Week 8

Last Class: Randomized Complexity classes (BPP, RP, etc)

other essential uses of randomness:

Differential Privacy

zero knowledge PFs / Interactive PFs

Today: Interactive Proofs

Interactive Proofs

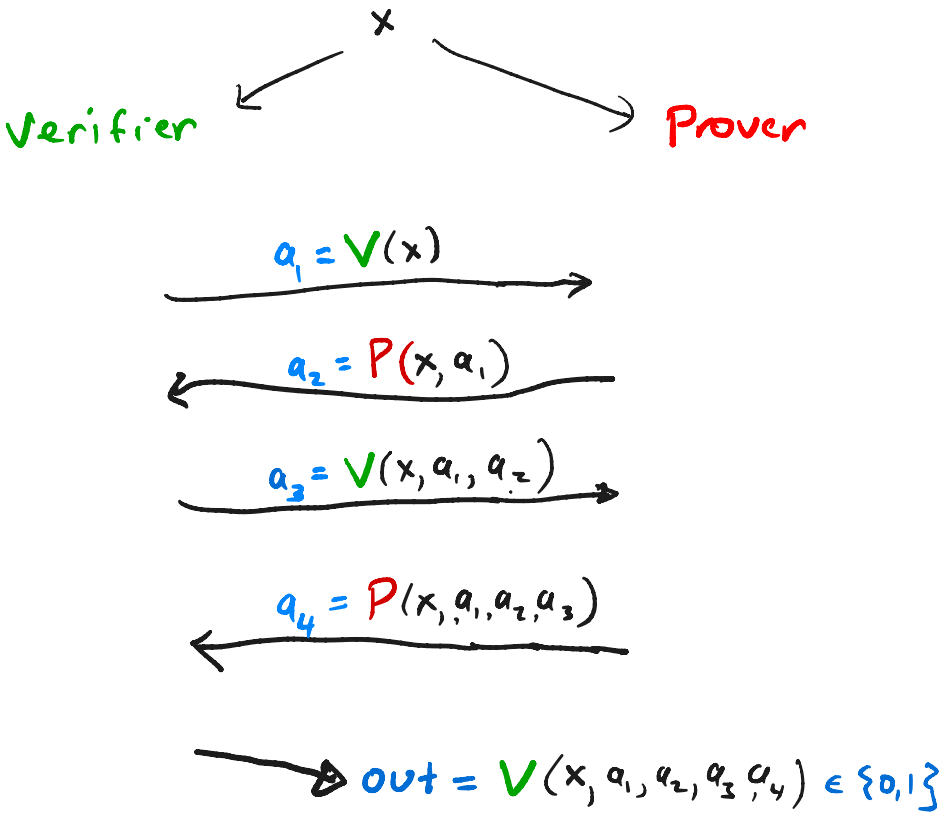
2 Players: Verifier V : computationally bounded

Prover P : computationally unbounded

Idea: IP's give a way for a computationally bounded verifier to decide whether $x \in L$ with the help of an all-powerful prover

However: We don't know if prover (chatgpt, internet?) is honest so protocol has to protect against cheating provers.

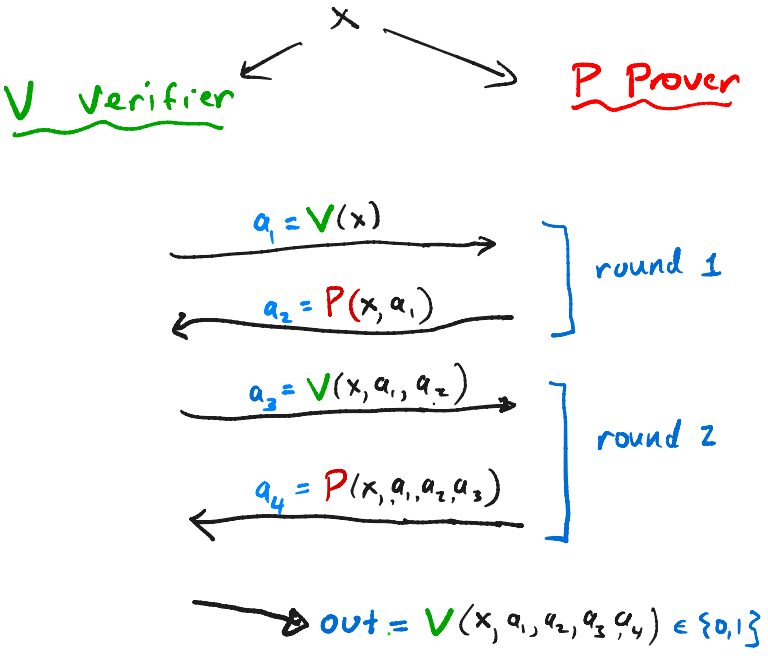
Let $L \subseteq \{0,1\}^*$. A k-round IP for L:



A k-round IP between P + V:

2 functions $P : \{0,1\}^* \to \{0,1\}^*$ — P is randomized, polytime

$V : \{0,1\}^* \to \{0,1\}^*$ — V is any TM (not polytime)



Denote output by $OUT_V\langle V, P\rangle(x)$

Deterministic Interactive Proofs

Defn Let $L \subseteq \{0,1\}^*$ be a language. L has a $k(n)$-round deterministic IP if there is a deterministic TM V that runs in time $\text{poly}(|x|)$ such that the protocol always halts in $k(n)$ rounds and:

(Completeness) $x \in L \Rightarrow \exists P : \{0,1\}^* \to \{0,1\}^* : \text{out}_V \langle V, P \rangle (x) = 1$

(Soundness) $x \notin L \Rightarrow \forall P : \{0,1\}^* \to \{0,1\}^* : \text{out}_V \langle V, P \rangle (x) = 0$

Deterministic Interactive Proofs (dIP)

Q: How powerful are deterministic IPs ?

1. any $L \in P$ has a dIP

   Verifier just ignores prover & outputs answer

2. $NP \in dIP$? Yes.

3. Languages L & NP ?

Deterministic Interactive Proofs

Fact

Let $dIP = \{L \mid L$ has a deterministic IP protocol$\}$

Then $dIP = NP$

says deterministic interaction no stronger than NP

Randomized IPs (IP)

Defn A Language L is in $IP[k]$ if there is a randomized TM V that runs in time $poly(|x|)$ and such that the protocol always halts in k rounds and satisfies:

(Completeness) $x \in L \Rightarrow \exists P \quad Pr[OUT_V\langle V, P\rangle(x) = 1] \geq 2/3$

(Soundness) $x \notin L \Rightarrow \forall P \quad Pr[OUT_V\langle V, P\rangle(x) = 1] \leq 1/3$

$$IP = \bigcup_{k \geq 1} IP[n^k]$$

← # rounds is poly in $|x|$

Randomized IPs (IP)

Defn A Language L is in IP if there is a randomized TM V that runs in time $\text{poly}(|x|)$ and such that the protocol always halts in $\text{poly}(|x|)$ rounds and satisfies:

(Completeness) $x \in L \Rightarrow \exists P \ \Pr[\text{OUT}_V\langle V, P\rangle(x) = 1] \geq 2/3$

(Soundness) $x \notin L \Rightarrow \forall P \ \Pr[\text{OUT}_V\langle V, P\rangle(x) = 1] \leq 1/3$

Notes

1. Probability is over V's random string $\vec{r}$ (Prover is still deterministic)
2. It turns out that randomness of V ($= \vec{r}$) can be made public
   (IP with public $\vec{r}$ $\equiv$ IP with private $\vec{r}$) counterintuitive! Needs extra rounds
3. Can amplify probabilities like we did for BPP

How powerful is IP?

1. $P \subseteq BPP \subseteq IP$
   Also $NP \subseteq IP$ (we saw deterministic $IP = NP$)

2. Last class you saw an IP for graph Isomorphism
   (that was zero knowledge)

   But graph ISO $\in NP$ so it is easy to see it has an ordinary IP (hard part: ZK proof)

3. What about languages harder than NP?
   → IP for graph Non-Isomorphism

Example: IP for graph Nonisomorphism

Recall input to GI : 2 undirected graphs $G_1 = (V_1, E_1)$ $\quad V = V_1 = V_2$
$G_2 = (V_2, E_2)$

$(G_1, G_2) \in GI$ if $\exists$ permutation $\pi : V \to V$ such that

$$\forall (i,j) : (i,j) \in G_1 \iff (\pi(i), \pi(j)) \in G_2$$

Fact $GI \in NP$ so $GI \in IP$

What about $\overline{GI} = \{(G_1, G_2) \mid G_1, G_2 \text{ are NOT isomorphic}\}$?

