Data Sketching

CSE 4/562: Database Systems | Lecture 12

Count Distinct


```sql
SELECT COUNT(DISTINCT A) FROM R;
SELECT A, COUNT(*) FROM R GROUP BY A;
SELECT A, COUNT(*) ... ORDER BY COUNT(*) DESC LIMIT 10;
```

Count Distinct


```sql
SELECT COUNT(DISTINCT A) FROM R;
SELECT A, COUNT(*) FROM R GROUP BY A;
SELECT A, COUNT(*) ... ORDER BY COUNT(*) DESC LIMIT 10;
```

These are all “Holistic” aggregates ($O(|A|)$ memory).

What happens when you run out of memory?

Sketch

A lossy data structure that can estimate useful statistical properties of a dataset.

Hyperloglog (Flajolet Martin Sketches)

Estimating Count-Distinct

```
SELECT COUNT(DISTINCT A) FROM R;
```

Count Sketch

Estimating GroupBy Count

```
SELECT A, COUNT(*) FROM R GROUP BY A;
```

Count-Min Sketch

Estimating TopK GroupBy Count

```
SELECT A, COUNT(*) ... ORDER BY COUNT(*) DESC LIMIT 10;
```

Count Distinct

Count-Distinct

Input Records:

Working Memory:

Count-Distinct

Input Records: 3

Working Memory: 3

Input Records: 3 5

Working Memory: 3 5

Count-Distinct

Input Records: 3 5 4

Working Memory: 3 5 4

Count-Distinct

Input Records: 3 5 4 4

Working Memory: 3 5 4

Count-Distinct

Input Records: 3 5 4 4 2 4 3 ...

Working Memory: 3 5 4 2 ...

Count-Distinct

Input Records: 3 5 4 4 2 4 3 ...

Working Memory: 3 5 4 2 ...

Challenge: To avoid double counting, we need to track which inputs we've seen.

$O(|A|)$ memory required.

The Coin Flip Game

The Coin Flip Game

Start with 0 points and flip a coin:

Tails (🐈)
You get a point and flip again

Heads (🥷)
The game ends

The Coin Flip Game

Start with 0 points and flip a coin:

Tails (🐈)
You get a point and flip again

Heads (🥷)
The game ends

The Coin Flip Game

Flips	Score

The Coin Flip Game

Flips	Score

The Coin Flip Game

Flips	Score
	0

The Coin Flip Game

Flips	Score
🥷 🐈	0

The Coin Flip Game

Flips	Score
	0

The Coin Flip Game

Flips	Score
🥷	0
🐈 🥷	1

The Coin Flip Game

Flips	Score
🥷	0
🐈 🥷	1
🐈	

The Coin Flip Game

Flips	Score
🥷	0
🐈 🥷	1
🐈 🐈	

The Coin Flip Game

Flips	Score
	0
	1

The Coin Flip Game

Flips	Score
	0
	1

The Coin Flip Game

Flips	Score
	0
	1

The Coin Flip Game

Flips	Score
	0
	1

The Coin Flip Game

Flips	Score
	0
	1
	5

Analyzing The Coin Flip Game

Flips	Score	Probability
🥷	0	0.5
🐕🥷	1	0.25
🐕🐕🥷	2	0.125

Analyzing The Coin Flip Game

Flips	Score	Probability	E[# Games]
🥷	0	0.5	
🐈🥷	1	0.25	
🐈🐈🥷	2	0.125	

Analyzing The Coin Flip Game

Flips	Score	Probability	E[# Games]
🥷	0	0.5	2
🐈🥷	1	0.25	
🐈🐈🥷	2	0.125	

Analyzing The Coin Flip Game

Flips	Score	Probability	E[# Games]
🥷	0	0.5	2
🐈🥷	1	0.25	4
🐈🐈🥷	2	0.125	

Analyzing The Coin Flip Game

Flips	Score	Probability	E[# Games]
🥷	0	0.5	2
🐈🥷	1	0.25	4
🐈🐈🥷	2	0.125	8

Analyzing The Coin Flip Game

Flips	Score	Probability	E[# Games]
🥷	0	0.5	2
🐈🥷	1	0.25	4
🐈🐈🥷	2	0.125	8
🐈 $\times n$ 🥷	n	$\frac{1}{2^{n+1}}$	

Analyzing The Coin Flip Game

Flips	Score	Probability	E[# Games]
🥷	0	0.5	2
🐈🥷	1	0.25	4
🐈🐈🥷	2	0.125	8
🐈 $\times n$ 🥷	n	$\frac{1}{2^{n+1}}$	2^{n+1}

Analyzing The Coin Flip Game

Flips	Score	Probability	E[# Games]
🥷	0	0.5	2
🐈🥷	1	0.25	4
🐈🐈🥷	2	0.125	8
🐈 $\times n$ 🥷	n	$\frac{1}{2^{n+1}}$	2^{n+1}

If I told you that in a series of games, my best score was n...

