Probabilistic Counting in a Stream

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

- Simple counting
  - Accumulate count in $\log_2(n)$ bits where $n$ is the current count

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

- Can we reduce storage size if an approximate count suffices?
Probabilistic Counting in a Stream

a, b, b, c, b, b, b, c,

• Counting multiple keys: \( n_a, n_b, n_c \) etc.

• Can we tune counting to focus resources on “important” keys
  – Frequent keys

• Example:
  – Packet stream; focus on large flows (high counts \( n \))
Outline

• Morris counting algorithm
• Frequent element counting
• Concise samples
• Counting samples
• Sample and hold
Morris Algorithm 1978

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

• The first streaming algorithm
  – Stream of positive increments

• Idea
  – Track log n instead of n
  – Use log log n bits instead of log n bits
Deterministic Approach?

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

• Can we simply maintain a count of $\log_2 n$?
  – using $\log_2 \log_2 n$ bits

• Problem
  – We are actually maintaining integer part $x = \text{floor}(\log_2 n)$
  – Fractional part of $\log_2 n$ is lost

<table>
<thead>
<tr>
<th>n</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>floor($\log_2 n$)</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

• When to increment $x$?
Morris Algorithm

- Maintain a “log” counter $x$
- Initialize to 0
- Each arrival:
  - increment with probability $p_x = 2^{-x}$
- Query: output estimate $n' = 2^x - 1$
Morris Algorithm: Birth Process

• Let \( X(n) \) denote count after arrival \( n \)
• Pure birth process
  – Transition \( x \rightarrow x+1 \) with probability \( 2^{-x} \)
Morris Algorithm: Unbiasedness

• Initialize $x = 0$; increment w.p. $p_x = 2^{-x}$; estimate $n' = 2^x - 1$

• $n = 1$
  – before: $x = 0$ $p_0 = 1$
  – prob. 1: $x \rightarrow 1$
  – estimate $n' = 2^1 - 1 = 1 = n$

• $n = 2$;
  – before: $x = 1$; $p_1 = \frac{1}{2}$
  – prob. $\frac{1}{2}$: $x$ stays at 1; $n' = 2^1 - 1 = 1$
  – prob. $\frac{1}{2}$: $x \rightarrow 2$. $n' = 2^2 - 1 = 3$
  – $E[n'] = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 3 = 2 = n$
Morris algorithm: general case

• Let $X(n)$ denote random counter $x$ after $n$th arrival
• Initialize $X(0) = 0$; increment w.p. $p_x = 2^{-x}$
• Estimate $n' = 2^{X(n)} - 1$

\[
E[2^{X(n)}] = \sum_{j=1}^{n-1} \Pr[X(n)-1 = j] E[2^{X(n)} | X(n)-1 = j] \\
= \sum_{j=1}^{n-1} \Pr[X(n)-1 = j] \left( p_j 2^{j+1} + (1-p_j) 2^j \right) \\
= \sum_{j=1}^{n-1} \Pr[X(n)-1 = j] \left( 2^j + 1 \right) \\
= E[2^{X(n-1)}] + 1
\]

• Iterating: $E[2^{X(n)}] = E[2^{X(0)}] + n = 1 + n$
• Therefore: $E[2^{X(n)} - 1] = n$

• Conclusion: $n' = 2^{X(n)} - 1$ is an unbiased estimator of $n$
Morris algorithm: variance

- \( \text{Var}[n'] = \text{Var}[2^X(n) - 1] = \text{Var}[2^X(n)] \)
  \[ = E[2^{2X(n)}] - E[2^X(n)]^2 \]
  \[ = E[2^{2X(n)}] - (n+1)^2 \]

- \( E[2^{2X(n)}] = \sum_{j=1}^{n-1} \text{Pr}[X(n-1) = j] E[2^{2X(n)} | X(n-1) = j] \)
  \[ = \sum_{j=1}^{n-1} \text{Pr}[X(n-1) = j] (p_j 2^{2j+2} + (1-p_j) 2^{2j}) \]
  \[ = \sum_{j=1}^{n-1} \text{Pr}[X(n-1) = j] (2^{j+2} + 2^{2j} - 2^j) \]
  \[ = \sum_{j=1}^{n-1} \text{Pr}[X(n-1) = j] (3*2^j + 2^{2j}) \]
  \[ = 3 E[2^{X(n-1)}] + E[2^{2X(n-1)}] \]
  \[ = 3n + E[2^{2X(n-1)}] \]

