Lecture 5

- Random bits for Interactive Proofs
- IP public vs. private coins
- IP protocol for lower bounding a set size
- Derandomizing via method of conditional expectations
Arthur–Merlin Games

V's random tape is public!

⇒ this protocol breaks

Can Graph have IPs with only public coins?

YES! [Goldwasser Sipser]

(important for complexity, crypto, interesting tool for checking delegated computations...)

How do they show this?

First, a notation:

\[ [A] \equiv \text{graphs } \equiv \text{to } A \]

+ an assumption:

Assume \( A, B \) graphs with no "nontrivial automorphisms"

\[ \uparrow \equiv \text{not } \equiv \text{to self under relabeling} \]

then \( |[A]| = |[B]| = |V|! \)

Why useful? let \( U \leftarrow [A] \cup [B] \)

A \( \equiv B \) \hspace{1cm} A \neq B

All \( M \)-node graphs \hspace{1cm} A \neq B

\[ |[A]| = |[B]| \quad |U| = |V|! \]

"small" \hspace{1cm} "big"

Goal: IP for proving a set is large
First idea: Random Sampling?

Repeat \( \tilde{O}(d) \) times:

1. \( V \to P \): random \( 1/\tilde{O}(d) \)-node graph \( g \)
2. \( P \to V \): if \( g \in U \), a proof that it is a "success"
   - i.e., show \( l \) to \( A \) or \( B \)

Finally, \( V \) outputs \( \frac{\# \text{successes}}{\text{total \# loops}} \)

How many loops needed? \( \Omega \left( \frac{|U|}{|1/\tilde{O}(d)\text{-node graphs}|} \right) \) just to hit one success

Problem: \( |U| \) is very small compared to \( |1/\tilde{O}(d)\text{-node graphs}| \)

\( \Rightarrow \) need many loops

Fix: Universal Hashing

- \( m \) bits used to describe graph \( m \approx O(d^n) \)
- \( m \approx O(1/\tilde{O}(d)) \)
- Sample randomly here and estimate \( \frac{|h(u)|}{2^d} \)

need:

1. \( |h(u)| \approx |u| \)
2. \( |h(u)| \) is big if \( |u| \) big
3. \( |h(u)| \approx \frac{1}{\text{poly}(m)} \) (in our case, constant)
4. \( h \) computable in poly time
Protocol:

Given $H$, collection of pi. Pems mapping $\mathbb{S}_0 \mathbb{S}_1^m \rightarrow \mathbb{S}_0 \mathbb{S}_1^l$

1. $V$ picks $h \in_k H$
2. $V \rightarrow P: h$
3. $P \rightarrow V: x \in U \text{ st. } h(x) \in 0^l$

with proof that $x \in U$

Idea

$u$ big (i.e., $2^{lu!}$!): $h(u)$ usually hits $0^m$ so $P$ can usually do it

$u$ small (i.e., $lu!$!): $h(u)$ usually doesn't hit $0^m$ so $P$ usually can't do it

How?

map $u$ to range of size $\approx 2^l$!

if $u$ big, it "fills" the range

$\Rightarrow$ probably hits $"0"$

if $u$ small, it only hits part of the range

$\Rightarrow$ less chance of hitting $"0"$

Recall $H$ is pi., if $\forall x, y \in \mathbb{S}_0 \mathbb{S}_1^m$ and $a, b \in \mathbb{S}_0 \mathbb{S}_1^l$

$$\Pr_h [h(x) = a \land h(y) = b] = 2^{-2l}$$

Lemma

$H$ pi., $u \in \mathbb{S}_0 \mathbb{S}_1^m$, $a = \frac{lu}{2}$

Then $a - \frac{a^2}{2} \leq \Pr_h [0^l \in h(u)] \leq a$
Pf.

\[ \text{RHS: } \forall x \Pr_h[0^l = h(x)] = 2^{-l} \quad (\text{since } h \text{ is p.a.}) \]

\[ \text{so } \Pr_h[0^l \in h(w)] \leq \sum_{x \in U} \Pr[0^l = h(x)] = \frac{|U|}{2^l} = a \]

\[ \uparrow \text{ union bd} \]

\[ \text{LHS: use inclusion-exclusion bd: } \]

\[ \Pr[I \cup A_i] = \sum \Pr[A_i] - \sum \Pr[A_i \cap A_j] \]

\[ \Pr_h[0^l \in h(w)] \geq \sum_{x \in U} \Pr[0^l = h(x)] - \sum_{x \neq y} \Pr[0^l = h(x) = h(y)] \]

\[ = \frac{|U|}{2^l} - \left( \frac{1}{2} \right) \frac{1}{2^l} \geq \frac{|U|}{2^l} - \frac{|U|^2}{2^l} \cdot \frac{1}{2} \geq a - \frac{a^2}{2} \]

\[ \text{Finishing up?} \]

pick \( l \) s.t. \( 2^{l-1} \leq 2|V|! \leq 2^l \)

