won't be able to estimate it in polytime. As an exercise, look at the Chernoff bounds and make sure you see how the number of required samples depends on $\frac{|U|}{\#graphs}$.

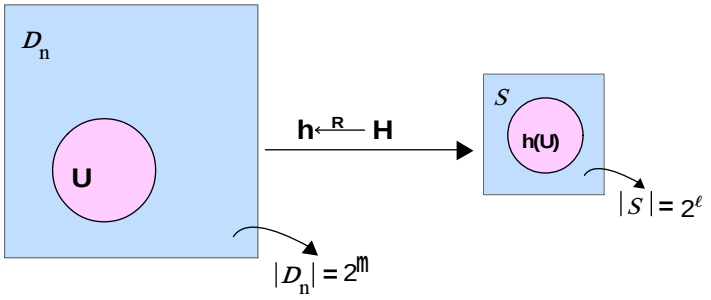

Figure 5: Hashing m bit strings into ℓ bit strings using randomly chosen pairwise independent function h.

The solution is to use hashing. The set $U = [A] \cup [B]$ is a tiny fraction of D_n. But now we cleverly hash D_n down to a small set S. It will turn out that if A and B are not isomorphic, then the image of $[A] \cup [B]$ under the hash function will be a big fraction of S. If A and B are isomorphic, then the image of $[A] \cup [B]$ will be only a small fraction of S. In this way, hashing from D_n to S converts an "tiny fraction" vs. "half a tiny fraction" problem in D_n into a "big fraction" vs. "small fraction" problem in S (See figure 5).

To obtain the objective we need h to have the following properties:

1. $|h(U)| \approx |U|$ i.e $h(U)$ is big iff U is big
2. $\frac{|h(U)|}{2^l}$ is $\geq \frac{1}{poly(n)}$
3. h computable in polytime (Since the verifier is going to have to compute it).

3.3 Universal Hashing

Definition 11. A family of functions $H = \left\{h : \{0,1\}^m \to \{0,1\}^\ell\right\}$ is called *pairwise independent* if for any $\alpha \neq \beta \in \{0,1\}^m$ and for any $\gamma, \delta \in \{0,1\}^\ell$:

$$Pr_h\left[h(\alpha) = \gamma \wedge h(\beta) = \delta\right] = 2^{-2\ell}$$

Note that immediately from the definition it also follows that for any α, γ (and some $\beta \neq \alpha$):

$$Pr_h\left[h(\alpha) = \gamma\right] = \sum_{\delta} Pr_h\left[h(\alpha) = \gamma \wedge h(\beta) = \delta\right] = 2^{\ell} \cdot 2^{-2\ell} = 2^{-\ell}$$

Remark 12. We'll be using the terms "universal hash functions" and "pairwise independent hash functions" interchangeably.

The family of all functions from $\{0,1\}^m$ to $\{0,1\}^\ell$ is universal, but is so large that even specifying a particular element requires excessive space. Fortunately, more manageable universal hash families exist. For example, the family $\{h_{M,b}(x) = Mx + b\}_{M \in M_{\ell \times m},\, b \in \{0,1\}^\ell}$ where $M_{\ell \times m}$ is the set of ℓ

by m binary matrices, and the arithmetic is done in $GF(2)$, is pairwise independent and form a universal hash family. In addition, in a previous lecture we saw a construction of pairwise independent bits that would work here as well.

Now we are ready to present an interactive proof protocol with public coins for graph-nonisomorphism. Call the verifier V and the prover$P$.

Let $H = \left\{h : \{0,1\}^m \to \{0,1\}^\ell\right\}$ be a family of pairwise independent functions. Let D_n be the set of all n node graphs, encoded using m bits. Let U be the subset of D_n corresponding to graphs in $[A] \cup [B]$. Let S be the set of 0-1 column vectors of size ℓ where ℓ will be set later.

