Bloom Filters

- Given a set **S** = {$x_1, x_2, x_3, \ldots, x_n$} on a universe **U**, want to answer queries of the form:

  Is ***y*** $\in$ ***S*** *?*

- Bloom filter provides an answer in
  - "Constant" time (to hash).
  - Very small amount of space.
  - But with small probability of a false positive
    - Particularly useful when the answer is usually NO
    - When care a lot about space.

Bloom Filters

*Start with an **m** bit array, filled with 0s.*

B																
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

*Insertion: Hash each item **x_j** in **S k** times. If **$h_i(x_j) = a$**, set **$B[a] = 1$**.*

B																
	0	1	0	0	1	0	1	0	0	1	1	1	0	1	1	0

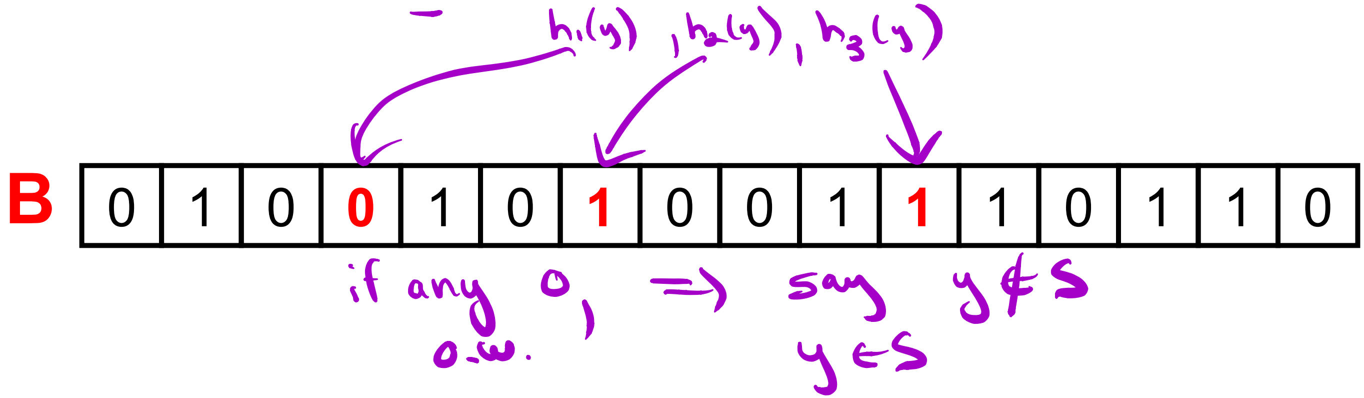
***n** items*

***m** = **cn** bits*

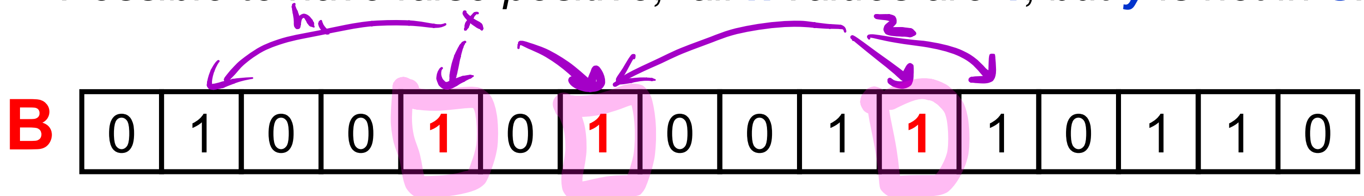
***k** hash functions*

Bloom Filters

Lookup: To check if ***y*** *is in* ***S****, check* ***B*** *at* ***h_i(y)****. All* ***k*** *values must be* ***1****.*



Possible to have false positive; all ***k*** *values are* ***1****, but* ***y*** *is not in* ***S****.*



n *items* ***m = cn*** *bits* ***k*** *hash functions*

Purposes of analysis: size of table $= c \cdot$ # elts in set S storing

Estimate False Positive Probability

n items ***m = cn*** bits ***k*** hash functions

Assume hash functions completely random

is $y \in S$?

Inserted elements $S = \{x_1, x_2, \ldots, x_n\}$ into table.

Lookup(y) $y \notin S$

$Pr(\text{false positive}) = Pr(h_1(y) = h_2(y) \cdots = h_k(y) = 1)$

false positive is when we say that an elt is in the set when in fact, it was never inserted.

$Pr(\text{specific bit in table is } 0) = \left(\frac{m-1}{m}\right)^{kn} = \left(1 - \frac{1}{m}\right)^{kn} \approx e^{-\frac{kn}{m}}$

n elts → k darts

kn random darts

$1 - \frac{1}{m} \approx e^{-\frac{1}{m}}$

$1 - x \approx e^{-x}$

x very close to 0.

Let β denote the fraction of bits that are set to 0. $\approx Pr(\text{specific bit is } 0)$

Expected # of bits that are $0 = m \cdot Pr(\text{spec bit is bit})$

$Pr(\text{false positive}) = (1-\beta)^k \approx \left(1 - e^{-\frac{kn}{m}}\right)^k$

find k that mimizes this fn.

