Lecture 8 — Derandomization (Derandomizácia)

> Course **2-INF-135/15 Pravdepodobnostné algoritmy**, LS 2025/26. Source slides: `07_RA_slidy_derandAlg.pdf` (22 pages) + `RA_slidy_derand.pdf` (23 pages). Štátnicové syllabus topics covered here: *derandomization — the method of conditional probabilities (with pessimistic estimators), limited (k-wise) independence and small sample spaces, pseudorandom generators, and the hardness-vs-randomness paradigm (Nisan–Wigderson). Worked algorithms: Schöning k-SAT via covering codes, parallel MIS, MaxCut, Set Balancing.*

The one-paragraph map of the whole lecture

A randomized algorithm is allowed to flip coins. **Derandomization** asks: can we get the same guarantee *without* the coins — deterministically — and not pay too much in running time? The whole lecture is a menu of five answers, ordered from "brute and obvious" to "deep and conditional":

1. **Enumeration** — just try *all* coin sequences. Always works, but costs $2^{(\#\text{random bits})}$. Cheap only if the algorithm uses $O(\log n)$ bits.
2. **Non-uniform / non-constructive advice** — *one* fixed bit-string works for all inputs of a given length. It exists by counting, but we may not be able to *find* it. Lands us in $\mathsf{P/poly}$, not P.
3. **Method of conditional probabilities** — fix the random choices *one at a time*, always steering toward a better-than-average outcome. The workhorse for "the average object is good, give me a concrete good one."
4. **Limited (k-wise) independence** — if the analysis only ever uses interactions among k variables at a time, replace true randomness by a tiny sample space that *looks* k-wise independent. Few random bits $\Rightarrow$ small space $\Rightarrow$ enumerate it.
5. **Pseudorandom generators (PRG)** — stretch a short truly-random seed into a long string that *no small circuit can tell* from random. Then enumerate seeds. This is the heavy machinery, and it rests on a beautiful trade: **a hard function buys you randomness** (Nisan–Wigderson).

The recurring question is always the same: *what exactly did the probabilistic analysis actually need?* The less it needed (only the expectation? only pairwise interactions? only fooling small circuits?), the cheaper the derandomization.

---

0. Two flavors, and the two trivial bookends

The slides split derandomization into:

- **Algorithmic** (problem-specific): take *this* concrete Monte-Carlo / RP algorithm and surgically remove its coins.
- **Complexity-theoretic** (general strategies): theorems of the form "if such-and-such exists, then a whole *class* of randomized algorithms can be derandomized."

Before the clever methods, two bookends that need almost no thought.

Bookend 1 — Enumeration ("hrubá sila")

If $A(x, r)$ uses $r(|x|)$ random bits, simulate it for **every** $r \in \{0,1\}^{r(|x|)}$ and take a majority vote (or, for RP, an OR). Correct by definition. Cost: a factor $2^{r(|x|)}$. So this is polynomial **iff the algorithm uses only $O(\log n)$ random bits**. That single observation is *why* so much of the lecture is about squeezing the random-bit budget down to $O(\log n)$ — see limited independence (§4) and PRGs (§7).

Bookend 2 — Non-constructive / non-uniform advice

> **Lemma.** If $A(x; r)$ decides a language L with error $< 2^{-|x|}$, then for every n there is a *fixed* string $r^{(n)}$ such that $A(x; r^{(n)})$ is correct for **all** $x \in \{0,1\}^n$ simultaneously.

Proof (probabilistic method / union bound). Fix n. For a random r, the bad event "$A(\cdot; r)$ is wrong on x" has probability $< 2^{-n}$ for each fixed x. There are 2^n inputs of length n, so

$$\Pr\Big[\exists x :\ A(x;r) \text{ wrong}\Big] \le \sum_{x \in \{0,1\}^n} \Pr[A(x;r) \text{ wrong}] < 2^n \cdot 2^{-n} = 1.$$

A bad event of probability < 1 leaves room for a good r, so a universal $r^{(n)}$ exists. □

> **Punchline.** This puts every **BPP** language in **P/poly** (the $r^{(n)}$ are the polynomial *advice*). But the proof is **non-constructive** — it tells you a good advice string *exists*, not how to compute it. Turning "exists" into "here it is, in polynomial time" is the entire art of the rest of the lecture.

The first amplification step (error $\to 2^{-|x|}$) is just running the algorithm $O(n)$ times and voting — standard probability amplification from Lecture 3.

---

1. The RP "small witness set" theorem — a greedy set cover

This is the first non-trivial complexity-theoretic statement, and its proof is a clean **greedy set-cover** argument worth keeping.

> **Theorem.** Let M be an RP algorithm using k random bits on inputs of length n. Then there is a set $S \subseteq \{0,1\}^k$ with $|S| = n$ such that for **every** input w of length n, some $r \in S$ makes $M(w, r)$ answer correctly.

So instead of 2^k random strings, n **fixed strings** suffice — we just don't yet know how to find them in poly time, but they exist and are *few*.

Proof. Build the Boolean matrix A of size $2^n \times 2^k$:

$$A[\text{input}, \text{rand}] = \begin{cases} 1 & \text{if } M(\text{input}, \text{rand}) \text{ is correct,} \\ 0 & \text{otherwise.} \end{cases}$$

Because M is RP (after one amplification, say), **every row has at least 2^{k-1} ones**. Now greedily cover the rows:

```
S ← ∅
while A still has uncovered rows:
    every remaining row has ≥ 2^{k-1} ones
    ⇒ by averaging, ∃ a column j hitting ≥ half of the remaining rows
    S ← S ∪ {j};  delete the rows covered by column j
```

Each chosen column halves the number of uncovered rows. Starting from 2^n rows, after n steps fewer than 1 remain. Hence $|S| \le n$. □

> **Why a column with "half the ones" must exist.** If every remaining row has a $\ge 1/2$ fraction of 1s, then the *total* count of 1s is $\ge \frac{1}{2} \cdot (\#\text{rows}) \cdot 2^k$; by averaging over the 2^k columns, some column carries $\ge \frac{1}{2}(\#\text{rows})$ of them. This "an average column is good" move is the seed of the conditional-probabilities method.

