Computational Complexity Theory

Lecture 12: Perfect matching is in RNC; Class BPL; GNI is in BP.NP

Department of Computer Science,
Indian Institute of Science

Randomness brings in simplicity

- The use of randomness helps in designing *simple* and efficient algorithms for many problems.
- We'll see one such algorithm in this lecture, namely an efficient randomized, parallel algorithm to check if a given bipartite graph has a perfect matching.

Class RNC

- The use of randomness helps in designing *simple* and efficient algorithms for many problems.
- We'll see one such algorithm in this lecture, namely an efficient randomized, parallel algorithm to check if a given bipartite graph has a perfect matching.
- Definition. A language L is in RNC^i if there's a *randomized* $O((\log n)^i)$-time parallel algorithm M that uses $n^{O(1)}$ parallel processors s.t. for every $x \in \{0,1\}^*$,

  $x \in L \Rightarrow \Pr[M(x) = 1] \geq 2/3$,

  $x \notin L \Rightarrow \Pr[M(x) = 0] = 1$.

  Here, n is the input length.

Class RNC

- The use of randomness helps in designing *simple* and efficient algorithms for many problems.
- We'll see one such algorithm in this lecture, namely an efficient randomized, parallel algorithm to check if a given bipartite graph has a perfect matching.

- Definition. $RNC = \bigcup_{i>0} RNC^i$.

- RNC stands for randomized NC. Like NC, we can alternatively define RNC using (uniform) circuits.

Perfect matching in RNC

- Let PerfectMatching = {Bipartite graph G : G has a perfect matching}.
- Theorem. *(Lovasz 1979)* PerfectMatching $\in$ RNC^2.
- The input G = (L$\dot\cup$R, E) is given as a n x n *biadjacency matrix* $A = (a_{ij})_{i,j \in n}$, where n = |L| = |R|.

Perfect matching in RNC

- Let PerfectMatching = {Bipartite graph G : G has a perfect matching}.
- Theorem. *(Lovasz 1979)* PerfectMatching $\in RNC^2$.
- The input $G = (L \dot\cup R, E)$ is given as a $n \times n$ *biadjacency matrix* $A = (a_{ij})_{i,j \in n}$, where $n = |L| = |R|$.

$a_{ij} = 1$ if there's an edge from the i-th vertex in L to the j-th vertex in R, otherwise $a_{ij} = 0$.

Perfect matching in RNC

- Let PerfectMatching = {Bipartite graph G : G has a perfect matching}.
- Theorem. *(Lovasz 1979)* PerfectMatching $\in$ RNC^2.
- The input $G = (L \dot\cup R, E)$ is given as a $n \times n$ *biadjacency matrix* $A = (a_{ij})_{i,j \in n}$, where $n = |L| = |R|$.
- Algorithm.

1. Construct $B = (b_{ij})_{i,j \in n}$ as follows: If $a_{ij}=0$, then $b_{ij}=0$. Else, pick b_{ij} independently and uniformly <u>at random</u> from $[2n]$.
2. Compute $\det(B)$.
3. If $\det(B) \neq 0$ output "yes", else output "no".

Perfect matching in RNC

- Let PerfectMatching = {Bipartite graph G : G has a perfect matching}.
- Theorem. *(Lovasz 1979)* PerfectMatching $\in$ RNC^2.
- The input $G = (L \dot\cup R, E)$ is given as a $n \times n$ *biadjacency matrix* $A = (a_{ij})_{i,j \in n}$, where $n = |L| = |R|$.
- Algorithm. (RNC^2 algorithm)

1. Construct $B = (b_{ij})_{i,j \in n}$ as follows: If $a_{ij}=0$, then $b_{ij}=0$. Else, pick b_{ij} independently and uniformly <u>at random</u> from $[2n]$. (This can be done using n^2 processors.)
2. Compute $\det(B)$. ($\det$ is in NC^2, *Csanky '76*)
3. If $\det(B) \neq 0$ output "yes", else output "no".

Perfect matching in RNC

- Let PerfectMatching = {Bipartite graph G : G has a perfect matching}.
- Theorem. *(Lovasz 1979)* PerfectMatching $\in RNC^2$.
- Correctness of the Algorithm.

1. Define $X = (x_{ij})_{i,j \in n}$ as follows: If $a_{ij}=0$, then $x_{ij}=0$. Else, x_{ij} is a formal variable.
2. $\det(X) = \sum_{\sigma \in S_n} (-1)^{sign(\sigma)} \prod_{i \in [n]} x_{i\,\sigma(i)}$.

- S_n is the set of all permutations on [n].

Perfect matching in RNC

- Let PerfectMatching = {Bipartite graph G : G has a perfect matching}.
- Theorem. *(Lovasz 1979)* PerfectMatching $\in$ RNC^2.
- Correctness of the Algorithm.

1. Define $X = (x_{ij})_{i,j \in n}$ as follows: If $a_{ij}=0$, then $x_{ij}=0$. Else, x_{ij} is a formal variable.
2. $\det(X) = \sum_{\sigma \in S_n} (-1)^{\text{sign}(\sigma)} \prod_{i \in [n]} x_{i\,\sigma(i)}$.

- Obs. $\det(X) \neq 0$ $\Leftrightarrow$ G has a perfect matching.

Perfect matching in RNC

- Let PerfectMatching = {Bipartite graph G : G has a perfect matching}.
- Theorem. *(Lovasz 1979)* PerfectMatching $\in$ RNC^2.
- Correctness of the Algorithm.

1. Define $X = (x_{ij})_{i,j \in n}$ as follows: If $a_{ij}=0$, then $x_{ij}=0$. Else, x_{ij} is a formal variable.
2. $\det(X) = \sum_{\sigma \in S_n} (-1)^{sign(\sigma)} \prod_{i \in [n]} x_{i\,\sigma(i)}$.

- Obs. $\det(X) \neq 0$ $\Longleftrightarrow$ G has a perfect matching.

Polynomial in the x_{ij} variables.

Perfect matching in RNC

- Let PerfectMatching = {Bipartite graph G : G has a perfect matching}.
- Theorem. *(Lovasz 1979)* PerfectMatching $\in RNC^2$.
- Correctness of the Algorithm.

1. Define $X = (x_{ij})_{i,j \in n}$ as follows: If $a_{ij}=0$, then $x_{ij}=0$. Else, x_{ij} is a formal variable.
2. $\det(X) = \sum_{\sigma \in S_n} (-1)^{\text{sign}(\sigma)} \prod_{i \in [n]} x_{i\,\sigma(i)}$.

- Obs. $\det(X) \neq 0$  G has a perfect matching.
- In the algorithm, we set $x_{ij} = b_{ij}$, where b_{ij} is picked randomly from $[2n]$ if $x_{ij} \neq 0$, otherwise $b_{ij} = 0$.

Perfect matching in RNC

- Let PerfectMatching = {Bipartite graph G : G has a perfect matching}.
- Theorem. *(Lovasz 1979)* PerfectMatching $\in RNC^2$.
- Correctness of the Algorithm.

1. Define $X = (x_{ij})_{i,j \in n}$ as follows: If $a_{ij}=0$, then $x_{ij}=0$. Else, x_{ij} is a formal variable.
2. $\det(X) = \sum_{\sigma \in S_n} (-1)^{sign(\sigma)} \prod_{i \in [n]} x_{i\,\sigma(i)}$.

- Obs. $\det(X) \neq 0$  G has a perfect matching.
- If $\det(X) = 0$ then $\det(B) = 0$. (So, the algorithm has one-sided error.)

