Leveraging Big Data

http://www.cohenwang.com/edith/bigdataclass2013

Instructors:

Edith Cohen

Amos Fiat

Haim Kaplan

Tova Milo

Disclaimer: This is the first time we are offering this class (new material also to the instructors!)

- EXPECT many glitches
- Ask questions

What is Big Data ?

Huge amount of information, collected continuously: network activity, search requests, logs, location data, tweets, commerce, data footprint for each person ....

What's new ?

- **Scale**: terabytes -> petabytes -> exabytes -> ...
- **Diversity**: relational, logs, text, media, measurements
- **Movement**: streaming data, volumes moved around

Eric Schmidt (Google) 2010: **"Every 2 Days We Create As Much Information As We Did Up to 2003"**

The Big Data Challenge

To be able to handle and leverage this information, to offer better services, we need

- Architectures and tools for data storage, movement, processing, mining, ….
- Good models

Big Data Implications

- Many classic tools are not all that relevant
  - Can't just throw everything into a DBMS
- Computational models:
  - map-reduce (distributing/parallelizing computation)
  - data streams (one or few sequential passes)
- Algorithms:
  - Can't go much beyond "linear" processing
  - Often need to trade-off accuracy and computation cost
- More issues:
  - Understand the data: Behavior models with links to Sociology, Economics, Game Theory, …
  - Privacy, Ethics

This Course

Selected topics that

- We feel are important
- We think we can teach
- Aiming for breadth
  - but also for depth and developing good working understanding of concepts

http://www.cohenwang.com/edith/bigdataclass2013

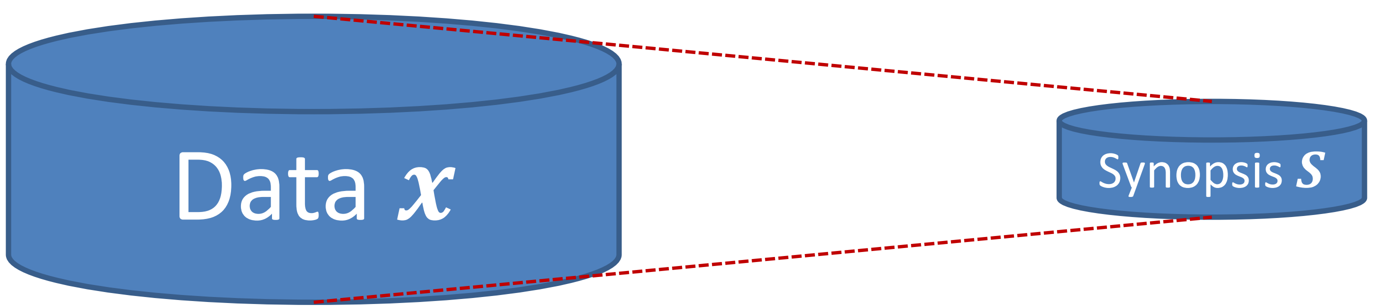
Today

- Short intro to synopsis structures
- The data streams model
- The Misra Gries frequent elements summary
  - Stream algorithm (adding an element)
  - Merging Misra Gries summaries
- Quick review of randomization
- Morris counting algorithm
  - Stream counting
  - Merging Morris counters
- Approximate distinct counting

Synopsis (Summary) Structures

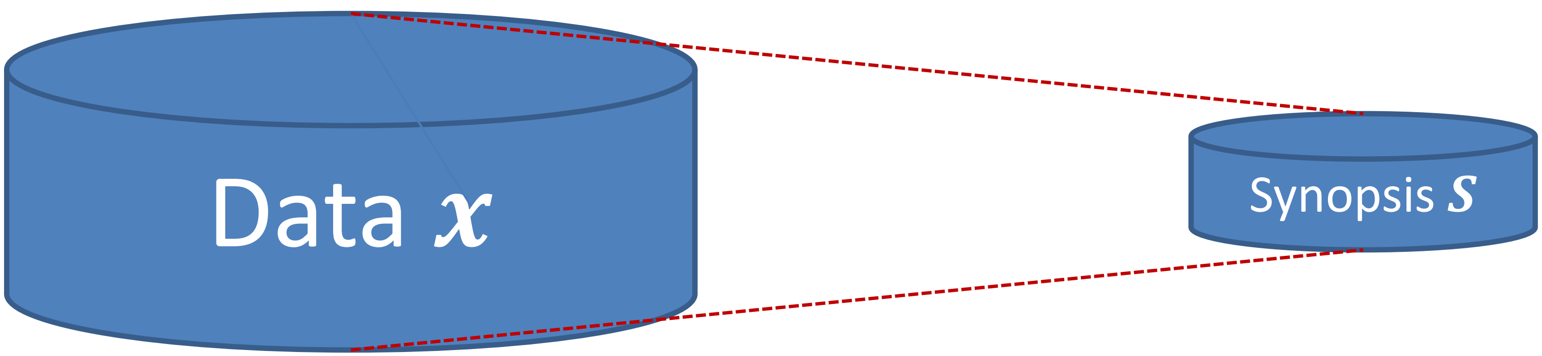
A small summary of a large data set that (approximately) captures some statistics/properties we are interested in.

Examples: random samples, sketches/projections, histograms, …



Query a synopsis: Estimators

A function $\hat{f}$ we apply to a synopsis $\boldsymbol{S}$ in order to obtain an estimate $\hat{f}(\boldsymbol{S})$ of a property/statistics/function $f(\boldsymbol{x})$ of the data $\boldsymbol{x}$



