BIG DATA

Streaming Algorithms - 2

K V Subramaniam
Computer Science and Engineering

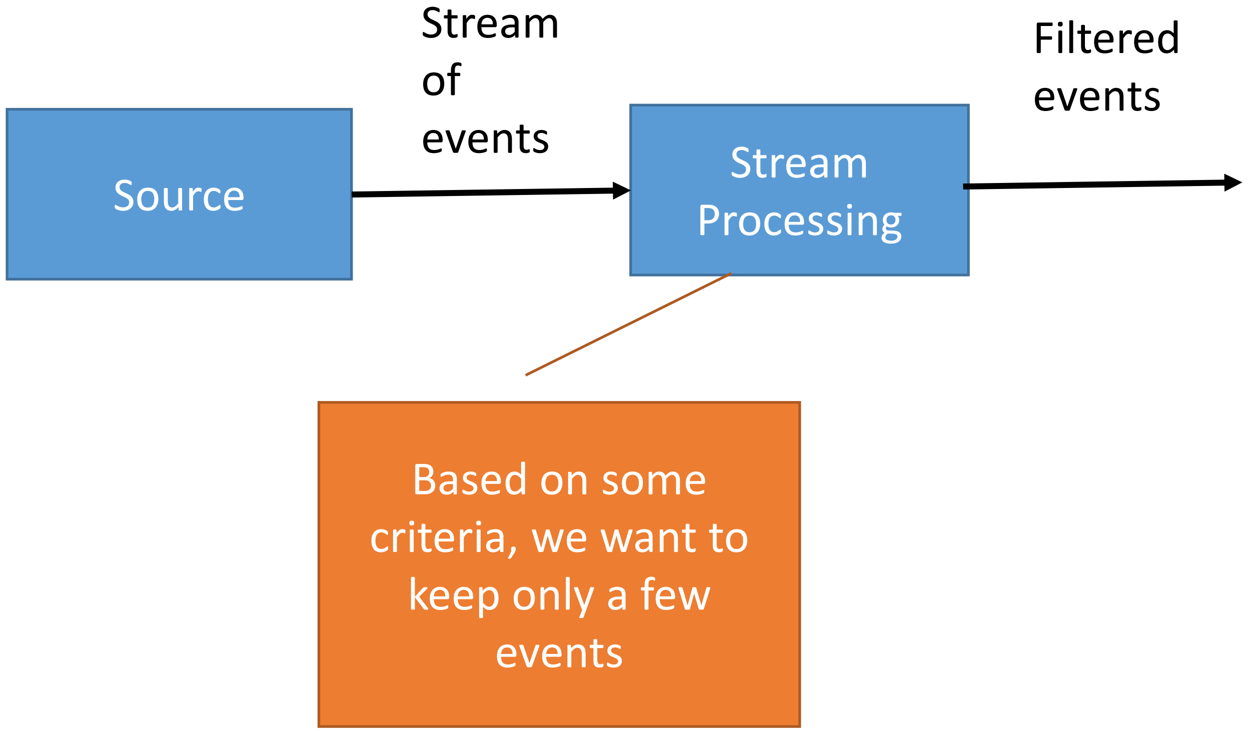
Overview: Streaming Algorithms

- Filtering algorithms – Bloom Filter
  - Motivation
  - General bloom filters
  - Extensions
- Counting unique elements
  - Motivation
  - Flajolet Martin Algorithm and working
  - Practical considerations

