Randomized Algorithms

Limited Independence and Derandomization

Alex Gittens
Department of Computer Science, Rensselaer Polytechnic Institute

Fall 2026

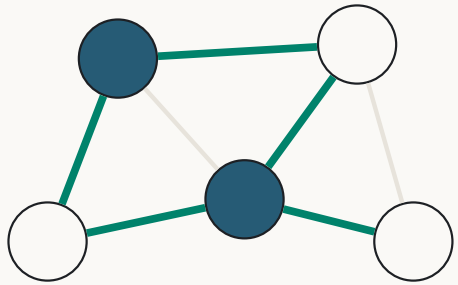
Last time: a random cut has $m/2$ edges on average

Random cut

Input: graph $G = (V, E)$, n vertices, m edges.

For each vertex v, flip a fair bit B_v.

Output: the cut between $\{v : B_v = 0\}$ and $\{v : B_v = 1\}$.



blue: $B_v = 1$; cream: $B_v = 0$; green: cut edge

The number of cut edges is

$$C = \sum_{\{u,v\} \in E} \mathbf{1}\{B_u \neq B_v\}.$$

Last time we showed $\mathbb{E}[C] = m/2$, then used conditional expectation to find a cut at least that large deterministically.

Which facts about the random bits did the analysis use?

By linearity of expectation,

$$\begin{aligned}\mathbb{E}[C] &= \sum_{\{u,v\}\in E} \Pr(B_u \neq B_v) \\ &= \sum_{\{u,v\}\in E} \big[\Pr(B_u{=}0, B_v{=}1) + \Pr(B_u{=}1, B_v{=}0)\big] \\ &= m\Big(\frac{1}{4} + \frac{1}{4}\Big) = \frac{m}{2}.\end{aligned}$$

Which facts about the random bits did the analysis use?

By linearity of expectation,

$$\begin{aligned}\mathbb{E}[C] &= \sum_{\{u,v\}\in E} \Pr(B_u \neq B_v) \\ &= \sum_{\{u,v\}\in E} \big[\Pr(B_u{=}0, B_v{=}1) + \Pr(B_u{=}1, B_v{=}0)\big] \\ &= m\Big(\frac{1}{4} + \frac{1}{4}\Big) = \frac{m}{2}.\end{aligned}$$

The only fact used: for each edge, the *pair* (B_u, B_v) takes each of the four values $00, 01, 10, 11$ with probability $\frac{1}{4}$. The analysis never looks at three bits together.

Question

The algorithm flips n random bits. Could bits computed from far fewer random bits still have this property, so that the same analysis works?

Today: how much randomness does an analysis need?

Suppose the analysis still works when all the random bits are computed from a **seed**: a uniformly random string of only $O(\log n)$ bits.

Then there are only $2^{O(\log n)} = \text{poly}(n)$ possible seeds, so the algorithm has only polynomially many possible runs.

Part 1 With few random bits we can try *every* run and keep the best: a deterministic algorithm.

Part 2 The runs don't depend on each other, so we can try them all *at the same time*: a deterministic *parallel* algorithm.

Part 3 Some algorithms must stay randomized and need more than an expectation. How much independence does a guarantee that holds with high probability need?

PART 1

Reducing the randomness an algorithm uses

1

Pairwise independence

Pairwise independence

Random variables $X_1, \ldots, X_n$ are **pairwise independent** if every two of them are independent: for all $i \neq j$ and all values x, y,

$$\Pr(X_i = x,\ X_j = y) = \Pr(X_i = x)\ \Pr(X_j = y).$$

Pairwise independence

Pairwise independence

Random variables $X_1, \ldots, X_n$ are **pairwise independent** if every two of them are independent: for all $i \neq j$ and all values x, y,

$$\Pr(X_i = x,\ X_j = y) = \Pr(X_i = x)\ \Pr(X_j = y).$$

For bits, there is a simpler test. Bits X, Y are *fair and independent* exactly when

$$\Pr(X = x,\ Y = y) = \tfrac{1}{4} \quad \text{for all four pairs } (x, y).$$

Pairwise independence

Pairwise independence

Random variables $X_1, \ldots, X_n$ are **pairwise independent** if every two of them are independent: for all $i \neq j$ and all values x, y,

$$\Pr(X_i = x,\ X_j = y) = \Pr(X_i = x)\ \Pr(X_j = y).$$

For bits, there is a simpler test. Bits X, Y are *fair and independent* exactly when

$$\Pr(X = x,\ Y = y) = \tfrac{1}{4} \quad \text{for all four pairs } (x, y).$$

Why: if each pair has probability $\frac{1}{4}$, then $\Pr(X = x) = \Pr(X{=}x, Y{=}0) + \Pr(X{=}x, Y{=}1) = \frac{1}{2}$, so X is fair. Likewise Y. Then $\Pr(X{=}x, Y{=}y) = \frac{1}{4} = \frac{1}{2} \cdot \frac{1}{2}$: independent. The converse is immediate.

Recall: XOR

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

$x \oplus y$ ("exclusive or") is addition mod 2: it is 1 when exactly one of x, y is 1.

- $x \oplus 0 = x$: XOR with 0 changes nothing.
- $x \oplus 1 = 1 - x$: XOR with 1 flips the bit.

XOR is associative and commutative, so $x_1 \oplus x_2 \oplus \ldots \oplus x_k$ makes sense: it is 1 when an odd number of the x_i are 1.

Recall: the inner product mod 2

For bit strings $s, a \in \{0,1\}^L$, the **inner product mod 2** is

$$\langle s, a \rangle = s_1 a_1 \oplus s_2 a_2 \oplus \ldots \oplus s_L a_L,$$

the XOR of the bits s_i at the positions where $a_i = 1$.

Example: $\langle 110, 011 \rangle = (1 \cdot 0) \oplus (1 \cdot 1) \oplus (0 \cdot 1) = 0 \oplus 1 \oplus 0 = 1$.

Recall: the inner product mod 2

For bit strings $s, a \in \{0,1\}^L$, the **inner product mod 2** is

$$\langle s, a \rangle = s_1 a_1 \oplus s_2 a_2 \oplus \ldots \oplus s_L a_L,$$

the XOR of the bits s_i at the positions where $a_i = 1$.

Example: $\langle 110, 011 \rangle = (1 \cdot 0) \oplus (1 \cdot 1) \oplus (0 \cdot 1) = 0 \oplus 1 \oplus 0 = 1$.

Key fact, used in every proof today

Flip one bit s_i of s.

- If $a_i = 1$, then $\langle s, a \rangle$ flips: the XOR includes s_i.
- If $a_i = 0$, then $\langle s, a \rangle$ is unchanged: the XOR ignores s_i.

Warm-up: three bits from two random bits

Let the seed be two independent fair bits s_1, s_2. Output $s_1,\ s_2,\ s_1 \oplus s_2$.

seed	s_1	s_2	$s_1 \oplus s_2$
00	0	0	0
01	0	1	1
10	1	0	1
11	1	1	0

Warm-up: three bits from two random bits

Let the seed be two independent fair bits s_1, s_2. Output $s_1,\ s_2,\ s_1 \oplus s_2$.

seed	s_1	s_2	$s_1 \oplus s_2$
00	0	0	0
01	0	1	1
10	1	0	1
11	1	1	0

Each seed has probability $\frac{1}{4}$. Outputs s_1 and $s_1 \oplus s_2$ give the pairs

$$00,\ 01,\ 11,\ 10:$$

each pair once, so each with probability $\frac{1}{4}$. By the test, they are fair and independent.

Warm-up: three bits from two random bits

Let the seed be two independent fair bits s_1, s_2. Output $s_1,\ s_2,\ s_1 \oplus s_2$.

seed	s_1	s_2	$s_1 \oplus s_2$
00	0	0	0
01	0	1	1
10	1	0	1
11	1	1	0

Each seed has probability $\frac{1}{4}$. Outputs s_1 and $s_1 \oplus s_2$ give the pairs

$$00,\ 01,\ 11,\ 10:$$

each pair once, so each with probability $\frac{1}{4}$. By the test, they are fair and independent.

The other two pairs of columns check the same way: the three bits are pairwise independent.

Warm-up: three bits from two random bits

Let the seed be two independent fair bits s_1, s_2. Output $s_1,\ s_2,\ s_1 \oplus s_2$.

seed	s_1	s_2	$s_1 \oplus s_2$
00	0	0	0
01	0	1	1
10	1	0	1
11	1	1	0

Each seed has probability $\frac{1}{4}$. Outputs s_1 and $s_1 \oplus s_2$ give the pairs

$$00,\ 01,\ 11,\ 10:$$

each pair once, so each with probability $\frac{1}{4}$. By the test, they are fair and independent.

The other two pairs of columns check the same way: the three bits are pairwise independent.

They are *not* fully independent: the triple 111 never occurs, but three independent fair bits would give it with probability $\frac{1}{8}$.

The pattern: each output bit XORs a subset of the seed

Describe each output by a **label**: which seed bits it XORs.

output	s_1	s_2	$s_1 \oplus s_2$
label a	10	01	11
output $= \langle s, a \rangle$	$\langle s, 10 \rangle$	$\langle s, 01 \rangle$	$\langle s, 11 \rangle$

The pattern: each output bit XORs a subset of the seed

Describe each output by a **label**: which seed bits it XORs.

output	s_1	s_2	$s_1 \oplus s_2$
label a	10	01	11
output $= \langle s, a \rangle$	$\langle s, 10 \rangle$	$\langle s, 01 \rangle$	$\langle s, 11 \rangle$

The general construction. Choose a seed of L uniformly random bits, $s \in \{0,1\}^L$. Each nonzero label $a \in \{0,1\}^L$ gives one output bit $\langle s, a \rangle$. There are $2^L - 1$ nonzero labels.

The pattern: each output bit XORs a subset of the seed

Describe each output by a **label**: which seed bits it XORs.

output	s_1	s_2	$s_1 \oplus s_2$
label a	10	01	11
output $= \langle s, a \rangle$	$\langle s, 10 \rangle$	$\langle s, 01 \rangle$	$\langle s, 11 \rangle$

The general construction. Choose a seed of L uniformly random bits, $s \in \{0,1\}^L$. Each nonzero label $a \in \{0,1\}^L$ gives one output bit $\langle s, a \rangle$. There are $2^L - 1$ nonzero labels.

For the random cut we need one bit per vertex. Number the vertices $1, \ldots, n$ and give vertex v the label $a_v =$ (v written in binary), using

$$L = \lceil \log_2(n+1) \rceil \text{ bits}, \qquad B_v = \langle s, a_v \rangle.$$

These labels are distinct and nonzero.

The XOR generator

XOR generator

Let $s \in \{0,1\}^L$ be uniformly random, and let $a_1, \ldots, a_n \in \{0,1\}^L$ be distinct, nonzero labels. Then the bits $B_v = \langle s, a_v \rangle$, $v = 1, \ldots, n$, are fair and pairwise independent.

So the random-cut analysis goes through unchanged, using $\lceil \log_2(n+1) \rceil$ random bits instead of n.

Proof plan. All 2^L seeds are equally likely, so a probability is a fraction of seeds.

1. **One bit:** exactly half the seeds give $B_v = 1$.
2. **Two bits:** for $u \neq v$, each of the four values of (B_u, B_v) comes from exactly a quarter of the seeds. By the test, this proves the theorem.

Proof, step 1: one output bit is fair

Fix a nonzero label a. It has some position i with $a_i = 1$.

Proof, step 1: one output bit is fair

Fix a nonzero label a. It has some position i with $a_i = 1$.

Pair up the seeds: match each seed s with the seed s' that differs from it only in position i. Every seed is in exactly one pair.

Proof, step 1: one output bit is fair

Fix a nonzero label a. It has some position i with $a_i = 1$.

Pair up the seeds: match each seed s with the seed s' that differs from it only in position i. Every seed is in exactly one pair.

By the key fact, flipping s_i flips $\langle s, a \rangle$. So in every pair, one seed gives 0 and the other gives 1.

Proof, step 1: one output bit is fair

Fix a nonzero label a. It has some position i with $a_i = 1$.

Pair up the seeds: match each seed s with the seed s' that differs from it only in position i. Every seed is in exactly one pair.

By the key fact, flipping s_i flips $\langle s, a \rangle$. So in every pair, one seed gives 0 and the other gives 1.

Example: $a = 011$, flip position $i = 3$

s	$\langle s, a \rangle$	s'	$\langle s', a \rangle$
000	0	001	1
010	1	011	0
100	0	101	1
110	1	111	0

So exactly half the seeds give 1: $\Pr(\langle s, a \rangle = 1) = \frac{1}{2}$.

Proof, step 2: two output bits are uniform on four pairs

Take distinct nonzero labels $a \neq b$. Name them so that at some position i, $a_i = 1$ and $b_i = 0$. Since $b \neq 0$, $b_j = 1$ at some position $j \neq i$.

Proof, step 2: two output bits are uniform on four pairs

Take distinct nonzero labels $a \neq b$. Name them so that at some position i, $a_i = 1$ and $b_i = 0$. Since $b \neq 0$, $b_j = 1$ at some position $j \neq i$.

