Sp2014

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!

$\Rightarrow$ this protocol breaks

Can Graph $\not\cong$ 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:

$[A]$ = graphs $\cong$ to A

+ an assumption:

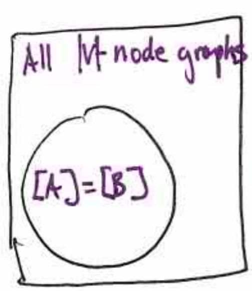
Assume A, B graphs with no "nontrivial automorphisms"

i.e. not $\cong$ to self under relabeling

then $|[A]| = |[B]| = |V|!$

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

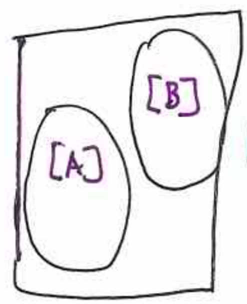
$A \cong B$



$|U| = |V|!$

"small"

$A \not\cong B$



$|U| = 2|V|!$

"big"

Goal: IP for proving a set is large

First idea: Random Sampling?

Repeat **?** times:

$V \to P$: random $|V|$-node graph g

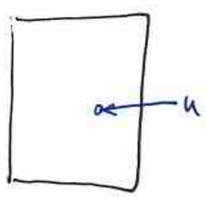
$P \to V$: if $g \in \mathcal{U}$, a proof that it is a "success" (i.e. show $\cong$ to A or B)

else nothing

Finally, V outputs $\dfrac{\#\text{successes}}{\text{total } \# \text{ loops}}$

How many loops needed? $\Omega\left(\dfrac{|\mathcal{U}|}{\# |V|\text{-node graphs}}\right)$ just to hit one success

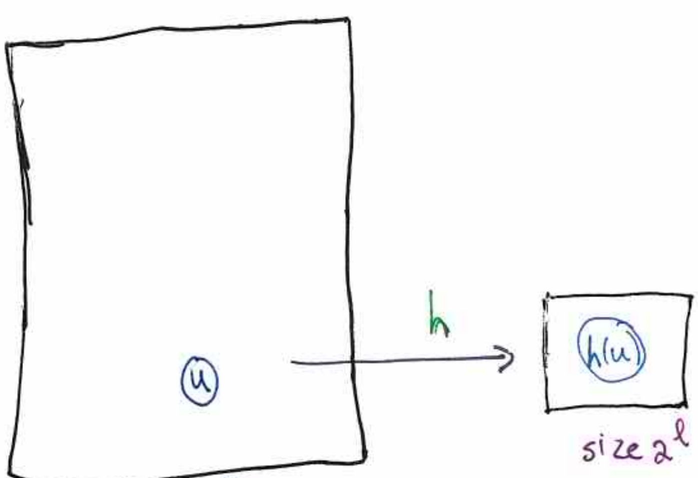
Problem: $|\mathcal{U}|$ is very small compared to # $|V|$-node graphs



$\Rightarrow$ need many loops

$|h(\mathcal{U})| \le |\mathcal{U}|$ is obvious! but also not much smaller

Fix: Universal Hashing



size 2^m

size 2^ℓ

m bits used to describe graph
$m \approx O(|V|^2)$

Sample randomly here + estimate $\dfrac{|h(\mathcal{U})|}{2^\ell}$

need:

1. $|h(\mathcal{U})| \approx |\mathcal{U}|$

   $h(\mathcal{U})$ big iff $|\mathcal{U}|$ big

2. $\dfrac{|h(\mathcal{U})|}{2^\ell}$ is $\dfrac{1}{\text{poly}(m)}$

   (in our case, constant)

3. h computable in poly time

Protocol :

given H, collection of p.i. fctns mapping $\{0,1\}^m \to \{0,1\}^\ell$

1. V picks $h \in_R H$
2. $V \to P$: h
3. $P \to V$: $x \in U$ s.t. $h(x) \in 0^\ell$
   with proof that $x \in U$

idea
U big (i.e. $2|V|!$) : $h(U)$ usually hits 0^m so P can usually do it
U small (i.e. $|V|!$) : $h(U)$ usually doesn't hit 0^m so P usually can't do it

how?
map U to range of size $\approx 2V!$
if U big, it "fills" the range
& probably hits "0"
if U small, it only hits part of the range
$\Rightarrow$ less chance of hitting "0"

