Bloom filter:

huge universe $U = \{0, 1, \ldots, N-1\}$ of size $N = |U|$.

0-1 table $H[0, 1, \ldots, n-1]$ of size $n = |H|$.

Want to maintain a subset $S \subset U$.

$m = |S|$ size of S after all of the insertions.

Example: U = possible password strings
S = unacceptable passwords

Want fast queries, small space,
allow false positives with small probability

k = # of hash functions.

Hash functions $h_1, h_2, \ldots, h_k$ where each $h_i : U \to \{0, \ldots, n-$

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

Initialize H to all 0's.

To insert x into S:

for all $i = 1 \to k$:

- Compute $h_i(x)$
- Set $H[h_i(x)] = 1$

To query whether or not $x \in S$:

for all $i = 1 \to k$:

- Compute $h_i(x)$
- Check whether $H[h_i(x)] = 1$?

If for all i it is set $= 1$, then return(YES)

else return (NO)

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

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

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

false positive = incorrectly output YES

What is the probability of a false positive?

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

let $c = \frac{n}{m} > 1$

First let's compute the prob. an entry i is set to 0:

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

$$= \left(1 - \frac{1}{n}\right)^{km}$$

$$\le e^{-km/n} = e^{-k/c}$$

because $e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} \pm \cdots$

for small x, e^{-x} is a close approx. to $1-x$, which is the case for $\frac{1}{n}$ for large n.

for $x \notin S$,

$$\Pr(\text{false positive})$$
$$= \Pr(\forall_j \; H[h_j(x)] = 1)$$
$$\stackrel{?}{=} \left(1 - e^{-k/c}\right)^k$$

This is not correct.

Technically, we should do:

$$\Pr(\forall_j \; H[h_j(x)] = 1)$$
$$= \Pr(H[h_1(x)] = 1) \times \Pr(H[h_2(x)] = 1 \mid H[h_1(x)] = 1)$$
$$\times \cdots \times \Pr(H[h_k(x)] = 1 \mid H[h_1(x)] = 1, \ldots, H[h_{k-1}(x)] = 1)$$

$= 1 - e^{-k/c}$ but what about later terms?

but we do have that:

$$\Pr(H[h_2(x)] = 1 \mid H[h_1(x)] = 1) \leq \Pr(H[h_2(x)] = 1)$$
$$\leq 1 - e^{-k/c}$$

Hence,

$$\Pr(\text{false positive}) \leq \underbrace{\left(1 - e^{-k/c}\right)^k}$$

And we'll use this as our estimate for the false positive rate.

Next question: What's the best choice for k?

k big $\Rightarrow$ putting too many 1's in H

k small $\Rightarrow$ checking too few bits of H

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

& let's minimize f as a function of k.

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

So $f = e^g$

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

Set $k = c \ln 2$

then $\frac{dg}{dk} = -\ln 2 + \ln 2 = 0$ & can check this is a minimum.

Plugging in $k = c\ln 2$, the false positive estimate is

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

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

So H is a random 0-1 string (except the bits are not independent as discussed earlier)

Examples:

		false positive rate
$k = 1$:	$c = 10$:	.09516
	$c = 100$:	.00995
$k = c\ln 2$:	$c = 10$:	.0082
	$c = 100$:	1.3×10^{-21}

Alternative scheme: Cuckoo hashing

Universe U

static S: we do a set of insertions and setup S

Then we want to be able to do fast queries on S.

Goal: $O(1)$ query time (with no errors)

& in expectation $O(1)$ insertion time.

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 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 let $y = H[h_2(x)]$
  Set $H[h_2(x)] = x$
  & move y to its other possible location
  and repeat

To query: is x in S?

Just check $H[h_1(x)]$ & $H[h_2(x)]$.

For inserting x,

this may misplace y_1

moving y_1 may misplace y_2

etc.

Either this process halts by hitting an empty spot or we get a cycle

So if it doesn't halt after n cycles then we choose 2 new hash functions h_1 & h_2 & we reinsert all of S.

Pseudocode for insert:

Notation:

Pos = position currently trying to insert at

$x \leftrightarrow y$ = swap x & y

Insert(x):

$pos = h_1(x)$

Do $\leq n$ times:

- if $H[pos] = \emptyset$
  - then $H[pos] = x$; exit()
- $x \Leftrightarrow H[pos]$ (so swap x with item at $H[pos]$)
- if $pos = h_1(x)$ then $pos = h_2(x)$
  - else $pos = h_1(x)$

If repeated n times then Rehash().

Rehash: starts with empty H
& reinserts every x into S.

How often do we rehash?
What's the running time of an insert?

Cuckoo graph:

Directed graph representing H.