Bloom Filters

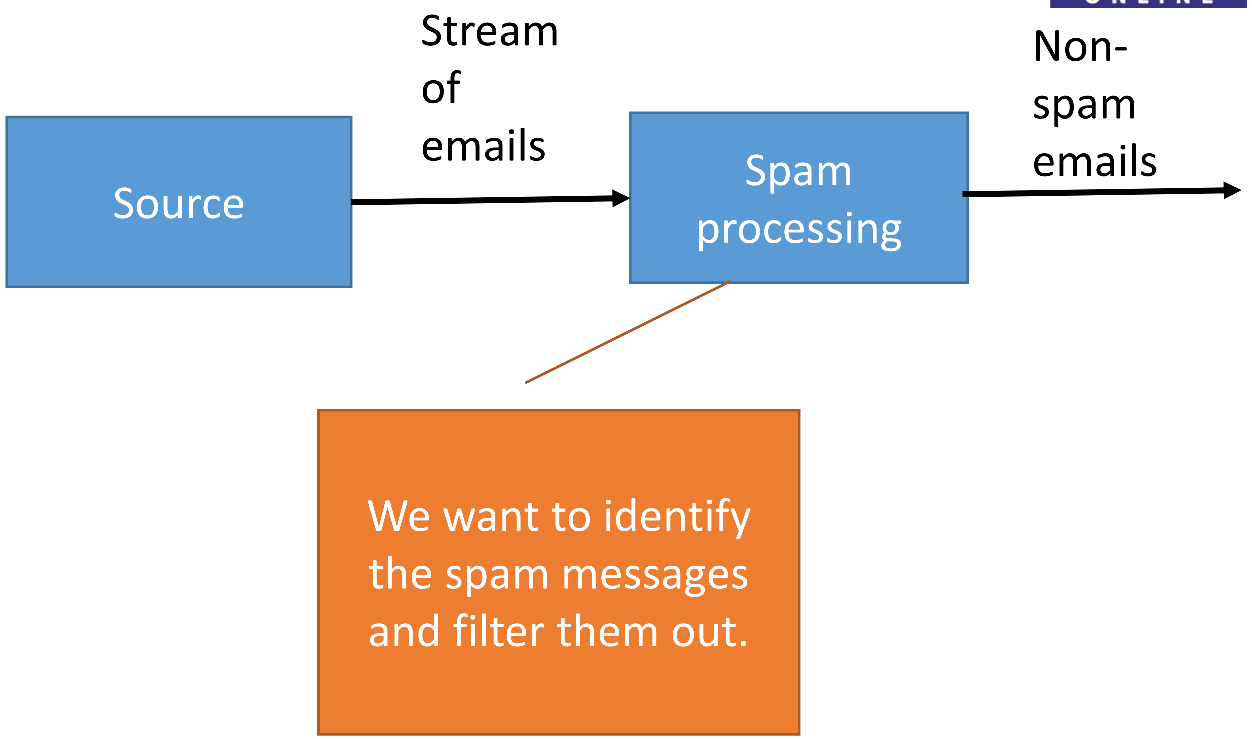
Filtering Data

- Sometimes we need to take a decision
  - To filter out certain events
  - The decision has to be taken instanteously
  - Large number of events need to be processed.



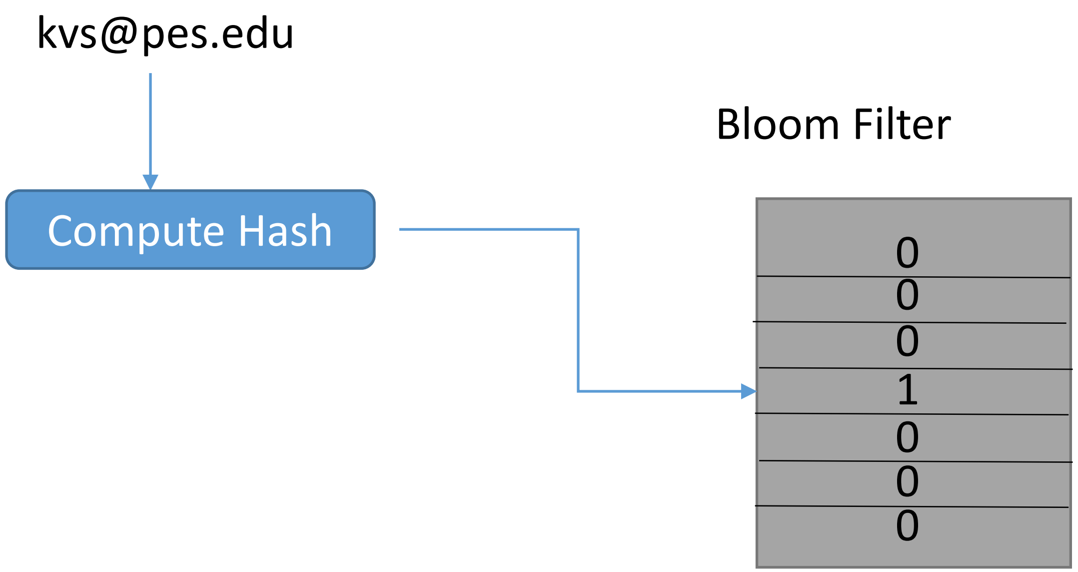
Filtering Data – Motivational example

- Incoming email
  - Remove all spam emails
- Constraints
  - 1GB of main memory
  - 1 billion non-spam email ids (well known)
  - 20 bytes/email address
- Can’t store all email ids in memory
  - Disk is a slow store



