Hashing and Sketching

Randomness with small memory

Reusable randomness, compact data structures

A hash function assigns a consistent random label to every key. With a carefully chosen amount of independence, these labels support exact dictionaries, approximate membership tests, distinct-element counting, and frequency estimation. The central questions are which features of a random function the analysis actually uses, and how to preserve those features with a short seed.

Advanced Algorithms teaching team

Nanjing University

Learning goals and reading routes

This lecture continues the randomization strand after fingerprinting. By the end, you should be able to:

1. separate collision universality from pairwise independence and construct small-seed hash families;
2. derive FKS perfect hashing from a bound on collision pairs;
3. derive the Bloom-filter heuristic, identify its independence gap, and distinguish it from a rigorous guarantee;
4. prove the guarantees of the minimum, trailing-zero, and bottom-k distinct-count sketches; and
5. derive Count-Min's additive error bound and account for both counters and random seeds.

For a first pass, follow the collision argument in Section 1, the universal and pairwise-independent constructions in Section 2, and the dictionaries and filters in Sections 3–4. In Section 5, first understand the minimum sketch under ideal hashing, then study the finite trailing-zero and bottom-k sketches. Finish with Count-Min in Section 6.

The higher-independence construction, the supplementary elementary Bloom-filter bounds in Section 4.5, and the research extensions can be revisited after this first pass. The main Bloom-filter discussion follows the heuristic calculation through its independence gap to an exact formula; a later lecture will return to the problem using concentration of measure. The appendices contain only limited independence and deviation inequalities. Finite-field basics are prerequisites from Lecture 2 (Fingerprinting); the hash constructions recall the particular facts they use. Each problem introduces its own notation and computational assumptions where they are needed.

Contents

1 Random maps, collisions, and the birthday scale

1.1 Random maps as a model for hashing

Let $\mathcal{U}$ be a universe of $N \geq 2$ possible keys, and write $[r] = \{1, \ldots, r\}$ for an integer $r \geq 0$ (so $[0] = \varnothing$). A *hash function* $h : \mathcal{U} \to [m]$ maps each key to one of $m \geq 1$ values. The same key always receives the same value. We use $\{0, \ldots, r-1\}$ explicitly when working with residues instead of indices.

A uniformly random map gives independent uniform values to *distinct* keys. This ideal model is the simple uniform hashing assumption (SUHA). We fix the input keys before choosing the map; probabilities refer to this random choice. A full lookup table for the map would store N values, each needing $\lceil \log_2 m \rceil$ bits. Section 2 replaces this large table by a short random seed that retains the properties the analysis needs.

1.2 The birthday calculation

Fix $n \geq 1$ distinct keys. Regard each key as a ball and its hash value as the index of a bin. Under a uniformly random map, these are n independent uniform throws into m bins. Repeating one key would return it to the same bin, so repeated occurrences are not new independent throws. For $n \leq m$, the event $\mathcal{E}$ that all bins have load at most one satisfies

$$\mathbb{P}(\mathcal{E}) = \frac{m(m-1)\cdots(m-n+1)}{m^n} = \prod_{i=0}^{n-1} \left(1 - \frac{i}{m}\right). \tag{1}$$

The first expression counts injective maps. Alternatively, condition on the first i balls occupying different bins: the next ball avoids them with probability $1 - i/m$, and the chain rule gives the product. If $n > m$, the probability is zero.

Approximating each factor by $e^{-i/m}$ suggests $\mathbb{P}(\mathcal{E}) \approx e^{-n(n-1)/(2m)}$. Thus about $\sqrt{m}$ distinct keys should suffice for a substantial chance of collision: this is the *birthday scale*. For 365 equally likely birthdays, the exact formula gives collision probability about 0.9917 for 58 students.

For a quantitative justification when $n \leq m$, set $\rho = (n-1)/m < 1$ and $A = n(n-1)/(2m)$. The elementary inequality

$$-x - \frac{x^2}{2(1-\rho)} \leq \ln(1-x) \leq -x \qquad (0 \leq x \leq \rho)$$

gives

$$-A - \frac{n(n-1)(2n-1)}{12m^2(1-\rho)} \;\leq\; \ln \mathbb{P}(\mathcal{E}) \;\leq\; -A. \tag{2}$$

In particular, $n^3/m^2 \to 0$ makes the relative error in the exponential approximation tend to zero. If $n/\sqrt{m} \to c > 0$, the no-collision probability tends to $e^{-c^2/2}$, confirming the predicted scale.

Surjectivity (coupon collection) and the maximum bin load are other questions about the same random map. This class uses the collision question to design dictionaries and sketches.

1.3 The collision count needs much less randomness

For distinct keys $x_1, \ldots, x_n$, define

$$C = \sum_{1 \leq i < j \leq n} \mathbf{1}\{h(x_i) = h(x_j)\}.$$

If every pair collides with probability at most $1/m$, then linearity of expectation and Markov's inequality (recalled in Appendix B) give

$$\mathbb{E}C \le \frac{\binom{n}{2}}{m}, \qquad \mathbb{P}(C \ge 1) \le \frac{\binom{n}{2}}{m}. \tag{3}$$

Neither inequality needs independence among the collision indicators. In particular, $m = n^2$ makes a fixed set collision-free with probability greater than $1/2$.

This upper bound on the probability of a collision is enough for perfect hashing. It does not reproduce the entire birthday distribution, or establish that a collision is likely above the birthday scale. Those stronger conclusions require stronger assumptions.

2 Hash families with short descriptions

A *hash family* is a collection of functions together with a rule for sampling one of them. In an implementation, a short random *seed* specifies the sampled function. We sample the seed once and reuse the resulting function for every evaluation. The question is which distributional properties can be obtained with a small seed.

2.1 Two distinct randomness requirements

Definition 2.1. A distribution on functions $h : \mathcal{U} \to [m]$ is *universal* (or collision-universal) if, for all $x \neq y$,

$$\mathbb{P}[h(x) = h(y)] \le 1/m.$$

It is *pairwise independent and uniform* if, for every $x \neq y$ and $a, b \in [m]$,

$$\mathbb{P}[h(x) = a,\ h(y) = b] = 1/m^2.$$

The second property is also called *strong 2-universality*; it implies the first by summing over $a = b$. The converse is false. Collision universality already suffices for the bound in (3); pairwise independence also controls the joint distribution of two outputs and will be useful for variance calculations. Carter and Wegman introduced universal hashing [1].

We first construct pairwise-independent field-valued hashes, then adapt the range to the intended application.

2.2 Affine hashing over a field

For a prime power q, let $\mathbb{F}_q$ be a field of q elements, with the universe embedded injectively in it. Choose a, b independently and uniformly from $\mathbb{F}_q$, *including* $a = 0$, and set

$$h_{a,b}(x) = ax + b.$$

For distinct x, y and prescribed u, v, the equations $ax + b = u$, $ay + b = v$ have the unique solution

$$a = (u - v)(x - y)^{-1}, \qquad b = u - ax.$$

Exactly one of the q^2 seeds gives the prescribed pair. Thus this family is pairwise independent and uniform, using $2 \log_2 q$ seed bits when q is a power of two.

Recall the finite-field toolkit from Lecture 2 (Fingerprinting): every nonzero field element has an inverse, and elements of $\mathbb{F}_{2^w}$ can be represented by w bits. The field operations matter: addition is bitwise XOR, and multiplication is polynomial multiplication modulo a fixed irreducible polynomial, not ordinary integer multiplication modulo 2^w. The construction of these fields is covered in Lecture 2.

2.3 Universal hashing to an arbitrary table size

For $\mathcal{U} = \{0, \ldots, N-1\}$, choose a prime $p \geq N$. Choose a uniformly from $\{1, \ldots, p-1\}$ and b uniformly from $\{0, \ldots, p-1\}$, and set

$$g_{a,b}(x) = ((ax + b) \bmod p) \bmod m.$$

The seed uses $O(\log p)$ bits. For distinct x, y, the ordered pair of intermediate residues is uniform over all $p(p-1)$ distinct pairs.

Write $p = qm + r$, $0 \leq r < m$. The residue classes modulo m have sizes q and $q+1$, so the number of ordered distinct pairs lying in the same class is

$$A = mq(q-1) + 2rq.$$

Since

$$p(p-1) - mA = mq(m-1) + r(r-1) \geq 0,$$

we have $A/[p(p-1)] \leq 1/m$. Thus g is universal. The counting argument also shows why exact uniformity is not needed for the collision bound.

Reduction modulo a table size does not preserve uniformity

This construction is generally *not* pairwise independent. Already when $m = p$, its outputs on distinct keys never coincide, whereas two independent uniform values coincide with probability $1/p$. When m does not divide p, the output marginals are usually nonuniform as well.