Estimate False Positive Probability

- **Pr**(specific bit of filter is 0) is

  $$p' \equiv (1-1/m)^{kn} \approx e^{-kn/m} \equiv p \qquad (p' \leq p)$$

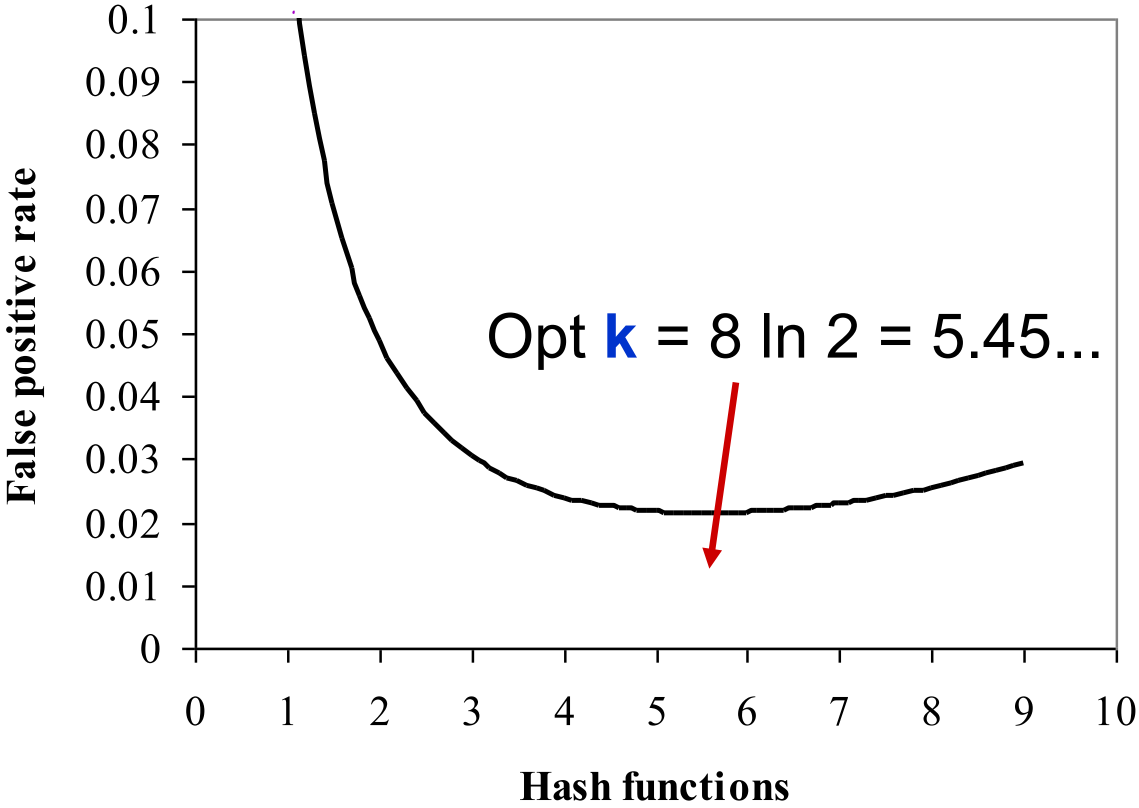
- If β is fraction of **0** bits in the filter then false positive probability for a new element is

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

- Find optimal at $k = (\ln 2)\, m/n$ by calculus.
  - So optimal false positive prob is about $(0.6185)^{m/n}$

n items ***m = cn*** bits ***k*** hash functions

Graph of $(1-e^{-k/c})^k$ for c=8



m/n = 8

n items ***m = cn*** bits ***k*** hash functions

Applications

- Original (40 years ago) use:
  - Spellcheckers (false positive = misspelled word)
  - Forbidden passwords (false positive = no biggie)

Applications – More modern

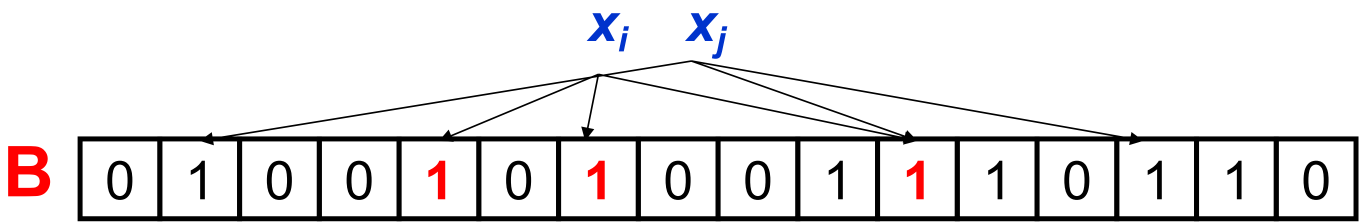
- In databases:
  - Join: combine two tables with a common domain into a single table
  - Semi-join: a join in distributed databases in which only the joining attribute from one site is transmitted to other and used for selection. The selected records sent back
  - Bloom-join: a semi-join where we send only a Bloom filter of the joining attribute

Applications- modern

- Monitor all traffic going through a router, checking for signatures of bad behavior (e.g. strings associated to worms, viruses)
- Must be fast and simple - operate at hardware/line speed.
- Use a Bloom filter for signatures.
- On positive: send off to analyzer for action
  - False positive = extra work on slow path.

Handling Deletions

- Bloom filters can handle insertions, but not deletions.



- If deleting x_i means resetting 1's to 0's, then deleting x_i will "delete" x_j.

Bloom filter numerous variations and applications

- See papers on website.

- ``The Bloom Filter principle: wherever a list or set is used, and space is at a premium, consider using a Bloom filter if the effect of false positives can be mitigated.''