IP protocol for $\overline{GI}$

V on input $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$:

V:

1. Pick $i \in \{1, 2\}$ uniformly at random ← i not known to P

   Pick random permutation $\pi : V \to V$ and send

   $$H = \pi(G_1) \text{ to } P$$

2. P on input G_1, G_2, H

   Send G_j to V_j $j \in \{1, 2\}$

   (if honest, P sends graph that is isomophic to H to V)

3. V accepts iff $i = j$

IP protocol for $\overline{GI}$

V on input $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$:

1. Pick $i \in \{1, 2\}$ uniformly at random
   Pick random permutation $\pi : V \to V$ and send
   $H = \pi(G_i)$ to V

2. P on input G_1, G_2, H
   Send G_j to V_j $j \in \{1,2\}$
   (if honest, P sends graph that is isomophic to H to V)

3. V accepts iff $i = j$

---

correctness:

(1) $(G_1, G_2) \in \overline{GI}$ (not isomorphic): $\exists P$ s.t. $\Pr[\text{out}_V \langle P, V \rangle (G_1, G_2) = 1] = 1$

(2) $(G_1, G_2) \notin \overline{GI}$ (so they are isomophic): prover has $\frac{1}{2}$ prob of fooling V

$\therefore \forall P \Pr_V[\text{out}_V \langle P V \rangle (G_1, G_2) = 1] \leq \frac{1}{2}$

← repeat $o(1)$ times to get error $\leq \frac{1}{3}$

Major Result 1990 : IP = PSPACE [LFKN, Shamir]

Easier Direction : IP ⊆ PSPACE

Idea : given V, define an "optimal" prover P to be one that maximizes prob. that V accepts x.

Harder (and more surprising) Direction : PSPACE ⊆ IP

Weaker Result $\#SAT \in IP$

(Has most of the main ideas in proof of $PSPACE \subseteq IP$)

$\#SAT$: Input is a 3CNF $F(x_1 .. x_k)$, and $K \in \{0, 1, .., 2^n\}$ K in binary

Accept (F, K) iff F has exactly K satisfying assignments.

Step 0 Arithmetize $F(\vec{x}) \longrightarrow$ polynomial $g_F(\vec{x})$ s.t. $\forall \vec{x} \in \{0,1\}^n \quad F(\vec{x}) \equiv g(\vec{x})$.

1. First assume F over $\wedge, \neg$

$$F = C_1 \wedge C_2 \wedge \ldots \wedge C_m$$

Say $C_i = (x_1 \vee x_2 \vee \bar{x}_3)$

Then rewrite as: $C_i = \neg(\neg C_i) = \neg(\bar{x}_1 \wedge \bar{x}_2 \wedge x_3)$

2. Recursive definition of $g_F(\vec{x})$:

$$\neg A \rightarrow 1 - g_A$$

$$A \wedge B \rightarrow g_A \cdot g_B$$

So for C_i above $\longrightarrow$ $1 - ((1-x_1)(1-x_2)x_3)$

* If F is m-clause 3CNF, $\text{degree}(g_F) \leq 3m$

Now $\#F$ = # satisfying assignments for F

$$= \#\, g_F(\vec{x})$$

$$= \sum_{x \in \{0,1\}^n} g_F(\vec{x}) \bmod p$$

where $p > 2^n$, p prime

So our goal is to give an IP for

$$L = \{(g(\vec{x}), k) \mid \sum_{x_1 \in \{0,1\}} \sum_{x_2 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} g(\vec{x}) = k\}$$

over mod p

where $p > 2^n$

Recursive IP protocol (recurse on # variables in g)

Input : $g(\vec{x}), K$

1. P computes prime p, $2^n < p < 2^{2n}$ and sends to V
2. V checks that p is prime in range (& rejects if not)
3. P proves to V using SubsetSum Protocol

that $\sum_{x_1 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} g(\vec{x}) \bmod p = k$

Subset Sum Protocol

Input: K, p, $g(\vec{x})$ Want to know if $K = \sum_{x \in \{0,1\}^n} g(\vec{x}) \bmod (p)$

Base Case: If $|\vec{x}| = 1$ (g has one variable):

V checks equality $\sum_x g(x) = K \bmod p$ manually

accept iff equality holds

Else: Suppose $|\vec{x}| = n > 1$

P sends $S(x_1)$ [If P is honest: $S(x_1) = \underbrace{\sum_{x_2 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} g(\vec{x}) \bmod p}_{\text{polynomial in one variable } x_1}$]

V checks if $S(0) + S(1) = K$ and rejects if not

V selects random $a \in \{0, 1, \ldots, p-1\}$ and sends to P

P recursively proves to V that

$$S(a) = \sum_{x_2 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} g(a, x_2, \ldots x_n) \bmod p$$

[So call subset sum on $S(a)$, p, $\underbrace{g'(x_2 \ldots x_n) = g(a, x_2 \ldots x_n)}_{n-1 \text{ variables}}$]

Soundness + Completeness of Subset Sum Protocol

Completeness: If $K = \sum_{x \in \{0,1\}^n} g(\vec{x}) \bmod (p) \Rightarrow \exists$ prover s.t. Verifier accepts w.p. 1

honest prover $\Rightarrow$ No step of protocol rejects $1 - \frac{3 \cdot n}{2^n}$ ✓.

