Lecture 3 — Models of Probabilistic Computation & Complexity Classes

> Course **2-INF-135/15 Pravdepodobnostné algoritmy**, LS 2025/26. Source slides: `03_slidy.pdf` (28 pages).

What this lecture is about

Lectures 1–2 *built and analysed* randomized algorithms. This lecture steps back and asks the structural questions:

1. **Where can the randomness sit, and what kind of error does it cause?** (Two ways to classify *every* randomized algorithm.)
2. **How do we drive the error down?** (Repetition + majority — and the one number, the *gap* ε, that decides whether this is cheap or ruinously expensive.)
3. **Which complexity classes do these algorithms define?** (RP, coRP, ZPP, PP, BPP — their definitions, the lemmas that make them robust, and how they nest.)
4. **What if our coins are bad?** (A *weak* random source — biased and correlated — can wreck an algorithm; yet we will prove that for `BPP` it makes **no difference at all**.)

The single thread running through all of it: **a randomized algorithm is a deterministic algorithm plus a string of coin flips r.** Everything below is about *how good those coins have to be* and *what we promise about the answer*.

---

1. Two ways to place the randomness

There are two equally valid pictures of *where* the probability lives.

Model I — a distribution over deterministic strategies. We have a fixed pool of deterministic algorithms $\{A_1, \ldots, A_m\}$. On input w we pick an index $i \in_R \{1, \ldots, m\}$ **once**, uniformly at random, and then run that one deterministic computation $C_i = A_i(w)$.

> Picture: *roll a die once, then follow the rulebook it selected.* The only randomness is the initial choice of strategy. Examples: the database equality test $X \overset{?}{=} Y$ (the random choice is *which*

> *position to probe*), Freivalds' $AB \overset{?}{=} C$ (random vector).

Model II — a nondeterministic algorithm with a distribution over its choices. The algorithm branches as it runs, and *each* branch point is resolved by a coin flip. Randomness is used **repeatedly**, throughout the computation.

> Picture: *flip a fresh coin every time the algorithm reaches a fork.* Example: randomized QuickSort re-randomizes the pivot at every level of recursion.

The two models have the same power (Model II's coin-flip choices, read off in order, *are* a long random string — exactly the r from the spine sentence). But they sharpen different intuitions: Model I = "randomly pick a method", Model II = "randomness woven through the run".

---

2. Classification by error: Las Vegas vs Monte Carlo

Let $A(\cdot)$ be the algorithm and $F(\cdot)$ the function we are trying to compute. There are three error regimes.

Type	Promise	Errs on…	Example
Las Vegas	$\Pr[A(x) = F(x)] = 1$	never (only *time* is random)	boss-election, R-Select / Lazy-Select
Monte Carlo, one-sided	$x \in L:\ \Pr[A(x) = 1] \geq \frac{1}{2}$; $x \notin L:\ A(x) = 0$	only on yes-instances	database **equality** test
Monte Carlo, two-sided	$\Pr[A(x) = F(x)] \geq \frac{1}{2} + \varepsilon$	either direction (but biased right)	database **inequality** test

The three differ in *what they promise*:

- **Las Vegas never lies.** It is always correct; randomness only affects how *long* it runs (we engineer a good *expected* time). "I'll always give you the right answer — I just can't promise exactly when."
- **One-sided Monte Carlo lies in one direction only.** A "yes" is the truth; a "no" might be a miss. (For the equality test: if it ever finds a differing bit it is *certain* the strings differ; if it sees a match it can only *guess* they are equal.)
- **Two-sided Monte Carlo can be wrong either way**, but it is *biased toward the truth* — correct with probability strictly above $\frac{1}{2}$.

> **Why the $+\varepsilon$ matters so much.** "$\geq \frac{1}{2} + \varepsilon$" is not cosmetic. The whole machinery of §4 and the gulf between `BPP` and `PP` in §6 turn on whether that ε is a *constant* or is allowed to shrink to 0 as the input grows. Keep your eye on the gap.