Perfect matching in RNC

- Let PerfectMatching = {Bipartite graph G : G has a perfect matching}.
- Theorem. *(Lovasz 1979)* PerfectMatching $\in RNC^2$.
- Correctness of the Algorithm.

1. Define $X = (x_{ij})_{i,j \in n}$ as follows: If $a_{ij}=0$, then $x_{ij}=0$. Else, x_{ij} is a formal variable.
2. $\det(X) = \sum_{\sigma \in S_n} (-1)^{sign(\sigma)} \prod_{i \in [n]} x_{i\,\sigma(i)}$.

- Obs. $\det(X) \neq 0$ $\Leftrightarrow$ G has a perfect matching.
- If $\det(X) \neq 0$, what is the probability that $\det(B) \neq 0$?

The answer is given by the **Schwartz-Zippel lemma**

Schwartz-Zippel lemma

- Lemma. *(Schwartz 1980, Zippel 1979)* Let $f(x_1, \ldots, x_n) \neq 0$ be a multivariate polynomial of (total) degree at most d over a field F. Let $S \subseteq F$ be finite, and $(a_1, \ldots, a_n) \in S^n$ such that each a_i is chosen independently and uniformly at random from S. Then,

$$\Pr_{(a_1, \ldots, a_n) \in_r S^n} [f(a_1, \ldots, a_n) = 0] \leq d/|S|.$$

- *Proof idea.* Roots are far fewer than non-roots. Use induction on the number of variables.

(Homework / reading exercise)

Perfect matching in RNC

- Let PerfectMatching = {Bipartite graph G : G has a perfect matching}.
- Theorem. *(Lovasz 1979)* PerfectMatching $\in RNC^2$.
- Correctness of the Algorithm.

1. Define $X = (x_{ij})_{i,j \in n}$ as follows: If $a_{ij}=0$, then $x_{ij}=0$. Else, x_{ij} is a formal variable.
2. $\det(X) = \sum_{\sigma \in S_n} (-1)^{sign(\sigma)} \prod_{i \in [n]} x_{i\,\sigma(i)}$.

- Obs. $\det(X) \neq 0$ $\Leftrightarrow$ G has a perfect matching.
- If $\det(X) \neq 0$, then $Pr[\det(B) \neq 0] \geq ½$ as degree of $\det(X) = n$ (by the Schwartz-Zippel lemma).

Perfect matching in RNC

- Theorem. *(Mulmuley, Vazirani, Vazirani 1987)* Finding a maximum matching in a general graph is in RNC^2.

- Is finding maximum matching in NC ? Open!

Perfect matching in RNC

- Theorem. *(Mulmuley, Vazirani, Vazirani 1987)* Finding a maximum matching in a general graph is in RNC^2.

- Is finding maximum matching in NC ? Open!
- Theorem. *(Fenner, Gurjar, Thierauf 2016; Svensson, Tarnawski 2017)* Finding a maximum matching in a general graph is in quasi-NC.

In $O((\log n)^3)$ time using $\exp(O((\log n)^3))$ processors,

Randomized space bounded computation

Space bounded PTMs

- We say a PTM M uses $S(n)$ space if on a length-n input, M halts using at most $S(n)$ cells of it work-tape *regardless of its random choices*.

- Definition. A language L is in BPL if there's a PTM M such that M uses $O(\log n)$-space and for every $x \in \{0,1\}^*$, $\Pr[M(x) = L(x)] \geq 2/3$.

Space bounded PTMs

- We say a PTM M uses $S(n)$ space if on a length-n input, M halts using at most $S(n)$ cells of it work-tape *regardless of its random choices*.

- Definition. A language L is in BPL if there's a PTM M such that M uses $O(\log n)$-space and for every $x \in \{0,1\}^*$, $\Pr[M(x) = L(x)] \geq 2/3$.

- The success probability can be amplied as before as the BPP error reduction trick can be implemented using log-space.

Space bounded PTMs

- We say a PTM M uses $S(n)$ space if on a length-n input, M halts using at most $S(n)$ cells of it work-tape *regardless of its random choices*.

- Definition. A language L is in RL if there's a PTM M s.t. M uses $O(\log n)$-space and for every $x \in \{0,1\}^*$,

$$x \in L \Rightarrow \Pr[M(x) = 1] \geq 2/3$$

$$x \notin L \Rightarrow \Pr[M(x) = 0] = 1.$$

- Clearly, $RL \subseteq NL \subseteq P$ and $BPL \subseteq BPP$.

Space bounded PTMs

- We say a PTM M uses S(n) space if on a length-n input, M halts using at most S(n) cells of it work-tape *regardless of its random choices*.

- Claim. BPL ⊆ P .
- *Proof idea*. Think of the adjancency matrix A of the configuration graph of the O(log n)-space PTM. Compute the probability of acceptance by taking powers of A. *(Assignment problem)*

Space bounded PTMs

- We say a PTM M uses S(n) space if on a length-n input, M halts using at most S(n) cells of it work-tape *regardless of its random choices*.

- Claim. BPL $\subseteq$ P .
- *Proof idea*. Think of the adjancency matrix A of the configuration graph of the O(log n)-space PTM. Compute the probability of acceptance by taking powers of A. *(Assignment problem)*

- Is BPL = L ? Many believe that the answer is "Yes" !

Space bounded PTMs

- Theorem. *(Nisan '92, '94)* If $L \in BPL$ then there's a poly-time, $O((\log n)^2)$-space TM that decides L.

- Theorem. *(Saks, Zhou '99)* If $L \in BPL$ then there's a $n^{O(\sqrt{\log n})}$-time, $O((\log n)^{1.5})$-space TM that decides L.

- The second result extends Nisan's techniques.

Randomized reductions

Randomized reduction

- Definition. We say a L_1 reduces to a L_2 in *randomized polynomial-time*, denoted $L_1 \leq_r L_2$, if there's a poly-time PTM M s.t. for every $x \in \{0,1\}^*$

$$\Pr [L_1(x) = L_2(M(x))] \geq 2/3.$$

Success probability

- For arbitrary L_1 and L_2, we may not be able to boost the success probability $2/3$, and so, the above kind of reductions **needn't be transitive**.

Randomized reduction

- Definition. We say a L_1 reduces to a L_2 in *randomized polynomial-time*, denoted $L_1 \leq_r L_2$, if there's a poly-time PTM M s.t. for every $x \in \{0,1\}^*$

$$\Pr [L_1(x) = L_2(M(x))] \geq 2/3.$$

- For arbitrary L_1 and L_2, we may not be able to boost the success probability $2/3$, and so, the above kind of reductions **needn't be transitive**. However,
- Obs. If $L_1 \leq_r L_2$ and $L_2 \in$ BPP, then $L_1 \in$ BPP.

(Easy homework)

Randomized reduction

- Definition. We say a L_1 reduces to a L_2 in *randomized polynomial-time*, denoted $L_1 \leq_r L_2$, if there's a poly-time PTM M s.t. for every $x \in \{0,1\}^*$

$$\Pr [L_1(x) = L_2(M(x))] \geq 2/3.$$

- Obs. If $L_2 = SAT$, then we can boost the success probability from $\frac{1}{2} + |x|^{-c}$ to $1 - \exp(-|x|^d)$.
- *Proof idea.* BPP error reduction trick + Cook-Levin.

