Basic Probability: Some Applications!

CS70: Discrete Mathematics and Probability Theory

UC Berkeley – Summer 2026

Lecture 18

Ref: Note 15

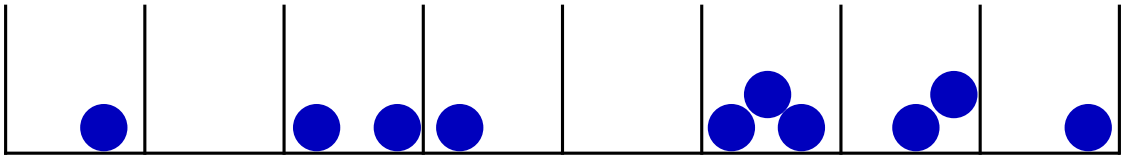
Today

We have covered only basic probability theory – can we do useful things?

Yes!!

- Analyzing hash functions
  Used in: data structures, file identification/integrity, cryptography, blockchain, ...
- Data coverage by random sampling
  Exploring randomly – how long to see everything?
- Load balancing
  Spreading out computational work or resources evenly

Balls and Bins



Experiment: We throw m balls into n bins (uniformly at random)

$\Rightarrow$ In the illustration above, $n = 8$ and $m = 10$

Questions we might ask (for arbitrary n and m):

- What is the probability that some bin contains > 1 ball?
- How many bins do we need so that the probability of > 1 ball in any bin is small?
- How many balls do we need to throw for probability $> 1/2$ of every bin having a ball?
- What is the "worst-case bin" size – number k so that it's unlikely any bin has $> k$ balls.

While stated as "balls and bins" all of these have important CS applications!

Hashing

A hash function $h : D \to C$ maps from a large domain to a finite set of codes

$\Rightarrow$ D is always much larger than C, and can be infinite

$\Rightarrow$ Values in C are called "hash values," "hashes," "codes," or "digests"

Typically: For a fixed set of values $S \subseteq D$, the set $\{h(x) \mid x \in S\}$ "looks uniform"

Question: Is the output *random* for a function h?

No! Deterministic function. Predictable.

Can start with picking h randomly, then it's random (universal hashing)

In many applications we can usefully analyze as if output is random

Not in all applications!

Cryptographic hash functions have more important properties

Cryptography: using hash fn for random oracle can fail spectacularly!

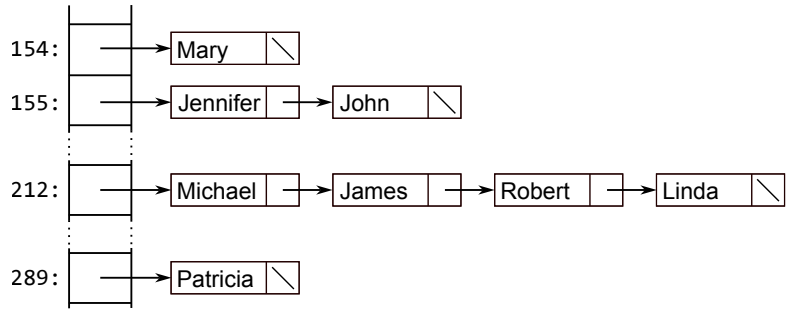
Keep in mind: Cryptography is weird (and subtle)

Hashing Applications: Data Structures

A hash function $h : D \to C$ maps from a large domain to a finite set of codes

Idea: Look up data quickly from a key (e.g., name)

- Unstructured data in array or list: look through everything to find key
- Hash function gives storage location ("bucket"): Go directly to data
  More accurately: Look through all data with same hash value



- Multiple keys with same hash value: collision (buckets 155 and 212)
- Ideally, evenly spread-out data: no "*clumping*"

Question 1: What is the probability that we have a collision?

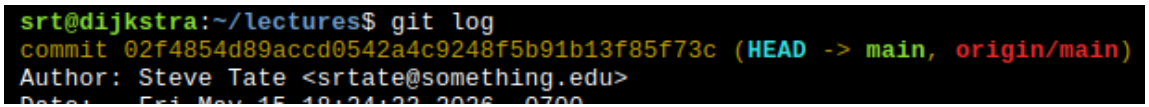
Question 2: What is the probability that the largest "bucket" is small?

