Foundations of Data Mining

http://www.cohenwang.com/edith/dataminingclass2017

Instructors:

Edith Cohen

Haim Kaplan

Amos Fiat

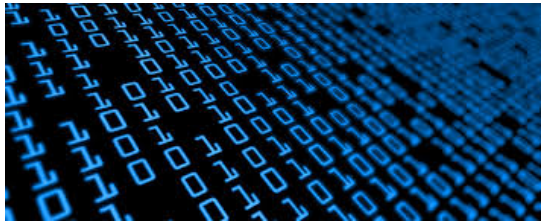
Lecture 1

TEL AVIV UNIVERSITY אוניברסיטת תל אביב

Course logistics

- Tuesdays 16:00-19:00, Sherman 002
- Slides for (most or all) lectures will be posted on the course web page: **http://www.cohenwang.com/edith/dataminingclass2017**
- Office hours: Email instructors to set a time
- Grade: 70% final exam, 30% on 5 problem sets

Data



Collected: network activity, people activity, measurements, search/assistant queries, location, online interactions and transactions, text, media,

Generated (processed) data: parameters in large scale model, partly curated raw data,

- **Scale**: petabytes -> exabytes -> …
- **Diverse formats**: relational, logs, text, media, measurements
- **Location:** distributed, streamed,

Data to Information

Mining and learning from data

- Aggregates, statistics, properties
- Models that allow us to generalize/predict

Scalable (efficient, fast) computation:

- Data available as streamed or distributed (limit data movement for efficiency/privacy)
- Platforms that use computation resources (Map-reduce, Tensor-Flow,…) across scales:
  - GPUs, multi-core CPUs, Data Center, wide area, federated (on device) computing
- Algorithm design:
  - “linear” processing on large data,
  - trade-off accuracy and computation cost

Social issues:

- Privacy, Fairness

Topics for this course

Selection criteria of topics:

- Broad demonstrated applicability
- Promote deeper understanding of concepts
- Simplicity, elegance, principled
- Instructor bias

Topics:

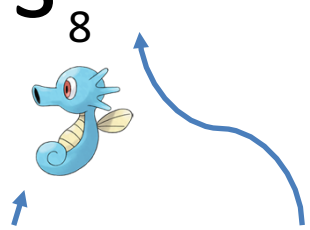
- Data modeled as: key value pairs, metric (vectors, sets), graphs
- Properties, features, statistics of interest
- Summary structures for efficient storage/movement/computation
- Algorithms for distributed/parallel/streamed computation
- Data representations that support generalization (recover missing relations, identify spurious ones)
- Data privacy

Today

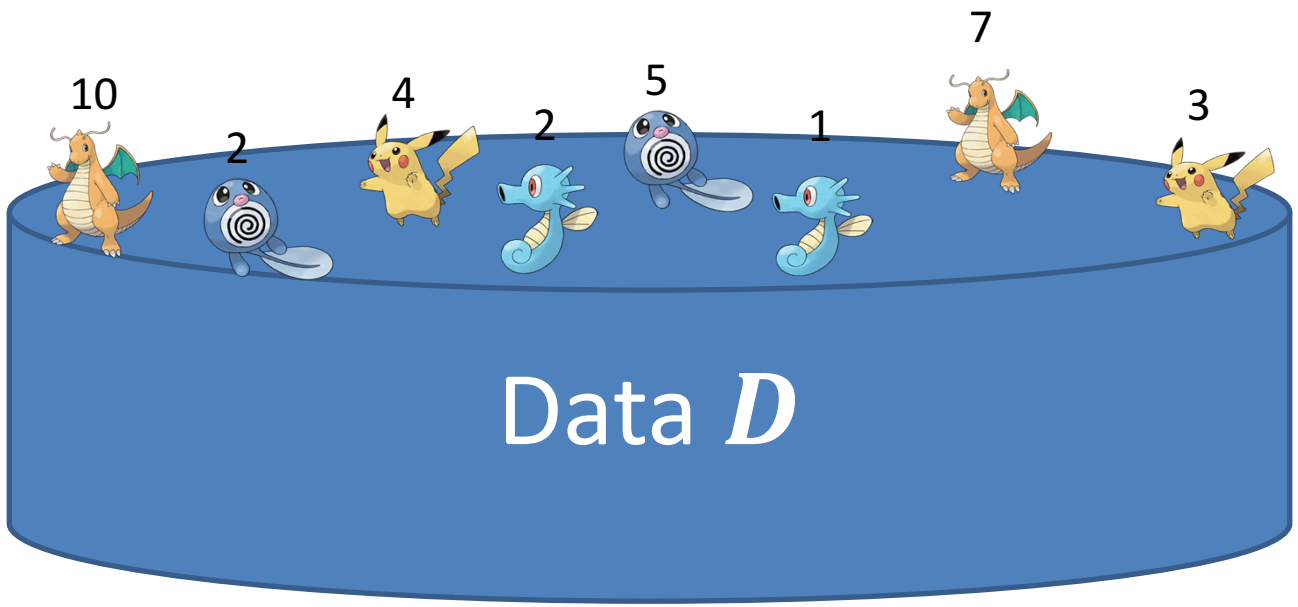
- Key-value pairs data
- Intro to summary structures (sketches)
- Computation over streamed/distributed data

- Frequent keys: The Misra Gries structure
- Set membership: Bloom Filters
- Counting: Morris counters

Key-Value pairs



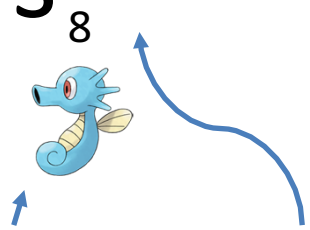
Data element $e \in \mathrm{D}$ has key and value *(e.key,e.value)*



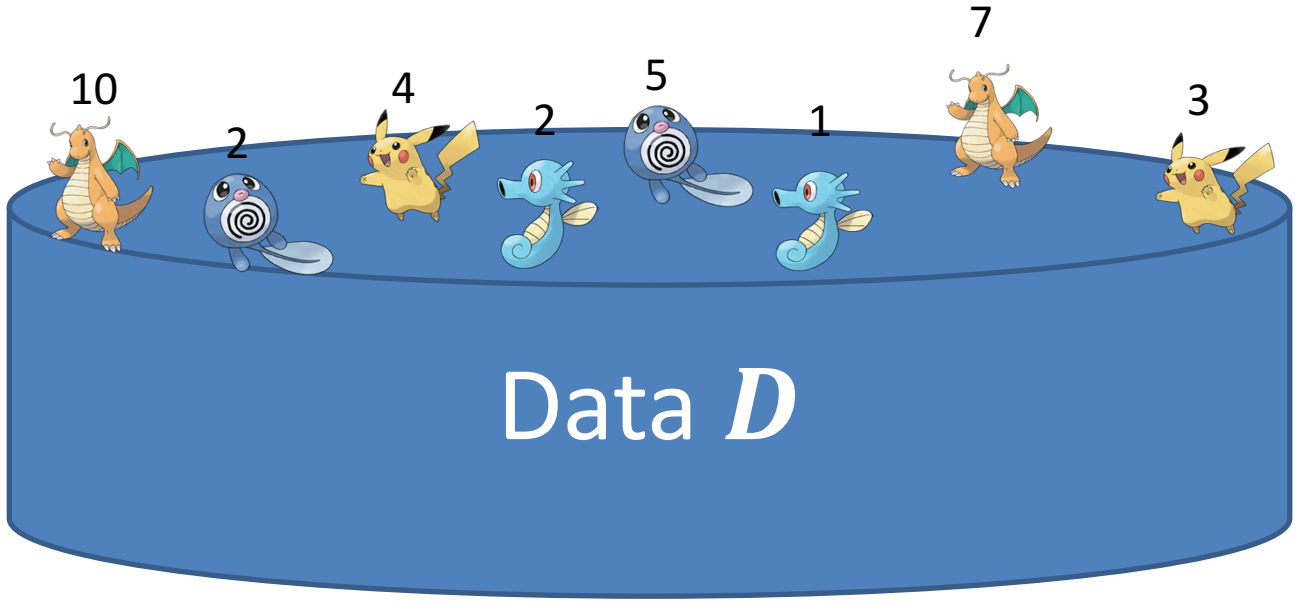
Example data

- Search queries
- IP network packets/flow records
- Online interactions
- Parameter updates (training ML model)

Key-Value pairs



Data element $e \in \mathrm{D}$ has key and value *(e.key,e.value)*



Example tasks/queries

- Sum/Max value
- Membership: Is  in D?
- How many distinct keys?
- Very frequent keys (heavy hitters)