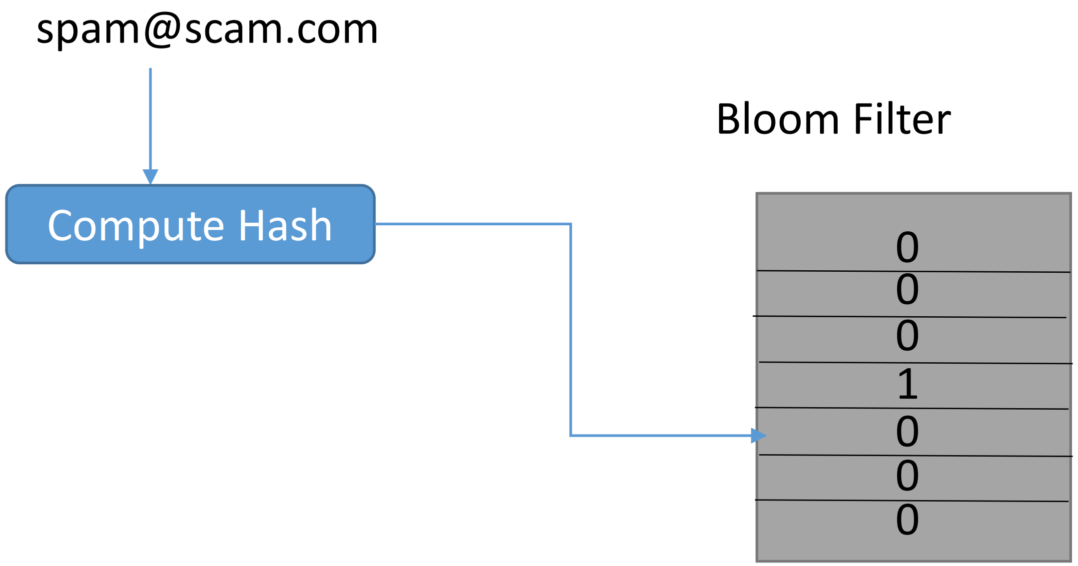
Bloom filter basic: initialization

- 1 GB memory => 8 billion bit string
- Bloom filter initialization
  - Hash non-spam email ids to 0..8 billion-1
  - Set corresponding bit to 1