- Iterate: \( E[2^{2X(n)}] = 3 \sum_{m=1}^n m + E[2^{2X(0)}] \)
  \[ = 3n(n+1)/2 + 1 \]

- \( \text{Var}[n'] = 3n(n+1)/2 + 1 - (n+1)^2 = n(n-1)/2 \)
Morris algorithm

• Coefficient of Variation = StdDev / Mean \approx 1/\sqrt{2}:
  – doesn’t improve as \( n \) grows

• How to improve?
Morris Algorithm: Reducing Variance 1

- Change base of logarithms $2 \rightarrow b > 1$
- Instead of counting $\log_2(n)$, count $\log_b(n)$
- Increment counter $x$ with probability $b^{-x}$
  - Method of base $2$ analysis carries through
- $E[b^{X(n)}] = (b-1)n + 1$
  - $n' = (E[b^{X(n)}] - 1)/(b-1)$ is an unbiased estimator of $n$
- $\text{Var}[n'] = (b-1)n(n-1)/2$
- By decreasing $b$ closer to $1$
  - Decrease variance
  - Increase size of storage needed
    - $b \rightarrow \log_b(n)$ increases
Morris Algorithm: Reducing Variance 2

• Familiar approach  
  – Multiple independent estimates  
• Mean of estimates  
• Median of means
Frequent Element Counting

\[ a, b, b, c, b, b, b, c, \]

- Elements occur multiple times
- Want to find which elements occur most often
- Stream size \( n \)
- \( m \) distinct elements
Frequent Elements

\[ a, \ b, \ b, \ c, \ b, \ b, \ b, \ c, \]

• Applications
  – Networking: find “elephant” flows
  – Search: find the most frequent queries

• Pareto Principle
  – Typical frequency distributions are highly skewed
  – Small proportion of elements are very frequent

• Zipf’s Law
  – Rank elements by frequency
  – Frequency of rank \( k \) element proportional to \( 1/k^s \), some \( s > 1 \)
Frequent Elements: exact solution

a, b, b, c, b, b, b, c,

• Maintain counter for each distinct element
  – Instantiate on first occurrence
  – Increment on every occurrence

• Problem
  – Need to maintain m counters
  – Generally only have room for k << m counters
Frequent Elements: Misra & Gries 1982

a, b, b, c, b, b, b, c,

• Processing an element $x$
  – If: already have counter for $x$, increment it
  – Else if: no counter for $x$, but fewer than $k$ counters, create a counter for $x$ and initialize it to 1
  – Else: decrease all counters by 1. Remove counters containing 0.

• Query: how many times did $x$ occur?
  – If: we have a counter for $x$, return counter value
  – Else: return 0

• Clearly an underestimate
Misra & Gries: Analysis

For each $x$: true value – counter = # decrements

How many possible decrements to counter for $x$?

Suppose sum of counters is $n' < n = \text{length of stream}$

Each decrement step removes $k$ counts
  – Also did not count the current arrival

Therefore $k+1$ undercounts from each decrement
  – There are at most $d = (n-n')/(k+1)$ decrement steps
Misra & Gries: Analysis

- There are at most $d = (n-n')/(k+1)$ decrement steps
- Counter for $x$ is smaller than count by at most $d$
  - Good estimates when $\text{counter}(x) >> d$
  - Error bound inversely proportional to $k$
  - Track $n$ by count (or estimate)
- Works since typical distributions have few frequent elements
Bibliography

• Approximate counting (Morris Algorithm)
  • Philippe Flajolet Approximate counting: A detailed analysis. BIT 25 1985
    – http://algo.inria.fr/flajolet/Publications/Flajolet85c.pdf

• Frequent element summaries