Soundness Want to show

if $K \neq \sum g(\vec{x}) \bmod p \Rightarrow \forall P$ Verifier rejects wp $\geq 1 - \frac{3mn}{p}$

Base case $|\vec{x}| = 1$ easy since V checks on their own so verifier rejects wp 1

Ind. hyp: Assume true $\forall g(\vec{x})$, $|\vec{x}| \leq n-1$, if equality fails verifier rejects wp $\geq 1 - \frac{3m(n-1)}{p}$

Soundness + Completeness of Subset Sum Protocol

Soundness Want to show

if $K \neq \sum g(\vec{x}) \bmod p \Rightarrow \forall P$ Verifier rejects wp $\geq 1 - \frac{3mn}{p}$

Ind Step Let $\sum_{x \in \{0,1\}^n} g(\vec{x}) \neq K \bmod p$, $|\vec{x}| = n$

Then protocol can fail (and V accept) for 2 reasons

(1) Recursive step fails $\longrightarrow$ by IH. happens wp $\leq \frac{3m(n-1)}{p}$

(2) Equation chosen by P equals correct equation at $x_1 = a$:

$s(x_1)$: poly returned by prover. supposed to equal $g_1(x_1) := \sum_{x_2 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} g(\vec{x}) \bmod p$

holds only if $\underbrace{\sum_{x_2} \cdots \sum_{x_n} g(\vec{x}) - s(x_1)}$ has a as its root

has degree at most $3m$, so has at most $3m$ roots.

$\therefore$ random $a \sim \{0, \ldots p-1\}$ is a root with prob $\leq \frac{3m}{p}$

$\therefore$ Total failure prob is $\leq \frac{3mn}{p} < \frac{1}{n}$ $(p \geq 2^n)$

$S(x_1)$ is returned by prover.

$S(x_1)$ is supposed to equal $g_1(x_1) := \sum_{x_2} \sum_{x_3} \dots \sum_{x_n \in \{0,1\}^n} g(x_1 \dots x_n)$

This is a PIT poly identity testing problem!

Next Verifier picks random $a \in \{0, \dots, p-1\}$ $2^{2n} > p > 2^n$

then check if:

$$S(a) = \sum_{x_2} \dots \sum_{x_n} g(a, x_1 \dots x_n) \quad (*)$$

(*) holds iff the poly

$p := S(x_1) - \sum_{x_2} \sum_{x_3} \dots \sum_{x_n} g(x_1 x_2 \dots x_n)$ has 0 as a root

$\deg(p) \leq 3m$

$\therefore$ # roots of $p \leq 3m$.

We pick a at random from $\{0 \dots p-1\}$

$\therefore$ prob (*) holds if $S(x_1) \neq g_1(x_1)$ is $\leq \frac{3m}{p}$

Example: F is CNF over x_1, x_2, x_3, x_4 $(n = 4)$. Problem: $P_F(x_1 x_2 x_3 x_4) \stackrel{?}{=} K$

V asks for $h_1(x_1) = \sum_{b_2} \sum_{b_3} \sum_{b_4} P_f(x_1, b_2 b_3 b_4)$

P sends $s_1(x_1)$

$s_1(0) + s_1(1) \neq K$

$s_1(0) + s_1(1) = K$

V picks random $a_1 \in [1, \ldots, 2^n]$

V asks for $h_2(x_2) = \sum_{b_3} \sum_{b_4} P_F(a_1, x_2, b_3 b_4)$

P sends $s_2(x_2)$

$s_2(0) + s_2(1) \neq s_1(a_1)$

$s_2(0) + s_2(1) = s_1(a_1)$

Example: F is CNF over x_1, x_2, x_3, x_4 $(n = 4)$. Problem: $P_F(x_1 x_2 x_3 x_4) \stackrel{?}{=} K$

V asks for $h_1(x_1) = \sum_{b_2} \sum_{b_3} \sum_{b_4} P_f(x_1 b_2 b_3 b_4)$
P sends $s_1(x_1)$