---

3. A worked two-sided example: testing $X \neq Y$

The cleanest illustration. Computer R_I holds $X = x_1 \dots x_n$, R_{II} holds $Y = y_1 \dots y_n$, and we want to decide $L = \{(X, Y) \mid X \neq Y\}$ using *few communicated bits*.

Protocol. R_I picks a position $j \in_R \{1, \dots, n\}$ and sends the pair (j, x_j) to R_{II}. Then R_{II}:

$$\begin{cases} \textbf{accept } (\text{declare } X \neq Y) & \text{if } x_j \neq y_j, \\ \textbf{flip a biased coin: } \begin{cases} \textbf{accept} & \text{w.p. } \frac{1}{2} - \frac{1}{2n} \\ \textbf{reject} & \text{w.p. } \frac{1}{2} + \frac{1}{2n} \end{cases} & \text{if } x_j = y_j. \end{cases}$$

Communication cost: $2 \log n$ bits (send an index and one bit) versus n for any deterministic protocol. This is the payoff.

Correctness — check both kinds of input. The worst case for detecting a difference is when the strings differ in *exactly one* position, $\exists! j : x_j \neq y_j$ (fewest chances to catch it), so we analyse that.

- $X = Y$ (so $(X, Y) \notin L$, correct answer = reject). Every probe finds $x_j = y_j$, so we always fall into the coin branch and reject with $\Pr[\text{correct}] = \frac{1}{2} + \frac{1}{2n} > \frac{1}{2}$. ✓
- $X \neq Y$ (so $(X, Y) \in L$, correct answer = accept). With probability $\frac{1}{n}$ we hit the one differing position and accept outright; with probability $\frac{n-1}{n}$ we hit a matching position and accept via the coin (prob. $\frac{1}{2} - \frac{1}{2n} = \frac{n-1}{2n}$):
  $\Pr[\text{correct}] = \frac{1}{n} + \frac{n-1}{n} \cdot \frac{n-1}{2n} = \frac{2n + (n-1)^2}{2n^2} = \frac{n^2+1}{2n^2} = \frac{1}{2} + \frac{1}{2n^2} > \frac{1}{2}$. ✓

> **The point.** A *single* probe and a *tiny* engineered bias ($\frac{1}{2n}$) are enough to beat $\frac{1}{2}$ in **both** directions. That is precisely a two-sided-error Monte Carlo algorithm — correct with probability $> \frac{1}{2}$ on every input. The gap here ($\varepsilon \sim \frac{1}{n^2}$) is *not* constant, which (foreshadowing §6) is exactly the kind of vanishing gap that lands a problem in `PP` rather than `BPP`.

---

4. Driving the error down by repetition

4a. Las Vegas has two faces, and they are interchangeable

There are two ways to *define* Las Vegas, and proving them equivalent is a tiny gem that reuses **Markov's inequality** from Lecture 2.

1. **"Never lies, may shrug":** $A^?$ outputs $A^?(x) = F(x)$ **or** $A^?(x) =$ **?** ("don't know"), with $\Pr[\,?\,] \le \frac{1}{2}$.
2. **"Always correct, random time":** $\Pr[A(x) = F(x)] = 1$.

$1 \Rightarrow 2$: repeat $A^?$ until it returns a real answer. Each trial succeeds with probability $\ge \frac{1}{2}$, so $E[\#\text{repetitions}] = \frac{1}{\Pr[\text{answer}]} \le \frac{1}{1/2} = 2$**.** We get a *guaranteed-correct* algorithm at the price of 2 runs in expectation.

$2 \Rightarrow 1$: run A for at most $2\,E[T]$ steps. By Markov, $\Pr[T > 2E[T]] \le \frac{1}{2}$, so with probability $\ge \frac{1}{2}$ it has finished (and is correct); otherwise output "**?**".

> **Deep point.** "Always right but slow sometimes" and "fast but occasionally admits ignorance" are the *same class*, convertible for a factor of 2. Markov is the bridge from the random-time form to the shrug form — the very same "restart on FAIL, $E[\text{runs}] < 2$" move that made R-Select linear in Lecture 2.

