Data Streams:

Bloom Filters

Mining Massive Datasets

Prof. Carlos Castillo — https://chato.cl/teach

Universitat Pompeu Fabra Barcelona

Sources

- Mining of Massive Datasets (2014) by Leskovec et al. (chapter 4)
  - Slides part 1, part 2
- Tutorial: Mining Massive Data Streams (2019) by Michael Hahsler

Bloom filters

Filtering a data stream

- Suppose we have a large set S of keys
- We want to filter a stream <key, data> to let pass only the elements for which key $\in$ S
- Example: key is an e-mail address, we have a total of $|S|=10^9$ allowed e-mail addresses

What's the Naïve solution?

Filtering a data stream

- Suppose we have a large set S of keys
- We want to filter a stream `<key, data>` to let pass only the elements for which `key` ∈ S
- Example: key is an e-mail address, we have a total of $|S|=10^9$ allowed e-mail addresses
- Naïve solution? Hash table won't work, too big!

Bloom Filter (1-bit case)

- Given a set of keys S
- Create a bit array $B[\,]$ of n bits
  - Initialize to all 0s
- Pick a hash function h with range $[0,n)$
  - For each member of $s \in S$
    - Hash to one of n buckets
    - Set that bit to 1, i.e., $B[h(s)] \leftarrow 1$
- For each element a of the stream
  - Output a if and only if B[h(a)] == 1

Bloom filter creation

Stream processing

Bloom Filter is an approximate filter

- Can it output an element with a key not in S?

- Can it not output an element with a key in S?

Bloom Filter is an approximate filter

- Can it output an element with a key not in S?

  Yes, due to hash collisions $h(x) = h(y)$ when $x \neq y$

- Can it not output an element with a key in S?

  No, because $h(x)$ is always the same for x

Bloom filters are *permissive* (not *strict*)

Bloom filter

- A bloom filter is:
  - An array of n bits, initialized as 0
  - A collection of hash functions h_1, h_2, …, h_k
  - A set S of m key values
- The purpose of the bloom filter is to allow all stream items whose key is in S

Bloom filter initialization

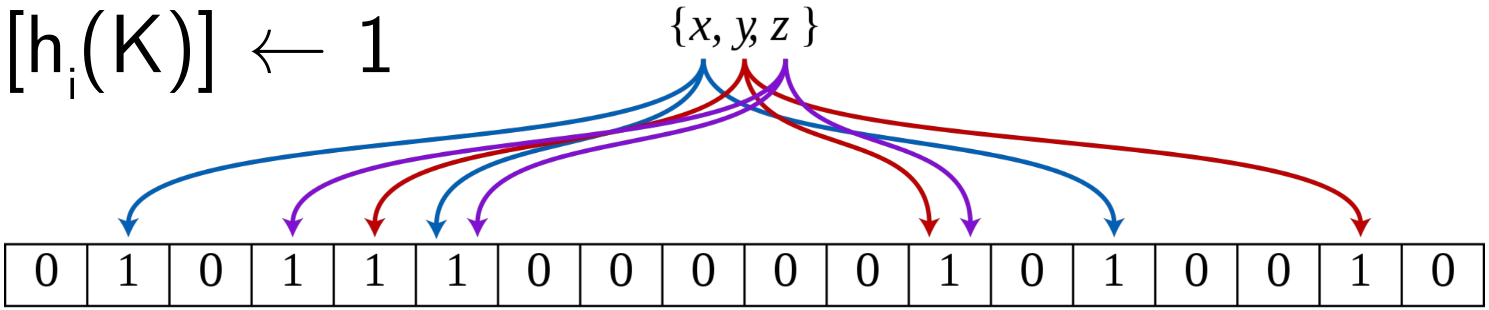
For all positions i in [0, n-1]

$B[i] \leftarrow 0$

For all keys K in S:

