Lecture 6 — Complexity Theory II: Interactive Proofs, IP=PSPACE, and PCP

> Course **2-INF-135/15 Pravdepodobnostné algoritmy**, LS 2025/26. Source slides: `RA_zlozitostII_tlac.pdf` (34 pages) and `RA_slidy_IPaPSPACEPCPstart_tlac.pdf` (38 pages). Štátnicové syllabus topics covered here: *the polynomial hierarchy; BPP inside the hierarchy (Sipser–Gács* $\mathbf{BPP} \subseteq \Sigma_2^p \cap \Pi_2^p$*); interactive proofs IP, public vs. private coins, Arthur–Merlin classes AM/MA; graph non-isomorphism* $\mathbf{GNI} \in \mathbf{AM}[2]$ *and the set–lower-bound protocol;* $\mathbf{IP} = \mathbf{PSPACE}$ *via arithmetization and SumCheck; the PCP theorem* $\mathbf{NP} = \mathbf{PCP}[\log n, O(1)]$*, linearity testing, self-correction, and inapproximability.*

The one-paragraph map of the whole lecture

So far "randomness" lived **inside the algorithm**. This lecture puts randomness **inside a verifier** and asks what a coin-flipping checker can be convinced of. Three movements, each a famous theorem.

1. **Where does BPP sit?** A randomized decider with two-sided error is not obviously anywhere in the classical hierarchy. **Sipser–Gács** pins it down: $\mathbf{BPP} \subseteq \Sigma_2^p \cap \Pi_2^p$ — only **two quantifier alternations** above P. The whole proof is one clever idea: *a large set of random witnesses can be shifted a few times to cover the entire cube, a small one never can.*

2. **What can a prover convince a randomized verifier of?** If the prover is all-powerful and the verifier flips coins and interacts, the answer is *astonishingly large*: $\mathbf{IP} = \mathbf{PSPACE}$ (Shamir). The engine is **arithmetization** — turn a Boolean formula into a polynomial — plus the **SumCheck protocol**, where the verifier pins down a giant exponential sum by asking for one low-degree polynomial per variable and spot-checking it at a random point. The same idea handles graph non-isomorphism cheaply ($\mathbf{GNI} \in \mathbf{AM}$), which is *evidence that graph isomorphism is not NP-complete*.

3. **How short and how local can a proof be?** Shockingly local. The **PCP theorem** says $\mathbf{NP} = \mathbf{PCP}[O(\log n), O(1)]$: every NP statement has a proof you can verify by flipping $O(\log n)$ coins and reading only a **constant number of bits**. Its alter ego is the **hardness of approximation**: getting a good approximation for Max-3-SAT (or Max-Clique / Independent Set) is as hard as solving it exactly. We build the bottom layer by hand: **linearity testing**, **self-correction**, and arithmetization give $\mathbf{NP} \subseteq \mathbf{PCP}[O(n^3), O(1)]$.

The unifying thread: **arithmetization + a random spot-check**. Convert "is this formula satisfied / how many ways" into "does this polynomial equal that value," then exploit the one magic fact about low-degree polynomials — *two different degree-d polynomials agree on at most d points, so a random point exposes any lie.*

Part I — The polynomial hierarchy and BPP

1. The polynomial hierarchy (PH)

NP is "∃ a short witness, checkable in poly time." coNP is "∀ short witnesses…". The **polynomial hierarchy** is what you get by **stacking quantifier alternations**. Fix a poly-time machine M and a polynomial q.

$$\Sigma_2^p: \quad x \in L \iff \exists y \in \{0,1\}^{q(|x|)}\ \forall z \in \{0,1\}^{q(|x|)}\ M(x,y,z)=1,$$

$$\Pi_2^p: \quad x \in L \iff \forall y \in \{0,1\}^{q(|x|)}\ \exists z \in \{0,1\}^{q(|x|)}\ M(x,y,z)=1.$$

In general $L \in \Sigma_i^p$ if there is a poly-time M and polynomial q with

$$x \in L \iff \exists u_1\, \forall u_2\, \exists u_3 \cdots Q_i u_i \;\; M(x,u_1,\ldots,u_i)=1,$$

with i alternating quantifiers (the $u_j \in \{0,1\}^{q(|x|)}$), starting with ∃. The class $\Pi_i^p = \mathrm{co}\Sigma_i^p$ starts with ∀. Then

$$\Sigma_1^p = \mathrm{NP}, \quad \Pi_1^p = \mathrm{coNP}, \qquad \mathrm{PH} = \bigcup_{i \geq 1} \Sigma_i^p.$$

> **Intuition.** Think of a **two-player game with a bounded number of moves**. "$\exists y\, \forall z$" is "I (the prover) make a move y; you (the refuter) reply z; I win iff M accepts." Σ_i = the existential player moves first and the game lasts i rounds. PH measures *how many rounds of this debate* you need.

A concrete Σ_2^p statement worth keeping: **"k is the size of the maximum independent set."** "∃ an IS of size k" is plain NP. "k is the *maximum*" needs a second quantifier: ∃ an IS of size k such that ∀ vertex sets of size $k+1$, that set is not independent — an ∃∀, i.e. Σ_2^p statement.

> **Punchline — the collapse principle.** The hierarchy is believed to be strict (infinitely many genuinely harder levels), but it is **fragile from below**:
> $\Sigma_i^p = \Pi_i^p \;\Rightarrow\; \mathrm{PH} = \Sigma_i^p \quad (\text{collapse to level } i), \qquad \mathrm{P} = \mathrm{NP} \;\Rightarrow\; \mathrm{PH} = \mathrm{P}.$ One coincidence

> anywhere makes everything above it cave in. "PH collapses" is therefore the standard way to say "*this would be a shocking coincidence*" — and it is exactly the currency we will pay in §6 (if GI were NP-complete, PH collapses).

2. Adleman & the target: where is BPP?

Two facts frame the question. First, the **easy non-uniform bound** (proved earlier in the course):

> **Theorem (Adleman).** $\mathrm{BPP} \subseteq \mathrm{P/poly}$.

i.e. with polynomial *advice* (a good random string hard-wired per input length), randomness is free. But P/poly contains undecidable languages — it is too coarse to say BPP is "almost P." We want a **uniform** bound. The headline:

> **Theorem (Sipser–Gács–Lautemann).** $\mathrm{BPP} \subseteq \Sigma_2^p \cap \Pi_2^p$.

So a two-sided-error randomized class sits just **two alternations** above P — strong evidence BPP is genuinely low (and consistent with the belief BPP = P).