There is also a direct probabilistic proof (the "inak" on the slide): let F_x be the event that n independent runs of M all fail on x. Then $\Pr[F_x] < (1/2)^n$, and $\Pr[\bigcup_x F_x] \le \sum_x \Pr[F_x] < 2^n \cdot 2^{-n} = 1$, so a single universal random string r works — the same union bound as Bookend 2.

---

2. The method of conditional probabilities (metóda podmienených pravdepodobností)

This is the most important *constructive* technique in the lecture, and the one the exams hammer (3-cuts, monochromatic edges, K_4 colorings). Master this section.

The idea in one sentence

The probabilistic method says "a random object is good *on average*." Conditional probabilities turns that existence proof into an **algorithm**: decide the random choices $x_1, x_2, \ldots, x_n$ **one at a time**, and at each step pick the value that keeps the **conditional expectation** at least as good as before. Since the conditional expectation never drops below the unconditional one, the final fully-fixed object is at least as good as average.

Formally, suppose we want a large quantity C and we know $\mathbb{E}[C] \geq \mu$. We fix $x_1, \ldots, x_n$ in order so that

$$\mu \leq \mathbb{E}[C] \leq \mathbb{E}[C \mid x_1] \leq \mathbb{E}[C \mid x_1, x_2] \leq \cdots \leq \mathbb{E}[C \mid x_1, \ldots, x_n] = C(x_1, \ldots, x_n).$$

The middle inequalities hold because, by the law of total expectation,

$$\mathbb{E}[C \mid x_1, \ldots, x_i] = \tfrac{1}{2}\,\mathbb{E}[C \mid \ldots, x_{i+1} = 0] + \tfrac{1}{2}\,\mathbb{E}[C \mid \ldots, x_{i+1} = 1] \leq \max\{\text{the two children}\}.$$

So **always walk to the larger child.** The conditional expectation is an average of its two children, hence at most the larger one — you can always avoid going down.

> **The one prerequisite.** You must be able to **compute the conditional expectation in polynomial time** at each node. When you can compute it exactly, you greedily maximize it (§2.1). When you *can't*, you replace it with a cleverly chosen **pessimistic estimator** (§2.2).

2.1 Worked example — MaxCut / LargeCut

Color each vertex v_i with $x_i \in \{0, 1\}$. An edge is **cut** if its endpoints get different colors. Under a uniformly random coloring each edge is cut with probability $1/2$, so $\mathbb{E}[C] = m/2$. We want a deterministic cut of size $\geq m/2$.

Decide colors $x_1, \ldots, x_n$ in order. After fixing $x_1, \ldots, x_i$, split the first i vertices into $V_0(i)$ (colored 0) and $V_1(i)$ (colored 1). Let C_i be the edges already cut, and E_i the still-undecided edges (both endpoints uncolored). When we color v_{i+1}:

$$\mathbb{E}[C \mid \ldots, x_{i+1} = 0] = C_i + \left|E \cap (V_1(i) \times \{v_{i+1}\})\right| + \tfrac{|E_i|}{2},$$

$$\mathbb{E}[C \mid \ldots, x_{i+1}=1] = C_i + \big|E \cap (V_0(i) \times \{v_{i+1}\})\big| + \tfrac{|E_i|}{2}.$$

Reading this off: coloring v_{i+1} with 0 newly cuts its edges to the **already-1** neighbors; coloring it 1 cuts its edges to the **already-0** neighbors; the not-yet-decided edges still contribute $1/2$ each either way. So the rule is simply:

> **Greedy rule.** Color v_{i+1} the **opposite** of the majority color among its already-colored neighbors (break ties arbitrarily). Equivalently, place it on the side it has *more* edges *to* — those edges become cut.

Each step keeps $\mathbb{E}[C \mid \cdot]$ from dropping, so the final cut has size $\geq m/2$. Pure $O(n+m)$ greedy, no coins.

> **Exam bridge (exam-1, the 3-cut).** Partition vertices into **three** sets S, T, U. A random 3-partition cuts each edge with probability $2/3$, so $\mathbb{E}[\text{3-cut}] = \frac{2}{3}m$. The identical conditional-expectation argument — at each vertex pick the part it has the *fewest* already-placed neighbors in — deterministically yields a 3-cut of size $\geq \frac{2}{3}m$ in polynomial time. The generalization to k parts / k colors and "few monochromatic edges $\leq m/k$" (exam-3) is the same machinery.

2.2 Worked example — Set Balancing (Spencer) and the pessimistic estimator

This is the subtle, professor-pleasing case: **what if the "good event" is not an expectation you can track?** Then you track a *pessimistic upper bound on the probability of failure* instead.

Problem (Set Balancing). Given sets $S_1, \ldots, S_m \subseteq B = \{b_1, \ldots, b_n\}$, find a 2-coloring $f : B \to \{+1, -1\}$ making every set as *balanced* as possible. The **discrepancy** of set S_i is

$$\Delta(S_i, f) = \Big|\sum_{b_j \in S_i} f_j\Big|, \qquad \Delta(S, f) = \max_i \Delta(S_i, f).$$

Assume (as the slides do) $n = m$ and each $|S_i| = \delta$.