For every hash function $h_1, h_2, \ldots, h_k$

$B[h_i(K)] \leftarrow 1$



Bloom filter usage

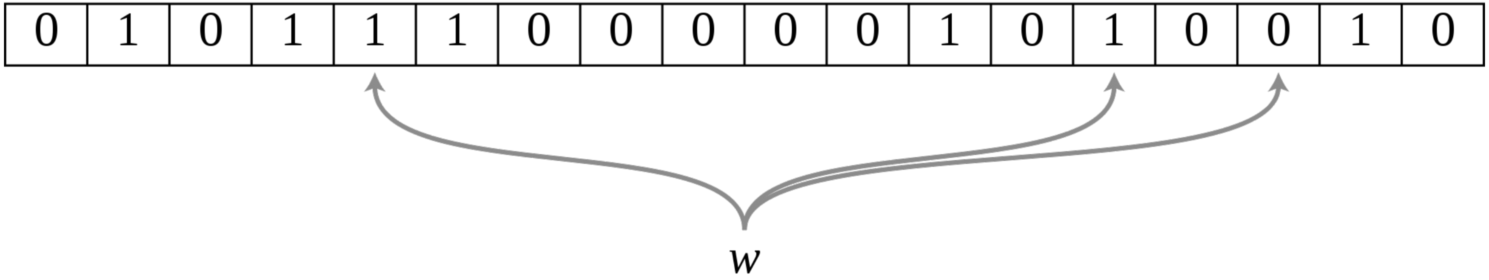
For each input element <key, data>

allow ← TRUE

For every hash function $h_1, h_2, \ldots, h_k$

allow ← allow $\wedge$ $B[h_i(K)] == 1$

output element if allow $==$ TRUE



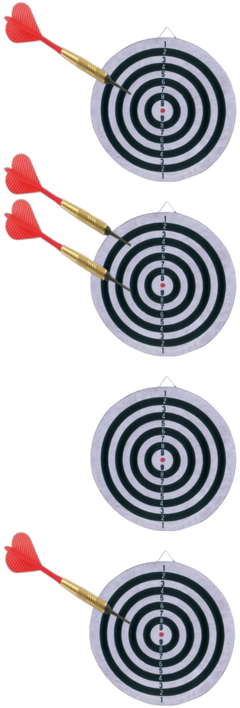
Characteristics of Bloom Filters

- Are lax (not strict) and let some items pass
  - May require a second-level check to make filter strict, for instance store output on disk files and then check against hash tables (slower)
- Implementations can be very fast

  E.g., use hardware words for the bit table

Preliminaries for the analysis: targets and darts

- Suppose we throw y darts at x targets
  - All darts will hit one of the targets

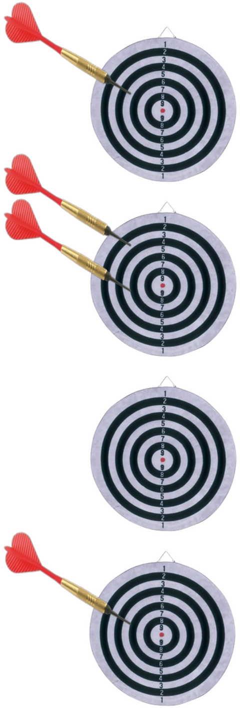


y=4 darts

x=4 targets

Preliminaries for the analysis: targets and darts (cont.)

- How many distinct targets can we expect to hit at least once?
  - Prob. that a given dart will hit a specific target is $1/x$
  - Prob. that a given dart will **not** hit a specific target is $1\text{-}1/x$
  - Prob. none of the y darts will hit a specific target is $(1\text{-}1/x)^{y} = (1\text{-}1/x)^{x(y/x)}$
  - Using that $(1\text{-}\varepsilon)^{1/\varepsilon} \simeq 1/e$ for small ε
  - If x is large, $1/x$ is small, and prob. that none of the y darts will hit a specific target is $(1/e)^{y/x}$



y=4 darts **x=4 targets**

Analysis of the 1-bit Bloom Filter

- Each element of the signature S is a dart $|S|=y$
- Each bit in the array is a target $n=x$

- Suppose $y=|S|=10^9$ (1 G) and $x=n=8 \times 10^9$ (8 G)
- Prob. that a given bit is **not** set to 1 (dart does not hit the target) is $(1/e)^{y/x} = (1/e)^{1/8}$
- Prob. that a given bit is set to 1 is $1 - (1/e)^{1/8} = 0.1175$
- Expected number of bits that is set to 1 = 11.75% x 8GB