4b. Bounded two-sided error: a constant number of repetitions

Suppose $\Pr[A(x) = F(x)] \ge \frac{1}{2} + \varepsilon$. Run A a total of t times and **decide by majority**. Let X = number of correct runs; $X \sim \text{Bin}(t, p)$ with $p = \frac{1}{2} + \varepsilon_x$, $\varepsilon_x \ge \varepsilon$. Majority is wrong iff fewer than half the runs are correct: $1 - \Pr[\text{majority correct}] = \sum_{i=0}^{t/2-1} \Pr[X = i]$**.**

The elementary bound (no Chernoff needed). Write $p(1-p) = (\frac{1}{2} + \varepsilon_x)(\frac{1}{2} - \varepsilon_x) = \frac{1}{4} - \varepsilon_x^2$. For a *losing* term ($i < t/2$, so the exponent on the smaller factor $(\frac{1}{2} - \varepsilon_x)$ dominates), $\Pr[X = i] = \binom{t}{i}\left(\frac{1}{2} + \varepsilon_x\right)^i\left(\frac{1}{2} - \varepsilon_x\right)^{t-i} \le \binom{t}{i}\left(\frac{1}{4} - \varepsilon_x^2\right)^{t/2}$**.** Summing over all i and using $\sum_{i=0}^{t}\binom{t}{i} = 2^t$:

$$\sum_{i=0}^{t/2-1}\Pr[X=i] < \left(\tfrac{1}{4} - \varepsilon_x^2\right)^{t/2} \cdot 2^t = \underbrace{\left(\tfrac{1}{4}\right)^{t/2}2^t}_{=1}(1 - 4\varepsilon_x^2)^{t/2} = (1-4\varepsilon_x^2)^{t/2} \le (1-4\varepsilon^2)^{t/2}.$$

Set the target error δ: solving $(1-4\varepsilon^2)^{t/2} = \delta$, $\boxed{t = \dfrac{2\ln\delta}{\ln(1-4\varepsilon^2)}}$.

> **The headline.** If ε and δ are **constants**, then t is a **constant**. A bounded-error algorithm can be amplified to any fixed confidence with $O(1)$ repetitions — this is exactly what makes `BPP` a well-behaved class. (Push $\delta = 2^{-p(n)}$ and you still only pay $t = O(p(n))$ repetitions, because the denominator is a constant.)

4c. Unbounded error: the gap can cost you everything

Now suppose $\Pr[A(x) = F(x)] > \frac{1}{2}$ but the gap is allowed to **shrink with the input**, e.g. $\varepsilon = 2^{-|x|}$. The *same formula* gives $t = \frac{2\ln\delta}{\ln(1-4\cdot 2^{-2|x|})} \overset{\ln(1-4y)\le -4y}{\geq} \frac{2\ln\delta}{-4\cdot 2^{-2|x|}} = -\frac{1}{2}(\ln\delta)\, 2^{\,2|x|}$. The number of repetitions needed is **exponential in** $|x|$.

> **The single most important contrast in this lecture.** The amplification formula is the *same*; only the gap ε changes.
>
> - **Constant gap** (bounded away from $\frac{1}{2}$) $\Rightarrow$ constant amplification $\Rightarrow$ the friendly class `BPP`.
> - **Vanishing gap** (may approach $\frac{1}{2}$) $\Rightarrow$ exponential amplification $\Rightarrow$ the monstrous class `PP` (which, we'll see, contains all of `NP`).
>
> Everything separating `BPP` from `PP` is *whether the gap is bounded*.

---

5. The complexity classes

All classes below use **polynomial-time** probabilistic Turing machines (PTMs). They differ only in the *promise* on yes- and no-instances.

