Data Sketching

The Bloom Filter

(membership with controlled-error)

Dictionary problem

What data structures you know for storing a **set of keys** and supporting **exact searches** and **insert operations** over them?

Hashing

What about false positives?

Reminder

Wherever a list or set is used, and space is a consideration, a Bloom Filter should be considered.

When using a Bloom Filter, consider the effects of false positives.

Bloom [1970]

Membership query

Definition

The *Membership Problem*

- ◇ Given a set S and an element y: $y \overset{?}{\in} S$
- ◇ Given a set S compute its characteristic function χ_S

$$\chi_S(y) = \begin{cases} 1, & \text{if} \quad y \in S \\ 0, & \text{if} \quad y \notin S \end{cases}$$

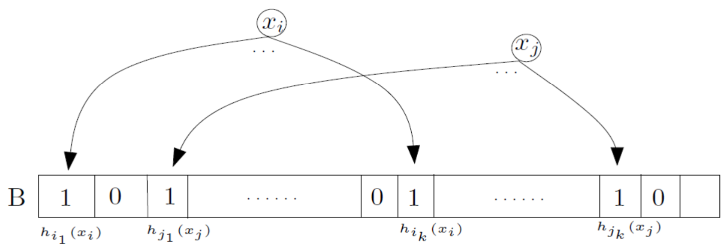
- well-known solutions
  - Linear Scan
  - Hash Functions

Preconditions

- given a set of objects $S = \{x_1, \ldots, x_n\}$
  - no restrictions on objects
- a vector B of m bits were $b_i \in \{0, 1\}$
  - we will discuss next about the value of m
- suppose we have k *hash functions* $h_1, \ldots, h_k$
  - each h_i is defined as $h_i : U \supseteq S \to [1; m]$

Building $B[1, m]$

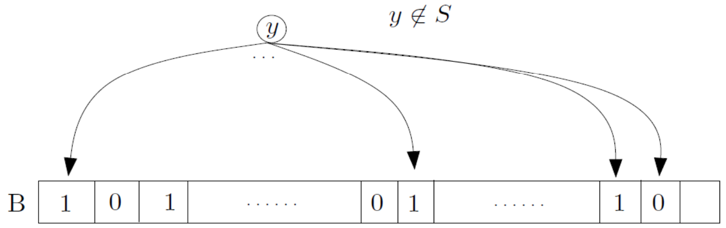
- For each $x \in S$, we set $B[h_j(x)] = 1, \forall j = 1, 2, \ldots, k$.



The *build*-time is $\Theta(k|S|)$ time and $\Theta(|B|) = \Theta(m)$ space

Searching in $B[1, m]$

- We claim "$y \in S$" if $b_{h_i(y)} = 1, \forall i = 1, \ldots, k$.

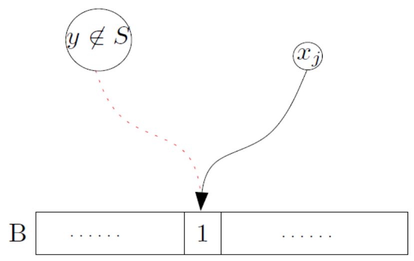


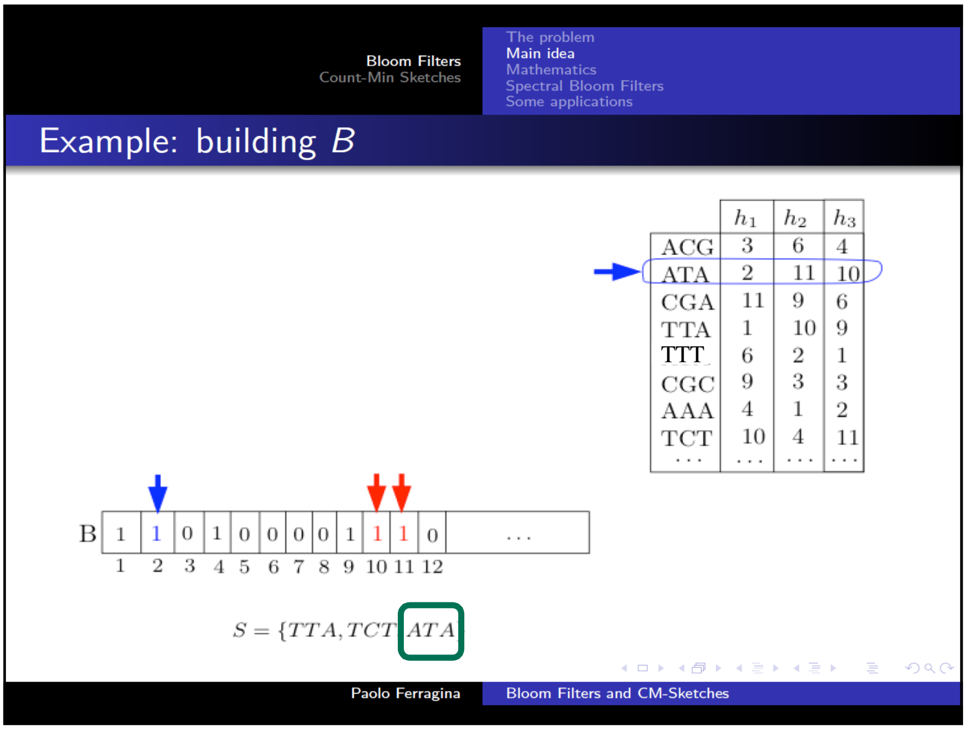
The *search*-time is $O(k)$ time.