3. The Sipser–Gács proof — covering the cube by shifts

Let M be a BPP machine for L using $m = p(n)$ random bits. First **amplify** the success probability (run many times, majority vote) until the error is below 2^{-n}:

$$x \in L \Rightarrow |S_x| \geq (1 - 2^{-n})\, 2^m, \qquad x \notin L \Rightarrow |S_x| < 2^{\,m-n},$$

where $S_x = \{r \in \{0,1\}^m : M(x,r) = 1\}$ is the set of **accepting random strings**. So "$x \in L$" $\approx$ "S_x is almost everything"; "$x \notin L$" $\approx$ "S_x is a tiny 2^{-n} sliver."

The trick is to detect "big vs. tiny" with **two quantifiers** using **XOR-shifts**. For $t \in \{0,1\}^m$ let $S_x \oplus t = \{r \oplus t : r \in S_x\}$ — the set translated by t (a bijection of the cube, so $|S_x \oplus t| = |S_x|$). Set $k = \lceil m/n \rceil$.

> **Claim.** $x \in L \iff \exists\, t_0, \ldots, t_k \in \{0,1\}^m \; \forall r \in \{0,1\}^m : \bigvee_{i=0}^{k} \left[r \in S_x \oplus t_i \right]$ — i.e. **a few shifts of S_x cover the entire cube.** Equivalently $\forall r \; \exists i : \; M(x,\, r \oplus t_i) = 1$.

This is literally a Σ_2^p formula ($\exists$ shifts $\forall r$). Two directions:

If $x \notin L$ (S_x tiny), no shifts can cover. A union of $k+1$ shifts has size at most $\left|\bigcup_{i=0}^{k}(S_x \oplus t_i)\right| \le (k+1)\,|S_x| \le \left(1+\lceil \frac{m}{n} \rceil\right) 2^{\,m-n} < 2^m$ for $n \ge 2$ — strictly smaller than the cube. So **for every** choice of shifts some r is missed; the $\forall r$ clause fails. (This is the Π_2 half: $x \notin L \Rightarrow \forall t_0 \ldots t_k\, \exists r\, \forall i\, M(x, r \oplus t_i) = 0$.)

If $x \in L$ (S_x huge), random shifts cover w.h.p. Pick $t_0, \ldots, t_k$ uniformly. For a fixed r, $\Pr[\, r \notin S_x \oplus t_i \,] = \Pr[\, r \oplus t_i \notin S_x \,] = 1 - \frac{|S_x|}{2^m} \le 2^{-n}$, and the t_i are independent, so $\Pr[\, r \text{ uncovered by all } i \,] \le 2^{-n(k+1)}$. Union bound over the 2^m values of r: $\Pr[\exists r \text{ uncovered}] \le 2^m \cdot 2^{-n(k+1)} = 2^{\,m-n(k+1)} < 1$, since $n(k+1) \ge n(\frac{m}{n}+1) = m+n > m$. Probability of failure < 1 means **some** choice of shifts covers everything — the $\exists$ is satisfiable. ■

> **Professor-pleaser — the one idea.** The whole theorem is a **set-size test with quantifiers**: a set that fills almost the whole cube can be *translated $k+1$ times to tile the cube*; a set that is an exponentially small sliver provably *cannot* — its $k+1$ copies don't have enough total mass. "$\exists$ shifts that cover" is a Σ_2 sentence, and because BPP is closed under complement we also get the Π_2 sentence, hence $\mathbf{BPP} \subseteq \Sigma_2^p \cap \Pi_2^p$. The probabilistic method ("failure probability $< 1 \Rightarrow$ a good object exists") is doing the existential work — the same move as everywhere else in the course, now used to populate a quantifier.

Part II — Interactive proofs

4. What an interactive proof is

A classical NP-proof is a static string the verifier reads once. An **interactive proof** is a *conversation*: a (possibly cheating) **prover** P of unbounded computational power tries to convince a **probabilistic, polynomial-time verifier** V that $x \in L$. They exchange messages; **the verifier has the last word**.

> **The cast.** V = "the examiner": poly-time, flips private coins. P = "the student / oracle": unlimited power, **does not see V's random bits** (in the private-coin model). The verifier must be convincible by a truthful prover and un-foolable by a lying one.

$$x \in L \;\Rightarrow\; \exists P:\ \Pr_r[\, V \text{ accepts in } (V,P) \,] \ge \tfrac{2}{3} \quad \textbf{(completeness)},$$

$$x \notin L \;\Rightarrow\; \forall P:\ \Pr_r[\, V \text{ accepts in } (V,P) \,] < \tfrac{1}{3} \quad \textbf{(soundness)}.$$

$\mathbf{IP}$ = languages with such a protocol (poly many rounds). As always, the $\frac{2}{3}/\frac{1}{3}$ gap can be amplified to $1-2^{-n}/2^{-n}$ by repetition.

> **Why this is more powerful than NP.** In NP the prover commits to one fixed proof. Here the verifier asks *random, unpredictable questions*: the prover would have to have prepared a consistent answer to all of them at once, which a cheating prover cannot. The randomness is the verifier's leverage. (NP = IP with **zero** coins and **one** message.)

5. Public vs. private coins; Arthur–Merlin

The protocol above uses **private coins**: P never learns r. A weaker-looking model uses **public coins** — the verifier's messages *are* its random bits, sent in the clear. This is the **Arthur–Merlin** model (Arthur = the poor mortal king who can only flip coins; Merlin = the all-knowing wizard/prover).

- $\mathbf{MA}$ ("**M**erlin then **A**rthur"): Merlin sends a proof y, then Arthur flips z and checks.
  $x \in L \Rightarrow \exists y \ \Pr_z[V(x,y,z)=1] \geq \frac{2}{3}, \qquad x \notin L \Rightarrow \forall y \ \Pr_z[V(x,y,z)=1] < \frac{1}{3}.$
- $\mathbf{AM} = \mathbf{AM}[2]$ ("**A**rthur then **M**erlin"): Arthur flips y and sends it, Merlin replies z, Arthur checks **deterministically**.
  $x \in L \Rightarrow \Pr_y[\exists z \, V(x,y,z)=1] \geq \frac{2}{3}, \qquad x \notin L \Rightarrow \Pr_y[\forall z \, V(x,y,z)=0 \text{ w.h.p.}].$

Public coins look strictly weaker — surely hiding the coins helps the verifier? The surprise:

> **Theorem (Goldwasser–Sipser).** $\mathbf{IP}[k] \subseteq \mathbf{AM}[k+2]$. Private coins buy almost nothing: any private-coin protocol becomes a public-coin one with two extra rounds.

The mechanism: instead of secretly tossing r and acting on it, Arthur tosses **public** coins, Merlin is then asked to *certify properties of those coins* (e.g. "this random hash hit the set") without further use of hidden randomness. The set–lower-bound protocol below is exactly this in action.

6. Graph Non-Isomorphism is in AM — the set lower-bound protocol

The poster child: **GNI** (given G_1, G_2, decide $G_1 \not\cong G_2$). GNI is in coNP (a single isomorphism refutes "non-isomorphic"), but no short *certificate of non-isomorphism* is known — yet it has a 2-message Arthur–Merlin proof.

