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?
- Monte Carlo algorithms: fast, probably accurate

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
  - Like above, with $p_i = 1$
  - Goal: use $O(\log_2 \log_2 n)$ bits, not $\lceil \log_2 n \rceil$
- Flajolet & Martin (FM) Alg. for Distinct Elements
  - Count # distinct integers among $i_1, \ldots, i_m$
  - Goal: use $O(\log_2 n)$ bits, $i_j \in \{1, \ldots, n\}$

Randomized Approximate Counting

- Goal: compute estimate $\tilde{n}$ of n where
  $$P(|\tilde{n}-n| > \varepsilon \cdot n) < \delta$$
  for some $0 < \varepsilon, \delta < 1$, that you can choose, using $O(\log_2(\log_2 n))$ bits
- Chebyshev's Inequality:
  $$P(|X - \mathbb{E}(X)| > \lambda) = P(|X - \mathbb{E}(X)|^2 > \lambda^2)$$
  $$\leq \mathbb{E}(|X-\mathbb{E}(X)|^2)/\lambda^2 \quad \text{by Markov's Ineq.}$$
  $$= \mathrm{Var}(X)/\lambda^2 \quad \text{definition of Variance}$$
- X, Y independent $\Rightarrow \mathrm{Var}(aX+bY) = a^2\mathrm{Var}(X) + b^2\mathrm{Var}(Y)$
- Ex: $X_1, \ldots, X_n$ independent, identically distributed (i.i.d.)
  $$S = \frac{1}{n}\sum_{i=1}^{n} X_i \Rightarrow \mathbb{E}S = \mathbb{E}(X_i), \ \mathrm{Var}(S) = \frac{1}{n}\mathrm{Var}(X_i)$$

First Try at Randomized Counting

Initialize: $c = 0$

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

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

- Let $X_i = 1$ w.p. p, 0 w.p. $1-p$, i.i.d. $\Rightarrow c = \sum_{i=1}^{n} X_i$
- Thm: $\mathbb{E}(\tilde{n}) = \frac{1}{p}\mathbb{E}(c) = \frac{1}{p}\sum_{i=1}^{n}\mathbb{E}(X_i) = \frac{1}{p}\sum_{i=1}^{n} p = n$
- Thm: $\mathrm{Var}(\tilde{n}) = \frac{1}{p^2}\mathrm{Var}(c) = \frac{1}{p^2}\sum_{i=1}^{n}\mathrm{Var}(X_i) = \frac{n}{p^2}(p-p^2) = \frac{n(1-p)}{p}$
- Do we save any bits?

  $\#\text{bits} = \lceil \log_2 c \rceil \approx \lceil \log_2 np \rceil \approx \log_2 n - \log_2 \frac{1}{p} \Rightarrow$ save $\log_2 \frac{1}{p}$ bits
- Are we accurate? Not if we save many bits ($p \ll 1$)

  $$\mathbb{P}(|n - \tilde{n}| > \varepsilon n) \leq \frac{\mathrm{Var}(\tilde{n})}{(\varepsilon n)^2} = \frac{(1-p)}{\varepsilon^2 p n}$$

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: $\mathbb{E}(\tilde{n}) = \mathbb{E}(2^{X_n} - 1) = n$, or $\mathbb{E}(2^{X_n}) = n+1$

Proof: induction on n; base case is $n = 0$

$$\mathbb{E}(2^{X_{n+1}}) = \sum_{j=0}^{\infty} P(X_n = j \text{ and we increment } X) \cdot 2^{j+1}$$
$$+ P(X_n = j \text{ and we don't}) \cdot 2^j$$
$$= \sum_{j=0}^{\infty} P(X_n = j) \cdot \frac{1}{2^j} \cdot 2^{j+1} + P(X_n = j) \cdot \left(1 - \frac{1}{2^j}\right) 2^j$$
$$= \sum_{j=0}^{\infty} P(X_n = j)(2 + 2^j - 1) = \sum_{j=0}^{\infty} P(X_n = j) + \sum_{j=0}^{\infty} P(X_n = j) \cdot 2^j$$
$$= 1 + \mathbb{E}(2^{X_n}) = 1 + (n+1) = n+2$$

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: we are approximating $\log_2 n$ instedof $n \Rightarrow$ need $\log_2(\log_2 n)$ bits, exponentially fewer than $\log_2 n$ needed for n
- Could we use even fewer bits to approximate n to within a factor $1\pm\varepsilon$? No: need to distinguish $[1,(1+\varepsilon)^2), [(1+\varepsilon)^2,(1+\varepsilon)^4) \cdots [(1+\varepsilon)^k,(1+\varepsilon)^{k+2}), \ldots [(1-\varepsilon)n,(1+\varepsilon)n)$

$\#$intervals $= \frac{\log_2 n(1+\varepsilon)}{\log_2(1+\varepsilon)^2} = \Theta\left(\frac{\log_2 n}{\varepsilon}\right) \Rightarrow$ need $\Omega(\log_2\log_2 n)$ bits

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: $\mathrm{Var}(\tilde{n}) = \frac{1}{2}n^2 - \frac{1}{2}n - 1$

