Approximation Techniques

Supplemental Reading
(papers posted on Piazza)

Announcements

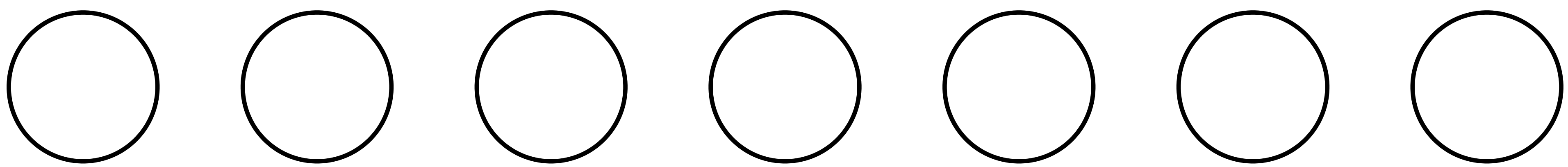
- First P3 Time-Trial-Submissions Tonight.
- P2 Grades Out By Friday.
- Homework 7 due Monday.

Review: WINDOW

```
SELECT L.state, T.month,
       AVG(S.sales) OVER W as movavg
FROM   Sales S, Times T, Locations L
WHERE  S.timeid = T.timeid
  AND  S.locid = L.locid
WINDOW W AS (
   PARTITION BY L.state
   ORDER BY T.month
   RANGE BETWEEN INTERVAL ‘1’ MONTH PRECEDING
         AND INTERVAL ‘1’ MONTH FOLLOWING
)
```