Hashing Applications: Unique Identifiers

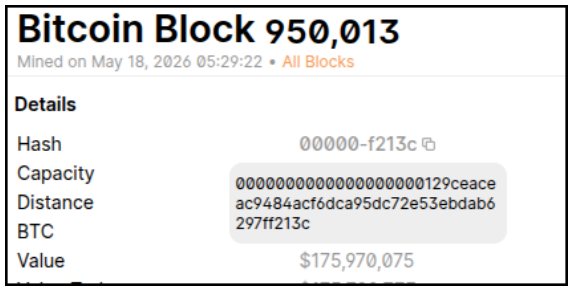
A hash function $h : D \to C$ maps from a large domain to a finite set of codes

We want a *fingerprint* of a large file/piece of data

- Git commit identifiers



- Bitcoin blockchain blocks



- Dropbox uses hashes for *content-addressable storage*
- Software suppliers use hashes for users to verify downloads

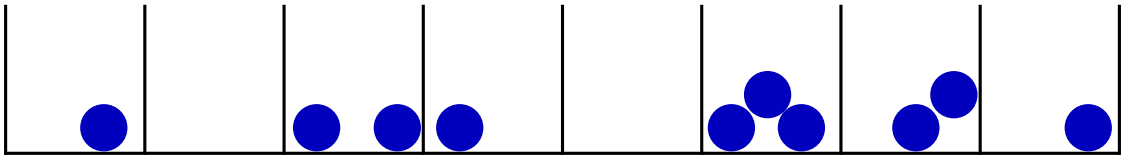
Popular SHA-256 produces a 256-bit hash $\rightarrow$ $|C| = 2^{256}$

Are we making a table of size 2^{256}?

Hint: There are roughly 2^{167} atoms in the earth and everything on it.

We want these to be *unique* – the same output for different inputs is bad!

Back to Balls and Bins...



Hash function $h : D \to C$ — hashing a set $S \subseteq D$

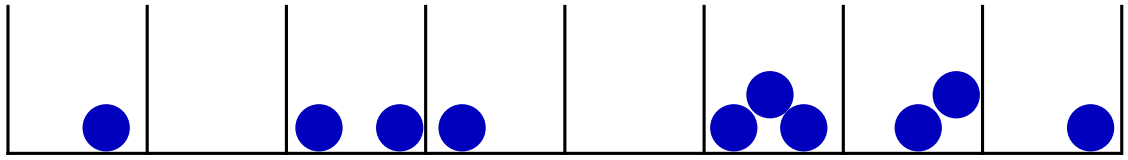
Input element $x \in S$ (ball) "thrown into" a hash value (bin)

$m = |S|$ balls $\quad n = |C|$ bins

Questions:

- Given m and n, what is probability that some bin has > 1 ball (collision)?
- For given n, how large does m have to be for probability $\geq 1/2$ of some bin with multiple balls
- For given m, how large does n have to be for very small probability of multiple balls in some bin?
- For given m and n, what is the probability that the number of balls in some bin is $\geq b$?

Balls in Bins: Probability of Collision



Question 1: What is probability that some bin has > 1 ball (collision)?

Recall the Birthday "Paradox"...

Events: $A =$ there is a collision ; $B_{i,j} =$ balls i and j ($i \neq j$) collide

First observation: $A = \bigcup_{i \neq j} B_{i,j}$

Second observation: $\mathbb{P}[B_{i,j}] = 1/n$

Why?

Ball i falls in some bin ; probability ball j falls in that same bin $= 1/n$

This is really using "Total Probability" if you write it out!