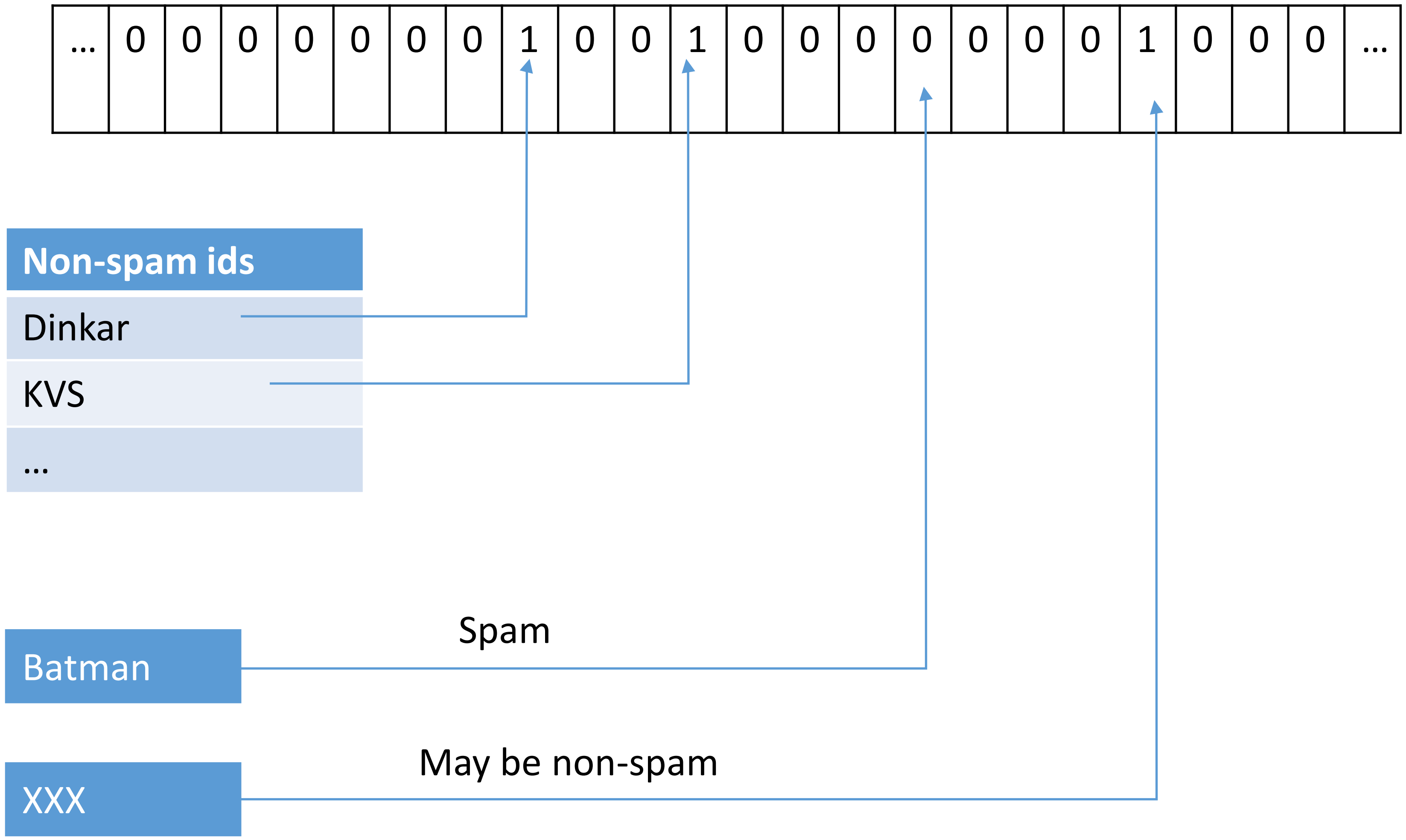


Bloom filter basic: initialization

- Usage
  - Hash incoming email id
  - Check bloom filter entry
    - If (0)
      - Definitely has not been seen before → it is a spam
    - If (1)
      - Not sure if it has been seen before



Bloom filter basic: illustration



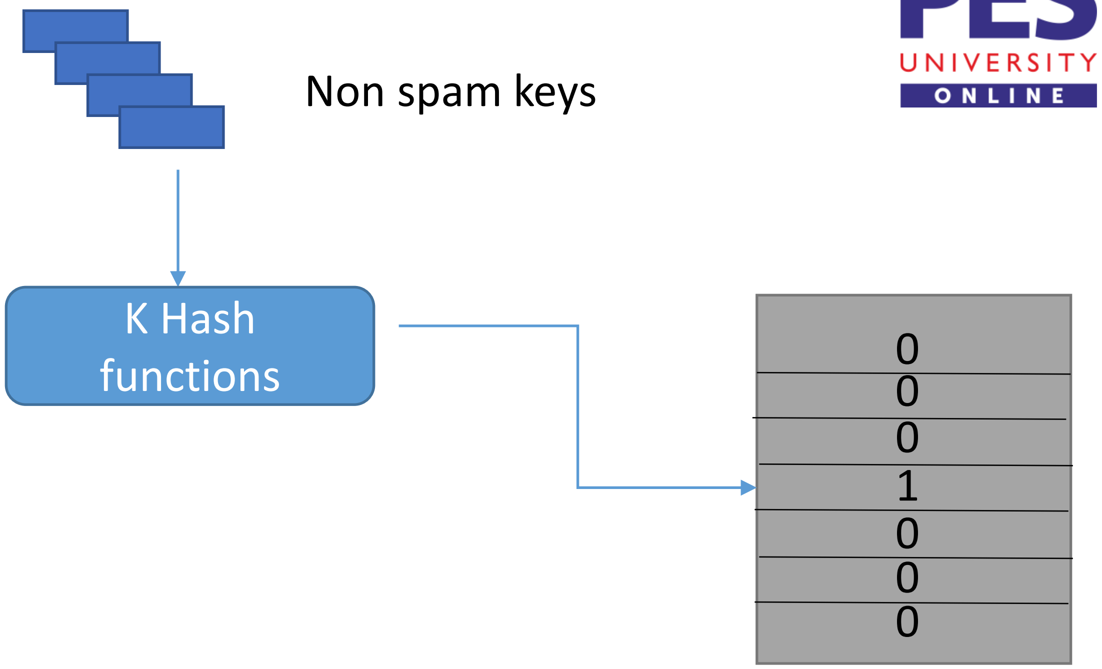
General Bloom Filters

General Bloom Filters

Bloom filter consists of

- Array of n bits (size of memory in example)
- A collection of k hash functions h1, ..., hk
- A set S of keys with m elements (non-spam email ids in example)
- Purpose: given a key a, determine if it is in S (in example, given email id, determine if non-spam)