? $f(\boldsymbol{x}) \longrightarrow \hat{f}(\boldsymbol{S})$

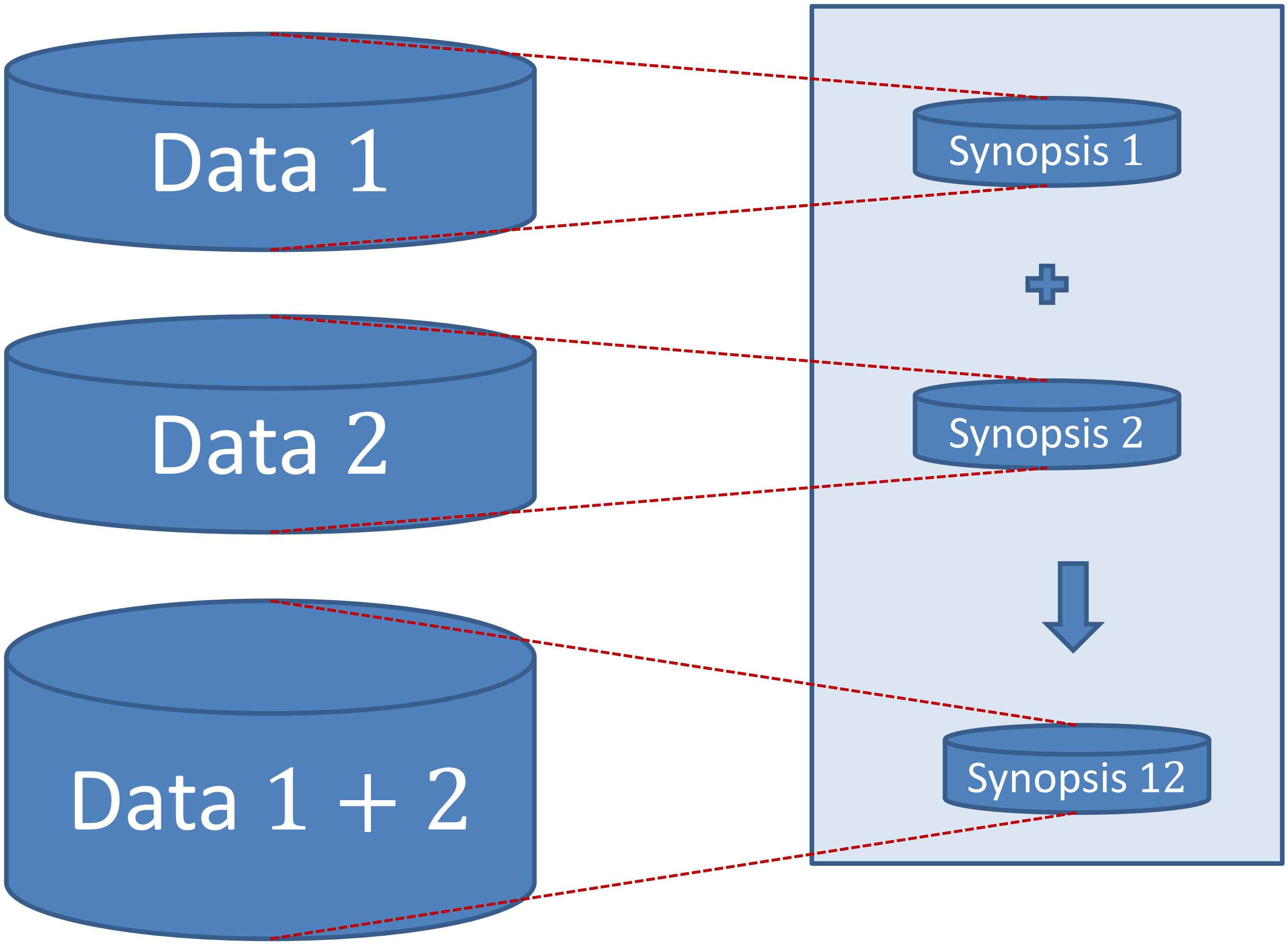
Synopsis Structures

A small summary of a large data set that (approximately) captures some statistics/properties we are interested in.

Useful features:

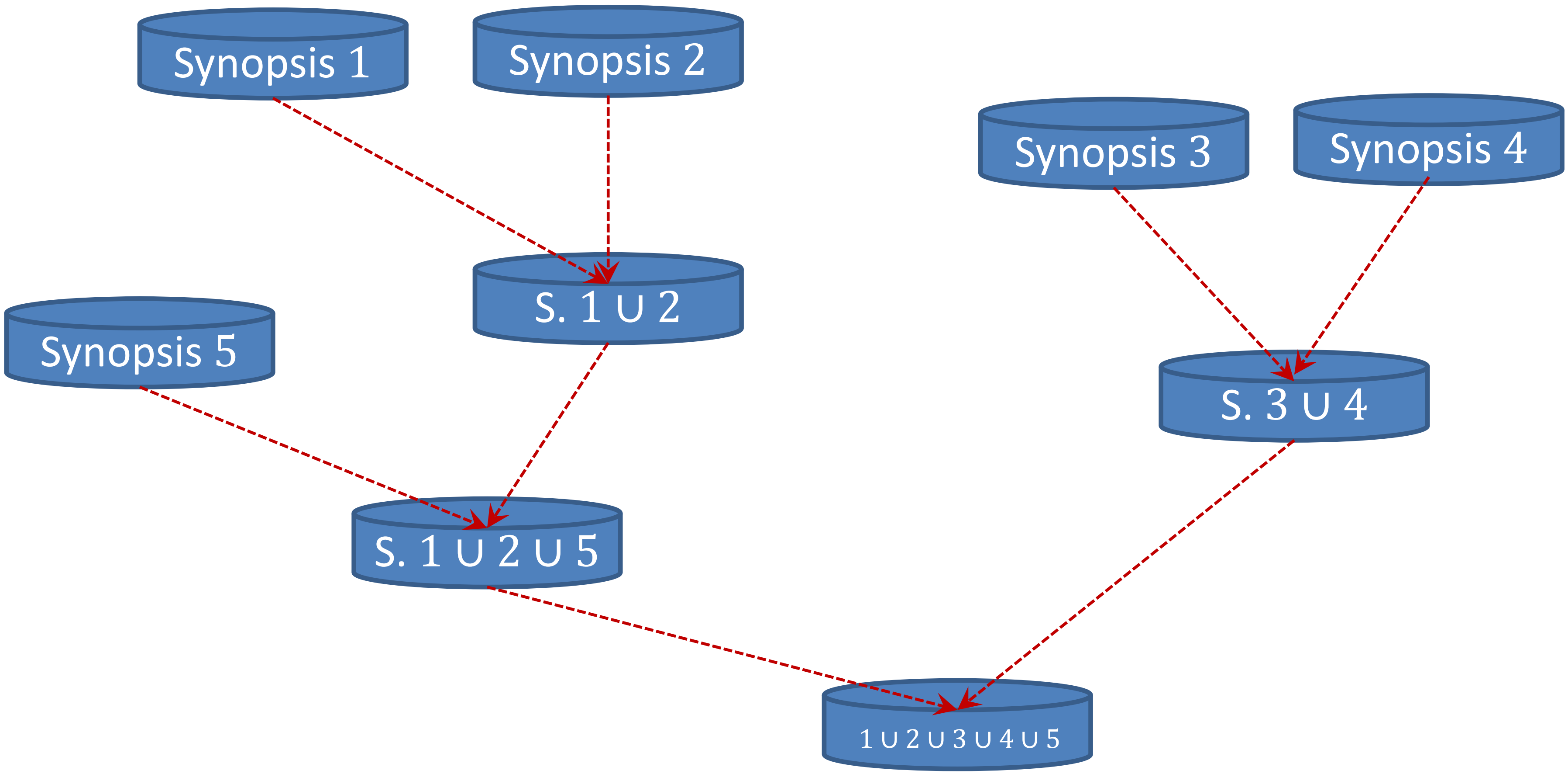
- Easy to add an element
- Mergeable : can create summary of union from summaries of data sets
- Deletions/“undo” support
- Flexible: supports multiple types of queries

