Today: hashing but first a toy problem

Balls into bins:

n balls & n bins (or buckets)

Each ball is thrown into a random bin
(independent of what happened for
the other balls)

Let load of bin i = # of balls assigned
to bin i.

How large is the max load?

= Max # of balls in a bin.

Max load is $O(\log n)$ with high probability

$\geq 1 - 1/\text{poly}(n)$

Can refine to $\Theta(\frac{\log n}{\log \log n})$
Better idea:

Assigns balls sequentially into bins

For $i=1 \rightarrow n$

- Choose 2 random bins say $j \& k$
- Let $L(j)$ & $L(k)$ denote their current load
- If $L(j) < L(k)$, assign ball $i$ to bin $j$
  else assign ball $i$ to bin $k$
  (So ball $i$ goes to the smaller of the 2 bins)

Then max load is $O(\log \log n)$ with high prob.

If choose 2 bins then get $O(\frac{\log \log n}{\log d})$
  (instead of 2)
Hashing:

Running example:

Database of unacceptable passwords

Queries check if a password is allowed or not. Need to check quickly

HUGE set $U = \text{universe of possible passwords}$

HashTable $H$ size $n$

$H = [0,1,\ldots, n-1]$

Map elements of $U$ into bins $[0,1,\ldots, n-1]$

Use hash function $h: U \rightarrow [0,1,\ldots, n-1]$

Chain hashing, $H[i]$ is a linked list of elements stored there.

Start with empty list at each $H[i]$.

Let $S$ be the set of unacceptable passwords. We store $S$ in $H$. 
To add $x \in U$ into $S$
- compute $h(x)$
- add $x$ onto linked list at $H[h(x)]$

To check: is yes?
- compute $h(y)$
- check linked list at $H[h(y)]$
  \[ \text{to see if it contains } y. \]

query time = load at bin $h(y)$.

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

We're putting $m$ balls into $n$ bins.

If $h$ is a random function so $h(x)$ is random over $\{0, 1, \ldots, n-1\}$

Max query time is max load.

When $m = n$ max load is $O(\log n)$.

To get max load $O(1)$ need $n = \Omega(m^2)$. 
Better approach:

Use 2 hash functions

\[ h : U \rightarrow \{0, 1, \ldots, n-1\} \]
\[ h_2 : U \rightarrow \{0, 1, \ldots, n-1\} \]

To add \( x \in U \) into \( S \):
- compute \( h_1(x) \) & \( h_2(x) \)
- add \( x \) to least loaded of \( h_1(x) \) & \( h_2(x) \)

To check: is \( y \) yes?
- compute \( h_1(y) \) & \( h_2(y) \)
- check list at \( H[\bar{h}_1(y)] \) & \( H[\bar{h}_2(y)] \)
  (Requires checking 2 lists but both are much smaller)

Now, for random \( h_1 \) & \( h_2 \),
if \( m = n \), max load is only \( O(\log \log n) \).
Bloom filters:

- Faster queries = $O(1)$ time
- Less space
- Simpler
- But: false positives with small probability

Maintain set $S \subseteq U$.

Operations:

- $\text{Insert}(x)$: add $x$ into $S$
- $\text{Query}(x)$: is $x \in S$?
  - if $x \in S$, we always output YES
  - if $x \notin S$, we usually output NO
  - but we have a false positive (output YES) with small probability
  - false positive rate.
H is a 0-1 array of size n. (No linked list!)
Start with H as all 0s.
hash function \( h: U \to \{0, 1, \ldots, n\} \)

To add \( x \) into \( S \), set \( H[h(x)] = 1 \).

To check if \( x \) is in \( S \):
if \( H[h(y)] = 1 \) then output YES
if \( H[h(y)] = 0 \) then output NO.

Problem:
if \( y \in S \) but we added \( z \) into \( S \)
where \( h(y) = h(z) \)

then we get a false positive for the query: is \( y \in S \)?
Better approach:

\[ k \text{ hash functions: } h_1, h_2, \ldots, h_k \]

for each \( i \), \( h_i : V \rightarrow \{0, 1, \ldots, n-1\} \)

Start: Set \( H[i] = 0 \) for all \( i \in \{0, 1, \ldots, n-1\} \)

So \( H \) is the all \( 0 \)-vector.

To add \( x \) into \( S \):

for \( i = 1 \rightarrow k \), set \( H[h_i(x)] = 1 \)

(sor set \( k \) bits to 1)

To check if \( \text{YES?} \)

if for all \( i \), \( H[h_i(y)] = 1 \)

then output \( \text{YES} \)

else (if 1 or more = 0)

output \( \text{NO} \).

if \( \text{YES} \), then for query: is \( \text{YES?} \)

we always output \( \text{YES} \)

if \( \text{YES} \), then we might output \( \text{YES} \) if

for all \( i \), there is \( z \in S \) & \( j \)

where \( h_i(y) = h_j(z) \).
What is the probability of a false positive?

Recall, hash table $H$ of size $|H| = n$
& storing database $S$ of size $|S| = m$
where $|H| = n > |S| = m$

Want to make $n$ as small as possible
let $c = \frac{n}{m}$ so $c > 1$.
want $c$ small.

For $y \notin S$ to get a false positive need that
the $k$ bits at $h_1(y), \ldots, h_k(y)$ are set to 1.

For a specific bit $b$ where $b \in \{0, \ldots, n-1\}$
what's the probability $H[b] = 1$?

$\Pr(H[b] = 1) = 1 - \Pr(H[b] = 0)$

We are throwing $mk$ balls randomly into $n$ bins,
$\Pr(H[b] = 0) = \Pr(\text{all } mk \text{ balls miss bin } b)$

$= (1 - \frac{1}{n})^{mk} = e^{-mk/n} = e^{-k/c}$ since $c = \frac{n}{m}$

Recall $e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \cdots \geq 1 - x$
& for small $x$, $e^{-x} \approx 1 - x$,

$\Pr(H[b] = 1) = 1 - e^{-k/c}$
For $y \in S$,
\[
\Pr(\text{false positive for } y) = \Pr(\text{for all } i, \ H[h_i(y)] = 1) \sim (1 - e^{-k/c})^k = \text{false positive rate}
\]

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 $f = (1 - e^{-k/c})^k$ = false positive rate.

Let's minimize $f$ as a function of $k$.

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

So $f = e^g$ & we'll minimize $g$ to make it easier.

\[
\frac{dg}{dk} = \ln (1 - e^{-k/c}) + \frac{k}{1 - e^{-k/c}} \times \frac{1}{c} \times e^{-k/c}
\]

Set $k = \frac{c}{c} \ln 2$.

Then $\frac{dg}{dk} = -\ln 2 + \ln 2 = 0$ & can check this is a minimum.
Plugging in $k = c \ln 2$, the false positive rate is

\[ f = (1 - e^{-k/c})^c = \left(1 - \frac{1}{2}\right)^{c \ln 2} = \left(\frac{1}{2}\right)^c = 0.6185^c \]

and

\[ \Pr(\mathcal{H}[\mathcal{E}] = 0) = e^{-k/c} = \frac{1}{2} \]

So $\mathcal{H}$ is a random 0-1 string.

### Examples:

<table>
<thead>
<tr>
<th>$k = 1$: $c = 10$:</th>
<th>0.09516</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c = 100$:</td>
<td>0.00995</td>
</tr>
<tr>
<td>$k = c \ln 2$: $c = 10$:</td>
<td>0.0082</td>
</tr>
<tr>
<td>$c = 100$:</td>
<td>$1.3 \times 10^{-21}$</td>
</tr>
</tbody>
</table>