- **Spencer '85:** there is *always* a coloring with $\Delta(S, f) \leq 6\sqrt{n}$ — but his proof is non-constructive.
- **Easy randomized bound.** Color each $b_j \in_R \{+1, -1\}$. Then $T_i := \Delta(S_i, f)$ is a sum of δ independent ± 1's, $\mathbb{E}[T_i] = 0$, so by Chernoff

  $$\Pr[\,|T_i| \geq a\,] \leq 2\,e^{-a^2/(2\delta)}.$$

  Union over the n sets and choose $a = \sqrt{2\delta \ln 2n}$:

$$\Pr\left[\exists i:\ |T_i| \geq a\right] \leq n \cdot 2\, e^{-a^2/(2\delta)} = n \cdot 2 \cdot \tfrac{1}{2n} = 1 \ \Rightarrow\ (\text{strict } < 1)\ \exists f:\ \Delta(S,f) \leq \sqrt{2\delta \ln 2n}.$$

Now derandomize. The obstacle: the "good" event (Δ small) is a max of absolute values, not a sum — its conditional expectation is awkward. **Fix:** drive down a *pessimistic estimator* $\widetilde{P}_v$ of the probability that the final leaf is a **bad** coloring. Build the binary tree over colorings; a root-to-leaf path is *good* if it induces $\Delta(S,f) \leq \sqrt{2\delta \ln 2n}$, *bad* otherwise. Let P_v be the true fraction of bad leaves under v. We have $P_v = \frac{1}{2}(P_{v1} + P_{v,-1}) \geq \min\{P_{v1}, P_{v,-1}\}$, so walking to the smaller child never increases P. We can't compute P_v, so we want $\widetilde{P}_v$ satisfying four conditions:

1. $\widetilde{P}_{\text{root}} < 1$,
2. $P_v \leq \widetilde{P}_v$ at every node (it really is an upper bound),
3. $\widetilde{P}_v \geq \min\{\widetilde{P}_{v1}, \widetilde{P}_{v,-1}\}$ (a child is no worse),
4. $\widetilde{P}_v$ is computable in polynomial time.

The estimator comes from the Chernoff moment-generating-function bound *before* taking limits. For a parameter t,

$$\Pr[\,|T_i| \geq \Delta\,] \leq \frac{\mathbb{E}[e^{tT_i}]}{e^{t\Delta}}, \qquad \widetilde{P}_{\text{root}} = \sum_{i=1}^{n} \frac{\mathbb{E}[e^{tT_i}]}{e^{t\Delta}}.$$

Because a coloring with $\Delta \leq \sqrt{2\delta \ln 2n}$ exists, there is a t making $\widetilde{P}_{\text{root}} < 1$ (condition 1). At a node v where $f_1, \ldots, f_j$ are already decided, split $T_i = T_{i,d}^{j} + T_{i,n}^{j}$ into the **d**ecided and **n**ot-yet-decided parts, and estimate

$$\widetilde{P}_v = \sum_{i=1}^{n} \frac{\mathbb{E}\left[e^{t\,T_{i,n}^{j}}\right]}{e^{t(\Delta - T_{i,d}^{j})}}.$$

The "averaging over the next bit" identity $\widetilde{P}_v = \frac{1}{2}\widetilde{P}_{v1} + \frac{1}{2}\widetilde{P}_{v,-1} \geq \min\{\widetilde{P}_{v1}, \widetilde{P}_{v,-1}\}$ gives condition 3, and each $\mathbb{E}[e^{tT_{i,n}}]$ is a product of per-element factors $\frac{1}{2}(e^t + e^{-t})$, computable in polynomial time (condition 4).

> **The finish (why $\widetilde{P} < 1$ forces a *good* leaf).** Walk down always to the smaller-$\widetilde{P}$ child. By condition 3, $\widetilde{P}$ never increases, so the leaf has $\widetilde{P} < 1$. But at a leaf the coloring is fully fixed, so the true P is either 0 (good) or 1 (bad), and $P \leq \widetilde{P} < 1$ forces $P = 0$. We have **constructed** a good coloring. □

> **The deep point.** Conditional probabilities does not require the *quantity* you care about to be an expectation. It only requires a polynomial-time-computable quantity that (i) upper-bounds the failure probability, (ii) starts below 1, and (iii) has the "average-of-children" property. That quantity is the **pessimistic estimator**, and the MGF/Chernoff bound is the standard place to

> get one.

---

3. Parallel MIS (Luby) and derandomization via pairwise independence

The **Maximal Independent Set** problem: find an inclusion-maximal independent set. Luby's randomized algorithm is *parallel* and finishes in $O(\log n)$ rounds; the point here is that it can be **derandomized into NC^2** (poly processors, $O(\log^2 n)$ depth) because its analysis only ever uses **pairwise independence**.

The algorithm

```
MIS ← ∅
repeat:
    each vertex v marks itself with probability 1/(2·d(v))        → set S
    on each edge with both endpoints marked, the higher-degree endpoint wins → S′
    MIS ← MIS ∪ S′
    V ← V − ( S′ ∪ N(S′) )       (remove the new IS vertices and their neighbors)
until V = ∅
```

Two claims drive everything:

- $\mathbb{E}[\#\text{iterations}] = O(\log n)$;
- in every iteration a **constant fraction of the edges** is removed.

The good/bad bookkeeping

Call a vertex **bad** if more than $2/3$ of its neighbors have degree $\geq d(v)$ (it is "dominated"); otherwise **good**. An edge is **bad** if *both* endpoints are bad.

