Hashing: Bloom filters & Cuckoo hashing

Bloom filter:

Setting: HUGE universe $U = \{0, 1, \ldots, N-1\}$ of possible elements

Want to maintain a subset $S \subset U$ where $|S| = m$

Using 0-1 table/array $H[0, 1, \ldots, n-1]$ where $n = |H|$

$n = cm$ for $c \geq 1$.

Example: U = possible password strings
S = unacceptable passwords

— Want fast queries, small space, simple, allow false positives with small probability.

k = # of hash functions

Hash functions: $h_1, h_2, \ldots, h_k : U \rightarrow \{0, 1, \ldots, n-1\}$

Operations:

- Insert x into S
- Query: is $x \in S$?

No Deletions

Bloom filter:

- Initialize H to all 0's.

To insert x into S:

for all $i = 1 \rightarrow k$:

- compute $h_i(x)$
- Set $H[h_i(x)] = 1$ (keep as is if already set to 1)

For a query: is $x \in S$?

for all $i = 1 \to k$:

- compute $h_i(x)$
- check whether $H[h_i(x)] = 1$?

If for all i it is set to 1,

then return(YES)

else return(NO).

Note, if $x \in S$, then we always output YES

but if $x \notin S$, we might have a

false positive = incorrectly output YES.

What is the false positive rate

as a function of k & c?

$$c = \frac{n}{m} = \frac{|H|}{|S|} = \frac{|\text{hash table}|}{|\text{subset to maintain}|}$$

Note, big k: more robust/redundancy, i.e.,

check more bits

but add more 1's when insert.

So what's optimal k?

What is the probability of a false positive?

First, what's the prob. an entry $H[i]$ is 0 or 1?

$$\Pr(H[i]=0) = \Pr(\forall y \in S, \forall 1 \le j \le k, h_j(y) \ne i)$$

$$= \left(1-\frac{1}{n}\right)^{km} \qquad |H|=n, \ |S|=m$$

$$= \left(1-\frac{1}{cm}\right)^{km} \qquad n=cm, \ c \ge 1$$

$$\le e^{-k/c}$$

in fact, $\left(1-\frac{1}{cm}\right)^{km} \approx e^{-k/c}$ for m large

So we'll use this approximation.

<u>False Positive:</u>

$$\Pr(\text{output } x \in S \mid x \notin S) = \Pr(\forall j, H[h_j(x)]=1)$$

$$\approx \left(1-e^{-k/c}\right)^k$$

Let $f := \Pr(\text{false positive}) = \left(1-e^{-k/c}\right)^k$

What's the optimal choice of k as a function of c?

Let's minimize f as a function of k.

Let $g = \ln f = k \ln\left(1 - e^{-k/c}\right)$

$$\frac{\partial g}{\partial k} = \ln\left(1 - e^{-k/c}\right) + \frac{k}{1 - e^{-k/c}} \times \frac{1}{c} \times e^{-k/c}$$

Set $k = c \ln 2$

then $\frac{\partial g}{\partial k} = -\ln 2 + \ln 2$ & can check this is a minimum by looking at the 2nd derivative.

Plugging in $k = c\ln 2$,

$$\Pr(\text{false positive}) = f = \left(1 - e^{-k/c}\right)^k = \left(\left(\frac{1}{2}\right)^{\ln 2}\right)^c \approx (.6185)^c$$

Note, $\Pr(H[i] = 0) \approx e^{-k/c} = 1/2$

So H is a random 0-1 string

Examples — false positive rate

$k=1$: $c=10$:	.09516
$c=100$:	.00995
$k=5$: $c=10$:	.0094
$k=10$: $c=100$:	6×10^{-11}
$k=c\ln 2$, $c=10$:	.0082
$c=100$:	1.3×10^{-21}

$$f=\left(1-e^{-k/c}\right)^k$$

for $\frac{k}{c}$ small, $f\approx\left(\frac{k}{c}\right)^k$

Cuckoo hashing:

As before, HUGE universe U
but static S: do a set of insertions to setup S
& then we want fast queries.

Goal: $O(1)$ query time (as with Bloom filter)
but no errors
& $O(1)$ expected insertion time
(instead of worst-case, as for Bloom filter)

Use 2 hash functions $h_1, h_2 : U \to \{0, 1, \ldots, n-1\}$
store ≤ 1 item at each location $H[i]$.