One may fix a prime $N \leq p < 2N$ for $N > 1$. Selecting field parameters is a setup step; the dictionary construction bounds below count trials after a suitable hash family is available.

2.4 Exact independence on a smaller range

Some sketches need uniform bit strings, not just a collision bound. When the desired range is a power of two, a balanced projection preserves both uniformity and pairwise independence. If $q = 2^w$ and $m = 2^\ell$ with $\ell \leq w$, view field elements as w-bit strings and retain any fixed ℓ coordinates of $ax + b$. Each output string has exactly $2^{w-\ell}$ preimages. Applying this balanced projection separately to independent uniform field elements preserves their independence and makes the projected outputs uniform. This yields pairwise independent maps to a power-of-two range. Taking $w \geq \lceil \log_2 N \rceil$ and $w \geq \ell$ costs $O(\log N + \log m)$ seed bits.

2.5 Extension to higher independence

The finite-family implementations later in this lecture need only collision universality or pairwise independence within each hash function. We now extend the same construction to more outputs

at once. A family is *k-wise independent and uniform* if the outputs at any $t \le k$ distinct inputs are independent and uniform; Appendix A recalls the underlying notion of limited independence.

Embed the universe in a field $\mathbb{F}_q$ with $q \ge N$. Choose independent uniform coefficients $a_0, \dots, a_{k-1} \in \mathbb{F}_q$ and use

$$h(x) = a_0 + a_1 x + \cdots + a_{k-1} x^{k-1}.$$

The degree is *at most* $k-1$. To prove k-wise independence directly, fix $t \le k$ distinct inputs $x_1, \dots, x_t$. The degree-$(t-1)$ Lagrange basis polynomials

$$\ell_i(X) = \prod_{\substack{j \in [t] \\ j \ne i}} \frac{X - x_j}{x_i - x_j}$$

are well-defined because the denominators are nonzero. They satisfy $\ell_i(x_i) = 1$ and $\ell_i(x_j) = 0$ for $j \ne i$. Thus any prescribed values $y_1, \dots, y_t$ are attained by $\sum_i y_i \ell_i(X)$, a polynomial of degree at most $t - 1 \le k - 1$. The evaluation map from the k coefficients to the t outputs is therefore a surjective linear map. Its kernel has dimension $k - t$, so each output tuple has exactly q^{k-t} coefficient preimages and probability q^{-t}.

The description has $k \log_2 q$ bits, and Horner's rule evaluates the hash in $O(k)$ field operations. The balanced projection above also preserves k-wise independence.

The term *k-universal* has several conventions in the literature. We use *universal* for a pairwise collision bound and explicitly say *k-wise independent* for joint uniformity. The all-k-keys-collide bound, $\mathbb{P}[h(x_1) = \cdots = h(x_k)] \le m^{-(k-1)}$ for distinct inputs, follows by summing the joint probabilities of the m constant output tuples. This collision condition alone is weaker than k-wise independence.

3 Exact set membership: FKS perfect hashing

A *static dictionary* represents a fixed set $S \subseteq \mathcal{U}$, with $n = |S|$, and answers whether a queried key belongs to S. Our goal is an exact answer in constant time using linear space. A hash function is *perfect for S* if it is injective on S; it need not be injective on the whole universe.

For the running-time bounds, we use a word-RAM model: a word holds $\Theta(\log N)$ bits, and memory access and arithmetic on a constant number of words cost constant time. We also assume constant-time evaluation of the chosen hash family. The space bounds count stored keys, table metadata, and hash seeds. The empty set needs only an empty marker; the construction below first considers $n \ge 1$.

3.1 A quadratic-space starting point

Choose a universal hash function into n^2 slots. By (3), a collision-free choice exists and a random trial succeeds with probability greater than $1/2$. Test a trial by inserting all keys and detecting a collision; repeat if necessary. Once successful, store each key x in slot $h(x)$.

A query must compare the stored key with x, not just inspect whether the slot is occupied. A nonmember can hash to the slot of a member. Empty slots have a separate empty marker. This produces an exact dictionary with $O(1)$ query time but $O(n^2)$ slots.

3.2 Two levels turn squared bucket sizes into linear space

The FKS scheme of Fredman, Komlós, and Szemerédi [2] first partitions S into n buckets with a universal primary hash $h : \mathcal{U} \to [n]$. Let

$$B_i = \{x \in S : h(x) = i\}, \qquad b_i = |B_i|, \qquad Q = \sum_{i=1}^{n} b_i^2.$$

Let C now count collision pairs under this primary hash. Counting within its buckets gives the *deterministic* identity

$$Q = \sum_i \left(b_i + 2\binom{b_i}{2} \right) = n + 2C.$$

Consequently,

$$\mathbb{E}Q \le n + 2\frac{\binom{n}{2}}{n} = 2n - 1 < 2n. \tag{4}$$

Markov's inequality shows $\mathbb{P}(Q > 4n) < 1/2$. We can therefore find a primary hash with $Q \le 4n$ by a constant expected number of trials. Then bucket i receives its own collision-free secondary table with b_i^2 slots.

FKS construction and query

Build on a set of n distinct keys.

1. If $n = 0$, store the empty dictionary and return.
2. Resample a universal primary hash to $[n]$ until $\sum_i b_i^2 \le 4n$.
3. For each nonempty bucket B_i, independently resample a universal secondary hash $h_i : \mathcal{U} \to [b_i^2]$ until it has no collisions on B_i. Store its seed, a pointer to the secondary table, and the keys in that table. A singleton bucket needs only its key.

Query x. If the dictionary is empty, return no. Compute $i = h(x)$. If B_i is empty, return no. For a singleton bucket, compare its stored key directly with x. Otherwise return yes exactly when the secondary slot selected by $h_i(x)$ contains x.

Theorem 3.1 (FKS static dictionary)**.** *Given constant-time hash evaluation and a suitable universal family, the construction uses $O(n)$ words, takes expected $O(n)$ time, and answers every membership query correctly in worst-case $O(1)$ time. A word has $\Theta(\log N)$ bits.*

Proof. The key comparison proves exact correctness after construction. A query uses at most two hash evaluations and a constant number of memory accesses.

The accepted primary hash gives at most $4n$ secondary slots. Bucket sizes, pointers, and secondary seeds contribute $O(n)$ further words. In the accepted layout $b_i^2 \le 4n$, so all addresses fit in $O(\log N)$ bits.

Each primary trial costs $O(n)$ time to initialize bucket counts and distribute the keys, and succeeds with probability greater than $1/2$. For a fixed nonempty bucket, a fresh secondary trial fails with probability at most

$$\frac{\binom{b_i}{2}}{b_i^2} < \frac{1}{2}.$$

Even clearing the whole secondary table costs only $O(b_i^2)$ per trial. Thus the expected secondary work, conditional on the accepted primary partition, is $O(\sum_i b_i^2) = O(n)$. Adding the primary work proves the expected bound. □

This is a *Las Vegas* construction: randomness affects the construction time, while the completed dictionary answers all queries exactly. The expectation concerns preprocessing; it does not weaken the worst-case query guarantee.

3.3 Dynamic and succinct dictionaries

FKS is static. Dynamic perfect hashing supports updates as well as membership queries; Dietzfelbinger et al. [3] give linear-word space, worst-case constant lookup time, and constant amortized expected update time. This is a separate construction.

To assess how compact a dictionary can be, any exact representation for all n-element sets must distinguish $\binom{N}{n}$ possibilities. It therefore needs at least $\lceil \log_2 \binom{N}{n} \rceil$ bits. FKS uses $O(n \log N)$ bits, which need not match that benchmark for every relation between n and N.

Succinct dynamic dictionaries study the time needed for operations when space approaches this information bound. Bender et al. [4] obtain a tradeoff, for $\Theta(\log n)$-bit keys and values, between $O(k)$ update time and $O(\log^{(k)} n)$ wasted bits per key, with constant query time and high-probability guarantees. Here $\log^{(k)}$ means k iterated logarithms, and k ranges up to $\log^* n$. Li et al. [5] prove matching $\Omega(k)$ expected-time lower bounds for at least one of insertion, deletion, or query in the corresponding redundancy regime. Their cell-probe model has universe size $N = n^{1+\Theta(1)}$ and word size $\Theta(\log N)$; computation between memory probes is free. The value-bearing setting also yields update lower bounds regardless of query time. These are model-specific time–space results; their proofs are beyond this introductory treatment.