\( l \Rightarrow |U| = 2|V|! \)

\[ \frac{1}{2} \leq a \leq 1 \]

so \( \Pr[V \text{ accepts}] \geq a - \frac{a^2}{2} \geq \frac{3}{8} = \alpha \)

\( \Downarrow \Rightarrow |U| = |V|! \)

\[ \frac{1}{4} \leq a \leq \frac{1}{2} \]

so \( \Pr[V \text{ accepts}] \leq \frac{1}{2} = \beta \)

\text{Whoops! need } \alpha > \beta \]

\text{solution: fix }
Idea for general Thm:

\[ 1^P_{\text{private coins}} = 1^P_{\text{public coins}} \]

argue that i.e. protocol can be used to show that size of accepting region probability mass is large.

(need that am verify a conversation/ random coin to be in accept region)
More derandomization: The method of conditional expectations

Idea: view coin tosses of algorithm as path down a tree of depth $m \Rightarrow$ $m$ coin tosses

$\delta = H$
$1 = T$

$\text{good} = \text{correct/randomized/Pass...}$

good bad bad bad

good good

good randomized algorithm $\iff$ most leaves are good

Idea find a good path to leaf "bit-by-bit"

more formally:

Fix randomized algorithm $A$
input $x$
m = # random bits used by $A$ on $x$

for $1 \leq i \leq m$ + $r_1 \ldots r_i \in \{0,1\}$, let $p(r_1 \ldots r_i) = \text{fraction of continuations that end in "good" leaf}$

$$p(r_1 \ldots r_i) = \frac{1}{2} \cdot p(r_1 \ldots r_i, 0) + \frac{1}{2} \cdot p(r_1 \ldots r_i, 1)$$

by averaging, $\delta$ setting of $r_i$ to 0 or 1

$s.t. \ p(r_1 \ldots r_i) \approx p(r_1 \ldots r_i) \:\:\:\\ \text{can we figure this out?}$
if \( p(r_1 \ldots r_n) \geq p(r_{i} \ldots r) \) \( \forall i \)

then \( p(r_1 \ldots r_n) \geq p(r_{i} \ldots r_{m-1}) \geq \ldots \geq p(r_i) = p(\Lambda) \geq 2/3 \)

\[ \uparrow \]

this is a leaf
so value is 1 or 0;
but if \( \geq 2/3 \)
must be 1

\[ \uparrow \]

fraction
of good
paths

main issue: how do we figure out the best setting of \( r_{\text{ah}} \) at
each step?

An example: Max Cut  (another way to derandomize)

recall algorithm:
flip \( n \) coins \( r_1 \ldots r_n \)
put node \( i \) in \( S \) if \( r_i = 0 \) + \( T \) if \( r_i = 1 \)
output \( S, T \)

derandomization:
\[
e(r_1 \ldots r_n) = E_{R_{\text{ah}} \ldots R_N} \left[ 1 \text{ cut}(S,T) \right] \text{ given } r_1 \ldots r_n \text{ choices made}
\]
\[
e(\Lambda) = \frac{|E|}{2} \quad \text{(from previous lecture)}
\]

how do we calculate \( e(r_1 \ldots r_n) \)?
Let
\[ S_{i+1} = \{ j \mid j \leq i+1 \} \quad r_j = 0 \beta \]
\[ T_{i+1} = \{ j \mid j \leq i+1 \} \quad r_j = 13 \]
\[ V_{i+1} = \{ j \mid j \geq i+2 \} \quad \beta \]

so
\[ e(r_1 \ldots r_{i+1}) = \left( \# \text{ edges between } S_{i+1} \cup T_{i+1} \right) + \frac{1}{2} \left( \# \text{ edges touching } V_{i+1} \right) \]

Note: don't need to calculate \( e(r_1 \ldots r_{i+1}) \)
just need to compare \( e(r_1 \ldots r_{i+2}) \) vs. \( e(r_1 \ldots r_{i+1}) \) - is it \( e_{i+2} = 53 \)

Note:
- \( V_{i+1} \) term is same for both
- first term differs only on edges adjacent to node \( i+1 \)

- to maximize this, place node \( i+1 \)
to maximize cut size

i.e. \( \# \text{ edges between node } i+1 \cup S_i \)

vs. \( \# \text{ " " " " " } + T_i \)
Corresponds to:

**Greedy Algorithm:**

1) $S \leftarrow \emptyset, T \leftarrow \emptyset$

2) For $i = 0 \ldots N-1$
   
   place node $i$ in $S$ if $\# \text{edges between } i + T \geq \# \text{edges} \quad i + S$

   else place in $T$