

Randomized Data Structures

Skip Lists, Bloom Filters, Count-Min Sketch

Jared Saia

University of New Mexico

Outline

- Skip Lists
- Bloom Filters
- Count-Min Sketch

Dictionary ADT

A dictionary ADT implements the following operations

- *Insert(x)*: puts the item x into the dictionary
- *Delete(x)*: deletes the item x from the dictionary
- *IsIn(x)*: returns true iff the item x is in the dictionary

Skip Lists

- An ordered dictionary with expected $O(\log(n + 1))$ search, insertion, and deletion time.
- Predecessor and successor queries use the same search path.
- The bounds hold for fixed keys and queries, independent of the random promotions.
- Rank and select (the i -th smallest key) additionally require each horizontal link to store how many bottom-level keys it spans.

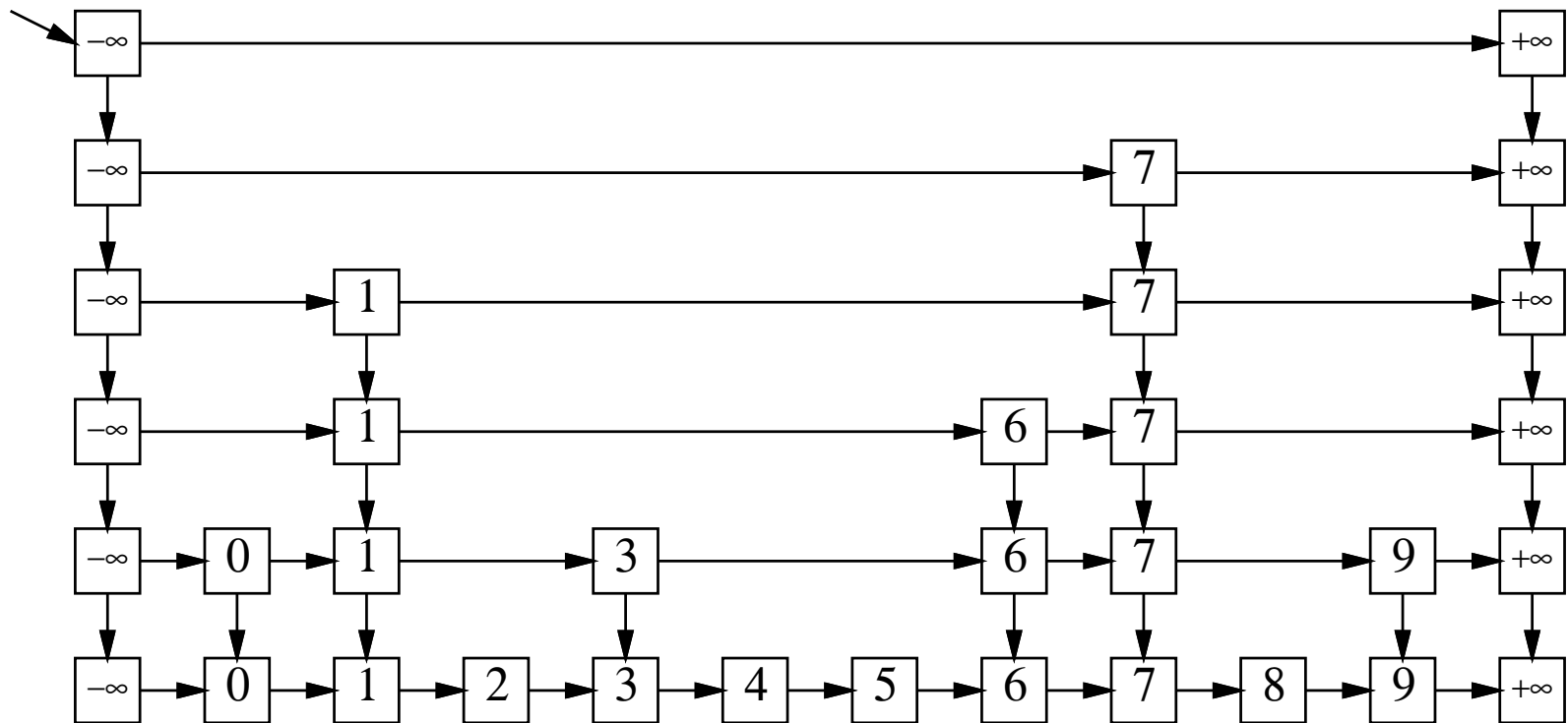
Skip-List Representation

- Distinct keys occupy sorted, doubly linked lists $L_0, L_1, \dots, L_H$. Every key appears in L_0 .
- Each level has head and tail sentinels with keys $-\infty$ and $+\infty$.
- A key of height h has one copy at every level $0, \dots, h$.
- H is the largest key height. An empty dictionary has only L_0 .