  About 12% of bits are set to one in this Bloom Filter

  this is also the false-hit probability in this case

General case

- $|S| = m$ keys, array has n bits
- k hash functions
- Targets $x = n$, darts $y = km$
- Probability that a bit remains 0 is $(1/e)^{km/n} = e^{-km/n}$
- False positive rate with k bits: $(1 - e^{-km/n})^{k}$
  - This is the probability that all of the k bits are set to 1
- Example: we can pick $k = n/m$ to obtain collision probability $1/e = 37\%$

Analysis of a 2-bit Bloom Filter

- Suppose $|S|=10^9$ (1 G) and $n=8 \times 10^9$ (8 GB)
- Suppose we use two hash functions
- Prob. that a given bit is NOT set to 1 (dart does not hit the target) is $(1/e)^{y/x} = (1/e)^{1/4}$
- Prob. a bit is set to 1 is $1 – (1/e)^{1/4}$
- Prob. two bits are set to 1 is $(1 – (1/e)^{1/4})^2 = 0.0493$
- We have a false hit probability of about 5% with two hash functions, while the probability was about 12% with only one

How many hash functions to use?

Too few: test is too unspecific. Too many: table becomes too crowded.

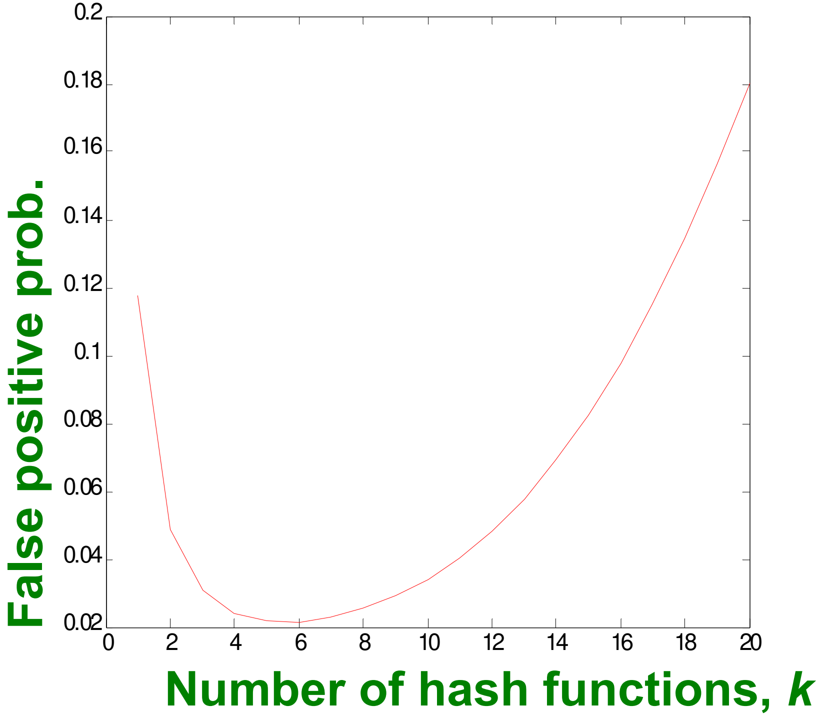
- **m = 1 billion, n = 8 billion**

  False positive rate with k bits: $(1 - e^{-km/n})^k$

  k = 1: $(1 - e^{-1/8})^1 = (1 - e^{-1/8}) =$ **0.1175**

  k = 2: $(1 - e^{-2/8})^2 = (1 - e^{-1/4})^2 =$ **0.0493**

- **What happens as we keep increasing k?**
  - "Optimal" value of k: $n/m \ln(2)$
  - In our case: Optimal **k = 8 ln(2) = 5.54 ≈ 6**
  - **Error at k = 6**: $(1 - e^{-6/8})^6 =$ **0.0216**



Summary

Things to remember

- How to initialize a Bloom filter
- How to use a Bloom filter
- Proofs for 1-bit, 2-bit case

Exercises for TT22-T26

- Mining of Massive Datasets (2014) by Leskovec et al.
  - Exercises 4.2.5
  - Exercises 4.3.4
  - Exercises 4.4.5
  - Exercises 4.5.6