Analyzing The Coin Flip Game

Flips	Score	Probability	E[# Games]
🥷	0	0.5	2
🐈🥷	1	0.25	4
🐈🐈🥷	2	0.125	8
🐈 $\times n$ 🥷	n	$\frac{1}{2^{n+1}}$	2^{n+1}

If I told you that in a series of games, my best score was n...

... you might reasonably expect that I played 2^{n+1} games

Analyzing The Coin Flip Game

Flips	Score	Probability	E[# Games]
🥷	0	0.5	2
🐈🥷	1	0.25	4
🐈🐈🥷	2	0.125	8
🐈 $\times n$ 🥷	n	$\frac{1}{2^{n+1}}$	2^{n+1}

If I told you that in a series of games, my best score was n...

... you might reasonably expect that I played 2^{n+1} games

(I can get this estimate by only tracking my **top score**.)

Idea: Simulate coin flips with a hash function

`score(value) =` the least `i` such that `(h(value) >> i) & 0x1 == 1`

(the index of the lowest-order non-zero bit)

Example

Object	Hash Bits	Score
O_1	01011011	0

Example

Object	Hash Bits	Score
O_1	01011011	0
O_2	00110111	0

Example

Object	Hash Bits	Score
O_1	01011011	0
O_2	00110111	0
O_3	00111000	3

Example

Object	Hash Bits	Score
O_1	01011011	0
O_2	00110111	0
O_3	00111000	3
O_4	10010010	1

Example

Object	Hash Bits	Score
O_1	01011011	0
O_2	00110111	0
O_3	00111000	3
O_4	10010010	1
O_3	00111000	3

Example

Object	Hash Bits	Score
O_1	01011011	0
O_2	00110111	0
O_3	00111000	3
O_4	10010010	1
O_3	00111000	3
		$\boxed{3}$

Example

Object	Hash Bits	Score
O_1	01011011	0
O_2	00110111	0
O_3	00111000	3
O_4	10010010	1
O_3	00111000	3
		$\boxed{3}$

Estimate: $2^{3+1} = 16$

Example

Object	Hash Bits	Score
O_1	01011011	0
O_2	00110111	0
O_3	00111000	3
O_4	10010010	1
O_3	00111000	3
		$\boxed{3}$

Estimate: $2^{3+1} = 16$

Duplicates can't raise the top score!

This estimate is noisy!

Denoising the Estimate

Idea: Track the lowest score that you have **not** gotten yet.

(Call this number the leading run: R)

Leading Run Size

Object	Hash Bits	Score
O_1	01011011	0
O_2	00110111	0
O_3	00111000	3
O_4	10010010	1
O_3	00111000	3

Leading Run Size

Object	Hash Bits	Score
O_1	01011011	0
O_2	00110111	0
O_3	00111000	3
O_4	10010010	1
O_3	00111000	3

$\{0, 1, 3\}$

Leading Run Size

Object	Hash Bits	Score
O_1	0101101**1**	0
O_2	0011011**1**	0
O_3	0011**1**000	3
O_4	1001001**0**	1
O_3	0011**1**000	3

{0, 1, 3}

R = 2

Leading Run Size

Object	Hash Bits	Score
O_1	01011011	0
O_2	00110111	0
O_3	00111000	3
O_4	10010010	1
O_3	00111000	3

{0, 1, 3}

R = 2

Estimate: $\frac{2^R}{\phi} = \frac{2^2}{0.77351} \approx 5.2$

Denoising the Estimate

Idea:

1. Compute 2 or more estimates in parallel (using different hash functions)
2. Average the results

Hyperloglog ($\approx$ Flajolet-Martin) Sketches

0. Initialize the sketch as an empty set (empty bitvector)

1. For each record...
- Hash the record
- Find the index of the lowest order bit
- Add the index to the set (set the corresponding bit)