Group the seeds in fours: seeds that agree everywhere except positions i and j. By the key fact:

flip	changes $\langle \cdot, a \rangle$?	changes $\langle \cdot, b \rangle$?
s_i	yes ($a_i = 1$)	no ($b_i = 0$)
s_j	only if $a_j = 1$	yes ($b_j = 1$)

Proof, step 2: two output bits are uniform on four pairs

Take distinct nonzero labels $a \neq b$. Name them so that at some position i, $a_i = 1$ and $b_i = 0$. Since $b \neq 0$, $b_j = 1$ at some position $j \neq i$.

Group the seeds in fours: seeds that agree everywhere except positions i and j. By the key fact:

flip	changes $\langle \cdot, a \rangle$?	changes $\langle \cdot, b \rangle$?
s_i	yes ($a_i = 1$)	no ($b_i = 0$)
s_j	only if $a_j = 1$	yes ($b_j = 1$)

Start from one seed in the group, with outputs (x, y).

- Flipping s_i gives $(1-x, y)$.
- The other two seeds have s_j flipped, so their second output is $1-y$. They differ from each other in s_i, so their first outputs differ.

Proof, step 2: two output bits are uniform on four pairs

Take distinct nonzero labels $a \neq b$. Name them so that at some position i, $a_i = 1$ and $b_i = 0$. Since $b \neq 0$, $b_j = 1$ at some position $j \neq i$.

Group the seeds in fours: seeds that agree everywhere except positions i and j. By the key fact:

flip	changes $\langle \cdot, a \rangle$?	changes $\langle \cdot, b \rangle$?
s_i	yes ($a_i = 1$)	no ($b_i = 0$)
s_j	only if $a_j = 1$	yes ($b_j = 1$)

Start from one seed in the group, with outputs (x, y).

- Flipping s_i gives $(1-x, y)$.
- The other two seeds have s_j flipped, so their second output is $1-y$. They differ from each other in s_i, so their first outputs differ.

So each group produces all four pairs, once each, and each pair comes from exactly $\frac{1}{4}$ of the seeds. □

Derandomized MAX-CUT: try every seed

Seed MAX-CUT

Input: graph G with vertices $1, \ldots, n$ and m edges.

Let $L = \lceil \log_2(n+1) \rceil$ and $a_v =$ binary of v.

For every seed $s \in \{0,1\}^L$:

- compute $B_v(s) = \langle s, a_v \rangle$ for every vertex v;
- count the cut edges:
  $C(s) = \#\{\{u,v\} \in E : B_u(s) \neq B_v(s)\}$.

Output: the cut of a seed with the largest $C(s)$.

No random bits are used: the algorithm tries *every* seed, all $2^L \leq 2n+1$ of them.

Question

Why must the best seed cut at least $m/2$ edges?

Why the best seed cuts at least $m/2$ edges

Step 1: the average over all seeds is $m/2$. Swap the order of summation:

$$\begin{aligned}\frac{1}{2^L}\sum_s C(s) &= \frac{1}{2^L}\sum_s \sum_{\{u,v\}\in E} \mathbf{1}\{B_u(s) \neq B_v(s)\} \\ &= \sum_{\{u,v\}\in E} \frac{\#\{s : B_u(s) \neq B_v(s)\}}{2^L} \\ &= \sum_{\{u,v\}\in E} \Pr_s(B_u \neq B_v).\end{aligned}$$

Why the best seed cuts at least $m/2$ edges

Step 1: the average over all seeds is $m/2$. Swap the order of summation:

$$\begin{aligned}\frac{1}{2^L}\sum_s C(s) &= \frac{1}{2^L}\sum_s \sum_{\{u,v\}\in E} \mathbf{1}\{B_u(s) \neq B_v(s)\} \\ &= \sum_{\{u,v\}\in E} \frac{\#\{s : B_u(s) \neq B_v(s)\}}{2^L} \\ &= \sum_{\{u,v\}\in E} \Pr_s(B_u \neq B_v).\end{aligned}$$

By the XOR generator, each pair (B_u, B_v) is uniform on four values, so $\Pr_s(B_u \neq B_v) = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$, and the average is $m/2$.

Why the best seed cuts at least $m/2$ edges

Step 1: the average over all seeds is $m/2$. Swap the order of summation:

$$\begin{aligned}\frac{1}{2^L}\sum_s C(s) &= \frac{1}{2^L}\sum_s \sum_{\{u,v\}\in E} \mathbf{1}\{B_u(s) \neq B_v(s)\} \\ &= \sum_{\{u,v\}\in E} \frac{\#\{s : B_u(s) \neq B_v(s)\}}{2^L} \\ &= \sum_{\{u,v\}\in E} \Pr_s(B_u \neq B_v).\end{aligned}$$

By the XOR generator, each pair (B_u, B_v) is uniform on four values, so $\Pr_s(B_u \neq B_v) = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$, and the average is $m/2$.

Step 2: some seed is at least the average. If every seed had $C(s) < m/2$, their average would also be below $m/2$. So $\max_s C(s) \geq m/2$.

The price: more total work

Conditional expectation (last time) places the vertices in order; each vertex goes opposite most of its already-placed neighbors.

- Vertex v looks at its $\deg(v)$ neighbors: $O(\deg v)$ work.
- Total: $\sum_v O(1 + \deg v) = O(n + m)$.

The price: more total work

Conditional expectation (last time) places the vertices in order; each vertex goes opposite most of its already-placed neighbors.

- Vertex v looks at its $\deg(v)$ neighbors: $O(\deg v)$ work.
- Total: $\sum_v O(1 + \deg v) = O(n + m)$.

Seed MAX-CUT does a full run for every seed.

- Computing one bit $B_v(s)$ XORs up to L seed bits: $O(L) = O(\log n)$.
- One seed: n bits plus checking m edges, $O(n \log n + m)$.
- All $2^L \leq 2n + 1$ seeds: $O\big(n(n \log n + m)\big)$.

The price: more total work

Conditional expectation (last time) places the vertices in order; each vertex goes opposite most of its already-placed neighbors.

- Vertex v looks at its $\deg(v)$ neighbors: $O(\deg v)$ work.
- Total: $\sum_v O(1 + \deg v) = O(n + m)$.

Seed MAX-CUT does a full run for every seed.

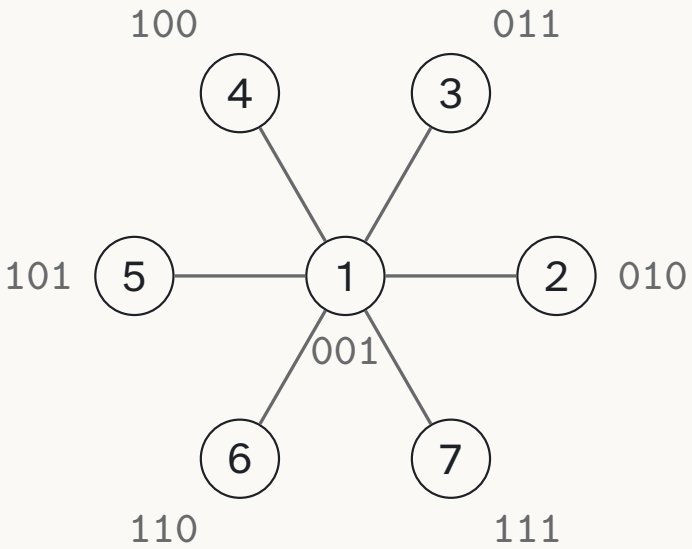
- Computing one bit $B_v(s)$ XORs up to L seed bits: $O(L) = O(\log n)$.
- One seed: n bits plus checking m edges, $O(n \log n + m)$.
- All $2^L \le 2n + 1$ seeds: $O\big(n(n \log n + m)\big)$.

Both guarantee $m/2$ edges. The seed method does roughly n times more work, but its runs are independent of each other; conditional expectation's choices are not. Part 2 cashes this in.

Could a seed find a *maximum* cut?

Seed MAX-CUT matches conditional expectation's guarantee. But it tries $2n + 1$ cuts, not one. Could the best of them be optimal?

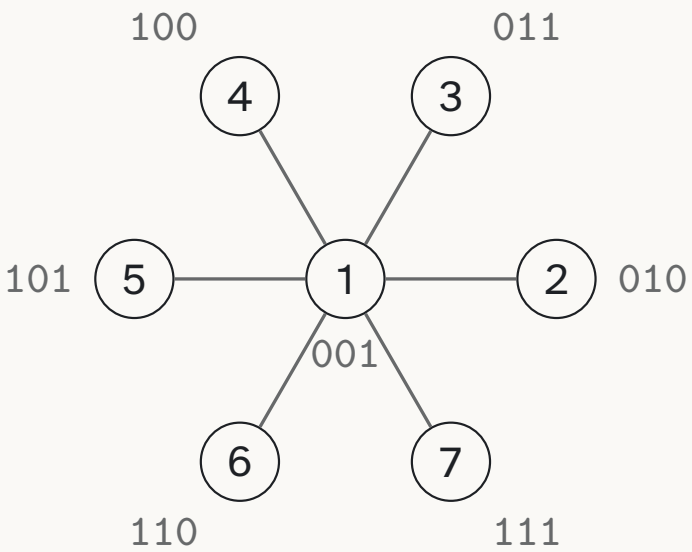
Test case: the star with center 1 and leaves $2, \ldots, 7$. Each vertex carries its label (its number in binary, $L = 3$).

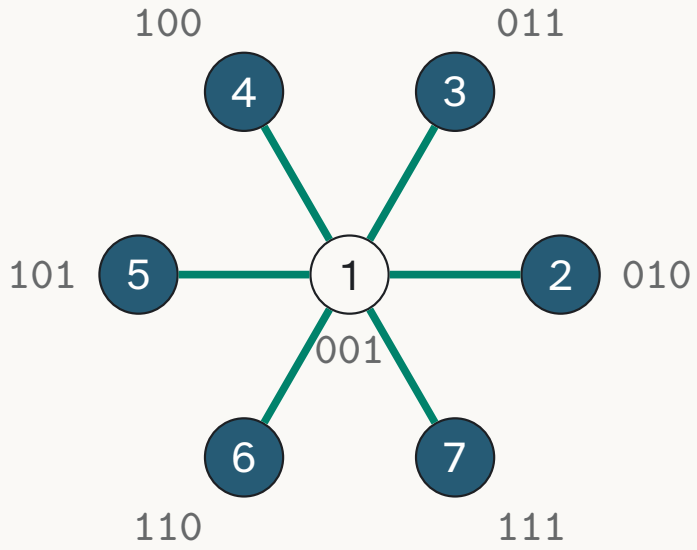


Could a seed find a *maximum* cut?

Seed MAX-CUT matches conditional expectation's guarantee. But it tries $2n + 1$ cuts, not one. Could the best of them be optimal?

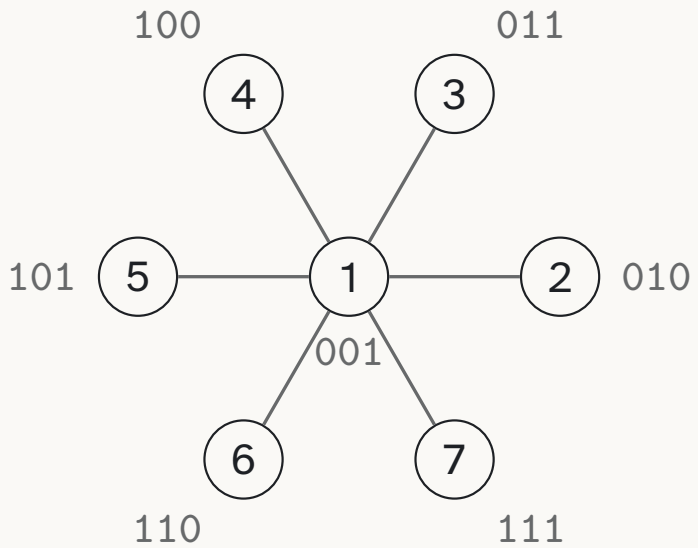
Test case: the star with center 1 and leaves $2, \ldots, 7$. Each vertex carries its label (its number in binary, $L = 3$).





The maximum cut: center alone, all 6 edges cut.