The main problem

Definition

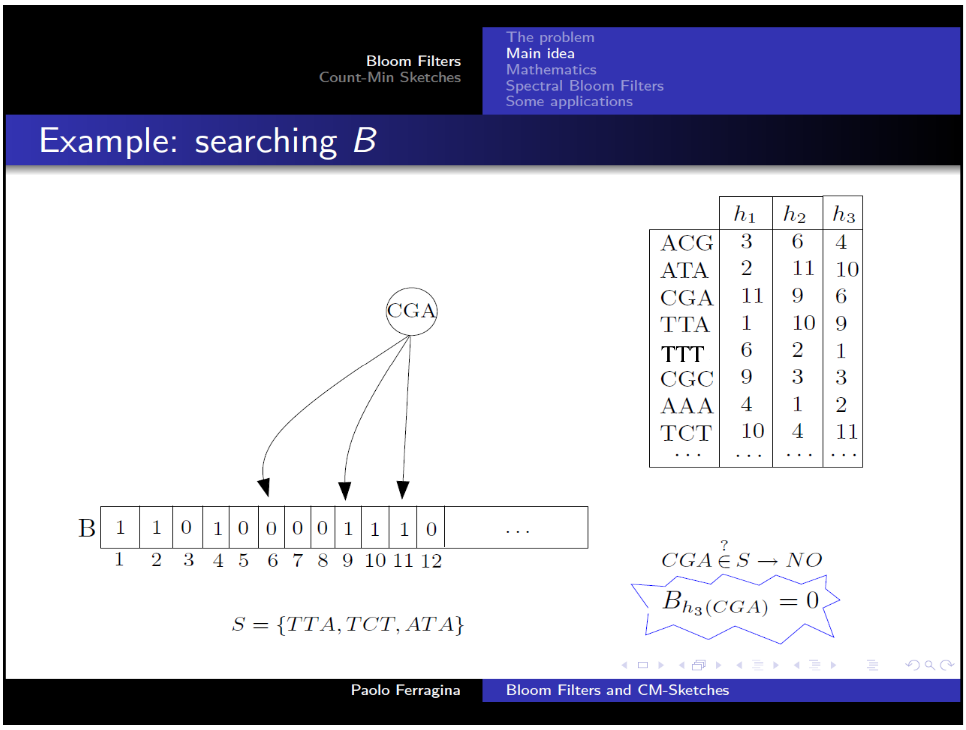
The main problem are *false positives*:
it may exist $x_j \neq y$ such that $h_i(x_j) = h_i(y), \forall i = 1, \ldots, k$.




Bloom Filters
Count-Min Sketches
The problem
Main idea
Mathematics
Spectral Bloom Filters
Some applications
Example: building B

	h_1	h_2	h_3
ACG	3	6	4
ATA	2	11	10
CGA	11	9	6
TTA	1	10	9
TTT	6	2	1
CGC	9	3	3
AAA	4	1	2
TCT	10	4	11
...	...	...	...

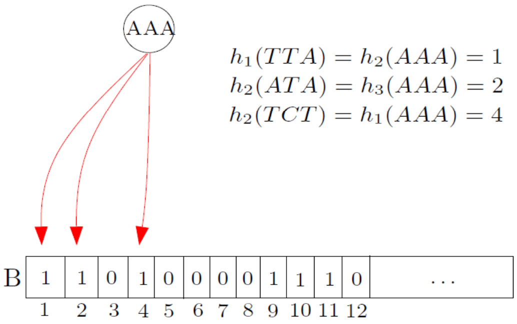
B 1 1 0 1 0 0 0 0 1 1 1 0 ...
1 2 3 4 5 6 7 8 9 10 11 12
$S = \{TTA, TCT, ATA\}$
Paolo Ferragina
Bloom Filters and CM-Sketches


Bloom Filters
Count-Min Sketches
The problem
Main idea
Mathematics
Spectral Bloom Filters
Some applications
Example: searching B

	h_1	h_2	h_3
ACG	3	6	4
ATA	2	11	10
CGA	11	9	6
TTA	1	10	9
TTT	6	2	1
CGC	9	3	3
AAA	4	1	2
TCT	10	4	11
...	...	...	...