RP — one-sided error, "yes" is trustworthy

$$L \in \mathsf{RP}: \quad \begin{cases} x \in L & \Rightarrow \Pr[A(x)=1] \geq \frac{1}{2} \quad (\text{may miss}), \\ x \notin L & \Rightarrow A(x) = 0 \quad (\text{never a false "yes"}). \end{cases}$$

- `RP` $\subseteq$ `NP`**.** A random string r on which A accepts is exactly an `NP` *witness* — this is the "guess = certificate" equivalence from the NP session, read through randomness: where `NP` *guesses* a good r, `RP` *samples* one and succeeds with decent probability.
- **Robustness lemma.** $L \in \mathsf{RP}$ iff the threshold $\frac{1}{2}$ can be replaced by $1/q(|x|)$ for *any* polynomial q. The reason is one-sided amplification: a no-instance can never produce a false yes, so t independent runs with an **OR** vote err only if *all* miss:

$\Pr[\text{error after } t = q(|x|) \text{ runs}] \le \left(1 - \frac{1}{q(|x|)}\right)^{q(|x|)} \approx e^{-1} < \frac{1}{2}$, and more runs push it below any δ. So *any* polynomially-small success probability is as good as $\frac{1}{2}$ for `RP`.

coRP is the mirror image: errs only on no-instances ("no" is trustworthy, "yes" might be a false alarm). `coRP` $\subseteq$ `coNP`.

ZPP — zero error (the Las Vegas class)

$$\mathsf{ZPP} := \mathsf{RP} \cap \mathsf{coRP}, \qquad \mathsf{ZPP} \subseteq \mathsf{NP} \cap \mathsf{coNP}.$$

- **Lemma (the Las Vegas characterisation).** $L \in \mathsf{ZPP}$ iff there is a PTM $M^{?}$ with $x \in L \Rightarrow$ **accept** or **?**, $\quad x \notin L \Rightarrow$ **reject** or **?**, $\quad \Pr[\,?\,] \le \frac{1}{2}$. That is *exactly* the "never lies, may shrug" Las Vegas machine of §4a.
- $\Rightarrow$ (build the shrug machine from the two one-sided machines): run the `RP` machine M and the `coRP` machine $\overline{M}$ —

```
if M(x) = 1     then accept      // RP says yes ⇒ truly yes
else if coM(x)=1 then reject     // coRP says no ⇒ truly no
else return ?                    // neither was sure
```

- $\Leftarrow$ (recover the two one-sided machines from $M^{?}$): turn "**?**" into **reject** to get an `RP` machine, and into **accept** to get a `coRP` machine.

> **Deep point.** `RP` errs one way, `coRP` the other; *intersecting* them cancels both error directions and leaves **no error at all** — only an occasional "don't know", i.e. extra expected time. **ZPP = zero-error = Las Vegas.** The intersection of two opposite one-sided errors is honesty.

PP — unbounded two-sided error

$$L \in \mathsf{PP}: \quad x \in L \iff \Pr[A(x) = 1] > \tfrac{1}{2}.$$

The threshold is just a strict $\frac{1}{2}$ — **the gap may be exponentially small** (this is the §4c regime). Consequences:

- **`NP` $\subseteq$ `PP`.** Take an `NP` machine and graft on an *equal-sized always-accepting* subtree of computations. If there was even one accepting path, the accepting fraction now tips just over $\frac{1}{2}$; if there were none, it stays at exactly $\frac{1}{2}$. The tip can be a single path out of 2^{poly} — a vanishing gap, which is exactly what `PP` tolerates.
- **`PP` $\subseteq$ `PSPACE`.** Simulate the machine and *count* accepting computations in polynomial space (reuse the space across the exponentially many branches — the space-reuse principle).

> `PP` is essentially a *counting* class (it can detect a strict majority of an exponential tree). Its power comes entirely from the unbounded gap; the moment you demand a constant gap you drop to `BPP`.

BPP — bounded two-sided error (the practical class)

$$L \in \mathsf{BPP}: \quad x \in L \Rightarrow \Pr[A(x)=1] \geq \tfrac{3}{4}, \quad x \notin L \Rightarrow \Pr[A(x)=0] \geq \tfrac{3}{4}.$$

(Any constant $> \frac{1}{2}$ works — the $\frac{3}{4}$ is just a convenient choice; what matters is a *constant gap* $\varepsilon > 0$.)

