6.1 Occupancy Problem

Bins and Balls  Throw n balls into n bins at random.

1. Pr[Bin 1 is empty] = \((1 - \frac{1}{n})^n \sim \frac{1}{e}\).

2. Pr[Bin 1 has k balls] = \(\binom{n}{k} \frac{1}{n^k} (1 - \frac{1}{n})^{n-k} \leq \frac{1}{e^k k!}\).

Sterling’s Approximations

\[
\binom{n}{k}^k \leq \binom{n}{k} \leq (\frac{ne}{k})^k
\]

Thus, letting \(A_{i,k}\) be the event that bin \(i\) contains at least \(k\) balls, we have

\[
\Pr(A_{i,k}) = \sum_{i=k}^{n} \begin{pmatrix} n \\ i \end{pmatrix} i^i \left(\frac{i}{n}\right)^i \left(1 - \frac{i}{n}\right)^{n-k}
\]

Thus, by the union bound,

\[
\Pr(\text{any bin contains more than } k \text{ balls}) \leq \sum_{i=1}^{n} \Pr(A_{i,k})
\]

In order to approximate this, we need to derive a simple upper bound for \(\Pr(A_{i,k})\). We’ll make use of the following elementary inequality, for any \(i \leq n\):

\[
\binom{n}{i} \leq \binom{n}{i} \leq \binom{ne}{i}
\]

Using this we can easily derive the bound

\[
\Pr(A_{i,k}) \leq \sum_{i=k}^{n} \binom{ne}{i} i^i \left(\frac{1}{n}\right)^i
\]

\[
= \left(\frac{e}{i}\right)^k \left(1 + \frac{e}{k} + \left(\frac{e}{k}\right)^2 + \cdots\right)
\]

\[
= \left(\frac{e}{k}\right)^k \frac{1}{1 - e/k}
\]
Now comes the tedious part. Let $k = \lceil (3 \log n) / \log \log n \rceil$. Then

$$\Pr(A_{i,k}) \leq \left( \frac{e}{k} \right)^k \frac{1}{1 - e/k} \leq 2 \left( \frac{e}{3 \log n / \log \log n} \right)^k \leq 2 \left( e^{1 - \log 3 - \log \log n + \log \log \log n} \right)^k \leq 2 \left( e^{-3 \log n + 3 \log \log n / \log \log n} \right)^k \leq 2 \left( e^{-2 \log n} \right) \leq 2 \frac{n^2}{n^2}$$

for $n$ sufficiently large that $(\log \log n) / \log n < 1/3$.

It follows that

$$\Pr(\text{no bin contains more than } \lceil (3 \log n) / \log \log n \rceil \text{ balls}) = 1 - \sum_{i=1}^{n} \Pr(A_{i,k}) \geq 1 - \frac{2}{n}$$

**Theorem 6.1.1 Max Load**

When $n$ balls are thrown into $n$ bins, the maximum number of balls in any bin is $O\left( \frac{\log n}{\log \log n} \right)$ with high probability, i.e.,

$$E[\text{max load}] = \frac{\ln n}{\ln \ln n} (1 + o(1))$$

$$\text{max load} = \Theta\left( \frac{\ln n}{\ln \ln n} \right) \text{ w.h.p.}$$

It can be shown that this is a tight bound.

**Coupon Collector’s Problem** Suppose I throw $kn$ balls.

$$\Pr(\text{bin 1 is empty}) \sim \left( \frac{1}{e} \right)^k$$

If $k = c \ln n + d$, then

$$\Pr(\text{bin 1 is empty}) \sim \frac{1}{e^{dn^c}}$$
\( \Pr[\exists \text{some bin empty}] \leq \frac{n}{n^c} \leq \frac{1}{n^{c-1}} \)

Therefore, w.h.p. \( O(n \log n) \) balls suffice.

**Claim:**

\[ E[\text{number of balls to see all bins}] = n \cdot H_n \]

Imagine a counter (starting at 0) that tells us how many boxes have at least one ball in it. Let \( X_1 \) denote the number of throws until the counter reaches 1 (so \( X_1 = 1 \)). Let \( X_2 \) denote the number of throws from that point until the counter reaches 2. In general, let \( X_k \) denote the number of throws made from the time the counter hit \( k-1 \) up until the counter reaches \( k \).

So, the total number of throws is \( X_1 + ... + X_n \), and by linearity of expectation, what we are looking for is \( E[X_1] + ... + E[X_n] \).

How to evaluate \( E[X_k] \)? Suppose the counter is currently at \( k-1 \). Each time we throw a ball, the probability it is something new is \( \frac{n-(k-1)}{n} \). So, another way to think about this question is as follows:

Coin flipping: we have a coin that has probability \( p \) of coming up heads (in our case, \( p = \frac{n-(k-1)}{n} \)). What is the expected number of flips until we get a heads?

It turns out that the "intuitively obvious answer", \( 1/p \), is correct. But why? Here is one way to see it: if the first flip is heads, then we are done; if not, then we are back where we started, except we’ve already paid for one flip. So the expected number of flips \( E \) satisfies: \( E = p*1 + (1-p)*(1+E) \). You can then solve for \( E = 1/p \).