CGA
B 1 1 0 1 0 0 0 0 1 1 1 0 ...
1 2 3 4 5 6 7 8 9 10 11 12
$S = \{TTA, TCT, ATA\}$
$CGA \stackrel{?}{\in} S \rightarrow NO$
$B_{h_3(CGA)} = 0$
Paolo Ferragina
Bloom Filters and CM-Sketches

Example: searching B



$h_1(TTA) = h_2(AAA) = 1$
$h_2(ATA) = h_3(AAA) = 2$
$h_2(TCT) = h_1(AAA) = 4$

	h_1	h_2	h_3
ACG	3	6	4
ATA	2	11	10
CGA	11	9	6
TTA	1	10	9
TTT	6	2	1
CGC	9	3	3
AAA	4	1	2
TCT	10	4	11
...	...	...	...

$AAA \overset{?}{\in} S \rightarrow YES$

false positive

$S = \{TTA, TCT, ATA\}$

Index

Probability of a *false positive*

- assumption that hash are perfectly random
- after *build*

$$\mathcal{P}(b_i = 0) = \left(1 - \frac{1}{m}\right)^{kn} \approx e^{-kn/m} = p$$

- probability of a *false positive* is

$$(1 - e^{-kn/m})^k = (1 - p)^k = \varepsilon$$

Not perfectly true but...

- other formulations are *asymptotically* equivalent

Optimizing number of *hash functions*

- higher k-value: more chances to find a 0-bit for $y \notin S$.
- lower k-value: increase fraction of 0-bits in B.
- minimize the ε function

$$\tilde{k} = \ln 2 \cdot (m/n)$$

- With this value of $\tilde{k}$, we have $p = 0.5$ and thus

$$\varepsilon = (0.5)^{\tilde{k}} = (0.6185)^{m/n}$$

How big should be the B vector?

- depends on the ε value we want: given n we fix m.

m	ε
n	0.61
$2n$	0.38
$5n$	0.09
$10n$	0.008

$m = \Theta(n)$ is generally a good choice

Bloom Filters v.s. hash functions

$\diamond$	hash functions	Bloom Filters
build time	$\Theta(n)$	$\Theta(n)$
space needed	$\Theta(n \log n)$	$\Theta(m)$
search time	$O(1)$	$(m/n) \ln 2$
ε value		$(0.6185)^{m/n}$

- Hash functions are Bloom Filters with $k = 1$

Bloom Filters tricks

- union by *OR*
  1. we have sets S_1, S_2 and Bloom Filters B^1, B^2
  2. suppose $m_1 = m_2$ and same hashing functions
  3. just OR the counters

$$B_i^{12} = B_i^1 \vee B_i^2$$

- halved size
  1. suppose $m = 2^\alpha$
  2. make union by *OR* of the two halves
  3. when hashing, mask high-order bit

Spectral Bloom Filters (SBF)

Definition

$M = \langle S, f_x \rangle$ is a multiset were

- S is a set
- f_x is a function returning the #occurrences of x in M

Notice that a *stream* might be looked at as a *multiset*.

<u>*ex*</u> Given $\{A, A, B, C, C\}$
We have $S = \{A, B, C\}$ and $f_A = f_C = 2$, $f_B = 1$

SBF

- B vector is replaced by a vector of counters $C_1, C_2, \ldots, C_m$
  - C_i is the sum of f_x values for elements $x \in S$ mapping to i
- Approximations of f_x are stored into

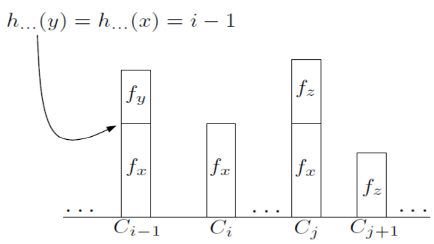
$$C_{h_1(x)}, C_{h_2(x)}, \ldots, C_{h_k(x)}$$

- Due to conflicts, the C_i provide approximations...

Upper bounds

Bloom Filters
Count-Min Sketches
The problem
Main idea
Mathematics
Compressed Bloom Filters
Spectral Bloom Filters
Some applications

The *Minimum Selection*



- C_{i-1} is not a good approximation of f_x (neither of f_y)
- C_i is an exact approximation of f_x
- C_{j+1} is an exact approximation of f_z

Insertion and Deletion

- insertion is simple
  - increase each counter by 1

```
...
  for each h in H do
    C[h(x)] = C[h(x)] + 1;
  done
...
```

- deletion is simple
  - decrease each counter by 1
- search for an element x
  - return the *Minimum Selection* (MS) value $m_x = \min\{C_{h_1(x)}, C_{h_2(x)}, \dots, C_{h_k(x)}\}$

Upper bounds

Recurring minimum for improving the estimate + 2 SBF

On the error of SBF

- The error is the same as for Bloom Filters

Theorem

For all x, it is $f_x \leqslant m_x$. Furthermore $f_x \neq m_x$ with probability

$$E_{SBF} = \varepsilon \approx (1 - p)^k$$

Proof.