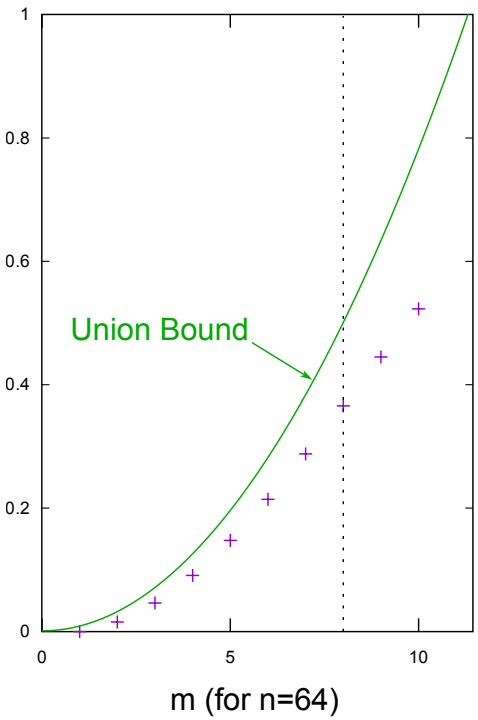
Theorem: $\mathbb{P}[A] \leq \frac{m^2}{2n}$

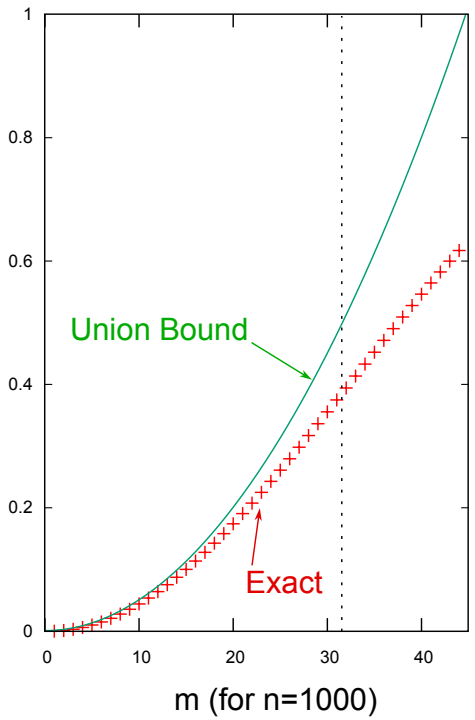
Proof: (using union bound) $\mathbb{P}[A] \leq \sum_{i \neq j} \mathbb{P}[B_{i,j}] = \binom{m}{2} \frac{1}{n} = \frac{m(m-1)}{2} \frac{1}{n} \leq \frac{m^2}{2n}$. □

Balls in Bins: Probability of Collision

Question 1: What is probability that some bin has > 1 ball (collision)?

Theorem: $\mathbb{P}[A] \leq \frac{m^2}{2n}$ ← How tight is this bound?





Dashed line: $m = \sqrt{n}$

OK accuracy for small m and medium n ($m \leq \sqrt{n}$ and $n > 100$)
Not so good for $m > \sqrt{n}$
For very large n and $m \ll \sqrt{n}$ (crypto applications) this is quite good...

Some Actual Errors

Calculated to 5 digits after decimal point.

For $m = \sqrt{n}$:

n	64	1024	1,000,000
m	8	32	1,000
Union bound	0.50000	0.50000	0.50000
Exact	0.36597	0.38702	0.39327
Error	0.13403	0.11298	0.10673

For $m = \frac{1}{2}\sqrt{n}$:

n	64	1024	1,000,000
m	4	16	500
Union bound	0.12500	0.12500	0.12500
Exact	0.09109	0.11111	0.11730
Error	0.03391	0.01389	0.00770

For $m = \frac{1}{4}\sqrt{n}$:

n	64	1024	1,000,000
m	2	8	250
Union bound	0.03125	0.03125	0.03125
Exact	0.01563	0.02704	0.03065
Error	0.01563	0.00421	0.00060

Error for large n and $m \leq \frac{1}{4}\sqrt{n}$ pretty small!