Data access

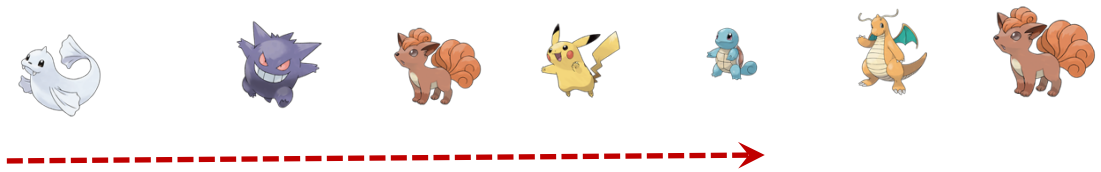
Distributed data/parallel computation

GPUs, CPUs, VMs, Servers, wide area, devices

- Distributed data sources
- Distribute for faster/scalable computation

Challenges: Limit data movement, Support updates to D

Data streams



Data read in one (or few) sequential passes

- Can not be revisited (IP traffic)
- **I/O efficiency** (sequential access is cheaper than random access)

Challenges: ”State” must be much smaller than data size, Support updates to D

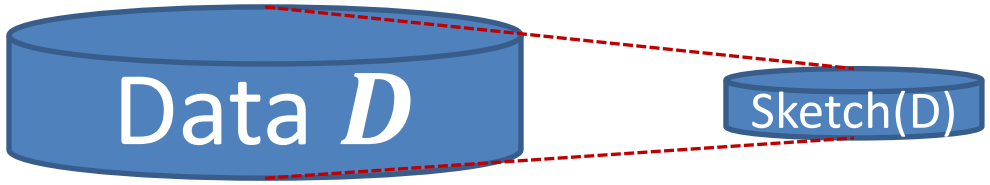
Summary Structures (Sketches)

$\boldsymbol{D}$: data set ; $f(\boldsymbol{D})$: some statistics/properties

$\boldsymbol{Sketch}(\boldsymbol{D})$: A summary of $\boldsymbol{D}$ that acts as "surrogate" and allows us to estimate $f(\boldsymbol{D})$

$\hat{f}()$: estimator we apply to $\boldsymbol{Sketch}(\boldsymbol{D})$ to estimate $f(\boldsymbol{D})$

Examples: random samples, projections, histograms, …



? $f(\boldsymbol{D}) \longrightarrow \hat{f}(\boldsymbol{Sketch}(\boldsymbol{D}))$

Why sketch?

Data can be too large to:

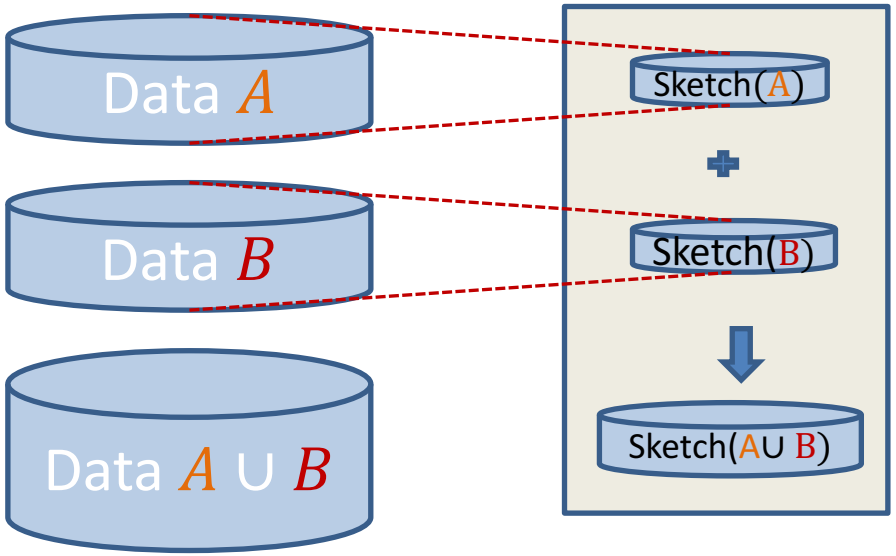
- Store in full for long or even short term
- Transmit
- Slow/costly processing of exact queries
- Data updates do not necessitate full recomputation

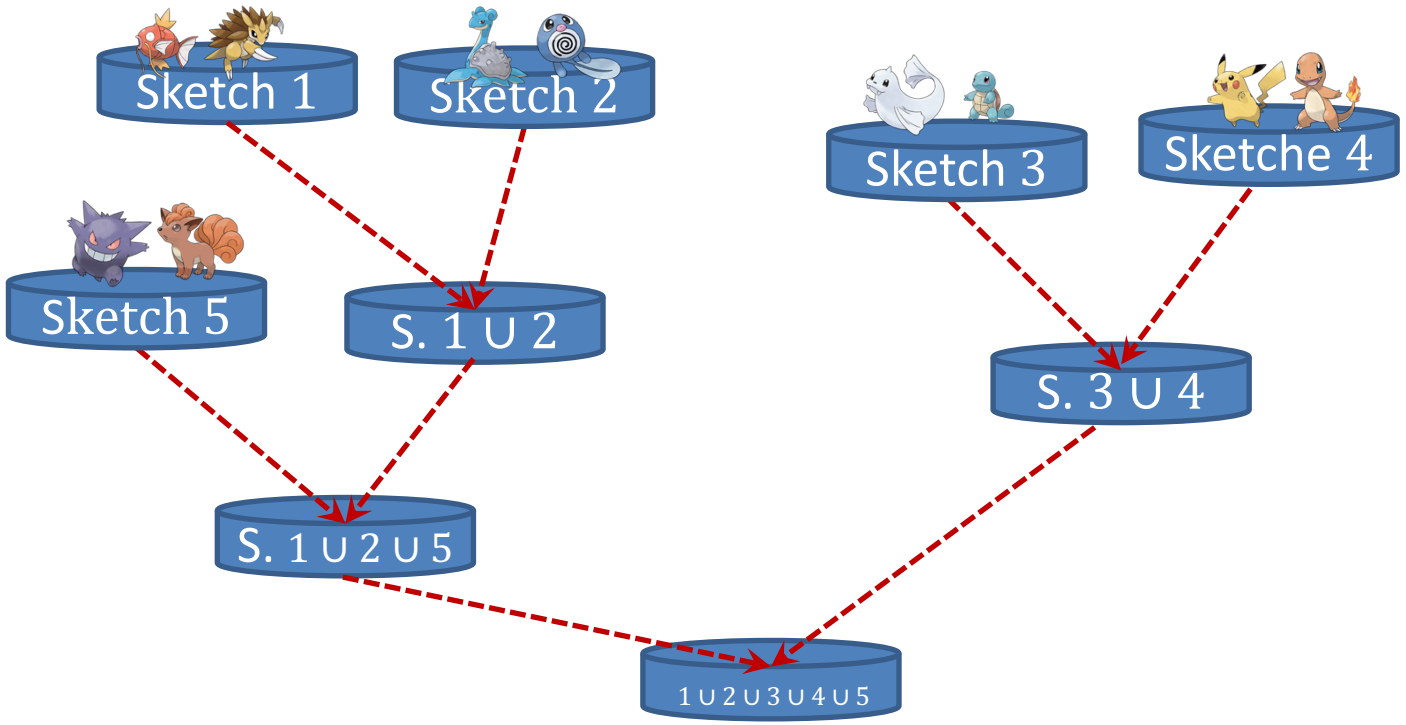
- Multi-objective $f(q, \boldsymbol{D})$ **,** $\hat{f}(q, \boldsymbol{Sketch}(\boldsymbol{D}))$ **sketch** supports multiple query types

Composable sketches

Distributed data/parallelize computation

- Sketch($A \cup B$) **from** Sketch(A) **and** Sketch(B)