(homework)

Randomized reduction

- Definition. We say a L_1 reduces to a L_2 in *randomized polynomial-time*, denoted $L_1 \leq_r L_2$, if there's a poly-time PTM M s.t. for every $x \in \{0,1\}^*$

$$\Pr [L_1(x) = L_2(M(x))] \geq 2/3.$$

- Obs. If $L_2 = SAT$, then we can boost the success probability from $\frac{1}{2} + |x|^{-c}$ to $1 - \exp(-|x|^d)$.
- Recall, $NP = \{L : L \leq_p SAT\}$. It makes sense to define a similar class using randomized poly-time reduction.

Class BP.NP

- Definition. We say a L_1 reduces to a L_2 in *randomized polynomial-time*, denoted $L_1 \leq_r L_2$, if there's a poly-time PTM M s.t. for every $x \in \{0,1\}^*$

$$\Pr [L_1(x) = L_2(M(x))] \geq 2/3.$$

- Obs. If $L_2 = SAT$, then we can boost the success probability from $\frac{1}{2} + |x|^{-c}$ to $1 - \exp(-|x|^d)$.
- Definition. $BP.NP = \{L : L \leq_r SAT\}$.
- Class BP.NP is also known as AM (*Arthur-Merlin protocol*) in the literature.

Class BP.NP

- Definition. BP.NP = $\{L : L \leq_r SAT\}$.
- Observe that NP $\subseteq$ BP.NP and BPP $\subseteq$ BP.NP. Is BP.NP = NP ?

Class BP.NP

- Definition. BP.NP = $\{L : L \leq_r SAT\}$.
- Observe that NP $\subseteq$ BP.NP and BPP $\subseteq$ BP.NP. Is BP.NP = NP ? Many believe that the answer is "yes".

- Theorem. If certain reasonable circuit lower bounds hold, then BP.NP = NP.
- *Proof idea*. Similar to Nisan & Wigderson's conditional BPP = P result.

Class BP.NP

- Definition. BP.NP = $\{L : L \leq_r SAT\}$.
- Observe that NP $\subseteq$ BP.NP and BPP $\subseteq$ BP.NP. Is BP.NP = NP ? Many believe that the answer is "yes".

- We may further ask:

1. Is BP.NP in PH? Recall, BPP is in PH.
2. Is SAT $\in$ BP.NP? Recall, if SAT $\in$ BPP then PH collapses. (SAT $\in$ BP.NP as NP $\subseteq$ BP.NP .)

Class BP.NP

- Definition. BP.NP = $\{L : L \leq_r SAT\}$.

- Theorem. BP.NP is in $\sum_3$. (In fact, BP.NP is in $\prod_2$.)
- *Proof idea.* Similar to the Sipser-Gacs-Lautemann theorem. *(Assignment problem)*

Class BP.NP

- Definition. BP.NP = $\{L : L \leq_r SAT\}$.

- Theorem. BP.NP is in $\sum_3$. (In fact, BP.NP is in $\prod_2$.)
- *Proof idea.* Similar to the Sipser-Gacs-Lautemann theorem. *(Assignment problem)*

- Wondering if BP.NP $\subseteq \prod_2$ implies BP.NP $\subseteq \sum_2$? Is BP.NP = co-BP.NP ? (Recall, BPP = co-BPP).
- If BP.NP = co-BP.NP then co-NP $\subseteq$ BP.NP. The next theorem shows that this would lead to PH collapse.

Class BP.NP

- Definition. BP.NP = $\{L : L \leq_r SAT\}$.

- Theorem. If $\overline{SAT} \in$ BP.NP then PH = $\sum_3$ (in fact, PH = $\sum_2$).
- *Proof idea.* Similar to Adleman's theorem + Karp-Lipton theorem. *(Assignment problem)*

Class BP.NP

- Definition. BP.NP = $\{L : L \leq_r SAT\}$.
- Theorem. If $\overline{SAT} \in$ BP.NP then PH = $\sum_2$.
- We would use the above theorem to show that if GI is NP-complete then PH collapses.
- Thus, even without designing an efficient algorithm for GI, we know GI is unlikely to be NP-complete.

Class BP.NP

- Definition. BP.NP = $\{L : L \leq_r SAT\}$.
- Theorem. If $\overline{SAT} \in$ BP.NP then PH = $\sum_2$.
- We would use the above theorem to show that if GI is NP-complete then PH collapses.
- Theorem. *(Goldwasser-Sipser '87, Boppana, Hastad, Zachos '87)* GNI $\in$ BP.NP.
- *Proof.* We'll prove it.

Class BP.NP

- Definition. BP.NP = $\{L : L \leq_r SAT\}$.
- Theorem. If $\overline{SAT} \in$ BP.NP then PH = $\sum_2$.
- We would use the above theorem to show that if GI is NP-complete then PH collapses.
- Theorem. *(Goldwasser-Sipser '87, Boppana, Hastad, Zachos '87)* GNI $\in$ BP.NP.
- If GI is NP-complete then GNI is co-NP-complete. If so, then the above two theorems imply PH = $\sum_2$.

Graph Isomorphism in Quasi-P

- Theorem. *(Babai 2015)* There's a deterministic $\exp(O((\log n)^3))$ time algorithm to solve the graph isomorphism problem.

GNI is in BP.NP

Graph Non-isomorphism

- Definition. Let G_1 and G_2 be two undirected graphs on n vertices. Identify the vertices with $[n]$. We say G_1 is isomorphic to G_2, denoted $G_1 \cong G_2$, if there's a bijection/permutation $\pi:[n] \to [n]$ s.t. for all $u, v \in [n]$, (u,v) is an edge in G_1 if and only if $(\pi(u),\pi(v))$ is an edge in G_2.

- Definition. $GNI = \{(G_1, G_2) : G_1 \ncong G_2\}$.

- Clearly, $GNI \in$ co-NP, it is not known if $GNI \in NP$.

GNI is in BP.NP

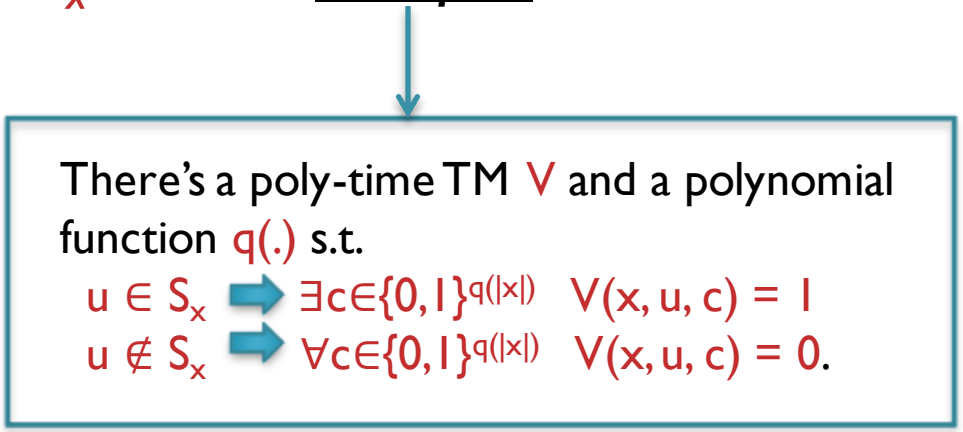
- The idea.

1. **Step 1**: Let $x = (G_1, G_2)$. Associate a set S_x with (G_1, G_2) s.t. $|S_x|$ is "large" $(2n!)$ if $G_1 \ncong G_2$, and $|S_x|$ is "small" $(n!)$ if $G_1 \cong G_2$. Elements of S_x can be represented using $m = n^{O(1)}$ bits. *Furthermore, membership in S_x can be certified in $m^{O(1)} = n^{O(1)}$* time.