Mergeability



Enough to consider merging two sketches

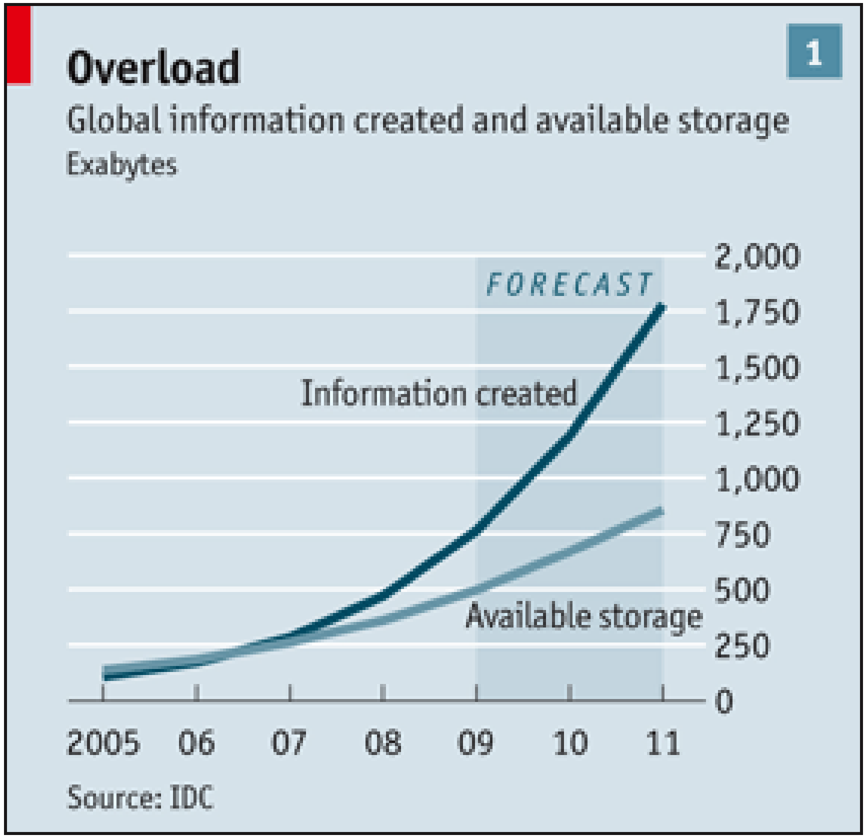
Why megeability is useful



Synopsis Structures: Why?

Data can be too large to:

- Keep for long or even short term
- Transmit across the network
- Process queries over in reasonable time/computation



Data, data, everywhere. Economist 2010

The Data Stream Model

- Data is read sequentially in one (or few) passes
- We are limited in the size of working memory.
- We want to create and maintain a synopsis which allows us to obtain good estimates of properties

Streaming Applications

- **Network management**: traffic going through high speed routers (data can not be revisited)
- **I/O efficiency** (sequential access is cheaper than random access)
- Scientific data, satellite feeds

Streaming model

Sequence of elements from some domain

<x1, x2, x3, x4, ... >

- Bounded storage:

  working memory << stream size

  usually $O(\log^k n)$ or $O(n^\alpha)$ for $\alpha < 1$
- Fast processing time per stream element

What can we compute over a stream ?

32, 112, 14, 9, 37, 83, 115, 2,

Some functions are easy: min, max, sum, …

We use a single register $\boldsymbol{s}$, simple update:

- Maximum: **Initialize** $\boldsymbol{s} \leftarrow \mathbf{0}$

  For element $\boldsymbol{x}$ **,** $\boldsymbol{s} \leftarrow \max \boldsymbol{s}, \boldsymbol{x}$
- Sum: **Initialize** $\boldsymbol{s} \leftarrow \mathbf{0}$

  For element $\boldsymbol{x}$ **,** $\boldsymbol{s} \leftarrow \boldsymbol{s} + \boldsymbol{x}$

The “synopsis” here is a single value.

It is also mergeable.

Frequent Elements

32, 12, 14, 32, 7, 12, 32, 7, 6, 12, 4,

- Elements occur multiple times, we want to find the elements that occur very often.
- Number of distinct element is $\boldsymbol{n}$
- Stream size is $\boldsymbol{m}$

Frequent Elements

32, 12, 14, 32, 7, 12, 32, 7, 6, 12, 4,

Applications:

- Networking: Find “elephant” flows
- Search: Find the most frequent queries

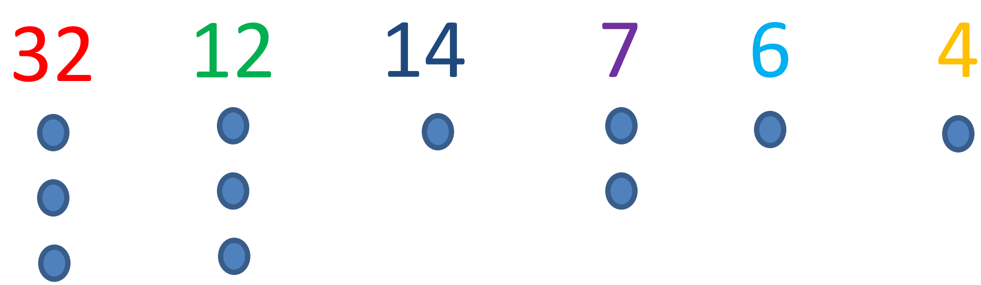
Zipf law: Typical frequency distributions are highly skewed: with few very frequent elements.
Say top 10% of elements have 90% of total occurrences.
We are interested in finding the heaviest elements

Frequent Elements: Exact Solution

32, 12, 14, 32, 7, 12, 32, 7, 6, 12, 4,

Exact solution:

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



Problem: Need to maintain n counters.