> **Theorem.** $\mathrm{GNI} \in \mathbf{AM}[2]$.

The idea: count by hashing. Consider the set of "graphs reachable from the inputs together with a witnessing automorphism": $S = \{(H, \pi) \ : \ H \cong G_1 \text{ or } H \cong G_2,\ \pi \in \mathbf{aut}(H)\}$**.** For each input G_i, the number of distinct labelings $H \cong G_i$ is $n!/|\mathbf{aut}(G_i)|$, and each such H contributes $|\mathbf{aut}(H)| = |\mathbf{aut}(G_i)|$ pairs — so G_i alone contributes exactly $n!$ pairs. Therefore $G_1 \cong G_2 \Rightarrow |S| = n!, \qquad G_1 \ncong G_2 \Rightarrow |S| = 2 \cdot n!$**. Non-isomorphism doubles the set.** If Arthur can tell "size $n!$" from "size $2n!$" with Merlin's help, he's done. That is the generic problem:

> **Set lower-bound problem.** P, V both know a set $S \subseteq \{0,1\}^m$ (membership in S is certifiable) and a threshold K. We want a protocol where
>
> - if $|S| \geq K$, the prover convinces V w.h.p.;
> - if $|S| \leq K/2$, V rejects w.h.p. (Here $K = 2 \cdot n!$ vs. $|S| = n! = K/2$.)

The protocol uses pairwise-independent (universal) hashing. Pick k with $2^{k-2} < K \leq 2^{k-1}$ and a hash family $\mathcal{H}(m,k)$ of functions $h : \{0,1\}^m \to \{0,1\}^k$ that is **pairwise independent**: for any $x \neq x'$, the pair $(h(x), h(x'))$ is uniform on $\{0,1\}^k \times \{0,1\}^k$.

> **Protocol (one round, public coin).**
>
> 1. **Arthur** picks $h \in_R \mathcal{H}(m,k)$ and $y \in_R \{0,1\}^k$, sends (h, y).
> 2. **Merlin** sends some $x \in S$ with $h(x) = y$, plus a certificate that $x \in S$.
> 3. **Arthur** accepts iff the certificate checks and $h(x) = y$.

Arthur is asking: "*does the random hash bucket y contain an element of S?*" A big S hits a random bucket; a small S usually misses.

> **Lemma.** Let $h \in \mathcal{H}(m,k)$, $S \subseteq \{0,1\}^m$ with $|S| \leq 2^{k-1}$, and $p = |S|/2^k$. Then $p \ \geq \ \Pr_{h,y}[\exists x \in S : \ h(x) = y] \ \geq \ \frac{3}{4}\, p$**.** (Even pointwise: for every fixed y, $\Pr_h[\exists x \in S : h(x) = y] \geq \frac{3}{4} p$.)

Proof of the lower bound (inclusion–exclusion / Bonferroni). Let E_x be the event $h(x) = y$. Then $\Pr[E_x] = 2^{-k}$ and (pairwise independence) $\Pr[E_x \cap E_{x'}] = 2^{-2k}$. By Bonferroni, $\Pr\left[\bigcup_{x \in S} E_x\right] \ \geq \ \sum_{x \in S} \Pr[E_x] - \frac{1}{2} \sum_{x \neq x'} \Pr[E_x \cap E_{x'}] = \frac{|S|}{2^k} - \frac{1}{2}\frac{|S|^2}{2^{2k}} = p - \frac{1}{2}p^2 \ \geq \ \frac{3}{4}p,$ using $p \leq \frac{1}{2}$. The upper bound p is the plain union bound. ■

So the acceptance probability is $\geq \frac{3}{4}p$ **when** $|S| \geq K$ and $\leq p$ **when** $|S| \leq K/2$ (where the relevant p halves) — a constant multiplicative gap. With the calibration $p^* = K/2^k$, Arthur accepts iff the fraction of accepting repetitions is at least $\frac{5}{8}p^*$; **Chernoff** over a constant number of repetitions pushes the gap to $\frac{2}{3}$ vs. $\frac{1}{3}$. ■

> **Professor-pleaser.** GNI has *no known short classical certificate*, yet a verifier who can flip coins decides it in **two messages** — by reducing "are these graphs non-isomorphic" to "**is this set twice as big**," and deciding *that* by checking whether a random hash bucket is occupied. This is the prototype of "*interaction + randomness > static proof.*"

7. The structure of AM and MA

A run of small but important structural facts (state them; the proofs reuse the Sipser–Gács shift trick).

Perfect completeness. Both MA and AM can be made to **never reject a true statement**: $\mathbf{MA}$**:** $x \in L \Rightarrow \exists y\ \Pr_z[V(x,y,z)=1]=1;$ $\qquad$ $\mathbf{AM}$**:** $x \in L \Rightarrow \Pr_y[\exists z\, V(x,y,z)=1]=1.$ The proof is *exactly* the covering-by-shifts argument of §3: amplify so accepting random strings fill almost the cube, then let the prover supply $k+1$ shifts $t_0,\dots,t_k$ that cover it; Arthur checks he accepts on some $r \oplus t_i$.

Collapse of the hierarchy of rounds. Public-coin interaction with a *constant* number of rounds gives nothing beyond two: $\mathbf{MA} \subseteq \mathbf{AM},$ $\qquad$ $\mathbf{AM}[k]=\mathbf{IP}[k]=\mathbf{AM}[2]=\mathbf{AM}\quad(k=O(1)).$ ("$\mathbf{MAM}=\mathbf{AM}$, $\mathbf{AMM}=\mathbf{AM}$" — adjacent same-player moves merge, and a leading Merlin can be swapped past an Arthur.) So for constant rounds there is essentially **one** Arthur–Merlin class, AM.

Location in PH. Unwinding the quantifiers: $\mathbf{MA} \subseteq \Sigma_2^p \cap \Pi_2^p,$ $\qquad$ $\mathbf{AM} \subseteq \Pi_2^p.$ Sketch for MA $\subseteq \Pi_2^p$ vs. the matching Σ_2 characterization: a perfectly complete MA is "$x \in L \Rightarrow \exists m\, \forall r\ V(x,m,r)=1$" (that's Σ_2), while "$x \notin L \Rightarrow \forall m\, \exists r\ V(x,m,r)=0$" gives the Π_2 side. AM is "$x \in L \Rightarrow \forall r\, \exists m$" / "$x \notin L \Rightarrow \exists r\, \forall m$" — a Π_2 pattern.

> **Theorem (Boppana–Håstad–Zachos).** If $\mathbf{coNP} \subseteq \mathbf{AM}$, then $\mathbf{PH}=\Sigma_2^p$ (the hierarchy collapses to the second level).