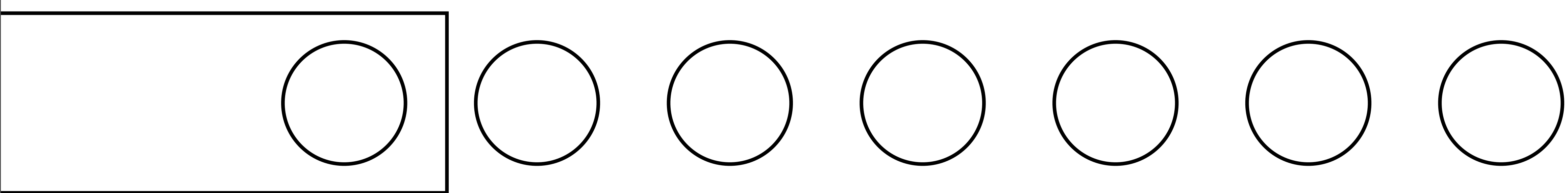
Review: WINDOW

Windowed SUM, Window Size 3



Review: WINDOW

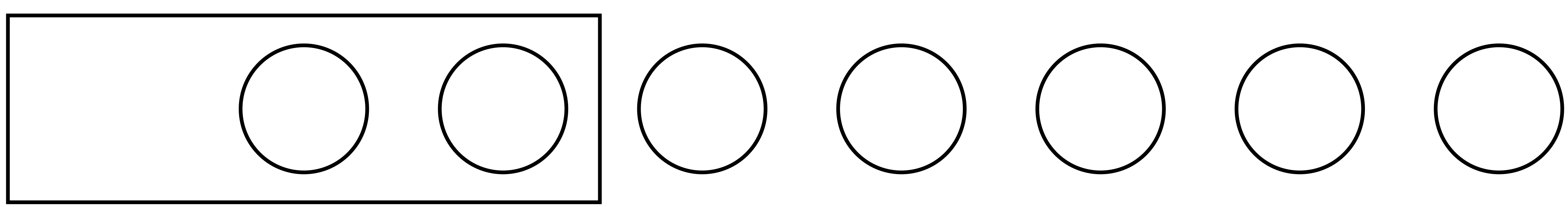
Windowed SUM, Window Size 3



SUM_1

Review: WINDOW

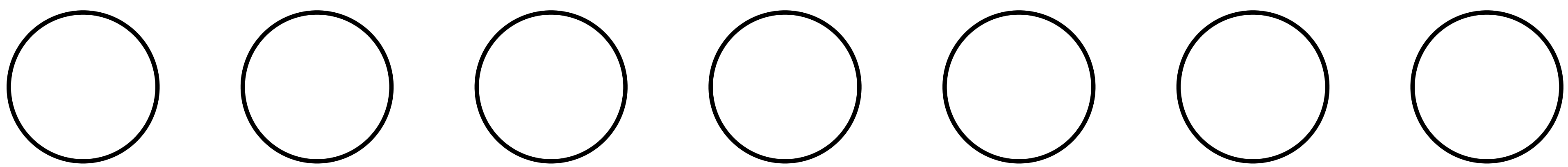
Windowed SUM, Window Size 3



SUM_1 SUM_2

Review: WINDOW

Windowed SUM, Window Size 3



SUM_1 SUM_2 SUM_3 SUM_4 SUM_5 SUM_6 SUM_7 …

Review: WINDOW

```
SELECT L.state, T.month,
       AVG(S.sales) OVER W as movavg
FROM   Sales S, Times T, Locations L
WHERE  S.timeid = T.timeid
  AND  S.locid = L.locid
WINDOW W AS (
   PARTITION BY L.state
   ORDER BY T.month
   RANGE BETWEEN INTERVAL ‘1’ MONTH PRECEDING
         AND INTERVAL ‘1’ MONTH FOLLOWING
)
```

Partition By is like Group By

Required: Define a sort order

Required: Define the size of window (need not be a fixed # of tuples)

Review: Data Warehousing

- Data warehouses store massive datasets
- Workload involves…
  - Frequent, low-latency reads.
  - Lots of aggregation/summarization.

Review: Data Warehousing

- Data warehouses store massive datasets
- Workload involves…
  - Frequent, low-latency reads.
  - Lots of aggregation/summarization.
  - … and may not require precise answers.

Query Approximation

- Summarize the data with a **Sketching** Algorithm.
  - Bloom Filters (Set Containment)
  - Flajolet & Martin Count-Distinct Sketch
  - Count Sketch (Frequent Items/Top-K)
- Sample the data and estimate the error
  - … or better yet, keep generating samples!
    - Online Aggregation & Ripple Joins

Count-Distinct Sketches

- Count-Distinct is Holistic (all tuples needed)
  - Naive External Algorithm: Sort/Eliminate
- Flajolet & Martin Sketch:
  - Summarize dataset in a bit vector.
  - Bit vector 'fills up' as more values added.
  - Hashes eliminate duplicate contributions.