4 Approximate membership: Bloom filters

An exact dictionary stores enough information to distinguish a key from every nonmember. A *filter* saves space by allowing *false positives*: it may answer yes on a nonmember, but must answer yes on every stored key.

Let $S \subseteq \mathcal{U}$ have size $n \geq 1$, and fix $0 < \delta \leq 1/2$. For each fixed $x \notin S$, we seek false-positive probability at most δ. Both S and x are fixed independently of the hash seeds. This is a per-query probability guarantee; it does not assert that every nonmember is rejected simultaneously.

4.1 One bit array and one hash function

Initialize an m-bit array to zero and set bit $h(x)$ for every $x \in S$. Query x by testing this bit. With an ideal uniform hash function, a fixed nonmember avoids all n stored hashes with probability $(1 - 1/m)^n$, so

$$\mathbb{P}(\text{false positive}) = 1 - (1 - 1/m)^n.$$

With a universal family the simpler union bound gives at most n/m. Taking $m \geq n/\delta$ suffices; under ideal hashing the exact expression has this scaling for small δ. Repetition improves that dependence.

4.2 The shared-array Bloom filter: a heuristic analysis

Bloom's construction [6] uses $k \geq 1$ hash functions $h_1, \ldots, h_k : \mathcal{U} \to [m]$ to the same array. Throughout the shared-array analysis, assume these functions are fully random and mutually independent. The kn hash values of the n distinct stored keys are therefore independent uniform throws into m positions.

Bloom filter

Initialize $A[1], \ldots, A[m]$ to zero and choose hash functions $h_1, \ldots, h_k$. For every inserted key x, set $A[h_j(x)] = 1$ for every $j \in [k]$. On query x, answer yes if and only if all k tested bits are one.

There are no false negatives. To estimate the false-positive probability, first examine one array position. It remains zero precisely when all kn insertion hashes avoid it. Thus its probability of being one is exactly

$$p_1 = 1 - \left(1 - \frac{1}{m}\right)^{kn}.$$

A fixed nonmember also tests a one with probability p_1 at each of its k query positions. If we *temporarily treat these k events as independent*, we obtain the heuristic

$$\mathbb{P}(\text{false positive}) \overset{\text{heuristic}}{\approx} p_1^k = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k \approx \left(1 - e^{-kn/m}\right)^k. \tag{5}$$

There are two distinct steps here: multiplying the marginal probabilities is an unproved independence assumption; replacing $(1 - 1/m)^{kn}$ by $e^{-kn/m}$ is an analytic approximation. We will examine the first step shortly.

The parameter choice suggested by the heuristic. Write $c = m/n$ for the number of array bits per stored key. The suggested error formula becomes

$$\mathbb{P}(\text{false positive}) \approx (1 - e^{-k/c})^k. \tag{6}$$

For fixed c, minimizing the right-hand side over positive real k gives $k = c \ln 2$. At this choice the expected occupied fraction is approximately $1/2$, and the predicted error is

$$\exp[-c(\ln 2)^2] \approx (0.6185)^c, \qquad m \approx \frac{n \ln(1/\delta)}{(\ln 2)^2}.$$

An implementation uses a positive integer k: compare the adjacent admissible integers (use $k = 1$ if the real optimum is below one). These formulas explain the usual design rule, but they are not yet a finite-size error guarantee.

4.3 Why conditional independence is not enough

The query hashes are independent, so why can we not multiply the probabilities? The issue is that all k tests inspect the *same random array*.

Let B denote the number of one-bits after insertion, and fix the entire built array. The query hashes of a fixed nonmember remain independent and uniform. Given this array, the tested-one

events really are independent, each with probability B/m. Averaging their product over the random array gives the exact identity

$$\mathbb{P}(\text{false positive}) = \mathbb{E}\left[\left(\frac{B}{m}\right)^k\right]. \tag{7}$$

In contrast, the heuristic uses $(\mathbb{E}B/m)^k = p_1^k$. Conditional independence does not permit us to move the expectation inside the power. Arrays with more occupied bits make *all* query positions more likely to test positive, producing dependence after we average over the array.

Indeed, convexity of $u \mapsto u^k$ on $[0,1]$ gives

$$p_1^k = \left(\frac{\mathbb{E}B}{m}\right)^k \le \mathbb{E}\left[\left(\frac{B}{m}\right)^k\right].$$

The product is a *lower bound*, not a general finite upper bound. For a concrete example, let $n = 1, k = 2, m = 2$. The two insertion hashes coincide with probability $1/2$, giving $B = 1$; otherwise $B = 2$. Hence

$$\mathbb{P}(\text{false positive}) = \frac{1}{2}\left(\frac{1}{2}\right)^2 + \frac{1}{2}\cdot 1 = \frac{5}{8}, \qquad p_1^k = \left(\frac{3}{4}\right)^2 = \frac{9}{16}.$$

This small example isolates the gap without invoking any approximation to an exponential.

4.4 An exact occupancy calculation

The distribution of B gives an exact finite formula. Write $\left\{ {t \atop b} \right\}$ for a *Stirling number of the second kind*: the number of ways to partition t labelled objects into b nonempty, unlabelled blocks.

To have exactly b occupied positions, choose those positions in $\binom{m}{b}$ ways, partition the kn labelled insertion throws into b nonempty groups, and assign the groups to the chosen positions in $b!$ ways. All m^{kn} insertion outcomes are equally likely, so

$$\mathbb{P}(B = b) = \frac{\binom{m}{b}\, b!}{m^{kn}} \left\{ {kn \atop b} \right\}, \qquad 1 \le b \le \min(m, kn).$$

Multiplying by the conditional false-positive probability $(b/m)^k$ and summing yields

$$\mathbb{P}(\text{false positive}) = \frac{1}{m^{k(n+1)}} \sum_{b=1}^{\min(m,kn)} b^k \binom{m}{b}\, b! \left\{ {kn \atop b} \right\}. \tag{8}$$

This is rigorous, but less useful for understanding the parameter tradeoff than the simple heuristic. The Wikipedia discussion of Bloom-filter false positives displays both calculations; Bose et al. [7] give the correction and a detailed analysis.

A question for the next few lectures: concentration of measure. We will revisit this analysis after developing concentration inequalities. The goal is to prove that B/m is very unlikely to differ much from its mean p_1, and then use the *valid conditional calculation* in (7). For example, for any $\eta > 0$, splitting according to whether $B/m \le p_1 + \eta$ gives the rigorous bound

$$\mathbb{P}(\text{false positive}) \le \min\{1, p_1 + \eta\}^k + \mathbb{P}(B/m > p_1 + \eta).$$

Concentration will control the last term. In particular, with $m/n \to c > 0$ and fixed k, it will justify

$$\mathbb{P}(\text{false positive}) = (1 - e^{-k/c})^k + o(1) \qquad (n, m \to \infty).$$

Thus the later proof will recover the heuristic's asymptotic prediction and give rigorous bounds with an explicit correction. It will not make the independence assumption true or turn p_1^k itself into a finite upper bound.

4.5 Supplement: elementary rigorous bounds

The classroom analysis above motivates the later concentration tools. The following two arguments are optional: they give guarantees now using only elementary estimates.

A finite guarantee with a larger constant. There are at most kn occupied bits, regardless of how the throws land. Therefore, if

$$k = \lceil \log_2(1/\delta) \rceil, \qquad m = 2kn, \tag{9}$$

then (7) is at most $2^{-k} \le \delta$. This uses $O(n \log(1/\delta))$ array bits and $O(\log(1/\delta))$ hash evaluations per operation. It proves the desired space–error scaling without the sharper heuristic constant. The hashes are still ideal primitives, so this counts array space, not a succinct description of the complete random functions.

A second-moment justification of the asymptotic formula. A variance calculation already suffices when k is fixed. For $m \ge 2$, let Z_i indicate that position i is empty. For distinct i, j,

$$\mathbb{E}[Z_i Z_j] = (1 - 2/m)^{kn} \le (1 - 1/m)^{2kn} = \mathbb{E}Z_i \, \mathbb{E}Z_j.$$

Thus the empty-bit indicators have nonpositive pairwise covariance. Since $B = m - \sum_i Z_i$ and each indicator has variance at most $1/4$, we have $\mathrm{Var}(B) \le m/4$.