To insert: use $h_1(x)$ or $h_2(x)$, whichever is empty.
If neither is empty, then push one of the occupied elements to its other choice & repeat if necessary.
Potential problem: cycle of pushes
in which case: start over with
2 new hash functions h_1, h_2.

To insert x into S:

- compute $h_1(x)$
- if $H[h_1(x)]$ is empty

  then add x at $H[h_1(x)]$

  else:
  - compute $h_2(x)$
  - if $H[h_2(x)]$ is empty

    then add x at $H[h_2(x)]$

    else (so $h_1(x)$ & $h_2(x)$ are occupied)
    - let $y = H[h_2(x)]$
    - set $H[h_2(x)] = x$

      & move y to its other possible location

      & repeat for y.

Query: is x in S?

Check $H[h_1(x)]$ & $H[h_2(x)]$

Cuckoo graph:

Directed graph representing H.

Vertex for each entry of H, so n vertices.

Edges show possible locations for items.

if $H[i] = x$ & $h_1(x) = i$

then edge $i \to h_2(x)$

if $H[i] = x$ & $h_2(x) = i$

then edge $i \to h_1(x)$

Insertion succeeds if no cycle.

If there's a cycle we do a rehash (choose 2 new hash functions)

Recall, $|S| = m$ & $|H| = n = cm$

we'll choose so that $n > 6m$, i.e., $c > 6$.

First, we'll show that the expected insertion time is $O(1)$.

<u>Claim 1:</u> For $\ell \geq 1$, for positions i & j, Prob. of a shortest path from $i \rightsquigarrow j$ of length $= \ell$ is $\leq \dfrac{3^{-\ell}}{n}$

Using the claim, say x & y collide if there's a path $x \rightsquigarrow y$ or $y \rightsquigarrow x$.

In other words, a path from $\left\{ \begin{matrix} h_1(x) \\ \text{or} \\ h_2(x) \end{matrix} \right\}$ to $\left\{ \begin{matrix} h_1(y) \\ \text{or} \\ h_2(y) \end{matrix} \right\}$

or from $\left\{ \begin{matrix} h_1(y) \\ \text{or} \\ h_2(y) \end{matrix} \right\}$ to $\left\{ \begin{matrix} h_1(x) \\ \text{or} \\ h_2(x) \end{matrix} \right\}$.

By the claim, the prob. x & y collide is

$$\leq 4 \sum_{l=1}^{\infty} \frac{3^{-l}}{n} = 4 \times \frac{1}{2} \times \frac{1}{n} = \frac{2}{n}$$

Hence, # of expected collisions with x is $O(1)$

So when adding x into S there's $O(1)$ other elements that are moved in expectation.

Proof of claim: induct on l.

Base case: $l=1$ so edge ~~$\leftrightarrow$~~ $i \to j$ or $j \to i$

Fix i & j. Prob. $x \in S$ has $h_1(x)=i$ & $h_2(x)=j$ (or reverse) is $2/n^2$

Summing over $x \in S$,

Prob. of edge $i \to j$ or $j \to i$ is

$$\leq m \times \frac{2}{n^2} \leq \frac{1}{3n} \text{ for } n > 6m.$$

In general, for $l > 1$:

Want shortest path of length l so length $l-1$ path.

Consider penultimate position k on shortest path:

thus there is a path of length $l-1$ from $i \rightsquigarrow k$

& edge $k \to j$

the prob. is $\leq \frac{3^{-(l-1)}}{n} \times \frac{1}{3n} = \frac{1}{3^l n^2}$

Summing over the n choices of k we have:

$$\leq \frac{1}{3^l n}. \qquad \blacksquare$$

Rehashing:

To get a rehash we need a cycle.

We'll show that with Prob. $\geq \frac{1}{2}$ no cycles exist.

By the claim,

$$\text{Prob. of a cycle involving position } i \text{ of length} = l \leq \frac{3^{-l}}{n}$$

thus, Prob. of some cycle involving ~~...~~ Position i

$$\leq \frac{1}{n} \sum_{l=1}^{\infty} 3^{-l} = \frac{1}{2n}$$

therefore prob. of some cycle is

$$\leq n \times \frac{1}{2n} = 1/2$$

So prob. $\leq \frac{1}{2}$ of a rehash

& prob. $\leq \left(\frac{1}{2}\right)^k$ of k rehashes

So expect 1 rehash & each takes $O(n)$ time.