Vertex for each entry of H, so n vertices.

Directed edges show other possible locations for items.

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

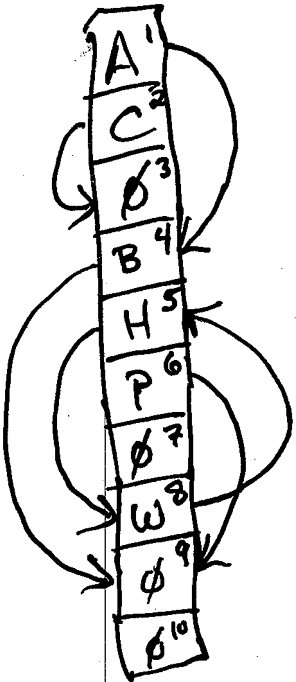
then edge $i \rightarrow h_2(x)$

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

then edge $i \rightarrow h_1(x)$.

Example:

Example:



– Suppose we Insert(Z) & $h_1(Z) = 8$, $h_2(Z) = 1$

1. Place Z at pos. 8 & add edge $8 \to 1$ (drop edge $8 \to 5$)
2. Move W to pos. 5 & add edge $5 \to 8$ (already there)
3. Move H to pos. 8 & add edge $8 \to 5$ (drop $8 \to 1$)
4. Move Z to pos. 1 & add edge $1 \to 8$ (drop $1 \to 4$)
5. Move A to pos. 4 & add $4 \to 1$ (drop $4 \to 9$)
6. Move B to pos. 9 & add $9 \to 4$

– Suppose we Insert(D) with $h_1(D) = 5$, $h_2(D) = 8$

Then we get a cycle $D \to H \to W$:

1. Place D at pos. 5
2. Move H to pos. 8
3. Move W to pos. 5
4. Move D to pos. 8

etc.

An insertion succeeds if no cycle

If there's a cycle we do a rehash.

First let's ignore rehash's & we'll show the expected insertion time is $O(1)$.

Then we'll show that the expected # of rehashes is $O(1)$. Since they take $O(n)$ time per rehash, this is amortized $O(1)$ time per insert.

$|S| = m$, $|H| = n$, and we'll choose so that

$$n > 6m.$$

Lemma: For $l \geq 1$, for positions i & j,

Probability of a shortest path from $i \rightsquigarrow j$ of length $= l$ $\leq \frac{3^{-l}}{n}$

Hence from the lemma:

Say x & y collide if there's a path $x \rightsquigarrow y$
or $y \rightsquigarrow x$.

In other words, a path from $\{h_1(x)$ or $h_2(x)\}$
to $\{h_1(y)$ or $h_2(y)\}$
or from $\{h_1(y)$ or $h_2(y)\}$
to $\{h_1(x)$ or $h_2(x)\}$

By the lemma,
the Prob. that x & y collide is

$$\leq 4 \sum_{l=1}^{\infty} \frac{3^{-l}}{n} = \frac{2}{n} \quad \text{since } \sum_{l=1}^{\infty} 3^{-l} = 1/2$$

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 lemma: Induct on ℓ.

Base case: $\ell = 1$:

Fix i & j. Prob. $x \in S$ has $h_1(x) = i, h_2(x) = j$ (or vice versa)

$$= \frac{2}{n^2}$$

Summing over x,

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

$$\le m \times \frac{2}{n^2} \le \frac{1}{3n} \checkmark$$

For $\ell > 1$:

Looking for shortest path of length ℓ so no length $\ell - 1$ path.

Take penultimate position k on the shortest path.

Then (a) there is a path (shortest path) of length $\ell - 1$ from $i \rightsquigarrow k$

& (b) there's an edge $k \to j$.

Prob. of (a) by induction is $\le 3^{-(\ell-1)}/n$

& (b) by the base case analysis $= \frac{1}{3n}$

Hence, it's $\le \frac{1}{3^{\ell-1} n} \times \frac{1}{3n} = \frac{1}{3^\ell n^2}$

Summing over the n choices of k we have:

$$\le 1/3^\ell n. \checkmark$$

Rehashing:

To get a rehash we need a cycle.
We'll show that with prob. $\geq \frac{1}{2}$ no cycles exist.

By lemma,

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

$$\text{Prob. of some cycle involving } i \text{ is} \leq \frac{1}{n}\sum_{l=1}^{\infty} 3^{-l} = \frac{1}{2n}$$

$$\text{Prob. of some cycle is} \leq n \times \frac{1}{2n} = \frac{1}{2}.$$

Hence, Prob. $\leq \frac{1}{2}$ of a rehash.
and Prob. $\leq \left(\frac{1}{2}\right)^k$ of k rehashes

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