2. Find R, the least index not in the set (without a corresponding set bit)

3. Estimate Count-Distinct as $\frac{2^R}{\phi}$ (where $\phi \approx 0.77351$)

4. Repeat (in parallel) as needed

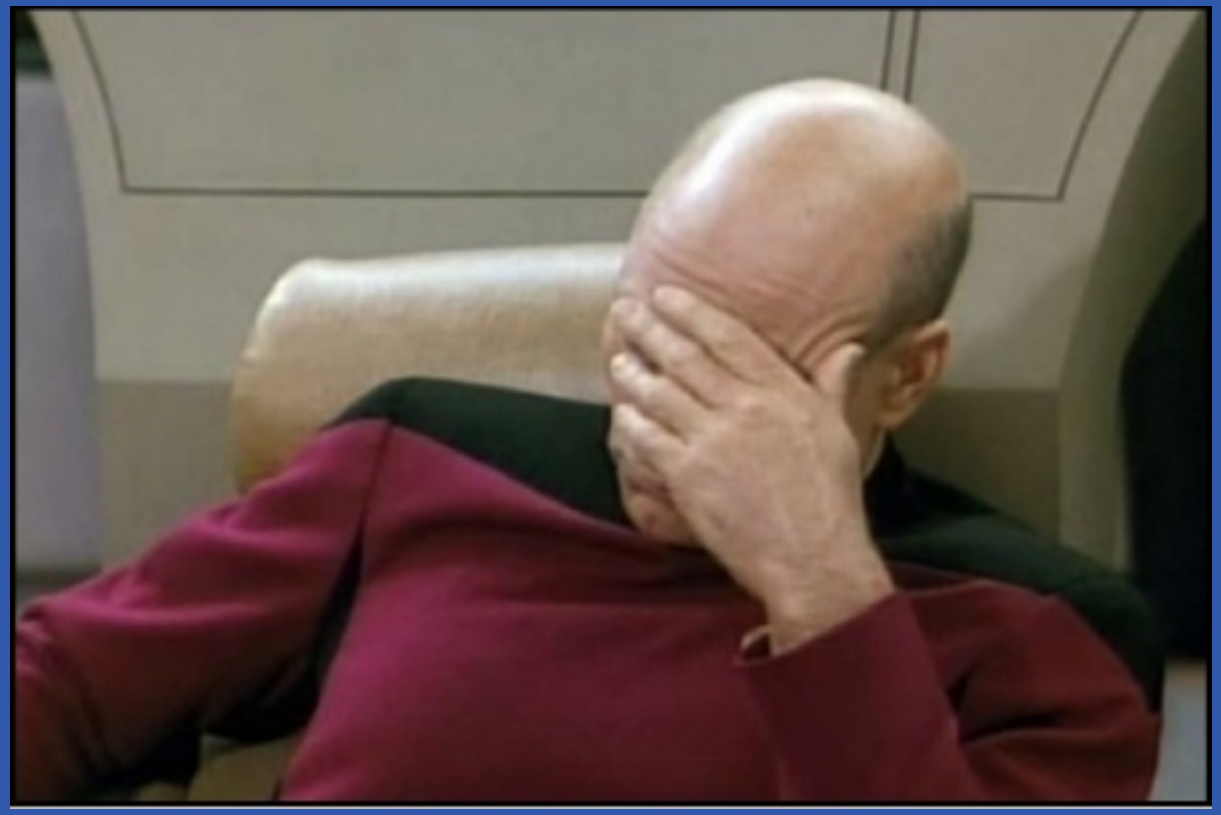
Group-By Count

```sql
SELECT A, COUNT(*) FROM R GROUP BY A;
```

Problem: We need a counter for each group (each unique `A`).

A wacky idea...

Idea: Combine all the counters into a single number!