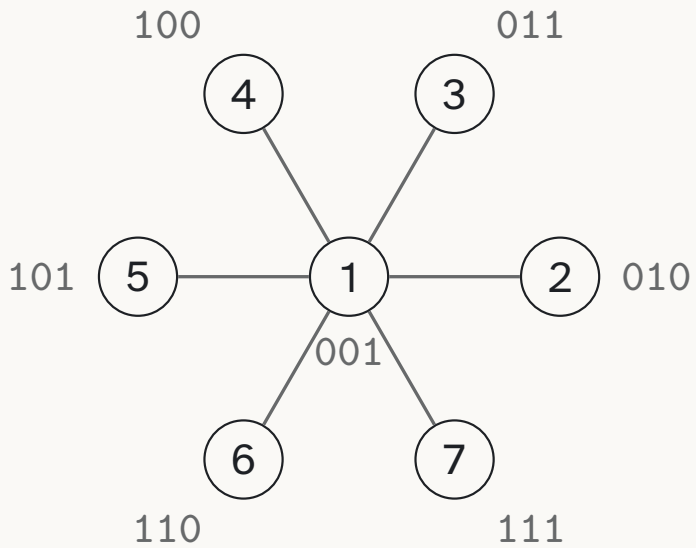
One seed on the star: $s = 010$



$s = 010$ picks out the middle bit of each label, so $B_v =$ middle bit of a_v.

v	a_v	B_v
1	001	
2	010	
3	011	
4	100	
5	101	
6	110	
7	111	

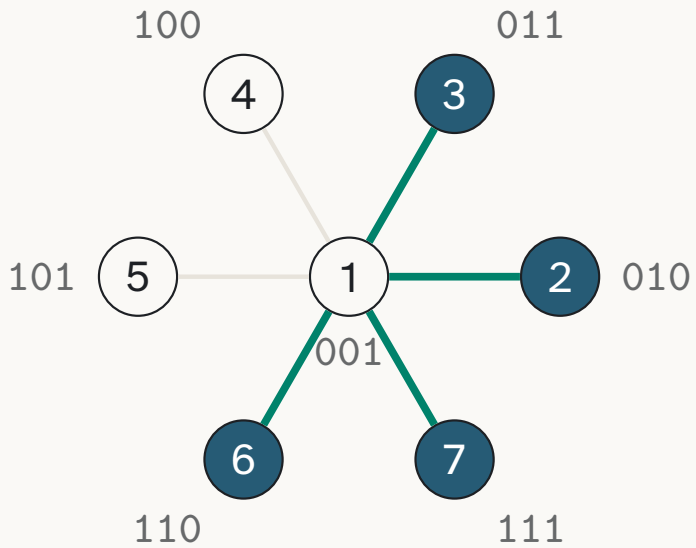
One seed on the star: $s = 010$



$s = 010$ picks out the middle bit of each label, so $B_v =$ middle bit of a_v.

v	a_v	B_v
1	001	0
2	010	1
3	011	1
4	100	0
5	101	0
6	110	1
7	111	1

One seed on the star: $s = 010$



$s = 010$ picks out the middle bit of each label, so $B_v =$ middle bit of a_v.

v	a_v	B_v
1	001	0
2	010	1
3	011	1
4	100	0
5	101	0
6	110	1
7	111	1

The center has bit 0; leaves 2, 3, 6, 7 have bit 1. 4 **edges cut.**

Every seed on the star

seed s	000	001	010	011	100	101	110	111
edges cut	0	3	4	3	4	3	4	3

Average $24/8 = 3 = m/2$, as promised. Best: 4. Optimum: 6.

▸ All bits: C.1

Every seed on the star

seed s	000	001	010	011	100	101	110	111
edges cut	0	3	4	3	4	3	4	3

Average $24/8 = 3 = m/2$, as promised. Best: 4. Optimum: 6.

▸ All bits: C.1

Why no seed reaches 6. For a nonzero s, step 1 of the proof (with the roles of s and a swapped) shows exactly half of the 8 labels give $\langle s, a \rangle = 1$. Label 000 gives 0, so exactly 4 of our 7 labels give 1.

- Center bit 0: the 4 ones are all leaves, so 4 edges cut.
- Center bit 1: 3 leaves are 1 and 3 are 0, so 3 edges cut.

Every seed on the star

seed s	000	001	010	011	100	101	110	111
edges cut	0	3	4	3	4	3	4	3

Average $24/8 = 3 = m/2$, as promised. Best: 4. Optimum: 6.

▸ All bits: C.1

Why no seed reaches 6. For a nonzero s, step 1 of the proof (with the roles of s and a swapped) shows exactly half of the 8 labels give $\langle s, a \rangle = 1$. Label 000 gives 0, so exactly 4 of our 7 labels give 1.

- Center bit 0: the 4 ones are all leaves, so 4 edges cut.
- Center bit 1: 3 leaves are 1 and 3 are 0, so 3 edges cut.

The seeds preserve the *average* the analysis needed, not the *best* possible cut.

What we did, in general

	In general	For MAX-CUT
1.	Find what the analysis uses about the random bits	each pair uniform on four values
2.	Generate bits with that property from a short seed	XOR generator, $\lceil \log_2(n+1) \rceil$ bits
3.	Run the algorithm on every seed and measure each run's quality	cut size $C(s)$, for $2n+1$ seeds
4.	Return the best run; it is at least the average	$\geq m/2$ edges

This takes polynomial time when

- the seed has $O(\log n)$ bits, so there are polynomially many seeds, and
- we can compute each run's quality, so we can tell which seed is best.

PART 2

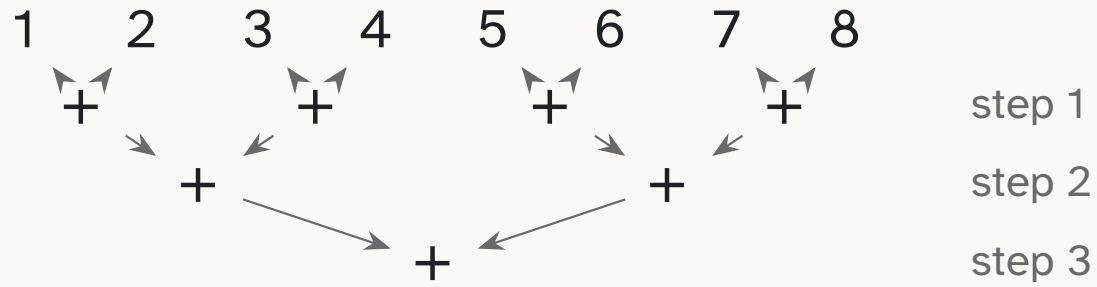
Short seeds give deterministic parallel algorithms

2

Measuring a parallel algorithm

Model: many processors share one memory and take steps in lock-step.

- **Parallel time** (depth): the number of steps, where in each step every processor does $O(1)$ work at once.
- **Work**: the total number of operations over all processors.

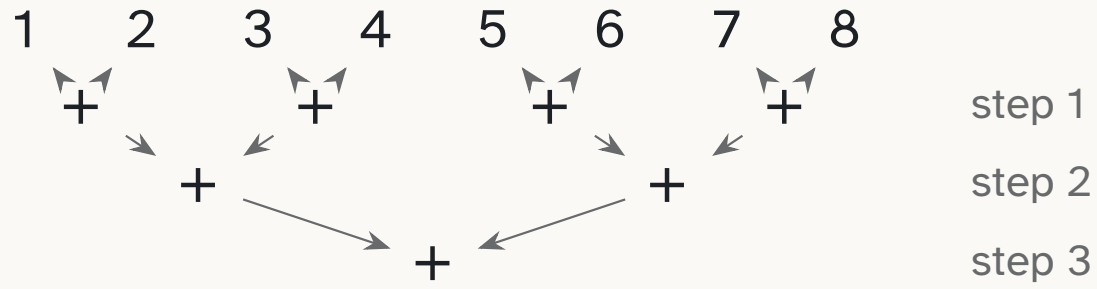


Summing 8 numbers: work 7, parallel time 3. Summing, ANDing, or taking the max of M values takes parallel time $O(\log M)$.

Measuring a parallel algorithm

Model: many processors share one memory and take steps in lock-step.

- **Parallel time** (depth): the number of steps, where in each step every processor does $O(1)$ work at once.
- **Work**: the total number of operations over all processors.

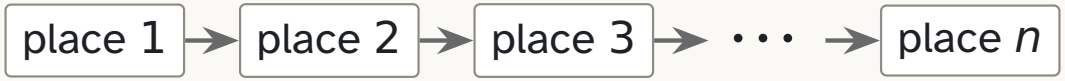


Summing 8 numbers: work 7, parallel time 3. Summing, ANDing, or taking the max of M values takes parallel time $O(\log M)$.

Processors: we allow $\mathrm{poly}(n)$ processors, where n is the input size. This is the standard theoretical model (the class NC asks for $\mathrm{polylog}(n)$ time with $\mathrm{poly}(n)$ processors). It asks how fast a problem can be solved in principle, ignoring hardware limits.

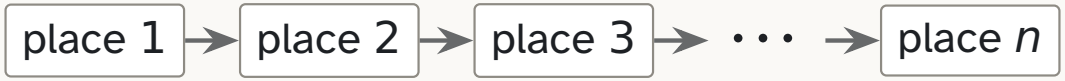
Seed MAX-CUT runs in parallel

Conditional expectation: where vertex v goes depends on where the earlier vertices went.



Seed MAX-CUT runs in parallel

Conditional expectation: where vertex v goes depends on where the earlier vertices went.



Seed MAX-CUT: each seed's run is independent of the others, so give each seed its own processors.

- each bit $B_v(s)$: XOR of $\leq L = O(\log n)$ bits, time $O(\log \log n)$;
- each edge: compare two bits, time $O(1)$;
- count each seed's cut edges: time $O(\log m) = O(\log n)$;
- best of $2n + 1$ seeds: time $O(\log n)$.

Total: parallel time $O(\log n)$, with $O(n(n \log n + m))$ processors.

A harder problem: maximal independent set

An **independent set** has no two adjacent vertices. It is **maximal** if adding any other vertex would break this.



{2} is maximal

Greedy MIS

Input: graph G with vertices $1, \ldots, n$. Start with $I = \emptyset$.
For $v = 1, \ldots, n$ in order: if no neighbor of v is in I, add v to I.
Output: I, a maximal independent set.

Whether v joins depends on the decisions for $1, \ldots, v-1$: a chain of n steps.

Question

For graphs of constant maximum degree Δ: is there a *deterministic* algorithm with parallel time $\mathrm{polylog}(n)$?

A randomized parallel algorithm

Random-marking MIS

Input: graph G, maximum degree $\leq \Delta$ (a power of two). Start with $I = \emptyset$, all vertices *active*. Repeat until no vertex is active:

1. **Mark** each active vertex with probability $p = \frac{1}{2\Delta}$.
2. **Accept** each marked vertex with no marked active neighbor. Call this set T.
3. **Update:** add T to I; make T and all neighbors of T inactive.

Output: I.

Each vertex's steps use only its own mark and its neighbors' marks, so all vertices do each step at the same time.

Why the output is a maximal independent set

I is independent.

- Two vertices accepted in the same round are not adjacent: each would have seen the other's mark and been rejected.
- After a round, every neighbor of T is inactive, so no later round can add a neighbor of a vertex already in I.

Why the output is a maximal independent set

I is independent.

- Two vertices accepted in the same round are not adjacent: each would have seen the other's mark and been rejected.
- After a round, every neighbor of T is inactive, so no later round can add a neighbor of a vertex already in I.

I is maximal. A vertex becomes inactive only if it is in T (so in I) or is a neighbor of T (so it has a neighbor in I). At the end every vertex is inactive, so no vertex can be added.

Why the output is a maximal independent set

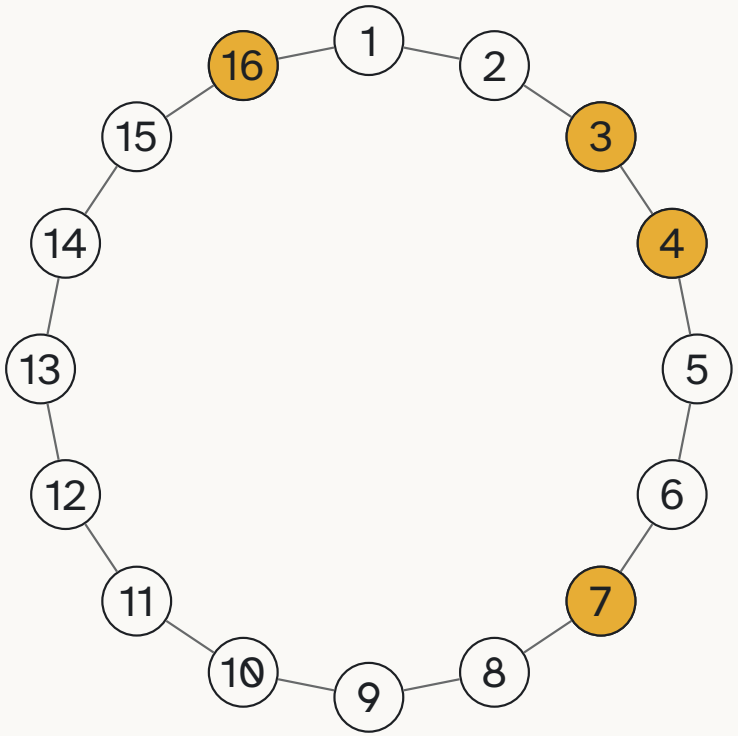
I is independent.

- Two vertices accepted in the same round are not adjacent: each would have seen the other's mark and been rejected.
- After a round, every neighbor of T is inactive, so no later round can add a neighbor of a vertex already in I.