GNI is in BP.NP

- The idea.

1. **Step 1**: Let $x = (G_1, G_2)$. Associate a set S_x with (G_1, G_2) s.t. $|S_x|$ is "large" $(2n!)$ if $G_1 \ncong G_2$, and $|S_x|$ is "small" $(n!)$ if $G_1 \cong G_2$. Elements of S_x can be represented using $m = n^{O(1)}$ bits. *Furthermore, membership in S_x can be certified in $m^{O(1)} = n^{O(1)}$* time.



GNI is in BP.NP

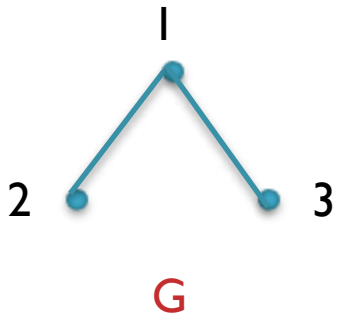
- The idea.

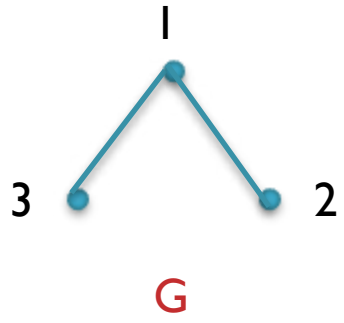
1. **Step 1**: Let $x = (G_1, G_2)$. Associate a set S_x with (G_1, G_2) s.t. $|S_x|$ is "large" (2n!) if $G_1 \ncong G_2$, and $|S_x|$ is "small" (n!) if $G_1 \cong G_2$. Elements of S_x can be represented using $m = n^{O(1)}$ bits. *Furthermore, membership in S_x can be <u>certified</u> in $m^{O(1)} = n^{O(1)}$* time.

2. **Step 2:** Devise a *<u>randomized</u>* poly-time reduction that maps x to a CNF $\phi_{x,r}$ s.t. over the randomness of r, $\phi_{x,r}$ is satisfiable w.h.p if S_x is "large" and unsatisfiable w.h.p if S_x is "small".

GNI is in BP.NP

- **Step 1**: Let $x = (G_1, G_2)$. Associate a set S_x with (G_1, G_2) s.t. $|S_x|$ is "large" $(2n!)$ if $G_1 \ncong G_2$, and $|S_x|$ is "small" $(n!)$ if $G_1 \cong G_2$. Elements of S_x can be represented using $m = n^{O(1)}$ bits. *Furthermore, membership in S_x can be certified in $m^{O(1)} = n^{O(1)}$* time.
- Defn. $\mathrm{Aut}(G) = \{\text{bijection } \pi:[n] \to [n] : \pi(G) = G\}$.





Permutation $\pi = (1,3,2)$ is in $\mathrm{Aut}(G)$.

GNI is in BP.NP

- **Step 1**: Let $x = (G_1, G_2)$. Associate a set S_x with (G_1, G_2) s.t. $|S_x|$ is "large" $(2n!)$ if $G_1 \ncong G_2$, and $|S_x|$ is "small" $(n!)$ if $G_1 \cong G_2$. Elements of S_x can be represented using $m = n^{O(1)}$ bits. *Furthermore, membership in S_x can be certified in $m^{O(1)} = n^{O(1)}$* time.
- Defn. $\mathrm{Aut}(G) = \{\text{bijection } \pi:[n] \to [n] : \pi(G) = G\}$.
- Let $S_x = \{(H, \pi): H \cong G_1 \text{ or } H \cong G_2 \text{ and } \pi \in \mathrm{Aut}(H)\}$.
- Obs. S_x satisfies the properties stated in Step 1.

(Homework)

GNI is in BP.NP

- **Step 2:** Devise a *randomized* poly-time reduction that maps x to a CNF $\phi_{x,r}$ s.t. over the randomness of r, $\phi_{x,r}$ is satisfiable w.h.p if S_x is "large" and unsatisfiable w.h.p if S_x is "small".

GNI is in BP.NP

- **Step 2:** Devise a *randomized* poly-time reduction that maps x to a CNF $\phi_{x,r}$ s.t. over the randomness of r, $\phi_{x,r}$ is satisfiable w.h.p if S_x is "large" and unsatisfiable w.h.p if S_x is "small".
- Lemma *. There's a poly-time TM M that takes input $x = (G_1, G_2)$, y & r, and a polynomial function $q(.)$ s.t.

  $|S_x| = 2n!$ (large) ⇒ $Pr_r [\exists y \text{ s.t. } M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ⇒ $Pr_r [\forall y \text{ s.t. } M(x, y, r) = 0] \geq 2/3$.

$r \in \{0,1\}^{q(|x|)}$ $y \in \{0,1\}^{q(|x|)}$

GNI is in BP.NP

- **Step 2:** Devise a *randomized* poly-time reduction that maps x to a CNF $\phi_{x,r}$ s.t. over the randomness of r, $\phi_{x,r}$ is satisfiable w.h.p if S_x is "large" and unsatisfiable w.h.p if S_x is "small".
- Lemma *. There's a poly-time TM M that takes input $x = (G_1, G_2)$, y & r, and a polynomial function $q(.)$ s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r[\exists y \text{ s.t. } M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r[\forall y \text{ s.t. } M(x, y, r) = 0] \geq 2/3$.
- *Proof.* Uses ***Goldwasser-Sipser set lower bound protocol***. We'll see the proof in a while.

GNI is in BP.NP

- **Step 2:** Devise a *randomized* poly-time reduction that maps x to a CNF $\phi_{x,r}$ s.t. over the randomness of r, $\phi_{x,r}$ is satisfiable w.h.p if S_x is "large" and unsatisfiable w.h.p if S_x is "small".
- Lemma *. There's a poly-time TM M that takes input $x = (G_1, G_2)$, y & r, and a polynomial function $q(.)$ s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r[\exists y \text{ s.t. } M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r[\forall y \text{ s.t. } M(x, y, r) = 0] \geq 2/3$.

We can think of M's computation as a Boolean circuit $\psi_{x,r}(y)$, which can be computed in randomized $|x|^{O(1)}$ time by fixing x and picking $r \in \{0,1\}^{q(n)}$ randomly. *Cook-Levin*

GNI is in BP.NP

- **Step 2:** Devise a *randomized* poly-time reduction that maps x to a CNF $\phi_{x,r}$ s.t. over the randomness of r, $\phi_{x,r}$ is satisfiable w.h.p if S_x is "large" and unsatisfiable w.h.p if S_x is "small".
- Corollary. There's randomized poly-time reduction that maps x to a Boolean circuit $\psi_{x,r}$ s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r[\psi_{x,r}(y)$ is satisfiable$] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r[\psi_{x,r}(y)$ is unsatisfiable$] \geq 2/3$.

GNI is in BP.NP

- **Step 2:** Devise a *randomized* poly-time reduction that maps x to a CNF $\phi_{x,r}$ s.t. over the randomness of r, $\phi_{x,r}$ is satisfiable w.h.p if S_x is "large" and unsatisfiable w.h.p if S_x is "small".
- Corollary. There's randomized poly-time reduction that maps x to a CNF $\phi_{x,r}$ s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r[\phi_{x,r}(z)$ is satisfiable$] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r[\phi_{x,r}(z)$ is unsatisfiable$] \geq 2/3$.

$\phi_{x,r}$ is a CNF and z = y + auxiliary variables.