- **Lemma.** $L \in \mathsf{BPP}$ iff there is a PTM with $\Pr[\text{error}] \leq 2^{-p(|x|)}$ for some polynomial p — i.e. the error is amplifiable to *exponentially small*. By §4b with $\varepsilon = \frac{1}{4}$, $\delta = 2^{-p(|x|)}$, this costs only $t = \frac{2\ln\delta}{\ln(1-4\varepsilon^2)} = \frac{2p(|x|)}{\log_2(4/3)} = O(p(|x|))$ repetitions. A *polynomial* number of runs buys *exponentially* tiny error.

This is the class of randomized algorithms we *actually trust*: constant gap, amplifiable to astronomically small error for free.

BPP ⊆ P/poly (Adleman's theorem)

A `BPP` language can be decided by polynomial-size **circuits** — one fixed "advice string" per input length works for *all* inputs of that length.

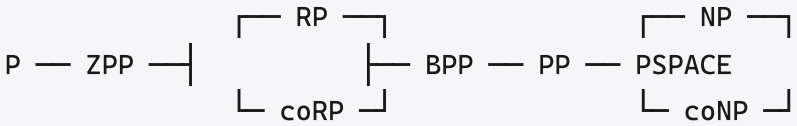
Proof (probabilistic method on the coins, with a union bound over inputs). First amplify so a single run errs with probability $\leq$ small, then take m runs and majority-vote; by **Chernoff**, with $m/4$ runs bad in expectation, $\Pr[\#\text{bad} \geq m/2] \leq e^{-\delta^2\mu/3} = e^{-m/12}$**.** Fix an input x and let S_x = set of random-string-vectors $A(n) = (r_1, \ldots, r_m)$ for which the majority is *wrong*. Then $E[|S_x|] \leq e^{-m/12}\, 2^{|A(n)|}$, and summing over all 2^n inputs of length n: $E\left[\sum_x |S_x|\right] \leq 2^n \cdot e^{-m/12} \cdot 2^{|A(n)|}$**.** If this is $< 2^{|A(n)|}$ then some vector $A(n)$ lies in *no* S_x — i.e. it is **correct on every x of length n**. The condition is $2^n e^{-m/12} < 1$, satisfied by $m = 12(n+1)$. Hard-code that one vector as advice ⇒ a poly-size circuit per length.

> **Same engine as the non-uniform derandomization tool (Adleman, `RP` ⊆ `P/poly`).** Amplify until the bad-fraction is below 2^{-n}, union-bound over the 2^n inputs, conclude *one* random string is good for *all* of them, and freeze it. The probabilistic method applied to the coins themselves. Caveat (same as there): this *proves existence* of the advice; it does not hand you an efficient way to find it.

The class zoo, nested

$$\mathsf{P} \subseteq \mathsf{ZPP} \subseteq \mathsf{RP} \subseteq \mathsf{BPP} \subseteq \mathsf{PP} \subseteq \mathsf{PSPACE},$$

with the one-sided wings and the `NP` connections:
$\mathsf{RP} \subseteq \mathsf{NP}, \quad \mathsf{coRP} \subseteq \mathsf{coNP}, \quad \mathsf{ZPP} = \mathsf{RP} \cap \mathsf{coRP} \subseteq \mathsf{NP} \cap \mathsf{coNP}, \quad \mathsf{BPP} \subseteq \mathsf{P/poly}.$



A reading of the picture: error-free in the middle (`P` , `ZPP`), one-sided just outside (`RP` / `coRP` , each tucked under `NP` / `coNP`), bounded two-sided wider still (`BPP`), unbounded two-sided enormous (`PP` $\supseteq$ `NP`), all swallowed by `PSPACE` . The famous open question lives here: **is `BPP` = `P` ?** (Lecture 5's PRGs say *yes, under plausible hardness assumptions* — randomness is a convenience, not a necessity.) Note `BPP` vs `NP` is *not* known either way.

---

6. Where do the coins come from? Sources of randomness

Every class above silently assumed *perfect* coins. Time to question that.