I is maximal. A vertex becomes inactive only if it is in T (so in I) or is a neighbor of T (so it has a neighbor in I). At the end every vertex is inactive, so no vertex can be added.

Still to show: the algorithm finishes in few rounds. That depends on how much one round accomplishes.

One round on the 16-cycle

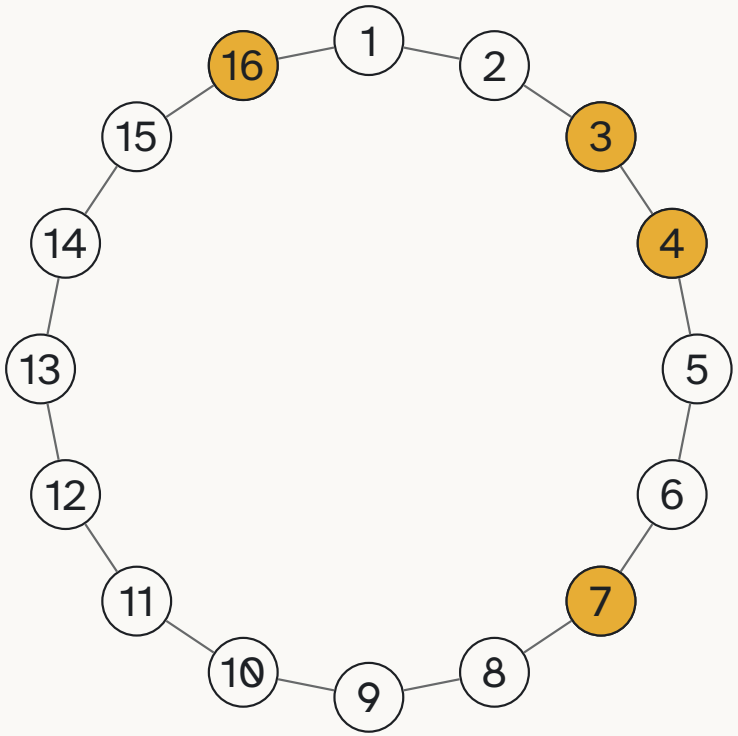


Here $\Delta = 2$, so each vertex marks itself with probability $\frac{1}{4}$.

Suppose the gold vertices come up marked:

$$M = \{3, 4, 7, 16\}.$$

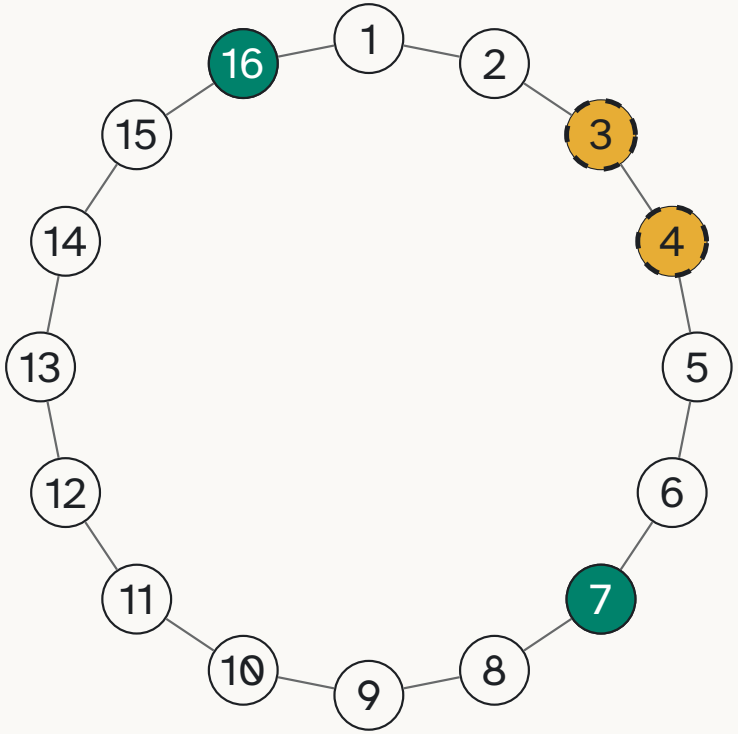
One round on the 16-cycle



Question

Which marked vertices have no marked neighbor?

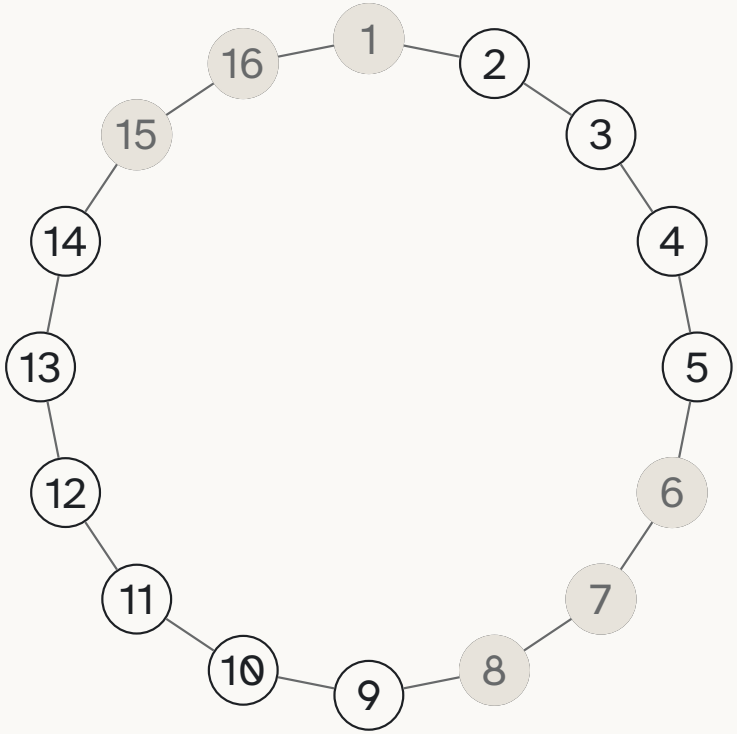
One round on the 16-cycle



3 and 4 are adjacent and both marked, so *both* are rejected.

7 and 16 have no marked neighbors. Accepted: $T = \{7, 16\}$.

One round on the 16-cycle



Add $7, 16$ to I; make them and their neighbors $6, 8, 15, 1$ inactive.

Ten vertices stay active, on two paths. The next round starts here.

How much does one round accomplish?

Let $M_v = 1$ if v is marked. Vertex v is accepted exactly when $M_v = 1$ and no neighbor is marked. So

$$\Pr(v \in T) = \Pr(M_v = 1) - \Pr(M_v = 1, \text{ a neighbor marked})$$

How much does one round accomplish?

Let $M_v = 1$ if v is marked. Vertex v is accepted exactly when $M_v = 1$ and no neighbor is marked. So

$$\begin{aligned}\Pr(v \in T) &= \Pr(M_v = 1) - \Pr(M_v = 1,\ \text{a neighbor marked}) \\ &\geq p - \textstyle\sum_{u \in N(v)} \Pr(M_v = 1,\ M_u = 1) \quad \text{(union bound)}\end{aligned}$$

How much does one round accomplish?

Let $M_v = 1$ if v is marked. Vertex v is accepted exactly when $M_v = 1$ and no neighbor is marked. So

$$\begin{aligned}
\Pr(v \in T) &= \Pr(M_v = 1) - \Pr(M_v = 1,\ \text{a neighbor marked}) \\
&\geq p - \textstyle\sum_{u \in N(v)} \Pr(M_v = 1,\ M_u = 1) \quad \text{(union bound)} \\
&= p - \deg(v)\, p^2 \quad \text{(pairs of marks are independent)} \\
&\geq p - \Delta p^2.
\end{aligned}$$

The middle step is the only use of independence, and it needs only *pairs* of marks.

How much does one round accomplish?

Let $M_v = 1$ if v is marked. Vertex v is accepted exactly when $M_v = 1$ and no neighbor is marked. So

$$\begin{aligned}
\Pr(v \in T) &= \Pr(M_v = 1) - \Pr(M_v = 1, \text{ a neighbor marked}) \\
&\geq p - \textstyle\sum_{u \in N(v)} \Pr(M_v = 1,\ M_u = 1) \quad \text{(union bound)} \\
&= p - \deg(v)\, p^2 \quad \text{(pairs of marks are independent)} \\
&\geq p - \Delta p^2.
\end{aligned}$$

The middle step is the only use of independence, and it needs only *pairs* of marks.

With $p = \frac{1}{2\Delta}$, $\Pr(v \in T) \geq \frac{1}{2\Delta} - \frac{1}{4\Delta} = \frac{1}{4\Delta}$. Summing over the n_j active vertices: $\mathbb{E}\,|T| \geq n_j/(4\Delta)$.

Marks with probability $\frac{1}{2\Delta}$ from the XOR generator

Recall: seed $s \in \{0,1\}^L$, label a_v = binary of v, bit $B_v = \langle s, a_v \rangle$. These bits are fair and pairwise independent.

Marks: use $\ell = \log_2(2\Delta)$ independent seeds $s^{(1)}, \ldots, s^{(\ell)}$, giving each vertex ℓ bits. Mark v when all ℓ of its bits are 0.

Example: 16-cycle, $\Delta = 2$, $\ell = 2$, $L = 5$, seeds $s^{(1)} = 00011$, $s^{(2)} = 01000$.

vertex	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
bit from $s^{(1)}$	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
bit from $s^{(2)}$	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0

Gold columns (all bits 0) are marked: $\{3, 4, 7, 16\}$, the marking from the previous example.

$\Pr(M_v = 1) = (\frac{1}{2})^\ell = \frac{1}{2\Delta}$. This is why Δ is a power of two.

The marks are pairwise independent

For $u \neq v$: both are marked when all 2ℓ of their bits are 0. The seeds are independent, so

$$\Pr(M_u = 1, M_v = 1) = \prod_{k=1}^{\ell} \Pr\big(B_u^{(k)} = 0,\ B_v^{(k)} = 0\big).$$

The marks are pairwise independent

For $u \neq v$: both are marked when all 2ℓ of their bits are 0. The seeds are independent, so

$$\Pr(M_u = 1, M_v = 1) = \prod_{k=1}^{\ell} \Pr\big(B_u^{(k)} = 0,\ B_v^{(k)} = 0\big).$$

By the XOR generator, each pair $(B_u^{(k)}, B_v^{(k)})$ is uniform on four values, so each factor is $\frac{1}{4}$:

$$\Pr(M_u = 1, M_v = 1) = \Big(\frac{1}{4}\Big)^{\ell} = \Big(\frac{1}{2}\Big)^{\ell}\Big(\frac{1}{2}\Big)^{\ell} = \Pr(M_u{=}1)\,\Pr(M_v{=}1).$$

For 0/1 variables this one equation gives independence; the other three cases follow by subtraction.

The marks are pairwise independent

For $u \neq v$: both are marked when all 2ℓ of their bits are 0. The seeds are independent, so

$$\Pr(M_u = 1, M_v = 1) = \prod_{k=1}^{\ell} \Pr\big(B_u^{(k)} = 0,\ B_v^{(k)} = 0\big).$$

By the XOR generator, each pair $(B_u^{(k)}, B_v^{(k)})$ is uniform on four values, so each factor is $\frac{1}{4}$:

$$\Pr(M_u = 1, M_v = 1) = \Big(\frac{1}{4}\Big)^{\ell} = \Big(\frac{1}{2}\Big)^{\ell}\Big(\frac{1}{2}\Big)^{\ell} = \Pr(M_u{=}1)\Pr(M_v{=}1).$$

For 0/1 variables this one equation gives independence; the other three cases follow by subtraction.

Seed length: $\ell L = \log_2(2\Delta) \cdot \lceil \log_2(n+1) \rceil = O(\log n)$ bits for constant Δ: polynomially many seeds.

Derandomized MIS: search the seeds every round

Seed MIS

Input: graph G, constant maximum degree $\leq \Delta$ (a power of two).
Start with $I = \emptyset$, all vertices active. Repeat until no vertex is active:

1. For *every* seed $(s^{(1)}, \ldots, s^{(\ell)})$, in parallel: compute its marks and its accepted set T.
2. Pick a seed with the largest $|T|$.
3. Add that T to I; make T and its neighbors inactive.

Output: I.

Derandomized MIS: search the seeds every round

Seed MIS

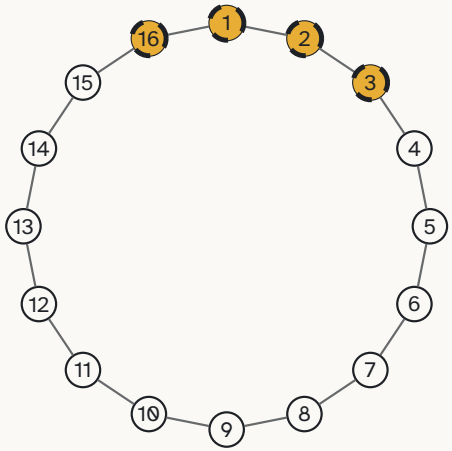
Input: graph G, constant maximum degree $\leq \Delta$ (a power of two).
Start with $I = \emptyset$, all vertices active. Repeat until no vertex is active:

1. For *every* seed $(s^{(1)}, \ldots, s^{(\ell)})$, in parallel: compute its marks and its accepted set T.
2. Pick a seed with the largest $|T|$.
3. Add that T to I; make T and its neighbors inactive.