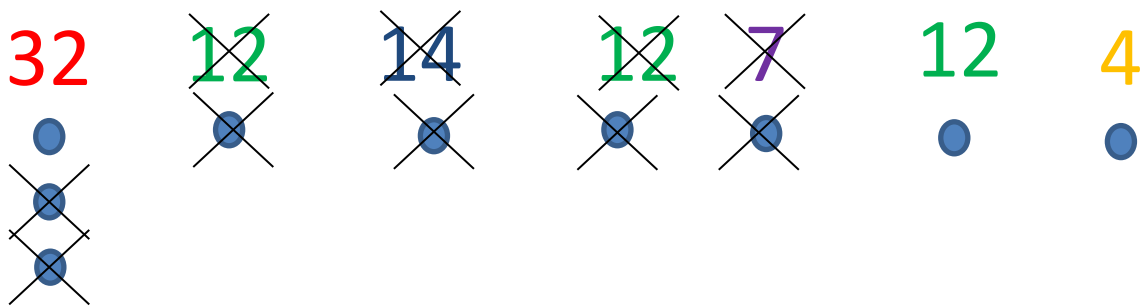
But can only maintain $k \ll n$ counters

Frequent Elements: Misra Gries 1982

32, 12, 14, 32, 7, 12, 32, 7, 6, 12, 4,

<u>Processing an element $\boldsymbol{x}$</u>

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



$$n = 6$$
$$k = 3$$
$$m = 11$$

Frequent Elements: Misra Gries 1982

32, 12, 14, 32, 7, 12, 32, 7, 6, 12, 4,

<u>Processing an element x</u>

- 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.

<u>Query: How many times x occurred ?</u>

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

This is clearly an under-estimate.
What can we say precisely?

Misra Gries 1982 : Analysis

How many decrements to a particular x can we have ?

$\Leftrightarrow$ How many decrement steps can we have ?

- Suppose total weight of structure (sum of counters) is m'
- Total weight of stream (number of occurrences) is m
- Each decrement step results in removing k counts from structure, and not counting current occurrence of the input element. That is $k+1$ "uncounted" occurrences.
- $\Rightarrow$ There can be at most $\frac{m-m'}{k+1}$ decrement steps

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

Misra Gries 1982 : Analysis

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

$\Rightarrow$ We get good estimates for x when the number of occurrences $\gg \frac{m-m'}{k+1}$

- Error bound is inversely proportional to k
- The error bound can be computed with summary: We can track m (simple count), know m' (can be computed from structure) and k.
- MG works because typical frequency distributions have few very popular elements "Zipf law"

Merging two Misra Gries Summaries

[ACHPWY 2012]

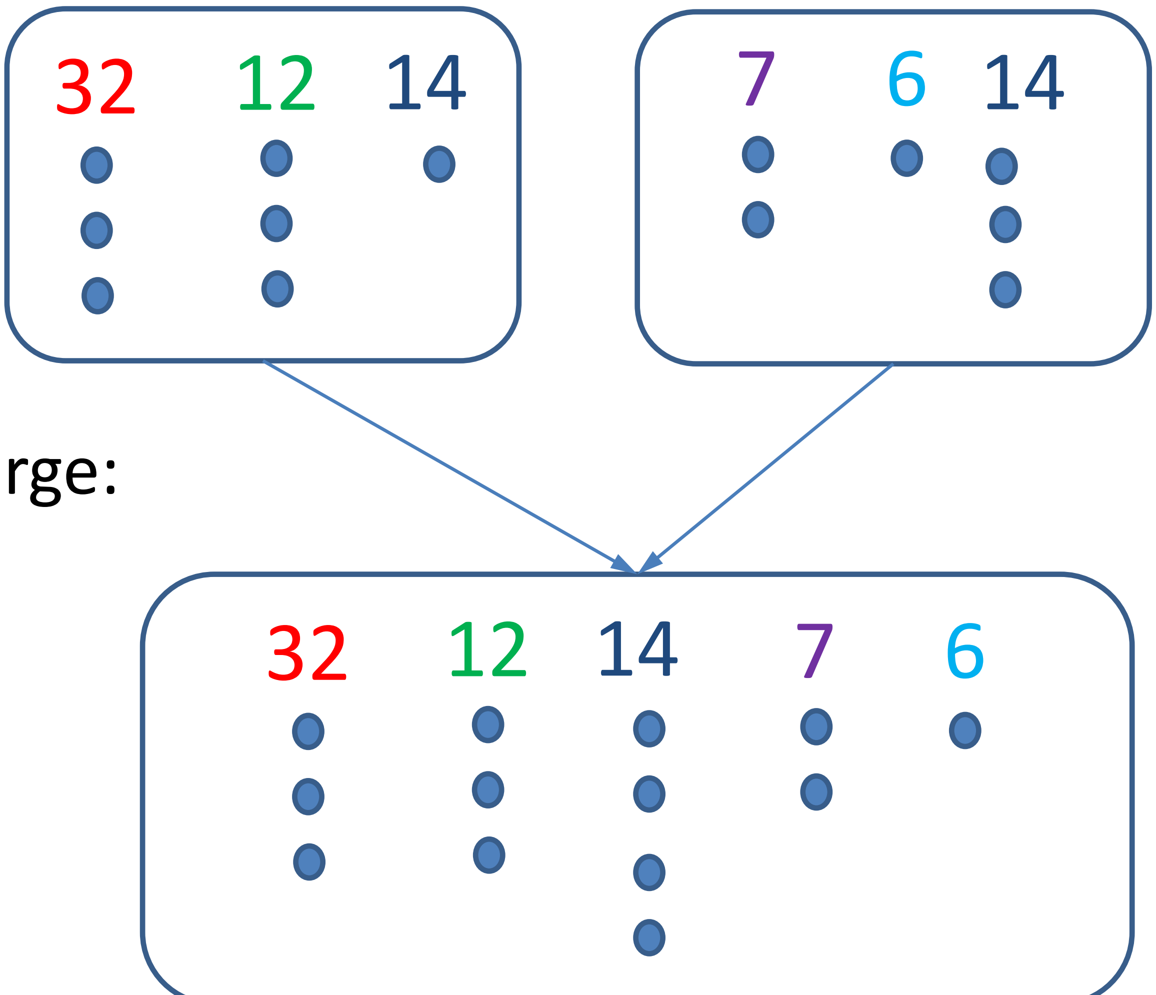
Basic merge:

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

Reduce: If there are more than k counters

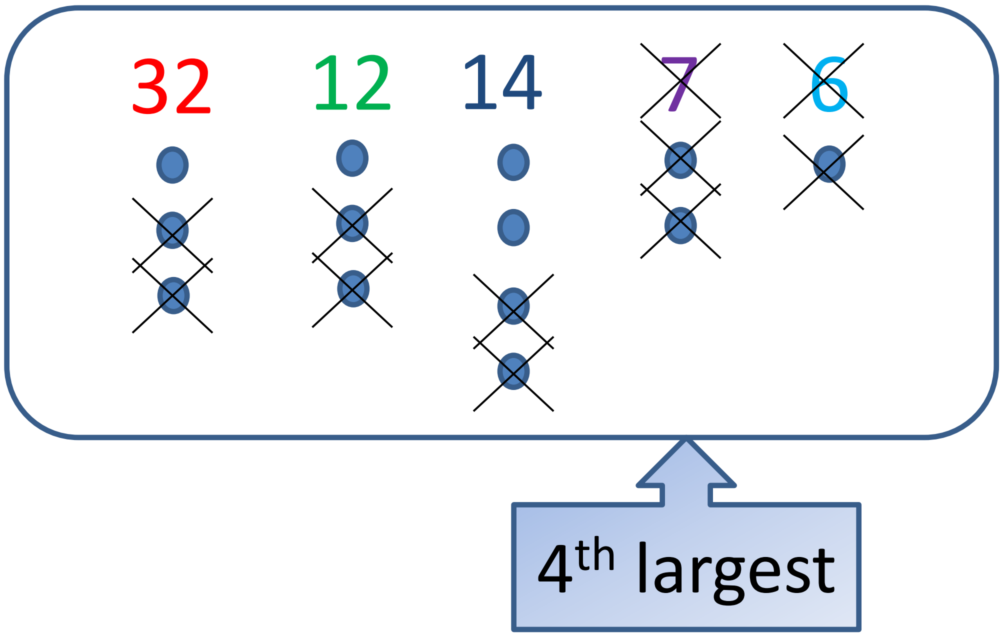
- Take the $(k+1)^{\text{th}}$ largest counter
- Subtract its value from all other counters
- Delete non-positive counters

Merging two Misra Gries Summaries



Basic Merge:

Merging two Misra Gries Summaries



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

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

Merging MG Summaries: Correctness

Claim: Final summary 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 element, final summary count is smaller than true count by at most $\frac{m-m'}{k+1}$

Merging MG Summaries: Correctness

Claim: For each element, final summary count is smaller than true count by at most $\frac{m-m\prime}{k+1}$

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

We add up the bounds on the losses

Part 1:

Total occurrences: m_1

In structure: $m_1{}'$

Count loss: $\leq \frac{\boldsymbol{m_1 - m_1\prime}}{\boldsymbol{k+1}}$

Part 2:

Total occurrences: m_2

In structure: $m_2{}'$

Count loss: $\leq \frac{\boldsymbol{m_2 - m_2\prime}}{\boldsymbol{k+1}}$

Reduce loss is at most $\boldsymbol{X = (k+1)}$th largest counter

Merging MG Summaries: Correctness

$\Rightarrow$ "Count loss" of one element is at most

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

Part 1:

Total occurrences: m_1

In structure: m_1'

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

Part 2:

Total occurrences: m_2

In structure: m_2'

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

Reduce loss is at most $X = (k+1)^{\text{th}}$ largest counter

Merging MG Summaries: Correctness

Counted occurrences in structure:

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

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

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

"Count loss" of one element is at most

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

$\Rightarrow$ at most $\dfrac{m - m'}{k+1}$ uncounted occurrences

Using Randomization

- Misra Gries is a *deterministic* structure
- The outcome is determined uniquely by the input
- Usually we can do much better with *randomization*

Randomization in Data Analysis

Often a critical tool in getting good results

- Random sampling / random projections as a means to reduce size/dimension
- Sometimes data is treated as samples from some distribution, and we want to use the data to approximate that distribution (for prediction)
- Sometimes introduced into the data to mask insignificant points (for robustness)

Randomization: Quick review

- Random variable (discrete or continuous) X
- Probability Density Function (PDF)

  $f_X(x)$: Probability/density of $X = x$

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

- Cumulative Distribution Function (CDF)

  $F_X(x) = \int_{-\infty}^{x} f_X(t)\,dt$: probability that $X \leq x$

  ➢ **Properties:** $F_X(x)$ **monotone non-decreasing from 0 to 1**

Quick review: Expectation

- ***Expectation***: "average" value of X :

$$\mu = E[X] = \int_{-\infty}^{\infty} x\, f_X(x)\, dx$$

- ***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*

$$\mathbf{V}[X] = \sigma^2 = E\left[(X-\mu)^2\right]$$
$$= \int_{-\infty}^{\infty} (x-\mu)^2 f_X(x)\,dx$$

- Useful relations: $\sigma^2 = E\left[x^2\right] - \mu^2$

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

- The *standard deviation* is $\sigma = \sqrt{V[X]}$
- *Coefficient of Variation* $\frac{\sigma}{\mu}$

Quick review: CoVariance

- CoVariance (measure of dependence between two random variables) X, Y

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

- X, Y are independent $\Longrightarrow$ $\sigma(X, Y) = 0$

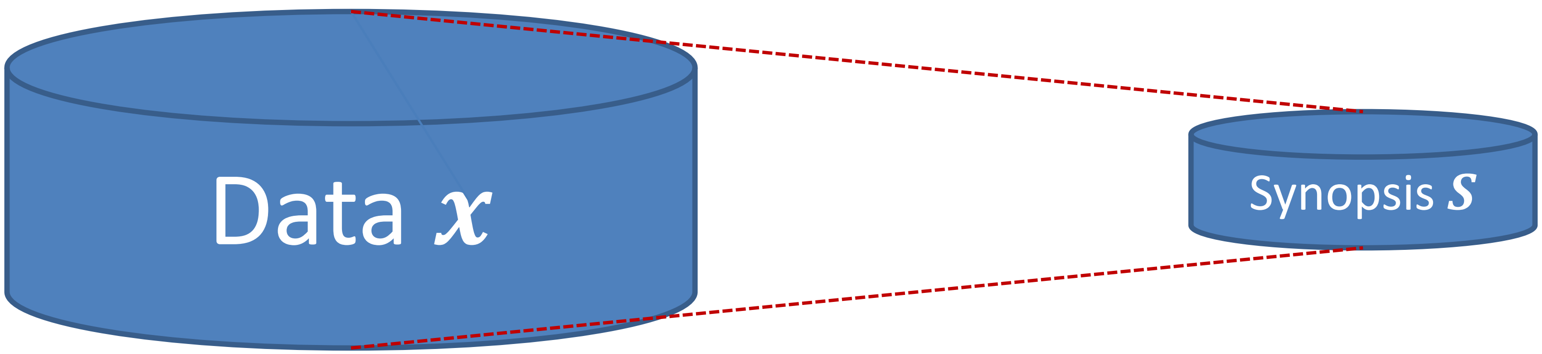
- Variance of the sum of $X_1, X_2, \ldots, X_k$