Count-Distinct Sketches

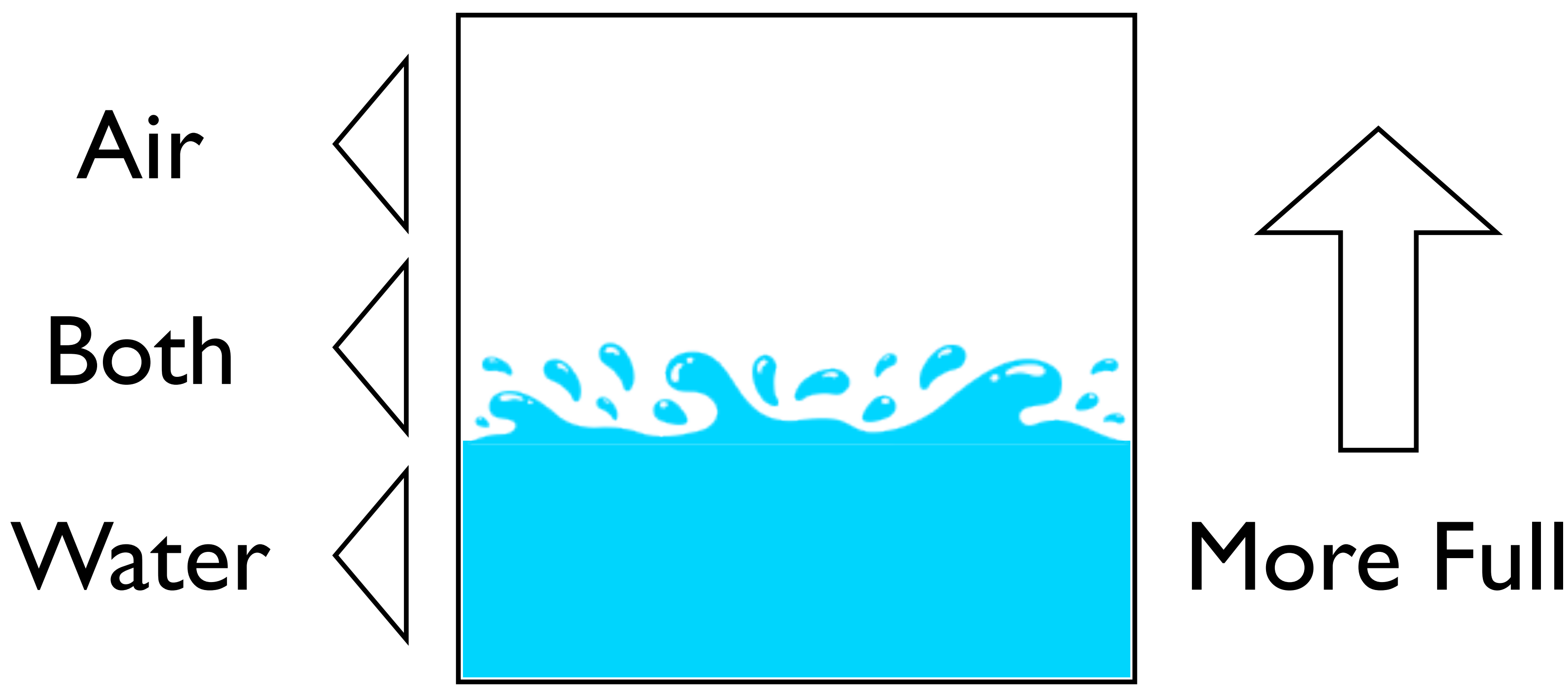


image credit: openclipart.org

Count-Distinct Sketches

0s Both 1s

`0001 0100 0|111 1111`

'Fill' Boundaries Approximate # of Distinct Items in Set

The ρ function

$$\text{hash}(\lozenge) = \texttt{00010010100}$$

$$P[\ \rho(\lozenge) = 0\ ] = ?$$

$$P[\ \rho(\lozenge) = 1\ ] = ?$$

$$P[\ \rho(\lozenge) = k\ ] = ?$$

The ρ function

hash(♢) = 00010010100

Smallest Position with a Non-Zero Bit
(or |bit vector|)

ρ(♢) = 2

P[ρ(♢) = 0] = ?

P[ρ(♢) = 1] = ?

P[ρ(♢) = k] = ?

The ρ function

$$\text{hash}(\diamondsuit) = 00010010100$$

$$\text{sketch}(\diamondsuit) = 2^{\rho(\diamondsuit)}$$

$$= 00000000100$$

$$\text{sketch}(\diamondsuit_1, \diamondsuit_2, \ldots) = \text{sketch}(\diamondsuit_1) \vee \text{sketch}(\diamondsuit_2) \vee \ldots$$

Count-Distinct Sketches

Each item $\diamondsuit$ has a 1/2 chance of $\rho(\diamondsuit) = 0$

What is the probability that $\rho(\diamondsuit) \neq 0$ for all N items?

Count-Distinct Sketches

Each item $\diamondsuit$ has a 1/2 chance of $\rho(\diamondsuit) = 0$

What is the probability that $\rho(\diamondsuit) \neq 0$ for all N items?

Each item $\diamondsuit$ has a $1/2^k$ chance of $\rho(\diamondsuit) = k$

What is the probability that $\rho(\diamondsuit) \neq k$ for all N items?

Count-Distinct Sketches

Given a sketch bit vector,
let R be the position of the lowest zero value.

`0001011111`

$$E[\ R\ ] = \log_2(\ \varphi * |Set|\ )$$

$$\varphi = 0.77351$$

Count-Distinct Sketches

Given a sketch bit vector,
let R be the position of the lowest zero value.

```
0001011111
```

R=6

$$E[\ R\ ] = \log_2(\ \varphi * |Set|\ )$$

$$\varphi = 0.77351$$

Count-Distinct Sketches

$$\varphi = 0.77351$$

$$E[\ R\ ] = \log_2(\ \varphi * |Set|\ )$$

$$2^{R/\varphi} = E[\ |Set|\ ]$$

$$2^{6/\varphi} = 60$$

0001011111 Summarizes a set with 60 distinct elements

Count-Distinct Sketches

- Problem: Estimate has a high variance.
  - Solution: Multiple sketches in parallel.
    - Use average or median of all estimates.
- Question: How does this algorithm count only unique values (count distinct)?
- Question: How big is the bit vector?