Output: I.

Why search one round at a time? The active graph in round 2 depends on which seed won round 1. Searching whole runs at once would mean trying every *sequence* of seeds:
$(\text{poly}(n))^{\#\text{rounds}} = n^{O(\log n)}$ candidates. Too many.

Three seeds on the 16-cycle



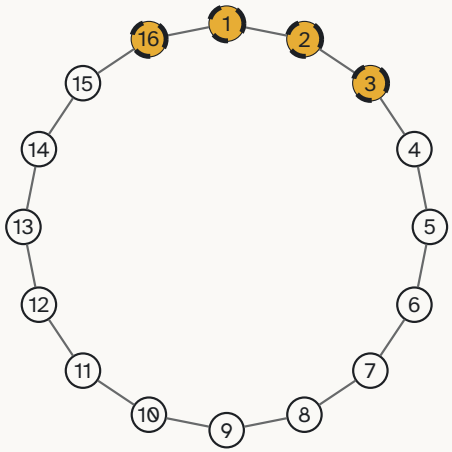
00100, 01000

marks $\{16, 1, 2, 3\}$

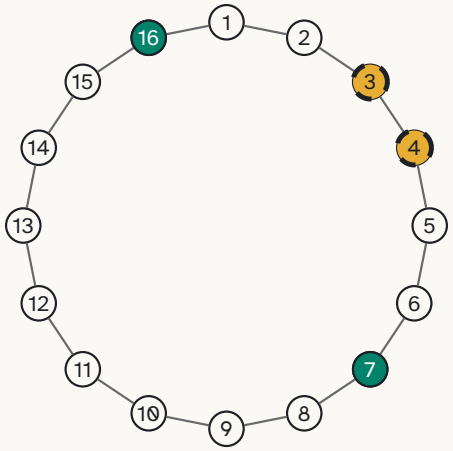
$|T| = 0$

Marks on four consecutive vertices: everyone sees a marked neighbor, nothing is accepted.

Three seeds on the 16-cycle



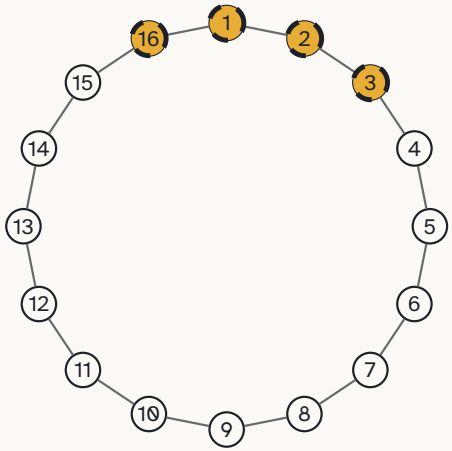
00100, 01000
marks $\{16, 1, 2, 3\}$
$|T| = 0$



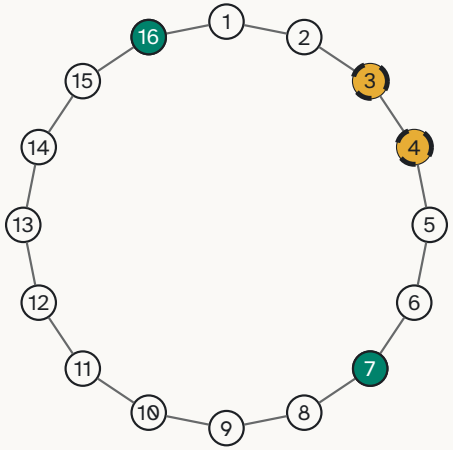
00011, 01000
marks $\{3, 4, 7, 16\}$
$|T| = 2$

The round we traced: one conflict, two accepted.

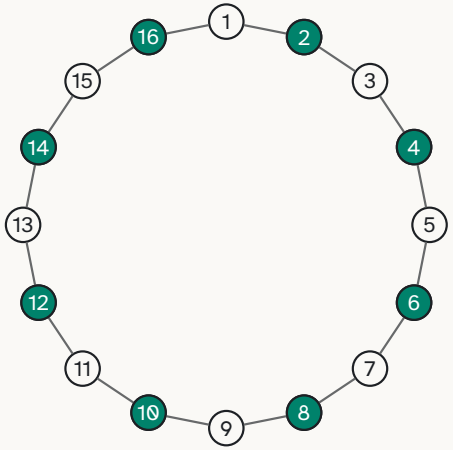
Three seeds on the 16-cycle



00100, 01000
marks $\{16, 1, 2, 3\}$
$|T| = 0$



00011, 01000
marks $\{3, 4, 7, 16\}$
$|T| = 2$



00000, 00001
marks: even vertices
$|T| = 8$

The first seed bit is always 0; the second is the last bit of the label. So exactly the even vertices are marked, no two adjacent: all 8 accepted, and they cover the cycle. One round finishes.

Predict, then look: all 1,024 seeds

Two 5-bit seeds: $2^{10} = 1{,}024$ seeds. The analysis promises an average of at least $n/(4\Delta) = 16/8 = 2$.

Question

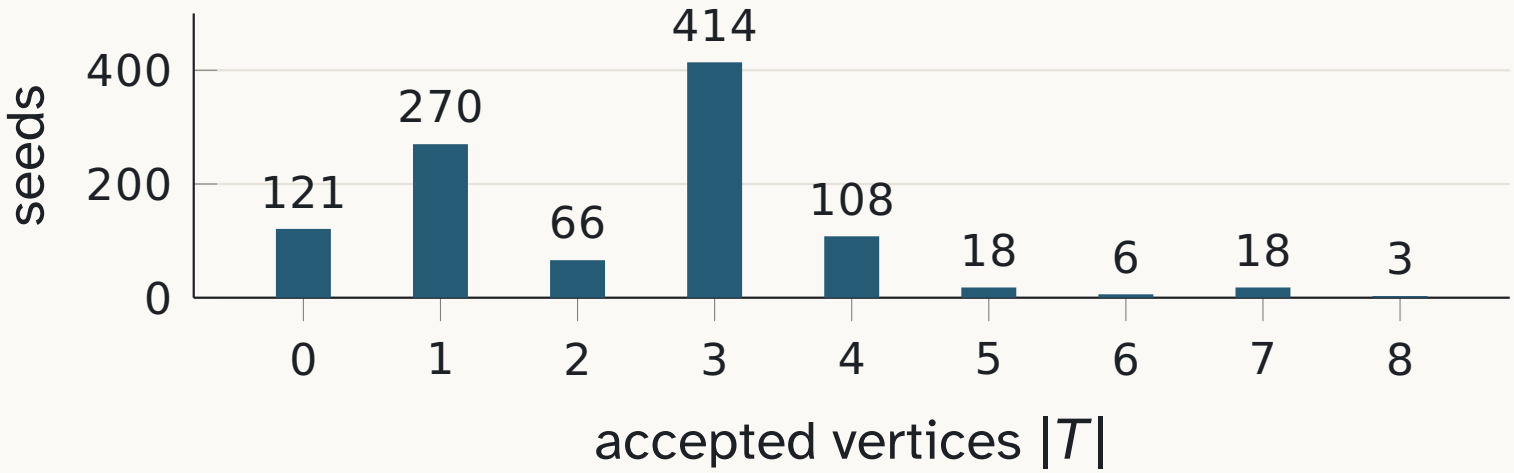
Guess the actual average of $|T|$ over all seeds, and the best score.

Predict, then look: all 1,024 seeds

Two 5-bit seeds: $2^{10} = 1{,}024$ seeds. The analysis promises an average of at least $n/(4\Delta) = 16/8 = 2$.

Question

Guess the actual average of $|T|$ over all seeds, and the best score.



Average $2352/1024 \approx 2.3$; best 8, reached by 3 seeds. We only need one good seed.

▸ Counts: C.2

Seed MIS finishes in $O(\log n)$ rounds

Each round removes a fixed fraction. The progress bound only used that every degree is at most Δ, which stays true for the active part of the graph. So in round j, with n_j active vertices, the average over seeds is at least $n_j/(4\Delta)$, and the best seed does at least that well.

Accepted vertices become inactive, so

$$n_{j+1} \;\le\; n_j - \frac{n_j}{4\Delta} = \Big(1 - \frac{1}{4\Delta}\Big) n_j.$$

Seed MIS finishes in $O(\log n)$ rounds

Each round removes a fixed fraction. The progress bound only used that every degree is at most Δ, which stays true for the active part of the graph. So in round j, with n_j active vertices, the average over seeds is at least $n_j/(4\Delta)$, and the best seed does at least that well.

Accepted vertices become inactive, so

$$n_{j+1} \;\le\; n_j - \frac{n_j}{4\Delta} = \Big(1 - \frac{1}{4\Delta}\Big) n_j.$$

Iterate. After r rounds, using $1 - x \le e^{-x}$,

$$n_r \;\le\; n\Big(1 - \frac{1}{4\Delta}\Big)^r \;\le\; n\, e^{-r/(4\Delta)}.$$

When $r > 4\Delta \ln n$, this is below 1, so no vertex is active. With Δ constant: $O(\log n)$ rounds.

We computed each winning $|T|$, so this bound is guaranteed, not just likely.

Seed MIS runs in $O(\log^2 n)$ parallel time

One round, with each seed handled by its own processors:

- each of a vertex's ℓ bits: XOR of $\leq L$ seed bits, time $O(\log \log n)$;
- mark (AND of $\ell = O(1)$ bits) and accept (check $\leq \Delta$ neighbors): time $O(1)$;
- count $|T|$ for each seed, then take the best of $\mathrm{poly}(n)$ seeds: time $O(\log n)$.

So one round takes time $O(\log n)$ with $\mathrm{poly}(n)$ processors.

Seed MIS runs in $O(\log^2 n)$ parallel time

One round, with each seed handled by its own processors:

- each of a vertex's ℓ bits: XOR of $\leq L$ seed bits, time $O(\log \log n)$;
- mark (AND of $\ell = O(1)$ bits) and accept (check $\leq \Delta$ neighbors): time $O(1)$;
- count $|T|$ for each seed, then take the best of $\mathrm{poly}(n)$ seeds: time $O(\log n)$.

So one round takes time $O(\log n)$ with $\mathrm{poly}(n)$ processors.

$$O(\log n) \text{ rounds} \times O(\log n) \text{ per round} = O(\log^2 n) \text{ parallel time}$$

For constant-degree graphs, this answers the question, with no randomness. Luby (1986) extends the idea to all graphs.

PART 3

Beyond expectations: high-probability guarantees from limited independence

3

A different goal for Part 3

In Parts 1 and 2:

- the analysis needed only an *expectation* ($\mathbb{E}C = m/2$, $\mathbb{E}|T| \geq n_j/4\Delta$), and
- we could measure each run's quality, so we tried every seed and removed the randomness.

A different goal for Part 3

In Parts 1 and 2:

- the analysis needed only an *expectation* ($\mathbb{E}C = m/2$, $\mathbb{E}|T| \geq n_j/4\Delta$), and
- we could measure each run's quality, so we tried every seed and removed the randomness.

In Part 3, the algorithm stays randomized. Its guarantee is a *concentration bound*:

$$\Pr(\text{error} \geq \varepsilon) \leq \delta.$$

A different goal for Part 3

In Parts 1 and 2:

- the analysis needed only an *expectation* ($\mathbb{E}C = m/2$, $\mathbb{E}|T| \geq n_j/4\Delta$), and
- we could measure each run's quality, so we tried every seed and removed the randomness.

In Part 3, the algorithm stays randomized. Its guarantee is a *concentration bound*:

$$\Pr(\text{error} \geq \varepsilon) \leq \delta.$$

Question: how much independence does a concentration bound need, and how many random bits?

Answer (preview): pairwise independence gives Chebyshev's bound. $O(\log \frac{1}{\delta})$-wise independence gives a bound as good as Hoeffding's, up to constants. Both use a seed of a few random numbers.

Example: estimating a classifier's error

A fixed classifier, and N labelled test records numbered $0, \ldots, N-1$. Let

$$f(i) = \mathbf{1}\{\text{wrong on record } i\}, \qquad \mu = \frac{1}{N}\sum_{i=0}^{N-1} f(i) \ \text{(the error rate)}.$$

Sample mean

Input: access to f; sample size t.
Choose random indices $J_0, \ldots, J_{t-1} \in \{0, \ldots, N-1\}$.
Output: $\widehat{\mu} = \frac{1}{t}\sum_{j=0}^{t-1} f(J_j)$.

Goal: choose t so that $\Pr(|\widehat{\mu} - \mu| \geq \varepsilon) \leq \delta$.