Random Promotions

- For each new key, flip independent fair coins until the first tails. Its height is the number of heads.
- Promote the key one level for each heads. Coins for different keys are independent.
- Each copy stores its key and *left*, *right*, *up*, and *down* pointers. Missing vertical neighbors are NIL.
- Head sentinels are also linked vertically.

Skip-List Example



The diagram includes an optional empty top level.

Search

- Start at the highest head sentinel and move right while the next key is strictly less than x .
- When the next key is at least x , move down. Continue to level 0 even if x appears above it.
- At the bottom, let q be the next node to the right.
- Return q if its key equals x . Otherwise, return NIL.

Search

Search(x)

$v \leftarrow \text{head}[H]$

for $\ell \leftarrow H, H - 1, \dots, 0$

while $\text{key}(\text{right}(v)) < x$

$v \leftarrow \text{right}(v)$

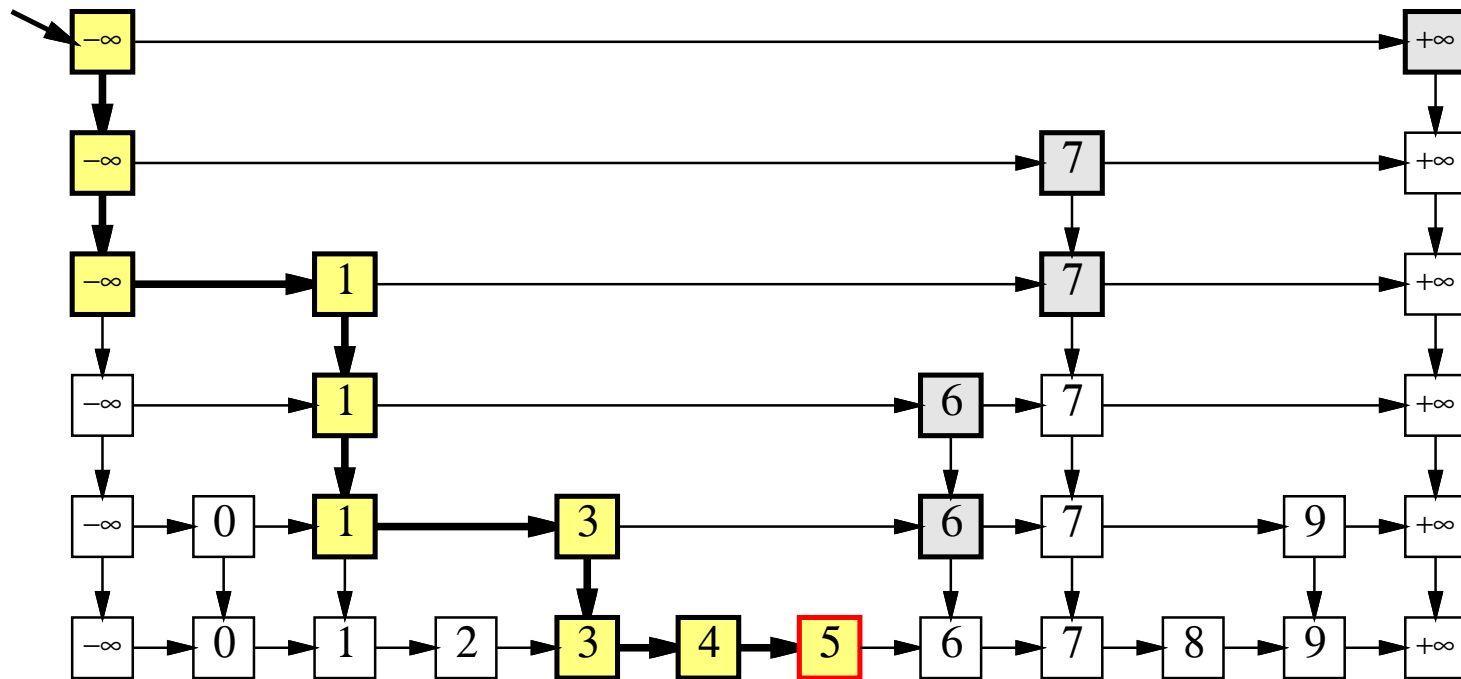
if $\ell > 0$ **then** $v \leftarrow \text{down}(v)$