- **L1 — at least half the edges are good.** Orient each edge toward its higher-degree endpoint and define an injection $f : E_B \to E$ that charges each bad edge to two out-edges of its endpoints; this is impossible for more than $|E|/2$ edges. So $\geq |E|/2$ edges are good.
- **L2 (uses *full* independence).** If v is good, $\Pr[N(v) \cap S \neq \emptyset] > 2\alpha$, where $\alpha := (1 - e^{-1/6})/2$. *Proof:* let $L(v) = \{w \in N(v) : d(w) \leq d(v)\}$; goodness gives $|L(v)| \geq d(v)/3$, and

$$\Pr[N(v) \cap S \neq \emptyset] = 1 - \prod_{w \in N(v)} \Pr[w \notin S] \geq 1 - \prod_{w \in L(v)} \left(1 - \tfrac{1}{2d(w)}\right) \geq 1 - \left(1 - \tfrac{1}{2d(v)}\right)^{|L(v)|} \geq 1 - e^{-1/6}.$$

- **L3 (uses only *pairwise* independence).** $\Pr[w \notin S' \mid w \in S] \le 1/2$. *Proof:* w is killed only by a marked neighbor z of $\ge$ degree; sum over $H(w) = \{z \in N(w) : d(z) \ge d(w)\}$ and use pairwise independence:

  $$\Pr[w \notin S' \mid w \in S] \le \sum_{z \in H(w)} \Pr[z \in S \mid w \in S] = \sum_{z \in H(w)} \Pr[z \in S] = \sum_{z \in H(w)} \tfrac{1}{2d(z)} \le \sum_{z \in H(w)} \tfrac{1}{2d(w)} \le \tfrac{1}{2}.$$

- **L4.** If v good, $\Pr[v \in N(S')] \ge \alpha$. Combine L2 and L3:
  $\Pr[v \in N(S')] \ge \Pr[w \in S' \mid w \in S] \cdot \Pr[N(v) \cap S \ne \emptyset] \ge \tfrac{1}{2} \cdot 2\alpha = \alpha$.

- **Corollaries.** A good vertex is removed with prob $\ge \alpha$; a good edge is removed with prob $\ge \alpha$.

- **Progress lemma.** $\mathbb{E}[|E_j| \mid E_{j-1}] \le |E_{j-1}|(1 - \alpha/2)$, because $\ge$ half the edges are good (L1) and each good edge dies with prob $\ge \alpha$. Iterating, $\mathbb{E}[|E_j|] \le m(1-\alpha/2)^j \le m\, e^{-j\alpha/2} \le 1$ once $j > \frac{2}{\alpha} \ln m = O(\log n)$.

Why this derandomizes

The progress lemma is the crux, and its proof (L3, the part that actually bounds removals) uses **only pairwise independence** of the marks. So:

- replace the truly-random marks by marks drawn from a **pairwise-independent sample space of polynomial size** (§4);
- there are only polynomially many sample points, so spread them across **polynomially many processors**; by the lemma, at least one sample point removes $\ge$ the expected fraction of edges. Pick it (a deterministic max over poly choices).

Each round is one deterministic parallel step over a poly-size space; $O(\log n)$ rounds $\Rightarrow \mathbf{NC}^2$.

> **The lemma that licenses the swap (full → pairwise).** For indicators X_i with $p_i = \Pr[X_i = 1]$:
>
> 1. independent: $\Pr[\sum X_i > 0] \ge 1 - \prod_i (1 - p_i)$;
> 2. **pairwise** independent: $\Pr[\sum X_i > 0] \ge \frac{1}{2} \min\{1, \sum_i p_i\}$.
>
> Part 2 is a Bonferroni / inclusion–exclusion bound:
> $\Pr[\sum X_i > 0] \ge \sum_i p_i - \frac{1}{2} \sum_{i \ne j} \Pr[X_i = X_j = 1] = \sum p_i - \frac{1}{2} \sum_{i \ne j} p_i p_j \ge \sum p_i \big(1 - \frac{1}{2} \sum p_i\big)$
> , which is $\ge \frac{1}{2} \sum p_i$ when $\sum p_i \le 1$; and when $\sum p_i \ge 1$ pick a sub-collection with $\frac{1}{2} \le \sum_S p_i \le 1$ to get $\ge 1/4$. The point: "*at least one event fires*" needs only **pairwise** correlations — exactly what a small sample space can supply.

---

4. Limited independence and small sample spaces

This is the engine behind §3 and the exams. The principle:

> If the analysis of a randomized algorithm only ever uses interactions among k **variables at a time**, then we may feed it variables that are merely k**-wise independent** instead of fully independent. Such variables live in a **tiny sample space**, generated from few truly-random bits, which we can then **enumerate**.

We want $f : \Sigma^{\ell(n)} \to \Sigma^{r(n)}$ (short seed $\to$ long pseudo-random output) with $\Pr_r[A(x,r)=1] \approx \Pr_y[A(x,y)=1]$ for the *real* uniform y.

4.1 n pairwise-independent uniform bits from $\lceil \log(n+1) \rceil$ truly-random bits

This is the construction to have at your fingertips (it is *the* answer to the pairwise-independent-coloring exam questions).

- Let $t = \lceil \log(n+1) \rceil$ and draw $b_1, \dots, b_t$ **truly** uniform.
- Pick n distinct **nonempty** subsets $J_1, \dots, J_n \subseteq \{1, \dots, t\}$ (there are $2^t - 1 \geq n$ of them).
- Output $r_i := \bigoplus_{j \in J_i} b_j$.