Idea. Since $\mathbf{AM} \subseteq \Pi_2^p$, it suffices to show $\Sigma_2^p \subseteq \mathbf{AM}$ under the hypothesis. Take $L \in \Sigma_2^p$, so $x \in L \iff \exists y\,(x,y) \in L'$ with $L' \in \Pi_1^p = \mathbf{coNP}$. Build an **MAM** protocol: Merlin sends y, then run the assumed AM protocol for the coNP statement $(x,y) \in L'$. Thus $L \in \mathbf{MAM}=\mathbf{AM}$, so $\Sigma_2^p \subseteq \mathbf{AM} \subseteq \Pi_2^p$, forcing the collapse. ■

> **Corollary (evidence GI is not NP-complete).** If $\mathbf{GI} \in \mathbf{NPC}$, then $\mathbf{PH}=\Sigma_2^p$.

> *Why:* if GI were NP-complete, then its complement GNI would be **coNP-complete**; since $\mathbf{GNI} \in \mathbf{AM}$ (§6) and AM is closed appropriately, every coNP language reduces into AM, i.e. $\mathbf{coNP} \subseteq \mathbf{AM}$, and Boppana–Håstad–Zachos collapses PH. Almost nobody believes PH collapses — so almost nobody believes GI is NP-complete. **This is the precise sense in which "graph isomorphism is probably not NP-complete," and AM is the tool that says it.**

Part III — IP = PSPACE

The crowning theorem of interactive proofs: an all-powerful prover and a coin-flipping verifier can settle **any** problem solvable in polynomial space.

> **Theorem (Shamir 1990).** $\mathrm{IP} = \mathrm{PSPACE}$.

Two inclusions.

8. The easy direction: IP ⊆ PSPACE

Given an IP protocol, we want to decide, in polynomial space, whether the **best** prover makes V accept with probability $\geq \frac{2}{3}$. The interaction is a **game tree**: at each prover move we **maximize**, at each verifier coin-flip we **average**. The verifier uses $r(n)$ private random bits and the prover sends messages of length $m(n)$; both polynomial. We can evaluate the optimal acceptance probability by recursively walking this tree — $\forall r \in \{0,1\}^{r(n)}$, simulating the V–P communication for each prover response $m \in \{0,1\}^{m(n)}$ — reusing space across branches. Polynomial space suffices because the tree has polynomial depth and we never store a whole level. ■

9. The hard direction: PSPACE ⊆ IP — arithmetization

We give an interactive protocol for a **PSPACE-complete** problem, **TQBF** (true fully quantified Boolean formulas). Warm up on counting first.

9a. Arithmetization — Boolean formulas become polynomials

Replace logic by arithmetic over the integers (later reduced mod a prime):
$\neg x \rightsquigarrow (1-x), \qquad x \wedge y \rightsquigarrow x \cdot y, \qquad x \vee y \rightsquigarrow 1-(1-x)(1-y).$ A formula $\varphi(x_1,\dots,x_n)$ with m clauses becomes a polynomial $P_\varphi(X_1,\dots,X_n) = \prod_{j \leq m} p_j(X_1,\dots,X_n)$ that agrees with φ on $\{0,1\}^n$: $\varphi(x_1,\dots,x_n) = 1 \iff P_\varphi(x_1,\dots,x_n) = 1.$ Example:
$\varphi = (x \vee y \vee z) \wedge (\neg x \vee y \vee z) \wedge (x \vee \neg y \vee z)$ arithmetizes to
$P_\varphi = \big(1-(1-X)(1-Y)(1-Z)\big)\big(1-X(1-Y)(1-Z)\big)\big(1-(1-X)Y(1-Z)\big).$

The number of satisfying assignments of φ is then an **exponential sum**: $\#\varphi = \sum_{b_1\in\{0,1\}}\cdots\sum_{b_n\in\{0,1\}} P_\varphi(b_1,\dots,b_n)$. So $\#3\mathrm{SAT}_D = \{(\varphi, K) : \varphi$ has exactly K satisfying assignments$\}$ becomes: "$\sum_b P_\varphi(b) = K$." We will verify such a sum interactively.

9b. The SumCheck protocol — the heart of everything

We must convince a poly-time verifier that $K \equiv_p \sum_{b_1\in\{0,1\}}\cdots\sum_{b_n\in\{0,1\}} g(b_1,\dots,b_n)$, where g is a polynomial of degree $\le d$ in each variable, computed mod a **prime** p. The verifier cannot evaluate the 2^n-term sum directly. Instead it **peels one variable at a time** and trusts the prover for a single-variable polynomial, then **spot-checks at a random point**.

> **SumCheck (prime p, per-variable degree d).**
>
> - **If $n = 1$:** V checks $g(0) + g(1) = K$ and accepts/rejects.
> - **If $n \ge 2$:** V asks P for the univariate polynomial $h(X_1) = \sum_{b_2,\dots,b_n\in\{0,1\}} g(X_1, b_2, \dots, b_n)$. P sends a polynomial $s(X_1)$ (claiming $s = h$), with $s(0) + s(1) = K$.
> - **V rejects if $s(0) + s(1) \ne K$.** Otherwise it picks a random $a \in_R \mathrm{GF}(p)$ and **recurses**, verifying $s(a) \equiv_p \sum_{b_2,\dots,b_n\in\{0,1\}} g(a, b_2, \dots, b_n)$ — a SumCheck on $n-1$ variables with new target $K' = s(a)$.

> **Why it is sound (the magic fact).** If the prover lied, $s \ne h$ as polynomials. Two distinct degree-$\le d$ polynomials agree on **at most d points**, so a uniformly random a has $s(a) = h(a)$ with probability $\le d/p$. Unless the verifier is unlucky, the lie *propagates* into the recursive call (now the prover must defend a false $K' = s(a) \ne h(a)$). Over n levels, $\Pr[V \text{ rejects a false claim}] \ge \left(1 - \frac{d}{p}\right)^n \ge 1 - \frac{nd}{p}$, which is overwhelming for a large enough prime p. The verifier never computes the big sum; it only forces the prover to be **consistent** between each claimed polynomial and a random evaluation of the next.

This already gives $\#3\mathrm{SAT}_D \in \mathrm{IP}$ — and $\#3\mathrm{SAT}_D$ is #P-hard, so **interactive proofs already reach the counting hierarchy**.

9c. TQBF ∈ IP — handling quantifiers

A TQBF instance $B = \forall x_1 \exists x_2 \forall x_3 \cdots \exists x_n\, \varphi(x_1,\dots,x_n)$ arithmetizes by turning quantifiers into **sum and product** over $\{0,1\}$: $\exists \rightsquigarrow \sum$, $\forall \rightsquigarrow \prod$, $x \wedge y \rightsquigarrow xy$, $x \vee y \rightsquigarrow x + y$, $\neg x \rightsquigarrow (1 - x)$. Then $B \in \mathrm{TQBF} \iff A = \prod_{b_1}\sum_{b_2}\cdots\sum_{b_n} P_\varphi(b_1,\dots,b_n) \ne 0$. The protocol mirrors SumCheck, but a quantifier value is now defended by a univariate polynomial $g(z)$ with $g(0)\, J\, g(1) = K$, where

$J = +$ for an ∃ ($\sum$) and $J = \cdot$ for a ∀ ($\prod$); after checking, the verifier substitutes a random a and recurses on the inner expression. **Two technical obstacles** must be handled — they are the real content of the proof.