$q \leftarrow \text{right}(v)$

if $\text{key}(q) \neq x$ **then** $q \leftarrow \text{NIL}$

return q

Search Example



For $x = 5$, the bottom predecessor is 4. The final check finds 5. The optional empty top level adds one downward step.

Insertion

For insertion, augment Search to fill $\text{pred}[\ell]$ with the rightmost node visited at level ℓ , whose key is less than x .

Insert(x)

$q \leftarrow \text{Search}(x)$

if $q \neq \text{NIL}$ **then return**

$h \leftarrow 0$

while a fair coin is heads: $h \leftarrow h + 1$

Create any missing levels through h (see below)

for $\ell \leftarrow 0, \dots, h$

 Splice a copy of x after $\text{pred}[\ell]$

 Link it to the copy below, if $\ell > 0$

$H \leftarrow \max(H, h)$

Each new level has linked sentinels and $\text{pred}[\ell] = \text{head}[\ell]$.

Deletion

Delete(x)

$v \leftarrow \mathbf{Search}(x)$

while $v \neq \text{NIL}$

$u \leftarrow \text{up}(v)$

$\text{right}(\text{left}(v)) \leftarrow \text{right}(v)$

$\text{left}(\text{right}(v)) \leftarrow \text{left}(v)$

 Delete this copy and set $v \leftarrow u$

while $H > 0$ and level H has no keys

 Remove its sentinels and set $H \leftarrow H - 1$

The search returns the bottom copy. Repair surviving sentinel links when removing a level.

Analysis Plan

- Each level retains half the keys of the previous level in expectation.
- First bound the maximum height using a union bound.
- Next compute expected space using the tail-sum identity.
- Finally bound expected horizontal and vertical search steps separately. A high-probability height bound alone does not bound expected running time.

Height of One Key

Let H_u be the number of promotions of key u . For every integer $h \geq 0$,

$$\Pr(H_u \geq h) = 2^{-h}.$$

Reaching level h requires the first h coin flips to be heads.

For $n \geq 2$, let $h = \lceil 2 \log_2 n \rceil$. Then

$$\Pr(H_u \geq h) = 2^{-h} \leq n^{-2}.$$

Key u occupies $H_u + 1$ levels, starting at level 0.

Union Bound

For any events $A_1, \dots, A_n$,

$$\Pr\left(\bigcup_{u=1}^n A_u\right) \leq \sum_{u=1}^n \Pr(A_u).$$

Reminder: The indicator $I(A)$ is 1 when event A occurs and 0 otherwise. Thus $\mathbb{E}[I(A)] = \Pr(A)$.

Proof: For every outcome,

$$I\left(\bigcup_{u=1}^n A_u\right) \leq \sum_{u=1}^n I(A_u).$$

The left side says whether any event occurs. The right side counts how many occur. Take expectations to obtain the bound.

Maximum Height

Let $H = \max_{1 \leq u \leq n} H_u$. For every integer $h \geq 0$,

$$\Pr(H \geq h) \leq \sum_{u=1}^n \Pr(H_u \geq h) = n2^{-h}.$$

For any constant $c > 1$ and $n \geq 2$,

$$\Pr(H \geq \lceil c \log_2 n \rceil) \leq n^{1-c}.$$

In particular, $c = 2$ gives failure probability at most $1/n$.

Height with High Probability

- An event holds *with high probability* if its probability is $1 - o(1)$ as $n \rightarrow \infty$.
- The maximum height is $O(\log n)$ with high probability.
- This is a statement about the height. We will prove the expected search-time bound by summing tail probabilities.
- All expectations are over independent fair promotion coins, with the stored keys and query fixed in advance.