Cook-Levin

GNI is in BP.NP

- **Step 2:** Devise a *randomized* poly-time reduction that maps x to a CNF $\phi_{x,r}$ s.t. over the randomness of r, $\phi_{x,r}$ is satisfiable w.h.p if S_x is "large" and unsatisfiable w.h.p if S_x is "small".
- Corollary. There's randomized poly-time reduction that maps x to a CNF $\phi_{x,r}$ s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r[\phi_{x,r}(z)$ is satisfiable$] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r[\phi_{x,r}(z)$ is unsatisfiable$] \geq 2/3$.

- Hence, GNI is in BP.NP. It remains to prove Lemma *.

Set lower bound protocol

- Lemma *. There's a poly-time TM M that takes input x = (G_1, G_2), y & r, and a polynomial function q(.) s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r [\exists y$ s.t. $M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r [\forall y$ s.t. $M(x, y, r) = 0] \geq 2/3$.

$r \in \{0,1\}^{q(|x|)}$ $y \in \{0,1\}^{q(|x|)}$

Set lower bound protocol

- Lemma *. There's a poly-time TM M that takes input x = (G_1, G_2), y & r, and a polynomial function q(.) s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r[\exists y$ s.t. $M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r[\forall y$ s.t. $M(x, y, r) = 0] \geq 2/3$.

- *Proof idea.* Let $H = \{h_i\}$ be a "suitable" family of hash functions that map m-bit strings to k-bit strings for an appropriate k. Recall, m = size of an element in S_x.

The value of k will be fixed in the analysis.

Set lower bound protocol

- Lemma *. There's a poly-time TM M that takes input $x = (G_1, G_2)$, y & r, and a polynomial function $q(.)$ s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r[\exists y$ s.t. $M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r[\forall y$ s.t. $M(x, y, r) = 0] \geq 2/3$.

- *Proof idea.* Let $H = \{h_i\}$ be a "suitable" family of hash functions that map m-bit strings to k-bit strings for an appropriate k. Recall, m = size of an element in S_x.
- Let $t = n^{O(1)}$ be sufficiently large. M interprets r as $((i_1,v_1), (i_2,v_2),\ldots, (i_t,v_t))$, where $i_1,\ldots, i_t$ are indices of hash functions in H, and $v_1,\ldots, v_t$ are k-bit strings.

Set lower bound protocol

- Lemma *. There's a poly-time TM M that takes input $x = (G_1, G_2)$, y & r, and a polynomial function $q(.)$ s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r[\exists y$ s.t. $M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r[\forall y$ s.t. $M(x, y, r) = 0] \geq 2/3$.

- *Proof idea.* Let $H = \{h_i\}$ be a "suitable" <u>family of hash functions</u> that map m-bit strings to k-bit strings for an appropriate k. Recall, m = size of an element in S_x.

- Let $t = n^{O(1)}$ be sufficiently large. M interprets r as $((i_1,v_1), (i_2,v_2),\ldots, (i_t,v_t))$, where $i_1,\ldots, i_t$ are indices of hash functions in H, and $v_1,\ldots, v_t$ are k-bit strings.

↓

Actually, we don't need $v_1,\ldots, v_t$ to be random. We can assume these are all 0^k.

Set lower bound protocol

- Lemma *. There's a poly-time TM M that takes input $x = (G_1, G_2)$, y & r, and a polynomial function $q(.)$ s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r[\exists y$ s.t. $M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r[\forall y$ s.t. $M(x, y, r) = 0] \geq 2/3$.

- *Proof idea.* Let $H = \{h_i\}$ be a "suitable" <u>family of hash functions</u> that map m-bit strings to k-bit strings for an appropriate k. Recall, m = size of an element in S_x.

- Let $t = n^{O(1)}$ be sufficiently large. M interprets r as $((i_1,v_1), (i_2,v_2),\ldots, (i_t,v_t))$, where $i_1,\ldots, i_t$ are indices of hash functions in H, and $v_1,\ldots, v_t$ are k-bit strings.

$$|r| = n^{O(1)}.$$

Set lower bound protocol

- Lemma *. There's a poly-time TM M that takes input $x = (G_1, G_2)$, y & r, and a polynomial function $q(.)$ s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r[\exists y$ s.t. $M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r[\forall y$ s.t. $M(x, y, r) = 0] \geq 2/3$.

- *Proof idea.* Let $H = \{h_i\}$ be a "suitable" family of hash functions that map m-bit strings to k-bit strings for an appropriate k. Recall, m = size of an element in S_x.

- M interprets y as $((u_1,c_1), (u_2,c_2),\ldots, (u_t,c_t))$, where $u_1,\ldots, u_t$ are m-bit strings, and c_p is an alleged certificate of u_p's membership in S_x for every $p \in [t]$.

$$|y| = n^{O(1)}.$$

Set lower bound protocol

- Lemma *. There's a poly-time TM M that takes input $x = (G_1, G_2)$, y & r, and a polynomial function $q(.)$ s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r [\exists y \text{ s.t. } M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r [\forall y \text{ s.t. } M(x, y, r) = 0] \geq 2/3$.

- *Proof idea.* Let $H = \{h_i\}$ be a "suitable" family of hash functions that map m-bit strings to k-bit strings for an appropriate k. Recall, m = size of an element in S_x.

- For every $p \in [t]$: M uses c_p & x to check if $u_p \in S_x$. If yes, M checks if $h_{i_p}(u_p) = v_p$.

Recall, membership in S_x can be efficiently certified.

Set lower bound protocol

- Lemma *. There's a poly-time TM M that takes input x = (G_1, G_2), y & r, and a polynomial function q(.) s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r[\exists y$ s.t. $M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r[\forall y$ s.t. $M(x, y, r) = 0] \geq 2/3$.

- *Proof idea.* Let $H = \{h_i\}$ be a "suitable" family of hash functions that map m-bit strings to k-bit strings for an appropriate k. Recall, m = size of an element in S_x.

- For every $p \in [t]$: M uses c_p & x to check if $u_p \in S_x$. If yes, M checks if $h_{i_p}(u_p) = v_p$. If sufficiently many (say, t^*) of these checks pass, M outputs 1, else it o/ps 0.

Set lower bound protocol

- Lemma *. There's a poly-time TM M that takes input x = (G_1, G_2), y & r, and a polynomial function q(.) s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r[\exists y$ s.t. $M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r[\forall y$ s.t. $M(x, y, r) = 0] \geq 2/3$.

- *Proof idea.* Let $H = \{h_i\}$ be a "suitable" family of hash functions that map m-bit strings to k-bit strings for an appropriate k. Recall, m = size of an element in S_x.

- For every $p \in [t]$: M uses c_p & x to check if $u_p \in S_x$. If yes, M checks if $h_{i_p}(u_p) = v_p$. If sufficiently many (say, t^*) of these checks pass, M outputs 1, else it o/ps 0. Intuitively, $\exists y$ s.t. t^* of the checks pass iff S_x is large.

Set lower bound protocol

- Lemma *. There's a poly-time TM M that takes input x = (G_1, G_2), y & r, and a polynomial function q(.) s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r[\exists y$ s.t. $M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r[\forall y$ s.t. $M(x, y, r) = 0] \geq 2/3$.

- *Proof idea.* Let $H = \{h_i\}$ be a "suitable" (??) family of hash functions that map m-bit strings to k-bit strings for an appropriate k. Recall, m = size of an element in S_x.
- For every $p \in [t]$: M uses c_p & x to check if $u_p \in S_x$. If yes, M checks if $h_{i_p}(u_p) = v_p$. If sufficiently many (say, t^*) of these checks pass, M outputs 1, else it o/ps 0. Intuitively, $\exists y$ s.t. t^* of the checks pass iff S_x is large.

