How common are collisions?

It’s like the birthday paradox. Imagine all 366 birthdays equally likely

- One person in a room
- Pick a person uniformly at random
  - They go into the room
  - They call “stop” if shared birthday

\[
\frac{365}{366} \times \frac{364}{366} \times \frac{363}{366}
\]
Side question: Average Class Size

Some universities advertise average class size

- 10 classes offered
- One class has 910 enrolled
- Other nine have 10 each

Imagine every student takes only one class

What is the average class size?

\[
\frac{1000}{10} = 100
\]

What do students think it is?

\[
\frac{1}{1000} (90 \cdot 10 + 910 \cdot 910) = 829
\]
Analysis of Hash Chaining

Recall $\alpha = \frac{n}{m}$

- **Worst-case** time for hashing with chaining?

- Let’s consider **average-case** time.
  - Expected length of a bucket?

- Time for an unsuccessful search?
Hash Functions

For use in converting key to array index.

- Positive Integers: modular hashing
- Floating-point numbers
  - Multiply-and-round
  - Binary representation
- Strings
- Compound keys
A good hash function for a given data type:

- Consistent
- Efficient to compute
- Uniformly distribute the set of keys
Ways to View Keys

Standard is one of two ways

▶ Each key is a tuple: \((x_1, x_2, \ldots, x_d)\)

▶ Each key is a non-negative integer
Summing Components

Suppose your key is a $d$-tuple

$$h(k) = \sum_{i=1}^{d} x_i$$

$$h(\text{"mike"}) = m + i + k + e$$
$$= 13 + 9 + 11 + 5 = 38$$

$$h(\text{"arts"}) = h(\text{"rats"}) = h(\text{"star"}) = h(\text{"tars"})$$
Choose non-zero $a \neq 1$ and:

$$h(k) = x_1 a^{d-1} + x_2 a^{d-2} + \ldots + x_{d-1} a + x_d$$

This can be written as:

$$h(k) = x_d + a(x_{d-1} + a(x_{d-2} + \ldots + a(x_3 + a(x_2 + ax_1)) \ldots))$$

$$a=37$$

$$h("mike") = m \cdot 37^3 + i \cdot 37^2 + k \cdot 37 + e$$

$$= e + 37(k + 37(i + 37m))$$
Approaches We’ve Seen So Far

Assuming suitable hash function

- Separate Chaining:
  - Insert: $O(1)$ if LL reasonable
  - Search: expected $\alpha + 1$

- Linear and Quadratic Probing:
  - Can be $O(n)$ insertion/search

- Slow down insert to improve search?
If one hash table is so great...

- Why not two hash tables?
  - Equal sizes
- Two (different) hash functions
- Direct storage, no probing.
- How do we search?
- How do we delete?
- How about insertion?
Example Insertion

$h_0(k) = k \% 11 \quad h_1(k) = (k/11) \% 11$

Insert: 12, 26, 92, 23, 28, 94, 15

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td></td>
<td></td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>92</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Insert 26
Visualizing Cuckoo Hashing

<table>
<thead>
<tr>
<th>Key</th>
<th>12</th>
<th>26</th>
<th>92</th>
<th>23</th>
<th>28</th>
<th>94</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_0$</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>$h_1$</td>
<td>1</td>
<td>2</td>
<td>8</td>
<td>2</td>
<td>2</td>
<td>8</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>23</td>
<td></td>
<td></td>
<td>26</td>
<td></td>
<td>94</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

|   |   | 12 | 28 |   |   |   | 92 |   |   |   |

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
**Theorem:** Let $G$ be an undirected connected graph. We can direct the edges of $G$ such that no vertex has more than one incoming edge if and only if $G$ has at most one cycle.
Relevant Graph Theory

**Theorem:** Let $G$ be an undirected connected graph. We can direct the edges of $G$ such that no vertex has more than one incoming edge if and only if $G$ has at most one cycle.

**Importance:**

- Connected component has at most one cycle?
- What about when a second cycle?