For $u, v \in [0, 1]$, factoring $u^k - v^k$ gives $|u^k - v^k| \le k|u - v|$. Applying this inequality and then Cauchy–Schwarz, we obtain

$$\left| \mathbb{E}\left[(B/m)^k \right] - p_1^k \right| \le k\mathbb{E}|B/m - p_1| \le \frac{k\sqrt{\mathrm{Var}(B)}}{m} \le \frac{k}{2\sqrt{m}}.$$

Together with the lower bound above, this proves

$$p_1^k \le \mathbb{P}(\text{false positive}) \le p_1^k + \frac{k}{2\sqrt{m}}.$$

For fixed k and $m/n \to c > 0$, the correction tends to zero and $p_1 \to 1 - e^{-k/c}$, proving (6) in this regime. This elementary argument needs no exponential tail inequality. Its additive error need not be small relative to a very small target δ, especially if k grows; the later concentration analysis will provide more precise control. For $m = 1$, the false-positive probability is simply one.

4.6 An explicit implementation with limited independence

A partitioned Bloom filter places each hash in its own row. This slightly changes the layout and admits a particularly simple proof with universal families.

Theorem 4.1 (Partitioned filter)**.** *Let $n \geq 1$ and $0 < \delta \leq 1/2$. Use $k = \lceil \log_2(1/\delta) \rceil$ independent universal hashes to $b = 2n$ slots, one b-bit row per hash. Insert a key into every row and answer yes precisely when every row tests positive. Then there are no false negatives, and any fixed nonmember has false-positive probability at most δ. Total space, including seeds, is*

$$O\big(n \log(1/\delta) + \log N \log(1/\delta)\big) \quad \textit{bits}.$$

Proof. In row j, a fixed nonmember can test positive only by colliding with a stored key. The union bound gives probability at most $n/b = 1/2$. Rows use independent seeds, so the probability that all rows test positive is at most 2^{-k}. The arrays use $bk = 2nk$ bits; the universal construction of Section 2 uses $O(\log N)$ seed bits per row. □

Insertions and queries take $O(k)$ hash evaluations and bit accesses. The empty set can be represented by one flag. The guarantee presumes the filter was sized for an upper bound on the number of distinct inserted keys. Clearing a bit to delete a key can erase evidence for another key, so ordinary Bloom filters do not support deletion by simply reversing insertion.

The construction separates two roles of randomness: collision bounds within a row suffice, while independent row seeds provide exponential error reduction.

Merging filters. Filters built with identical array dimensions and hash seeds combine by bitwise OR. The result is the filter obtained by inserting the union of their sets. The dimensions must be chosen for that union's size. This rule applies to both array layouts above.

5 Frequency moments and distinct counting

5.1 The streaming model and frequency moments

We now change the input model. A sequence $x_1, \ldots, x_L$ of keys from $\mathcal{U}$ arrives one item at a time; repeated keys are allowed. An *insertion stream* records one additional occurrence at each arrival. Define the frequency of a key, the set of observed keys, and its size by

$$f_x = |\{i \in [L] : x_i = x\}|, \qquad D = \{x \in \mathcal{U} : f_x > 0\}, \qquad z = |D|.$$

Thus $L = \sum_{x \in \mathcal{U}} f_x$ is the stream length, whereas z counts distinct keys.

For $p > 0$, the *pth frequency moment* is

$$F_p = \sum_{x \in \mathcal{U}} f_x^p.$$

We define $F_0 = |D| = z$ separately, without using 0^0. On insertion streams, $F_1 = L$ is easy to count exactly. The second moment F_2 measures repetition: it equals L when all items are distinct and L^2 when all items are equal. For example, the stream (a, b, a, c, b, a) has $L = 6$, frequencies $3, 2, 1$, distinct count $F_0 = z = 3$, and $F_2 = 3^2 + 2^2 + 1^2 = 14$.

A one-pass algorithm processes each item as it arrives, retaining a small state called a *sketch*, and estimates the desired quantity from that state. Space is the number of bits retained; the stream itself is not stored for later access. Throughout the streaming sections, the stream is fixed independently of the random seeds. The guarantees concern a fixed completed stream, not inputs chosen adaptively after observing the sketch.

We first estimate F_0, which ignores multiplicity. Section 6 then estimates an individual coordinate f_x of the frequency vector. The broader results on other moments are discussed after the distinct-counting algorithms.

5.2 Accuracy, confidence, and the space requirement

An (ε, δ)-estimate of the distinct count satisfies

$$\mathbb{P}\big((1-\varepsilon)z \le \widehat{z} \le (1+\varepsilon)z\big) \ge 1-\delta.$$

Here $0 < \varepsilon < 1$ controls relative accuracy, and $0 < \delta \le 1/2$ bounds the failure probability. These parameters have different roles: a more accurate estimate has a smaller error interval, while a more confident estimate belongs to that interval with higher probability. Empty streams return zero, using a separate flag when needed. Repeated keys must not be treated as fresh independent samples.

An exact dictionary of the distinct keys uses $O(z \log N)$ bits; an N-bit bitmap is another option. A randomized sketch can use far less space when only an estimate is needed. This improvement is not available to deterministic streaming algorithms with a fixed small relative error.

Proposition 5.1 (A deterministic obstruction)**.** *Any deterministic one-pass algorithm that always estimates the distinct count within relative error* $1/10$ *requires* $\Omega(N)$ *bits in the worst case.*

Proof. For clarity take N divisible by 16 and put $s = N/8$. There exists a family $\mathcal{F}$ of $\lfloor 2^{N/64} \rfloor$ sets of size s, any two intersecting in fewer than $s/2$ elements. To see this, fix one s-set S and choose another uniformly. For each prescribed $s/2$-subset of S, its probability of being contained in the new set is at most $(s/N)^{s/2}$. A union bound gives

$$\mathbb{P}(|S \cap T| \ge s/2) \le \binom{s}{s/2} (s/N)^{s/2} \le 2^s 8^{-s/2} = 2^{-N/16}.$$

Choose $\lfloor 2^{N/64} \rfloor$ sets independently and union-bound over their pairs; the total probability of a bad pair is less than one.

If the algorithm has fewer than $\log_2 |\mathcal{F}|$ state bits, two sets $S, T \in \mathcal{F}$, presented in a fixed order, produce the same state. Append the same list of all keys in S to both streams. Determinism forces the same final output, but their distinct counts are s and $|S \cup T| > 3s/2$. The required intervals are disjoint, since $1.1s < 0.9(3s/2)$. This is a contradiction. For general N, restrict to a subuniverse of size $16\lfloor N/16 \rfloor$. □

This is a simple form of the deterministic lower-bound phenomenon in the work of Alon, Matias, and Szegedy [9]. It explains why randomization is central, even though the desired answer is just one number.

5.3 Minimum sketch: a statistical idea under ideal hashing

First assume an ideal hash function $h : \mathcal{U} \to (0,1]$ assigns independent continuous uniform values to distinct keys. For a nonempty stream, maintain

$$Y = \min_{1 \le i \le L} h(x_i) = \min_{x \in D} h(x).$$

There are z independent values in this minimum. For $0 \le t \le 1$,

$$\mathbb{P}(Y > t) = (1-t)^z.$$

Integrating the tail and its first moment (Appendix B) yields

$$\begin{aligned} \mu := \mathbb{E}Y &= \int_0^1 (1-t)^z \, dt = \frac{1}{z+1}, \\ \mathbb{E}Y^2 &= \int_0^1 2t(1-t)^z \, dt = \frac{2}{(z+1)(z+2)}, \\ \mathrm{Var}(Y) &= \frac{z}{(z+1)^2(z+2)} \le \mu^2. \end{aligned} \tag{10}$$

The minimum is typically on the order of $1/z$, suggesting inversion. But $\mathbb{E}(1/Y) = \infty$: the density $z(1-t)^{z-1}$ is bounded below by a positive constant near zero, and $\int_0^\eta dt/t$ diverges. In particular, taking a reciprocal of the mean identity does not produce an unbiased estimator.

The useful move is to *average independent minima before inversion*.

Minimum sketch with the mean trick (ideal hashes)

Choose $k = \lceil 9/(\varepsilon^2\delta) \rceil$ independent ideal hash functions. Maintain $Y_j = \min_i h_j(x_i)$ for every $j \in [k]$. For a nonempty stream, return

$$\widehat{z} = \frac{1}{\overline{Y}} - 1, \qquad \overline{Y} = \frac{1}{k} \sum_{j=1}^{k} Y_j.$$

Theorem 5.2. *The ideal minimum sketch is an (ε, δ)-estimator.*

Proof. Independence across the k functions gives $\mathbb{E}\overline{Y} = \mu$ and $\mathrm{Var}(\overline{Y}) \le \mu^2/k$. Chebyshev's inequality shows

$$\mathbb{P}\big(|\overline{Y} - \mu| > (\varepsilon/3)\mu\big) \le \frac{9}{k\varepsilon^2} \le \delta.$$

On the complementary event,

$$|\widehat{z} - z| = \left| \frac{1}{\overline{Y}} - \frac{1}{\mu} \right| \le \frac{(\varepsilon/3)(z+1)}{1 - \varepsilon/3} \le \varepsilon z.$$

The last step uses $z + 1 \le 2z$ and $1 - \varepsilon/3 \ge 2/3$. □

The factor $z+1$ in this conversion matters even when z is small. The proof is about $1/\overline{Y} - 1$, not the average of $1/Y_j - 1$.