"Crpyto sizes" give tiny error (we'll see this later)

Collisions: A Better Estimate

A_i = no collision when ith ball is thrown into a bin

$A_{i-1} \cap \cdots \cap A_1$ = no collisions in first $i-1$ balls
$\Rightarrow$ so exactly $i-1$ bins "occupied" and $n-(i-1)$ "free"

$\mathbb{P}[A_i | A_{i-1} \cap \cdots \cap A_1] = \frac{n-(i-1)}{n} = (1 - \frac{i-1}{n})$

Product rule: $\mathbb{P}[A_1 \cap \cdots \cap A_m] = \mathbb{P}[A_1] \cdot \mathbb{P}[A_2|A_1] \cdots \mathbb{P}[A_m|A_1 \cap \cdots \cap A_{m-1}]$

$$\Rightarrow \mathbb{P}[\text{no collision}] = 1 \cdot \left(1 - \tfrac{1}{n}\right) \cdots \left(1 - \tfrac{m-1}{n}\right)$$

Sums are easier to work with than products, so let's take a log:

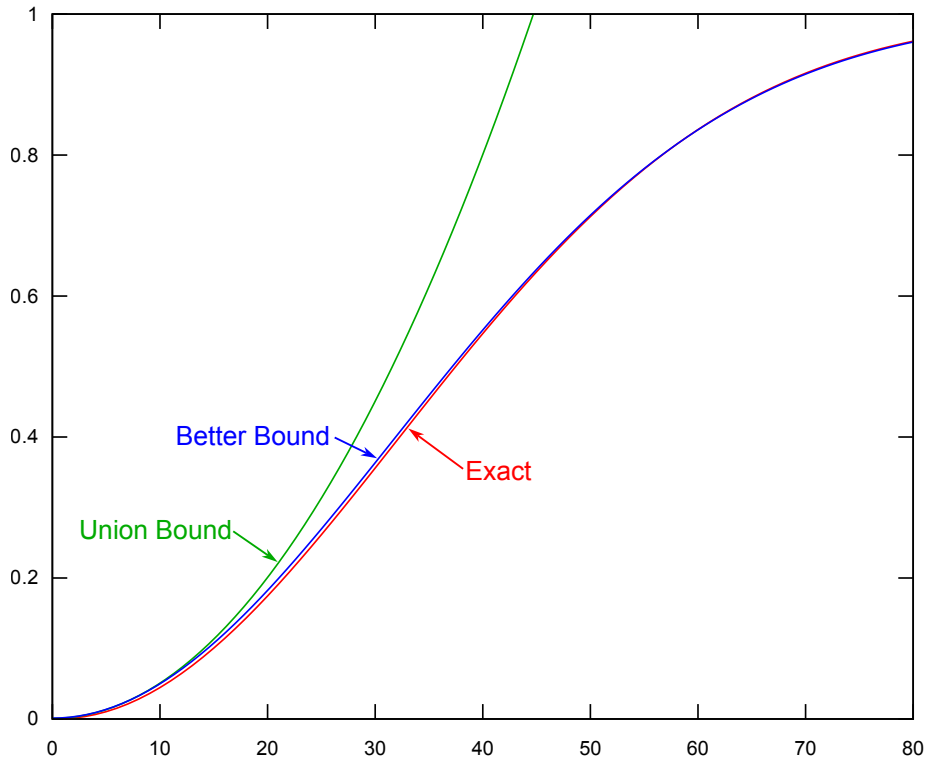
$$\ln(\mathbb{P}[\text{no collision}]) = \sum_{k=1}^{m-1} \ln(1 - \frac{k}{n})$$

Taylor series: $\ln(1-x) = -x - \frac{x^2}{2} - \frac{x^3}{3} - \cdots$, so $\ln(1-x) \approx -x$ for small x

So: $\ln(\mathbb{P}[\text{no collision}]) \approx -\frac{1}{n} \sum_{k=1}^{m-1} k = -\frac{m(m-1)}{2n} \implies \mathbb{P}[\text{collision}] \approx 1 - e^{-\frac{m^2}{2n}}$

Collisions: A Better Estimate

Is this better? Accuracy for $n = 1000$ bins and $m = 1, \ldots, 80$ balls



For $m \leq 15$, both approximations are pretty good
For $m > 30$, union bound is not so great, but "better estimate" is good!