(i) The numbers are astronomically large. A closed QBF of size n can have arithmetization value as big as $O(2^{2^n})$ — because each ∀ (a product) can *square* the magnitude. (E.g. $\prod_{z_1} \cdots \prod_{z_n} (z_1 + \cdots + z_n)$ produces a polynomial of degree 2^{n-1}.) **Fix: work modulo a prime p of polynomial bit-length.** We need $A \not\equiv_p 0 \iff B$ true, i.e. a prime that does *not* divide A. Since $A = O(2^{2^n})$ has at most $\sim 2^n$ prime factors, and there are plenty of poly-length primes, the prover can exhibit a prime p (with a primality certificate) for which $A \not\equiv_p 0$. (If $p_1 \cdots p_m$ all divided A then $\prod p_i = \Omega(2^{2^{n_d}})$ would exceed $A = O(2^{2^n})$ — impossible.)

(ii) The degree blows up. Products of quantifiers raise the polynomial's degree exponentially, and SumCheck needs **low degree**. **Fix: make the QBF "simple."**

> **Simple QBF:** between a variable and *its own* quantifier there is at most **one** ∀ in between. Any QBF can be converted to a simple one with only **polynomial** size growth, by re-introducing each variable as a fresh copy after each ∀ it must "survive": $\ldots x \forall y Q \rightsquigarrow \ldots x \ldots \forall y \exists x_1 \, (x_1 = x) \wedge Q, \qquad (x_i = x_j) \rightsquigarrow z_i z_j + (1 - z_i)(1 - z_j)$. **Key consequence:** if B is simple, the degree of the polynomial describing the "functional form" grows only **linearly** in $|B|$ — so SumCheck stays efficient.

Correctness. With B simple, the degree t of the functional form is linear in $|B|$, and the per-round error d/p accumulates over the rounds, giving B **false** $\Rightarrow \Pr[V \text{ accepts}] \leq \frac{\text{poly}(|B|)}{p}$, negligible for the chosen poly-length prime p. Combined with $\mathbf{IP} \subseteq \mathbf{PSPACE}$ and the PSPACE-completeness of TQBF: $\boxed{\mathbf{IP} = \mathbf{PSPACE}}$

> **A worked micro-example (from the slides).** For $B = \forall x_1 (\neg x_1 \vee \exists x_2 \forall x_3 (x_1 \wedge x_2 \vee x_3))$ the arithmetization is $A = \prod_{z_1} \Big[(1 - z_1) + \underbrace{\sum_{z_2} \prod_{z_3} (z_1 z_2 + z_3)}_{A'} \Big], \qquad K \equiv_p A, \; K = 2.$
>
> Each round: the prover sends a low-degree $g(z)$ with $g(0) \, J \, g(1) = K$ (e.g. $g(z_1) = z_1^2 + 1$ giving $g(0) \cdot g(1) = 1 \cdot 2 = 2 = K$ ✓), the verifier checks it, substitutes a random $z_1 = a$, updates $K \leftarrow$ (the value forced for that branch), and recurses on the inner sum/product — peeling z_2 (an ∃, so use "+"), then z_3 (a ∀, so use "·"), until no quantifiers remain.

> **Professor-pleaser — the whole arithmetization philosophy.** A Boolean formula is a brittle yes/no object; its **polynomial** lift is a rigid algebraic object that *cannot lie locally*. Replace "does this formula hold" with "does this polynomial sum to K mod p," and the single fact "*a nonzero*

> *low-degree polynomial has few roots*" turns one random evaluation into an avalanche of confidence. SumCheck is this fact deployed n times. **This same lift powers the PCP theorem next.**

Part IV — Probabilistically Checkable Proofs (PCP)

10. The PCP model

Now make the verifier **local**: it does not read the whole proof, only a few bits of it, chosen using a few random bits.

> **Definition.** $L \in \mathrm{PCP}[r(n), q(n)]$ if there is a probabilistic poly-time verifier V with oracle access to a proof string Π that
>
> - uses $r(|x|)$ **random bits** and reads $q(|x|)$ **bits of** Π (efficiency);
> - $x \in L \Rightarrow \exists \Pi: \ \Pr_r[V^{\Pi}(x, r) = 1] = 1$ (**completeness**, perfect);
> - $x \notin L \Rightarrow \forall \Pi: \ \Pr_r[V^{\Pi}(x, r) = 1] < \frac{1}{2}$ (**soundness**).

> **A tiny PCP for GNI (to fix the picture).** Let the proof be a giant table $\Pi(H) \in \{0,1\}$ indexed by all n-vertex graphs, intended to say "which input is H isomorphic to." V picks $b \in_R \{0,1\}$ and a random permutation τ, forms $H = \tau(G_b)$, reads the single bit $\Pi(H)$, and checks $\Pi(H) = b$. If $G_0 \not\cong G_1$, an honest table answers correctly every time. If $G_0 \cong G_1$, the graph H could equally have come from either input, so the table is wrong with probability $\geq \frac{1}{2}$. (Reads **one** bit!)

11. The PCP theorem and its two faces

> **Theorem (PCP, Arora–Safra–Lund–Motwani–Sudan–Szegedy, 1992).**
> $\boxed{\mathrm{NP} = \mathrm{PCP}[O(\log n),\, O(1)]}$

Every NP language has proofs that a verifier checks by tossing $O(\log n)$ coins and reading a **constant** number of bits — yet a false statement is caught with probability $\geq \frac{1}{2}$. Two readings, both fundamental.

Face 1 — locally checkable proofs. There exist **robust** proof formats: errors are "spread out" so that inspecting $O(1)$ random symbols already exposes any flaw. A correct proof passes always; a proof of a false statement fails on a constant fraction of the random checks.

Face 2 — hardness of approximation. This is the exam-relevant face.

> **Theorem (gap form).** There is a constant $\rho < 1$ such that for every $L \in \mathbf{NP}$ there is a poly-time f mapping inputs to 3-CNF formulas with
> $x \in L \Rightarrow \mathbf{val}(f(x)) = 1, \qquad x \notin L \Rightarrow \mathbf{val}(f(x)) < \rho,$ where $\mathbf{val}$ = max fraction of simultaneously satisfiable clauses.