1. V picks $h \in_R H$ and send h to the P
2. P sends to V an element $x \in U$ s.t $h(x) = 0^\ell$ and a proof that $x \in U$, where the proof is an automorphism from A or B to x.
3. V checks that $h(x) = 0^\ell$ and checks P proofs that $x \in U$. If x passes both checks, then V accepts.

Note that if U is "big" then P would be able to supply an appropriate x frequently and convince V. But, if U is "small" P would be able to convince V less often.

The following lemma is the tool we use to convert a "tiny fraction" vs. "half a tiny fraction" problem in D_n into a "big fraction" vs. "small fraction" problem in S.

Lemma 13. Let H be a Pairwise independent family of hash functions from $\{0,1\}^m$ to $\{0,1\}^\ell$. Let U be a subset of $\{0,1\}^m$ and define $\alpha = |U|/2^\ell$. Then

$$\alpha - \frac{\alpha^2}{2} \leq Pr_h\left[0^\ell \in h(U)\right] \leq \alpha$$

Proof. The right hand side is easy.

$$Pr_h\left[0^\ell \in h(U)\right] \leq \sum_{x \in U} Pr_h\left[0^\ell = h(x)\right] = |U|/2^\ell = \alpha$$

The left hand side is more tricky.

$$\begin{aligned} Pr_h\left[0^\ell \in h(U)\right] &\geq \sum_{x \in U} Pr_h\left[0^\ell = h(x)\right] - \sum_{\substack{x,y \in U \\ x \neq y}} Pr_h\left[0^\ell = h(x) = h(y)\right] \\ &= \alpha - \frac{\binom{|U|}{2}}{\left(2^\ell\right)^2} \\ &\geq \alpha - \frac{|U|^2}{2} \cdot \frac{1}{2^\ell} \\ &= \alpha - \frac{\alpha^2}{2} \end{aligned}$$

The first line uses an inclusion-exclusion principle truncated after two terms. The second line follows from the definition of universal hashing. The final line follows from the definition of α. □

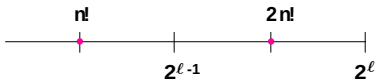

Figure 6: The setting of the parameter ℓ

3.3.1 Choice of ℓ

Now all we need to do is to choose ℓ. We pick ℓ s.t $2^{\ell-1} \leq 2n! \leq 2^{\ell}$ (See figure 6).

First, suppose that A and B are not isomorphic. Then $|U| = |[A] \cup [B]| = 2n!$, and $1 \geq \alpha \geq \frac{1}{2}$. Lemma 13 says that

$$Pr_h\left[0^{\ell} \in h\left(U\right)\right] \geq \alpha - \frac{\alpha^2}{2} \geq \frac{3}{8} = c_1$$

Now, suppose that A and B are isomorphic. Then $|U| = |[A] \cup [B]| = n!$, and $\alpha = |U|/2^{\ell} \leq \frac{1}{2}$. Therefor,

$$Pr_h[V \; accepts] \leq \alpha \leq \frac{1}{2} = c_2$$

The proof is not completely correct since we need $c_1 > c_2$. The solution to this problem is to run the protocol above three times and accept iff V accepted in all the three iterations.

3.4 General Case Idea

Theorem 14 (Goldwasser-Sipser)**.** *IP with k rounds $\subseteq$ IP-public with $k+3$ rounds.*

We will restrict ourselves to $k = 2$. Here is our 2-round interactive proof. Let ℓ be our poly, and let w be the word we are reasoning about. As usual, we denote the verifier by V and the prover by P . Let r be $V's$ random bits.

1. V gets a random string $r \in \{0,1\}^{\ell(|w|)}$. V uses r to produce a message q ("q" for "query") and sent q to P.
2. P answers with a response a; P should answer with a that causes acceptance on most random strings that produce q.
3. V checks P's answer and accepts or rejects it.

Assume $|q| = |a| = |r| = \ell$.