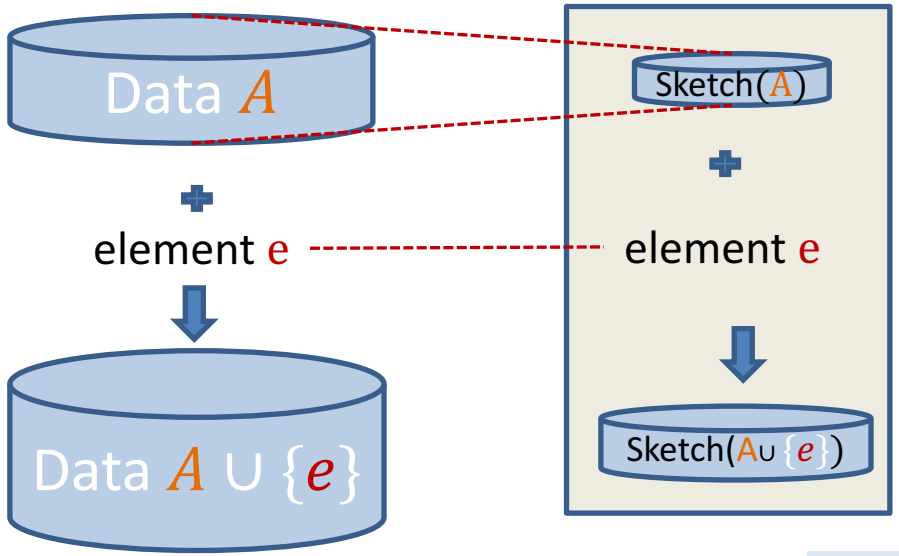


Only sketch structure moves between locations

Suffices to specify merging two sketches

Streaming sketches

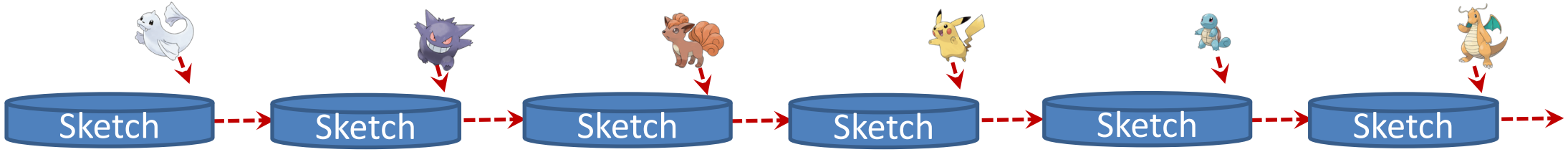
- Sketch($A \cup \{e\}$) **from** Sketch(A) **and element** $\{e\}$



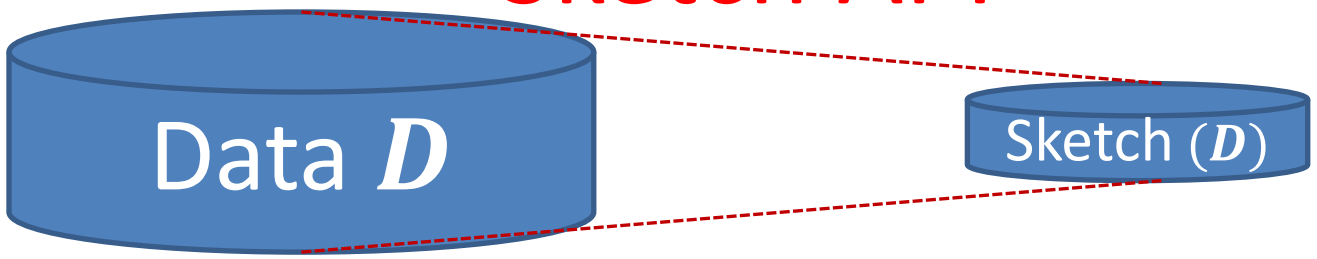
Weaker requirement than fully composable

Only "state" maintained is the sketch structure

Streamed data



Sketch API



Q: $f(\boldsymbol{D})$ **?** $\longrightarrow$ $\hat{f}(\mathrm{Sketch(D)})$

- Initialization $\mathrm{Sketch}(\emptyset)$
- Estimator specification $\hat{f}(\mathrm{Sketch(D)})$
- Merge two sketches $\mathrm{Sketch}(\mathrm{A} \cup \mathrm{B})$ **from** $\mathrm{Sketch}(\mathrm{A})$ **and** $\mathrm{Sketch}(\mathrm{B})$
- Process an element $e = (e.key, e.val)$: $\mathrm{Sketch}(\mathrm{A} \cup e))$ **from** $\mathrm{Sketch}(\mathrm{A})$ **and** e
- Delete e

Υ

optional

- Seek to optimize sketch-size vs. estimate quality

Easy sketches: min, max, sum, …

Element values: 32, 112, 14, 9, 37, 83, 115, 2,

Exact, composable, Sketch is just a single register s:

Sum

- **Initialize:** $s \leftarrow 0$
- **Process element** e **:** $s \leftarrow s + e.val$
- **Merge** s, s': $s \leftarrow s + s'$
- **Delete element** e **:** $s \leftarrow s - e.val$
- **Query:** return s

Max

- **Initialize:** $s \leftarrow 0$
- **Process element** e **:** $s \leftarrow \max(s, e.val)$
- **Merge** s, s': $s \leftarrow \max(s, s')$
- **Query:** return s

No delete support

Frequent Keys

- Data is streamed or distributed
- Very large #distinct keys, huge #elements
- **Find the keys that occur very often**

Occur in 3/11 elements

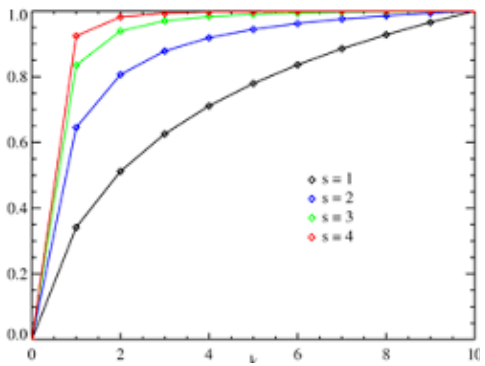
Example Applications:

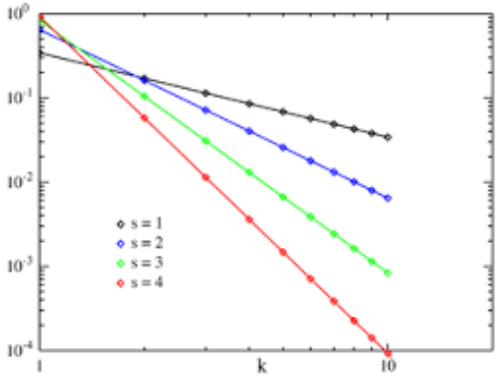
- Networking: Find "elephant" IP flows
- Search engines: Find the most frequent queries
- Text analysis: Frequent terms

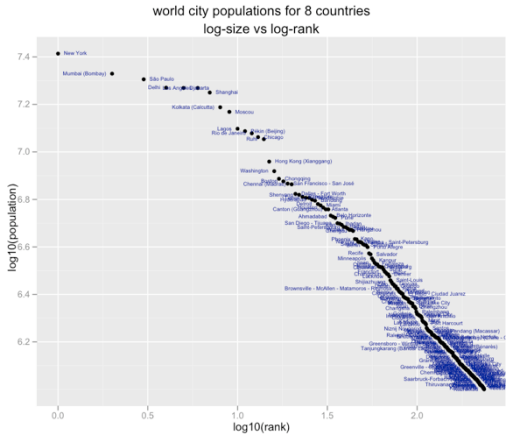
Zipf law: Frequency of ith heaviest key $\propto i^{-s}$

Say top 10% keys in 90% of elements

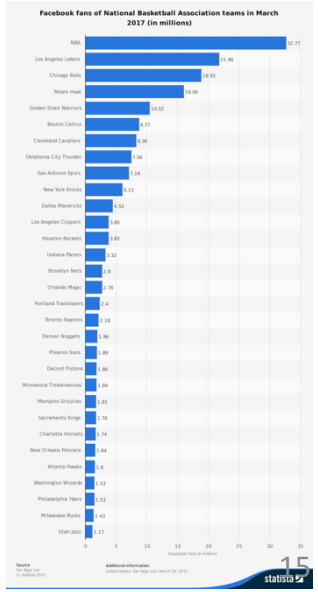
wikipedia







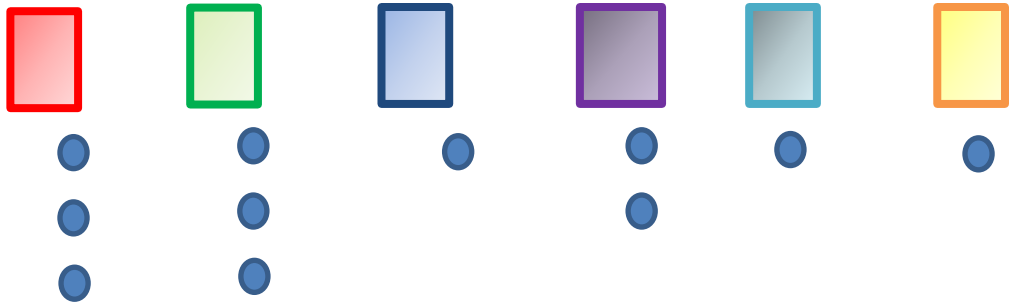
https://brenocon.com/blog/2009/05



Frequent Keys: Exact Solution

Exact solution:

- Create a counter for each distinct key on its first occurrence
- When processing an element with key x , increment the counter of x



Properties: Fully composable, exact, even supports deletions, recovers all frequencies

Problem: Structure size is $n =$ number of distinct keys. What can we do with size $\mathrm{k} \ll n$?