Recall H is p.i. if $\forall x, y \in \{0,1\}^m$ $\forall a, b \in \{0,1\}^\ell$

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

Lemma H p.i., $U \subseteq \Sigma^m$ $\quad a = \frac{|U|}{2^\ell}$ ← would be fraction if h maps U 1-1

then $a - \frac{a^2}{2} \le \Pr_h[0^\ell \in h(U)] \le a$

Pf.

RHS:

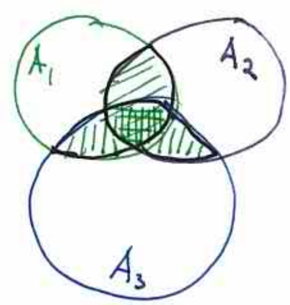
$\forall x \quad Pr_h[0^\ell = h(x)] = 2^{-\ell}$ (since H is p.i.)

so $Pr_h[0^\ell \in h(U)] \le \sum_{x \in U} Pr[0^\ell = h(x)] = \frac{|U|}{2^\ell} = a$

↑ union bnd

LHS: use inclusion-exclusion bnd:

$$Pr[\cup A_i] \ge \sum_i Pr[A_i] - \sum_{i \ne j} Pr[A_i \cap A_j]$$

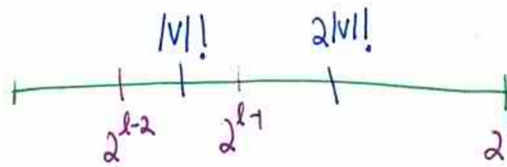


$$Pr_h[0^\ell \in h(U)] \ge \sum_{x \in U} \underbrace{Pr[0^\ell \in h(x)]}_{2^{-\ell}} - \sum_{\substack{x,y \in U \\ x \ne y}} \underbrace{Pr[0^\ell = h(x) = h(y)]}_{2^{-2\ell}}$$

$$= \frac{|U|}{2^\ell} - \binom{|U|}{2}\frac{1}{2^{2\ell}} \ge \frac{|U|}{2^\ell} - \frac{|U|^2}{2} \cdot \frac{1}{2^{2\ell}} \ge a - \frac{a^2}{2} \quad \blacksquare$$

Finishing up?

pick ℓ s.t. $2^{\ell-1} \le 2|V|! \le 2^\ell$



$\not\cong \Rightarrow |U| = 2|V|!$

$\frac{1}{2} \le a \le 1$

so $Pr[V \text{ accepts}] \ge a - \frac{a^2}{2} \ge 3/8 = \alpha$

$\cong \Rightarrow |U| = |V|!$

$\frac{1}{4} \le a \le 1/2$

so $Pr[V \text{ accepts}] \le 1/2 = \beta$

whoops!
need
$\alpha > \beta$
solution: HW

Idea for general Thm:

i.e. $IP_{\substack{\text{private}\\\text{coins}}} = IP_{\substack{\text{public}\\\text{coins}}}$

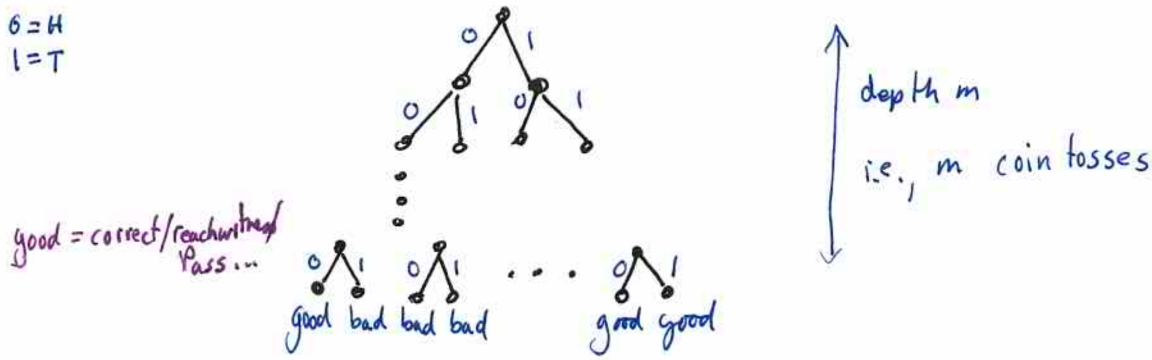
argue that l.b. protocol can be used to show that size of accepting region probability mass is large.