V may be pictured as a map from random strings r to questions q. Any interesting IP protocol has V not injective, because otherwise P knows what r V started with from q. We can divide the r into two types: those which, with a, cause acceptance, and those which don't.

- Let $R(q) = \{r \in \{0,1\}^{\ell(|w|)} \text{ s.t. } V(w,r) = q\}$
- Let a_q be chosen to maximize $Pr_r[V(w,r,q,a_q) \text{ accepts}]$ s.t. $V(w,r) = q$
- Let $A(q) = \left\{r \in \{0,1\}^{\ell(|w|)} \middle| \, V(w,r) = q \wedge V(w,r,q,a_q) \; accepts\right\}$. $A(q)$ are the random strings where P has a good probability to convince V (See figure 7).X

- Let $A = \underset{q}{\cup} A(q)$

and therefore $Pr\left[V\ accepts\right] = \frac{|A|}{2^{\ell(|w|)}}$.

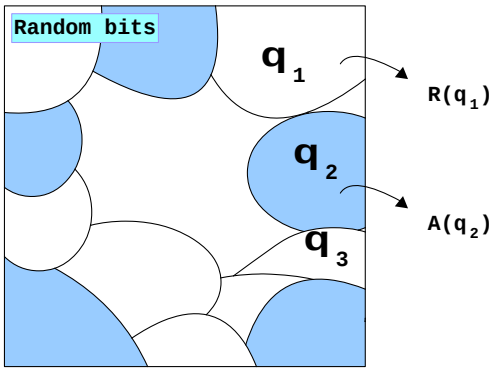

Figure 7: The partition of random bits into sets which ask the same question. The shaded sets are where P can make V to accept with good probability.

Note that where x is in the language $|A|$ is big and where x is not in the language $|A|$ is small.

3.5 MA(5) Simulation

In the MA(5) protocol which solves whatever language the previous IP(2) protocol did, P will attempt to show that $|A|$ is big. To do this, P will demonstrate that for some c and d, there are at least 2^c q's s.t., $|A(q)| \geq 2^d$, which will imply $|A| \geq 2^{c+d}$. If $|A| > 2^{l-1}$, then $\exists c, d$ s.t. this holds, where $2^{c+d} \geq \frac{2^{l-1}}{2l}$ (this can easily be verified - simply show the converse is not true by considering the l possibilities for c.) This requires performing the lower bound protocol on the number of random strings causing V to accept.

The rest of the proof is beyond the scope of the course.

4 Disjunctive Normal Form

A logical formula is considered to be in DNF if and only if it is a disjunction of one or more conjunctions of one or more literals (variables and their negation), such as:

$$F = x_1\bar{x_2}x_3 \vee \bar{x_1}x_2x_4x_5 \vee \ldots$$

In a later lecture we will look for algorithms that can output α such that

$$\frac{\#\text{satisfying assignments}}{(1+\epsilon)} \leq \alpha \leq (1+\epsilon)\cdot \#\text{satisfying assignments}$$

Every conjunction of k different literals (that do not contain a variable and its negation) is satisfied by 2^{n-k} assignments, where n is the total number of variables. The size of the union, however,

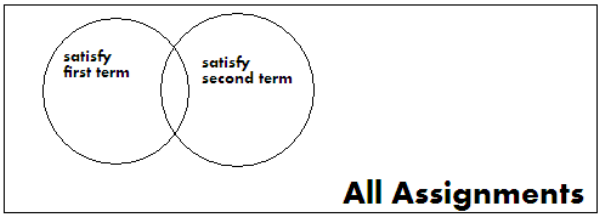

Figure 8: Satisfying Assignments of 2 Terms

could be less than the sum of sizes. For example, in the case of 2 clauses: $F = x_{i_1} \dots \bar{x_{i_k}} \vee x_{j_1} \dots \bar{x_{j_k}}$ we can get:

The size of the union is the sum of sizes each term's satisfying assignments minus the size of the intersection.