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

When (pairwise) independent

Back to Estimators

A function $\hat{f}$ we apply to "observed data" (or to a "synopsis") $\boldsymbol{S}$ in order to obtain an estimate $\hat{f}(\boldsymbol{S})$ of a property/statistics/function $f(\boldsymbol{x})$ of the data $\boldsymbol{x}$



? $f(\boldsymbol{x}) \longrightarrow \hat{f}(\boldsymbol{S})$

Quick Review: Estimators

A function $\hat{f}$ we apply to "observed data" (or to a "synopsis") $\boldsymbol{S}$ in order to obtain an estimate $\hat{f}(\boldsymbol{S})$ of a property/statistics/function $f(\boldsymbol{x})$ of the data $\boldsymbol{x}$

- ***Error*** $\mathrm{err}(\hat{f}) = \hat{f}(\boldsymbol{S}) - f(\boldsymbol{x})$
- ***Bias*** $\mathrm{Bias}[\hat{f} \mid \boldsymbol{x}] = \mathrm{E}[\mathrm{err}(\hat{f})] = E[\hat{f}] - f(\boldsymbol{x})$
  - When $\mathrm{Bias} = 0$ estimator is ***unbiased***
- ***Mean Square Error (MSE):***

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

- ***Root Mean Square Error (RMSE):*** $\sqrt{MSE}$

Back to stream counting

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

- Count: **Initialize $s \leftarrow 0$**

  For each element, $s \leftarrow s + 1$

Register (our synopsis) size (bits) is $\lceil \log_2 n \rceil$ where n is the current count

Can we use fewer bits ? Important when we have many streams to count, and fast memory is scarce (say, inside a backbone router)

What if we are happy with an ***approximate*** count ?

Morris Algorithm 1978

The first streaming algorithm

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

Stream counting:

Stream of +1 increments

Maintain an approximate count

Idea: track $\log n$ instead of n

Use $\log \log n$ bits instead of $\log n$ bits

Morris Algorithm

Maintain a "log" counter $\boldsymbol{x}$

- **Increment**: Increment with probability $\boldsymbol{2^{-x}}$
- **Query**: Output $\boldsymbol{2^{x} - 1}$

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 Algorithm: Unbiasedness

- When $\boldsymbol{n = 1}$, $\boldsymbol{x = 1}$,

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

  with $p = \frac{1}{2}$, $\boldsymbol{x = 1}$, $\hat{n} = \mathbf{1}$

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

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

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

Morris Algorithm: …Unbiasedness

- $\boldsymbol{X_n}$ is the random variable corresponding to the counter x when the count is $\boldsymbol{n}$
- We need to show that

$$\mathrm{E}[\hat{n}] = \mathbf{E}[\mathbf{2}^{X_n} - \mathbf{1}] = \boldsymbol{n}$$

- That is, to show that $\boxed{\mathbf{E}[\mathbf{2}^{X_n}] = \boldsymbol{n} + \mathbf{1}}$

$$\mathbf{E}[\mathbf{2}^{X_n}] = \sum_{j \geq 1} \mathbf{Prob}[X_{n-1} = j]\,\mathbf{E}[\mathbf{2}^{X_n} | X_{n-1} = j]$$

- We next compute: $\boxed{\mathbf{E}[\mathbf{2}^{X_n} | X_{n-1} = j]}$

Morris Algorithm: …Unbiasedness

Computing $\boxed{\mathbf{E}[2^{X_n} | X_{n-1} = j]:}$

- with probability $p = 1 - 2^{-j}$: $x = j$, $2^x = 2^j$
- with probability $p = 2^{-j}$: $x = j + 1$, $2^x = 2^{j+1}$

$$\mathbf{E}[2^{X_n} | X_{n-1} = j] = (1 - 2^{-j})2^j + 2^{-j}2^{j+1}$$
$$= 2^j - 1 + 2 = 2^j + 1$$

Morris Algorithm: …Unbiasedness

$$\boxed{\mathbf{E}[2^{X_n} | X_{n-1} = j] = 2^j + 1}$$

$$\mathbf{E}[2^{X_n}] = \sum_{j \geq 1} \mathbf{Prob}[X_{n-1} = j] \mathbf{E}[2^{X_n} | X_{n-1} = j]$$

$$= \sum_{j \geq 1} \mathbf{Prob}[X_{n-1} = j](2^j + 1)$$

$$= \underbrace{\sum_{j \geq 1} \mathbf{Prob}[X_{n-1} = j](2^j - 1)}_{= \mathbf{E}[2^{X_{n-1}} - 1] = n - 1 \text{ by induction hyp.}} + \underbrace{\sum_{j \geq 1} \mathbf{Prob}[X_{n-1} = j]\, 2}_{1}$$

$$= n + 1$$

Morris Algorithm: Variance

How good is the estimate?

- The r.v.'s $\hat{n} = 2^{X_n} - 1$and $\hat{n} + 1 = n = 2^{X_n}$ have the same variance V$[\hat{n}] = V[\hat{n} + 1]$
- $V[\hat{n} + 1] = E[2^{2X_n}] - (n + 1)^2$
- We can show $E[2^{2X_n}] = \frac{3}{2}n^2 + \frac{3}{2}n + 1$
- This means $V[\hat{n}] \approx \frac{1}{2}n^2$ and CV $= \frac{\sigma}{\mu} \approx \frac{1}{\sqrt{2}}$

How to reduce the error ?

