Data Mining and Analysis

Streaming Data

CSCE 676 :: Fall 2019

Texas A&M University
Department of Computer Science & Engineering
Prof. James Caverlee

Resources

MMDS Chapter 4 + slides

- http://infolab.stanford.edu/~ullman/mmds/ch4.pdf
- http://www.mmds.org/mmds/v2.1/ch04-streams1.pdf
- http://www.mmds.org/mmds/v2.1/ch04-streams2.pdf

Carlos Castillo course on Data Mining [https://github.com/chatox/data-mining-course]

Counting distinct elements

Motivation

Let n(u,h) be the number of nodes reachable through a path of length up to h from node u

Naïve method

Maintain a set for each node u, initialize S(u) = {u}

Repeat h times:

$$S(u) = S(u) \cup \bigcup_{v \text{ neighbor of } u} S(v)$$

Answer n(u,h) = |S(u)|

Problem?

Solution: Consider each node

- We will receive a stream of items
  - Our neighbors at distance <= h
  - Repeated many times because of loops
- We want to use a small amount of memory
- We don’t care which items are in the stream
- We just want to know how many are

Counting Distinct Elements

Data stream consists of a universe of elements chosen from a set of size N

Maintain a count of the number of distinct elements seen so far

Solution?

- Maintain the set of elements seen so far
- That is, keep a hash table of all the distinct elements seen so far

More Applications

How many different words are found among the Web pages being crawled at a site?

How many different Web pages does each customer request in a week?

How many distinct products have we sold in the last week?

Using Small Storage

Real problem: What if we do not have space to maintain the set of elements seen so far?

Estimate the count in an unbiased way

Accept that the count may have a little error, but limit the probability that the error is large

Probabilistic Counting (Morris

$x \leftarrow 0$

For each of the n events:

$x \leftarrow x + 1$ with probability $(1/2)^x$

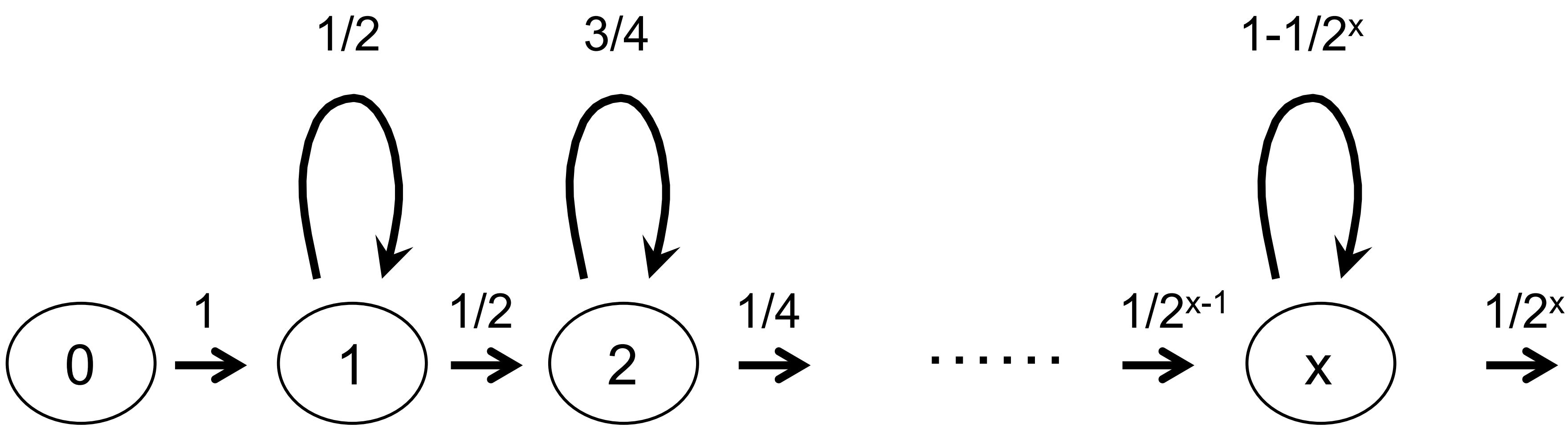
Return estimate $n' = 2^x+1$

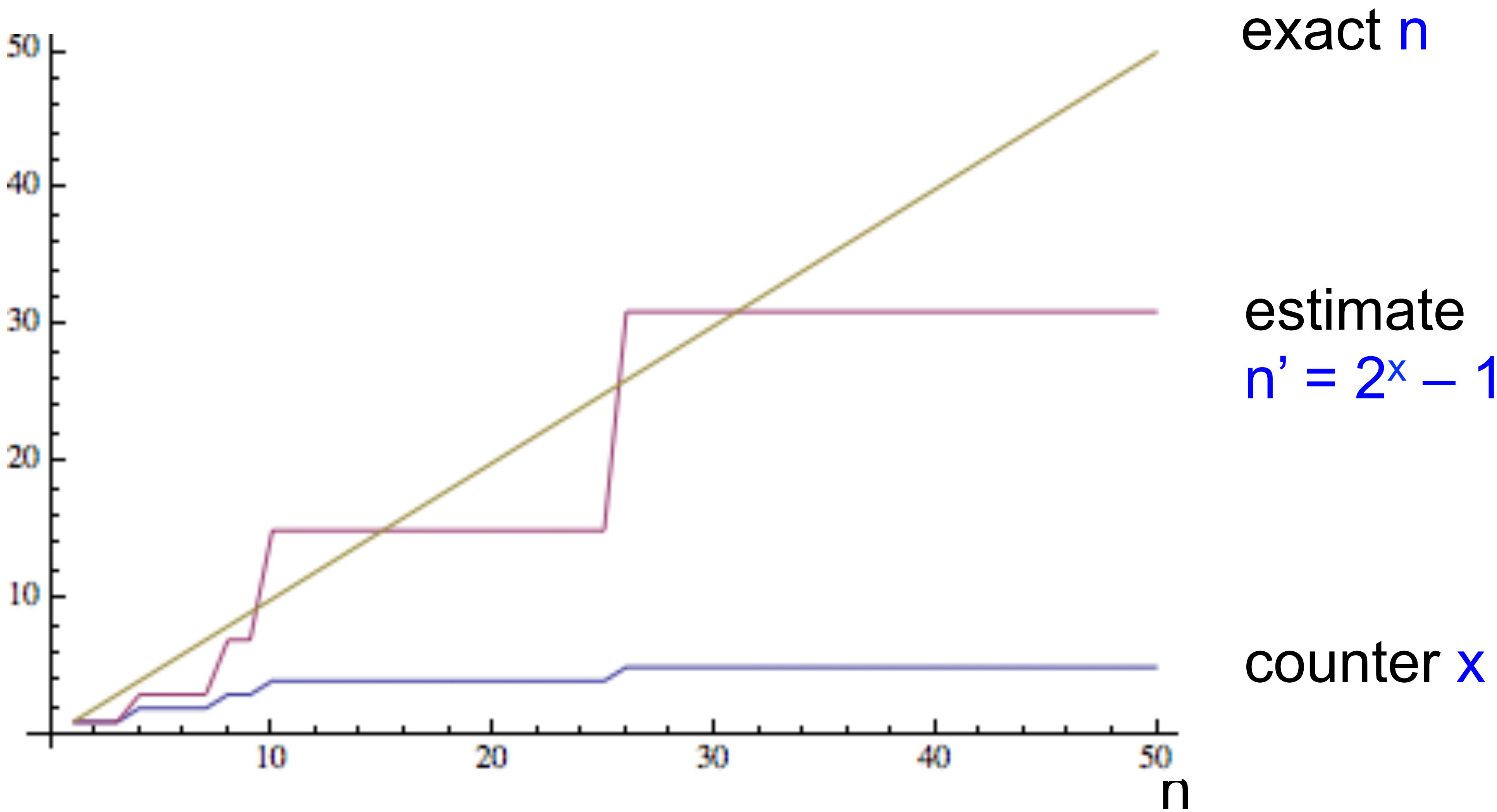
Counter x needs only $\log_2(n)$ bits

Morris Algorithm: Birth Process

Let X(n) denote count after arrival n