Pairwise independent hash functions

- Definition. A family $H_{m,k}$ of (hash) functions from $\{0,1\}^m$ to $\{0,1\}^k$ is *pairwise independent* if for every distinct $x, x' \in \{0,1\}^m$ and for every $y, y' \in \{0,1\}^k$,

$$\Pr_{h \in_r H_{m,k}} [h(x) = y \text{ and } h(x') = y'] = 2^{-2k}.$$

Pairwise independent hash functions

- Definition. A family $H_{m,k}$ of (hash) functions from $\{0,1\}^m$ to $\{0,1\}^k$ is *pairwise independent* if for every distinct $x, x' \in \{0,1\}^m$ and for every $y, y' \in \{0,1\}^k$,

$$\Pr_{h \in_r H_{m,k}} [h(x) = y \text{ and } h(x') = y'] = 2^{-2k}.$$

- Obs. Let $H_{m,k}$ be a pairwise independent hash function family. For every $x \in \{0,1\}^m$ and $y \in \{0,1\}^k$,

$$\Pr_{h \in_r H_{m,k}} [h(x) = y] = 2^{-k}.$$

Pairwise independent hash functions

- Definition. A family $H_{m,k}$ of (hash) functions from $\{0,1\}^m$ to $\{0,1\}^k$ is *pairwise independent* if for every distinct $x, x' \in \{0,1\}^m$ and for every $y, y' \in \{0,1\}^k$,

$$\Pr_{h \in_r H_{m,k}} [h(x) = y \text{ and } h(x') = y'] = 2^{-2k}.$$

$$= \Pr_{h \in_r H_{m,k}} [h(x) = y] \cdot \Pr_{h \in_r H_{m,k}} [h(x') = y'] .$$

Pairwise independent hash functions

- Definition. A family $H_{m,k}$ of (hash) functions from $\{0,1\}^m$ to $\{0,1\}^k$ is *pairwise independent* if for every distinct $x, x' \in \{0,1\}^m$ and for every $y, y' \in \{0,1\}^k$,

$$\Pr_{h \in_r H_{m,k}} [h(x) = y \text{ and } h(x') = y'] = 2^{-2k}.$$

$$= \Pr_{h \in_r H_{m,k}} [h(x) = y] \cdot \Pr_{h \in_r H_{m,k}} [h(x') = y'] .$$

- Example. Let $\ell > 0$ and F be the *finite field* of size 2^ℓ. We can identify F with $\{0,1\}^\ell$ as elements of F are ℓ-bit strings. For $a, b \in F$, define the function $h_{a,b}$ as $h_{a,b}(x) = ax + b$ for every $x \in F$. Then, $H_{\ell,\ell} = \{h_{a,b} : a,b \in F\}$ is a pairwise independent hash family.

Pairwise independent hash functions

- Example. Let $\ell > 0$ and F be the *finite field* of size 2^ℓ. We can identify F with $\{0,1\}^\ell$ as elements of F are ℓ-bit strings. For $a, b \in F$, define the function $h_{a,b}$ as $h_{a,b}(x) = ax + b$ for every $x \in F$. Then, $H_{\ell,\ell} = \{h_{a,b} : a,b \in F\}$ is a pairwise independent hash family.
- *Proof.* Let $x, x' \in F$ be distinct and $y, y' \in F$. Then, $h_{a,b}(x) = y$ & $h_{a,b}(x') = y'$ if and only if $a = (y–y')/(x-x')$ and $b = (xy' – x'y)/(x-x')$.

Pairwise independent hash functions

- Example. Let $\ell > 0$ and F be the *finite field* of size 2^ℓ. We can identify F with $\{0,1\}^\ell$ as elements of F are ℓ-bit strings. For $a, b \in F$, define the function $h_{a,b}$ as $h_{a,b}(x) = ax + b$ for every $x \in F$. Then, $H_{\ell,\ell} = \{h_{a,b} : a,b \in F\}$ is a pairwise independent hash family.
- *Proof.* Let $x, x' \in F$ be distinct and $y, y' \in F$. Then, $h_{a,b}(x) = y$ & $h_{a,b}(x') = y'$ if and only if $a = (y–y')/(x-x')$ and $b = (xy' – x'y)/(x-x')$. Therefore,

$$\Pr_{a,b \in_r F}[h_{a,b}(x) = y \;\&\; h_{a,b}(x') = y']$$
$$= \Pr_{a,b \in_r F}[a = (y–y')/(x-x') \;\&\; b = (xy' – x'y)/(x-x')]$$
$$= 2^{-2\ell} \quad \text{(as } a \text{ and } b \text{ are independently chosen).}$$

Pairwise independent hash functions

- Example. Let $\ell > 0$ and F be the *finite field* of size 2^ℓ. We can identify F with $\{0,1\}^\ell$ as elements of F are ℓ-bit strings. For $a, b \in F$, define the function $h_{a,b}$ as $h_{a,b}(x) = ax + b$ for every $x \in F$. Then, $H_{\ell,\ell} = \{h_{a,b} : a,b \in F\}$ is a pairwise independent hash family.

- Obs. If $m \geq k$, then we can construct a pairwise independent $H_{m,k}$ by considering $H_{m,m}$ as above. Truncate the output of a function to the first k bits.

(Homework)

Pairwise independent hash functions

- Example. Let $\ell > 0$ and F be the *finite field* of size 2^ℓ. We can identify F with $\{0,1\}^\ell$ as elements of F are ℓ-bit strings. For $a, b \in F$, define the function $h_{a,b}$ as $h_{a,b}(x) = ax + b$ for every $x \in F$. Then, $H_{\ell,\ell} = \{h_{a,b} : a,b \in F\}$ is a pairwise independent hash family.

- Obs. If $m \leq k$, then we can construct a pairwise independent $H_{m,k}$ by considering $H_{k,k}$ as above. Generate k-bit i/p for a function by padding with 0.

(Homework)

Set lower bound protocol (contd.)

- Lemma *. There's a poly-time TM M that takes input x = (G_1, G_2), y & r, and a polynomial function q(.) s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r[\exists y$ s.t. $M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r[\forall y$ s.t. $M(x, y, r) = 0] \geq 2/3$.

- *Proof.* Let $H_{m,k}$ be a family of pairwise independent hash functions.

Set lower bound protocol (contd.)

- Lemma *. There's a poly-time TM M that takes input x = (G_1, G_2), y & r, and a polynomial function q(.) s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r[\exists y$ s.t. $M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r[\forall y$ s.t. $M(x, y, r) = 0] \geq 2/3$.

- *Proof.* Let $H_{m,k}$ be a family of pairwise independent hash functions. Recall, $r = ((i_1,v_1), (i_2,v_2),\ldots, (i_t,v_t))$, where $i_1,\ldots, i_t$ are indices of functions in $H_{m,k}$, and $v_1,\ldots,v_t \in \{0,1\}^k$.

Set lower bound protocol (contd.)

- Lemma *. There's a poly-time TM M that takes input x = (G_1, G_2), y & r, and a polynomial function q(.) s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r[\exists y \text{ s.t. } M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r[\forall y \text{ s.t. } M(x, y, r) = 0] \geq 2/3$.