- Initialization: for all keys in S,
  - Compute the k hash functions
  - Set corresponding bits to 1



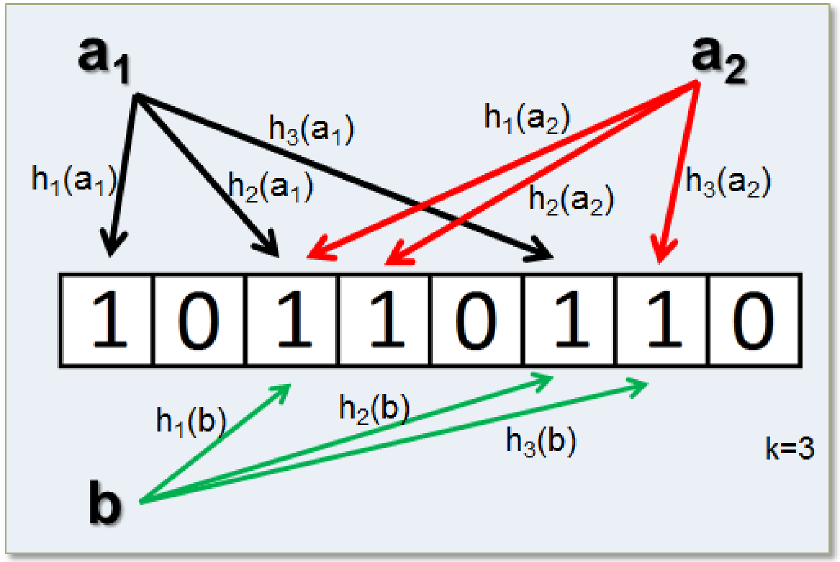
General Bloom Filters

- Usage
  - Hash a using the k hash functions
  - If all the corresponding k bits are 1, a€S

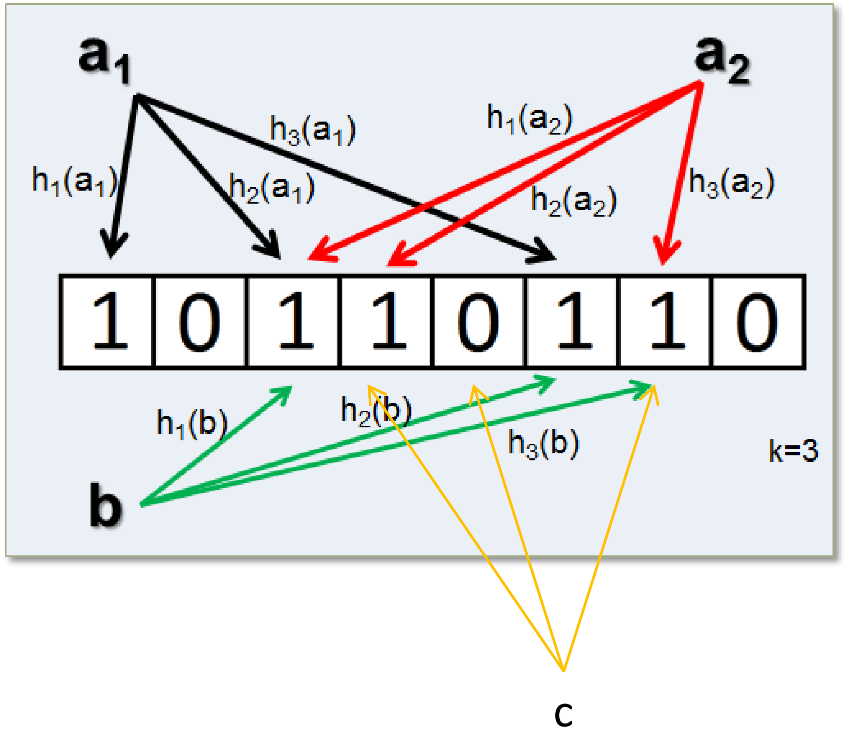
- Probability of false positive $(1 - e^{-km/n})^k$.
  - Read derivation in book

General Bloom Filters

- Top
  - Shows the insertion
- Bottom
  - Shows the check for set membership
  - B is checked to see if it is part of the bloom filter.