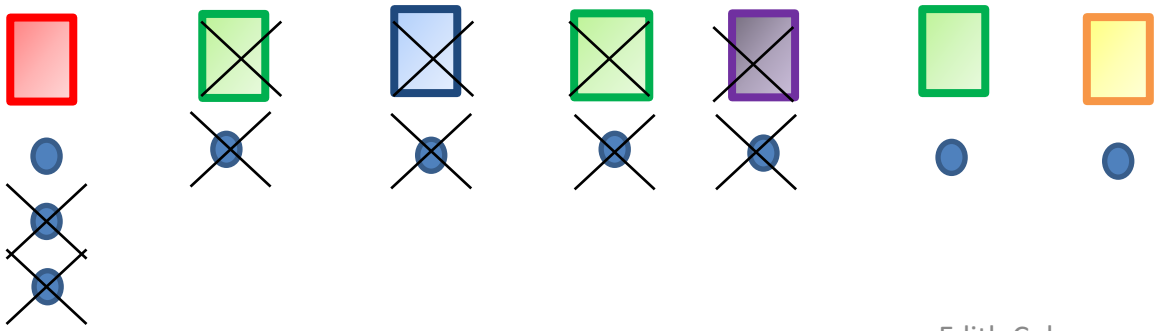
Solution: Sketch that got re-discovered many times [MG1982,DLM2002,KSP2003,MAA2006]

Frequent Keys: Streaming sketch [Misra Gries 1982]

Sketch size parameter k: Use (at most) k counters indexed by keys. Initially, no counters

Processing an element with key x

- If we already have a counter for x, increment it
- Else, If there is no counter, but there are fewer than k counters, create a counter for x initialized to $\mathbf{1}$.
- Else, decrease all counters by $\mathbf{1}$. Remove $\mathbf{0}$ counters.



Query: #occurrences of x?

- If we have a counter for x, return its value
- Else, return $\mathbf{0}$.

Clearly an under-estimate.
What can we say precisely?

$n = 6$ #distinct
$k = 3$ #structure size
$m = 11$ #element

MG sketch: Analysis

<u>Lemma:</u> Estimate is smaller than true count by at most $\frac{m-m'}{k+1}$

m' : Sum of counters in structure; m: #elements in stream; k: structure size

We charge each "missed count" to a "decrease" step:

- If key in structure, any decrease in count is due to "decrease" step.
- Element processed and not counted results in decrease step.

We bound the number of "decrease" steps

Each decrease step removes k "counts" from structure, together with input element, it results in $k+1$ "uncounted" elements.

$\Rightarrow$ Number of decrement steps $\leq \frac{m-m'}{k+1}$

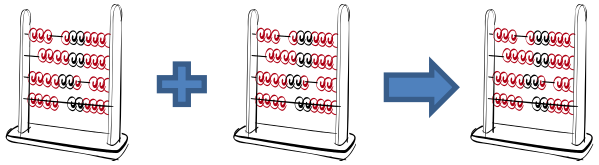
MG sketch: Analysis (contd.)

Estimate is smaller than true count by at most $\boldsymbol{\frac{m-m'}{k+1}}$

$\Rightarrow$ We get good estimates for x with frequency $\gg \frac{m-m'}{k+1}$

- Error bound is inversely proportional to k. Typical tradeoff of sketch-size and quality of estimate.
- Error bound can be computed with sketch: Track m (element count), know m' (can be computed from structure) and k.
- MG works because typical frequency distributions have few very popular keys "Zipf law"

Making MG fully Composable: Merging two MG sketches [MAA 2006, ACHPWY 2012]



Basic merge:

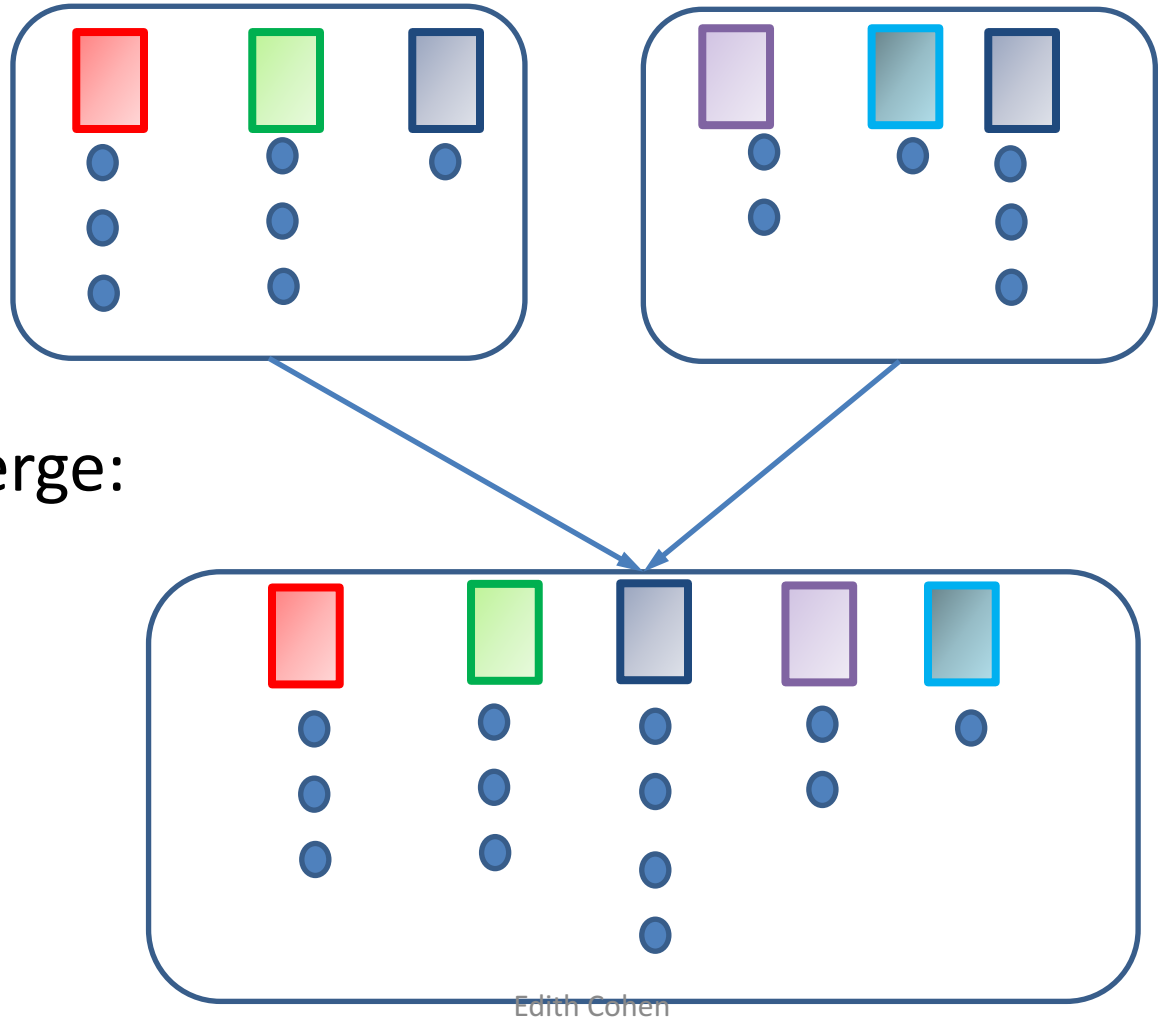
- If a key x is in both structures, keep one counter with sum of the two counts
- If a key x is in one structure, keep the counter

Reduce: If there are more than k counters

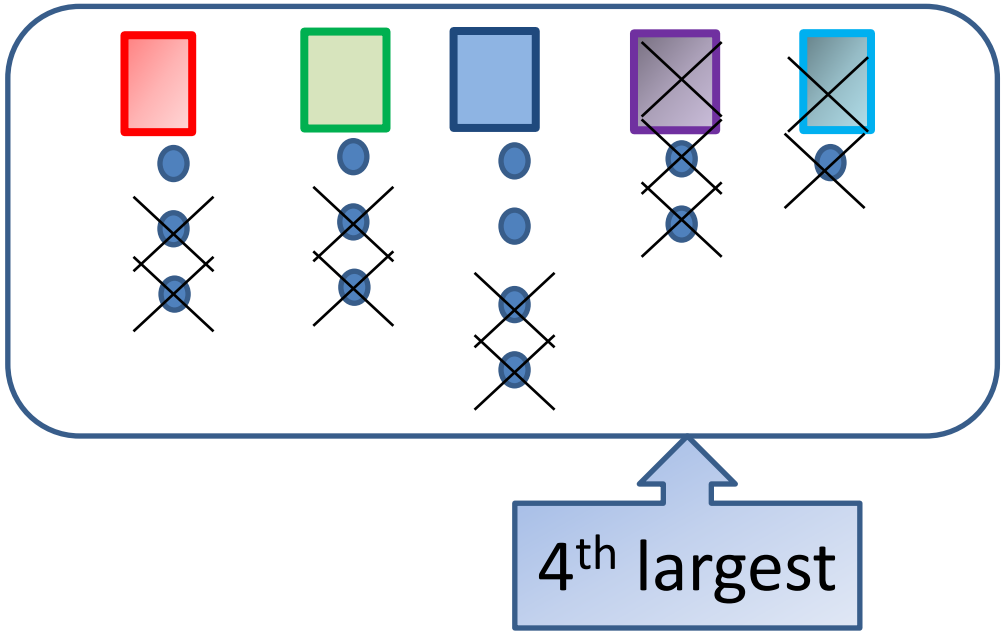
- Take the $(k+1)^{\text{th}}$ largest counter
- Subtract its value from all other counters
- Delete non-positive counters

Merging two Misra Gries Sketches

Basic Merge:



Merging two Misra Gries Summaries



Reduce since there are more than $k = 3$ counters :

- Take the $(k+1)^{\text{th}}$ = 4th largest counter
- Subtract its value (2) from all other counters
- Delete non-positive counters

Merging MG Summaries: Correctness

Claim: Final merged sketch has at most k counters

Proof: We subtract the $(k+1)^{\text{th}}$ largest from everything, so at most the k largest can remain positive.