Morris Algorithm: Reducing variance 1

$V[\hat{n}] = \sigma^2 \approx \frac{1}{2}n^2$ and CV $= \frac{\sigma}{\mu} \approx \frac{1}{\sqrt{2}}$

Dedicated Method: Base change –

IDEA: Instead of counting $\mathbf{log_2}\, \boldsymbol{n}$, count $\mathbf{log_b}\, \boldsymbol{n}$

- Increment counter with probability $\boldsymbol{b^{-x}}$

When $\boldsymbol{b}$ is closer to 1, we increase accuracy but also increase counter size.

Morris Algorithm: Reducing variance 2

$V[\hat{n}] = \sigma^2 \approx \frac{1}{2}n^2$ and CV $= \frac{\sigma}{\mu} \approx \frac{1}{\sqrt{2}}$

Generic Method:

- Use $\boldsymbol{k}$ *independent* counters $\boldsymbol{y_1, y_2, \ldots, y_k}$
- Compute estimates

$$\boldsymbol{Z_i = 2^{y_i} - 1}$$

- Average the estimates

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

Reducing variance by averaging

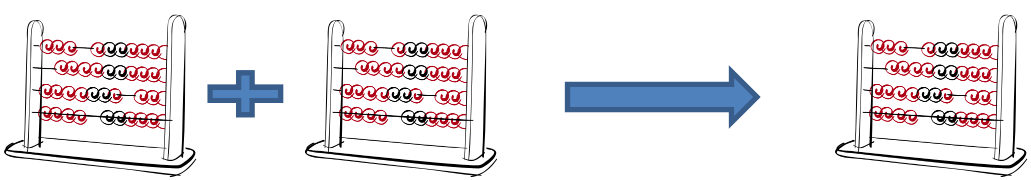
$\boldsymbol{k}$ (pairwise) *independent* estimates $\boldsymbol{Z_i}$ with expectation $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma^2}$.

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

- Expectation: $E\left[\widehat{n'}\right] = \frac{1}{k}\sum_{i=1}^{k} E[Z_i] = \frac{1}{k} k\mu = \mu$
- Variance: $\left(\frac{1}{k}\right)^2 \sum_{i=1}^{k} V[Z_i] = \left(\frac{1}{k}\right)^2 k\sigma^2 = \frac{\sigma^2}{k}$
- CV : $\frac{\sigma}{\mu}$ decreases by a factor of $\sqrt{k}$

Merging Morris Counters

- We have two Morris counters $\boldsymbol{x}, \boldsymbol{y}$ for streams X, Y of sizes n_x , n_y
- Would like to *merge* them: obtain a single counter $\boldsymbol{z}$ which has the same distribution (is a Morris counter) for a stream of size $n_x + n_y$



Merging Morris Counters

- Morris-count stream X to get $\boldsymbol{x}$
- Morris-count stream Y to get $\boldsymbol{y}$

Merge the Morris counts $\boldsymbol{x}, \boldsymbol{y}$ (into $\boldsymbol{x}$):

- For $i = 1 \ldots \boldsymbol{y}$
- Increment $\boldsymbol{x}$ with probability $\mathbf{2}^{-x+\boldsymbol{i}-\mathbf{1}}$

Correctness for $\boldsymbol{x} = 0$: at all steps we have we $\boldsymbol{x} = i - 1$ and probability$= 1$. In the end we have $\boldsymbol{x} = \boldsymbol{y}$

Correctness (Idea): We will show that the final value of $\boldsymbol{x}$ "corresponds" to counting Y after X

Merging Morris Counters: Correctness

We want to achieve the same effect as if the Morris counting was applied to a concatenation of the streams $X\ Y$

- We consider two scenarios :

1. Morris counting applied to Y
2. Morris counting applied to Y after X

We want to simulate the result of (2) given $\boldsymbol{y}$ (result of (1)) and $\boldsymbol{x}$

Merging Morris Counters: Correctness

Restated Morris (for sake of analysis only)

Associate an (independent) random $\mathrm{u}(z) \sim U[0,1]$ with each element z of the stream

- **Process element** z : Increment $\boldsymbol{x}$ if $\mathrm{u}(z) < \mathbf{2}^{-\boldsymbol{x}}$

- We "map" executions of (1) and (2) by looking at the same randomization u.
- We will see that each execution of (1), in terms of the set of elements that increment the counter, maps to many executions of (2)

Merging algorithm: Correctness Plan

- We fix the *whole run (and randomization)* on X.
- We fix *the set of elements that result in counter increments* on Y in (1)
- We work with the distribution of u: Y *conditioned* on the above.
- We show that the corresponding distribution over executions of (2) (set of elements that increment the counter) emulates our merging algorithm.

What is the conditional distribution?

- Elements that did not increment counter when counter value was x have $u(z) \geq 2^{-x}$
- Elements that did increment counter have $u(z) \leq 2^{-x}$

$\boldsymbol{u}$:		$[0,1]$	$[\frac{1}{2},1]$	$[0,\frac{1}{2}]$	$[\frac{1}{4},1]$	$[\frac{1}{4},1]$	$[\frac{1}{4},1]$	$[0,\frac{1}{4}]$	$[\frac{1}{8},1]$
Stream:		1,	1,	1,	1,	1,	1,	1,	1,
$p = 2^{-x}$:	1	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{8}$

Merge the Morris counts $\boldsymbol{x}, \boldsymbol{y}$ (into $\boldsymbol{x}$):

- For $i = 1 \ldots \boldsymbol{y}$
- Increment $\boldsymbol{x}$ with probability $\mathbf{2}^{-x+\boldsymbol{i}-\mathbf{1}}$

To show correctness of merge, suffices to show:

- Elements of Y that did not increment in (1) do not increment in (any corresponding run of) (2)
- Element z that had the i^{th} increment in (1), conditioned on x in the simulation so far, increments in (2) with probability $\mathbf{2}^{-x+\boldsymbol{i}-\mathbf{1}}$

We show this inductively.

Also show that at any point $x \geq y'$, where y' is the count in (1).

Merge the Morris counts $\boldsymbol{x}$, $\boldsymbol{y}$ (into $\boldsymbol{x}$):

- For $i = 1 \ldots \boldsymbol{y}$
- Increment $\boldsymbol{x}$ with probability $\mathbf{2}^{-x+\boldsymbol{i}-\mathbf{1}}$

The first element of Y incremented the counter in (1). It has $u(z) \in [0,1]$.