The case $m_x < f_x$ cannot happen.

The event $m_x > f_x$ is "*all counters $C_{h_i(x)}$ have a collision*", that corresponds to a "*false positive*" event of classical BF. □

Index

Pattern Matching

A set of objects whose keys are complex and time-costly to be compared (e.g. URLs, matrices, MP3,...).

- Use BF to reduce the number of explicit comparisons.
- Effective in hierarchical memories.
- Example on Dictionary matching [Bloom '70].

Set Intersection

We have two machines M_A and M_B each storing a set of items A and B, respectively. We wish to compute $A \cap B$ exchanging a *small* number of bits.
Typical applications: *data replication check*, *distributed search engines*.

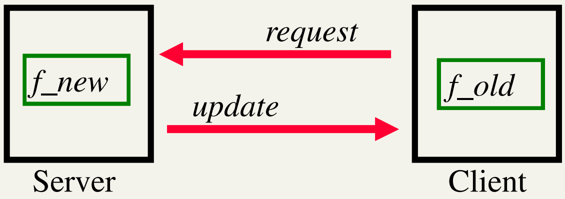
- M_A sends $BF(A)$ to B, using $m = m_{opt} = k|A|/ln2$ bits.
- M_B checks B into $BF(A)$ in $O(k|B|)$ time, and sends back *explicitly* the found items, say Q. Note that $Q \supseteq A \cap B$.
- M_A computes $Q \cap A$, and returns it.

The bit-cost is $\frac{k|A|}{\ln 2} + (|A \cap B| + |B|0.5^k) \log |U| \ll |A| \log |U|$.
Good for long keys,

Web Algorithmics

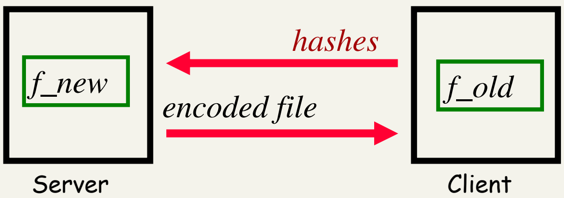
File Synchronization

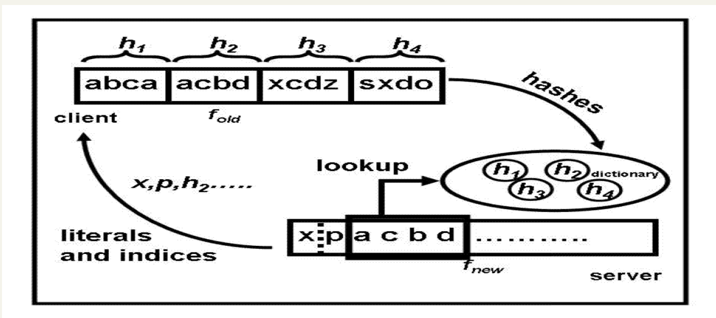
File synch: The problem



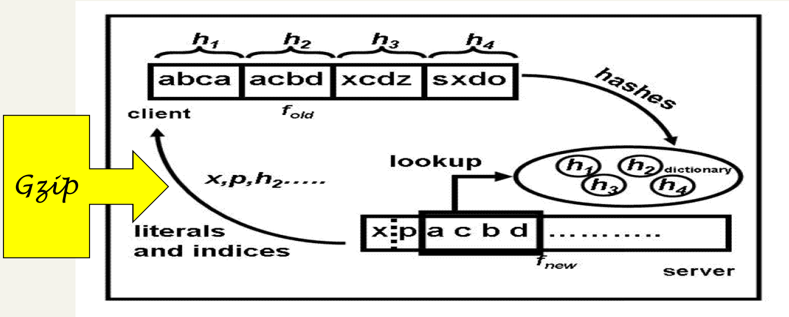
- **client** wants to update an out-dated file
- server has new file but does not know the old file
- update without sending entire *f_new* *(using similarity)*
- *rsync*: file synch tool, distributed with Linux

The rsync algorithm





The rsync algorithm (contd)



- simple, widely used, **single** roundtrip
- optimizations: 4-byte rolling hash + 2-byte MD5, *gzip* for literals
- choice of block size problematic (*default*: max{700, √n} bytes)
- not good in theory: granularity of changes may disrupt use of blocks