Transition x→x+1 with probability 2^{-x}




50
40
30
20
10
10
20
30
40
50
n
exact n
estimate
n’ = 2^x – 1
counter x

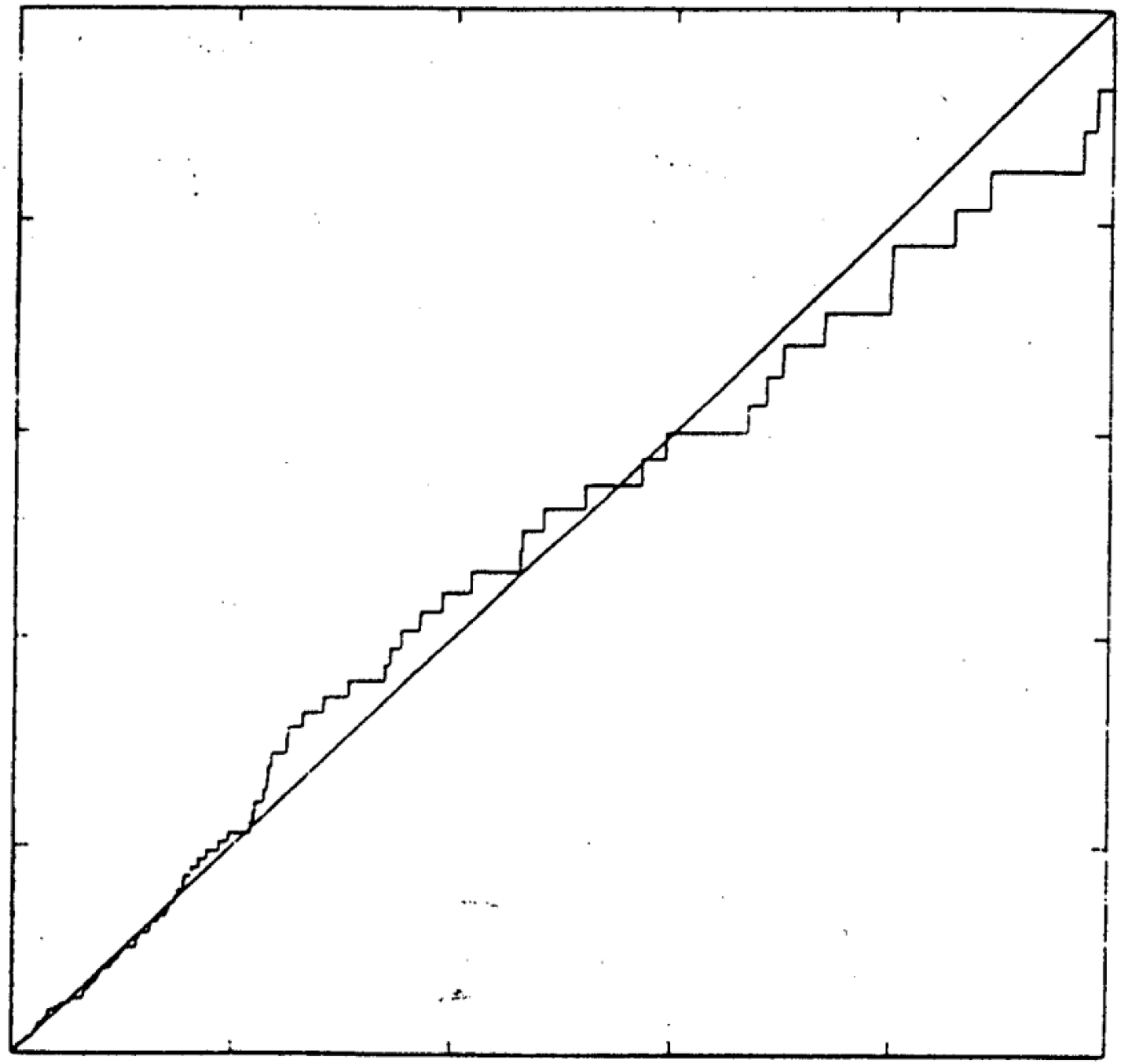


Fig. 3. A simulation of the linear estimate D_n of approximate counting plotted against n, for $0 \leqq n \leqq 10^5$.