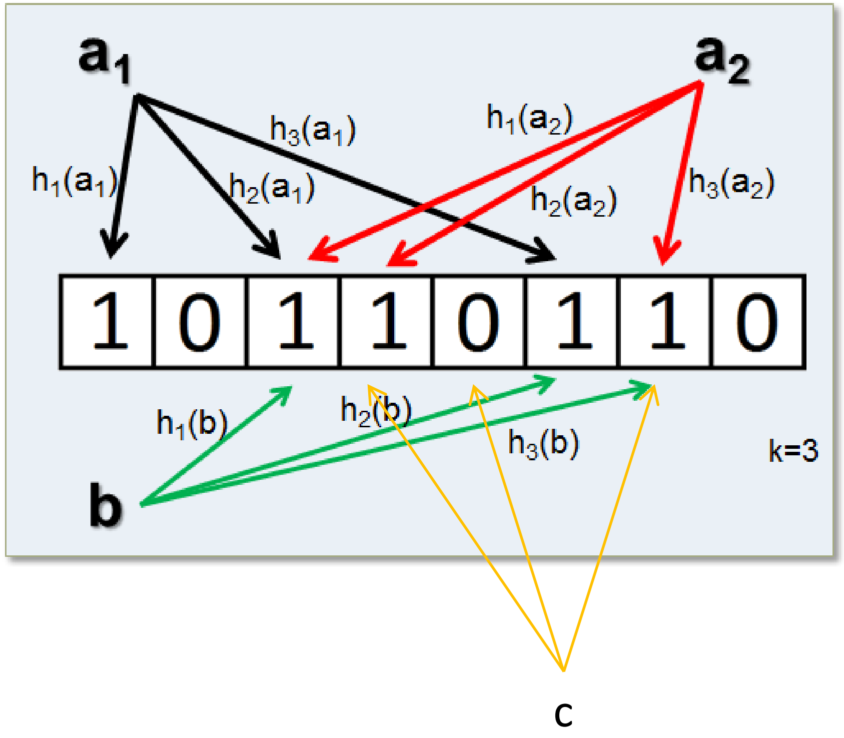


Exercise



- Suppose c hashes as shown
- Is c spam, not spam, or possibly spam?

Exercise - solution

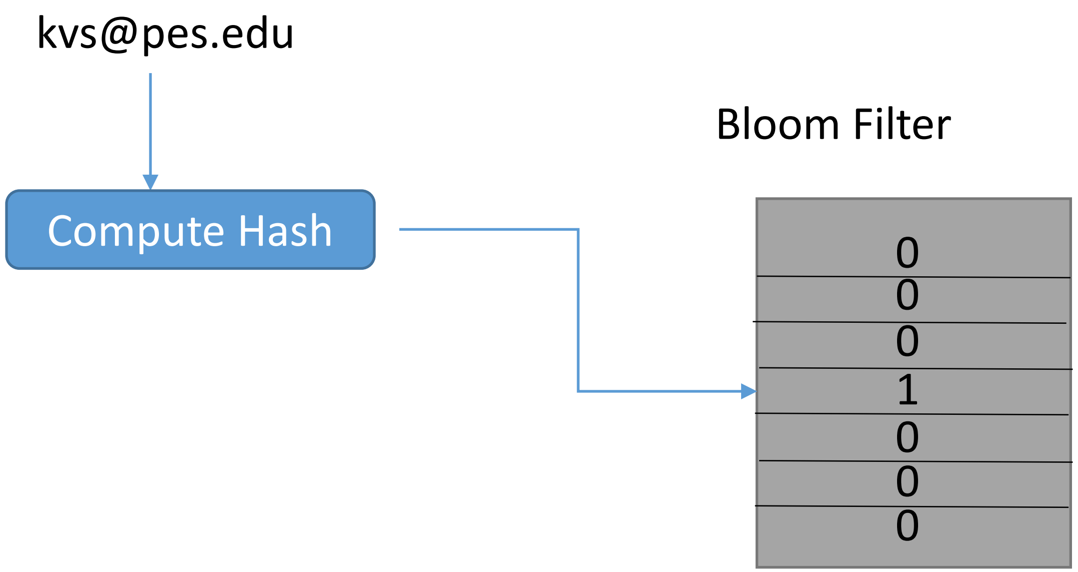


- Suppose c hashes as shown
- Is c spam, not spam, or possibly spam?
- C – is a spam email as it hashes to one bucket that contains a 0