Fully independent indices use $t \log_2 N$ random bits.

Why not try every seed here?

In Part 1, a run's quality was its cut size, which we could compute. Here a run's quality is its error $|\widehat{\mu} - \mu|$, and μ is the unknown we are estimating. We can't tell which seed is accurate.

Why not try every seed here?

In Part 1, a run's quality was its cut size, which we could compute. Here a run's quality is its error $|\widehat{\mu} - \mu|$, and μ is the unknown we are estimating. We can't tell which seed is accurate.

Averaging over all seeds instead would be exact, but it would read far more records than just reading all N once.

Why not try every seed here?

In Part 1, a run's quality was its cut size, which we could compute. Here a run's quality is its error $|\widehat{\mu} - \mu|$, and μ is the unknown we are estimating. We can't tell which seed is accurate.

Averaging over all seeds instead would be exact, but it would read far more records than just reading all N once.

So the algorithm stays randomized. Using fewer random bits is still useful:

- a seed of two numbers can be logged, and the exact sample replayed later;
- workers that share a short seed all agree on the same sample.

Pairwise-independent indices give Chebyshev's bound

Write $X_j = f(J_j) \in \{0, 1\}$. If each J_j is uniform, then $\mathbb{E}X_j = \mu$ and $\mathbb{E}\widehat{\mu} = \mu$. For the variance of a sum,

$$\operatorname{Var}(\widehat{\mu}) = \frac{1}{t^2}\Big(\sum_j \operatorname{Var}(X_j) + 2\sum_{i<j} \operatorname{Cov}(X_i, X_j)\Big).$$

Pairwise-independent indices give Chebyshev's bound

Write $X_j = f(J_j) \in \{0, 1\}$. If each J_j is uniform, then $\mathbb{E}X_j = \mu$ and $\mathbb{E}\widehat{\mu} = \mu$. For the variance of a sum,

$$\operatorname{Var}(\widehat{\mu}) = \frac{1}{t^2}\Big(\sum_j \operatorname{Var}(X_j) + 2\sum_{i<j} \operatorname{Cov}(X_i, X_j)\Big).$$

- If J_i, J_j are independent, so are X_i, X_j, and $\operatorname{Cov}(X_i, X_j) = 0$. Only *pairs* matter.
- A 0/1 variable with mean μ has variance $\mu(1-\mu) \le \frac{1}{4}$.

So $\operatorname{Var}(\widehat{\mu}) \le \frac{1}{4t}$, and Chebyshev's inequality gives

$$\boxed{\Pr(|\widehat{\mu} - \mu| \ge \varepsilon) \;\le\; \frac{\operatorname{Var}(\widehat{\mu})}{\varepsilon^2} \;\le\; \frac{1}{4t\varepsilon^2}.}$$

Pairwise-independent indices give Chebyshev's bound

Write $X_j = f(J_j) \in \{0, 1\}$. If each J_j is uniform, then $\mathbb{E}X_j = \mu$ and $\mathbb{E}\widehat{\mu} = \mu$. For the variance of a sum,

$$\mathrm{Var}(\widehat{\mu}) = \frac{1}{t^2}\Big(\sum_j \mathrm{Var}(X_j) + 2\sum_{i<j} \mathrm{Cov}(X_i, X_j)\Big).$$

- If J_i, J_j are independent, so are X_i, X_j, and $\mathrm{Cov}(X_i, X_j) = 0$. Only *pairs* matter.
- A 0/1 variable with mean μ has variance $\mu(1-\mu) \le \frac{1}{4}$.

So $\mathrm{Var}(\widehat{\mu}) \le \frac{1}{4t}$, and Chebyshev's inequality gives

$$\boxed{\Pr(|\widehat{\mu} - \mu| \ge \varepsilon) \;\le\; \frac{\mathrm{Var}(\widehat{\mu})}{\varepsilon^2} \;\le\; \frac{1}{4t\varepsilon^2}.}$$

What we need: indices that are each uniform on $\{0, \ldots, N-1\}$ and pairwise independent. The XOR generator makes bits, not indices, so we need a new construction.

Recall: arithmetic mod a prime

Let N be prime. $\mathbb{F}_N = \{0, 1, \ldots, N-1\}$ with $+$ and $\times$ taken mod N.

Every nonzero element has an inverse, so we can *divide* by any nonzero element.

Example, mod 11: $3 \cdot 4 = 12 \equiv 1$, so $3^{-1} = 4$, and $5/3 = 5 \cdot 4 = 20 \equiv 9$.

Recall: arithmetic mod a prime

Let N be prime. $\mathbb{F}_N = \{0, 1, \ldots, N-1\}$ with $+$ and $\times$ taken mod N.

Every nonzero element has an inverse, so we can *divide* by any nonzero element.

Example, mod 11: $3 \cdot 4 = 12 \equiv 1$, so $3^{-1} = 4$, and $5/3 = 5 \cdot 4 = 20 \equiv 9$.

Consequence: a linear equation $x \cdot z = w$ with $x \neq 0$ has exactly one solution $z = w/x$. Likewise, two equations

$$ai + b = u, \qquad aj + b = v \qquad (i \neq j)$$

have exactly one solution (a, b): subtract to get $a(j-i) = v-u$, divide by $j - i \neq 0$, then $b = u - ai$.

Pairwise-independent indices from two random numbers

Affine index generator

Let N be prime, $t \le N$, and $a, b \in \mathbb{F}_N$ independent and uniform. Then

$$J_j = aj + b \bmod N, \qquad j = 0, \ldots, t-1,$$

are each uniform on $\{0, \ldots, N-1\}$ and pairwise independent.

All N^2 seeds (a, b) are equally likely.

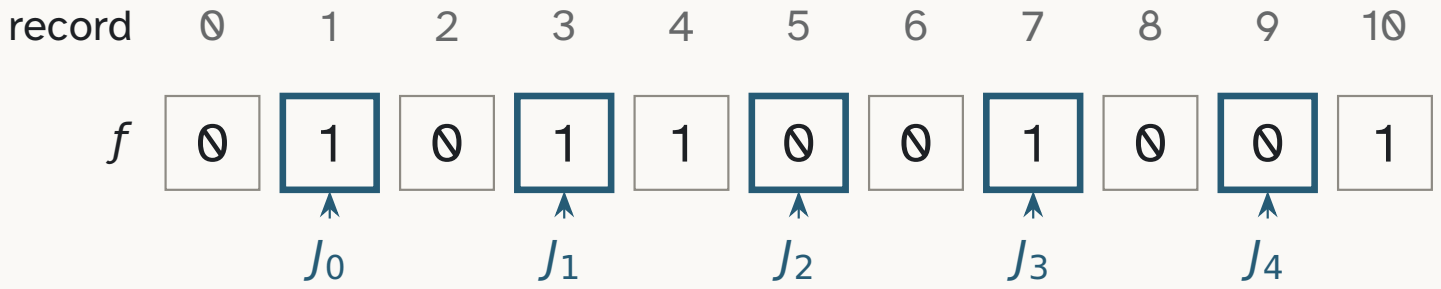
- **One index:** for each a, exactly one b gives $J_j = u$: N seeds of N^2, so $\Pr(J_j = u) = \frac{1}{N}$.
- **Two indices** $i \neq j$: exactly one seed gives $J_i = u$ and $J_j = v$, so $\Pr(J_i = u, J_j = v) = \frac{1}{N^2} = \frac{1}{N} \cdot \frac{1}{N}$.

Random bits: about $2 \log_2 N$, instead of $t \log_2 N$.

▸ N not prime: B.1

The sampler with one seed

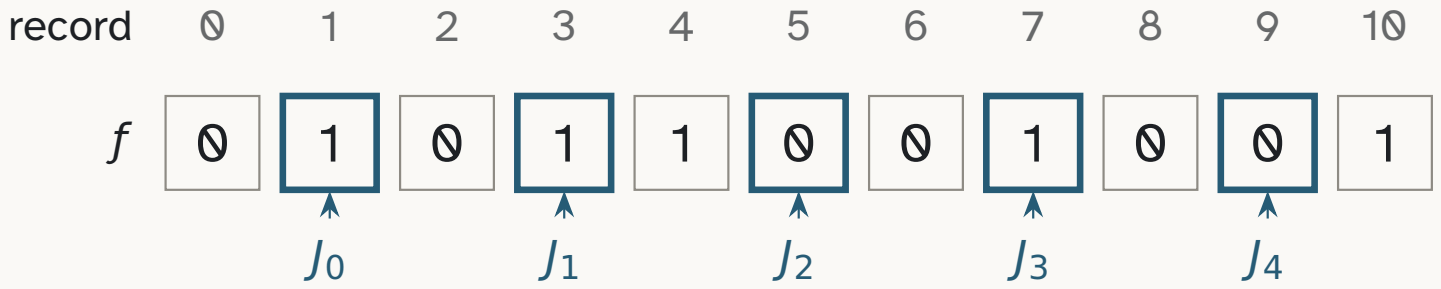
Here is what the sampler does with one seed: $N = 11$ records and $t = 5$ samples, seed $(a, b) = (2, 1)$, so $J_j = 2j + 1 \bmod 11$.



The sampled losses are $1, 1, 0, 1, 0$, so $\widehat{\mu} = \frac{3}{5} = 0.6$. The true error rate is $\mu = \frac{5}{11} \approx 0.45$.

The sampler with one seed

Here is what the sampler does with one seed: $N = 11$ records and $t = 5$ samples, seed $(a, b) = (2, 1)$, so $J_j = 2j + 1 \bmod 11$.



The sampled losses are $1, 1, 0, 1, 0$, so $\widehat{\mu} = \frac{3}{5} = 0.6$. The true error rate is $\mu = \frac{5}{11} \approx 0.45$.

Some seeds are bad: $a = 0$ samples one record five times. That is allowed. The guarantee is about a seed chosen at random, and Chebyshev bounds how often we land on a bad one.

Chebyshev is too weak when δ is small

Take $\varepsilon = 0.01$ and $\delta = 10^{-6}$. How many samples does each bound need?

indices	failure bound	samples needed
pairwise independent	$\frac{1}{4t\varepsilon^2}$ (Chebyshev)	2.5×10^9
fully independent	$2e^{-2t\varepsilon^2}$ (Hoeffding)	7.3×10^4

Chebyshev is too weak when δ is small

Take $\varepsilon = 0.01$ and $\delta = 10^{-6}$. How many samples does each bound need?

indices	failure bound	samples needed
pairwise independent	$\dfrac{1}{4t\varepsilon^2}$ (Chebyshev)	2.5×10^9
fully independent	$2e^{-2t\varepsilon^2}$ (Hoeffding)	7.3×10^4

Where Hoeffding uses full independence. With $Y_j = X_j - \mu$ and $S = \sum_j Y_j = t(\widehat{\mu} - \mu)$, the proof bounds $\Pr(S \geq u) \leq e^{-\lambda u}\, \mathbb{E}\, e^{\lambda S}$ and then splits

$$\mathbb{E}\, e^{\lambda S} = \mathbb{E} \prod_j e^{\lambda Y_j} = \prod_j \mathbb{E}\, e^{\lambda Y_j}.$$

That split is a statement about all t samples at once.

Can pairwise independence give exponential tails? No

Could some cleverer proof get exponential tails from pairwise independence alone? A counterexample says no.

Construction. Use the XOR generator with *every* nonzero label: a seed of L random bits gives $t = 2^L - 1$ output bits, fair and pairwise independent. Their average should be close to $\frac{1}{2}$.

> **Question**
>
> What is the average of the t outputs when the seed is all zeros?

Can pairwise independence give exponential tails? No

Could some cleverer proof get exponential tails from pairwise independence alone? A counterexample says no.

Construction. Use the XOR generator with *every* nonzero label: a seed of L random bits gives $t = 2^L - 1$ output bits, fair and pairwise independent. Their average should be close to $\frac{1}{2}$.

> **Question**
>
> What is the average of the t outputs when the seed is all zeros?

Every output is $\langle 0 \cdots 0, \alpha \rangle = 0$, so the average is 0, far from $\frac{1}{2}$.

That seed has probability $2^{-L} = \frac{1}{t+1}$:

$$\Pr\Big(\Big|\frac{1}{t}\sum_j X_j - \frac{1}{2}\Big| \geq \frac{1}{4}\Big) \geq \frac{1}{t+1}.$$

Can pairwise independence give exponential tails? No

Could some cleverer proof get exponential tails from pairwise independence alone? A counterexample says no.

Construction. Use the XOR generator with *every* nonzero label: a seed of L random bits gives $t = 2^L - 1$ output bits, fair and pairwise independent. Their average should be close to $\frac{1}{2}$.

> **Question**
>
> What is the average of the t outputs when the seed is all zeros?

Every output is $\langle 0 \cdots 0, \alpha \rangle = 0$, so the average is 0, far from $\frac{1}{2}$.