Tail-Sum Identity

If X is a nonnegative integer-valued random variable, then

$$\mathbb{E}[X] = \sum_{r=1}^{\infty} \Pr(X \geq r).$$

Proof: Pointwise, $X = \sum_{r \geq 1} I(X \geq r)$. Taking expectations of these nonnegative terms gives the identity.

The support may be unbounded. If $X \leq N$, the sum stops at N .

In-Class Exercise: Space

Let $C_u = H_u + 1$ be the number of copies of key u .

- What are $\Pr(C_u \geq 1)$, $\Pr(C_u \geq 2)$, and $\Pr(C_u \geq 3)$?
- What is $\Pr(C_u \geq r)$ for an integer $r \geq 1$?
- Use the tail-sum identity to compute $\mathbb{E}[C_u]$.
- What is the expected total number of key nodes?

Expected Space

For $r \geq 1$, $\Pr(C_u \geq r) = 2^{-(r-1)}$. Thus

$$\mathbb{E}[C_u] = \sum_{r=1}^{\infty} 2^{-(r-1)} = 2.$$

By linearity of expectation,

$$\mathbb{E}\left[\sum_{u=1}^n C_u\right] = 2n.$$

This counts key nodes. Including sentinels still gives expected $O(n)$ space: $H + 1 \leq \sum_u C_u$ for $n \geq 1$.

Search Viewed Backwards

Fix the stored keys and a query x , independently of the coins.

- Search ends at the bottom predecessor of x , then checks its right neighbor for equality.
- Trace the predecessor path backwards: go up when an upper copy exists, otherwise go left. At a head sentinel, go up until reaching the starting level.
- At each level, the forward path scans the keys below x that lie after the last promoted key below x .
- This description applies whether or not x is present.

Horizontal Steps at One Level

Let R_ℓ be the number of rightward steps at level ℓ . Condition on the keys in L_ℓ and scan those below x from right to left.

- Stop at the first promoted key or the head sentinel. The skipped suffix has length R_ℓ .
- Promotions from L_ℓ to $L_{\ell+1}$ are independent fair coins, even after conditioning on L_ℓ .

Thus, for $r \geq 1$,

$$\Pr(R_\ell \geq r \mid L_\ell) \leq 2^{-r}, \quad \mathbb{E}[R_\ell] \leq \sum_{r \geq 1} 2^{-r} = 1.$$

A short list or a sentinel can only shorten this scan.

Expected Horizontal Steps

There are $n2^{-\ell}$ expected keys in L_ℓ . Since R_ℓ cannot exceed that level's number of keys,

$$\mathbb{E}[R_\ell] \leq \min\{1, n2^{-\ell}\}.$$

Set $K = \lceil \log_2 n \rceil$ for $n \geq 1$. Then

$$\begin{aligned} \mathbb{E}\left[\sum_{\ell \geq 0} R_\ell\right] &\leq \sum_{\ell=0}^K 1 + \sum_{\ell=K+1}^{\infty} n2^{-\ell} \\ &= K + 1 + n2^{-K} \leq K + 2. \end{aligned}$$

Levels above H contribute zero. No independence between levels is needed.

Expected Search Time

The number of downward steps is H . By the tail-sum identity,

$$\begin{aligned}\mathbb{E}[H] &\leq \sum_{\ell=1}^K 1 + \sum_{\ell=K+1}^{\infty} n2^{-\ell} \\ &\leq K + 1.\end{aligned}$$

Hence the expected number of pointer steps is at most

$$(K + 2) + (K + 1) = 2K + 3 = O(\log(n + 1)).$$

The final check costs $O(1)$, and an empty list takes $O(1)$ time. Insertion and deletion add expected $O(\log(n + 1))$ work for copies and level maintenance.

Bloom Filters

- A randomized data structure for representing a set using little space.
- **Insert**(x): add x to the represented set.
- **IsMember**(x): report whether x may belong to the set.
- Membership queries can produce false positives.
- Applications include databases, networking, and packet routing.

Bloom Filters

- Have m slots, k hash functions, n elements; assume hash functions are all independent
- Each slot stores 1 bit, initially all bits are 0
- Insert(x) : Set the bit in slots $h_1(x), h_2(x), \dots, h_k(x)$ to 1
- IsMember(x) : Return yes iff the bits in $h_1(x), h_2(x), \dots, h_k(x)$ are all 1

Bloom Filter: Insertions

Use $m = 8$ bits and $k = 2$ hash functions. Insert a , then b .

