Hashing: Bloom filters & Cuckoo hashing

Bloom filter:

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

Want to maintain a subset \( S \subseteq U \)

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.
\( k = \# \text{ of hash functions} \)

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

\underline{Operations:}
- Insert \( x \) into \( S \)
- Query: is \( x \in S \)?

No Deletions

\underline{Bloom filter:}
- Initialize \( H \) to all 0's

\underline{To insert} \( x \) into \( S \):
  \[ \text{for all } i = 1 \rightarrow k: \]
  - 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 \( H[i,j] \) is 0 or 1?

\[
Pr(H[i,j] = 0) = Pr(\forall y \in S, \forall j \leq k, h_j(y) \neq i) = (1 - \frac{1}{n})^{km}
\]

\[
= (1 - \frac{1}{cm})^{kn} = (1 - \frac{1}{cm})^{kn} = (1 - \frac{1}{cm})^{kn} = \leq e^{-k/c}
\]

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

So we'll use this approximation.

False Positive:

\[
Pr(output \times \in S | \times \in S) = Pr(\forall j, H[j,x] = 1) \propto (1 - e^{-k/c})^k
\]

Let \( f = Pr(false\ positive) = (1 - e^{-k/c})^k \).
What's the optimal choice of k as a function of c?

Let's minimize f as a function of k.

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

\[
\frac{\partial g}{\partial k} = \ln (1 - e^{-k/c}) + \frac{k}{1 - e^{-k/c}} \cdot \frac{1}{c} \cdot e^{-k/c}
\]

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 \( k = c \ln 2 \),

\[
\Pr(\text{false positive}) = f = (1 - e^{-k/c})^k = \left(\frac{1}{2}\right)^k \cdot 6.185^f
\]

Note, \( \Pr(H|\overline{H} = 0) = e^{-k/c} = \frac{1}{2} \)

So \( H \) is a random 0-1 string.
Examples

\[ f = \left(1 - e^{-\frac{k}{c}}\right)^k \]

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

- \( k=1 \): \( c=10 \):
  - \( c=100 \):
    - \( 0.09516 \)
    - \( 0.00995 \)

- \( k=5 \): \( c=10 \):
  - \( 0.0049 \)

- \( k=10 \): \( c=100 \):
  - \( 6 \times 10^{-11} \)

- \( k=\ln 2 \): \( c=10 \):
  - \( c=100 \):
    - \( 0.0082 \)
    - \( 1.3 \times 10^{-21} \)
Cuckoo hashing:

As before, HUGE universe $U$

but **static** $S$: Do a set of insertions to set up $S$

& 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 $h, h_2: U \rightarrow \{0, \ldots, n-1\}$

Store $\leq 1$ item at each location $H[i,j]$.

To insert: use $h(x)$ 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 $h, h_2$. 
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 $i \rightarrow h_2(x)$
- if $H[i] = x \& h_2(x) = i$
  - then edge $i \rightarrow h_1(x)$

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 $O(1)$.

Claim 1: For $l \geq 1$ for positions $i \& j$ 
Prob. of a shortest path from $i \to j$ 
of length $l$ is $\leq \frac{3^{-l}}{n}$

Using the claim, say $x \& y$ collide 
if there's a path $x \to y$ or $y \to x$.
In other words, a path from $\{h_1(x)\}$ to $\{h_1(y)\}$ 
or from $\{h_1(x)\}$ to $\{h_2(x)\}$ 
or from $\{h_1(y)\}$ to $\{h_2(y)\}$.
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: induction on l.

**Base case:** \( l=1 \) so edge \( i \rightarrow j \) or \( j \rightarrow i \)

Fix i & j. Prob. \( x \in S \) has \( h_1(x) = i \) & \( h_2(x) = j \) (or reverse)

is \( \frac{2}{n^2} \)

Summing over \( x \in S \)

Prob. of edge \( i \rightarrow j \) or \( j \rightarrow i \) is

\[ \leq \frac{m \times 2}{n^3} \leq \frac{1}{3n} \quad \text{for } n>6m. \]
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 \( i \to k \)

& edge \( k \to j \)

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

Summing over the \( n \) choices of \( k \) we have:

\[ \leq \frac{1}{3^n} \]
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, \[
\text{Prob. of some cycle involving Position } i \leq \frac{1}{n} \sum_{l=1}^{\infty} 3^{-l} = \frac{1}{2n}
\]

therefore prob. of some cycle is

\[
\leq n \times \frac{1}{2n} = \frac{1}{2}
\]

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.