$K \stackrel{?}{=} \sum_{b_1 b_2 b_3 b_4} P_F(b_1 b_2 b_3 b_4)$

$s_1(0) + s_1(1) \neq K$ → Reject

$s_1(0) + s_1(1) = K$

V picks random $a_1 \in [1 \ldots 2^n]$
V asks for $h_2(x_2) = \sum_{b_3} \sum_{b_4} P_f(a_1 x_2 b_3 b_4)$
P sends $s_2(x_2)$

$s_1(a_1) \stackrel{?}{=} \sum_{b_2 b_3 b_4} P_F(a_1 b_2 b_3 b_4)$

$s_2(0) + s_2(1) \neq s_1(a_1)$ → Reject

$s_2(0) + s_2(1) = s_1(a_1)$

V picks a_2
V asks for $h_3(x_3) = \sum_{b_4} P_F(a_1 a_2 x_3 b_4)$
P sends $s_3(x_3)$

$s_2(a_2) \stackrel{?}{=} \sum_{b_3 b_4} P_F(a_1 a_2 b_3 b_4)$

$s_3(0) + s_3(1) \neq s_2(a_2)$ → Reject

V picks a_3
V asks for $h_4(x_4) = P_F(a_1 a_2 a_3 x_4)$
P sends $s_4(x_4)$

$s_3(a_3) \stackrel{?}{=} \sum_{b_4} P_F(a_1 a_2 a_3 b_4)$

$s_4(0) + s_4(1) \neq s_3(a_3)$ → Reject

accept if $s_4(x_4) = h_4(x_4)$

Approximation Algorithms & the PCP Theorem

So far we have mostly focused on decision problems,
or equivalently problems with yes/no answer
e.g. Is CNF F satisfiable?
Does G have a size k clique?

Optimization Problems:

MaxSAT : given CNF F, Find assignment that maximizes # of accepting clauses

minCOL : given G, Find a proper coloring that minimizes # of colors

TSP : given directed graph with edge distances
Find shortest path that visits every city

Approximation Algorithms & the PCP Theorem

NP-completeness : very important method to establish conditional hardness of solving an optimization problem exactly

If $\text{maxSAT} \in P \Rightarrow 3\text{SAT} \in P$

If $\text{minCOL} \in P \Rightarrow 3\text{COL} \in P$

What about complexity of solving an optimization problem approximately ?

Approximation Algorithms & the PCP Theorem

Approximation Versions:

An α-approximation alg for MaxSAT: $\forall$ 3CNF F, output an assignment satisfying at least $\alpha \cdot \underbrace{OPT(F)}_{\text{value of optimal assignment}}$

MaxSAT is maximization problem so $\alpha \leq 1$

An α-approx alg for minCOL: given g, output a proper coloring of g that uses $\leq \alpha \cdot \underbrace{OPT(g)}_{\text{min \# colors to properly color } g}$

MINCOL is minimization problem so $\alpha \geq 1$

How to prove hardness of finding a good approximation alg?

Typical NP-hardness is too brittle!

generic polytime Nondet TM M $\xrightarrow[\text{polytime}]{\Pi}$ 3SAT

given input x to M $\longrightarrow$ $\Pi(x) = \phi_x$

such that ϕ_x satisfiable iff x accepted by M

To show 3SAT hard to .9-approximate we need reduction to satisfy:

$\forall x$ if M accepts x $\Rightarrow$ ϕ_x is satisfiable

if M rejects x $\Rightarrow$ NO assignment satisfies .9 fraction of clauses

How to prove hardness of finding a good approximation alg?

Generic polytime Nondet TM M $\xrightarrow[\text{polytime}]{}$ 3SAT

To show 3SAT hard to .9-approximate we need reduction to satisfy:

$\forall x$ if M accepts x, $\Rightarrow$ φ_x is satisfiable

if M rejects x $\Rightarrow$ no assignment satisfies .9 fraction of clauses

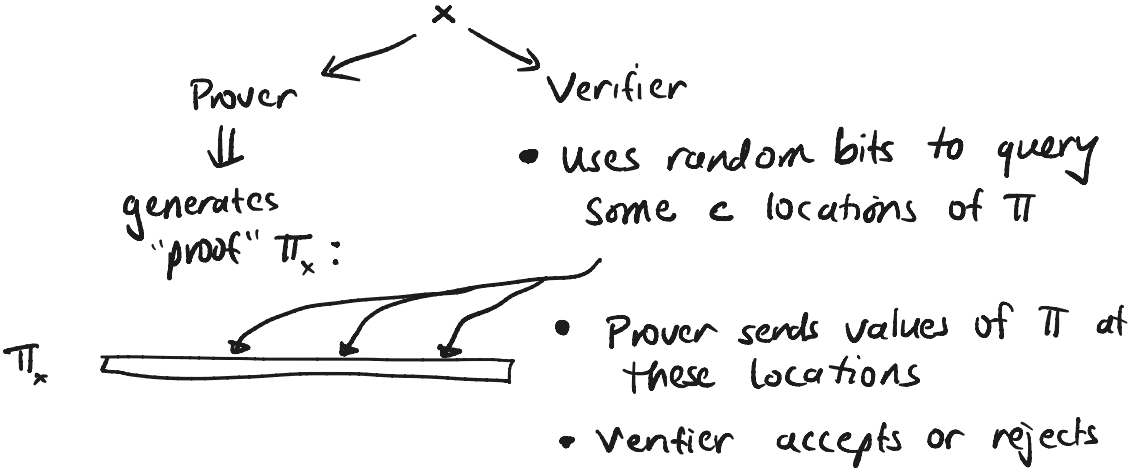
TMs are too brittle: its easy even when M rejects x to construct an assignment satisfying all but a few clauses of φ_x

PCP theorem

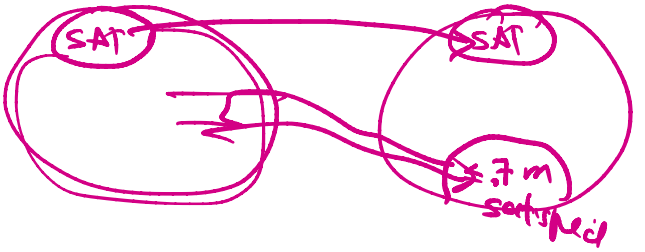
PCP theorem is a "robust" characterization of the class NP.

Idea: comes from Interactive Proofs

Algs/protocols are one-round, and we want verifier to use few random bits and to probe provers proof at only a few locations



Think of it like this.



Initially 2 kinds of 3CNF's F:

case 1 F's that have an assignment α satisfying all clauses

case 2 remain F's ~~that~~ where all α satisfy $\leq m-1$ of the m clauses

Want a reduction $3SAT \xrightarrow{\Pi} 3SAT$

- with the property:

F is sat $\Rightarrow \Pi(F) = g$ is sat

F is unsat $\Rightarrow \Pi(F) = g'$ ~~only~~ all ass's satisfy at most .7 of the clauses

More formally

Verifier $V(x, \pi, b_1 \ldots b_{r(n)})$ is probabilistic and runs in time polynomial in $|x|$



V is $(r(n), q(n))$ - restricted if:

1. V has access to $O(r(n))$ random bits $b_1 \ldots b_{r(n)}$

2. $\forall x \in \{0,1\}^n$, $\forall \pi$,
   $V(x, \pi, \vec{b})$ makes at most $O(q(n))$ queries into π

Defn $L \in PCP[r(n), q(n)]$ if $\exists$ an $(r(n), q(n))$-restricted verifier V such that:

① completeness: $x \in L \Rightarrow \exists \pi$ s.t. $\Pr_{\vec{b}}[V(x, \pi, \vec{b}) = 1] = 1$

② Soundness: $x \notin L \Rightarrow \forall \pi \ \Pr_{\vec{b}}[V(x, \pi, \vec{b}) = 1] \leq \frac{1}{2}$

Theorem (PCP Theorem) $NP = PCP[\log n, 1]$

Theorem (PCP Theorem) $NP = PCP[\log n, 1]$

Comments

1. Since number of random bits used by Verifier is $O(\log n)$, V can be derandomized: simulate $V(x, \pi, *)$ on all $2^{r(n)} = \text{poly}(n)$ random bits and accept iff all runs accept.
   So one direction follows from this ($PCP[\log n, 1] \subseteq NP$)
2. Other direction: $NP \subseteq PCP[\log n, 1]$ hard. Breakthrough result!

   Applications: • Hardness of approx for almost all important NP approx problems] Next Class

   • Crypto - SNARKs
   • Error correcting codes
   • Complexity theory – PRGs
   • Quantum analogs

Recent Announcements (relevant to course)

→ A stronger form of PCP Theorem "UGC" The Unique Games Conjecture is true! [Not yet verified fully]

→ ETH is false