Claim: For each key, merged sketch count is smaller than true count by at most $\frac{m-m'}{k+1}$

Merging MG Summaries: Correctness

Claim: For each key, merged sketch count is smaller than true count by at most $\frac{m-m'}{k+1}$

Proof: "Counts" for key x can be missed in part1, part2, or in the reduce component of the merge

We add up the bounds on the misses

Part 1:

Total elements: m_1

Count in structure: $m_1{}'$

Count missed: $\leq \frac{m_1-m_1{}'}{k+1}$

Part 2:

Total elements: m_2

Count in structure: $m_2{}'$

Count missed: $\leq \frac{m_2-m_2{}'}{k+1}$

"Reduce" missed count per key is at most $\boldsymbol{R} =$ the $(\boldsymbol{k}+\mathbf{1})^{\text{th}}$ largest count before reduce

Merging MG Summaries: Correctness

$\Rightarrow$ "Count missed" of one key in merged sketch is at most

$$\frac{m_1 - m_1'}{k+1} + \frac{m_2 - m_2'}{k+1} + R$$

Part 1:
Total elements: m_1
Count in structure: $m_1{}'$
Count missed: $\leq \frac{m_1 - m_1'}{k+1}$

Part 2:
Total elements: m_2
Count in structure: $m_2{}'$
Count missed: $\leq \frac{m_2 - m_2'}{k+1}$

"Reduce" missed count per key is at most $R =$ the $(k+1)^{\text{th}}$ largest count before reduce

Merging MG Summaries: Correctness

Counted elements in structure:

- After basic merge and before reduce: $m_1' + m_2{}'$
- After reduce: m'

Claim: $\mathrm{m}_1' + \mathrm{m}_2' - m' \geq R(k+1)$

Proof: R are erased in the reduce step in each of the $k+1$ largest counters. Maybe more in smaller counters.

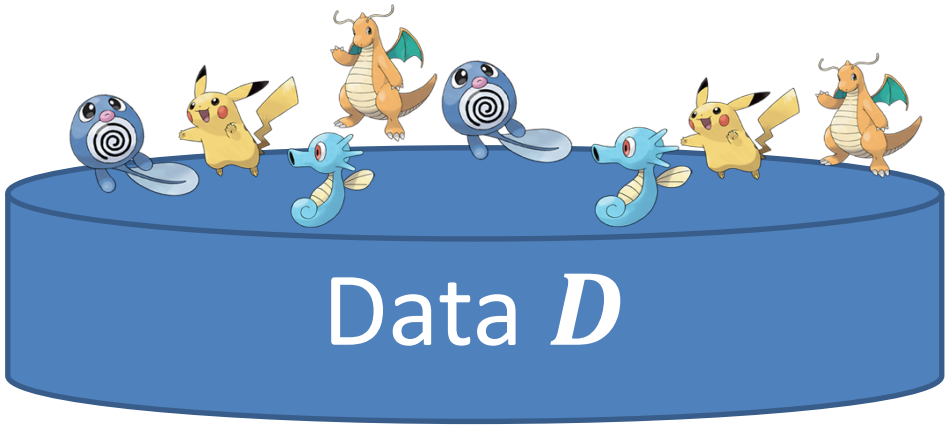
"Count missed" of one key is at most

$$\frac{m_1 - m_1{}'}{k+1} + \frac{m_2 - m_2{}'}{k+1} + R \leq \frac{1}{k+1}(m_1 + m_2 - m') = \frac{m - m'}{k+1}$$

Probabilistic structures

- Misra Gries is a *deterministic* structure
- The outcome is determined uniquely by the input
- Probabilistic structures/algorithms can be much more powerful
  - *Provide privacy/robustness to outliers*
  - *Provide efficiency/size*

Set membership



- Data is streamed or distributed
- Very large #distinct keys, huge #elements, large representation of keys

Structure that supports membership queries: Is  in D?

Example applications:

- Spell checker: Insert a corpus of words. Check if word is in corpus.
- Web crawler: Insert all urls that were visited. Check if current url was explored.
- Distributed caches: Maintain a "summary" of keys of cached resources. Send requests to a cache that has the resource.
- Blacklisted IP addresses: Intercept traffic from blacklisted sources

Exact solution: Dictionary (hash map) structure. **Problem:** stores representation of all keys

Set membership: Bloom Filters [Bloom 1970]

- Very popular in many applications
- Probabilistic data structure
- Reduces representation size to few bits (8) per key
- False positives possible, no false negatives
- Tradeoff between size and false positive rate
- Composable
- Analysis relies on having independent random hash functions (practice work well, theoretical issues)

Independent Random Hash Functions

Simplified and Idealized

Domain $\boldsymbol{D}$ of keys; probability distribution $\boldsymbol{F}$ over $\boldsymbol{R}$

Distribution $\boldsymbol{H}$ of *hash functions* $h: \boldsymbol{D} \rightarrow \boldsymbol{R}$ with the following properties:

Over $h \sim H$

- For each $x \in \boldsymbol{D}, h(x) \sim \boldsymbol{F}$ (over $h \sim H$)
- $h(x)$ are independent for different keys $x \in \boldsymbol{D}$

We use random hash functions as a way to have random draws with "memory": Attach a "permanent" random value to a key

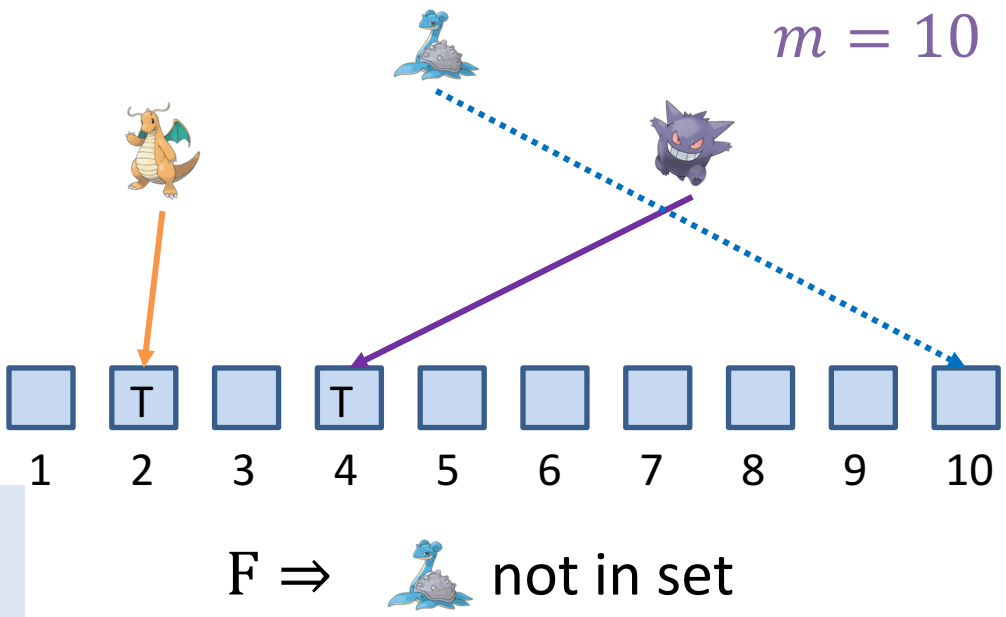
Set membership warmup: Hash solution

Parameter: m
Structure: Boolean array of size m
Random hash function h where $h(x) \sim U[1, \dots, m]$

<u>Initialize</u>:
Declare boolean array S of size m;
For $i = 1, \dots, m$: $S[i] \leftarrow F$