Aside

Robert Morris Sr —> Robert Morris Jr

…

The three golden rules to ensure computer security are:

do not own a computer;

do not power it on; and

Morris algorithm: Unbiasedness

Initialize $x = 0$; increment w.p. $p = 2^{-x}$; 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: $x = 1$; $p_1 = 1/2$

prob. 1/2: x stays at 1; $n'=2^1 - 1=1$

prob. 1/2: $x \rightarrow 2$; $n'=2^2 - 1=3$

Flajolet-Martin Algorithm for

For every element u in the stream, compute hash h(u)

Let r(u) be the number of trailing zeros in hash value

Example: if `h(u) =` `001011101`<u>`000`</u> then r(u) = 3

Maintain R = max r(u) seen so far

Output 2^R as an estimate of the number of distinct elements seen so far

Flajolet-Martin explained

Let $r(u)$ be the number of trailing zeroes in hash value, keep $R = \max r(u)$, output 2^R as estimate

Repeated items don't change our estimates because their hashes are equal

- About 1/2 of distinct items hash to *******0
  - To actually see a *******0, we expect to wait until seeing 2 distinct items
- About 1/4 of distinct items hash to ******00
  - To actually see a ******00, we expect to wait until seeing 4 items
- …

If we actually saw a hash value of ***000…0 (having R trailing zeroes), then in expectation we saw 2^R different items

Flajolet-Martin explained (formal)

Let m be the number of distinct elements

Let z(r) be the probability of finding a tail of r zeroes

We will prove that

$z(r) \to 1$ if $m >> 2^R$

$z(r) \to 0$ if $m << 2^R$

Hence, 2^R should be around m

Flajolet-Martin explained

Probability a hash value ends in r zeroes = $(1/2)^r$

Assuming h(u) produces values at random

Probability a random binary ends in r zeroes = $(1/2)^r$

Probability of seeing m distinct elements and NOT seeing a tail of r zeroes = $(1 - (1/2)^r)^m$

Flajolet-Martin explained

Probability of seeing m distinct elements and NOT seeing a tail or r zeroes = $(1-(1/2)^r)^m$

Remember $(1-\varepsilon)^{1/\varepsilon} \simeq 1/e$ for small ε

Hence,

$$\left(1-\left(\frac{1}{2}\right)^r\right)^m = \left(1-\left(\tfrac{1}{2}\right)^r\right)^{\frac{m\left(\frac{1}{2}\right)^r}{\left(\frac{1}{2}\right)^r}} \approx \left(\frac{1}{e}\right)^{\left(\frac{m}{2^r}\right)}$$

Flajolet-Martin explained

Probability of seeing m distinct elements and NOT seeing a tail of r zeroes ≈ $(1/e)^{(m/2r)}$

If $m >> 2^r$, this tends to 0

- We almost certainly will see a tail of r zeroes

If $m << 2^r$, this tends to be 1

- We almost certainly will not see a tail of r zeroes

Hence, 2^r should be around m

Flajolet-Martin: Increasing

Idea: repeat many times or compute in parallel for multiple hash functions

How to combine?

Average? $E[2^r]$ is infinite, extreme values will skew the number excessively

Median? 2^r is always a power of 2

Solution: group hash functions, take median values obtained in each group,

Recall: Counting Neighbors

ANF: A Fast and Scalable Tool for Data Mining in Massive Graphs

Christopher R. Palmer
Computer Science Dept
Carnegie Mellon University
Pittsburgh, PA
crpalmer@cs.cmu.edu

Phillip B. Gibbons
Intel Research Pittsburgh
Pittsburgh, PA
phillip.b.gibbons@intel.com

Christos Faloutsos
Computer Science Dept
Carnegie Mellon University
Pittsburgh, PA
christos@cs.cmu.edu