> **Corollary.** There is $\rho < 1$ such that a polynomial-time ρ-**approximation for Max-3-SAT implies** $\mathbf{P} = \mathbf{NP}$**.** Consequently (unless P = NP):
> $\mathbf{Max\text{-}3\text{-}SAT} \notin \mathbf{PTAS}, \qquad \mathbf{MaxClique} \notin \mathbf{PTAS}, \qquad \mathbf{Independent\ Set\ is\ hard\ to\ approximate.}$

> **Intuition for "gap = inapproximability."** The reduction creates a **promise gap**: either *all* clauses are satisfiable, or *no assignment beats* ρ. An approximation algorithm that always got within a factor better than ρ could tell these two cases apart — and that decides the original NP problem exactly. **No gap-crossing algorithm can exist unless P = NP.** This is the standard route to *Independent Set is hard to approximate* asked on the exams.

12. Basic PCP containments

Read these as "the model is robust and the constants are tunable":

- $\mathbf{PSPACE} \subseteq \mathbf{PCP}[\mathbf{poly}, \mathbf{poly}]$ — immediate from $\mathbf{IP} = \mathbf{PSPACE}$.
- $\mathbf{PCP}[r(n), q(n)] \subseteq \mathbf{NTIME}\big(2^{O(r(n))} \cdot q(n)\big)$ — the proof has length $\leq q(n) \cdot 2^{r(n)}$ (only that many bits are ever queried), so a nondeterministic machine can guess it. In particular $\mathbf{PCP}[\log n, 1] \subseteq \mathbf{NP}$ and $\mathbf{PCP}[\log n, \mathbf{poly}] = \mathbf{NP}$.
- **Soundness amplifies:** repeating the verifier c times drops soundness error to $1/2^c$.
- We assume a **non-adaptive** verifier (all queried positions fixed up front as a function of the random string); for **constant** query count adaptive = non-adaptive.

13. PCP ⇔ gap-amplifying reductions

The bridge between "local proofs" and "inapproximability" is a single equivalence.

Definition (gap-amplifying / c-zosilňujúca reduction). For $c < 1$, a poly-time f on 3-CNF formulas such that $\varphi \in \mathrm{SAT} \Rightarrow f(\varphi) \in \mathrm{SAT}\ \ (\mathrm{maxSAT}(f(\varphi)) = 1)$,
$\varphi \notin \mathrm{SAT} \Rightarrow \forall \alpha:\ f(\varphi)\ \text{satisfies}\ \le c\text{-fraction of clauses}\ \ (\mathrm{maxSAT}(f(\varphi)) < c)$. It manufactures a **gap** where there was none.

Lemma. $\mathrm{NP} \subseteq \mathrm{PCP}[\log n, 1] \iff$ 3SAT has a gap-amplifying reduction.

(⇐) gap reduction ⇒ local verifier. Given f, the verifier on input φ uses the proof Π = a satisfying assignment of $f(\varphi)$. It picks a **random clause** of $f(\varphi)$ (its three variable indices i, j, k need $O(\log n)$ random bits), reads the **3 bits** $\Pi(i), \Pi(j), \Pi(k)$, and checks the clause. If $\varphi \in \mathrm{SAT}$, $f(\varphi)$ is satisfiable, so an honest Π passes always. If $\varphi \notin \mathrm{SAT}$, every assignment leaves $\ge (1 - c)$ of the clauses unsatisfied, so a random clause rejects with probability $\ge 1 - c$ — amplify to $\frac{1}{2}$. ($O(\log n)$ coins, 3 queries. ✓)

(⇒) local verifier ⇒ gap reduction. Take V for SAT using $c \log n$ random bits and t queries. For **each** random string $r \in \{0,1\}^{c \log n}$ (there are only polynomially many), determine the queried positions $i_1, \ldots, i_t$ and the predicate "V accepts," build a CNF $\varphi'_r(x_{i_1}, \ldots, x_{i_t})$ that is true iff $V(\varphi, r, \cdot)$ accepts, convert it to 3-CNF φ_r with auxiliary variables, and set $f(\varphi) := \bigwedge_{r \in \{0,1\}^{c \log n}} \varphi_r$. If $\varphi \in \mathrm{SAT}$ then (completeness) all φ_r are satisfiable simultaneously, so $f(\varphi) \in \mathrm{SAT}$. If $\varphi \notin \mathrm{SAT}$ then (soundness) every assignment α has $\Pr_r[\varphi_r(\alpha) = 0] \ge \frac{1}{2}$; with $t' = \max_r\{\#\text{clauses in } \varphi_r\}$, at least a $d = 1/(2t')$ fraction of all clauses of $f(\varphi)$ are unsatisfied, so at most $(1 - d)$ are satisfied — a constant gap $c > 1 - d$. ■

Punchline. "*Locally checkable proof*" and "*gap reduction*" are literally the same object viewed two ways: the verifier's accepting condition on a random check **is** a clause, and "caught with constant probability" **is** "constant fraction of clauses violated." This is why the PCP theorem and inapproximability are one statement.

14. Building the bottom layer by hand: NP ⊆ PCP[O(n³), O(1)]

The full PCP theorem composes three pieces:

1. **Exponentially long** proof, $O(1)$ queries — *via arithmetization* (this section).
2. **Polynomially long** proof, **polylog** queries — via low-degree polynomial equality tests.
3. A **composition lemma** stitching (1) into (2) to reach $O(\log n)$ coins, $O(1)$ queries.

We construct step 1: a constant-query verifier for 3-SAT with an exponential proof. The tools are **linear functions and their testability**.

14a. δ-close functions, linear functions

For finite D, R and $f, g : D \to R$, write $\Pr_{x \in D}\{P(x)\} = |\{x : P(x)\}|/|D|$. Then f, g are **δ-close** if $\Pr_{x \in D}[\, f(x) \neq g(x)\,] \leq \delta$. A function $f : \mathbb{Z}_2^m \to \mathbb{Z}_2$ is **linear** if $f(x+y) = f(x) + f(y)$ for all x, y (equivalently $f(x) = \sum_i a_i x_i$ for some coefficient vector a).

> **Lemma (Blum–Luby–Rubinfeld).** Let $\delta < \frac{1}{3}$ and suppose $g : \mathbb{Z}_2^m \to \mathbb{Z}_2$ satisfies $\Pr_{x,y}[\, g(x+y) \neq g(x) + g(y)\,] \leq \delta/2$. Then there is a **linear** f that is δ-close to g, namely the "plurality vote" $f(x) := \textbf{the } b \textbf{ for which } \Pr_y\left[g(x+y) - g(y) = b\right] \geq \frac{1}{2}$.