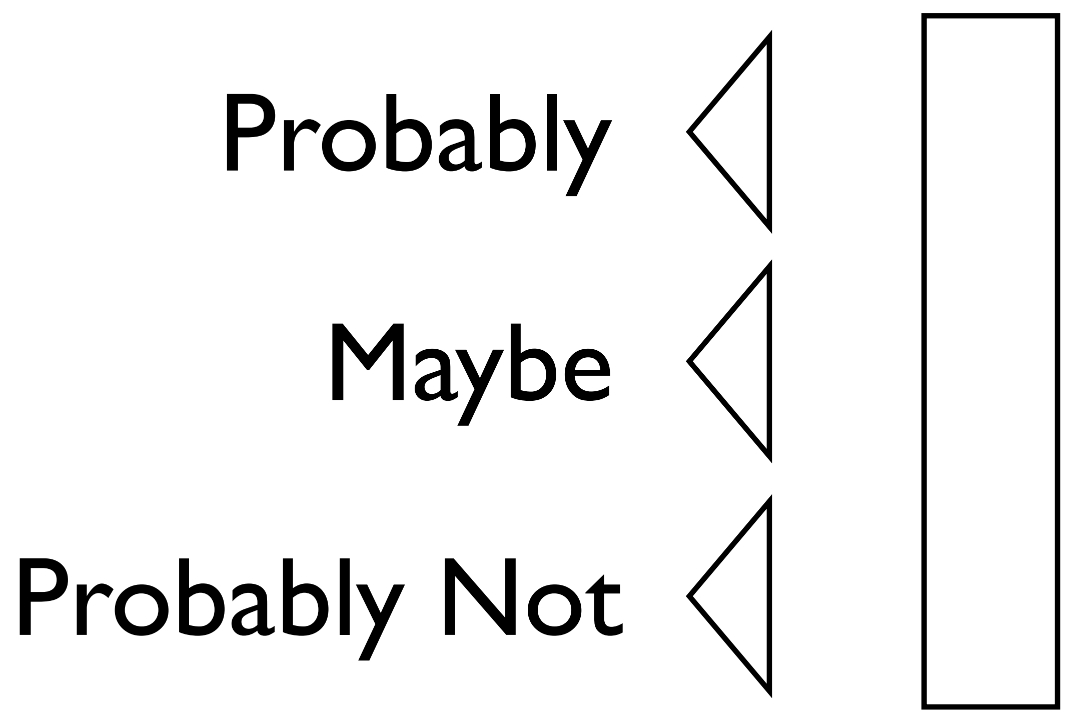
Count Sketch

- Top-K-Count
  - Compute Group-by Count Aggregate.
  - Find the K items with the highest counts.
- Top-K-Count is Holistic
  - Sketch provides some "wiggle room"
    - "Borderline" entries may be excluded.