// Set $\mathcal{M}(x, 0) = \{x\}$
FOR each node x DO
    $M(x, 0) =$ concatenation of k bitmasks
        each with 1 bit set ($P(\text{bit } i) = .5^{i+1}$)
FOR each distance h starting with 1 DO
    FOR each node x DO $M(x, h) = M(x, h-1)$
    // Update $\mathcal{M}(x, h)$ by adding one step
    FOR each edge (x, y) DO
        $M(x, h) = (M(x, h)$ BITWISE-OR $M(y, h-1))$
    // Compute the estimates for this h
    FOR each node x DO
        Individual estimate $I\hat{N}(x, h) = (2^b)/.77351$
        where b is the average position of the least zero bits
        in the k bitmasks
    The estimate is: $\hat{N}(h) = \sum_{\text{all x}} I\hat{N}(x, h)$

Computing Moments

Moments of order k

If a stream has A distinct elements, and each element has frequency m_i

The kth order moment of the stream is $\sum_{i \in A} (m_i)^k$

The 0th order moment is the number of distinct elements in the stream

Moments of order k

The kth order moment of the stream $\sum_{i \in A} (m_i)^k$ is

The 2nd order moment is also known as the "surprise number" of a stream

Large values = more uneven

mi	i=1	i=2	i=3	i=4	i=5	i=6	i=7	i=8	i=9	i=10	i=11	2nd mom
Seq1	9	9	9	9	9	9	9	9	9	9	9	910
Seq2	90	1	1	1	1	1	1	1	1	1	1	8110

Method for Second Moment

Assume (for now) that we know n, the length of the stream

We will sample s positions

For each sample we will have X.element and X.count

We sample s random positions in the stream

  - X.element = element in that position, X.count ← 1
  - When we see X.element again, X.count ← X.count + 1

Estimate second moment as n(2 * X.count - 1)

Method for Second Moment

Example: a, b, c, b, d, a, c, d, a, b, d, c, a, a, b

$m_a = 5$, $m_b = 4$, $m_c = 3$, $m_d = 3$

Second moment $= 5^2 + 4^2 + 3^2 + 3^2 = 59$

Suppose we sample $s = 3$ variables X_1, X_2, X_3

Suppose we pick the 3rd, 8th, and 13th position at random

X_1.element = c, X_2.element = d, X_3.element = a

X_1.count = 3, X_2.count = 2, X_3.count = 2 (we count forward only)

Estimate n(2*X.count -1), first estimate = 15(6-1)=75, second estimate 15(4-1)=45, third estimate

Why does this work?

Let e(i) be the element in position I of the stream

Let c(i) be the number of times e(i) appears in positions i, i+1, i+2, …, n

Example: a, b, c, b, d, a, c, d, a, b, d, c, a, a, b

c(6) = ?

Why does this work?

c(i) is the number of times e(i) appears in positions i, i+1, i+2, …, n

E[n*(2*X.count-1)] is the average of n(2c(i)-1) over all positions i=1,…n

$$E[n(2 \times X.\mathrm{count} - 1)] = \frac{1}{n}\sum_{i=1}^{n} n(2c(i) - 1)$$

$$E[n(2 \times X.\mathrm{count} - 1)] = \sum_{i=1}^{n} (2c(i) - 1)$$

Why does this work?

Now focus on element a that appears m_a times in the stream

The last time a appears this term is 2c(i)-1 = 2*1-1 = 1

Just before that, 2c(i)-1=2*2-1 = 3

…

Until $2m_a$-1 for the first time a appears

Hence

$$E[n(2 \times X.\text{count} - 1)] = \sum_a 1 + 3 + 5 + \cdots + (2m_a - 1) = \sum_a m_a^2$$

For higher order moments

For second order moment

We use $n(2v-1) = n(v^2-(v-1)^2)$

For third order moment

We use $n(3v^2-3v+1) = n(v^3-(v-1)^3)$

For k^{th} order moment

We use $n(v^k - (v-1)^k)$

For infinite streams

We use a reservoir sampling strategy

If we want s samples

- Pick the first s elements of the stream setting X_i.element$\leftarrow$e(i) and X_i.count $\leftarrow$ 1 for i=1…s
- When element n+1 arrives
  - Pick X_{n+1}.element with probability s/(n+1), evicting one of the existing elements at random and setting X.count$\leftarrow$1