<u>Process element with key x</u>:
$$S[h(x)] \leftarrow T$$

<u>Membership query for x</u>:
Return $S[h(x)]$

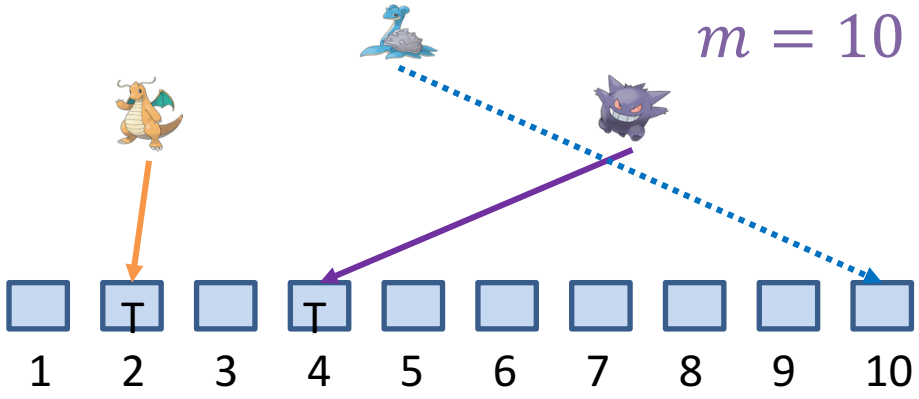


<u>Merge:</u> Two structures of same size and same hash function. Take bitwise OR

Hash solution: Probability of a false positive

m: Structure size; n number of distinct keys inserted

$\mathrm{b} = \frac{m}{\mathrm{n}}$, number of bits we use in structure per distinct key in data



Probability ε of false positive for x :

Probability of $h(x)$ hitting an occupied cell:

$$\varepsilon = \Pr_{h \sim H}[S[h(x)] = T] \approx \frac{\mathrm{n}}{m} = \frac{1}{\mathrm{b}}$$

Example:

$$\varepsilon = 0.02 \implies b = 50$$

Too high for many applications!! (IP address is 32 bits…)

Can we get a better tradeoff between ε and b ?

Set membership: Bloom Filters [Bloom 1970]

Two parameters: m and k

Structure: Boolean array of size m

Independent hash functions $h_1, h_2, \dots, h_k$ where $h_i(x) \sim U[1, \dots, m]$

<u>Initialize</u>:

Declare boolean array S of size m;

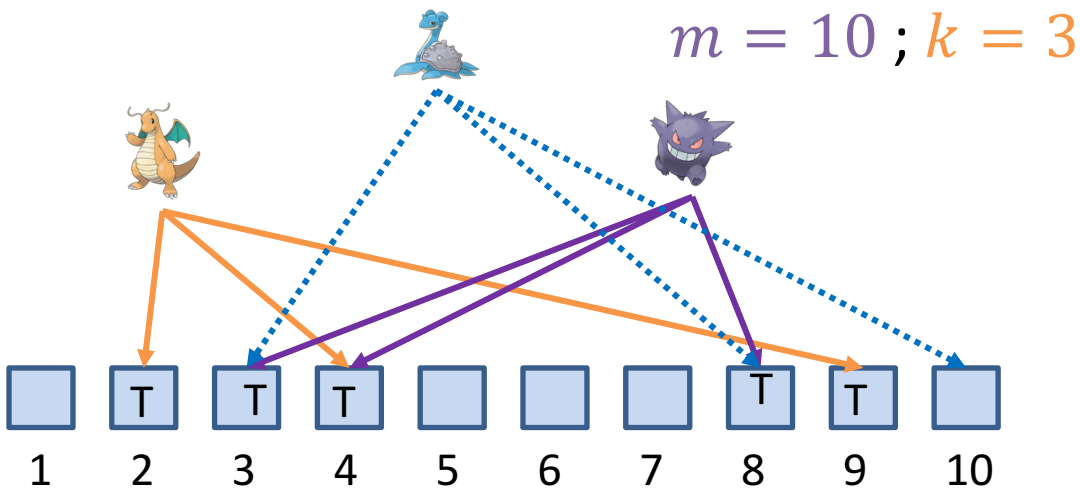
For $i = 1, \dots, m$: $S[i] \leftarrow F$

<u>Process element with key x</u>:

For $i = 1, \dots, k$: $S[h_i(x)] \leftarrow T$

<u>Membership query for x</u>:

Return $S[h_1(x)]$ **and** $S[h_2(x)]$ **and** $\cdots$ $S[h_k(x)]$



$\mathrm{T} \wedge \mathrm{T} \wedge \mathrm{F} = \mathrm{F} \Rightarrow$  not in set

<u>Merge:</u> Two structures of same size and same set of hash functions. Take bitwise OR

Bloom Filters Analysis: Probability of a false positive

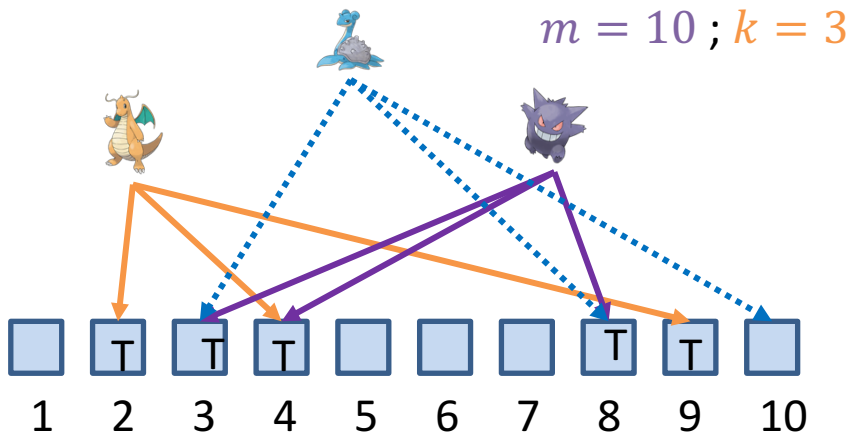
m: Structure size ; k: number of hash functions ; n number of distinct keys inserted

Probability of $h_i(x)$ NOT hitting a particular cell j:

$$\Pr_{h\sim H}[h_i(x) \neq j] = (1 - \frac{1}{\mathrm{m}})$$

Probability that cell j is F is that none of the nk "dart throws" hits cell j:

$$\Pr_{h\sim H}[S[j] = F] = \left(1 - \frac{1}{\mathrm{m}}\right)^{\mathrm{kn}}$$



$\mathrm{T} \wedge \mathrm{T} \wedge \mathrm{F} = \mathrm{F} \Rightarrow$ not in set

A false positive occurs for x when **all** k cells $h_i(x)$ for $i = 1, \ldots, k$ are T:

$$\varepsilon = \prod_{i=1,\ldots,k} (1 - \Pr_{h\sim H}[S[h_i(x)] = F]) = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^{k}$$

* Assume $k \ll m$ so $h_i(x)$ for different $i = 1, \ldots, k$ are very likely to be distinct

Bloom Filters: Probability of a false positive (contd)

m: Structure size ; k: number of hash functions ; n number of distinct keys inserted

False positive probability:

$$\varepsilon \le \left(1 - \underbrace{\left(1 - \frac{1}{m}\right)^{kn}}\right)^{k} \approx \left(1 - e^{-\frac{kn}{m}}\right)^{k} = \left(1 - e^{-\frac{k}{b}}\right)^{k}$$

$$\lim_{i \to \infty} \left(1 - \frac{1}{i}\right)^{i} = \frac{1}{e}$$

$$\left(1 - \frac{1}{\mathrm{m}}\right)^{\mathrm{kn}} = \left(\left(1 - \frac{1}{\mathrm{m}}\right)^{m}\right)^{\frac{kn}{m}} \approx \left(\frac{1}{e}\right)^{\frac{kn}{m}} = e^{-\frac{kn}{m}}$$

We can see that FP probability decreases with m

!! FP probability depends on $\mathrm{b} = \frac{m}{\mathrm{n}}$, number of bits we use per distinct key

Given b, which k minimizes the FP probability ε?