First use $k = \lceil 36/\varepsilon^2 \rceil$ minima to obtain failure probability $1/4$. Then take the median of $O(\log(1/\delta))$ independent estimates. The resulting sketch has $O(\varepsilon^{-2}\log(1/\delta))$ real registers; Lemma B.3 gives the confidence calculation. This remains an *idealized* space count: arbitrary real numbers and fully independent random functions do not constitute a finite-bit implementation.

Merging minimum sketches. When merging, ignore empty pieces and return zero if all pieces are empty. With the same hash functions and number of registers, take the coordinatewise minimum of sketches of separate stream pieces. This produces exactly the registers of their concatenation, including when keys occur in several pieces. The error guarantee is then applied to that combined stream.

The next algorithms obtain actual bit-space bounds.

5.4 A trailing-zero sketch with pairwise independence

A small register can record the rarest hash pattern seen so far. The following simplified Flajolet–Martin-style estimator is the pairwise-independent variant analysed by Alon, Matias, and Szegedy [8, 9]. It is not the full bitmap estimator in the original Flajolet–Martin paper.

Let $w = \lceil \log_2 N \rceil$ and choose a pairwise-independent uniform hash to the w-bit strings, using the affine field construction. For a nonzero string y, let $\mathrm{tz}(y)$ be its number of trailing zero bits; set

$$\mathrm{tz}(0) = w.$$

This convention makes every value finite. For every integer $0 \le r \le w$, precisely 2^{w-r} strings have at least r trailing zeros.

Maximum trailing-zero register

For a nonempty stream, maintain

$$R = \max_i \mathrm{tz}(h(x_i)), \qquad \text{and report } \widehat{z} = 2^R.$$

Use the same hash function on every occurrence of a key. For an empty stream, report zero.

Theorem 5.3 (Constant-factor distinct counting). *For every $C \ge 1$ and nonempty fixed stream,*

$$\mathbb{P}\left(\widehat{z} < z/C \text{ or } \widehat{z} > Cz\right) \le 3/C.$$

The full sketch uses $O(\log N)$ bits, including the seed.

Proof. For $0 \le r \le w$ define the threshold count

$$T_r = \sum_{x \in D} \mathbf{1}\{\mathrm{tz}(h(x)) \ge r\}.$$

Uniformity gives $\mathbb{E}T_r = z2^{-r}$. Pairwise independence gives

$$\mathrm{Var}(T_r) = z2^{-r}(1 - 2^{-r}) \le z2^{-r}, \qquad R \ge r \iff T_r \ge 1. \tag{11}$$

For the upper tail, if $Cz \ge 2^w$ the event $\widehat{z} > Cz$ is impossible. Otherwise choose $r = \lceil \log_2(Cz) \rceil \le w$. Markov's inequality gives

$$\mathbb{P}(\widehat{z} > Cz) \le \mathbb{P}(T_r \ge 1) \le z2^{-r} \le 1/C.$$

For the lower tail, if $z/C \le 1$ the event $\widehat{z} < z/C$ is impossible. Otherwise $r = \lceil \log_2(z/C) \rceil$ lies in $\{1, \dots, w\}$, and Chebyshev's inequality gives

$$\mathbb{P}(\widehat{z} < z/C) \le \mathbb{P}(T_r = 0) \le \frac{\mathrm{Var}(T_r)}{(\mathbb{E}T_r)^2} \le \frac{2^r}{z} \le \frac{2}{C}.$$

The union bound proves the probability estimate. The register $R \in \{0, \dots, w\}$ needs only $\lceil \log_2(w+1) \rceil$ bits. However, the affine seed has $2w$ bits; a field description, if stored, takes another $O(w)$ bits. Hence the full space bound is $O(\log N)$. □

Taking $C = 12$ gives failure probability at most $1/4$. The median of independent copies raises confidence to $1 - \delta$ using $O(\log N \log(1/\delta))$ bits. It retains the factor 12: confidence amplification does not turn a constant-factor estimate into an arbitrarily accurate relative estimate.

Merging trailing-zero registers. Ignore empty pieces, tracking emptiness with their flags. For nonempty pieces using the same hash seed, take the maximum of their registers. It equals the register of the combined stream. With several independent copies, merge corresponding registers separately.

The next construction addresses relative accuracy.

5.5 Bottom-k: a relative estimate from an order statistic

The smallest few hash values contain more information than a single extreme value. The bottom-k or k-minimum-values method is associated with Bar-Yossef, Jayram, Kumar, Sivakumar, and Trevisan [10]. We analyse a finite-range version of the BJKST method.

For one trial, choose

$$k = \left\lceil \frac{64}{\varepsilon^2} \right\rceil, \qquad M = \text{the smallest power of two at least } 4N^2. \tag{12}$$

Choose a pairwise-independent uniform hash $h : \mathcal{U} \to \{1, \dots, M\}$. Concretely, use an affine map in $\mathbb{F}_M$, interpret its output bits as an integer from 0 to $M-1$, and add one. This gives a positive denominator and uses $O(\log N)$ seed bits.

Bottom-k trial with a finite hash range

Maintain a set K of at most k *distinct hash values*, initially empty.

1. On item x, insert $h(x)$ into K if absent.
2. If $|K| = k+1$, delete its largest value.
3. At the end, if $|K| < k$, return $|K|$. Otherwise set $T = \max K$ and return kM/T.

For failure probability δ, run an odd number $t \geq 8 \ln(1/\delta)$ of independent trials and return the median of their answers.

At every time, K is the set of the k smallest distinct hash values seen, or all of them if fewer exist. A discarded value cannot become relevant later: once K is full, its largest retained value can only decrease. Repeated occurrences of a key consequently cause no overcounting.

Distinct *keys* can still have equal hashes. Let $\mathcal{E}$ be the event that all z distinct keys have different hashes. The collision bound gives

$$\mathbb{P}(\mathcal{E}^c) \leq \frac{\binom{z}{2}}{M} \leq \frac{1}{8}. \tag{13}$$

On $\mathcal{E}$, the branch $z < k$ returns z exactly. For $z \geq k$, the following lemma analyses the order statistic. The lemma is applied in the original probability space, without conditioning on $\mathcal{E}$.

Lemma 5.4 (Finite-range order statistic)**.** *Let $H_1, \dots, H_z$ be pairwise-independent uniform values in $\{1, \dots, M\}$, where $M \geq z \geq k$ and $k \geq 2/\varepsilon$. Let Y_k be their kth smallest value,* counting equal numeric values with multiplicity. *Then*

$$\mathbb{P}\left(\left|\frac{kM}{Y_k} - z\right| > \varepsilon z\right) \leq \frac{6}{k\varepsilon^2}.$$

Proof. An unusually small Y_k makes the estimate too large. Put $a = kM/((1+\varepsilon)z)$ and count

$$V = \sum_{i=1}^{z} \mathbf{1}\{H_i < a\}.$$

The event $kM/Y_k > (1+\varepsilon)z$ implies $V \geq k$. Its mean satisfies $\mu_V \leq za/M = k/(1+\varepsilon)$ and its variance is at most its mean. Thus

$$\mathbb{P}(V \geq k) \leq \frac{\mu_V}{(k-\mu_V)^2} \leq \frac{1+\varepsilon}{k\varepsilon^2} \leq \frac{2}{k\varepsilon^2}.$$

For an underestimate put $b = kM/((1-\varepsilon)z)$. If $b \geq M$, the required event $Y_k > b$ is impossible. Otherwise define $W = \sum_i \mathbf{1}\{H_i \leq b\}$. The event $Y_k > b$ implies $W < k$. Here rounding must be accounted for:

$$\frac{k}{1-\varepsilon} - 1 \leq \frac{k}{1-\varepsilon} - \frac{z}{M} \leq \mu_W = \frac{z\lfloor b \rfloor}{M} \leq \frac{k}{1-\varepsilon}.$$

Since $k \geq 2/\varepsilon$,

$$\mu_W - k \geq \frac{k\varepsilon}{1-\varepsilon} - 1 \geq \frac{k\varepsilon}{2(1-\varepsilon)}.$$

Using $\mathrm{Var}(W) \leq \mu_W$ and Chebyshev once more,

$$\mathbb{P}(W < k) \leq \frac{\mu_W}{(\mu_W - k)^2} \leq \frac{4(1-\varepsilon)}{k\varepsilon^2} \leq \frac{4}{k\varepsilon^2}.$$

Adding the two tails proves the lemma. □

Theorem 5.5 (Finite-bit bottom-k sketch)**.** *With the parameters in* (12)*, one trial fails to give relative error ε with probability at most $1/4$. The median construction is an (ε, δ)-estimator using*

$$O\big(\varepsilon^{-2} \log N \log(1/\delta)\big) \quad \textit{bits},$$

including hash seeds.