(need that can 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 ⟸ # coin tosses

0 = H
1 = T



good randomized algorithm ⟺ 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 $1 \le i \le m$ & $r_1 \cdots r_i \in \{0,1\}$, let $p(r_1 \cdots r_i)$ = fraction of continuations that end in "good" leaf

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

$$= \Pr_{R_{i+1} \cdots R_m}[A(x; r_1 \cdots r_i R_{i+1} \cdots R_m) \text{ correct}]$$

$$= \frac{1}{2}\left(\Pr_{R_{i+2} \cdots R_m}[A(x; r_1 \cdots r_i\, 0\, R_{i+2} \cdots R_m) \text{ correct}]\right) + \frac{1}{2}\left(\Pr_{R_{i+2} \cdots R_m}[A(x; r_1 \cdots r_i\, 1\, R_{i+2} \cdots R_m) \text{ correct}]\right)$$

by averaging, $\exists$ setting of r_{i+1} to 0 or 1 s.t. $p(r_1 \cdots r_{i+1}) \ge p(r_1 \cdots r_i)$ Can we figure this out?

if $p(r_1 \cdots r_{i+1}) \geq p(r_1 \cdots r_i) \quad \forall i$

then $p(r_1 \cdots r_m) \geq p(r_1 \cdots r_{m-1}) \geq \cdots \geq pr(r_1) \geq pr(\Lambda) \geq 2/3$

this is a leaf
so value is 1 or 0,
but if $\geq 2/3$
must be 1

fraction
of good paths

main issue: how do we figure out the best setting of r_{i+1} at each step?

An example: Max Cut (another way to derandomize)

recall algorithm:

flip n coins $r_1 \cdots r_n$
put node i in S if $r_i = 0$ & T if $r_i = 1$
output S, T

derandomization:

$$e(r_1 \cdots r_i) = E_{R_{i+1} \cdots R_N}[\,|\,cut(S,T)\,|\ \text{given } r_1 \cdots r_i \text{ choices made}]$$

$$e(\Lambda) \geq \frac{|E|}{2} \quad \text{(from previous lecture)}$$

how do we calculate $e(r_1 \cdots r_{i+1})$?

Let

$$S_{i+1} = \{\overset{\text{nodes}}{j} \mid j \le i+1 \quad r_j = 0\}$$
$$T_{i+1} = \{\overset{\text{nodes}}{j} \mid j \le i+1 \quad r_j = 1\}$$

} S + T <u>so far</u>

$$U_{i+1} = \{\overset{\text{nodes}}{j} \mid j \ge i+2 \;+\; j \le n\}$$

"Undecided"

so

$$e(r_1 \cdots r_{i+1}) = (\#\text{ edges between } S_{i+1} + T_{i+1}) + \tfrac{1}{2}(\#\text{edges touching } U_{i+1})$$

Note: don't need to calculate $e(r_1 \cdots r_{i+1})$

just need to compare $e(r_1 \cdots r_i\, 0)$ vs. $e(r_1 \cdots r_i\, 1)$ – is it $\{<, >, =\}$?

note:

- U_{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. $|\#$ edges between node $i+1$ + $S_i|$

vs. $|$" " " " + $T_i|$

Corresponds to:

<u>Greedy Algorithm</u>:

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

2) For $i = 0 .. N-1$

place node i in S if #edges between i & T $\geq$ # edges " i & S

else place in T