The perfect random source

A random variable producing an infinite stream $x_1 x_2 \cdots \in \{0,1\}^*$ such that every length-n prefix is uniform: $\forall (y_1, \ldots, y_n): \quad \Pr[x_1 = y_1, \ldots, x_n = y_n] = 2^{-n}$. Two requirements rolled into one:

- **Independence** — flip i does not depend on the previous flips;
- **Unbiasedness (correctness)** — $\Pr[x_i = 1] = \frac{1}{2}$ *exactly*.

This is an idealisation. Real physical sources (thermal noise, radioactive decay timings, mouse jitter) are biased and correlated. So:

The δ-random source (Santha–Vazirani)

Drop *both* guarantees. Fix $0 < \delta \leq \frac{1}{2}$. The probability that bit i is 1 may depend **arbitrarily** on all previous bits, written $p(y_1, \ldots, y_{i-1})$, subject to only one constraint: $p(y_1, \ldots, y_{i-1}) \in [\delta,\ 1-\delta]$.
$\Pr[x_1 = y_1, \ldots, x_n = y_n] = \prod_{i=1}^{n} \Big(y_i\, p(\cdot) + (1 - y_i)\big(1 - p(\cdot)\big) \Big).$

> **Read it as an adversary.** A demon sets each bit's probability, looking at everything you've flipped so far, free to bias it as hard as it likes — *except* it can never push past δ or $1-\delta$. So no bit is ever *fully* predictable ($\delta > 0$), but bits can be heavily skewed and tangled. $\delta = \frac{1}{2}$ forces

> every bit to a fair, independent coin (perfect source); $\delta = 0$ lets the demon fix bits outright (no randomness left).

Cautionary tale: 2-SAT random walk

Why we should worry. The classic 2-SAT algorithm (Papadimitriou): start with a random assignment $\alpha \in_R \{0,1\}^n$; while some clause is unsatisfied, pick one and **flip a random one of its two variables**.

With a fair coin, $E[\#\text{flips}] \le n^2$. Track the Hamming distance i to a fixed satisfying assignment b^*; let $t(i)$ = expected flips from distance i. An unsatisfied clause has ≥ 1 of its two variables set wrong, so flipping a random one of the two moves *toward* b^* with probability $\ge \frac{1}{2}$:
$t(0) = 0, \qquad t(i) \le \frac{1}{2}\big(t(i-1) + t(i+1)\big) + 1, \qquad t(n) \le t(n-1) + 1.$ Comparing with the equality version $x(i) = \frac{1}{2}(x(i-1) + x(i+1)) + 1$ gives $t(i) \le x(i) = 2in - i^2 \le n^2$, hence $E[\#\text{flips}] \le n^2$. (A symmetric random walk on a line of length n has quadratic hitting time — gambler's ruin.)

With a δ-random source, the adversary can bias every flip *away* from b^* (still within $[\delta, 1-\delta]$), turning the symmetric walk into one that drifts the wrong way ⇒ **exponentially many flips**.

> **Lesson.** An analysis that leaned on true $\frac{1}{2}$-coins (here: the symmetric walk) can be destroyed by a weak source. So: *which classes survive a δ-source?* That is the question the rest of the lecture answers — and the answer for `BPP` is wonderfully clean.

δ-RP, δ-BPP and the two easy boundary cases

Define `δ−RP` and `δ−BPP` exactly like `RP` / `BPP`, but the machine is fed by a δ-random source. Label each node's edges $F(\text{0-son}) + F(\text{1-son}) = 1$ with each value in $[\delta, 1-\delta]$;
$\Pr[\text{leaf}] = \prod_{\alpha \in \text{path}} F(\alpha)$.

- $\text{0-RP} = \text{0-BPP} = \text{P}$**.** At $\delta = 0$ the adversary can *force* any bit, so to be safe **every leaf must answer correctly** — that is just a deterministic algorithm. Worthless randomness collapses to `P`.
- $\frac{1}{2}\text{-RP} = \text{RP}$, $\frac{1}{2}\text{-BPP} = \text{BPP}$**.** At $\delta = \frac{1}{2}$ every edge is forced to $\frac{1}{2}$ = a perfect source. Trivially unchanged.

The whole game is the strict interior $0 < \delta < \frac{1}{2}$.