Uniform: each r_i contains at least one b_s; conditioning on the others, $r_i = (\bigoplus_{j \in J_i \setminus \{s\}} b_j) \oplus b_s$ flips with b_s, so it is fair. **Pairwise independent:** for $J_i \neq J_{i'}$ there is (wlog) an $s \in J_i \setminus J_{i'}$; b_s randomizes r_i but not $r_{i'}$, so

$$\Pr[r_i = a \wedge r_{i'} = a'] = \tfrac{1}{4} = \Pr[r_i = a]\Pr[r_{i'} = a'].$$

> **Punchline.** n pairwise-independent bits cost only $t = O(\log n)$ truly-random bits, i.e. a sample space of size $2^t = O(n)$. **Enumerate all $O(n)$ seeds** in polynomial time and keep the best outcome — the random bits are gone. This is exactly how the pairwise-independent graph-colorings of the exams are derandomized: the random coloring uses $O(\log n)$ true bits, you try all $O(\text{poly})$ seeds, and you keep the coloring with $\leq m/k$ monochromatic edges. Running time is poly(n, m, k).

4.2 Chor–Goldreich 2-independent generator (from 2-universal hashing)

A function f is an (m, t, ℓ)**-2-independent generator** if

1. $f : \Sigma^m \to \Sigma^{\ell \cdot t}$, output split as $f(x) = f_1(x)\, f_2(x) \cdots f_\ell(x)$, each $|f_i(x)| = t$;
2. f is computable in time poly$(t\ell)$;
3. $\forall i \neq j,\ \forall \alpha, \beta \in \Sigma^t :\ \Pr[f_i(x) = \alpha \wedge f_j(x) = \beta] = 2^{-2t}$.

> **Theorem.** For all ℓ, t, m with $2^m \geq \ell$ and $m \geq t$, an $(2m, t, \ell)$-2-independent generator exists.

Construction. Use the **2-universal** family $H = \{h_{a,b}(x) = ax + b : a, b \in GF[2^m]\}$. Fix distinct $\alpha_1, \ldots, \alpha_\ell \in GF[2^m]$ and set

$$f(a,b) = h_{a,b}(\alpha_1)\, h_{a,b}(\alpha_2) \cdots h_{a,b}(\alpha_\ell), \quad \text{keeping } t \text{ bits of each } h_{a,b}(\alpha_i).$$

Pairwise independence of $h_{a,b}$ over the random seed (a,b) gives $\Pr_{a,b}[f_i = u \wedge f_j = v] = 2^{-2t}$ — exactly property 3. The seed is $2m$ bits, far fewer than the ℓt output bits.

4.3 k-independent uniform bits from linear codes (Alon–Babai–Itai)

To go beyond pairwise, the right language is **linear codes**.

> **Theorem (Alon–Babai–Itai).** Let $L_1, \ldots, L_m \in \{0,1\}^\ell$ be such that **any k of them are linearly independent over $GF[2]$**. Draw $R \in_R \{0,1\}^\ell$ and set $X_i := \langle L_i, R \rangle = \sum_{j=1}^{\ell} L_{i,j} r_j \bmod 2$. Then $X_1, \ldots, X_m$ are **k-wise independent** (and uniform).

Proof. Take any k indices $i_1, \ldots, i_k$. The corresponding rows form a $k \times \ell$ matrix H of rank k (independence), so $R \mapsto (X_{i_1}, \ldots, X_{i_k}) = HR^\top$ is a surjective linear map whose every fiber has size $2^{\ell - k}$. Hence

$$\Pr[X_{i_1} = d_1, \ldots, X_{i_k} = d_k] = \frac{2^{\ell-k}}{2^\ell} = 2^{-k} = \prod_{j=1}^{k} \Pr[X_{i_j} = d_j]. \qquad \square$$

Where do the L_i come from? From the **parity-check matrix** of a code: for an $[n, k, d]$ code, *any $d-1$ columns of H are linearly independent*. Take the BCH-style check matrix with $n = 2^d - 1$ columns and rows the odd powers of the field elements $y_1, \ldots, y_n \in GF[2^d]$,

$$H = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ y_1 & y_2 & \cdots & y_n \\ y_1^3 & y_2^3 & \cdots & y_n^3 \\ \vdots & & & \vdots \\ y_1^{2t-1} & y_2^{2t-1} & \cdots & y_n^{2t-1} \end{pmatrix},$$

which has minimum distance $2t+2$, so **every $2t+1$ columns are independent**. The columns live in $\{0,1\}^\ell$ with $\ell = (t+1)d = (t+1)\log n$. Thus from $R \in_R \{0,1\}^{(t+1)\log n}$ we obtain n variables that are $(2t+1)$-wise independent and uniform — using only $O(t \log n)$ random bits.

4.4 k-independent bits with an *arbitrary* distribution

Finally, k-wise independent variables with **prescribed (possibly biased) marginals** $\Pr[X_i = 1] = p_i$, using a sample space of size $O(n^k)$.

> **Theorem.** For a prime p with $n \le p \le 2n$ and $k \ge 1$, there is a uniform probability space (Ω, P) with $|\Omega| = p^k$ and k-independent variables $\widehat{X}_1, \dots, \widehat{X}_n$ over it whose marginals match the targets up to rounding: $\mathbf{dist}(\widehat{X}_i, X_i) \le 1/p \le 1/n$.