- The probability that it gets counted in (2) is

$$\Pr[\mathrm{u}(\mathrm{z}) \leq 2^{-x} \mid u(z) \in [0,1]] = \mathbf{2}^{-x}$$

- Initially, $\boldsymbol{x} \geq \mathrm{y}' = 0$. After processing, $\boldsymbol{y}' = \mathbf{1}$. If $\boldsymbol{x}$ was initially 0, it is incremented with probability 1, so we maintain $\boldsymbol{x} \geq \mathrm{y}'$.

Merge the Morris counts $\boldsymbol{x}, \boldsymbol{y}$ (into $\boldsymbol{x}$):

- For $i = 1 \ldots \boldsymbol{y}$
- Increment $\boldsymbol{x}$ with probability $\mathbf{2}^{-x+\boldsymbol{i}-\mathbf{1}}$

- Elements of Y that did not increment in (1) do not increment in (any corresponding run of) (2)

Proof: An element z of Y that did not increment the counter when its value in (1) was y', has $u(z) \in [2^{-y'}, 1]$.

Since we have $x \geq y'$, this element will also not increment in (2), since $\mathrm{u}(z) \geq 2^{-y'} \geq 2^{-x}$.

The counter in neither (1) nor (2) changes after processing z, so we maintain the relation $x \geq y'$.

Merge the Morris counts $\boldsymbol{x}, \boldsymbol{y}$ (into $\boldsymbol{x}$):

- For $i = 1 \ldots \boldsymbol{y}$
- Increment $\boldsymbol{x}$ with probability $\mathbf{2}^{-x+\boldsymbol{i}-\mathbf{1}}$

- Element z that had the i^{th} increment in (1), conditioned on x in the simulation so far, increments in (2) with probability $\mathbf{2}^{-x+\boldsymbol{i}-\mathbf{1}}$

Proof: Element z has $\mathrm{u}(z) \in \left[0, 2^{-(i-1)}\right]$ (we had $\mathrm{y}' = i - 1$ before the increment).

Element z increments in (2) $\Leftrightarrow \mathrm{u}(z) \in [0, 2^{-x}]$.

$$\Pr\left[\mathrm{u}(z) \in [0, 2^{-x}] \,|\mathrm{u}(z) \in \left[0, 2^{-(i-1)}\right]\right] = \mathbf{2}^{-x+\boldsymbol{i}-\mathbf{1}}$$

- If we had equality $x = y' = i - 1$, x is incremented with probability 1, so we maintain the relation $x \geq y'$

Random Hash Functions

Simplified and Idealized

For a domain $\boldsymbol{D}$ and a probability distribution $\boldsymbol{F}$ over $\boldsymbol{R}$

A distribution over a family $\boldsymbol{H}$ of hash functions $h\colon \boldsymbol{D} \rightarrow \boldsymbol{R}$ with the following properties:

- Each function $h \in H$ has a concise representation and it is easy to choose $h \sim H$
- For each $x \in \boldsymbol{D},$ when choosing $h \sim H$
  - $h(x) \sim \boldsymbol{F}$ ($h(x)$ is a random variable with distribution $\boldsymbol{F}$)
  - The random variables $h(x)$ are independent for different $x \in D$.

We use random hash functions as a way to attach a "permanent" random value to each identifier in an execution

Counting Distinct Elements

32, 12, 14, 32, 7, 12, 32, 7, 6, 12, 4,

Elements occur multiple times, we want to count the number of *distinct* elements.

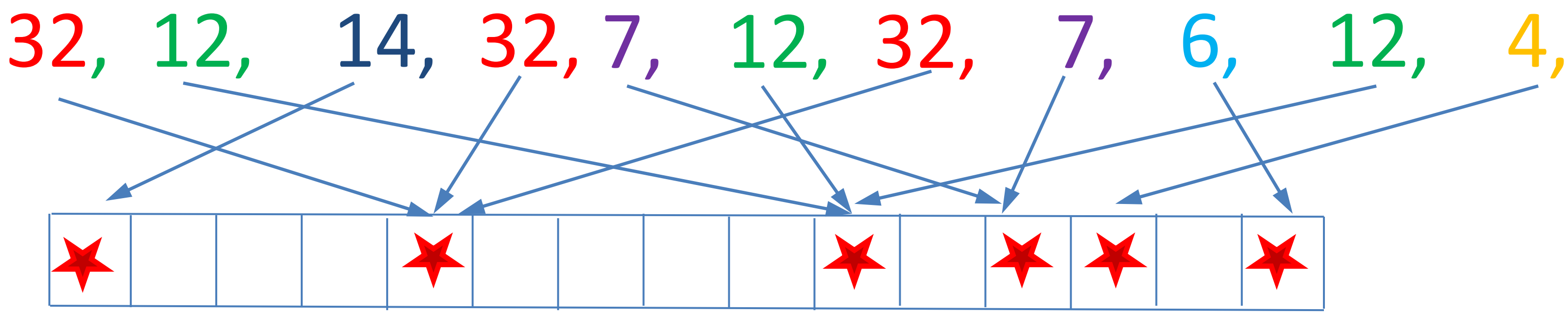
- Number of distinct element is $\boldsymbol{n}$ $(= 6$ in example)
- Number of elements in this example is 11

Counting Distinct Elements: Example Applications

32, 12, 14, 32, 7, 12, 32, 7, 6, 12, 4,

- 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: Find how many distinct search queries were issued to a search engine each day

Distinct Elements: Exact Solution



Exact solution:

- Maintain an array/associative array/ hash table
- Hash/place each element to the table
- Query: count number of entries in the table

Problem: For n distinct elements, size of table is $\Omega(n)$

But this is the best we can do (Information theoretically) if we want an **exact** distinct count.

Distinct Elements: Approximate Counting

32, 12, 14, 32, 7, 12, 32, 7, 6, 12, 4,

IDEA: Size-estimation/Min-Hash technique :

[Flajolet-Martin 85, C 94]

- Use a random hash function $h(x) \sim U[0,1]$mapping element IDs to uniform random numbers in [0,1]
- Track the minimum $h(x)$

Intuition: The minimum and n are very related :

- With n distinct elements, expectation of the minimum $\mathrm{E}[\min h(\mathrm{x})] = \frac{1}{n+1}$
- Can use the average estimator with k repetitions

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

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

Approximate distinct counting

- P. Flajolet and G. N. Martin. Probabilistic counting. In Proceedings of Annual IEEE Symposium on Foundations of Computer Science (FOCS), pages 76–82, 1983
- E. Cohen Size-estimation framework with applications to transitive closure and reachability, JCSS 1997