Collisions: A Better Estimate

Some real numbers

For birthday problem, with $m = 22$ and $n = 365$:

Union bound probability: $\frac{m^2}{2n} = \frac{22 \cdot 22}{2 \cdot 365} = \frac{242}{365} \approx 0.663$

Better bound: $1 - e^{-\frac{m^2}{2n}} = e^{-\frac{242}{365}} \approx 0.485$

Exact value (to 4 significant digits): 0.476

$65,536 = 2^{16}$ Git commit ids (160-bit hashes): $m = 2^{16}$ $n = 2^{160}$

$\Rightarrow$ *That's about 18 commits every day for 10 years*

Union bound: $\frac{m^2}{2n} = \frac{\left(2^{16}\right)^2}{2 \cdot 2^{160}} = \frac{2^{32}}{2^{161}} = 2^{-129} \approx 1.469368 \times 10^{-39}$

Better estimate: $1 - e^{-\frac{m^2}{2n}} \approx 1.469368 \times 10^{-39}$

$\Rightarrow$ *First difference is at the 40th significant digit...*

$\Rightarrow$ *Both estimates are great (and practically the same as exact)*

Context: Probability of winning the powerball lottery and getting hit by lightning the same day is $\approx 6.25 \times 10^{-18}$

Collisions: Using Collision Probability Estimate

How many balls for fixed number of bins?

Question: For n bins, how many balls can we throw before $\mathbb{P}[\textit{collision}] \approx 1/2$?

- This is the original birthday "paradox" (with $n = 365$).
- Hashing with a k-bit hash, how many before a duplicate?

Using our estimate, set $e^{-m^2/2n} = 1/2 \implies \frac{m^2}{2n} = \ln 2 \implies m^2 = (2\ln 2)n$

$$m = \sqrt{(2\ln 2)n} \approx 1.177\sqrt{n}$$

For birthday problem ($n = 365$), get $m = 1.177\sqrt{365} = 22.49$

$\Rightarrow$ So 23 or more for likely collision

For k-bit hash, $n = 2^k$, so $m = 1.177\sqrt{2^k} = 1.177 \times 2^{k/2}$

128-bit hash (i.e., MD-5 $\leftarrow$ don't use!): approx 2^{64} (seven months at a trillion/sec)

256-bit hash (i.e., SHA-256): approx 2^{128}

At a trillion/second, this would take "a trillion lifetimes of the universe"

Collisions: Using Collision Probability Estimate

How many bins for fixed number of balls?

Same formula – different knowns/unknowns...

What if: We know how many items are going to be hashed, and want to pick the size of the hash function to get particular probability bound?

Example: Hashing a trillion values ($m \approx 2^{40}$) – what n gives $\mathbb{P}[\textit{collision}]$ less than one in a billion ($\approx 2^{-30}$)?

Set $1 - e^{-m^2/2n} = 2^{-30}$ or... $e^{-m^2/2n} = 1 - 2^{-30}$

For small x, $e^{-x} \approx 1 - x$, so we want $m^2/2n \approx 2^{-30}$

$\Rightarrow$ *This is just the union bound! Cool.*

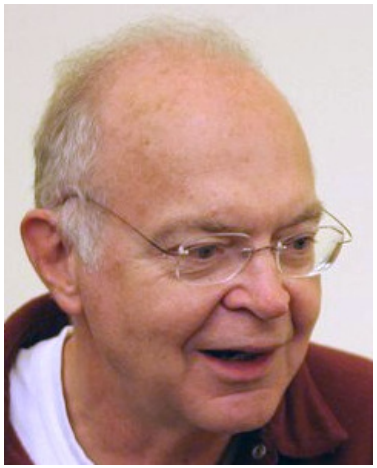
$$\frac{(2^{40})^2}{2n} = 2^{-30} \implies \frac{(2^{40})^2}{2n} = \frac{2^{80}}{2n} = \frac{2^{79}}{n} \quad \text{... set } = 2^{-30} \implies n = 2^{109}$$

So a 109-bit hash function is sufficient!