Proof. If $z < k$, failure is possible only on $\mathcal{E}^c$. If $z \geq k$, on $\mathcal{E}$ the maintained threshold T equals the order statistic in Lemma 5.4. A failure of the implemented algorithm is therefore contained in the union of $\mathcal{E}^c$ and the bad-order-statistic event. Its probability is at most

$$\frac{1}{8} + \frac{6}{k\varepsilon^2} \leq \frac{1}{8} + \frac{6}{64} < \frac{1}{4}.$$

This union bound does not require independence between the two events. In particular, we never assert that the hash values remain pairwise independent after conditioning on no collisions.

Lemma B.3 gives failure at most $e^{-t/8} \leq \delta$ after the independent trials. A trial stores at most k values of $\log_2 M = O(\log N)$ bits, plus an $O(\log N)$-bit seed and metadata. If $k > N$, we may instead keep an exact dictionary of all distinct keys, using at most $N < k$ key words. This also respects the asserted space bound. □

A balanced search tree for the retained values uses $O(\log k)$ comparisons per update per trial; it also suppresses duplicate hash values. Thus a straightforward implementation needs $O(t \log k)$ word operations per item, under the stated hash-evaluation model. One trial suffices at constant confidence, giving an $O(\varepsilon^{-2} \log N)$-bit bound.

The larger finite range and the confidence amplification serve different purposes. A large range limits collisions among distinct keys; independent trials reduce overall failure to arbitrary δ. Keeping a fixed range such as N^3 while ignoring its collision probability would not justify arbitrarily small error probabilities.

Merging bottom-k sketches. An empty piece has an empty retained set. Using the same hash function and value of k, retain the k smallest distinct values in the union of the retained sets. Any value omitted from a piece already has k smaller values in that piece, so it cannot belong to the global bottom-k. Thus merging gives the same state as processing the concatenated stream. Independent trials are merged one corresponding pair at a time.

5.6 Further results on frequency moments

The frequency-moment program of Alon, Matias, and Szegedy [9] established small-space randomized estimation of F_0 and F_2. For constant accuracy and confidence, F_0, F_1, F_2 admit space polynomial in $\log N + \log(L+1)$. For fixed $p > 2$ and streams of polynomial length in N, the classical space scale is $\widetilde{\Theta}(N^{1-2/p})$, where the tilde suppresses logarithmic factors; Indyk and Woodruff [11] give the matching upper-bound exponent. These results place distinct counting within the broader study of frequency moments. The algorithms above do not require an F_2 or higher-moment estimator.

For distinct counting alone, Kane, Nelson, and Woodruff [12] achieve $O(\varepsilon^{-2} + \log N)$ bits at constant confidence, improving the product $\varepsilon^{-2} \log N$ of the elementary bottom-k construction. The optimality statement is understood in the usual nontrivial accuracy regime; an exact N-bit bitmap is always available. These sharper algorithms are further reading.

6 Frequency estimation: the Count-Min sketch

Return to the frequency vector of the insertion stream in Section 5. Distinct counting asks how many coordinates are nonzero. A frequency query instead specifies a key x and asks for its coordinate f_x. Fix $0 < \varepsilon < 1$ and $0 < \delta \leq 1/2$. For a stream and query fixed independently of the seeds, the goal here is *additive* error:

$$\mathbb{P}\big(|\widehat{f}_x - f_x| > \varepsilon L\big) \leq \delta.$$

This is useful for identifying items with large frequency, but is weaker than a relative-error estimate of every f_x.

A hashed bucket can count every item sent to it. Its count includes all occurrences of the queried key, together with noise from collisions. Several independently hashed rows offer several such overestimates; taking their minimum reduces the noise. The Count-Min sketch of Cormode and Muthukrishnan [13] uses

$$b = \lceil e/\varepsilon \rceil, \qquad d = \lceil \ln(1/\delta) \rceil$$

and d independently sampled universal hash functions $h_j : \mathcal{U} \to [b]$. Independence is required across row seeds; collision universality suffices within a row.

Count-Min sketch for an insertion stream

Initialize a d-by-b array of counters A to zero.

- **Update on item y:** for every row j, increment $A[j, h_j(y)]$.
- **Query key x:** return $\widehat{f}_x = \min_{j \in [d]} A[j, h_j(x)]$.

A bucket counter includes every occurrence of the queried key, plus possible collisions from other keys. Thus it is always an overestimate; taking the minimum preserves that property.

Theorem 6.1 (Count-Min point-query guarantee)**.** *For any fixed insertion stream and fixed query x, independently of the seeds,*

$$\mathbb{P}\big(f_x \le \widehat{f}_x \le f_x + \varepsilon L\big) \ge 1 - \delta.$$

The sketch uses

$$O\left(\varepsilon^{-1} \log(1/\delta) \log(L+1) + \log(1/\delta) \log N\right) \quad \textit{bits} \tag{14}$$

for a nonempty stream, including the seeds. Updates and queries each use $O(\log(1/\delta))$ hash evaluations and counter accesses.

Proof. The statement is immediate for $L = 0$, since all counters are zero. Assume $L > 0$. For a fixed row j,

$$A[j, h_j(x)] = f_x + Z_j, \qquad Z_j = \sum_{y \ne x} f_y \, \mathbf{1}\{h_j(y) = h_j(x)\} \ge 0.$$

Universality and linearity of expectation give

$$\mathbb{E} Z_j \le \frac{1}{b} \sum_{y \ne x} f_y = \frac{L - f_x}{b} \le \frac{L}{b}.$$

The collision indicators need not be independent. By Markov's inequality,

$$\mathbb{P}(Z_j > \varepsilon L) \le \frac{1}{\varepsilon b} \le e^{-1}.$$

The entire stream is fixed, so Z_j depends only on the seed of row j. The Z_j are therefore independent across rows. Consequently,

$$\mathbb{P}(\widehat{f}_x > f_x + \varepsilon L) = \mathbb{P}\left(\bigcap_{j=1}^{d} \{Z_j > \varepsilon L\}\right) \le e^{-d} \le \delta.$$

Every counter is between zero and L, requiring $\lceil \log_2(L+1) \rceil$ bits. There are bd counters, plus d seeds of $O(\log N)$ bits each. This proves (14). Each operation touches one counter per row. □

The bit bound uses counters large enough for the processed stream, or a supplied upper bound on its length. Standard resizing can accommodate an unknown final length. The operation bound assumes words can hold a key, a counter, and an array address. Unlike the number of counters, their bit cost depends on the stream length.

6.1 Point queries, simultaneous guarantees, and heavy hitters

The theorem controls one fixed queried key. For a predetermined set of Q candidate keys, build with failure parameter δ/Q and apply a union bound. With probability at least $1-\delta$, all their estimates are accurate. In particular, depth $d = \lceil \ln(N/\delta) \rceil$ makes the guarantee simultaneous for every key in the universe at the final stream state. This also covers a key selected after looking at that state, provided the stream itself remains fixed independently of the seeds.

An item with $f_x > \phi L$ is a *heavy hitter* at threshold ϕ. On the simultaneous success event, accepting candidates whose estimates exceed ϕL includes every heavy hitter and excludes any candidate with $f_x \leq (\phi - \varepsilon)L$. Items in the intervening band may be reported. A point-query sketch does not itself list the candidate keys: one must supply candidates, enumerate the universe, or use an additional recovery algorithm. This distinction is essential when the universe is very large.

6.2 Merging sketches and extending the update model

If separate stream pieces use identical Count-Min parameters and hash seeds, add their arrays entrywise. Each resulting bucket counts exactly the updates that would reach it in the concatenated stream. Use the combined stream length in the additive error bound, and enough counter bits to hold the combined counts.

The same seed is needed when merging corresponding summaries because equal keys must use equal buckets. Independent seeds have a different purpose: they create the repetitions used for confidence amplification. A sketch with several independent rows can therefore be merged by adding corresponding rows that share their seeds across the pieces.

The proof uses nonnegative final frequencies. It extends to weighted nonnegative updates, replacing L by total mass and choosing an appropriate counter representation. If updates include deletions but all current frequencies remain nonnegative (the strict turnstile model), the same algebra applies at each fixed valid state. With arbitrary signed frequencies, collision noise can be negative, and the minimum need not be an upper bound.

7 Exercises

The exercises reinforce the algorithms and their mathematical assumptions.

1. **Birthday estimates.** Use (2) to show that if $n/\sqrt{m} \to c > 0$, then the collision probability tends to $1 - e^{-c^2/2}$. Explain why $n = o(m)$ alone does not guarantee a relative approximation of the no-collision probability by $e^{-n(n-1)/(2m)}$.

2. **Pairwise independence is weaker.** For $h(x) = ax + b$ over $\mathbb{F}_p$ with a, b uniform, fix any set of $2 \leq n \leq p$ keys. Show that some collision occurs if and only if $a = 0$, and hence with probability $1/p$. Compare this with the independent birthday model.

3. **Why the seed distribution matters.** Let a, b range uniformly over a field. Explain why excluding $a = 0$ destroys pairwise independence, and why reducing outputs modulo an arbitrary m may destroy uniform marginals.

4. **FKS constants.** Replace the primary acceptance threshold $4n$ by αn, where $\alpha > 2$. Bound

the expected number of primary trials. Explain separately which bounds concern random construction and which concern queries after construction.

5. **Bloom dependence.** For $n = 1, k = 2, m = 2$, enumerate the possible occupied-bit counts and derive the exact false-positive probability $5/8$. Where does the value $9/16$ come from?

6. **Bloom parameters.** Differentiate $\ln[(1 - e^{-k/c})^k]$ to find its optimum. Explain why the optimizing real k must be converted to an integer in an implementation.

7. **Reciprocals and expectations.** Prove directly that $\mathbb{E}(1/Y) = \infty$ for the ideal minimum. Then show that the mean-sketch proof can retain the smaller choice $k = \lceil 4/(\varepsilon^2\delta) \rceil$ by using the exact variance in (10) and the exact reciprocal thresholds, rather than the simpler $\varepsilon/3$ event.

8. **Register versus seed space.** Explain why a trailing-zero register has only $O(\log \log N)$ bits but the analysed full sketch has $O(\log N)$ bits. Why does setting $\mathrm{tz}(0) = \infty$ invalidate the stated finite-valued estimator and its guarantee?

9. **Bottom-k with ties.** Modify the sketch to store records $(h(x), x)$, ordered lexicographically, with only one record per key. Show that Lemma 5.4 then applies even when distinct keys have equal hashes. Deduce that a power-of-two range $M \geq N$ suffices without a separate collision event.

10. **Amplification.** Why does the median preserve a relative-error interval while the minimum does not? Why is taking a minimum appropriate for Count-Min instead?

11. **Several queries.** Derive the Count-Min depth for Q fixed candidate queries. Explain why a bound for one fixed query is insufficient to justify reporting all heavy hitters.

12. **Mergeability.** Prove the bottom-k union rule. Give an example showing why sketches built with unrelated hash seeds cannot in general be merged by the stated coordinatewise operations.

A Independence and limited independence

For random variables $X_1, \ldots, X_s$, independence means that for every choice of sets $A_1, \ldots, A_s$,

$$\mathbb{P}(X_1 \in A_1, \ldots, X_s \in A_s) = \prod_{i=1}^{s} \mathbb{P}(X_i \in A_i).$$

For finite random variables it suffices to check singleton sets $A_i = \{a_i\}$. For continuous variables, singleton probabilities do not characterize independence. The variables are *k-wise independent* if every subcollection of at most k variables is independent. This property does not require the variables to have the same distribution or to be uniform.

Applying a deterministic function to each variable preserves independence: the event $g_i(X_i) \in B_i$ is the event $X_i \in g_i^{-1}(B_i)$. The same argument preserves k-wise independence. If X_i is uniform on a finite set A and $g_i : A \to B$ has equally sized fibers, then $g_i(X_i)$ is uniform on B as well. This is the balanced-projection principle used for binary field representations.

Many pairwise independent bits from a short seed. Let R be uniform on $\mathbb{F}_2^r$. For each nonzero vector $a \in \mathbb{F}_2^r$, set

$$X_a = a \cdot R = \sum_{i=1}^{r} a_i R_i \pmod 2.$$

Each X_a is a fair bit: after fixing all seed coordinates except one with $a_i = 1$, changing that coordinate changes the output. For distinct nonzero a, b, the two vectors are linearly independent over $\mathbb{F}_2$, since its only nonzero scalar is 1. The linear map $R \mapsto (a \cdot R, b \cdot R)$ has rank two. Each output pair has 2^{r-2} preimages, and therefore probability $1/4$.

These $2^r - 1$ bits are pairwise independent, but for $r \geq 2$ they are not mutually independent:

$$X_a + X_b = X_{a+b} \pmod 2 \qquad (a \neq b).$$

The smallest example is $R_1, R_2, R_1 \oplus R_2$. Any two are independent, while the third is determined by the first two. The construction saves seed length precisely because it does not supply full independence; it creates no additional entropy.

Independence has two roles in this lecture. Outputs at distinct inputs may need only pairwise independence. Independent *repetitions*, however, use separately sampled seeds. A pairwise-independent family of outputs does not make arbitrary events depending on the whole seed independent, and pairwise-independent repetitions do not justify an all-fail probability of p^t.

B Expectations and deviation bounds

B.1 Indicators, conditioning, and first moments

For an event A, its indicator $\mathbf{1}\{A\}$ equals 1 on A and 0 otherwise, so $\mathbb{E}\mathbf{1}\{A\} = \mathbb{P}(A)$. Linearity,

$$\mathbb{E}\left[\sum_i c_i X_i\right] = \sum_i c_i \mathbb{E} X_i,$$

requires no independence. The pointwise inequality $\mathbf{1}\{\bigcup_i A_i\} \leq \sum_i \mathbf{1}\{A_i\}$ gives the *union bound* $\mathbb{P}(\bigcup_i A_i) \leq \sum_i \mathbb{P}(A_i)$, also without independence. All expectations below are assumed finite when used in algebraic identities.

Conditioning on a finite random variable Y gives

$$\mathbb{P}(A) = \sum_y \mathbb{P}(Y = y)\mathbb{P}(A \mid Y = y), \qquad \mathbb{E}X = \sum_y \mathbb{P}(Y = y)\mathbb{E}[X \mid Y = y].$$

Terms of probability zero may be omitted. Repeated use of the definition of conditional probability gives the chain rule

$$\mathbb{P}\left(\bigcap_{i=1}^{t} A_i\right) = \mathbb{P}(A_1) \prod_{i=2}^{t} \mathbb{P}\left(A_i \mid \bigcap_{j<i} A_j\right)$$

when these conditioning events have positive probability. If a prefix event has probability zero, the whole intersection has probability zero.

Lemma B.1 (Markov's inequality). *For $X \geq 0$ and $a > 0$, $\mathbb{P}(X \geq a) \leq \mathbb{E}X/a$.*

Proof. Take expectations in $a\mathbf{1}\{X \geq a\} \leq X$. □

B.2 Variance and the mean trick

For variables with finite second moments, define

$$\mathrm{Var}(X) = \mathbb{E}[(X - \mathbb{E}X)^2] = \mathbb{E}[X^2] - (\mathbb{E}X)^2, \qquad \mathrm{Cov}(X, Y) = \mathbb{E}[XY] - \mathbb{E}X\mathbb{E}Y.$$

Expanding the square gives

$$\mathrm{Var}\left(\sum_i X_i\right) = \sum_i \mathrm{Var}(X_i) + 2\sum_{i<j} \mathrm{Cov}(X_i, X_j).$$

Independence gives $\mathbb{E}[XY] = \mathbb{E}X\mathbb{E}Y$, hence zero covariance. The converse fails: for X uniform on $\{-1, 0, 1\}$, X and X^2 have zero covariance but are dependent. For pairwise-independent variables the variance of a sum is the sum of the variances.

Lemma B.2 (Chebyshev's inequality)**.** *For $a > 0$, $\mathbb{P}(|X - \mathbb{E}X| \geq a) \leq \mathrm{Var}(X)/a^2$.*

Proof. Apply Markov's inequality to the nonnegative variable $(X - \mathbb{E}X)^2$. □

For pairwise-independent indicator variables B_i with means p_i, let $S = \sum_i B_i$ and $\mu = \mathbb{E}S$. Then

$$\mathrm{Var}(S) = \sum_i p_i(1 - p_i) \leq \mu, \qquad \mathbb{P}(S = 0) \leq \frac{1}{\mu} \quad (\mu > 0).$$

The second bound follows because $S = 0$ implies $|S - \mu| \geq \mu$. Such a sum need not have a binomial distribution.