Construction. Let Ω be all polynomials $q = q_1 + q_2 x + \cdots + q_k x^{k-1}$ of degree $\le k-1$ over $F = GF(p)$ (so $|\Omega| = p^k$), chosen uniformly. Set $Y_i = q(i)$, then threshold: $\widehat{X}_i = 1$ iff $q(i) < p \cdot p_i$. Because

$$\begin{pmatrix} 1 & i_1 & \cdots & i_1^{k-1} \\ 1 & i_2 & \cdots & i_2^{k-1} \\ \vdots & & & \vdots \\ 1 & i_k & \cdots & i_k^{k-1} \end{pmatrix} \begin{pmatrix} q_1 \\ \vdots \\ q_k \end{pmatrix} = \begin{pmatrix} Y_{i_1} \\ \vdots \\ Y_{i_k} \end{pmatrix}$$

is a **Vandermonde** (regular) system, for every target $(d_1, \dots, d_k)$ there is a **unique** q producing $Y_{i_j} = d_j$, so $\Pr[Y_{i_1} = d_1, \dots, Y_{i_k} = d_k] = p^{-k}$ — exactly k-wise independence and $\Pr[q(i) = c] = 1/p$. The threshold induces at most a $1/p$ rounding error on each marginal. □

---

5. Schöning's k-SAT, Promise-Ball-k-SAT, and covering codes

A self-contained, beautiful derandomization: Schöning's random-walk k-SAT algorithm is turned into the **fastest known deterministic** k-SAT algorithm, via **covering codes**, losing only an ε in the exponent.

Schöning's randomized algorithm

```
choose a ∈ {0,1}^n uniformly at random
repeat (a bounded number of times):
    if F(a)=1 return a
    pick an unsatisfied clause C; flip one of its literals chosen uniformly at rand
```

> **Theorem (Schöning).** If $F \in$ SAT (k-CNF), one run finds a satisfying assignment with

probability $\geq \left(\frac{k}{2(k-1)}\right)^n$.

Proof. Let α^* be a satisfying assignment and α the random start. Then $\Pr[\mathrm{dist}(\alpha, \alpha^*) = r] = \binom{n}{r}2^{-n}$. From Hamming distance r, the random walk drifts toward α^* — each forced flip moves *toward* α^* with probability $\geq 1/(k-1)$ (the unsatisfied clause has a literal wrong in α but right in α^*). So $\Pr[\text{reach } \alpha^* \mid \mathrm{dist} = r] \geq (k-1)^{-r}$, and

$$\Pr[\text{success}] \geq \sum_{r=0}^{n} \binom{n}{r} 2^{-n}(k-1)^{-r} = 2^{-n}\left(1 + \tfrac{1}{k-1}\right)^n = \left(\tfrac{k}{2(k-1)}\right)^n. \qquad \square$$

Repeating $\left(\frac{2(k-1)}{k}\right)^n$ times gives an **RP** algorithm running in $O^*(1.33^n)$ for 3-SAT and $O^*(1.5^n)$ for 4-SAT.

Promise-Ball-k-SAT, deterministically

Promise-Ball-k-SAT. *Promise:* a satisfying assignment lies inside $\mathrm{Ball}_r(\alpha)$ for given α, r. *Find* any satisfying assignment (not necessarily inside the ball).

Schöning's lemma. If $\mathrm{Ball}_r(\alpha)$ contains a satisfying β, then Schöning finds *some* satisfying assignment with probability $\geq (k-1)^{-r}$.

The deterministic version is a bounded-depth search:

```
Search(F, α, r):
    if F(α)=1 return true
    if r=0    return false
    C ← an unsatisfied clause of F        (length ≤ k)
    for each literal u in C:
        if Search(F|u=1, flip α at u, r-1) = true return true
    return false
```

The recursion branches on the literals of an unsatisfied clause. A naïve count gives k^r leaves; the sharper analysis notes that an unsatisfied clause needs to flip only $k-1$ "useful" literals (the assignment already disagrees with one), so

$$\mathrm{time}\big(\mathrm{Search}(F, \alpha, r)\big) = O^*\big((k-1)^r\big).$$

From the ball back to the whole cube — covering codes (Dantsin et al.)

We do not know where α^* is, so we **cover** $\{0,1\}^n$ by balls and run $\mathbf{Search}$ from each ball's center.

> **Reduction.** If A solves Promise-Ball-k-SAT in $O^*(a^r)$, then there is a B solving k-SAT in $O^*\big(\big(\frac{2a}{a+1}\big)^n\big)$; B is deterministic if A is.

Idea. Take a **covering code**: a set of codewords (ball centers) such that every $\alpha \in \{0,1\}^n$ is within distance r of some center. Run the deterministic Promise-Ball solver from each center. If $F \in \mathbf{SAT}$, some satisfying assignment sits within r of a center, and that call finds it. Balancing the ball radius against the number of centers (more, smaller balls vs fewer, larger ones) gives the $\big(\frac{2a}{a+1}\big)^n$ bound.

With $a = k-1$ (deterministic Search), B runs in $O^*\Big(\big(\frac{2(k-1)}{k}\big)^n\Big)$ — **matching the randomized Schöning running time, deterministically.**

> **The covering-code existence lemma.** Work over the alphabet $\{1,\dots,k\}^t$ (a k-ary code, because each useful flip in a clause is one of k literal choices). With $B_r^{(k)}(u) = \{u' : d_H(u,u') \le r\}$ and $\mathbf{vol}_k(t,r) = \binom{t}{r}(k-1)^r$, there exists a code $C \subseteq \{1,\dots,k\}^t$ of covering radius r with
>
> $$|C| \le \frac{t \ln k \cdot k^t}{\binom{t}{r}(k-1)^r}.$$
>
> *Proof (probabilistic method).* Pick $m = \dfrac{t \ln k \cdot k^t}{\binom{t}{r}(k-1)^r}$ centers uniformly at random. A fixed word w' is uncovered with probability
>
> $$\Big(1 - \tfrac{\mathbf{vol}_k(t,r)}{k^t}\Big)^m < e^{-\mathbf{vol}_k(t,r)\,m/k^t} = e^{-t\ln k} = k^{-t}.$$
>
> Union over all k^t words gives < 1, so with positive probability *every* word is covered $\Rightarrow$ such a code exists. The lecture then makes the radius $r = t/k$ and uses $\binom{t}{t/k}(k-1)^{t/k}$ estimates to push the deterministic Promise-Ball-k-SAT to $O^*\big((k-1+\varepsilon)^r\big)$ for any $\varepsilon > 0$.

