Lecture #23

CS 170

Spring 2021

Streaming Algorithms

- Motivation
  - Vast amounts of data "streaming" by, too much to store
    - Search engine tracking clicks on websites
    - Router monitoring network traffic
    - Data arriving from sensors
  - Is there a much (exponentially) smaller data structure that we can quickly update on the fly, and query when needed?
  -

3 Examples of Streaming Algorithms

- Simple: Counting Total Sales
  - Input: n sales with prices $p_1, p_2, \ldots p_n$
  - Desired output: $P = \sum_{i=1}^{n} p_i$
  - Initialize $C=0$, Update $C=C+p_i$, Query: return C
  - Memory Requirement: $\lceil \log_2 P \rceil$
- Morris's Alg. for Approximate Counting
  -
  -
- Flajolet & Martin (FM) Alg. for Distinct Elements
  -
  -

Randomized Approximate Counting

- Goal: compute estimate $\tilde{n}$ of n where

- Chebyshev's Inequality:

- X, Y independent $\Rightarrow$
-

First Try at Randomized Counting

Initialize: $c = 0$

Update: $c = c + 1$ with probability p

Query: return $\tilde{n} = c/p$

-

- Thm: $\mathbb{E}(\tilde{n}) =$

- Thm: $\mathrm{Var}(\tilde{n}) =$

- Do we save any bits?

- Are we accurate?

Morris's Algorithm: (1/3)

Initialize: $X = 0$

Update: $X = X + 1$ with probability

Query: return $\tilde{n} =$

-
-

Morris's Algorithm:

Initialize: $X = 0$

Update: $X = X + 1$ with probability $\frac{1}{2^X}$

Query: return $\tilde{n} = 2^X - 1$

- Let $X_n = X$ after n updates
- Thm: $\mathbb{E}(\tilde{n}) = \mathbb{E}(2^{X_n} - 1) = n$
- Intuition:

-

Morris's Algorithm:

Initialize: $X = 0$

Update: $X = X + 1$ with probability $\frac{1}{2^X}$

Query: return $\tilde{n} = 2^X - 1$

- Let $X_n = X$ after n updates
- Thm: $\mathbb{E}(\tilde{n}) = \mathbb{E}(2^{X_n} - 1) = n$
- Thm: $\mathrm{Var}(\tilde{n}) =$

-

Making Morris's Algorithm more accurate

- Run s "copies" of Morris, yielding $\tilde{n}_1, \ldots, \tilde{n}_s$, return average $\tilde{n} = \frac{1}{s}\sum_{i=1}^{s} \tilde{n}_i$
-
-
-
-

Flajolet + Muller (FM) Alg. for Distinct Element Counting

- Given stream $i_1, i_2, \ldots, i_m$, each $i_j \in \{1, \ldots, n\}$
  count $t = \#$ distint elements in stream

-

-

-

Idealized FM Alg.

- Goal: count $t = \#$distinct elements in $i_1, .., i_m$, $i_j \in \{1..n\}$
- Pick random function $h : \{1, .., n\} \to [0, 1]$
  -

- Initialize: $X = 1$
  Update: $X =$
  Query: return $\tilde{t} =$
- Intuition:

Idealized FM Alg.

- Goal: count $t = \#$distinct elements in $i_1, \dots, i_m$, $i_j \in \{1..n\}$
- X = min of t uniform random i.i.d numbers in $[0,1]$
- Thm: $\mathbb{E}(X) = \frac{1}{t+1}$

Proof: Analogous to discrete case where

$$\mathbb{E}(X) = \sum_{i=1}^{\infty} i\, p(i) = \sum_{i=1}^{\infty} \sum_{j=i}^{\infty} p(j) = \sum_{i=1}^{\infty} P(X \geq i)$$

$\mathbb{E}(X) =$

$=$

$=$

$=$

Making Idealized FM More Accurate

- Same as Morris: run s copies, average results
- Thm: $\mathrm{Var}(X) =$

  Proof:

- Thm: run $s = \lceil \quad \rceil$ independent copies $FM_1, \ldots, FM_s$

Proof:

Making FM Practical

- Can't generate uniform random real numbers in practice, need an approximation
-
-
- Recent version: