Computational Frameworks: Streaming

(Part 2)

OUTLINE

1. Introduction to the streaming model (Part 1)
2. Warm-up: finding the majority element (Part 1)
3. Sampling (Part 1)
   - Uniform sampling with reservoir sampling
   - Finding (approximate) frequent items
4. Sketching
   - Counting distinct elements with probabilistic counting
   - Estimating individual frequencies and the second moment
5. Filtering: Bloom filters for membership problem

Sketching

Sketch: space-efficient data structure that can be used to provide (typically probabilistic) estimates of (statistical) characteristics of a data stream.

Frequency Moments

Consider a stream $\Sigma = x_1, x_2, \ldots, x_n$, whose elements belong to a universe U.

For each $u \in U$ occurring in Σ define its (*absolute*) *frequency*

$$f_u = |\{j : x_j = u, 1 \leq j \leq n\}|,$$

i.e., the number of occurrences of u in Σ

Definition: Frequency Moments

For every $k \geq 0$, the k-th frequency moment F_k of Σ is

$$F_k = \sum_{u \in U} f_u^k.$$

(We assume $0^0 = 0$)

Frequency Moments

Relevant informations from frequency moments:

- $\boxed{F_0 = \text{number of distinct items in } \Sigma.}$ It is useful, for instance, in the analysis of web-logs.
- $\boxed{F_1 = |\Sigma|}$: trivial to maintain with a counter.
- $\boxed{1 - F_2/|\Sigma|^2 = \text{Gini index of } \Sigma.}$ It provides info on data skew: the closer to 0 and the higher is the skew. It is used for decision tree induction.

$\Sigma = m$ elementi con n/m occorrenze

$$F_2 = \sum_{u \in U} f_u^2 = \sum_{u \in U} \left(\frac{n}{m}\right)^2 = m \frac{n^2}{m^2} = \frac{n^2}{m}$$

$$\text{Gini index} = 1 - \frac{F_2}{|\Sigma|^2} = 1 - \frac{n^2}{m} \cdot \frac{1}{n^2} = 1 - \frac{1}{m}$$

$\Sigma = 1$ elemento, n occorrenze; $|U| = m$

$$F_2 = \sum_{u \in U} f_u^2 = n^2; \quad \text{Gini index} = 1 - \frac{n^2}{n^2} = 0$$

Estimating F_0 for Σ (i.e., distinct elements)

Exact computation: use $|U|$ counters or dictionary with $|F_0|$ elements (not suited for streaming setting).

Approximation: $\boxed{\text{Probabilistic counting algorithm}}$ ([Flajolet, Martin 1983])

- Working memory $O(\log |U|)$. ⇒ serve upper bound sei U per allocare
- F_0 estimated within a factor c with probability $\geq 1 - 2/c$ (accuracy-confidence tradeoff).
- **Main idea:** funzione hash
  - Map each $u \in U$ to a random integer $h(u) \in [0, |U| - 1]$.
    $\Rightarrow h(u)$ is a $O(\log |U|)$-bit binary string.
  - The more distinct elements in Σ, the more likely to have a $u \in \Sigma$ mapped to a string with many trailing 0's.

Probabilistic counting algorithm

We need the following ingredients:

- Array C of $\lceil \log |U| \rceil + 1$ bits, all initialized to zero.
- Hash Function $h : U \rightarrow [0..|U|-1]$. We assume: $\Pr[h(u) = x] = \frac{1}{|U|}$
  - For every $u \in U$, $h(u)$ has uniform distribution in $[0..|U|-1]$
  - All $h(u)$'s are pairwise independent.
- Notation: for $i \in [0..|U|-1]$, define

$$tr(i) = \text{number of trailing zeroes in binary representation of } i$$

e.g., $12 = (1100)_2 \Rightarrow tr(12) = 2$

Algorithm: **For each** $x_j \in \Sigma$ **do** $C[tr(h(x_j))] \leftarrow 1$

After processing x_n, estimate F_0 as

$$\tilde{F}_0 = 2^R,$$

where R is the largest index of C with $C[R] = 1$.

Example

$\Sigma = A, D, A, A, C, B, F, F, B, A, E, C$

	$h \Rightarrow$ mappato in 5 bit
A	10110
B	11000
C	10101
D	11001
E	10011
F	01000

$A \to h(A) = 10110 \Rightarrow tr(A) = 1$

$D \to h(D) = 11001 \Rightarrow tr(D) = 0$

$\vdots$

$C \to h(C) = 10101 \Rightarrow tr(C) = 0$

$B \to h(B) = 11000 \Rightarrow tr(B) = 3$

$F \to h(F) = 01000 \Rightarrow tr(F) = 3$

$\vdots$

$E \to h(E) = 10011 \Rightarrow tr(E) = 0$

$$C = \begin{array}{cccccc} 1 & 1 & & 1 & & \\ \cancel{0}, & \cancel{0}, & 0, & \cancel{0}, & 0, & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 \end{array}$$

$\Downarrow$

$R = 3$

$\Downarrow$

$\tilde{F}_0 = 2^3 = 8$

Idea: ogni volta che ho nuovo el., ripeto alg. x volte in parallelo, poi prendo mediana

Why does it work? (Intuition)

For simplicity, assume $|U|$ a power of 2.

For $x \in \Sigma$, what is the probability that $h(x)$ has at least j trailing zeros?

- $\text{Prob}(tr(h(x)) \geq 1) = \frac{1}{2}$
- $\text{Prob}(tr(h(x)) \geq 2) = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$

.....

- $\text{Prob}(tr(h(x)) \geq j) = \left(\frac{1}{2}\right)^{j}$

m el. distinti

$X_j = \#$ el. con $tr(X_j) \geq j$

$$X_j = \sum_{u \in U} Y_{u,j} \; , \; Y_{u,j} = \begin{cases} 1 & \text{if } tr(u) \geq j \\ 0 & \text{else} \end{cases} \Rightarrow \mathbb{E}[Y_{u,j}] = \frac{1}{2^j}$$

$$= \sum_{u \in \text{Eunici}} Y_{u,j}$$

$$\mathbb{E}[X_j] = \mathbb{E}\left[\sum Y_{uj}\right] = \sum \mathbb{E}[Y_{uj}] = \sum \frac{1}{2^j} = \frac{m}{2^j}$$

Se $2^j > m$, $\frac{m}{2^j} < 1 \Rightarrow$ non mi aspetto che posizione j sia 1

$\mathbb{E}[X_1] = n/2^1 = n/2$

$\mathbb{E}[X_1] = n/2^2 = n/4$

$\vdots$

$\mathbb{E}[X_j] = n/2^j \sim 1 \Rightarrow$ ci aspettiamo che $\log_2 n$ sia posizione uguale a 1
+ a destra

$\mathbb{E}[X_{j+1}] = n/2^{j+1} < 1$

$\Downarrow$

conviene avere + esecuzioni

Probabilistic guarantees

The following theorem summarizes the properties of the algorithm.

Theorem

For a stream Σ *of* n *elements, the* **probabilistic counting algorithm** *returns a value* $\tilde{F}_0$ *such that, for any* $c > 2$: ⇒ non in algoritmo

teorema vale per ogni c

$$\Pr(\tilde{F}_0 < F_0/c) \le 1/c \quad \text{and} \quad \Pr(\tilde{F}_0 > cF_0) \le 1/c,$$

sottostimare

sovrastimare

hence

union bound

$$Pr(F_0/c \le \tilde{F}_0 \le cF_0) \ge 1 - 2/c.$$

The algorithm requires a working memory of $O(\log |U|)$ *bits, 1 pass, and* $O(\log |U|)$ *time per element.*

Exercises

Exercise: High Probability Guarantees (median trick)

Consider a stream Σ with elements from a universe U and let F_0 be the number of distinct elements in Σ. Suppose that you run ℓ *independent instances* of the probabilistic counting algorithm, obtaining ℓ estimates $\tilde{F}_0^{(j)}$ for F_0, with $1 \leq j \leq \ell$. Assume ℓ odd and let $\tilde{F}_0$ be the mediam value of the $\tilde{F}_0^{(j)}$'s. Show that with a suitable choice of $\ell = \Theta(\log |U|)$ we have: ⇒ per fallire, almeno $\ell/2$ fallimenti

$$\boxed{\Pr(\tilde{F}_0 < F_0/16) \leq 1/|U| \quad \text{and} \quad \Pr(\tilde{F}_0 > 16F_0) \leq 1/|U|.}$$

Hint: Use the previous theorem and the Chernoff bound.

Estimating individual frequencies and F_2

Consider a stream $\Sigma = x_1, x_2, \ldots, x_n$, whose elements belong to U, with $|U| = M$. Recall that for each $u \in U$ its frequency in Σ is

$$f_u = |\{j : x_j = u, 1 \leq j \leq n\}|,$$

OUR OBJECTIVE: in one pass over Σ we want to compute a **small sketch** that enables to derive **unbiased estimates** of

- f_u for any given $u \in U$ (*individual frequencies*)
- $F_2 = \sum_{u \in U}(f_u)^2$ (*second moment*)

with provable space-accuracy tradeoffs

Observation: clearly, the exact computation of all f_u's or of F_2 might require space proportional to $|\Sigma|$.

Count-min sketch

The first approach we consider is based on the count-min sketch invented by [Cormode,Muthukrishnan 2003].

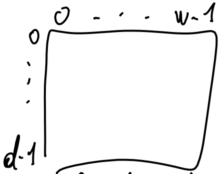
non è unbiased estimator

Main ingredients

- $d \times w$ array C of counters ($O(\log n)$ bits each)
- d hash functions: $h_0, h_1, \ldots, h_{d-1}$, with

$$h_j : U \to \{0, 1, \ldots, w-1\},$$

for every j.



hash function diversa per riga

Note that d and w are *design parameters* that regulate the space/time-accuracy tradeoff.

Count-min sketch: algorithm

Initialization: $C[j, k] = 0$, for every $0 \leq j < d$ and $0 \leq k < w$.

For each x_t in Σ **do**

For $0 \leq j \leq d - 1$ **do** $C[j, h_j(x_t)] \leftarrow C[j, h_j(x_t)] + 1$;

At the end of the stream: for any $u \in U$, its frequency f_u can be estimated as:

$$\tilde{f}_u = \min_{0 \leq j \leq d-1} C[j, h_j(u)].$$

Example: $n = 15, d = 3, w = 3$

$\Sigma = A, B, C, B, D, A, C, D, A, B, D, C, A, A, B$

u, f_u	h_0	h_1	h_2
A, 5	1	2	2
B, 4	2	3	2
C, 3	1	1	3
D, 3	2	2	3

Array C

	1	2	3
h_0	3	3	0
h_1	1	2	2
h_2	0	4	2

- $\tilde{f}_A = \min\{3,3,4\} = 3$
- $\tilde{f}_B = \min\{3,2,4\} = 2$
- $\tilde{f}_C = \min\{3,1,2\} = 1$
- $\tilde{f}_D =$

partizioni per MRFFT : 16
⇓
repartition on RDD

collect 16 coreset ⇒ 16k

R1: unisco 16 coreset → dopo, posso fare action (e.g. count) prima cache o persist (meglio)
R2: collect

Count-min sketch: analysis

We assume:

- the d hash functions $h_1, h_2, \ldots, h_d$ are mutually independent
- for each $j \in [0, d-1]$ and each $u, v \in U$ with $u \neq v$, $h_j(u)$ and $h_j(v)$ are independent random variables uniformly distributed in $[0, w-1]$.

Theorem

Consider a $d \times w$ count-min sketch for a stream Σ of length n, where $d = \log_2(1/\delta)$ and $w = 2/\epsilon$, for some $\delta, \epsilon \in (0,1)$. The sketch ensures that for any given $u \in U$ occurring in Σ trade-off qualità - numero colonne

$$\tilde{f}_u - f_u \leq \epsilon \cdot n,$$

with probability $\geq 1 - \delta$. $\Rightarrow$ trade-off probabilità - numero righe

Obs.: The bias in the estimated frequencies discourages their use to estimate the second moment F_2.

Dim: dato $u \in U$, $\Pr[\tilde{f}_u - f_u > \varepsilon n] < \delta$

$\tilde{f}_u = \min_{j \in [0, d-1]} C[j, h_j(u)]$

Prendiamo riga j:

$C[j, h_j(u)] = f_u + \text{noise}$

$\text{noise} = \sum_{\substack{v \neq u \\ v \in U}} Y_v$; $Y_v = \begin{cases} 1 & \text{if } h_j(u) = h_j(v) \\ 0 & \text{else} \end{cases}$

$$\mathbb{E}[Y_v] = \Pr[h_j(v) = h_j(u)] = 1/w = \sum_{j=0}^{w-1} \Pr[h_j(v) = h \mid h_j(u) = h] \cdot \Pr[h_j(u) = h]$$

$$\mathbb{E}[C[j, h_j(u)]] = \mathbb{E}[f_u] + \mathbb{E}[\text{noise}] = f_u + \sum_{\substack{v \neq u \\ v \in U}} \mathbb{E}[Y_v] = f_u + \frac{|U|}{w} \leq$$

$$\leq f_u + \frac{n}{w} = f_u + \frac{n}{2}\varepsilon$$

Markov's inequality $\Pr[X > c] \le \frac{\mathbb{E}[X]}{c}$

$$\Pr[C[j, h_j(u)] > f_u + \varepsilon n] \le \frac{1}{2}$$

$$\Pr[C[j, h_j(u)] > \mathbb{E}[\dots]/2] \le \frac{1}{2}$$

Prendiamo riga minima $\Rightarrow$ dobbiamo fallire in ognuna

$$\Pr[\hat{f}_u - f_u > \varepsilon n] = \prod_{i=0}^{d-1} \Pr[\hat{f}_{u_j} - f_u > \varepsilon n] = \left(\frac{1}{2}\right)^d = \left(\frac{1}{2}\right)^{\log_2 1/\delta} = \delta$$

Count sketch

The count sketch was invented by [Charikar,Chen,Farach-Colton 2002], and can be seen as an **unbiased variant of the count-min sketch**.

IDEA: for each item $u \in U$ multiply its contributions to each row by a value in $\{-1, +1\}$ randomly selected, so to cancel out collisions.

Main ingredients

- $d \times w$ array C of counters ($O(\log n)$ bits each)
- d hash functions: $h_0, h_1, \ldots, h_{d-1}$, with
  $$h_j : U \to \{0, 1, \ldots, w-1\},$$
  for every j.
- d hash functions: $g_0, g_1, \ldots, g_{d-1}$, with
  $$g_j : U \to \{-1, +1\},$$
  for every j.

Count sketch: algorithm

Initialization: $C[j,k] = 0$, for every $0 \le j < d$ and $0 \le k < w$.

For each x_t in Σ **do**

For $0 \le j \le d-1$ **do** $C[j, h_j(x_t)] \leftarrow C[j, h_j(x_t)] + g_j(x_t)$;

At the end of the stream: for any $u \in U$ and $0 \le j < d$, let

$$\tilde{f}_{u,j} = g_j(u) \cdot C[j, h_j(u)].$$ senza rumore, $C[j, h_j(u)] = \pm f_u$

The frequency of u can be estimated as:

$$\tilde{f}_u = \text{median of the } \tilde{f}_{u,j}\text{'s}$$

Example: $n = 15, d = 3, w = 3$

$\Sigma = \cancel{A}, \cancel{B}, \cancel{C}, \cancel{B}, D, A, C, D, A, B, D, C, A, A, B$

u, f_u	h_0	g_0	h_1	g_1	h_2	g_2
A, 5	1	+1	2	+1	2	+1
B, 4	2	-1	3	+1	2	-1
C, 3	1	-1	1	-1	3	+1
D, 3	2	-1	2	+1	3	+1

Array C

j	1	2	3
0	$\cancel{0} \cancel{+1}\ 0$	$\cancel{0} \cancel{-1}\ -2$	0
1	$\cancel{0}\ -1$	$\cancel{0}\ +1$	$\cancel{0} \cancel{+1}\ +2$
2	0	$\cancel{0} \cancel{+1} \cancel{0}\ -1$	$\cancel{0}\ +1$

$\tilde{f}_A = \text{median}\{0, 1, -1\} = 0$

$\tilde{f}_B = \text{median}\{2, 2, 1\} = 2$

- $\tilde{f}_A =$

- $\tilde{f}_B =$

- $\tilde{f}_C =$

- $\tilde{f}_D =$

Count sketch: analysis

Assumptions: for both sets of hash functions (the h_j's and the g_j's) we make the same assumptions of independence and uniform distribution, which we made for the h_j's in the analysis of the count-min sketch.

Theorem

Consider a $d \times w$ count sketch for a stream Σ of length n, where $d = \log_2(1/\delta)$ and $w = O\left(1/\epsilon^2\right)$, for some $\delta, \epsilon \in (0,1)$. The sketch ensures that for any given $u \in U$ occurring in Σ:

A
- $\mathrm{E}[\tilde{f}_{u,j}] = f_u$, *for any $j \in [0, d-1]$, i.e., $\tilde{f}_{u,j}$ is an unbiased estimator of f_u;*

da provare a casa

- *With probability $\geq 1 - \delta$,*

$$|\tilde{f}_u - f_u| \leq \epsilon \cdot \sqrt{F_2},$$

where $F_2 = \sum_{u \in U} (f_u)^2$ (true second moment).

Intuition. Due to the random signs, on average the "noise" created by several items colliding on the same colum as a u, cancel out.

Count-min sketch vs count sketch

COUNT-MIN: errore assoluto $\leq \varepsilon n$

COUNT: " " $\leq \varepsilon \sqrt{F_2}$ ← migliore

$F_2 = \sum_{u \in U} f_u^2 \leq \left(\sum_{u \in U} f_u \right)^2 \leq n^2$

Suggerimento dim. A: $Y_v = \begin{cases} +f_v & \text{se } h_j(u) = h_j(v),\ g_j(v) = +1 \Rightarrow \Pr[\ldots] = \frac{1}{2w} \\ 0 & \text{se } h_j(u) \neq h_j(v) \Rightarrow \Pr[\ldots] = \frac{1}{2w} \\ -f_v & \text{se } h_j(u) = h_j(v),\ g_j(v) = -1 \Rightarrow \Pr[\ldots] = 1 - \frac{1}{w} \end{cases}$

$\mathbb{E}[Y_v] = \frac{1}{2w} f_v - \left(1 - \frac{1}{w}\right) f_v = 0$

Estimation of F_2

Given a $d \times w$ count sketch for Σ, define

$$\tilde{F}_{2,j} = \sum_{k=1}^{w} (C[j,k])^2 \quad \text{for } 0 \leq j < d$$

We can derive the following estimate for the true second moment F_2:

$$\tilde{F}_2 = \text{median of the } \tilde{F}_{2,j}\text{'s}$$

Example (same as before)

$\Sigma = A, B, C, B, D, A, C, D, A, B, D, C, A, A, B$

$F_2 = (f_A)^2 + (f_B)^2 + (f_C)^2 + (f_D)^2 = 5^2 + 4^2 + 3^2 + 3^2 = 59.$

Array C from before		

- Estimate from row $j = 0$:
- Estimate from row $j = 1$:
- Estimate from row $j = 2$:

$\Rightarrow \tilde{F}_2 =$

Analysis of $\tilde{F}_2$

The following theorem can be proved under the same assumptions made for the analysis of the count sketch

come count-min, stima F_2 ha molto + rumore

Theorem

Consider a $d \times w$ count sketch for a stream Σ of length n, where $d = \log_2(1/\delta)$ and $w = O\left(1/\epsilon^2\right)$, for some $\delta, \epsilon \in (0,1)$. The sketch ensures that:

- $\mathrm{E}[\tilde{F}_{2,j}] = F_2$ *for any* $0 \le j < d$. *That is, any* $\tilde{F}_{2,j}$ *is an unbiased estimator of* F_2.
- *With probability* $\ge 1 - \delta$,

$$|\tilde{F}_2 - F_2| \le \epsilon \cdot \sqrt{F_2}.$$

In the following slides, we show that $\mathrm{E}[\tilde{F}_{2,j}] = F_2$ for every j, while we skip the proof of the second bullet point.

Analysis of performance metrics

Both count-min and count sketches can be computed in **1 pass**

To assess space and time performance, we assume:

- Each hash function can be applied in constant time
- The space occupied by the sketch dominates over the one needed to store the hash functions $\Rightarrow$ e.g. $h(x)=(ax+q) \bmod p$ $\Rightarrow$ salvare solo p, q
  $\Downarrow$ prime

For both sketches we have

- **Working memory:** $\boxed{O(d \cdot w)}$, which becomes $\boxed{O(\log(1/\delta)/\epsilon)}$ for the count-min sketch, and $\boxed{O(\log(1/\delta)/\epsilon^2)}$, for the count sketch, in order to attain the proabilistic accuracy stated before.
- **Processing time per element:** $\boxed{O(d) = O(\log(1/\delta))}$

Moreover, given the sketch, the estimates $\tilde{f}_u$'s (individual frequencies) and $\tilde{F}_2$ (second moment) can be computed in $\boxed{O(d) \text{ and } O(d \cdot w)}$ time, respectively.

Filtering

Motivation

For many applications, processing a data stream $\Sigma = x_1, x_2, \ldots$ entails essentially the identification of the x_i's which meet a certain criterion.

Some criteria can be checked very easily with a minimum cost in terms of space and time. However, this is not always the case.

Example. Suppose that the x_i's are email addresses and that when x_i arrives we need to check whether it belongs to a set S of *verified* addresses. If S is very large (e.g., 1 billion addresses of approximately 20 bytes each), we face two issues:

- If S does not fit into main memory, it must be stored on disk.
- Standard exact techniques to check $x_i \in S$, especially if S is on disk, may be time consuming and not compatible with a high arrival rate.

Can we check membership efficiently with reasonable accuracy?

Bloom filter

Approximate membership problem

Given a stream $\Sigma = x_1, x_2, \ldots$ of elements from some universe U, and let S be a set of m elements from U. Store S into a compact data structure that, for any given x_i, allows to check whether $x_i \in S$ with

- no error, when $x_i \in S$ (No false negatives)
- small probability error, when $x_i \notin S$ (Small false positive rate)

A solution to the problem comes from the **Bloom filter**, introduced in [Bloom 1970]. Its **main ingredients** are:

- Array A of n bits, all initially 0.
- k hash functions: $h_0, h_1, \ldots, h_{k-1}$, with

$$h_j \ : \ U \to \{0, 1, \ldots, n-1\} \qquad \text{for every } 0 \leq j < k$$

Note that n and k are *design parameters* that regulate the tradeoff between space/time and accuracy.

Bloom filter

Initialization:

For each $e \in S$ **do**

For $0 \leq j < k$ **do** $A[h_j(e)] \leftarrow 1$;

Membership test: for any x_i in Σ if

$$x_i \in S \quad \Leftrightarrow \quad A[h_0(x_i)] = A[h_1(x_i)] = \cdots = A[h_{k-1}(x_i)] = 1$$

Straighforward properties:

- The approach ensures that there are no false negatives
- Assuming that $k \ll n$, and that the hash functions can be stored compactly, the required working memory is dominated by the storage of $A \Rightarrow n$ bits.
- Assuming that each hash function can be applied in $O(1)$ time, the membership test requires $O(k)$ time.

Example

- $S = \{A, B, C\}$
- $n = 12$
- $k = 2$

	A	B	C
h_0	0	1	4
h_1	4	9	9

The resulting array is:

0	1	2	3	4	5	6	7	8	9	10	11
1	1	0	0	1	0	0	0	0	1	0	0

Bloom filter: analysis of false positive rate

Assumptions: for the set of hash functions (the h_j's) we make the same assumptions of independence and uniform distribution, which we made in the analysis of the count-min sketch.

Theorem

Suppose that n is sufficiently large. For any given x_i which does not belong to S, the probability that x_i is erroneously claimed to be in S is

$$\Pr(A[h_j(x_i)] = 1 \textit{ for each } 0 \leq j < k) \simeq (1 - e^{-km/n})^k$$

This probability is referred to as false positive rate.

Email example: In the case of email addresses mentioned before, $m = 10^9$ and storing the entire set S would require 20GB (assuming that each email takes 20 bytes). Using a Bloom filter with $n = 8m$ (hence $|A| = 1\text{GB}$), and $k = 6$, the false positive rate is about 2.15%.

Exercise

Consider a stream $\Sigma = x_1, x_2, \ldots, x_n$ of n measurements from sensors. Each measurement x_i is a pair (k_i, w_i), where k_i is the ID of a sensor and w_i is the value of the measurement (an integer). For a given sensor u occurring in Σ define

$$f_u = \sum_{(k_i, w_i) \in \Sigma \,:\, k_i = u} w_i,$$

i.e., the aggregate measurements taken by u.

1. Briefly describe a space-efficient unbiased estimator for $\sum_u (f_u)^2$, where the sum is over all sensors occurring in the stream.
2. What can you say about the unbiasedness of your estimator?

Exercise

Consider a Bloom filter built to assess membership for a set S of m elements. The Bloom filter consists of a n-bit array A and k hash functions $h_0, h_1, \ldots, h_{k-1}$, mutually independent and with values uniformly distributed in $[0, n-1]$. Assume that n is even, and that each hash function h_i is such that, for every $e \in S$, $h_i(e) \bmod n/2$ is uniformly distributed in $[0, n/2-1]$.

1. Show how to transform, in $O(n)$ time, the given Bloom filter into a new Bloom filter based on an $n/2$-bit array B, and describe how to assess membership with the new Bloom filter.
2. Compute the probability that a given cell $B[i]$ of the new array is 0.

References

- **[LRU14]** J. Leskovec, A. Rajaraman and J. Ullman. Mining Massive Datasets. Cambridge University Press, 2014. Chapter 4 (Sections 4.3-4.5) (pdf provided in Moodle)

- **[DF08]** C. Demetrescu and I. Finocchi. Algorithms for Data Streams. In *Handbook of Applied Algorithms*, Chapter 8, Section 3. Wiley-IEEE Press, 2008. (pdf provided in Moodle)
- **[CGHJ12]** G. Cormode, M.N. Garofalakis, P.J. Haas, C. Jermaine. Synopses for Massive Data: Samples, Histograms, Wavelets, Sketches. Foundations and Trends in Databases 4(1-3): 1-294, 2012. Chapter 5 (Sections 5.1-5.3) (pdf provided in Moodle)