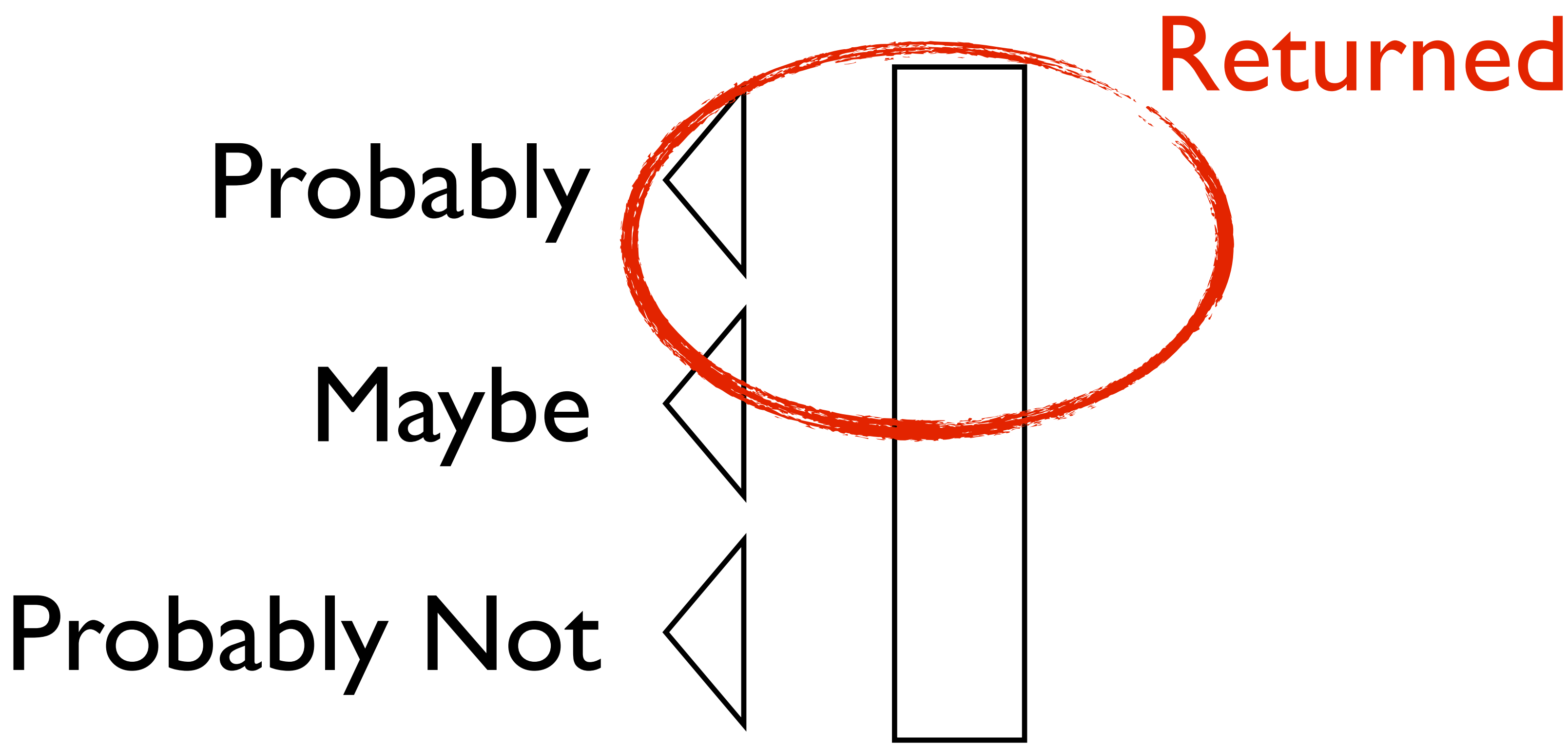
Count Sketch

- Count Sketch Parameters:
  - k : The number of elements to return
  - ϵ : The 'wiggle room'
- Count Sketch Guarantees:
  - If n_k is the lowest count in the top k, all objects returned have count $n_i > (1-\epsilon)\ n_k$.
  - w.h.p., all objects with $n_i > (1+\epsilon)\ n_k$ are returned.

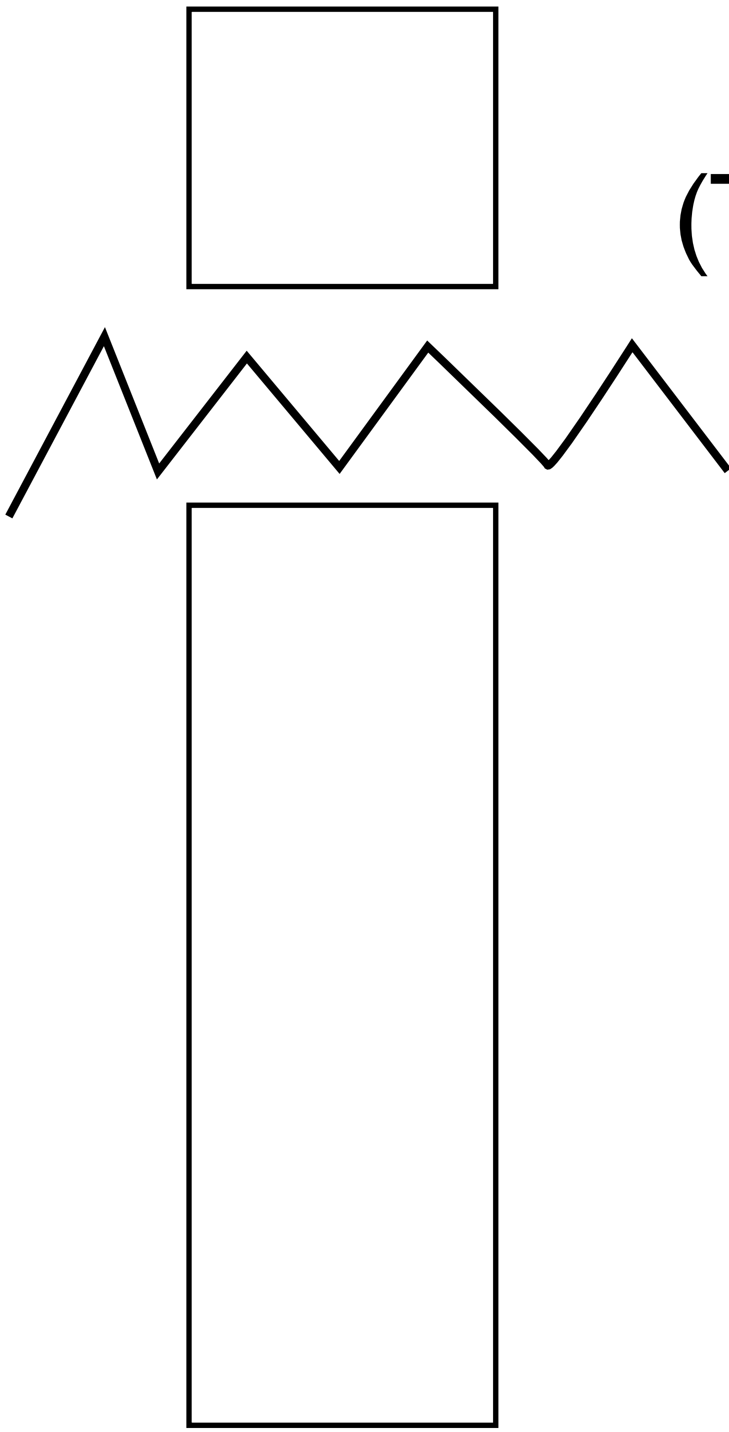
Count Sketch



Count Sketch



Intuition



K entries
(Tracked Explicitly)