No... seriously...

$$\delta(O_i) = \begin{cases} \textbf{if } (h(O_i) = 0 \bmod 2) \textbf{ then } (-1) \\ \textbf{if } (h(O_i) = 1 \bmod 2) \textbf{ then } (+1) \end{cases}$$

Ok... and?

$$\sum_i \delta(O_i)$$

Example

Object $\delta(O_i)$ **Running Count**

Example

Object	$\delta(O_i)$	Running Count
O_3	-1	-1

Example

Object	$\delta(O_i)$	Running Count
O_3	−1	−1
O_1	+1	0

Example

Object	$\delta(O_i)$	Running Count
O_3	−1	−1
O_1	+1	0
O_4	−1	−1

Example

Object	$\delta(O_i)$	Running Count
O_3	−1	−1
O_1	+1	0
O_4	−1	−1
O_2	+1	0

Example

Object	$\delta(O_i)$	Running Count
O_3	−1	−1
O_1	+1	0
O_4	−1	−1
O_2	+1	0
O_4	−1	−1

Example

Object	$\delta(O_i)$	Running Count
O_3	−1	−1
O_1	+1	0
O_4	−1	−1
O_2	+1	0
O_4	−1	−1
O_1	+1	0

Example

Object	$\delta(O_i)$	Running Count
O_3	−1	−1
O_1	+1	0
O_4	−1	−1
O_2	+1	0
O_4	−1	−1
O_1	+1	0
O_3	−1	−1

Example

Object	$\delta(O_i)$	Running Count
O_3	−1	−1
O_1	+1	0
O_4	−1	−1
O_2	+1	0
O_4	−1	−1
O_1	+1	0
O_3	−1	−1
O_3	−1	−2

Example

Object	$\delta(O_i)$	Running Count
O_3	−1	−1
O_1	+1	0
O_4	−1	−1
O_2	+1	0
O_4	−1	−1
O_1	+1	0
O_3	−1	−1
O_3	−1	−2
O_1	+1	−1

$$\text{Total Count} = \texttt{COUNT\_OF}\ (O_i) \cdot \delta(O_i) + \sum_{j \neq i} \texttt{COUNT\_OF}\ \big(O_j\big) \cdot \delta\big(O_j\big)$$

$$\text{Total Count} = \texttt{COUNT\_OF}\ (O_i) \cdot \delta(O_i) + \sum_{j \neq i} \texttt{COUNT\_OF}\ \left(O_j\right) \cdot \delta\left(O_j\right)$$

Fixed Values

- O_i: The object we want to count
- All the other O_j ($j \neq i$)
- $\texttt{COUNT\_OF}\ (O_i)$, each $\texttt{COUNT\_OF}\ \left(O_j\right)$
- $\delta(O_i)$ (We can compute it from O_i)

Random Values

- Each $\delta\left(O_j\right)$

The Math...

$$E[\text{Total Count}] = E[\texttt{COUNT\_OF}\ (O_i) \cdot \delta(O_i)] + E\left[\sum_{j \neq i} \texttt{COUNT\_OF}\ \left(O_j\right) \cdot \delta\left(O_j\right)\right]$$

The Math...

$$E[\text{Total Count}] = E[\texttt{COUNT\_OF}\ (O_i) \cdot \delta(O_i)] + E\left[\sum_{j \neq i} \texttt{COUNT\_OF}\ (O_j) \cdot \delta(O_j)\right]$$

Just the second part...

$E\left[\sum_{j \neq i} \texttt{COUNT\_OF}\ (O_j) \cdot \delta(O_j)\right]$

The Math...

$$E[\text{Total Count}] = E[\texttt{COUNT\_OF}\ (O_i) \cdot \delta(O_i)] + E\left[\sum_{j \neq i} \texttt{COUNT\_OF}\ (O_j) \cdot \delta(O_j)\right]$$

Just the second part...

$$E\left[\sum_{j \neq i} \texttt{COUNT\_OF}\ (O_j) \cdot \delta(O_j)\right]$$

$$= \sum_{j \neq i} E\left[\texttt{COUNT\_OF}\ (O_j) \cdot \delta(O_j)\right]$$

The Math...

$$E[\text{Total Count}] = E[\texttt{COUNT\_OF}\ (O_i) \cdot \delta(O_i)] + E\left[\sum_{j \neq i} \texttt{COUNT\_OF}\ (O_j) \cdot \delta(O_j)\right]$$

Just the second part...

$$E\left[\sum_{j \neq i} \texttt{COUNT\_OF}\ (O_j) \cdot \delta(O_j)\right]$$

$$= \sum_{j \neq i} E\left[\texttt{COUNT\_OF}\ (O_j) \cdot \delta(O_j)\right]$$

$$= \sum_{j \neq i} \texttt{COUNT\_OF}\ (O_j) \cdot E\left[\delta(O_j)\right]$$

The Math...

$$E[\text{Total Count}] = E[\texttt{COUNT\_OF}\ (O_i) \cdot \delta(O_i)] + E\left[\sum_{j \neq i} \texttt{COUNT\_OF}\ (O_j) \cdot \delta(O_j)\right]$$