The proof has three beats: **(1)** f, g are δ-close (else the linearity defect would exceed $\delta/2$, contradiction); **(2)** the vote is overwhelming, $p_a := \Pr_x[f(a) = g(a+x) - g(x)] \geq 1 - \delta$ (bootstrapped from $\geq \frac{1}{2}$ using a pairing/second-moment argument $1 - \delta \leq p_a^2 + (1-p_a)^2 \leq p_a$); **(3)** f is linear, because for fixed a, b, $\Pr_x[\, f(a) + f(b) + g(x) = f(a+b) + g(x)\,] \geq 1 - 3\delta > 0$, and that probability does not depend on x, so it is 0 or 1 — hence 1, i.e. $f(a) + f(b) = f(a+b)$.

14b. Two programs: Linearity Test and Self-Correction

> **Linearity Test (LT).** Input $\delta < \frac{1}{3}$, oracle g. Repeat $k = \lceil 2/\delta \rceil$ times: pick $x, y \in_R \mathbb{Z}_2^m$; if $g(x) + g(y) \neq g(x+y)$ return **NO**. If all pass, return **YES**.
> g linear $\Rightarrow$ YES; $\qquad$ g not δ-close to linear $\Rightarrow \Pr[\text{NO}] \geq \frac{1}{2}$.

> **Self-Correcting (SCF).** Input x; oracle g that is δ-close to a linear f. Pick $y \in_R \mathbb{Z}_2^m$ and return $g(x+y) - g(y)$. $\Pr[\,\text{SCF}(x) = f(x)\,] \;\geq\; 1 - 2\delta$. *Why:* both $g(x+y) \neq f(x+y)$ and $g(y) \neq f(y)$ have probability $\leq \delta$ (each of $x+y$ and y is uniform), and f linear gives $f(x+y) - f(y) = f(x)$; union bound.

> **Intuition.** A table that is *mostly* a linear function can be **read reliably at any single point** by averaging over a random "detour" y — even at points where the table is corrupted. The corruption is diluted because $x+y$ and y are each individually uniform. *Self-correction is what lets a verifier trust an untrusted, slightly-wrong table by reading just two of its entries.*

14c. Arithmetizing 3-SAT over $\mathbb{Z}_2$, and the proof format

A different arithmetization, tuned for linearity. For a literal: $u \mapsto p_u = 1 - x_u$, $\neg u \mapsto p_u = x_u$ (so $p_{\text{literal}} = 0$ exactly when the literal is **true**). A clause $C = l_1 \vee l_2 \vee l_3 \mapsto P_C = p_{l_1} p_{l_2} p_{l_3}$ (degree 3; $P_C = 0$ iff C is satisfied). A formula $\Phi = C_1 \wedge \cdots \wedge C_m$ is satisfied by a iff **all** $P_{C_i}(a) = 0$.

To test "all $= \mathbf{0}$" with one random check, use:

> **Fact.** For $v \in \mathbb{Z}_2^n$, $v \neq 0$: $\mathbf{Pr}_r\left[\sum_i r_i v_i = 1\right] = \frac{1}{2}$.

So take a random combination $P_{r,\Phi} = \sum_i r_i P_{C_i}$. If a satisfies Φ, then $P_{r,\Phi}(a) = 0$ for **all** r; if a fails some clause, $\mathbf{Pr}_r[P_{r,\Phi}(a) = 1] = \frac{1}{2}$. (Choosing the random combination is essential — testing a fixed P_Φ would require checking all 2^n assignments.)

Now the punchline that yields $O(1)$ **queries**: $P_{r,\Phi}(a)$ is a polynomial of degree ≤ 3 in a, so it expands as $p(a_1, \ldots, a_n) = \alpha + \sum_{i \in I_1} a_i + \sum_{(i,j) \in I_2} a_i a_j + \sum_{(i,j,k) \in I_3} a_i a_j a_k$, and each of these three sums is the value of a **linear** function evaluated at a fixed "characteristic" point: $A_a(x) = \sum_i a_i x_i$, $\quad B_a(y) = \sum_{i,j} a_i a_j\, y_{ij}$, $\quad C_a(z) = \sum_{i,j,k} a_i a_j a_k\, z_{ijk}$, linear on $\mathbb{Z}_2^n$, $\mathbb{Z}_2^{n^2}$, $\mathbb{Z}_2^{n^3}$ respectively. Then $p(a) = \alpha + A_a(q_1) + B_a(q_2) + C_a(q_3)$, where q_1, q_2, q_3 are the characteristic vectors of the index sets I_1, I_2, I_3. **Evaluating the clause-polynomial = reading three table entries — a constant!**

> **The proof Π.** A purported satisfying assignment a is encoded as the concatenation of the three full truth-tables $\Pi \sim A' \,\|\, B' \,\|\, C'$, $\quad |A'| = 2^n$, $|B'| = 2^{n^2}$, $|C'| = 2^{n^3}$ — **exponentially long**, but only $O(1)$ entries are ever read. The verifier must guard against a cheating proof in two ways: the tables might not be linear, and even if each is close to *some* linear function, those functions might be **inconsistent** (e.g. B not encoding the pairwise products of A's coefficients).

14d. The verifier: LT + Consistency + Satisfiability

Three sub-tests, each $O(1)$ queries, run with self-correction (SCF) so every read is reliable:

> **Linearity (LT).** Test that A', B', C' are each δ-close to linear; if not, NO with probability $\geq \frac{1}{2}$ (so we may assume genuine linear $\widetilde{A}, \widetilde{B}, \widetilde{C}$ with coefficient vectors $\widetilde{a}, \widetilde{a} \circ \widetilde{a}, \widetilde{a} \circ \widetilde{a} \circ \widetilde{a}$).

> **Consistency (CT).** Check the tables describe the *same* $\widetilde{a}$ and that B, C really hold the products: pick random x, x', compute $a = \mathrm{SCF}(x, A')$, $a' = \mathrm{SCF}(x', A')$, $b = \mathrm{SCF}(x \circ x', B')$ (where $(x \circ x')_{(i-1)n+j} = x_i x'_j$); reject if $a \cdot a' \neq b$. Similarly tie A, B to C. This enforces $\widetilde{a}_i \widetilde{a}_j = \tilde{b}_{(i-1)n+j}$ and $\widetilde{a}_i \widetilde{a}_j \widetilde{a}_k = \tilde{c}_{(i-1)n^2+(j-1)n+k}$.

Satisfiability (CSAT). Pick random $r \in \{0,1\}^m$, compute the coefficients α, I_1, I_2, I_3 of $P_{r,\Phi}$ and their characteristic points; read $a = \mathrm{SCF}(q_1, A')$, $b = \mathrm{SCF}(q_2, B')$, $c = \mathrm{SCF}(q_3, C')$; if $\alpha + a + b + c = 1$ return **NO** (a violated clause was exposed), else **YES**.