For each tuple…

Update a count
Move the tuple up
Update the approximation

Everything
Else
(Approximated)

Count Sketch

$s(\Diamond) \rightarrow \{ +1, -1 \}$ (1-bit of hash($\Diamond$))

$$\text{sketch}(\Diamond_1, \Diamond_2, \ldots) = \sum_i s(\Diamond_i)$$

Count Sketch

$s(\diamondsuit) \rightarrow \{ +1, -1 \}$ (1-bit of hash($\diamondsuit$))

$$\text{sketch}(\diamondsuit_1, \diamondsuit_2, \ldots) = \sum_i s(\diamondsuit_i)$$

For a set R containing precisely N instances of $\diamondsuit$ and nothing else, what is E[sketch(R)]?

Count Sketch

$s(\diamondsuit) \rightarrow \{ +1, -1 \}$ (1-bit of hash($\diamondsuit$))

$$\text{sketch}(\diamondsuit_1, \diamondsuit_2, \ldots) = \sum_i s(\diamondsuit_i)$$

For a set R containing precisely N instances of $\diamondsuit$

and nothing else, what is E[sketch(R)]?

For a set R containing an entirely random set of elements what is E[sketch(R)]?

Count Sketch

E[count(♢)] = s(♢) * sketch(R)

Count Sketch

E[count(♢)] = s(♢) * sketch(R)

Correct (but very non-intuitive)
Problem: EXTREMELY high variance

Count Sketch

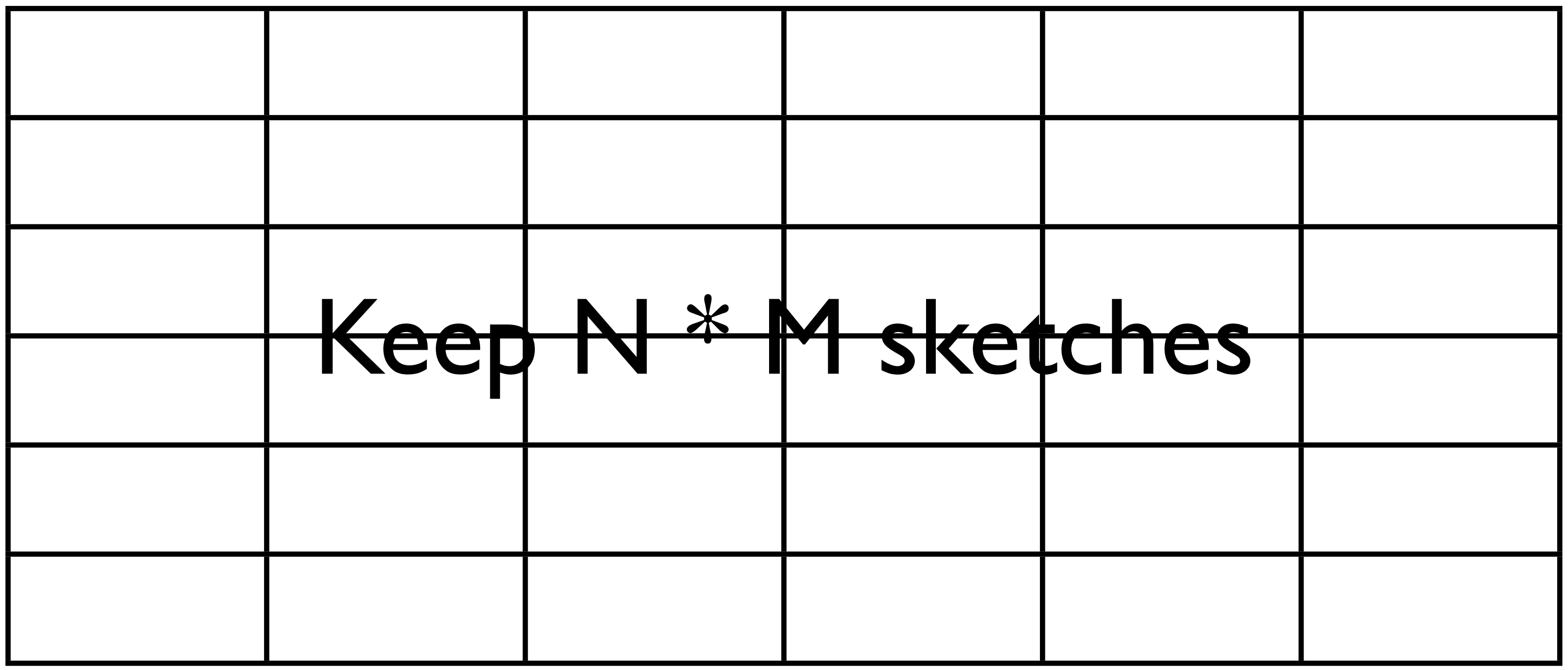
E[count(♢)] = s(♢) * sketch(R)