> **Why "find a *good* code" is the hard part.** The probabilistic argument shows a covering code *exists*; the algorithm needs an *explicit* one. The slides resolve this by pre-computing a small code for fixed r, k, t (via a maximal independent family of unsatisfied clauses), so the search

tree has only $\approx (k-1+\varepsilon)^r$ leaves. The recursion $(\alpha, r) \rightsquigarrow (\alpha(w'), r-\Delta)$ shrinks the radius by $\Delta = t - 2t/k$ each level, and the number of leaves is bounded by $|C|^{r/\Delta} \le (k-1)^{t^2/\Delta \cdot r/t} \rightsquigarrow (k-1+\varepsilon)^r$.

---

6. Pseudorandom generators and hardness vs. randomness

The deepest part: a *general* theorem that derandomizes **all of BPP** under a plausible hardness assumption. The slogan:

> **Hardness ⇒ randomness.** A function that is *hard to predict* can be used to *manufacture* pseudo-randomness; pseudo-randomness lets us replace coins by enumeration. If sufficiently hard functions exist, $\mathsf{BPP} = \mathsf{P}$.

Definitions

> **(S, ε)-pseudorandom.** A distribution R over $\{0,1\}^m$ is (S,ε)-pseudorandom if **every circuit C of size $\le S$** is fooled: $\left|\Pr[C(R)=1] - \Pr[C(U_m)=1]\right| < \varepsilon$. ($U_m$ = uniform.)

> **PRG.** A function $G : \{0,1\}^* \to \{0,1\}^*$, computable in time 2^n, is an **$S(\ell)$-pseudorandom generator** if $|G(z)| = S(|z|)$ and for every ℓ the distribution $G(U_\ell)$ is $\left(S(\ell)^3, \frac{1}{10}\right)$-pseudorandom.

The generator stretches an ℓ-bit seed into $S(\ell)$ output bits that fool every circuit of size $S(\ell)^3$. The bigger the stretch we can certify, the more randomness we save.

PRG ⇒ derandomization of BPP

> **Theorem.** If an $S(\ell)$-PRG exists (S time-constructible, nondecreasing), then for every polytime $\ell : \mathbb{N} \to \mathbb{N}$, $\mathsf{BPTIME}(S(\ell(n))) \subseteq \mathsf{DTIME}\left(2^{c\,\ell(n)}\right)$.

Proof sketch. Let $L \in \mathsf{BPTIME}(S(\ell(n)))$ via $A(x,r)$ with r of length $m \le S(\ell(n))$ and $\Pr_r[A(x,r) = L(x)] \ge 2/3$. Replace the m truly-random bits by $G(z)$ for $z \in_R \{0,1\}^{\ell(n)}$. Enumerate **all $2^{\ell(n)}$ seeds**, compute each $G(z)$ (time $2^{\ell(n)}$), run the length-$S(\ell)$ computation, and

take a majority. The success probability can shift by at most $1/10$ in passing from U_m to $G(U_\ell)$ — otherwise $A(x,\cdot)$ (a small circuit) would *distinguish* $G(U_\ell)$ from uniform, contradicting the PRG. So correctness stays $\geq 2/3 - 1/10$. Total time $\max\{2^{c_1\ell(n)}, S(\ell(n))\} = 2^{c\ell(n)}$.

> **Punchline.** For $\ell(n) = \log n$ this is $2^{c\log n} = \mathrm{poly}(n)$, i.e. **BPP = P**. Derandomizing **BPP** reduces to **building a PRG with a logarithmic-length seed.**

The hardness assumption

> **Average-case hardness.** $H_{avg}(f)(n)$ is the largest circuit size S such that *no* circuit C of size S satisfies $\Pr_{x\in\{0,1\}^n}[C(x) = f(x)] \geq \frac{1}{2} + \frac{1}{S}$. I.e. f cannot even be *approximated* — predicted noticeably better than a coin flip — by circuits up to size S.

Such hard f are believed to exist (e.g. they follow from **NP** $\not\subseteq$ **P/poly**, "3-SAT is hard").

Warm-ups — append one or two hard bits (via Yao)

> **Yao's theorem (unpredictability = pseudorandomness).** If for a distribution Y on $\{0,1\}^m$ no size-$2S$ circuit predicts bit i from bits $1..i-1$ with advantage $> \varepsilon/m$, then Y is (S,ε)-pseudorandom.

- **One hard bit.** If $\exists f \in \mathbf{DTIME}(2^{O(n)})$ with $H_{avg}(f) \geq n^4$, then $G(z) = z \cdot f(z)$ is an $(\ell+1)$-PRG. The only "new" bit is $f(z)$; predicting it with advantage $> 1/(20(\ell+1))$ would give a circuit approximating f better than $1/2 + 1/\ell^4$ — contradicting $H_{avg}(f) \geq n^4$. By Yao, $G(U_\ell)$ is pseudorandom.
- **Two hard bits.** $G(z) = z_{1..\ell/2}\, f(z_{1..\ell/2})\, z_{\ell/2+1..\ell}\, f(z_{\ell/2+1..\ell})$ is an $(\ell+2)$-PRG. The second hard bit is handled by an **averaging argument**: a predictor depending on two independent halves X, Y can be fixed on one half ($\exists x$ with $\Pr_Y[A(x,Y)] \geq \Pr_{X,Y}[A(X,Y)]$), reducing to a circuit D that approximates f on the other half — again a contradiction with $H_{avg}(f)$.