Key	h_1	h_2	Bit position	1	2	3	4	5	6	7	8
a	1	4	Initially	0	0	0	0	0	0	0	0
b	4	6	After a	1	0	0	1	0	0	0	0
			After b	1	0	0	1	0	1	0	0

Both insertions set bit 4. Setting a bit again leaves it at 1.

Bloom Filter: Queries

After inserting a and b , bits 1, 4, 6 are set.

Query	Hash positions	Bits read	Result
a	1, 4	1, 1	True positive
c	1, 6	1, 1	False positive
d	2, 4	0, 1	Definite negative

Neither c nor d was inserted. The filter accepts c because other keys set both of its positions.

Analysis Sketch

- m slots, k hash functions, n elements; assume hash functions are all independent
- Then $P(\text{fixed slot is still } 0) = (1 - 1/m)^{kn}$
- Useful fact from Taylor expansion of e^{-x} :
 $e^{-x} - x^2/2 \leq 1 - x \leq e^{-x}$ for $x < 1$
- Then if $x \leq 1$

$$e^{-x}(1 - x^2) \leq 1 - x \leq e^{-x}$$

Analysis

- Thus we have the following to good approximation.

$$\begin{aligned} Pr(\text{fixed slot is still 0}) &= (1 - 1/m)^{kn} \\ &\approx e^{-kn/m} \end{aligned}$$

- Let $p = e^{-kn/m}$ and let ρ be the fraction of 0 bits after n elements inserted then

$$Pr(\text{false positive}) = (1 - \rho)^k \approx (1 - p)^k$$

- Where this last approximation holds because ρ is very close to p (by a Martingale argument beyond the scope of this class)

Choosing the Number of Hashes

For $n > 0$, minimize the approximation $(1 - p)^k$, where $p = e^{-kn/m}$. Its logarithm is

$$g(p) = -\frac{m}{n} \ln p \ln(1 - p), \quad g'(p) = \frac{m}{n} \frac{p \ln p - (1 - p) \ln(1 - p)}{p(1 - p)}.$$

- The numerator is negative for $p < 1/2$ and positive for $p > 1/2$.* Thus g is minimized at $p = 1/2$.
- The continuous optimum is $k^* = (m/n) \ln 2$.
The resulting rate is $2^{-k^*} \approx (0.6185)^{m/n}$.
- For integer $k \geq 1$, compare $\max(1, \lfloor k^* \rfloor)$ and $\max(1, \lceil k^* \rceil)$ in the approximation.

*For $r(p) = p \ln p - (1 - p) \ln(1 - p)$, $r''(p) > 0$ on $(0, 1/2)$ and $r(0) = r(1/2) = 0$ by continuity. Use $r(1 - p) = -r(p)$.

Union and Resizing

- **Union:** OR the two bit vectors. Both filters must use the same size, hash functions, and hash seeds.
- **Halving:** For an even size m , index bits by $0, \dots, m - 1$. Replace each pair $B[j], B[j + m/2]$ by their OR.
- Queries in the folded filter must use indices $h_a(x) \bmod (m/2)$.
- Folding saves space but can increase the false-positive rate.

Counting Bloom Filters

- Replace bits with counters. Insertions increment the hashed counters, and deletions decrement them.
- Delete only an occurrence that was actually inserted and has not yet been deleted. A positive membership answer alone does not establish this.
- Prevent counter overflow. Invalid deletions or overflow can create false negatives.
- Repeated hash positions must receive matching increments and decrements.

Data Streams

- A router forwards a stream of packets.
- An administrator may want to identify fixed-length substrings that occur unusually often, for example in spam or worm traffic.
- The stream can contain too many distinct substrings to keep an exact counter for each one.
- We want compact frequency estimates and a way to identify frequent items.

Streaming Model

- In this lecture, the algorithm processes a stream sequentially in one pass.
- Available memory is much smaller than the stream.
- Each arrival contributes one occurrence of its item. There are no deletions or weighted updates.
- We allow approximate frequency estimates with a quantified error probability.

Frequency Queries

The stream is $j_1, j_2, \dots, j_T$. Define

$$f_i(T) = \text{Count}(i, T) = \sum_{t=1}^T I(j_t = i).$$

- A query supplies an item i and asks for an estimate of $f_i(T)$.
- The sum of all true frequencies is T .
- For the probability bounds, fix the stream, queried item, and query time independently of the sketch's random hashes.