Just the second part...

$$E\left[\sum_{j \neq i} \texttt{COUNT\_OF}\ (O_j) \cdot \delta(O_j)\right]$$

$$= \sum_{j \neq i} E\left[\texttt{COUNT\_OF}\ (O_j) \cdot \delta(O_j)\right]$$

$$= \sum_{j \neq i} \texttt{COUNT\_OF}\ (O_j) \cdot E\left[\delta(O_j)\right]$$

$$= \sum_{j \neq i} \texttt{COUNT\_OF}\ (O_j) \cdot ((0.5 \cdot 1) + (0.5 \cdot -1))$$

$$= 0 \cdot \sum_{j \neq i} \texttt{COUNT\_OF}\ (O_j) = 0$$

The Math...

$$E[\text{Total Count}] = E[\texttt{COUNT\_OF}\ (O_i) \cdot \delta(O_i)] + E\left[\sum_{j \neq i} \texttt{COUNT\_OF}\ \left(O_j\right) \cdot \delta\left(O_j\right)\right]$$

The Math...

$$E[\text{Total Count}] = E[\texttt{COUNT\_OF}\ (O_i) \cdot \delta(O_i)] + E\left[\sum_{j \neq i} \texttt{COUNT\_OF}\ (O_j) \cdot \delta(O_j)\right]$$

$$= E[\texttt{COUNT\_OF}\ (O_i) \cdot \delta(O_i)] + 0$$

The Math...

$$E[\text{Total Count}] = E[\texttt{COUNT\_OF}\ (O_i) \cdot \delta(O_i)] + E\left[\sum_{j \neq i} \texttt{COUNT\_OF}\ (O_j) \cdot \delta(O_j)\right]$$

$$= E[\texttt{COUNT\_OF}\ (O_i) \cdot \delta(O_i)] + 0$$

$$= \texttt{COUNT\_OF}\ (O_i) \cdot \delta(O_i)$$

The Math...

$$E[\text{Total Count}] = E[\texttt{COUNT\_OF}\ (O_i) \cdot \delta(O_i)] + E\left[\sum_{j \neq i} \texttt{COUNT\_OF}\ (O_j) \cdot \delta(O_j)\right]$$

$$= E[\texttt{COUNT\_OF}\ (O_i) \cdot \delta(O_i)] + 0$$

$$= \texttt{COUNT\_OF}\ (O_i) \cdot \delta(O_i)$$

$$\frac{E[\text{Total Count}]}{\delta(O_i)} = \texttt{COUNT\_OF}\ (O_i)$$

The Math...

$$E[\text{Total Count}] = E[\texttt{COUNT\_OF}\ (O_i) \cdot \delta(O_i)] + E\left[\sum_{j \neq i} \texttt{COUNT\_OF}\ (O_j) \cdot \delta(O_j)\right]$$

$$= E[\texttt{COUNT\_OF}\ (O_i) \cdot \delta(O_i)] + 0$$

$$= \texttt{COUNT\_OF}\ (O_i) \cdot \delta(O_i)$$

$$\frac{E[\text{Total Count}]}{\delta(O_i)} = \texttt{COUNT\_OF}\ (O_i)$$

$$\texttt{COUNT\_OF}\ (O_i) \approx \frac{\text{Total Count}}{\delta(O_i)}$$

Back to the example

Object $\delta(O_i)$ **Estimate**

Reminder: The running total was −1

Back to the example

Object	$\delta(O_i)$	**Estimate**
O_1	+1	−1

Reminder: The running total was −1

Back to the example

Object	$\delta(O_i)$	**Estimate**
O_1	+1	−1
O_2	+1	−1

Reminder: The running total was −1

Back to the example

Object	$\delta(O_i)$	**Estimate**
O_1	+1	−1
O_2	+1	−1
O_3	−1	+1

Reminder: The running total was −1

Back to the example

Object	$\delta(O_i)$	**Estimate**
O_1	+1	−1
O_2	+1	−1
O_3	−1	+1
O_4	−1	+1

Reminder: The running total was −1

This estimate is kinda bad...

Problem 1

All of the objects use the same counter, so there's no way to differentiate estimates for O_1 from O_2.

Problem 2

The estimate is **really** noisy.

Idea 1

Use a hash function ($h(\cdot)$) to assign each object to a bucket, and compute a separate counter for each bucket.