$\Rightarrow$ *SHA-1 is 160 bits – I'd use SHA-256 anyway.*

Coupon Collector Problem

Hot trend for 2026: Cards for Turing Award winners in cereal boxes!



Donald Knuth
1974

Some other school
in the bay area

Richard Karp
1985

Berkeley!

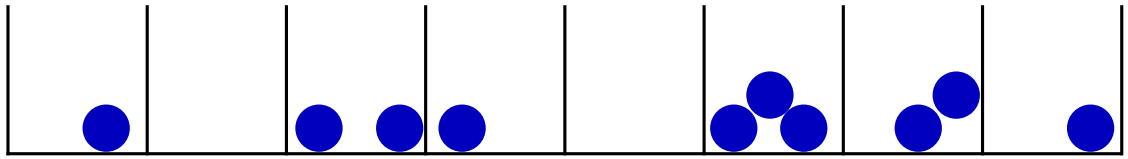
Shafi Goldwasser
2012

Berkeley!

There are n different cards (unlimited copies), each box has a random person

Question: What is the probability you have all after buying m boxes?

Back to Balls and Bins...



m balls
n bins

"Buy a box" = "Throw a ball"
$\Rightarrow$ "Buy m boxes" = "Throw m balls"

"Box has a random card" = "Ball goes into a random bin"
$\Rightarrow$ "n different cards" = "n bins to land in"

Question: What is the probability of no empty bins?

Result preview:

Theorem: If you buy m boxes,

(a) $\mathbb{P}[\text{miss one specific item}] \approx e^{-\frac{m}{n}}$

(b) $\mathbb{P}[\text{miss any one of the items}] \leq ne^{-\frac{m}{n}}$.

Coupon Collector Problem: Analysis

Event E_i = "fail to get Shafi Goldwasser (card i) in m cereal boxes"
Or: "bin i is empty" (Reminder: n bins total)

$\mathbb{P}[$ Fail the first time $] = (1 - \frac{1}{n})$
$\mathbb{P}[$ Fail the second time $] = (1 - \frac{1}{n})$
And so on ... for m times.

All trials are independent, so:

$$\mathbb{P}[E_i] = (1 - \tfrac{1}{n}) \times \cdots \times (1 - \tfrac{1}{n}) = (1 - \tfrac{1}{n})^m$$

Fact from Calculus: For large n, $(1 - \frac{1}{n})^n \approx e^{-1}$ [rel error $< 2\%$ for $n \geq 30$]

$$\text{So:} \qquad \mathbb{P}[E_i] = (1 - \tfrac{1}{n})^m = \left[(1 - \tfrac{1}{n})^n\right]^{m/n} \approx e^{-m/n}$$

For $\mathbb{P}[E_i] \approx \frac{1}{2}$, we need:

$$e^{-m/n} = 1/2 \implies e^{m/n} = 2 \implies m = (\ln 2)n \approx 0.69n \text{ boxes}$$

Coupon Collector: Collect All Cards!

Question: What is the probability that *some* card is missing?
Or: ... that some bin is empty?

Events: $E_i =$ "fail to get card i" (for $i = 1, \ldots, n$)

$E =$ "missing *some* card"
$= E_1 \cup E_2 \cup \cdots \cup E_n$

We just calculated $\mathbb{P}[E_i] \approx e^{-m/n}$

Using Union Bound:

$$\mathbb{P}[E] \leq \mathbb{P}[E_1] + \mathbb{P}[E_2] + \cdots + \mathbb{P}[E_n] \approx ne^{-m/n}$$

Coupon Collector: How many boxes do we need?

Events: $E_k =$ "fail to get card k" (for $k = 1, \ldots, n$)

$E =$ "missing *some* card"

Just saw that $\mathbb{P}[E] \leq ne^{-m/n}$

What if we want to ensure this probability is $\leq p$ for some p?