- *Proof.* Let $H_{m,k}$ be a family of pairwise independent hash functions. Recall, $r = ((i_1,v_1), (i_2,v_2),\ldots, (i_t,v_t))$, where $i_1,\ldots, i_t$ are indices of functions in $H_{m,k}$, and $v_1,\ldots,v_t \in \{0,1\}^k$. Also, $y = ((u_1,c_1), (u_2,c_2),\ldots, (u_t,c_t))$, where $u_1,\ldots, u_t \in \{0,1\}^m$, and c_p is an alleged certificate of u_p's membership in S_x for every $p \in [t]$.

Set lower bound protocol (contd.)

- Lemma *. There's a poly-time TM M that takes input $x = (G_1, G_2)$, y & r, and a polynomial function $q(.)$ s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r[\exists y \text{ s.t. } M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r[\forall y \text{ s.t. } M(x, y, r) = 0] \geq 2/3$.

- *Proof.* For every $p \in [t]$: M uses c_p & x to check if $u_p \in S_x$. If yes, M checks if $h_{i_p}(u_p) = v_p$.

Set lower bound protocol (contd.)

- Lemma *. There's a poly-time TM M that takes input x = (G_1, G_2), y & r, and a polynomial function q(.) s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r[\exists y$ s.t. $M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r[\forall y$ s.t. $M(x, y, r) = 0] \geq 2/3$.

- *Proof.* For every $p \in [t]$: M uses c_p & x to check if $u_p \in S_x$. If yes, M checks if $h_{i_p}(u_p) = v_p$.

- For a fixed p, what is the probability (over the randomness of i_p & v_p) that there's a $u_p \in S_x$ s.t. $h_{i_p}(u_p)=v_p$? We'll upper & lower bound this probability.

Set lower bound protocol (contd.)

- Lemma *. There's a poly-time TM M that takes input $x = (G_1, G_2)$, y & r, and a polynomial function $q(.)$ s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r[\exists y$ s.t. $M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r[\forall y$ s.t. $M(x, y, r) = 0] \geq 2/3$.

- *Proof.* For every $p \in [t]$: M uses c_p & x to check if $u_p \in S_x$. If yes, M checks if $h_{i_p}(u_p) = v_p$.

- Simplifying notations. As p is fixed, let $h_{i_p} = h$, $v_p = v$ and $u_p = u$.

Set lower bound protocol (contd.)

- Lemma *. There's a poly-time TM M that takes input x = (G_1, G_2), y & r, and a polynomial function q(.) s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r [\exists y \text{ s.t. } M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r [\forall y \text{ s.t. } M(x, y, r) = 0] \geq 2/3$.

- *Proof.* For every $p \in [t]$: M uses c_p & x to check if $u_p \in S_x$. If yes, M checks if $h_{i_p}(u_p) = v_p$.

- Upper bound. $Pr_{h,v} [\exists u \in S_x \text{ s.t. } h(u) = v] \leq |S_x|/2^k$.
- As $H_{m,k}$ is pairwise independent, for every $u \in \{0,1\}^m$ and $v \in \{0,1\}^k$, $Pr_h [h(u) = v] = 2^{-k}$.

Set lower bound protocol (contd.)

- Lemma *. There's a poly-time TM M that takes input $x = (G_1, G_2)$, y & r, and a polynomial function $q(.)$ s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r [\exists y \text{ s.t. } M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r [\forall y \text{ s.t. } M(x, y, r) = 0] \geq 2/3$.

- *Proof.* For every $p \in [t]$: M uses c_p & x to check if $u_p \in S_x$. If yes, M checks if $h_{i_p}(u_p) = v_p$.

- Lower bound. Fix a $v \in \{0,1\}^k$ arbitrarily. Then,

$$Pr_h [\exists u \in S_x \text{ s.t. } h(u) = v]$$

$$\geq \sum_{u \in S_x} Pr_h [h(u) = v] - \sum_{\substack{u,u' \in S_x \\ u \neq u'}} Pr_h [h(u) = v \text{ \& } h(u') = v]$$

(by inclusion-exclusion principle)

Set lower bound protocol (contd.)

- Lemma *. There's a poly-time TM M that takes input $x = (G_1, G_2)$, y & r, and a polynomial function $q(.)$ s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r [\exists y \text{ s.t. } M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r [\forall y \text{ s.t. } M(x, y, r) = 0] \geq 2/3$.

- *Proof.* For every $p \in [t]$: M uses c_p & x to check if $u_p \in S_x$. If yes, M checks if $h_{i_p}(u_p) = v_p$.

- Lower bound. Fix a $v \in \{0,1\}^k$ arbitrarily. Then,

  $$Pr_h [\exists u \in S_x \text{ s.t. } h(u) = v]$$

  $$\geq |S_x|/2^k - |S_x|^2 / 2^{2k+1}.$$ (as $H_{m,k}$ is pairwise independent)

Set lower bound protocol (contd.)

- Lemma *. There's a poly-time TM M that takes input $x = (G_1, G_2)$, y & r, and a polynomial function $q(.)$ s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r [\exists y \text{ s.t. } M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r [\forall y \text{ s.t. } M(x, y, r) = 0] \geq 2/3$.

- *Proof.* For every $p \in [t]$: M uses c_p & x to check if $u_p \in S_x$. If yes, M checks if $h_{i_p}(u_p) = v_p$.

- Lower bound. Fix a $v \in \{0,1\}^k$ arbitrarily. Then,

$$Pr_h [\exists u \in S_x \text{ s.t. } h(u) = v]$$

$$\geq |S_x|/2^k . (1 - |S_x|/2^{k+1}).$$

(as $H_{m,k}$ is pairwise independent)

Set lower bound protocol (contd.)

- Lemma *. There's a poly-time TM M that takes input x = (G_1, G_2), y & r, and a polynomial function q(.) s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r [\exists y \text{ s.t. } M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r [\forall y \text{ s.t. } M(x, y, r) = 0] \geq 2/3$.

- *Proof.* For every $p \in [t]$: M uses c_p & x to check if $u_p \in S_x$. If yes, M checks if $h_{i_p}(u_p) = v_p$.

- If $|S_x| = n!$ then (by the upper bound)

  $Pr_{h,v} [\exists u \in S_x \text{ s.t. } h(u) = v] \leq n!/2^k$.

Set lower bound protocol (contd.)

- Lemma *. There's a poly-time TM M that takes input $x = (G_1, G_2)$, y & r, and a polynomial function $q(.)$ s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r [\exists y$ s.t. $M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r [\forall y$ s.t. $M(x, y, r) = 0] \geq 2/3$.

- *Proof.* For every $p \in [t]$: M uses c_p & x to check if $u_p \in S_x$. If yes, M checks if $h_{i_p}(u_p) = v_p$.

- If $|S_x| = n!$ then (by the upper bound)

  $Pr_{h,v} [\exists u \in S_x$ s.t. $h(u) = v] \leq n!/2^k$. Hence,

- $Exp_r [\ |\{p \in [t] : \exists u_p \in S_x$ s.t. $h_{i_p}(u_p) = v_p\}|\ ] \leq t.\ n!/2^k$.

Set lower bound protocol (contd.)

- Lemma *. There's a poly-time TM M that takes input x = (G_1, G_2), y & r, and a polynomial function q(.) s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r [\exists y \text{ s.t. } M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r [\forall y \text{ s.t. } M(x, y, r) = 0] \geq 2/3$.

- *Proof.* For every $p \in [t]$: M uses c_p & x to check if $u_p \in S_x$. If yes, M checks if $h_{i_p}(u_p) = v_p$.

- Choosing k. Fix k s.t. $2^{k-2} < 2n! \leq 2^{k-1}$.