---

7. The theorem: a weak source is as good as a perfect one for BPP

> **Theorem.** For every $0 < \delta < \frac{1}{2}$, $\delta\text{-BPP} = \text{BPP}$.

- $\delta\text{-BPP} \subseteq \text{BPP}$ is trivial: a perfect source can simulate a δ-source.
- $\text{BPP} \subseteq \delta\text{-BPP}$ is the substance — **simulate near-perfect randomness using only a weak, adversarial δ-source.** This is an early *randomness extractor*.

The construction

Let N be a `BPP` machine for L with error **reduced to** $\leq \frac{1}{32}$, input x, running time $p(|x|)$, needing $n = p(|x|)$ random bits. Choose a constant block size $k = \frac{3\log n + 5}{2\delta - 2\delta^2}$. Pull n blocks $\beta_1, \ldots, \beta_n$, each k bits, from the δ-source. For each "seed" $Z \in \{0, 1, \ldots, 2^k - 1\}$, **extract** one bit per block by the inner product mod 2: $\beta_i \cdot Z = \sum_{\ell=1}^{k} (\beta_i)_\ell \, Z_\ell \pmod 2$. Run 2^k parallel simulations of N: simulation Z uses the extracted random string $(\beta_1 \cdot Z,\ \beta_2 \cdot Z,\ \ldots,\ \beta_n \cdot Z)$. **Decide by majority** over the 2^k simulations.

Let $T = \{(\beta_1 \cdot Z, \ldots, \beta_n \cdot Z) : Z = 0, \ldots, 2^k - 1\}$ be the $|T| = 2^k$ extracted strings, and B the set of "bad" strings on which N errs, $|B| \leq 2^n/32$. The majority is wrong exactly when at least half the extracted strings are bad: $\Pr[\text{majority wrong}] = \Pr\left[\,|T \cap B| \geq |T|/2\,\right]$. Goal: show this is $\leq \frac{1}{4}$, so the simulation is correct with probability $\geq \frac{3}{4}$ — landing L in `δ-BPP`.

Why inner products extract randomness — three lemmas

Define the **bias** of an extracted bit as $\text{bias}(\beta_i \cdot Z) = \big(\Pr[\beta_i \cdot Z = 1] - \Pr[\beta_i \cdot Z = 0]\big)^2$ and the **collision probability** of a source block $\Pr[\beta]^2$.

- **Lemma 1 (a Parseval identity).** Summing bias over all seeds equals 2^k times the block collision probability: $\sum_{Z=0}^{2^k-1} \text{bias}(\beta_i \cdot Z) = 2^k \sum_{\beta=0}^{2^k-1} \Pr[\beta]^2$. *Proof idea.* Write $\Pr[\beta \cdot Z = 0] - \Pr[\beta \cdot Z = 1] = \sum_\beta (-1)^{\beta \cdot Z} \Pr[\beta]$ (the ± 1 *character*). Square, sum over Z, and use orthogonality: $\sum_Z (-1)^{(\beta_1 + \beta_2)\cdot Z} = 0$ unless $\beta_1 = \beta_2$ (then it is 2^k). The cross-terms vanish, leaving $2^k \sum_\beta \Pr[\beta]^2$.

- **Lemma 2 (collisions decay in k).** For a δ-source block, $\sum_{\beta=0}^{2^k-1} \Pr[\beta]^2 \leq \big(\delta^2 + (1-\delta)^2\big)^k$. *Proof idea.* Pair up sequences differing in one bit; through the lens of that bit the contribution is $Ap_i^2 + A(1-p_i)^2$, maximised at the extreme $p_i \in \{\delta, 1-\delta\}$. The block factorises, giving $\sum_i \binom{k}{i} \delta^{2i}(1-\delta)^{2(k-i)} = (\delta^2 + (1-\delta)^2)^k$. Since $\delta < \frac{1}{2}$, we have $\delta^2 + (1-\delta)^2 = 1 - (2\delta - 2\delta^2) < 1$, so collisions — and hence bias — **decay exponentially in k.**