Bloom Filters: Probability of a false positive (contd)

k: number of hash functions
m: Structure size
n: number of distinct keys inserted
$b = \frac{m}{n}$: bits per key

False positive probability ε (upper bound):

$$\varepsilon \leq \left(1 - e^{-\frac{k}{b}}\right)^k$$

Given b, which k minimizes the FP probability?

$$k \approx \ln(2) b \approx 0.7\, b$$

$$\varepsilon \approx \left(\frac{1}{2}\right)^{b \ln 2}$$

!! FP error decreases exponentially in b
(recall $\varepsilon = \frac{1}{b}$ for $k = 1$)

Compute b for desired FP error ε:

$$b \approx 1.44 \log_2 \frac{1}{\varepsilon}$$

Example:
$b = 8$; $k = 6$; $\varepsilon \approx 0.02$

Quick review: Random Variables

Random variable X

Probability Density Function (PDF) $f(x)$:

- **Properties:** $f(x) \geq 0 \quad \int_{-\infty}^{\infty} f(x)\, dx = 1$

- Cumulative Distribution Function (CDF)

$F(t) = \int_{-\infty}^{t} f(x)\, dx$: probability that $X \leq t$

- **Properties:** $F \in [0,1]$ monotone non-decreasing

Quick review: Expectation

- ***Expectation***: "average" value of $\boldsymbol{X}$:

$$\mu_X \equiv E[X] = \int_{-\infty}^{\infty} x f(x)\, dx$$

- ***Linearity of Expectation***:

$$E[aX + b] = aE[X] + b$$

For random variables $X_1, X_2, X_3, \ldots, X_k$

$$E\left[\sum_{i=1}^{k} X_i\right] = \sum_{i=1}^{k} E[X_i]$$

Quick review: Variance

- *Variance*

$$\mathbf{Var}[X] \equiv \sigma_X^2 = E[(X-\mu)^2] = \int_{-\infty}^{\infty} (x-\mu)^2 f(x)dx$$

- Useful relations: $\sigma_X^2 = E[X^2] - \mu_X^2$

  $\mathbf{Var}[aX+b] = a^2 \mathbf{Var}[X]$

- The *standard deviation* is $\sigma_X = \sqrt{\mathbf{Var}[X]}$
- *Coefficient of Variation* $\frac{\sigma}{\mu}$ (normalized s.d.)

Quick review: Covariance

Measure of joint variability of two random variables) X, Y

$$\mathbf{Cov}[X,Y] = \sigma_{XY} = E[(X-\mu_X)(Y-\mu_Y)]$$
$$= \mathbf{E}[\mathbf{XY}] - \mu_X\mu_Y$$

- X, Y are independent $\Longrightarrow$ $\sigma_{XY} = 0$
- Variance of the sum of $X_1, X_2, \ldots, X_k$

$$\mathbf{Var}\left[\sum_{i=1}^{k} X_i\right] = \sum_{i=1}^{k}\sum_{j=1}^{k} \mathbf{Cov}[X_i, X_j] = \sum_{i=1}^{k} \mathbf{Var}[X_i] + \cancel{\sum_{i\neq j}^{k} \mathbf{Cov}[X_i, X_j]}$$

When (pairwise) independent

Quick Review: Estimators

A function $\hat{f}(S)$ applied to a probabilistic sketch S of data D to estimate a property/statistics $f(\mathrm{D})$ of the data D

- ***Error*** (random variable) $\mathrm{err}(\hat{f}) = \hat{f}(S) - f(D)$; ***Relative Error*** $\frac{\mathrm{err}(\hat{f})}{\mathrm{f(D)}}$
- ***Bias*** $\mathrm{Bias}[\hat{f}] = \mathrm{E}[\mathrm{err}(\hat{f})] = E[\hat{f}] - f(D)$
  - When Bias $=$ 0 estimator is ***unbiased***
- ***Mean Square Error (MSE):***

$$\mathrm{E}\left[\mathrm{err}(\hat{f})^2\right] = Var[\hat{f}] + \mathrm{Bias}[\hat{f}]^2$$

- ***Root Mean Square Error (RMSE):*** $\sqrt{MSE}$
- ***Normalized Root Mean Square Error (NRMSE):*** $\frac{\sqrt{MSE}}{f(D)}$

Simple Counting (revisited)

1, 1, 1, 1, 1, 1, 1, 1,

Initialize: $s \leftarrow 0$
Process element: $s \leftarrow s + 1$
Merge s, s′: $s \leftarrow s + s'$

Exact count: Size (bits) is $\lceil \log_2 n \rceil$ where n is the current count.

Can we count with *fewer* bits ?
Have to settle for an ***approximate*** count…

Applications: We have very many quantities to count, and fast memory is scarce (say, inside a backbone router,) or bandwidth is scarce (distributed training of a large ML model)

Morris Counter [Morris 1978]

Probabilistic stream counter: Maintain e "$\log n$" instead of n, use $\log \log n$ bits

- **Initialize:** $\mathbf{s = 0}$
- **Increment**: Increment $\mathbf{s}$ with probability $\mathbf{2^{-s}}$
- **Query**: Return $\mathbf{2^{s} - 1}$

$n = 10^9$,
Exact: $\log_2 10^9 \approx 30$bits
$\log_2 \log_2 10^9 \approx 5$bits

Stream:		1,	1,	1,	1,	1,	1,	1,	1,
Count n:		1,	2,	3,	4,	5,	6,	7,	8,
$p = 2^{-x}$:	1	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{8}$
Counter x :	0	1	1	2	2	2	2	3	3
Estimate $\hat{n}$:	0	1	1	3	3	3	3	7	7

Morris Counter: Unbiasedness

- **Initialize:** $\mathbf{s = 0}$
- **Increment**: $\mathbf{s \leftarrow s + 1}$ with probability $\mathbf{2^{-s}}$
- **Query**: Return $\mathbf{2^s - 1}$

- When $n = 0$, $s = 0$, estimate is $\hat{n} = \mathbf{2^0 - 1 = 0}$
- When $n = 1$, $s = 1$, estimate is $\hat{n} = \mathbf{2^1 - 1 = 1}$
- When $n = 2$,

  with $p = \frac{1}{2}$, $\mathrm{s} = \mathbf{1}$, $\hat{n} = \mathbf{1}$

  with $p = \frac{1}{2}$, $\boldsymbol{s} = \mathbf{2}$, $\hat{n} = \mathbf{2^2 - 1 = 3}$

Expectation: $\mathrm{E}[\hat{\boldsymbol{n}}] = \frac{\mathbf{1}}{\mathbf{2}} * \mathbf{1} + \frac{\mathbf{1}}{\mathbf{2}} * \mathbf{3} = \mathbf{2}$

- $\boldsymbol{n} = \mathbf{3, 4, 5} \ldots$ by induction….

Morris Counter: Unbiasedness (contd)

- **Initialize:** $\mathbf{s = 0}$
- **Increment**: $\mathbf{s \leftarrow s + 1}$ with probability $\mathbf{2^{-s}}$
- **Query**: Return $\mathbf{2^s - 1}$

It suffices to show that the expected increase of the estimate is always 1

- Suppose the counter value is s
- We increase with probability $\mathbf{2^{-s}}$
- The expected increase in the estimate is
$2^{-s}((2^{s+1} - 1) - (2^{s} - 1)) + (1 - 2^{-s})0 = 2^{-s}\, 2^{s} = 1$

Morris Counter: Variance

How good is our estimate?

X_n : random variable of counter with input n

- Our estimate is the random variable $\hat{n} = 2^{X_n} - 1$

$$\mathrm{Var}[\hat{n}] = Var[\hat{n}+1] = E[(\hat{n}+1)^2] - E[\hat{n}+1]^2$$
$$= E[2^{2X_n}] - (n+1)^2$$

- We can show by induction $E[2^{2X_n}] = \frac{3}{2}n^2 + \frac{3}{2}n + 1$
- This means $Var[\hat{n}] \approx \frac{1}{2}n^2$ and CV $= \frac{\sigma}{\mu} \approx \frac{1}{\sqrt{2}}$ (=NRMSE since unbiased)

How to reduce the error ?

Reducing variance by averaging

$\boldsymbol{k}$ (pairwise) *independent* unbiased estimates $\boldsymbol{Z_i}$ with expectation $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma^2}$.

The average estimator $\boldsymbol{\widehat{n'} = \frac{\sum_{i=1}^{k} Z_i}{k}}$

- Expectation: $E[\widehat{n'}] = \frac{1}{k}\sum_{i=1}^{k} E[Z_i] = \frac{1}{k}k\mu = \mu$
- Variance: $\left(\frac{1}{k}\right)^2 \sum_{i=1}^{k} \mathrm{Var}[Z_i] = \left(\frac{1}{k}\right)^2 k\sigma^2 = \frac{\sigma^2}{k}$ ($\times k$ decrease)
- CV : $\boldsymbol{\frac{\sigma}{\mu}}$ ($\times\sqrt{k}$ decrease)

Morris Counter: Reducing variance (generic method)

Morris counter: $Var[\hat{n}] = \sigma^2 \approx \frac{1}{2}n^2$ and CV $= \frac{\sigma}{\mu} \approx \frac{1}{\sqrt{2}}$

- Use $\boldsymbol{k}$ *independent* counters $\boldsymbol{y_1, y_2, \dots, y_k}$
- Compute estimates $\boldsymbol{Z_i = 2^{y_i} - 1}$
- Average the estimates $\boldsymbol{\widehat{n'} = \frac{\sum_{i=1}^{k} Z_i}{k}}$
- NRMSE=CV $= \frac{\sigma}{\mu} \approx \frac{1}{\sqrt{2k}} = \varepsilon$

Sketch size (bits): $k \log\log n = \frac{1}{2}\varepsilon^{-2} \log\log n$

Can we get a better tradeoff of sketch size and NRMSE ε ?