That seed has probability $2^{-L} = \frac{1}{t+1}$:

$$\Pr\Big(\Big|\frac{1}{t}\sum_j X_j - \frac{1}{2}\Big| \geq \frac{1}{4}\Big) \geq \frac{1}{t+1}.$$

The failure probability falls only like $1/t$, never like e^{-ct}. Better tails need more than pairwise independence.

k-wise independence

k-wise independence

Random variables $X_1, \ldots, X_t$ are **k-wise independent** if every k of them are mutually independent: for all distinct $i_1, \ldots, i_k$ and all values $x_1, \ldots, x_k$,

$$\Pr(X_{i_1} = x_1, \ldots, X_{i_k} = x_k) = \prod_{r=1}^{k} \Pr(X_{i_r} = x_r).$$

- Pairwise independence is 2-wise independence.
- k-wise independence implies k'-wise for every $k' \leq k$.
- The warm-up bits $s_1, s_2, s_1 \oplus s_2$ are 2-wise but not 3-wise independent.

Plan: Chebyshev looks at the 2nd moment, which needs 2-wise independence. Higher moments give sharper tails, and need more independence.

Try the fourth moment

Center the samples: $Y_j = X_j - \mu$, so $\mathbb{E}Y_j = 0$, and let $S = \sum_j Y_j = t(\widehat{\mu} - \mu)$.

Chebyshev is Markov's inequality applied to S^2. Apply Markov to S^4 instead:

$$\Pr(|\widehat{\mu} - \mu| \geq \varepsilon) = \Pr(|S| \geq t\varepsilon) = \Pr(S^4 \geq t^4\varepsilon^4) \leq \frac{\mathbb{E}[S^4]}{t^4\varepsilon^4}.$$

Try the fourth moment

Center the samples: $Y_j = X_j - \mu$, so $\mathbb{E}Y_j = 0$, and let $S = \sum_j Y_j = t(\widehat{\mu} - \mu)$.

Chebyshev is Markov's inequality applied to S^2. Apply Markov to S^4 instead:

$$\Pr(|\widehat{\mu} - \mu| \geq \varepsilon) = \Pr(|S| \geq t\varepsilon) = \Pr(S^4 \geq t^4\varepsilon^4) \leq \frac{\mathbb{E}[S^4]}{t^4\varepsilon^4}.$$

Expanding the fourth power,

$$S^4 = \Big(\sum_j Y_j\Big)^4 = \sum_{i,j,k,l} Y_i Y_j Y_k Y_l \qquad (t^4 \text{ terms}).$$

Try the fourth moment

Center the samples: $Y_j = X_j - \mu$, so $\mathbb{E}Y_j = 0$, and let $S = \sum_j Y_j = t(\widehat{\mu} - \mu)$.

Chebyshev is Markov's inequality applied to S^2. Apply Markov to S^4 instead:

$$\Pr(|\widehat{\mu} - \mu| \geq \varepsilon) = \Pr(|S| \geq t\varepsilon) = \Pr(S^4 \geq t^4\varepsilon^4) \leq \frac{\mathbb{E}[S^4]}{t^4\varepsilon^4}.$$

Expanding the fourth power,

$$S^4 = \Big(\sum_j Y_j\Big)^4 = \sum_{i,j,k,l} Y_i Y_j Y_k Y_l \qquad (t^4 \text{ terms}).$$

Each term involves at most 4 distinct samples. If the samples are 4-wise independent, each term has the same expectation as with fully independent samples, so $\mathbb{E}[S^4]$ is exactly what it would be under full independence.

Which fourth-moment terms survive?

Group the terms $Y_i Y_j Y_k Y_l$ by how often each index repeats:

pattern	example	expectation
some index appears once	$Y_1 Y_2^3$, $Y_1 Y_2 Y_3^2$	0
one index four times	Y_1^4	$\mathbb{E} Y_1^4$
two indices twice each	$Y_1^2 Y_2^2$	$\mathbb{E} Y_1^2 \, \mathbb{E} Y_2^2$

Which fourth-moment terms survive?

Group the terms $Y_i Y_j Y_k Y_l$ by how often each index repeats:

pattern	example	expectation
some index appears once	$Y_1 Y_2^3$, $Y_1 Y_2 Y_3^2$	0
one index four times	Y_1^4	$\mathbb{E} Y_1^4$
two indices twice each	$Y_1^2 Y_2^2$	$\mathbb{E} Y_1^2 \, \mathbb{E} Y_2^2$

Why the first row vanishes: the index that appears once is independent of the rest of the term, so its factor $\mathbb{E} Y_i = 0$ comes out.

Which fourth-moment terms survive?

Group the terms $Y_i Y_j Y_k Y_l$ by how often each index repeats:

pattern	example	expectation
some index appears once	$Y_1 Y_2^3$, $Y_1 Y_2 Y_3^2$	0
one index four times	Y_1^4	$\mathbb{E}Y_1^4$
two indices twice each	$Y_1^2 Y_2^2$	$\mathbb{E}Y_1^2 \, \mathbb{E}Y_2^2$

Why the first row vanishes: the index that appears once is independent of the rest of the term, so its factor $\mathbb{E}Y_i = 0$ comes out.

Counting the last row: for indices $i < j$, the ordered terms with two i's and two j's are $iijj$, $ijij$, $ijji$, $jiij$, $jiji$, $jjii$: six of them. So

$$\mathbb{E}[S^4] = \sum_j \mathbb{E}Y_j^4 + 6 \sum_{i<j} \mathbb{E}Y_i^2 \, \mathbb{E}Y_j^2.$$

4-wise independence: failure at most $\frac{1}{4t^2\varepsilon^4}$

Each $|Y_j| \le 1$, so $\mathbb{E}Y_j^4 \le \mathbb{E}Y_j^2 = \mathrm{Var}(X_j) \le \frac{1}{4}$. Therefore

$$\mathbb{E}[S^4] \;\le\; t \cdot \frac{1}{4} + 6\binom{t}{2} \cdot \frac{1}{16} \;=\; \frac{3t^2 + t}{16} \;\le\; \frac{t^2}{4},$$

and the fourth-moment bound gives

$$\boxed{\Pr(|\widehat{\mu} - \mu| \ge \varepsilon) \;\le\; \frac{t^2/4}{t^4\varepsilon^4} = \frac{1}{4t^2\varepsilon^4}.}$$

4-wise independence: failure at most $\frac{1}{4t^2\varepsilon^4}$

Each $|Y_j| \le 1$, so $\mathbb{E}Y_j^4 \le \mathbb{E}Y_j^2 = \mathrm{Var}(X_j) \le \frac{1}{4}$. Therefore

$$\mathbb{E}[S^4] \;\le\; t \cdot \frac{1}{4} + 6\binom{t}{2} \cdot \frac{1}{16} \;=\; \frac{3t^2 + t}{16} \;\le\; \frac{t^2}{4},$$

and the fourth-moment bound gives

$$\boxed{\Pr(|\widehat{\mu} - \mu| \ge \varepsilon) \;\le\; \frac{t^2/4}{t^4\varepsilon^4} = \frac{1}{4t^2\varepsilon^4}.}$$

indices	failure bound	samples at $\varepsilon = 0.01, \delta = 10^{-6}$
pairwise	$1/(4t\varepsilon^2)$	2.5×10^9
4-wise	$1/(4t^2\varepsilon^4)$	5×10^6

The general principle

Moments under k-wise independence

If $X_1, \ldots, X_t$ are k-wise independent, every polynomial in them of total degree at most k has the same expectation as when the X_j are fully independent with the same individual distributions.

Proof. Expand into monomials. Each monomial has degree at most k, so it involves at most k distinct variables. Their joint distribution is the same in both cases, so each monomial has the same expectation. Add up.

The general principle

Moments under k-wise independence

If $X_1, \ldots, X_t$ are k-wise independent, every polynomial in them of total degree at most k has the same expectation as when the X_j are fully independent with the same individual distributions.

Proof. Expand into monomials. Each monomial has degree at most k, so it involves at most k distinct variables. Their joint distribution is the same in both cases, so each monomial has the same expectation. Add up.

Use: S^{2q} has degree $2q$. With $2q$-wise independent samples, $\mathbb{E}[S^{2q}]$ equals its value under full independence. Any bound on that moment, and the tail bound Markov gets from it, carries over.

$O(\log \frac{1}{\delta})$-wise independence is nearly as good as full

Step 1 (full independence). Integrating Hoeffding's tail gives, for each integer $q \geq 1$,

$$\mathbb{E}[S^{2q}] \;\leq\; 2 \cdot q! \left(\frac{t}{2}\right)^{q} \;\leq\; 2\left(\frac{qt}{2}\right)^{q}. \qquad \blacktriangleright \text{A.1}$$

$O(\log \frac{1}{\delta})$-wise independence is nearly as good as full

Step 1 (full independence). Integrating Hoeffding's tail gives, for each integer $q \geq 1$,

$$\mathbb{E}[S^{2q}] \;\leq\; 2 \cdot q! \left(\frac{t}{2}\right)^q \;\leq\; 2\left(\frac{qt}{2}\right)^q. \qquad \blacktriangleright \text{A.1}$$

Step 2 ($2q$-wise independence). The moment is the same, so Markov on S^{2q} gives

$$\Pr(|\widehat{\mu} - \mu| \geq \varepsilon) \;\leq\; \frac{\mathbb{E}[S^{2q}]}{(t\varepsilon)^{2q}} \;\leq\; 2\left(\frac{q}{2t\varepsilon^2}\right)^q.$$

$O(\log \frac{1}{\delta})$-wise independence is nearly as good as full

Step 1 (full independence). Integrating Hoeffding's tail gives, for each integer $q \geq 1$,

$$\mathbb{E}[S^{2q}] \;\leq\; 2 \cdot q! \left(\frac{t}{2}\right)^q \;\leq\; 2\left(\frac{qt}{2}\right)^q. \qquad \text{A.1}$$

Step 2 ($2q$-wise independence). The moment is the same, so Markov on S^{2q} gives

$$\Pr(|\widehat{\mu} - \mu| \geq \varepsilon) \;\leq\; \frac{\mathbb{E}[S^{2q}]}{(t\varepsilon)^{2q}} \;\leq\; 2\left(\frac{q}{2t\varepsilon^2}\right)^q.$$

Step 3 (choose q, t). With $t = q/\varepsilon^2$ the bound is $2 \cdot 2^{-q}$, which is at most δ once $q = \lceil \log_2(2/\delta) \rceil$. That is

$t = O\left(\varepsilon^{-2} \log(1/\delta)\right)$ samples: the same order as Hoeffding.

k-wise-independent indices from a random polynomial

Recall (interpolation): through k points with distinct x-coordinates passes exactly one polynomial of degree $< k$ (for $k = 2$: one line). This holds mod a prime too.

k-wise-independent indices from a random polynomial

Recall (interpolation): through k points with distinct x-coordinates passes exactly one polynomial of degree $< k$ (for $k = 2$: one line). This holds mod a prime too.

Polynomial index generator

Let N be prime, $t \le N$. Pick $A_0, \ldots, A_{k-1} \in \mathbb{F}_N$ independently and uniformly, and set $J_j = A_0 + A_1 j + \cdots + A_{k-1} j^{k-1} \bmod N$. Then $J_0, \ldots, J_{t-1}$ are uniform and k-wise independent.

Why: fix k distinct inputs and k target outputs. Exactly one of the N^k equally likely coefficient seeds hits all the targets, so they have probability $N^{-k} = (\frac{1}{N})^k$. ▸ A.2

For $k = 2$ this is the affine generator. The seed is k numbers, about $k \log_2 N$ bits.

What each level of independence buys

At $\varepsilon = 0.01$ and $\delta = 10^{-6}$ (so $q = 21$):

indices	samples needed	numbers in the seed
pairwise	2.5×10^9	2
4-wise	5×10^6	4
42-wise	2.1×10^5	42
fully independent	7.3×10^4	7.3×10^4

A seed of 42 random numbers needs only about 3 times the samples of full independence.

What each level of independence buys

At $\varepsilon = 0.01$ and $\delta = 10^{-6}$ (so $q = 21$):

indices	samples needed	numbers in the seed
pairwise	2.5×10^9	2
4-wise	5×10^6	4
42-wise	2.1×10^5	42
fully independent	7.3×10^4	7.3×10^4

A seed of 42 random numbers needs only about 3 times the samples of full independence.

Contrast with Part 1: there are N^{42} such seeds. Drawing one at random is easy; trying them all is hopeless.

What we learned

Find what the analysis uses. The random cut and random-marking MIS analyses only use pairs of bits. Bits from the XOR generator, with a seed of $O(\log n)$ random bits, satisfy the same analysis.

Few seeds can replace randomness. With polynomially many seeds and a computable quality for each run, trying every seed gives a deterministic algorithm that meets the average. The runs are independent, so this also gives deterministic parallel algorithms (MIS in $O(\log^2 n)$ time).

More independence buys sharper tails. When an algorithm must stay randomized, k-wise independence preserves moments up to degree k: pairwise gives Chebyshev, $O(\log \frac{1}{\delta})$-wise gives Hoeffding-quality bounds, from a seed of a few random numbers.