Combining, $\sum_Z \mathbf{bias}(\beta_i \cdot Z) \le 2^k(\delta^2 + (1-\delta)^2)^k$. Call an extracted bit **skewed** if $\mathbf{bias} \ge 1/n^2$; an *unskewed* bit then satisfies $\Pr[\mathbf{bit}=1] \in (\frac{1}{2} - \frac{1}{2n},\ \frac{1}{2} + \frac{1}{2n})$ — nearly fair. The bias budget caps the number of skewed bits across the whole computation at $n^3\, 2^k(\delta^2 + (1-\delta)^2)^k$, and the choice $k \ge (5 + 3\log n)/(2\delta - 2\delta^2)$ forces this $\le 2^k/32$.

- **Lemma 3 (few bad strings in expectation).** $E\big[\,|T \cap B|\,\big] \le \frac{|T|}{8}$. *Proof idea.* Split T into the $\le 2^k/32$ *skewed* strings plus *unskewed* ones. For an unskewed string each bit is within $\frac{1}{2n}$ of fair, so its chance of landing in B is at most $\frac{|B|}{2^n}(1+\frac{1}{n})^n \le \frac{1}{32} \cdot e$. Adding the $\le 2^k/32$ skewed strings: $E[|T \cap B|] \le \frac{2^k}{32} + \frac{2^k}{32}(1+e) \le \frac{2^k}{8} = \frac{|T|}{8}$.

Finish with Markov. $\Pr\left[\,|T \cap B| > |T|/2\,\right] \le \frac{E[|T \cap B|]}{|T|/2} \le \frac{|T|/8}{|T|/2} = \frac{1}{4}$. So majority over the 2^k extracted simulations is correct with probability $\ge \frac{3}{4}$, i.e. $L \in \delta\text{-BPP}$. ■

> **The punchline.** A random source can be **biased and adversarially correlated** and it still does not matter for `BPP` — provided it is not *fully* predictable ($\delta > 0$). The recipe: chop the weak stream into blocks, **distil near-fair, near-independent bits by inner products**, run exponentially many (2^k) simulations on the distilled strings, and **vote**. Bias is bounded by a *collision* (Lemma 1–2), collision decays in the block size (Lemma 2), so few extracted bits are skewed, so few simulations are bad (Lemma 3), so the majority is right (Markov).
>
> **Randomness *quality* is free; only randomness *quantity / predictability* (the $\delta > 0$) is the real resource.** The two boundaries pin it exactly: $\delta = 0$ (fully predictable) collapses to `P`, $\delta = \frac{1}{2}$ is already perfect, and *everything strictly in between is secretly just as powerful as perfect randomness*. This is the historical seed of the entire theory of randomness extractors.

Closing themes

Idea	One-line takeaway
Two placements of randomness	distribution over deterministic strategies (Model I) vs. coin at every fork (Model II) — same power, different intuition.
Error taxonomy	Las Vegas (never lies, random time) · one-sided MC (one trustworthy answer) · two-sided MC (biased toward truth).
Las Vegas duality	"always correct, random time" ⇔ "fast, may shrug" — bridged by Markov, cost factor $\mathbf{2}$.
The gap ε is everything	constant gap ⇒ $O(1)$ amplification ⇒ `BPP`; vanishing gap ⇒ exponential amplification ⇒ `PP`.
Class zoo	$\mathsf{P} \subseteq \mathsf{ZPP} \subseteq \mathsf{RP} \subseteq \mathsf{BPP} \subseteq \mathsf{PP} \subseteq \mathsf{PSPACE}$; $\mathsf{RP} \subseteq \mathsf{NP}$, $\mathsf{ZPP} = \mathsf{RP} \cap \mathsf{coRP}$.
ZPP = zero error	intersecting two *opposite* one-sided errors cancels both ⇒ honesty (= Las Vegas).
BPP ⊆ P/poly	probabilistic method on the coins + union bound over 2^n inputs ⇒ one advice string per length (Adleman).
Weak sources	a biased, correlated δ-source ($\delta > 0$) is *as good as perfect* for `BPP`; extract by inner products, simulate, vote. Quality is free; only predictability ($\delta = 0$) kills it.