Theorem. For $\delta < \frac{1}{24}$ there is a constant number k of calls to LT and CT such that, if no single $\widetilde{a}$ makes A', B', C' jointly δ-close to the linear functions with coefficients $\widetilde{a}, \widetilde{a} \circ \widetilde{a}, \widetilde{a} \circ \widetilde{a} \circ \widetilde{a}$, then with probability $\geq 1 - \delta$ at least one call returns NO. Hence

$$\text{3-SAT} \in \mathrm{PCP}[O(n^3), O(1)], \qquad \text{so} \qquad \mathrm{NP} \subseteq \mathrm{PCP}[O(n^3), O(1)].$$

Professor-pleaser — why this is the moral heart of PCP. A would-be proof is an *exponential table of numbers*. The verifier never trusts it: it (a) **tests** the table is essentially linear, (b) **self-corrects** every read so corruption can't hide, (c) checks the linear pieces are **mutually consistent** (a true assignment, not three unrelated functions), and (d) checks a **single random algebraic identity** that holds iff the assignment satisfies Φ. All four are $O(1)$ queries. The exponential $O(n^3)$ randomness is then squeezed to $O(\log n)$ by the polynomial-proof construction (step 2) and the composition lemma — but the *idea*, "arithmetize, then spot-check a robust algebraic encoding," is already fully present here.

The spine — one line per movement

Randomness inside the verifier. BPP is low: Sipser–Gács shows $\mathrm{BPP} \subseteq \Sigma_2^p \cap \Pi_2^p$ because a near-full set of accepting random strings can be **shifted a few times to cover the cube** while a tiny one cannot — a two-quantifier size test. **Interaction + coins** beats static proofs: a coin-flipping verifier decides **GNI** in two Arthur–Merlin messages by reducing non-isomorphism to *"is this set twice as big,"* checked by a **pairwise-independent hash** hitting a random bucket; the same logic ($\mathrm{coNP} \subseteq \mathrm{AM}$ would collapse PH) says **GI is probably not NP-complete**. With an all-powerful prover the reach is *all of PSPACE*: $\mathrm{IP} = \mathrm{PSPACE}$ via **arithmetization** (formula $\to$ polynomial) and the **SumCheck protocol**, whose soundness is the single fact *a low-degree polynomial has few roots, so a random evaluation exposes any lie* (with mod-p arithmetic and "simple QBF" taming the size and degree). Made **local**, the same algebra gives the **PCP theorem** $\mathrm{NP} = \mathrm{PCP}[O(\log n), O(1)]$: proofs checkable by $O(1)$ random bit-reads, equivalently **gap-amplifying reductions**, hence **Max-3-SAT / Independent Set are hard to approximate** unless P = NP. The bottom layer is built by hand from **linearity testing**, **self-correction**, and a $\mathbb{Z}_2$ arithmetization giving $\mathrm{NP} \subseteq \mathrm{PCP}[O(n^3), O(1)]$.

Theme	The one idea
Polynomial hierarchy	stacked quantifier alternations; $\Sigma_i = \Pi_i \Rightarrow$ PH collapses
Sipser–Gács	$\mathbf{BPP} \subseteq \Sigma_2^p \cap \Pi_2^p$: shift a big set to tile the cube, a tiny one can't
Adleman	$\mathbf{BPP} \subseteq \mathbf{P/poly}$ — randomness free with advice (but too coarse)
Interactive proof IP	unbounded prover + poly-time coin-flipping verifier; verifier has last word
Public vs private coins	Goldwasser–Sipser: $\mathbf{IP}[k] \subseteq \mathbf{AM}[k+2]$ — hiding coins barely helps
GNI ∈ AM[2]	non-isomorphism **doubles** a set; detect size by a random hash bucket
AM/MA structure	perfect completeness; $\mathbf{AM}[k] = \mathbf{AM}$; $\mathbf{MA} \subseteq \Sigma_2 \cap \Pi_2$, $\mathbf{AM} \subseteq \Pi_2$
GI not NP-complete	$\mathbf{coNP} \subseteq \mathbf{AM} \Rightarrow \mathbf{PH} = \Sigma_2$ (Boppana–Håstad–Zachos)
IP = PSPACE	arithmetize; SumCheck pins an exponential sum via one poly per variable
SumCheck soundness	distinct degree-d polys agree on $\leq d$ points ⇒ random eval catches lies
TQBF ∈ IP	$\exists \to \sum, \forall \to \prod$; mod-$p$ + "simple QBF" tame value & degree
PCP[r,q]	local verifier: r coins, q bit-reads; perfect completeness, soundness $< \frac{1}{2}$
PCP theorem	$\mathbf{NP} = \mathbf{PCP}[O(\log n), O(1)]$: two faces — local proofs & inapproximability
Gap reduction	$\varphi \in$SAT ⇒ all clauses; $\varphi \notin$SAT ⇒$<$ c-fraction; $\equiv \mathbf{PCP}[\log, 1]$
Linearity test / SCF	mostly-linear table is testable and **self-correctable** with $O(1)$ reads
NP ⊆ PCP[n³,O(1)]	encode assignment as 3 linear tables A, B, C; LT + consistency + one random identity

Connections / threads to drill

- **Arithmetization is the through-line.** Boolean → polynomial appears **twice**: globally for SumCheck/IP=PSPACE (one polynomial per variable, random evaluation mod p) and locally for PCP (three linear tables, random algebraic identity). The shared lemma is "*a nonzero low-degree polynomial has few roots.*" Be able to state it and say where each use deploys it.

- **PCP → Independent Set inapproximability** is a *recurring exam question* (exam-1 Q4, exam-3 Q4). Drill the chain: PCP theorem ⇒ gap version of Max-3-SAT ⇒ (via the standard FGLSS-style reduction) a graph where $x \in L$ gives a large independent set and $x \notin L$ only a ρ-fraction one ⇒ a c-approximation for IS would cross the gap and decide an NP-complete problem.
- **The set–lower-bound protocol** (GNI ∈ AM) is the cleanest place to *use pairwise-independent hashing* — ties back to Lecture 5's universal hashing and the general "estimate a set size by hashing into buckets" move.
- **"PH collapses" as a hardness disclaimer.** $\mathrm{coNP} \subseteq \mathrm{AM}$, $\mathrm{P} = \mathrm{NP}$, $\Sigma_i = \Pi_i$ — each forces a collapse and is therefore the formal way to say "this would be a shock." GI-not-NP-complete is the showcase.
- **Sipser–Gács reuses the probabilistic method** ("failure probability < 1 ⇒ good shifts exist") and the **covering-by-shifts** trick recurs in the perfect-completeness proofs for MA/AM — same engine, three appearances.
- [ ] Be ready to **run SumCheck end-to-end** on a 2–3 variable example and quote the soundness $1 - nd/p$.
- [ ] Be ready to **state both faces of the PCP theorem** and prove the $\mathrm{PCP}[\log n, 1] \Leftrightarrow$ gap-reduction equivalence (at least the easy ⇐ direction).
- [ ] Be ready to explain **self-correction** (why reading two entries recovers a corrupted table value) and why the proof needs **three** tables A, B, C plus a consistency test.