Morris Counter: Reducing variance (dedicated method)

base change [Morris 1978+Flajolet 1985]

Morris counter: $Var[\hat{n}] = \sigma^2 \approx \frac{1}{2}n^2$ and CV $= \frac{\sigma}{\mu} \approx \frac{1}{\sqrt{2}}$

Single counter with base change –

IDEA: Change base $\mathbf{2}$ (count $\mathbf{log_2}\, \boldsymbol{n}$) to $\mathbf{1} + \boldsymbol{b}$ (count $\mathbf{log_{1+b}}\, \boldsymbol{n}$)

- Estimate: Return $(1+b)^s - 1$
- Increment:
  - Increase counter $\boldsymbol{s}$ by maximum amount so estimate increase $= 1 - \Delta \leq 1$.
  - Increment $\boldsymbol{s}$ with probability $\Delta\, \mathrm{b}^{-1}(1+\boldsymbol{b})^{-\boldsymbol{s}}$

For $\boldsymbol{b}$ closer to 0, we increase accuracy but also increase counter size.

We analyze a more general method

Weighted Morris Counter [C'15]

5, 14, 1, 7, 18, 9, 121, 17,

weighted values, composable, size/quality tuned by base parameter b

- Initialize: $s \leftarrow 0$
- Estimate: return $(1+b)^s - 1$

- Add V or merge with a Morris sketch $s_2 \leq s$ $(V = (1+b)^{s_2} - 1)$:
  - Increase s by max amount so that estimate increase by $Z \leq V$
  - $\Delta \leftarrow \mathrm{V} - \mathrm{Z}$; Increment s with probability $\frac{\Delta}{b(1+b)^s}$

We can show $\mathrm{Var}[\hat{n}] \leq bn(n+1) \implies \mathrm{CV} \leq \sqrt{b}\sqrt{1+\frac{1}{n}} \implies$ Choose $b = \varepsilon^2$

Sketch size: $\log_2 \log_{1+b} n \approx \log_2 \left(\frac{\log_2 n}{b \log_2 e}\right) \leq \log_2 \log_2 n + 2\log_2 \frac{1}{\varepsilon}$

$n = 10^9, \varepsilon = 0.1$
Exact: $\log_2 10^9 \approx 30$bits
Ave Morris: ≈ 250 bits
W-Morris: ≈ 12 bits

!! Much better than the averaging structure $\frac{1}{2}\varepsilon^{-2} \log\log n$

Weighted Morris Counter: Unbiasedness

- Initialize: $s \leftarrow 0$
- Estimate: return $(1+b)^s - 1$
- Add V or merge with a Morris sketch $s_2 \leq s$ $(V = (1+b)^{s_2} - 1)$:
  - Increase s by max amount so that estimate increase by $Z \leq V$
  - $\Delta \leftarrow \mathrm{V} - \mathrm{Z}$; Increment s with probability $\frac{\Delta}{b(1+b)^s}$

We show that the expected increase in the estimate when adding V is equal to V.
The increase has two components, deterministic, and probabilistic:

- Deterministic: We set $s \leftarrow s + \max\{\mathrm{i} \geq 0 \mid (1+\mathrm{b})^{\mathrm{s+i}} - (1+\mathrm{b})^{\mathrm{s}} \leq V\}$. This step increased the estimate by $Z = (1+\mathrm{b})^{\mathrm{s+i}} - (1+\mathrm{b})^{\mathrm{s}}$
- We then probabilistically increment s to account for $\Delta = \mathrm{V} - \mathrm{Z}$: The estimate increase is $(1+\mathrm{b})^{\mathrm{s}+1} - (1+\mathrm{b})^{\mathrm{s}} = \mathrm{b}(1+\mathrm{b})^{\mathrm{s}}$ with probability $\mathrm{p} = \frac{\Delta}{b(1+b)^s}$ and is 0 otherwise.

  Therefore, the expectation is $p\mathrm{b}(1+\mathrm{b})^{\mathrm{s}} = \Delta$.

Weighted Morris Counter: Variance bound

Estimate: $(1+b)^s - 1$

Add Δ : Increment s with probability $\frac{\Delta}{b(1+b)^s}$

Consider all data values $\mathrm{V_i}$ and the corresponding random variables A_i that is the increase in the estimate. Note that by definition $n = \sum_i \mathrm{V_i}$.

Lemma1: Consider value V and variable A. Then $\mathrm{Var}[A] \leq \Delta b(n+1)$

Lemma2: For any $i \neq \mathrm{j}$. $\mathrm{Cov}[A_i, A_j] = 0$.

Combining, we have that $\mathrm{Var}[\hat{n}\,] = \sum_i \mathrm{Var}[A_i] \leq \sum_i \mathrm{V}_i b(n+1) \leq bn(n+1)$

It remains to prove the Lemmas…

Weighted Morris Counter: Variance bound, Lemma1

Estimate: $(1+b)^s - 1$

Add Δ : Increment s with probability $\frac{\Delta}{b(1+b)^s}$

Consider all data values $\mathrm{V_i}$ and the corresponding random variables A_i that is the increase in the estimate. Note that by definition $n = \sum_i V_\mathrm{i}$.

Lemma1: Consider value V and variable A. Then $\mathrm{Var}[A] \leq \mathrm{V}b(n+1)$

Proof: The variance, conditioned on the state of the counter s , only depends on the "probabilistic" part which is $\Delta \leq \mathrm{V}$.

$$\mathrm{Var}[A \mid \mathrm{s}] = \left(\frac{1}{\mathrm{p}} - 1\right)\Delta'^2 \leq \frac{\mathrm{b}(1+\mathrm{b})^\mathrm{s}}{\Delta}\Delta^2 = \Delta b \ \ (1+\mathrm{b})^\mathrm{s}$$

The value of s at the time the element is processed is at most the final value $s' \geq s$ of the counter. So $\mathrm{Var}[A \mid \mathrm{s}] \leq \ \Delta b \ \ (1+\mathrm{b})^{\mathrm{s}'}$

The unconditioned variance is bounded by the expectation over the distribution of s'.

Note that $E[(1+b)^{s'}] = n+1$. Therefore

$$\mathrm{Var}[A] = \mathrm{E_s}[\mathrm{Var}[A \mid \mathrm{s}]] \leq \ \Delta b \ E_{s'}[(1+b)^{s'}] = \Delta b \ (n+1) \leq Vb(n+1)$$

Weighted Morris Counter: Variance bound, Lemma2

Estimate: $(1+b)^s - 1$

Add Δ : Increment s with probability $\frac{\Delta}{b(1+b)^s}$

Consider all data values V_i and the corresponding random variables A_i that is the increase in the estimate. Note that by definition $n = \sum_i V_i$.

Lemma2: For any $i \neq j$. $\mathrm{Cov}[A_i, A_j] = 0$.

Proof:

Suppose V_1 is processed first. We have $E[A_1] = V_1$. We now consider A_2 Conditioned on A_1. Recall that the expectation of A_2 conditioned on any value of the counter when V_2 is processed is $E[A_2 \mid s] = V_2 = E[A_2]$. Therefore, for any a ,

$E[A_2 \mid A_1 = a] = V_2$.

$$E[A_1 A_2] = \sum_a a \Pr[A_1 = a]\, E[A_2 \mid A_1 = a] = V_2 E[A_1] = E[A_2]E[A_1] = V_1 V_2$$

Preview: Counting Distinct Keys

Same keys can occur in multiple data elements, we want to count the number of *distinct* keys.

- Number of distinct keys is $\boldsymbol{n}$ $(= 6$ in example$)$
- Number of data elements in this example is 11

Counting Distinct Keys: Example Applications

- Networking:
  - Packet or request streams: Count the number of distinct source IP addresses
  - Packet streams: Count the number of distinct IP flows (source+destination IP, port, protocol)
- Search Engines: Find how many distinct search queries were issued to a search engine each day

Bibliography

Misra Gries Summaries

- J. Misra and David Gries, Finding Repeated Elements. Science of Computer Programming 2, 1982
  http://www.cs.utexas.edu/users/misra/scannedPdf.dir/FindRepeatedElements.pdf
- Merging: Agarwal, Cormode, Huang, Phillips, Wei, and Yi, Mergeable Summaries, PODS 2012

Bloom filters:

- Bloom, Burton H. (1970), "Space/Time Trade-offs in Hash Coding with Allowable Errors", *Communications of the ACM*, **13** (7)

- https://en.wikipedia.org/wiki/Bloom_filter

Approximate counting (Morris Algorithm)

- Robert Morris. Counting Large Numbers of Events in Small Registers. Commun. ACM, 21(10): 840-842, 1978
  http://www.inf.ed.ac.uk/teaching/courses/exc/reading/morris.pdf
- Philippe Flajolet Approximate counting: A detailed analysis. BIT 25 1985
  http://algo.inria.fr/flajolet/Publications/Flajolet85c.pdf
- Merging Morris counters: these slides