Putting this all together, let \( CC(n) \) be the expected number of throws until we have filled all the boxes. We then have:

\[
CC(n) = E[X_1] + ... + E[X_n] \\
= \frac{n}{n} + n/(n-1) + n/(n-2) + ... + n/1 \\
= n(1/n + 1/(n-1) + ... + 1/1) \\
= nH_n
\]

QED.

\[ \Pr[x \geq n \ln n + cn \text{ or } x \leq n \ln n - cn] \sim (e^{-e^{-c}} - e^{-e^c}) \]
6.2 Hashing

FORMAL SETUP

• Keys come from some large universe M. (e.g. all < 50-character strings)
• Some set S in M of keys we actually care about (which may be static or dynamic).
• Do inserts and lookups by having an array N of size \( |N| \), and a HASH FUNCTION \( h : M \rightarrow \{0, ..., |N| - 1\} \). Given element \( x \), store in \( N[h(x)] \).
• Will resolve collisions by having each entry in A be a linked list. Collision is when \( h(x) = h(y) \). There are other methods but this is cleanest – called "separate chaining". To insert, just put at top of list. If h is good, then hopefully lists will be small.

UNIVERSAL HASHING

A hash family \( \mathcal{H} \) is 2-universal if for all \( x \neq y \) in M,
\[
\Pr_{h \in \mathcal{H}}[h(x) = h(y)] \leq \frac{1}{|N|}
\]

Let \( x, y \in M \).

\[
C_{xy} = \begin{cases} 
1 & \text{if } h(x) = h(y) \\
0 & \text{otherwise}
\end{cases}
\]

\[
E[C_{xy}] \leq \frac{1}{|N|}
\]

\[
E[\text{number of elts of } S \text{ that collide with } y] = \sum_{x \neq y} C_{xy} \leq \frac{|S|}{|N|}
\]

= \( E[\text{amount of time when accessing } y] \)

If \( |N| \geq |S| \), then \( E[\text{amount of time when accessing } y] = o(1) \).

One way to construct a 2-universal hash family:

Here, let \( M = \{0, ..., m - 1\} \) and \( N = \{0, ..., n - 1\} \). Pick prime \( p \geq m \) (or, think of just rounding m up to nearest prime). Define

\[
h_{a,b}(x) = ((ax + b) \mod p) \mod n.
\]

\( \mathcal{H} = \{h_{ab}|a,b \in GF(p) \text{ and } a \neq 0\} \)

It is easy to show that \( |\mathcal{H}| = p(p - 1) \).
**Theorem 6.2.1 Lower Bound**

$H$ is a hash family $M \rightarrow N$, then $\exists x \neq y \in M$, s.t. $\Pr[h(x) = h(y)] \geq \frac{1}{|N|} - \frac{1}{|M|}$.

Pf: via Yao’s principle.

**Strongly 2-universal hash family** see Anupam’s notes

**Perfect hash functions** Definition: A hash function that maps each different key to a distinct integer. Usually all possible keys must be known beforehand. A hash table that uses a perfect hash has no collisions.

A family of hash functions $H = \{h : M \rightarrow N\}$ is said to be a perfect hash family if for each set $S \subseteq M$ of size $s \leq n$, there exists a hash function $h \in H$ that is perfect for $S$.

If $|N| = |S|$, every perfect hash family has size $2^{O(|N|)}$.

**2-level hashing** [Fredman Komlos Szemered]

Proposal: hash into table of size $N$. Will get some collisions. Then, for each bin, rehash it, squaring the size of the bin to get zero collisions.

To construct a 2-level hash function:

1. Pick $h \in H$, where $H$ is a 2-universal hash family $M \rightarrow N$ and $|N| = |S|$.
2. If number of collisions $> |N|$, goto step 1
3. If $N_i$ elements hashed to bin $i \leq N$, then pick $h_i : M \rightarrow N_i^2$. If any collisions goto step 3.
4. Do step 3 for all bins.

**Pr[x, y collide] $\leq \frac{1}{|N|}$

$E[\text{number of collisions}] \leq \left( \frac{|S|}{2} \right) \frac{1}{|N|}$

1. In step 1 and 2, since $|N| = |S|$, let $C$ denote number of collisions.

   $E[C] \leq \left( \frac{|S|}{2} \right) \frac{1}{|S|} < \frac{|S|}{2}$

   According to Markov Inequality,

   $\Pr\left[ C > 2 \cdot \frac{|S|}{2} \right] \leq \frac{1}{2}$

2. $C = \sum_i \left( \begin{array}{c} N_i \\ 2 \end{array} \right) \leq |N| = |S|$
3. If \( H_i : M \to N_i^2 \), set \( S \) is of size \( N_i \).

\[
E[C_i] = \leq \left( \frac{N_i}{2} \right) \cdot \frac{1}{N_i^2} \leq \frac{1}{2}
\]

Therefore, according to Markov Inequality,

\[
\Pr[C_i \geq 1] \leq \frac{1}{2}
\]

Now let’s study the space requirement of this scheme.

\[
Space \leq |N| + \sum_i N_i^2 \leq 2|S|
\]

In addition, to store the hash functions, we need to use \( O(|S|) \) more bits.

Unfortunately, this approach works for static dictionary only, but not dynamic dictionaries where we want to support insert/delete operations.