Count-Min Sketch

- Use d rows of w counters each, for $m = dw$ counters in total.
- Choose row hashes $h_1, \dots, h_d$ independently, each mapping keys to $\{1, \dots, w\}$.
- Within a row, require the collision bound

$$\Pr(h_a(i) = h_a(j)) \leq 1/w \quad \text{for } i \neq j.$$

- A universal hash family supplies this bound. Rows use independently chosen functions from the family.

Counter Array

- Let $C_{a,b}$ be the counter in row a and column b .
- Row a uses hash h_a . Item i maps to counter $C_{a,h_a(i)}$.
- Different items can share a counter, causing overcounting.
- Independent rows provide different collision patterns.

Unit Updates

All counters start at zero.

Update(i)

for $a \leftarrow 1, \dots, d$

$$C_{a, h_a(i)} \leftarrow C_{a, h_a(i)} + 1$$

Each arriving item adds exactly 1 in each row. An update takes $O(d)$ time, assuming constant-time hash evaluation and counter operations.

Count Estimates

After T unit updates, return

$$\hat{f}_i(T) = \min_{1 \leq a \leq d} C_{a, h_a(i)}(T).$$

- Every selected counter includes all $f_i(T)$ occurrences of i .
- Taking the minimum reduces collision overcounting.
- A query takes $O(d)$ time.

Count-Min Example: Updates

Use $d = 3$ rows and $w = 4$ columns. Process a, b, a, c, b, a .

Item	h_1	h_2	h_3
a	1	2	3
b	1	3	1
c	2	2	1

The first a increments $C_{1,1}, C_{2,2}, C_{3,3}$.

The next b increments $C_{1,1}, C_{2,3}, C_{3,1}$.

Continue in the same way for the remaining four arrivals.

Count-Min Example: Queries

After a, b, a, c, b, a , the counter array is

	Column 1	Column 2	Column 3	Column 4
Row 1	5	1	0	0
Row 2	0	4	2	0
Row 3	3	0	3	0

- Query a : $\hat{f}_a = \min(5, 4, 3) = 3 = f_a$. The third row removes the overcount in the first two rows.
- Suppose an unseen key z hashes to $(1, 2, 1)$. Then $\hat{f}_z = \min(5, 4, 3) = 3$, although $f_z = 0$.

Count-Min Guarantee

Fix a unit-update stream, item i , and time T , independently of the random hashes. Given $0 < \varepsilon, \delta < 1$, choose

$$w = \left\lceil \frac{e}{\varepsilon} \right\rceil, \quad d = \left\lceil \ln \frac{1}{\delta} \right\rceil.$$

- Always, $\hat{f}_i(T) \geq f_i(T)$.
- With probability at least $1 - \delta$,

$$\hat{f}_i(T) \leq f_i(T) + \varepsilon T.$$

- Space is dw counters. Update and query time are $O(d)$.

Overcount in One Row

For a fixed query (i, T) , let Z_a be the number of arrivals of other items that collide with i in row a .

$$C_{a, h_a(i)}(T) = f_i(T) + Z_a, \quad Z_a \geq 0.$$

Consequently,

$$\hat{f}_i(T) = f_i(T) + \min_a Z_a \geq f_i(T).$$

It remains to bound the probability that all rows have large overcount.

Collision Indicators

For row a and arrival t , define

$$X_{a,t} = I(j_t \neq i \text{ and } h_a(j_t) = h_a(i)).$$

- If $j_t = i$, then $X_{a,t} = 0$.
- If $j_t \neq i$, then $\mathbb{E}[X_{a,t}] \leq 1/w$ by the collision bound.
- Thus $Z_a = \sum_{t=1}^T X_{a,t}$.

Repeated arrivals may make indicators dependent within a row.
Linearity of expectation does not require independence.

Expected Overcount

By linearity of expectation,

$$\begin{aligned}\mathbb{E}[Z_a] &= \sum_{t=1}^T \mathbb{E}[X_{a,t}] \\ &\leq \frac{T - f_i(T)}{w} \leq \frac{T}{w}.\end{aligned}$$

Since $w \geq e/\varepsilon$, we obtain

$$\mathbb{E}[Z_a] \leq \frac{\varepsilon T}{e}.$$

Next use Markov's inequality to turn this expectation into a probability bound.