For pairwise-independent variables $Y_1, \ldots, Y_r$ with common mean μ and variances at most v, their average satisfies

$$\mathbb{E}\overline{Y} = \mu, \qquad \mathrm{Var}(\overline{Y}) \leq \frac{v}{r}, \qquad \mathbb{P}(|\overline{Y} - \mu| \geq \eta\mu) \leq \frac{v}{r\eta^2\mu^2} \quad (\eta, \mu > 0).$$

This is the *mean trick*: averaging reduces variance by the number of independent copies. Accuracy for a nonlinear function of the mean must still be established separately.

B.3 Two expectation inequalities

Nonnegativity of $\mathrm{Var}(|X|)$ gives $(\mathbb{E}|X|)^2 \leq \mathbb{E}[X^2]$, hence $\mathbb{E}|X| \leq \sqrt{\mathbb{E}[X^2]}$. This is the special case of Cauchy–Schwarz used to convert a variance bound into a bound on expected absolute deviation.

If $U \geq 0$ and k is a positive integer, then $\mathbb{E}[U^k] \geq (\mathbb{E}U)^k$ whenever these quantities are finite. For $k = 1$ this is equality. For $k \geq 2$, the increasing derivative of u^k gives $u^k \geq a^k + ka^{k-1}(u - a)$ for all $u, a \geq 0$. Set $a = \mathbb{E}U$ and take expectations. This is the instance of Jensen's inequality used for Bloom filters.

B.4 Moments from tails

For a nonnegative random variable W,

$$\mathbb{E}W = \int_0^\infty \mathbb{P}(W > u)\, du, \qquad \mathbb{E}[W^2] = \int_0^\infty 2u\mathbb{P}(W > u)\, du.$$

Indeed, pointwise $W = \int_0^\infty \mathbf{1}\{W > u\}\,du$ and $W^2 = \int_0^\infty 2u\mathbf{1}\{W > u\}\,du$. Interchanging expectation and a nonnegative integral proves both identities, even when the value is infinite. In particular, when $W \in [0,1]$ and $\mathbb{P}(W > u) = (1-u)^z$ for $0 \le u \le 1$,

$$\mathbb{E}W = \frac{1}{z+1}, \qquad \mathbb{E}[W^2] = \frac{2}{(z+1)(z+2)}.$$

The second integral follows by writing $u = 1-(1-u)$ and integrating two powers of $1-u$.

B.5 Independent repetition and medians

Independent events of probabilities at most p all occur with probability at most p^t. A repeated trial succeeding independently with probability $q > 0$ has trial count T satisfying $\mathbb{P}(T > r) = (1-q)^r$ for integers $r \ge 0$, and therefore $\mathbb{E}T = \sum_{r\ge 0}(1-q)^r = 1/q$. For two-sided error, a median supplies the following different guarantee.

Lemma B.3 (Median amplification)**.** *Let t be odd. Suppose $Z_1, \dots, Z_t$ are mutually independent and each belongs to an interval I with probability at least $3/4$. Then*

$$\mathbb{P}\big(\text{median}(Z_1, \dots, Z_t) \notin I\big) \le e^{-t/8}.$$

Thus any odd $t \ge 8\ln(1/\delta)$ makes this probability at most δ.

Proof. Let $B_i = \mathbf{1}\{Z_i \notin I\}$. A median outside I requires at least $(t+1)/2$ failures. Since the B_i are independent and $\mathbb{E}B_i \le 1/4$, Markov's inequality gives

$$\mathbb{P}\left(\sum_i B_i \ge t/2\right) \le 3^{-t/2}\mathbb{E}\left[3^{\sum_i B_i}\right] = 3^{-t/2}\prod_i (1 + 2\mathbb{E}B_i) \le \left(\frac{\sqrt{3}}{2}\right)^t.$$

Finally $(\sqrt{3}/2)^t = (1-1/4)^{t/2} \le e^{-t/8}$, using $1 - x \le e^{-x}$. □

B.6 An elementary logarithm bound

For $0 \le x \le \rho < 1$,

$$-x - \frac{x^2}{2(1-\rho)} \le \ln(1-x) \le -x.$$

To see both sides at once, integrate:

$$0 \le -\ln(1-x) - x = \int_0^x \frac{u}{1-u}\,du \le \frac{1}{1-\rho}\int_0^x u\,du = \frac{x^2}{2(1-\rho)}.$$

The upper bound also yields $1 - x \le e^{-x}$. Applying the logarithm inequality to each factor of a product gives explicit error bounds for exponential approximations.

Bibliographic notes

Universal hashing is due to Carter and Wegman [1], and the two-level static dictionary to Fredman, Komlós, and Szemerédi [2]. For dynamic perfect hashing, see Dietzfelbinger et al. [3]; succinct-dictionary tradeoffs are developed in [4, 5].

Bloom introduced the approximate membership structure [6]. Bose et al. [7] clarify why multiplying marginal bit-occupancy probabilities does not give its exact finite-size false-positive probability.

Flajolet and Martin developed probabilistic counting using rare hash patterns [8]. The maximum-trailing-zero estimator here is the pairwise-independent variant in Proposition 2.3 of Alon, Matias, and Szegedy [9], rather than the original bitmap estimator. Bottom-k is associated with Bar-Yossef et al. [10], and Count-Min with Cormode and Muthukrishnan [13]. For sharper space bounds, see Indyk and Woodruff [11] on higher frequency moments and Kane, Nelson, and Woodruff [12] on distinct counting.

References

[1] J. Lawrence Carter and Mark N. Wegman. Universal Classes of Hash Functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979. Primary publication record.

[2] Michael L. Fredman, János Komlós, and Endre Szemerédi. Storing a Sparse Table with $O(1)$ Worst Case Access Time. *Journal of the ACM*, 31(3):538–544, 1984. Original paper.

[3] Martin Dietzfelbinger, Anna R. Karlin, Kurt Mehlhorn, Friedhelm Meyer auf der Heide, Hans Rohnert, and Robert E. Tarjan. Dynamic Perfect Hashing: Upper and Lower Bounds. *SIAM Journal on Computing*, 23(4):738–761, 1994; conference version, *FOCS 1988*. doi:10.1137/S0097539791194094.

[4] Michael A. Bender, Martín Farach-Colton, John Kuszmaul, William Kuszmaul, and Mingmou Liu. On the Optimal Time/Space Tradeoff for Hash Tables. In *Proceedings of STOC*, pages 1284–1297, 2022. Author manuscript; doi:10.1145/3519935.3519969.

[5] Tianxiao Li, Jingxun Liang, Huacheng Yu, and Renfei Zhou. Tight Cell-Probe Lower Bounds for Dynamic Succinct Dictionaries. In *Proceedings of FOCS*, pages 1842–1862, 2023. Author manuscript; doi:10.1109/FOCS57990.2023.00112.

[6] Burton H. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Communications of the ACM*, 13(7):422–426, 1970. doi:10.1145/362686.362692.

[7] Prosenjit Bose, Hua Guo, Evangelos Kranakis, Anil Maheshwari, Pat Morin, Jason Morrison, Michiel Smid, and Yihui Tang. On the False-Positive Rate of Bloom Filters. *Information Processing Letters*, 108(4):210–213, 2008. Author manuscript; doi:10.1016/j.ipl.2008.05.018.

[8] Philippe Flajolet and G. Nigel Martin. Probabilistic Counting Algorithms for Data Base Applications. *Journal of Computer and System Sciences*, 31(2):182–209, 1985. Original paper; doi:10.1016/0022-0000(85)90041-8.

[9] Noga Alon, Yossi Matias, and Mario Szegedy. The Space Complexity of Approximating the Frequency Moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999; conference version, *STOC 1996*. Author manuscript.

[10] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. Counting Distinct Elements in a Data Stream. In *RANDOM 2002*, Lecture Notes in Computer Science 2483, pages 1–10, 2002. doi:10.1007/3-540-45726-7_1.

[11] Piotr Indyk and David Woodruff. Optimal Approximations of the Frequency Moments of Data Streams. In *Proceedings of STOC*, pages 202–208, 2005. doi:10.1145/1060590.1060621.

[12] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. An Optimal Algorithm for the Distinct Elements Problem. In *Proceedings of PODS*, pages 41–52, 2010. doi:10.1145/1807085.1807094.

[13] Graham Cormode and S. Muthukrishnan. An Improved Data Stream Summary: The Count-Min Sketch and Its Applications. *Journal of Algorithms*, 55(1):58–75, 2005. doi:10.1016/j.jalgor.2003.12.001.