Idea 2

Repeat the experiment multiple times with different ($\delta_{i(\cdot)}$, $h_{i(\cdot)}$) in parallel.

A better example

Object	$h_1(O_i)$	$\delta_1(O_i)$	$h_2(O_i)$	$\delta_2(O_i)$
O_1	Bucket 1	−1	Bucket 2	1
O_2	Bucket 1	−1	Bucket 1	−1
O_3	Bucket 2	1	Bucket 1	−1
O_4	Bucket 1	−1	Bucket 1	1

A better example

Objects Seen:

	Bucket 1	Bucket 2
Trial 1	0	0
Trial 2	0	0

Object	Trial 1	Trial 2	Estimate	Real
O_1	0	0	0	0
O_2	0	0	0	0
O_3	0	0	0	0
O_4	0	0	0	0

A better example

Objects Seen: O_2

	Bucket 1	Bucket 2
Trial 1	−1	0
Trial 2	−1	0

Object	Trial 1	Trial 2	Estimate	Real
O_1	1	0	0.5	0
O_2	1	1	1	1
O_3	0	1	0.5	0
O_4	1	−1	0	0

A better example

Objects Seen: O_2, O_1

	Bucket 1	Bucket 2
Trial 1	−2	0
Trial 2	−1	1

Object	Trial 1	Trial 2	Estimate	Real
O_1	2	1	1.5	1
O_2	2	1	1.5	1
O_3	0	1	0.5	0
O_4	2	−1	0.5	0

A better example

Objects Seen: O_2, O_1, O_4

	Bucket 1	Bucket 2
Trial 1	−3	0
Trial 2	0	1

Object	Trial 1	Trial 2	Estimate	Real
O_1	3	1	2	1
O_2	3	0	1.5	1
O_3	0	0	0	0
O_4	3	0	1.5	1

A better example

Objects Seen: O_2, O_1, O_4, O_1

	Bucket 1	Bucket 2
Trial 1	−4	0
Trial 2	0	2

Object	Trial 1	Trial 2	Estimate	Real
O_1	4	2	3	2
O_2	4	0	2	1
O_3	0	0	0	0
O_4	4	0	2	1

A better example

Objects Seen: O_2, O_1, O_4, O_1, O_2

	Bucket 1	Bucket 2
Trial 1	−5	0
Trial 2	−1	2

Object	Trial 1	Trial 2	Estimate	Real
O_1	5	2	3.5	2
O_2	5	1	3	2
O_3	0	1	0.5	0
O_4	5	−1	2	1

The Count Min Sketch

1. Initialize a $B \cdot T$ matrix C of counters to 0
2. For each record O_i...
   - For $t \in [0, T)$: Update counter $C[t, h_t(O_i)] += \delta_t(O_i)$
3. Estimate the count of O_i as the **median** of $C[t, h_t(O_i)] \cdot \delta_t(O_i)$ (for $t \in [0, T)$)
   - Median is more stable than mode

Top-K Group-By Count

```
SELECT A, COUNT(*) ... ORDER BY COUNT(*) DESC LIMIT 10;
```

Problem: “Heavy Hitters” overwhelm smaller counts

Idea

Give up and drop δ_i

Example

Object	**COUNT**	$h_1(O_i)$	$h_2(O_i)$
O_1	10	Bucket 1	Bucket 2
O_2	32	Bucket 1	Bucket 1
O_3	1002	Bucket 2	Bucket 1
O_4	500	Bucket 1	Bucket 1

	Bucket 1	**Bucket 2**
Trial 0	542	1002
Trial 1	1534	10

Example

	Bucket 1	Bucket 2
Trial 0	542	1002
Trial 1	1534	10

Object	Actual COUNT	Trial 1 Estimate	Trial 2 Estimate	Min
O_1	10	542	10	10
O_2	32	542	1534	542
O_3	1002	1002	1534	1002
O_4	500	542	1534	542

The Count Min Sketch

1. Initialize a $B \cdot T$ matrix C of counters to 0
2. For each record O_i...
   - For $t \in [0, T)$: Update counter $C[t, h_t(O_i)] += 1$
3. Estimate the count of O_i as the **min** of $C[t, h_t(O_i)]$ (for $t \in [0, T)$)
   - Bad estimate for any record that is not a heavy hitter.

Reminders

- Midterm grades posted
- Mid semester review grades posted