Bloom filter extensions

Bloom filter basic: extensions

- Use secondary storage
  - 7/8 of the time, we can filter from memory
    - 7/8 of the time, a spam id will hash to 0
    - Because, there are 8B bits, 1B are 1, 7B are 0
  - 1/8 of the time, verify non-spam by disk lookup



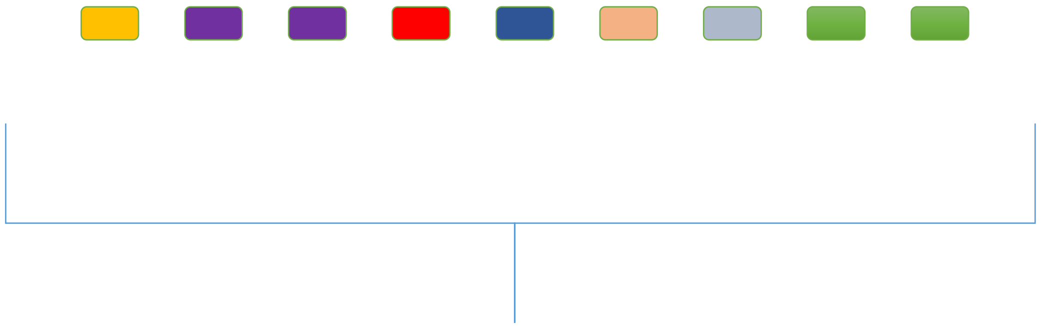
Cascaded Bloom filters

- We can have 2 Bloom filters in series with different hashes
- If bit is 1, use second Bloom filter
- We will reject much more of the spam

Counting Distinct Elements

Motivation

- Website that wants to know how many distinct users visit
  - Amazon: have userid
  - Google: have to use IP address
  - 4 billion IP addresses
- A more complex problem
  - How many different users visit each web page?
  - Users x Web pages combinations



How many distinct users are accessing the website?

Flajolet Martin Algorithm

- Pick hash function that is bigger than set to be hashed

- To count IP addresses: hash > 4 billion
- To count URLs: use 64 bits

Flajolet Martin Algorithm: basic property

- Tail length for hash function: number of 0's at the end of the hash for a given hash function
- Hash each element in stream
- Let R be the maximum tail length
- 2^R is approximately the number of distinct elements seen

11110100 has tail length 2

Exercise

Tail length for hash function: number of 0's at the end of the hash for a given hash function

11110100 has tail length 2

Hash each element in stream

Let R be the maximum tail length

2^R is approximately the number of distinct elements seen

- Suppose we want to count the number of userids that visit a Web page
  - Suppose userid is 0..15
- Mid square hash
  - Cube userid, make 12 bits, take middle 6 bits
  - Hint: powers of 2: 0 1 2 4 8 16 32 64 128 256 512 1024 2048 4096
- Suppose the userid sequence is 10 10 7 10 6 14 14 12 6 5 7

Exercise

Tail length for hash function: number of 0’s at the end of the hash for a given hash function

11110100 has tail length 2

Let r be the tail length

What is the probability that r is the tail length?

Flajolet Martin - working

Why does Flajolet Martin Algorithm work

- p ($h(a)$ ends in at least r 0's) = 2^{-r}
  - Suppose the hash is $h_1h_2\ldots h_n$
  - Probability any bit is 0 is ½
  - Probability h_n is o is ½ = 2^{-1}
  - Probability last two bits $h_{n-1}h_n$ are both 0 is (1/2) x (1/2) = 2^{-2}
  - Similarly, probability last r bits are all 0 is 2^{-r}

p (*tail length is* r) = 2^{-r}

If there are m elements in the stream, what is the probability that none of them have tail length r?

Generalization of the algorithm

Suppose there are m distinct elements in the stream

p(no element has tail length r) = $(1 - 2^{-r})^m$

Suppose the elements (e.g., userids) are $u_1, u_2, \ldots, u_m$

- $p\ (u_1$ *has tail length* $r) = 2^{-r}$
- $p\ (u_1$ *doesn't have tail length* $r) = 1-2^{-r}$
- Similarly, $p\ (u_2$ *doesn't have tail length* $r) = 1-2^{-r}$
- $p\ (u_1$ *and* u_2 *don't have tail length* $r) = (1-2^{-r})\ (1-2^{-r})$
- $p\ (u_1 \ldots u_m$ *all don't have tail length* $r) = (1-2^{-r})^m$