Check your understanding

1. Explain why bits X, Y with each pair $00, 01, 10, 11$ equally likely must be fair and independent.
2. For labels $a = 101$ and $b = 011$, list the four seeds in one group from the proof and check that they produce all four output pairs.
3. The $2n + 1$ seed cuts have average $m/2$. Why can they still miss every maximum cut?
4. Why does Seed MIS search seeds one round at a time?
5. Why is pairwise independence enough for Chebyshev, but not for Hoeffding?
6. Why can't we derandomize the sampler by trying every seed?

Questions?

Appendix

Deferred proofs

A.1 The independent moment bound

A.2 The polynomial generator, at fewer than k inputs

A.3 The compressed prime-field hash family is universal

Implementation details

B.1 Test sets whose size is not prime

Worked computations

C.1 All eight seeds on the star

C.2 Score counts for the 16-cycle

More applications

D.1 Deterministic perfect hashing

D.2 Set balancing

References

A.1 The independent moment bound

For independent samples in $[0,1]$, Hoeffding gives $\Pr(|S| \geq u) \leq 2e^{-2u^2/t}$. For an integer $q \geq 1$:

$$\begin{aligned}\mathbb{E}[S^{2q}] &= \int_0^\infty 2q\,u^{2q-1}\Pr(|S|>u)\,du \\ &\leq 4q\int_0^\infty u^{2q-1}e^{-2u^2/t}\,du \;=\; 4q\cdot\frac{(q-1)!}{2}\Big(\frac{t}{2}\Big)^q \\ &= 2\cdot q!\Big(\frac{t}{2}\Big)^q \;\leq\; 2\Big(\frac{qt}{2}\Big)^q.\end{aligned}$$

(The integral uses $\int_0^\infty u^{2q-1}e^{-cu^2}\,du = (q-1)!/(2c^q)$.)

Only this moment carries over to $2q$-wise independence. We never claim the Hoeffding tail itself holds there.

Back

A.2 The polynomial generator, at fewer than k inputs

Let $P(z) = \sum_{r=0}^{k-1} A_r z^r$ with independent uniform $A_r \in \mathbb{F}_N$. Fix $m \le k$ distinct inputs $j_1, \ldots, j_m$.

The outputs $(P(j_1), \ldots, P(j_m))$ are a linear function of $(A_0, \ldots, A_{k-1})$. Its first m columns form the Vandermonde matrix

$$V = \begin{pmatrix} 1 & j_1 & \cdots & j_1^{m-1} \\ \vdots & \vdots & & \vdots \\ 1 & j_m & \cdots & j_m^{m-1} \end{pmatrix}, \qquad \det V = \prod_{a<b} (j_b - j_a) \neq 0.$$

So the map has rank m, and every target vector has exactly N^{k-m} of the N^k coefficient vectors as preimages: probability N^{-m}. Back

A.3 The compressed prime-field hash family is universal

For prime P, pick $a \neq 0$ and b uniformly in $\mathbb{F}_P$. For $x \neq y$, the pair $(ax + b,\ ay + b)$ is uniform over ordered pairs of *distinct* field elements.

Reducing mod $R \leq P$ splits the field into R buckets of sizes $w_1, \ldots, w_R$. The two values collide with probability

$$\frac{\sum_i w_i(w_i - 1)}{P(P-1)} \leq \frac{1}{R}.$$

To check: write $P = Rw + e$ with $0 \leq e < R$. Then $\sum_i w_i(w_i - 1) = Rw(w-1) + 2ew$, and

$$P(P-1) - R\big[Rw(w-1) + 2ew\big] = Rw(R-1) + e(e-1) \geq 0.$$

If $R > P$, distinct field values never collide after reduction.

B.1 Test sets whose size is not prime

For $N_0 \geq 2$ real records, choose a prime N with $N_0 \leq N < 2N_0$ and add $N - N_0$ dummy records with loss 0. If μ' is the padded mean,

$$\mu = \frac{N}{N_0}\,\mu'.$$

Estimate μ' to accuracy $\varepsilon N_0/N > \varepsilon/2$ and rescale: this at most quadruples the sample count.

If the required t exceeds N, just read every record.

Uniform elements of $\mathbb{F}_N$ from fair bits: draw $\lceil \log_2 N \rceil$ bits and retry if the number is $\geq N$. Each try succeeds with probability $> 1/2$, so this uses $O(\log N)$ bits in expectation.

Back

C.1 All eight seeds on the star

Labels 001, …, 111 for vertices $1, \ldots, 7$; center 1.

s	bits $B_1 \ldots B_7$	edges cut
000	0000000	0
001	1010101	3
010	0110011	4
011	1100110	3
100	0001111	4
101	1011010	3
110	0111100	4
111	1101001	3

Average $24/8 = 3 = m/2$; best 4; optimum 6. Every nonzero s gives exactly four 1s.

Back

C.2 Score counts for the 16-cycle

Vertices $1, \ldots, 16$ with 5-bit labels; seeds $s^{(1)}, s^{(2)} \in \{0,1\}^5$: $1{,}024$ in all. Vertex v is marked when $\langle s^{(1)}, a_v \rangle = \langle s^{(2)}, a_v \rangle = 0$.

$\lvert T \rvert$	0	1	2	3	4	5	6	7	8
seeds	121	270	66	414	108	18	6	18	3

Total accepted over all seeds: $2{,}352$, average ≈ 2.30.

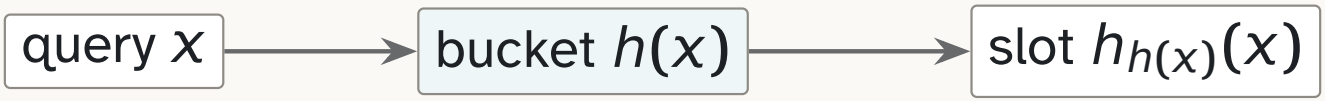
Each vertex is marked by $256 = 1024/4$ seeds, and each pair of distinct vertices is marked together by $64 = 1024/16$ seeds, exactly as pairwise independence with $p = \frac{1}{4}$ predicts.

Back

D.1 A static dictionary

Store K fixed keys so that membership queries take $O(1)$ time in the worst case, using $O(K)$ words of space.

A primary hash sends keys to buckets. Each bucket gets its own secondary hash into its own table, chosen so that no two of its keys collide.



A query reads one slot and compares the stored key with x.

> **Question**
>
> How big must a bucket's secondary table be for a collision-free hash to exist?

D.1 Sizing the secondary tables

A bucket holds b keys. Hash them with a universal family into R slots. The expected number X of colliding pairs is

$$\mathbb{E}[X] \le \binom{b}{2}\frac{1}{R}.$$

Question

For which R is $\mathbb{E}[X] < \frac{1}{2}$, so that at least half the functions have no collisions?

D.1 Sizing the secondary tables

A bucket holds b keys. Hash them with a universal family into R slots. The expected number X of colliding pairs is

$$\mathbb{E}[X] \le \binom{b}{2}\frac{1}{R}.$$

Question

For which R is $\mathbb{E}[X] < \frac{1}{2}$, so that at least half the functions have no collisions?

$R = b^2$ works: $\mathbb{E}[X] \le \dfrac{b(b-1)}{2b^2} < \dfrac{1}{2}$. So try functions until one has no collisions.

New problem: the total space is $\sum_i b_i^2$, which could be as large as K^2.

D.1 Choosing the primary hash

With bucket sizes b_i, count each key once and each colliding pair twice:

$$\sum_i b_i^2 \;=\; K + 2 \sum_{\{x,y\}} \mathbf{1}\{h(x) = h(y)\}.$$

For a universal family into K buckets,

$$\mathbb{E} \sum_i b_i^2 \;\le\; K + 2\binom{K}{2}\frac{1}{K} \;=\; 2K - 1,$$

so some primary hash has $\sum_i b_i^2 < 2K$. We can compute this quantity for each candidate and keep one that passes.

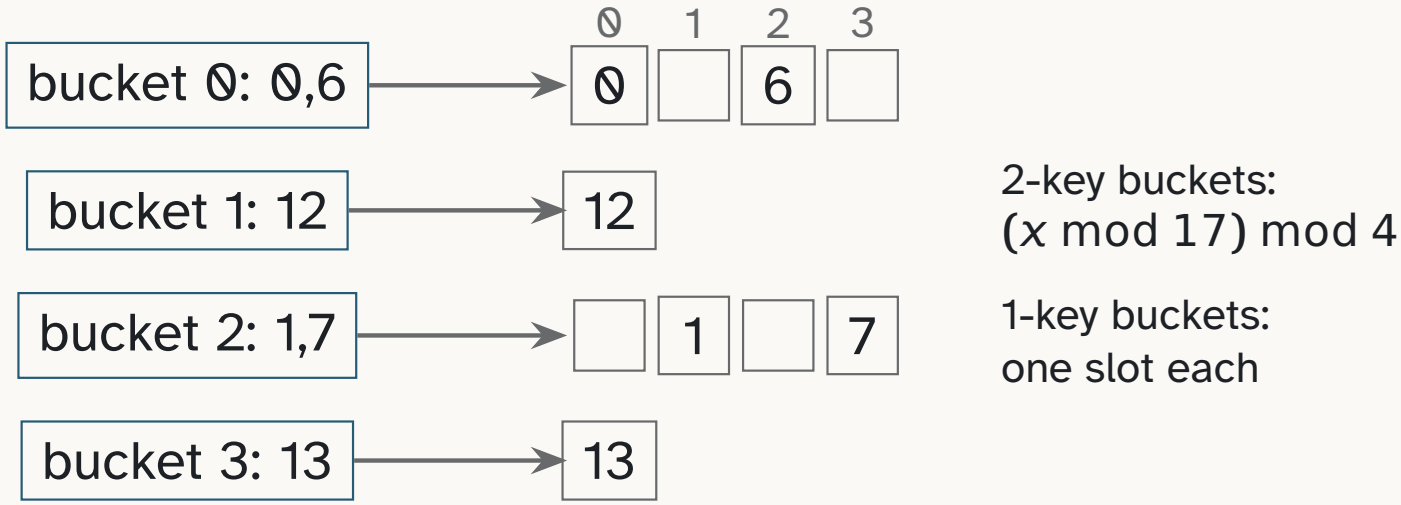
D.1 Example: six keys

Keys $\{0, 1, 6, 7, 12, 13\}$, universe $\{0, \ldots, 16\}$, family $h_{a,b}(x) = ((ax + b) \bmod 17) \bmod R$ with $1 \le a \le 16$, $0 \le b \le 16$.

Primary range $R = 6$; target $\sum_i b_i^2 < 12$:

bucket	$(a, b) = (1, 0)$	$(a, b) = (2, 0)$
0	0, 6, 12	0, 6
1	1, 7, 13	12
2		1, 7
3		13
4, 5		
$\sum_i b_i^2$	18 (fails)	10 (passes)

D.1 The finished two-level table



Secondary tables use $4 + 1 + 4 + 1 = 10$ slots; buckets 4 and 5 are empty. Storing the keys in their slots lets queries reject non-members.

With a polynomial-size universe, both searches try polynomially many functions, so construction is deterministic polynomial time.

▸ Universality: A.3

D.2 Set balancing

Given vectors $v_1, \ldots, v_n \in \mathbb{R}^d$, choose signs $\sigma_i \in \{\pm 1\}$ to keep $\left\|\sum_i \sigma_i v_i\right\|_2$ small. With fair pairwise-independent signs,

$$\mathbb{E}\left\|\sum_i \sigma_i v_i\right\|_2^2 = \sum_i \|v_i\|_2^2 + 2\sum_{i<j}\langle v_i, v_j\rangle\, \mathbb{E}[\sigma_i \sigma_j] = \sum_i \|v_i\|_2^2.$$

Set $\sigma_i = (-1)^{B_i}$ with B_i from the XOR generator, try all $2^L \le 2n+1$ seeds, and keep the smallest norm:

$$\left\|\sum_i \sigma_i v_i\right\|_2^2 \;\le\; \sum_i \|v_i\|_2^2.$$

This matches the random bound; it need not be the best signing.

References

- Mitzenmacher and Upfal, *Probability and Computing*, §15.1 and §15.2.1: short seeds, MAX-CUT derandomization, sampling with fewer random bits.
- J. P. Schmidt, A. Siegel, A. Srinivasan, *Chernoff–Hoeffding Bounds for Applications with Limited Independence*, SIAM J. Discrete Math. 8(2), 1995. `https://www.cs.umd.edu/~srin/PDF/ch-bounds.pdf`
- M. Luby, *A Simple Parallel Algorithm for the Maximal Independent Set Problem*, SIAM J. Computing 15(4), 1986. The lecture proves a bounded-degree variant. `https://www.cs.cmu.edu/~guyb/paralg/papers/Luby86.pdf`
- M. L. Fredman, J. Komlós, E. Szemerédi, *Storing a Sparse Table with $O(1)$ Worst Case Access Time*, JACM 31(3), 1984.