Correct (but very non-intuitive)

Problem: EXTREMELY high variance

Solution: Use multiple sketches

Count Sketch

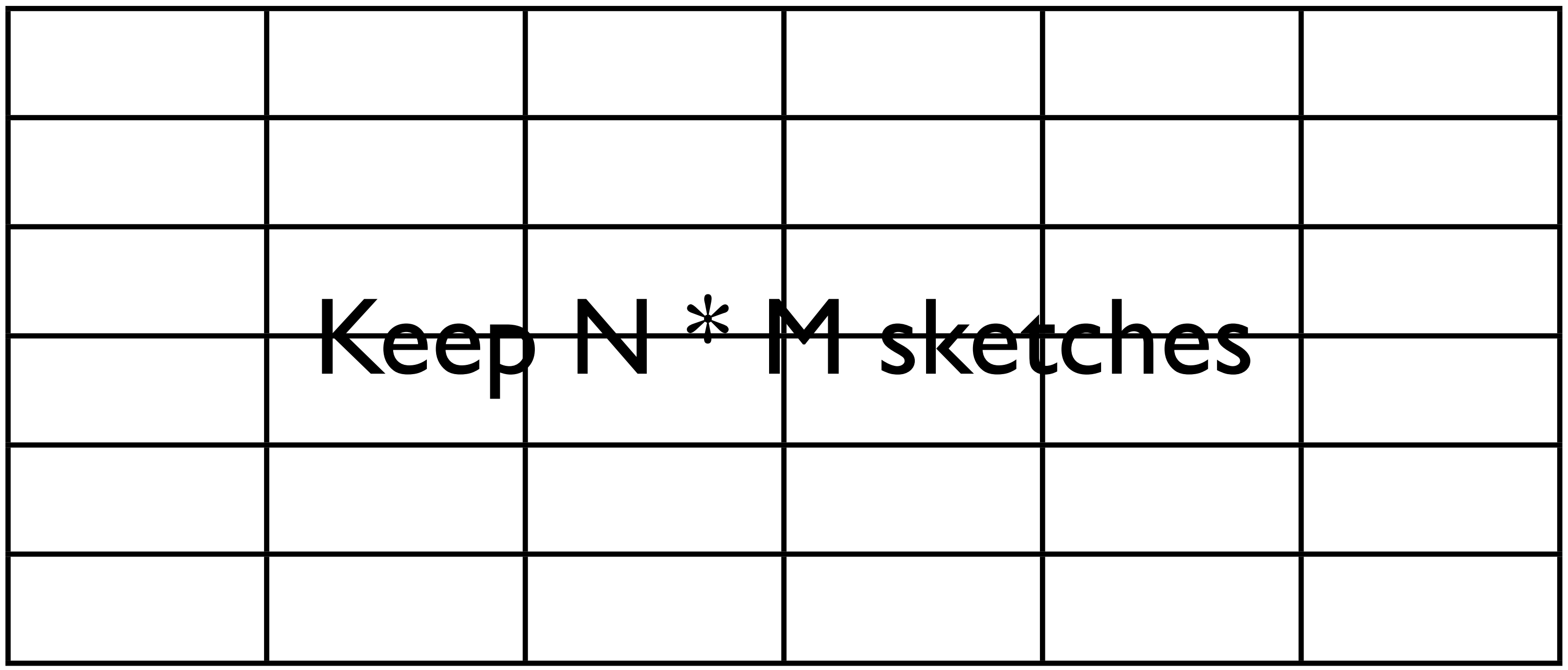


For each $\Diamond$

  For x in [0 to N)

    Update sketch (x, $\text{hash}_x(\Diamond)$ % M)

Count Sketch



To get approximate count($\diamondsuit$):

For x in [0 to N)