Recall

$p\ (h(a)$ ends in at least r 0's) = 2^{-r}

- Suppose the hash is $h_1 h_2 \ldots h_n$
- Probability any bit is 0 is ½
- Probability h_n is o is ½ = 2^{-1}
- Probability last two bits $h_{n-1}h_n$ are both 0 is (1/2) x (1/2) = 2^{-2}
- Similarly, probability last r bits are all 0 is 2^{-r}

Exercise

$p\ (h(a)$ ends in at least r 0's) $= 2^{-r}$

Probability any bit is 0 is 1/2

Suppose there are m distinct elements in the stream

p(no element has tail length r) $= (1 - 2^{-r})^m$

p(no element has tail length r) $= (1 - 2^{-r})^m \sim e^{-mx}$ where $x=2^{-r}$ (See textbook)

p(at least one element has tail length r) $= 1 - e^{-mx}$

PES UNIVERSITY ONLINE

p(at least one element has tail length r) = $1 - e^{-mx}$

$mx = m2^{-r} = m/2^{r}$

What will p be if

$m >> 2^{r}$

$m \sim 2^{r}$

$m << 2^{r}$

Flajolet Martin – why the algorithm works

PES UNIVERSITY ONLINE

p(at least one element has tail length r) = $1 - e^{-mx}$

$mx = m2^{-r} = m/2^r$

There are 3 cases: $m{>}{>}2^r$, $m{<}{<}2^r$, $m{\sim}2^r$

$m{>}{>}2^r$

$mx = m2^{-r} = m/2^r$ will be very large
Therefore, $e^{-mx} \sim 0$
Therefore, p(at least one element has tail length r) = $1 - e^{-mx} \sim 1$
So we are likely to find tail lengths r where $m{>}{>}2^r$

$m{\sim}2^r$

$mx = m2^{-r} = m/2^r$ will a small number
Therefore, e^{-mx} ~ will be some fraction
Therefore, p(at least one element has tail length r) = $1 - e^{-mx}$ ~ some fraction
So there is some probabilty to find tail lengths r where $m{\sim}2^r$

Flajolet Martin – why the algorithm works

p(at least one element has tail length r) = $1 - e^{-mx}$

$mx = m2^{-r} = m/2^r$

There are 3 cases: $m>>2^r$, $m \sim 2^r$, $m<<2^r$

Suppose, $m>>2^r$

So we are likely to find tail lengths r where $m>>2^r$

Suppose, $m \sim 2^r$

So there is some probabilty to find tail lengths r where $m \sim 2^r$

Suppose, $m<<2^r$

$mx = m2^{-r} = m/2^r$ will be very small

Therefore, $e^{-mx} \sim 1$ since $e^0 = 1$

Therefore, p(at least one element has tail length r) = $1 - e^{-mx} \sim 0$

So we are <u>not</u> likely to find tail lengths r where $m<<2^r$

Flajolet Martin – summary

p(at least one element has tail length r) = $1 - e^{-mx}$

$mx = m2^{-r} = m/2^r$

There are 3 cases: $m>>2^r$, $m\sim2^r$, $m<<2^r$

Suppose, $m>>2^r$

We are likely to find tail lengths r where $m>>2^r$

Suppose, $m\sim2^r$

There is some probabilty to find tail lengths r where $m\sim2^r$

Suppose, $m<<2^r$

We are <u>not</u> likely to find tail lengths r where $m<<2^r$

- We are likely to find tail lengths r where $m>>2^r$ or $m\sim2^r$
- If we take the largest r, we must have $m\sim2^r$

Flajolet Martin – practical considerations

Flajolet Martin – in practise

Simple approach

- If we have only one hash function, *m* will always be a power of 2
- We can pick *k* hash functions, estimate $m=2^R$ for each, take average or median
- Will also give better estimate

Problems with simple approach

- Average will be pulled towards max (maybe outlier)
- Median: estimate will always be power of 2

Flajolet Martin – in practise

Combined approach

- Divide k hashes into groups
  - E.g., if we have 6 hash functions $(h_1 \ldots h_6)$, we might have 3 groups where $g_1=(h_1\ h_2)$ and so on
- Compute average of each group
  - E.g.,estimate m_1 as average calculated from $g_1=(h_1\ h_2)$ and so on
- Then median of averages

THANK YOU

K V Subramaniam, Usha Devi
Dept. of Computer Science and Engineering

subramaniamkv@pes.edu
ushadevibg@pes.edu