Solve: $ne^{-m/n} = p \implies e^{-m/n} = \frac{p}{n} \implies e^{m/n} = \frac{n}{p} \implies m/n = \ln\left(\frac{n}{p}\right)$

$\implies m = n\ln\left(\frac{n}{p}\right)$

So when $m = n\ln\left(\frac{n}{p}\right)$, $\mathbb{P}[E] \leq p$

$\therefore$ To get $\mathbb{P}[E] \leq 1/2$ it suffices to have $m = n\ln(2n)$

Examples:

$n = 100 \implies$ Use $m = 530$

$n = 1000 \implies$ Use $m = 7601$

Load Balancing: Two equivalent problems

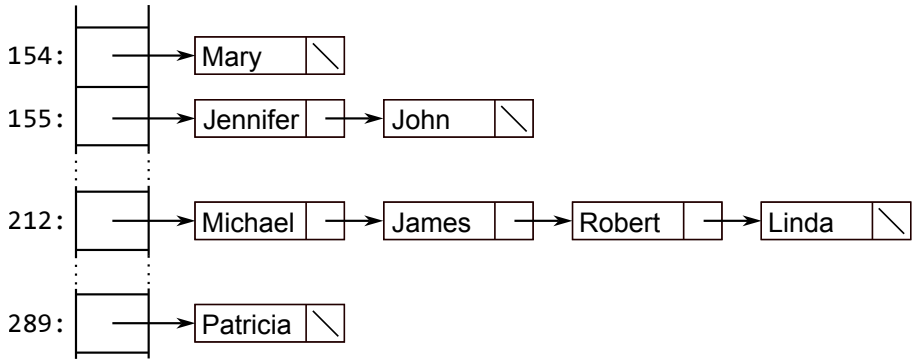
Problem from the class notes: Perform m jobs using n processors.

Proposed solution (doesn't require centralized coordination):

Assign each job to random processor

Question: What is probability that heaviest-loaded processor isn't too bad?

Alternatively: Hash table with chaining – hash values lead to linked lists:



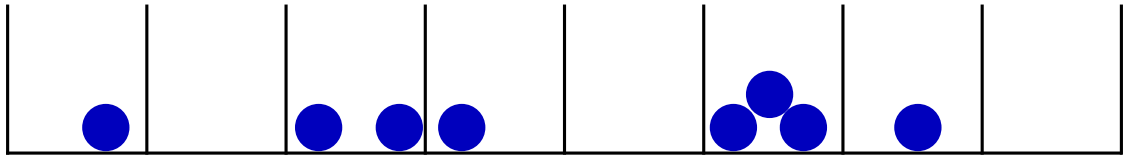
Worst-case lookup time? Length of longest list

Question: What is probability that worst-case lookup time isn't too bad?

This is the same problem!

Back to Balls and Bins (Again!)

Simpler: $n = m$ (can generalize to hash tables with any constant load factor)



Question: What is probability that bin with most balls doesn't have too many?

This is very vague... let's make it more concrete.

Event A_k = "some bin has at least k balls" (or: max balls $\geq k$)

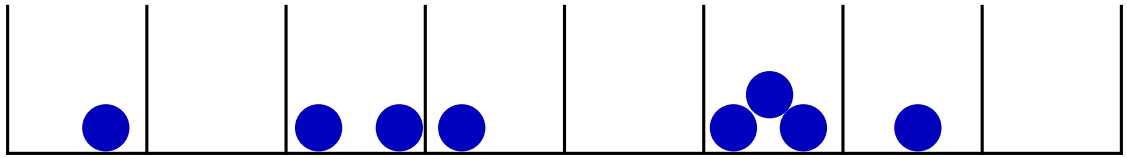
Question: What is the smallest k such that $\mathbb{P}[A_k] \leq \frac{1}{2}$?

Or: unlikely (prob $< \frac{1}{2}$) to have too many ($> k$) balls in any bin

Also consider this rephrasing:

For what value k is it more likely than not that the worst-case hash table lookup time is $< k$? (median worst-case lookup time)

Load Balancing: Analysis



n balls
n bins

Event A_k = "some bin has at least k balls" (or: max balls $\geq k$)

Question: What is the smallest k such that $\mathbb{P}[A_k] \leq \frac{1}{2}$?

