CS6931 Database Seminar

Lecture 5: Streaming Model (continued)
The Distinct Counting Problem
FM Sketch

- Estimates number of distinct inputs (count distinct)
- Uses hash function mapping input items to \( i \) with prob \( 2^{-i} \)
  - i.e. \( \Pr[h(x) = 1] = \frac{1}{2}, \Pr[h(x) = 2] = \frac{1}{4}, \Pr[h(x) = 3] = \frac{1}{8} \ldots \)
  - Easy to construct \( h() \) from a uniform hash function by counting trailing zeros
- Maintain FM Sketch = bitmap array of \( L = \log U \) bits
  - Initialize bitmap to all 0s
  - For each incoming value \( x \), set \( FM[h(x)] = 1 \)
FM Sketch

- If d distinct values, expect d/2 map to FM[1], d/4 to FM[2]...

- With $O(1/\varepsilon^2 \log 1/\delta)$ copies, get $(\varepsilon, \delta)$ approximation
  - 10 copies gets $\approx 30\%$ error, 100 copies $< 10\%$ error

- Let $R = \text{position of rightmost zero in FM, indicator of log}(d)$
- Basic estimate $d = c2R$ for scaling constant $c \approx 1.3$
- Average many copies (different hash fns) improves accuracy

[Diagram showing FM Sketch with positions and bits labeled]
COUNT Sketches

- Problem: Estimate the number of distinct item IDs in a data set with only one pass.
- Constraints:
  - Small space relative to stream size.
  - Small per item processing overhead.
  - Union operator on sketch results.
- Exact COUNT is impossible without linear space.
- First approximate COUNT sketch in [FM’85].
  - $O(\log N)$ space, $O(1)$ processing time per item.
Counting Paintballs

• Imagine the following scenario:
  – A bag of $n$ paintballs is emptied at the top of a long stair-case.
  – At each step, each paintball either bursts and marks the step, or bounces to the next step. 50/50 chance either way.

Looking only at the pattern of marked steps, what was $n$?
Counting Paintballs (cont)

- What does the distribution of paintball bursts look like?
  - The number of bursts at each step follows a binomial distribution.
  - The expected number of bursts drops geometrically.
  - Few bursts after $\log_2 n$ steps
Counting Paintballs (cont)

• Many different estimator ideas [FM'85, AMS'96, GGR'03, DF'03, ...]

• Example: Let $pos$ denote the position of the highest unmarked stair,
  \[ E(pos) \approx \log_2(0.775351 \ n) \]
  \[ \sigma^2(pos) \approx 1.12127 \]

• Standard variance reduction methods apply

• Either $O(\log n)$ or $O(\log \log n)$ space
• The COUNT sketches of [FM'85] are equivalent to the paintball process.
  – Start with a bit-vector of all zeros.
  – For each item,
    * Use its ID and a hash function for coin flips.
    * Pick a bit-vector entry.
    * Set that bit to one.
• These sketches are duplicate-insensitive:

∀A,B (Sketch(A) \ join Sketch(B)) = Sketch(A \cup B)
The Sampling Problem
Sampling From a Stream

• Fundamental prob: sample m items uniformly from stream
  – Useful: approximate costly computation on small sample
• Challenge: don’t know how long stream is
  – So when/how often to sample?
• Two solutions, apply to different situations:
  – Reservoir sampling (dates from 1980s?)
  – Min-wise sampling (dates from 1990s?)
Reservoir Sampling

- Sample first $m$ items
- Choose to sample the $i^{th}$ item ($i > m$) with probability $m/i$
- If sampled, randomly replace a previously sampled item
- Optimization: when $i$ gets large, compute which item will be sampled next, skip over intervening items. [Vitter 85]
Reservoir Sampling

- Analyze simple case: sample size $m = 1$
- Probability $i$'th item is the sample from stream length $n$:
  - Prob. $i$ is sampled on arrival $\times$ prob. $i$ survives to end

\[
\frac{1}{i} \times \frac{i}{i+1} \times \frac{i+1}{i+2} \cdots \frac{n-2}{n-1} \times \frac{n-1}{n} = \frac{1}{n}
\]

- Case for $m > 1$ is similar, easy to show uniform probability
- Drawbacks of reservoir sampling: hard to parallelize
Min-wise Sampling

- For each item, pick a random fraction between 0 and 1
- Store item(s) with the smallest random tag [Nath et al.’04]

Each item has same chance of least tag, so uniform
- Can run on multiple streams separately, then merge