Approximate with sketch (x, $\text{hash}_x(\diamondsuit)$ % M)

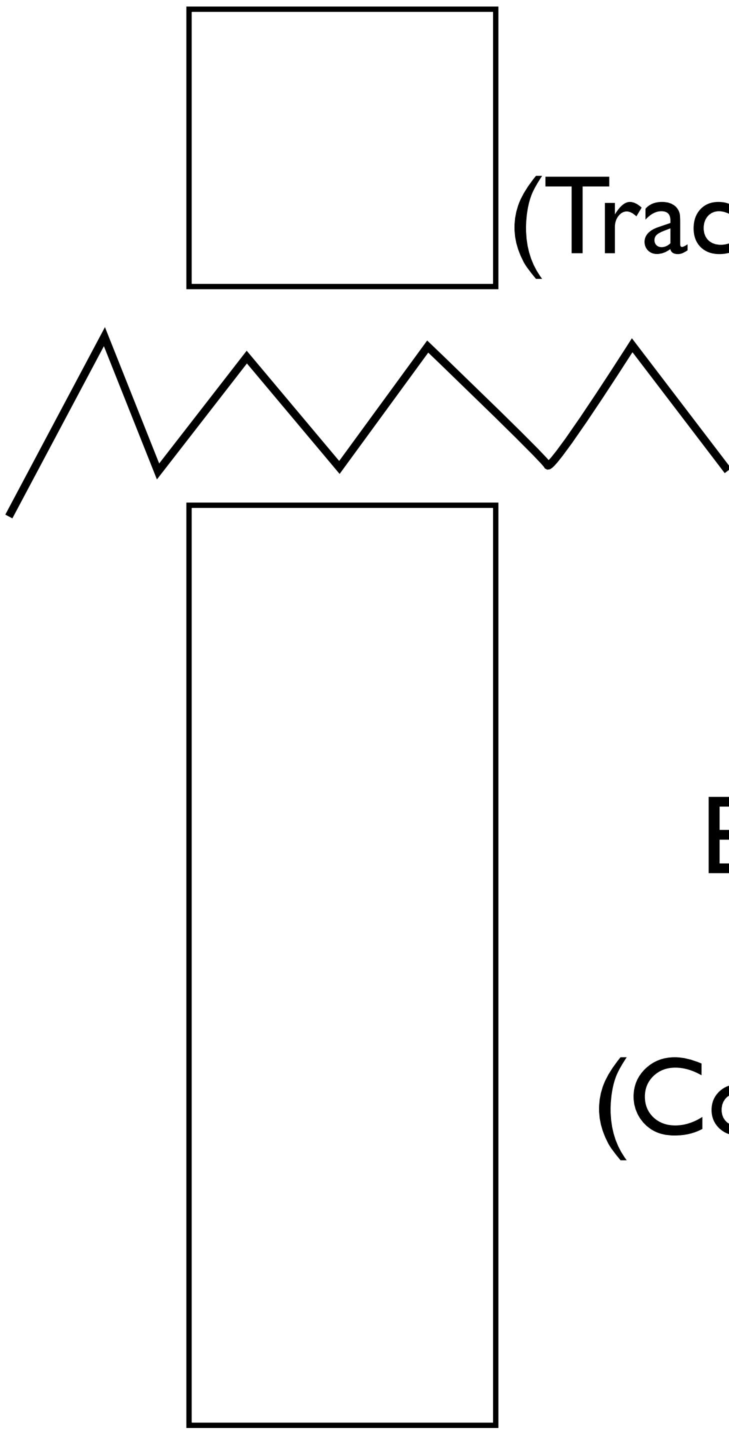
Take <u>median</u> approximation over all x

Count Sketch

Variance is still high.

But good enough for Top-K

Count Sketch



K entries
(Tracked Explicitly)

Everything
Else
(Count Sketch)

For each tuple T:

- Tuple already in Top K?
  - Update count of T
- Otherwise
  - E[count T] > min count of all tuples in Top K?
    - T replaces lowest
- Update Count Sketch

Sketching Algorithms

- Summarize data in a fixed amount of space.
    - Specialized for a specific aggregate.
- Provide probabilistic guarantees.
    - Use properties of how sketch is updated.
    - Use hash and idempotent sketch updates to deduplicate values from the set.

Bibliography

- **Probabilistic Counting Algorithms for Data Bases**
  - **Flajolet and Martin**
- **Finding Frequent Items in Data Streams**
  - **Charikar, Chen and Farach-Colton**
- Online Aggregation
  - Hellerstein, Haas, Wang
- Ripple Joins For Online Aggregation
  - Haas, Hellerstein