Markov's Inequality

For a nonnegative random variable X and $\lambda > 0$,

$$\Pr(X \geq \lambda) \leq \frac{\mathbb{E}[X]}{\lambda}.$$

Proof: $X \geq \lambda I(X \geq \lambda)$. Take expectations.

For $T > 0$, apply this with $X = Z_a$ and $\lambda = \varepsilon T$:

$$\Pr(Z_a > \varepsilon T) \leq \frac{\mathbb{E}[Z_a]}{\varepsilon T} \leq \frac{1}{e}.$$

At $T = 0$, every frequency and estimate is zero.

Independent Rows Reduce Error

The row overcounts $Z_1, \dots, Z_d$ are independent because each is a function of its independently chosen row hash and the fixed stream.

$$\begin{aligned}\Pr(\hat{f}_i(T) > f_i(T) + \varepsilon T) &= \Pr\left(\bigcap_{a=1}^d \{Z_a > \varepsilon T\}\right) \\ &= \prod_{a=1}^d \Pr(Z_a > \varepsilon T) \\ &\leq e^{-d} \leq \delta.\end{aligned}$$

The minimum fails only if every row fails.

Parameters and Multiple Queries

- Width controls additive error: $w = \lceil e/\varepsilon \rceil$.
- Depth controls failure probability: $d = \lceil \ln(1/\delta) \rceil$ for one fixed query.
- For $Q \geq 1$ queries fixed independently of the hashes, use

$$d = \left\lceil \ln \frac{Q}{\delta} \right\rceil.$$

A union bound then gives simultaneous success probability at least $1 - \delta$.

- Each counter needs $\lceil \log_2(T+1) \rceil$ bits, apart from hash storage. Space in counters and space in bits are different measures.

Heavy Hitters Need Candidates

Fix $0 < \varepsilon < \phi < 1$. At the final time T , call an item heavy if $f_i(T) > \phi T$.

- Count-Min estimates the frequency of a supplied key. It does not enumerate the keys to query.
- A list of every key whose estimate ever crosses a threshold has no useful small-space bound in general.
- One solution is a separate deterministic candidate summary with $s = \lceil 1/\phi \rceil$ key-counter pairs (Misra–Gries).
- Maintain that summary and Count-Min together. Recheck only the final candidates against the final threshold ϕT .

A Bounded Candidate Summary

On each arrival i , maintain at most s key-counter pairs:

1. If i is stored, increment its counter.
2. Otherwise, if a slot is free, insert $(i, 1)$.
3. Otherwise, decrement all s counters and remove zeros. Discard this occurrence of i .

Each full decrement cancels $s + 1$ distinct occurrences. There are at most $T/(s + 1)$ such steps. An absent key has had all its occurrences canceled, so its frequency is at most $T/(s + 1) < \phi T$.

Thus every heavy item is among the final s candidates.

Checking the Candidates

- Use width $\lceil e/\varepsilon \rceil$ and depth $\lceil \ln(s/\delta) \rceil$. Query every final candidate and report those with $\hat{f}_i(T) > \phi T$.
- Every truly heavy item is reported. With probability at least $1 - \delta$, no item with $f_i(T) \leq (\phi - \varepsilon)T$ is reported.
- The union bound applies because the candidate summary depends on the fixed stream, not on Count-Min's hashes.
- Extra candidate storage is at most s keys and s counters. The simple decrement step takes $O(s)$ time.

Recap

- Skip lists: exact ordered-dictionary operations with expected logarithmic time and expected linear space.
- Bloom filters: compact membership queries that may return false positives.
- Count-Min: compact frequency estimates with one-sided additive error, using unit updates here.
- Heavy hitters: combine frequency estimates with a bounded candidate summary.

References

- W. Pugh. *Skip Lists: A Probabilistic Alternative to Balanced Trees*. Communications of the ACM, 33(6), 1990.
- G. Cormode and S. Muthukrishnan. *An Improved Data Stream Summary: The Count-Min Sketch and Its Applications*. Journal of Algorithms, 55(1), 2005.
- J. Misra and D. Gries. *Finding Repeated Elements*. Science of Computer Programming, 2, 1982.