- If $|S_x| = 2n!$ then (by the lower bound)

  $Pr_{h,v} [\exists u \in S_x \text{ s.t. } h(u) = v] \geq |S_x|/2^k . (1 - |S_x|/2^{k+1})$

  $\geq |S_x|/2^k . 3/4 = 3/2 . n!/2^k$

Set lower bound protocol (contd.)

- Lemma *. There's a poly-time TM M that takes input x = (G_1, G_2), y & r, and a polynomial function q(.) s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r [\exists y$ s.t. $M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r [\forall y$ s.t. $M(x, y, r) = 0] \geq 2/3$.

- *Proof.* For every $p \in [t]$: M uses c_p & x to check if $u_p \in S_x$. If yes, M checks if $h_{i_p}(u_p) = v_p$.
- Choosing k. Fix k s.t. $2^{k-2} < 2n! \leq 2^{k-1}$.
- If $|S_x| = 2n!$ then (by the lower bound)

  $Pr_{h,v} [\exists u \in S_x$ s.t. $h(u) = v] \geq 3/2 \cdot n!/2^k$. Hence,

- $Exp_r [|\{p \in [t] : \exists u_p \in S_x$ s.t. $h_{i_p}(u_p) = v_p\}|] \geq 3/2 \cdot t \cdot n!/2^k$.

Set lower bound protocol (contd.)

- Lemma *. There's a poly-time TM M that takes input x = (G_1, G_2), y & r, and a polynomial function q(.) s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r[\exists y \text{ s.t. } M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r[\forall y \text{ s.t. } M(x, y, r) = 0] \geq 2/3$.

- *Proof.* For every $p \in [t]$: M uses c_p & x to check if $u_p \in S_x$. If yes, M checks if $h_{i_p}(u_p) = v_p$.

- If $|S_x| = 2n!$ then

  $Exp_r[\ |\{p \in [t] : \exists u_p \in S_x \text{ s.t. } h_{i_p}(u_p) = v_p\}|\ ] \geq 3/2 \cdot t \cdot n!/2^k$.

- If $|S_x| = n!$ then

  $Exp_r[\ |\{p \in [t] : \exists u_p \in S_x \text{ s.t. } h_{i_p}(u_p) = v_p\}|\ ] \leq t \cdot n!/2^k$.

Set lower bound protocol (contd.)

- Lemma *. There's a poly-time TM M that takes input x = (G_1, G_2), y & r, and a polynomial function q(.) s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r[\exists y \text{ s.t. } M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r[\forall y \text{ s.t. } M(x, y, r) = 0] \geq 2/3$.

- *Proof.* For every $p \in [t]$: M uses c_p & x to check if $u_p \in S_x$. If yes, M checks if $h_{i_p}(u_p) = v_p$.

- If $|S_x| = 2n!$ then

  $Exp_r[\ |\{p \in [t] : \exists u_p \in S_x \text{ s.t. } h_{i_p}(u_p) = v_p\}|\ ] \geq 3/2 \cdot t \cdot n!/2^k$.

- If $|S_x| = n!$ then

  $Exp_r[\ |\{p \in [t] : \exists u_p \in S_x \text{ s.t. } h_{i_p}(u_p) = v_p\}|\ ] \leq t \cdot n!/2^k$.

Set lower bound protocol (contd.)

- Lemma *. There's a poly-time TM M that takes input $x = (G_1, G_2)$, y & r, and a polynomial function $q(.)$ s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r[\exists y \text{ s.t. } M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r[\forall y \text{ s.t. } M(x, y, r) = 0] \geq 2/3$.

- *Proof.* For every $p \in [t]$: M uses c_p & x to check if $u_p \in S_x$. If yes, M checks if $h_{i_p}(u_p) = v_p$.

- If $|S_x| = 2n!$, by Chernoff bd. & $n!/2^k \in [1/8, 1/4]$,

  $Pr_r[|\{p \in [t] : \exists u_p \in S_x \text{ s.t. } h_{i_p}(u_p) = v_p\}| \geq 1.4.\ t.\ n!/2^k] \geq 2/3$.

- If $|S_x| = n!$, by Chernoff/Markov bd. & $n!/2^k \in [1/8, 1/4]$

  $Pr_r[|\{p \in [t] : \exists u_p \in S_x \text{ s.t. } h_{i_p}(u_p) = v_p\}| < 1.4.\ t.\ n!/2^k] \geq 2/3$.

(Easy homework)

Set lower bound protocol (contd.)

- Lemma *. There's a poly-time TM M that takes input x = (G_1, G_2), y & r, and a polynomial function q(.) s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r[\exists y \text{ s.t. } M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r[\forall y \text{ s.t. } M(x, y, r) = 0] \geq 2/3$.

- *Proof.* For every $p \in [t]$: M uses c_p & x to check if $u_p \in S_x$. If yes, M checks if $h_{i_p}(u_p) = v_p$. $t^* = 1.4.\, t.\, n!/2^k$

- If $|S_x| = 2n!$, by Chernoff bd. & $n!/2^k \in [1/8, 1/4]$,

  $Pr_r[|\{p \in [t] : \exists u_p \in S_x \text{ s.t. } h_{i_p}(u_p) = v_p\}| \geq 1.4.\, t.\, n!/2^k] \geq 2/3$.

- If $|S_x| = n!$, by Chernoff/Markov bd. & $n!/2^k \in [1/8, 1/4]$

  $Pr_r[|\{p \in [t] : \exists u_p \in S_x \text{ s.t. } h_{i_p}(u_p) = v_p\}| < 1.4.\, t.\, n!/2^k] \geq 2/3$.

Set lower bound protocol (contd.)

- Lemma *. There's a poly-time TM M that takes input x = (G_1, G_2), y & r, and a polynomial function q(.) s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r[\exists y \text{ s.t. } M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r[\forall y \text{ s.t. } M(x, y, r) = 0] \geq 2/3$.

- *Proof.* For every $p \in [t]$: M uses c_p & x to check if $u_p \in S_x$. If yes, M checks if $h_{i_p}(u_p) = v_p$. $t^* = 1.4.t.n!/2^k$

- If $|S_x| = 2n!$ then

  $Pr_r[|\{p \in [t] : \exists u_p \in S_x \text{ s.t. } h_{i_p}(u_p) = v_p\}| \geq t^*] \geq 2/3$.

- If $|S_x| = n!$ then

  $Pr_r[|\{p \in [t] : \exists u_p \in S_x \text{ s.t. } h_{i_p}(u_p) = v_p\}| < t^*] \geq 2/3$.

Set lower bound protocol (contd.)

- Lemma *. There's a poly-time TM M that takes input x = (G_1, G_2), y & r, and a polynomial function q(.) s.t.

  $|S_x| = 2n!$ (large) ➡ $Pr_r[\exists y \text{ s.t. } M(x, y, r) = 1] \geq 2/3$

  $|S_x| = n!$ (small) ➡ $Pr_r[\forall y \text{ s.t. } M(x, y, r) = 0] \geq 2/3$.

- *Proof.* For every $p \in [t]$: M uses c_p & x to check if $u_p \in S_x$. If yes, M checks if $h_{i_p}(u_p) = v_p$. $t^* = 1.4.t.n!/2^k$

- If $|S_x| = 2n!$ then

  $Pr_r[\exists y \text{ s.t. } M(x, y, r) = 1] \geq 2/3$.

- If $|S_x| = n!$ then

  $Pr_r[\forall y \text{ s.t. } M(x, y, r) = 0] \geq 2/3$.