The Nisan–Wigderson generator (the real thing)

> **Theorem (Nisan–Wigderson).** S time-constructible, nondecreasing. If there is $f \in \mathbf{DTIME}(2^{O(n)})$ with $H_{avg}(f)(n) \geq S(n)$, then there is an $S(\delta\ell)^\delta$-PRG, for some $\delta > 0$.

The two warm-ups appended **one or two** independent hard bits. NW appends **many** hard bits $f(Z_{I_1}), \ldots, f(Z_{I_m})$ — but to make m large from a seed of length ℓ, the index sets I_j must **overlap**. They must overlap *little*, or the outputs would be correlated. The right object is a **combinatorial design**.

> **(ℓ, n, d)-design.** A family $I = \{I_1, \ldots, I_m\}$ with $I_j \subseteq \{1, \ldots, \ell\}$, $|I_j| = n$, and pairwise overlaps $|I_j \cap I_{j'}| \leq d$. (Almost-disjoint n-subsets of an ℓ-universe.)

> **NW generator.** $\mathrm{NW}_I^f(Z) = f(Z_{I_1}) \cdot f(Z_{I_2}) \cdots f(Z_{I_m})$, where $Z \in \{0,1\}^\ell$ and Z_{I_j} is Z restricted to the indices in I_j.

Two lemmas assemble the theorem:

- **L14 (designs exist & are constructible).** For $n > d$ and $\ell > 10n^2/d$, an algorithm computes an (ℓ, n, d)-design with $m = 2^{d/10}$ subsets in time $2^{O(\ell)}$. *Greedy + probabilistic:* a random J (each element in with prob $2n/\ell$) has $\mathbb{E}[|J|] = 2n$ and $\mathbb{E}[|J \cap I_i|] = 2n^2/\ell < d/5$; Chernoff makes $|J| \geq n$ and all overlaps $\leq d$ simultaneously with probability ≥ 0.4, so the greedy "add the first valid J" succeeds and builds $2^{d/10}$ sets.
- **L15 (a hard f on a design $\Rightarrow$ PRG).** If I is an (ℓ, n, d)-design with $|I| = 2^{d/10}$ and $H_{avg}(f) > 2^{2d}$, then $\mathrm{NW}_I^f(U_\ell)$ is $(H_{avg}/10, 1/10)$-pseudorandom.

> **Proof of L15 — the key "overlap pays off" step.** Suppose (for contradiction, via Yao) a size-$S/2$ circuit C predicts output bit i, i.e. $\Pr_Z[C(f(Z_{I_1}), \ldots, f(Z_{I_{i-1}})) = f(Z_{I_i})] \geq \frac{1}{2} + \frac{1}{10 \cdot 2^{d/10}}$. Split $Z = (Z_1, Z_2)$ into the bits inside I_i and the rest, and **fix** Z_2 by an averaging argument. Now every *other* output $f(Z_{I_j})$ depends on the variable Z_1 only through $|I_i \cap I_j| \leq d$ coordinates — so each is computed by a tiny circuit of size $\leq d\,2^d$. Hard-wiring these into C yields a circuit B of size
>
> $$\underbrace{2^{d/10}}_{\#\text{other outputs}} \cdot \underbrace{d\,2^d}_{\text{each cheap}} + \underbrace{S/2}_{C} < S$$
>
> that approximates f on n-bit inputs with advantage $\geq \frac{1}{2} + \frac{1}{S}$ — **contradicting** $H_{avg}(f) > 2^{2d}$. $\square$

> **The whole arc in one breath.** A function f that is hard *on average* (no small circuit approximates it) can be sampled at m *almost-independent* windows of one short seed; because the windows overlap in $\leq d$ places, an adversary trying to predict one output bit could be

> turned into a small circuit for f — impossible. So the output *looks* random to all small circuits: a PRG. Plug the PRG into the BPP theorem, set the seed to $O(\log n)$, and **randomness was never needed**: $\mathsf{BPP} = \mathsf{P}$.

Closing themes table

Method	What the analysis *needed*	What you replace coins with	Cost / Yield
Enumeration	nothing	try all $2^{r(n)}$ seeds	poly iff $r(n) = O(\log n)$
Non-uniform advice	error $< 2^{-n}$	one fixed string per length	$\mathsf{BPP} \subseteq \mathsf{P/poly}$, non-constructive
RP matrix theorem	majority of seeds correct	n seeds (greedy set cover)	small witness set, size n
Conditional probabilities	only the *expectation* $\mathbb{E}[C]$	fix bits one-by-one, steer to better child	deterministic $\geq \mathbb{E}[C]$ object
Pessimistic estimator	a poly-computable upper bound on failure	drive $\widetilde{P}_{\text{root}} < 1$ down the tree	works when $\Pr[\text{good}]$ is not an expectation (Set Balancing)
Pairwise independence	only 2-variable interactions	$\bigoplus_{j \in J_i} b_j$, $O(\log n)$ bits	$O(n)$-size space, enumerate (MIS $\to \mathsf{NC}^2$)
k-wise independence	k-variable interactions	linear codes (any k columns indep.)	space $\mathrm{poly}(n)$ / $O(n^k)$
Covering codes	a ball contains a solution	run Promise-Ball solver at every center	deterministic k-SAT $\approx$ randomized time
PRG (Nisan–Wigderson)	fool size-S circuits	stretch hard function over a design	hardness $\Rightarrow \mathsf{BPP} = \mathsf{P}$

The unifying lesson: derandomization is *accounting*. Find out exactly how much randomness the proof actually consumed — the expectation only? pairwise correlations only? indistinguishability to small circuits only? — and supply *precisely that much* from a tiny, enumerable source. The less the analysis truly needed, the cheaper (and sometimes free) the coins turn out to be.