Proof: $\mathbb{E}(2^{2X_n}) = (3/2)n^2 + (3/2)n + 1$ by induction

$$\mathbb{E}(2^{2X_{n+1}}) = \sum_{j=0}^{\infty} P(X_n = j \text{ and we increment } X) \cdot 2^{2(j+1)}$$

$$+ P(X_n = j \text{ and we don't}) \cdot 2^{2(j)}$$

$$= \dots = (3/2)(n+1)^2 + (3/2)(n+1) + 1$$

$$\mathrm{Var}(\tilde{n}) = \mathrm{Var}(2^{X_n}) = \mathbb{E}(2^{2X_n}) - (\mathbb{E}(2^{X_n}))^2 = \dots = \frac{1}{2}n^2 - \frac{1}{2}n - 1$$

- Chebyshev: $P(|\tilde{n} - n| > \varepsilon n) \le \frac{1}{\varepsilon^2 n^2} \frac{n^2}{2} = \frac{1}{2\varepsilon^2}$. oops

Making Morris's Algorithm more accurate

- Run s "copies" of Morris, yielding $\tilde{n}_1, \ldots, \tilde{n}_s$, return average $\tilde{n} = \frac{1}{s} \sum_{i=1}^{s} \tilde{n}_i$
  - $\mathbb{E}(\tilde{n}) = \mathbb{E}(\tilde{n}_i) = n$, $\mathrm{Var}(\tilde{n}) = \frac{1}{s} \mathrm{Var}(\tilde{n}_i)$
- $P(|n - \tilde{n}| > \varepsilon n) \le \frac{1}{s} \frac{1}{2\varepsilon^2} < \delta$ if $s > \frac{1}{2\varepsilon^2 \delta} = \Theta\left(\frac{1}{\varepsilon^2 \delta}\right)$
- How many bits do we need?
- Intuition: s copies of Morris $\Rightarrow O\left(\frac{1}{\varepsilon^2 \delta} \cdot \log_2 \log_2 n\right)$ bits
- More carefully: with probability $1-\delta$, need

$$O\left(\frac{1}{\varepsilon^2 \delta} \cdot \log_2 \log_2 \frac{n}{\varepsilon \delta}\right) \text{ bits}$$

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

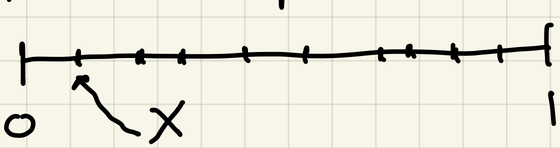
- Given stream $i_1, i_2, \ldots, i_m$, each $i_j \in \{1, \ldots, n\}$ count $t = \#$ distint elements in stream
- Ex if stream is $1, 2, 7, 2, 3, 7$, $t = 4$
- Straightforward solutions:
  1) Keep array of n bits y_j, initially all $y_j = 0$
     set $y_{i_k} = 1$ when i_k appears
  2) Store whole stream in memory $\Rightarrow O(m \cdot \log_2 n)$ bits
- Goal: use $o(n)$ bits to compute $\tilde{t}$
  where $P(|t - \tilde{t}| > \varepsilon t) < \delta$

Idealized FM Alg.

- Goal: count $t = \#$distinct elements in $i_1, \dots, i_m$, $i_j \in \{1..n\}$
- Pick random function $h : \{1, \dots, n\} \to [0,1]$
  - Each $h(i)$ is i.i.d. random real number uniformly distributed in $[0,1]$
- Initialize: $X = 1$

  Update: $X = \min(X, h(i))$

  Query: return $\tilde{t} = \frac{1}{X} - 1$
- Intuition: X = min of t distinct uniform random i.i.d. numbers in $[0,1]$, we expect $X \sim \frac{1}{t+1}$, so $t \sim \frac{1}{X} - 1$



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

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

$$\mathbb{E}(X) = \int_0^{\infty} P(X > \lambda)\, d\lambda = \int_0^1 P(X > \lambda)\, d\lambda$$

$$= \int_0^1 P(\text{all } t \text{ random\#s are} > \lambda)\, d\lambda$$

$$= \int_0^1 \prod_{i=1}^{t} P(\text{one random\# is} > \lambda)\, d\lambda$$

$$= \int_0^1 \prod_{i=1}^{t} (1-\lambda)\, d\lambda = \int_0^1 (1-\lambda)^t\, d\lambda = \frac{1}{t+1}$$

Making Idealized FM More Accurate

- Same as Morris: run s copies, average results
- Thm: $\mathrm{Var}(X) = \frac{t}{(t+1)^2(t+2)}$

  Proof: Follows from

$$\mathbb{E}(X^2) = \int_0^1 P(X^2 > \lambda)\,d\lambda = \int_0^1 P(X > \sqrt{\lambda})\,d\lambda$$

$$= \int_0^1 (1-\sqrt{\lambda})^t\,d\lambda \qquad \text{substitute } u = 1-\sqrt{\lambda}$$

$$= 2\int_0^1 u^t(1-u)\,du = \frac{2}{(t+1)(t+2)}$$

- Thm: run $s = \lceil \frac{1}{\varepsilon^2\delta} \rceil$ independent copies $FM_1, \ldots, FM_s$ with outputs $X_1, \ldots, X_s$, $Z = \frac{1}{s}\sum_{i=1}^{s} X_i$, output $\tilde{t} = 1/Z - 1$

  Then $P(|\tilde{t} - t| > O(\varepsilon)t) \leq \delta$

Proof: Apply Chebyshev's Ineq. to Z

Making FM Practical

- Can't generate uniform random real numbers in practice, need an approximation
- Generate random integers $\in \{0, 1, \ldots, B\}$, divide final minimum by B
- Use "pseudorandom number generators" to approximate actual random numbers
    - Uses $O(\log n + \log B)$ bits
- Recent version: Hyperloglog