Next: we'll get an upper bound for an upper bound for an upper bound...
$\Rightarrow$ *This is dangerous, but in this case it works pretty well!*

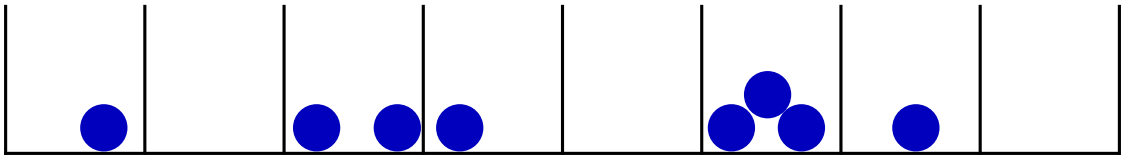
First upper bound: Analyze one bin, and use union bound.

Let event $A_k(i)$ = "bin i has at least k balls"

New question: What is the smallest k such that $\mathbb{P}[A_k(i)] \leq \frac{1}{2n}$?
$\Rightarrow$ *Note: all bins the same, so this does not depend on i.*

Then by union bound: $\mathbb{P}[A_k] = \mathbb{P}[\bigcup_i A_k(i)] \leq n \cdot \mathbb{P}[A_k(0)] \leq n \cdot \frac{1}{2n} = \frac{1}{2}$.

Load Balancing: Analysis



Let event $A_k(i) =$ "bin i has at least k balls"

New question: What is the smallest k such that $\mathbb{P}[A_k(i)] \leq \frac{1}{2n}$?

Note: If $\geq k$ balls in bin i, then $\exists$ a size k subset S of balls that went to bin i
$\Rightarrow$ *Might be larger subset, or more than one subset, ... but at least one.*

For any size k subset S, $\mathbb{P}[\text{"all balls in } S \text{ fall in bin } i\text{"}] = \frac{1}{n^k}$

Union bound... again... over $\binom{n}{k}$ subsets of size k:

$$\mathbb{P}[A_k(i)] = \mathbb{P}[\bigcup_{S,|S|=k} \text{"all balls in } S \text{ fall in bin } i\text{"}] \leq \binom{n}{k}\frac{1}{n^k}$$

Load Balancing: Analysis

n balls
n bins

Let event $A_k(i) =$ "bin i has at least k balls"

New question: What is the smallest k such that $\mathbb{P}[A_k(i)] \leq \frac{1}{2n}$?

$$\mathbb{P}[A_k(i)] = \mathbb{P}[\bigcup_{S,|S|=k} \text{"all balls in } S \text{ fall in bin } i\text{"}] \leq \binom{n}{k}\frac{1}{n^k}$$

Bounding the last term: $\binom{n}{k}\frac{1}{n^k} = \frac{n(n-1)\cdots(n-k+1)}{k!}\frac{1}{n^k} \leq \frac{1}{k!}$

... using Stirling's approximation, $k! \approx 2n$ when $k = \frac{\ln n}{\ln \ln n}$ (see notes)

So $k = \frac{\ln n}{\ln \ln n} \implies \mathbb{P}[A_k(i)] \leq \frac{1}{2n} \implies \mathbb{P}[A_k] \leq \frac{1}{2}$.

Or: With probability $\geq \frac{1}{2}$, no bin has more than $\frac{\ln n}{\ln \ln n}$ balls

Discuss: How does average or worst-case lookup compare to balanced tree?

Summary

Considered three real applications – all really just "balls and bins!"

- Analyzing hash functions
  For: data structures, file identification/integrity, cryptography, blockchain, ...

  $\mathbb{P}[\textit{collision}] = p \approx e^{-m^2/2n}$

  Given any two of p, m, and n, can solve for the other
- Data coverage by random sampling
  Exploring randomly – how long to see everything?

  Coupon collector: Get all with probability $\geq 1/2$ with $n\ln(2n)$ samples
- Load balancing
  Spreading out computational work or resources evenly

  When $m = n$, probability is $\leq 1/2$ that worst-case bin has $> \frac{\ln n}{\ln \ln n}$ balls