09-Hashing

Example 1: Seats in class

Option 1: Array with 40 spots, 11 have names

Option 2: Map/Linked List - 11 spots, 11 names

Example 2: Duke Student SSN's
Terms:

- $n$ - number of items in table 6000
- $m$ - number of spots in hash table 10000
- $\alpha$ - load factor $\frac{n}{m}$ 0.6
- $k$ - keys SSNs
- $U$ - universe of possible keys 0-999,999,999
- $|U|$ - number of things in universe 1,000,000,000
- $h(k)$ - hash function last 4 digits

What do we want our hash functions to do?
Hash Functions

Assumption of simple uniform hashing: each key is equally likely to hash to any of the m slots, independently of where any other key has hashed to.

the next key is equally likely to hash to any of the m slots

\[
\begin{align*}
\text{Division} \quad & h(k) = k \mod m \quad m = 10,000 \quad 0 \leq k \leq 15,000 \\
\text{Multiplication} \quad & h(k) = \lfloor m(kA \mod 1) \rfloor \quad 0 < A < 1 \\
\quad & \text{ranges from 0 to } m-1
\end{align*}
\]

\[
A = \frac{(15 - 1)}{2} = 7.0
\]
Chaining Analysis

Assuming that our hash function is uniform, how long does it take to:

- Insert? \( \text{Expected-case} \ 0(1) \quad \text{Worst-case} \ 0(1) \)
- Find? \( \Theta(1 + \alpha) \quad O(n) \)
- Delete? \( \Theta(1 + \alpha) \quad O(n) \)

Indicator Random Variables

\[ X_{ij} = \begin{cases} 1 & \text{if item } j \text{ hashes to the same spot as item } i \\ 0 & \text{otherwise} \end{cases} \]
\[ E \left[ I_{\text{FINO}(n)} \right] = E \left[ \frac{1}{n} \sum_{i=1}^{n} \left( 1 + \sum_{j=i+1}^{n} E[X_{ij}] \right) \right] \]

\[ = \frac{1}{n} \sum_{i=1}^{n} \left( 1 + \sum_{j=i+1}^{n} E[X_{ij}] \right) \]

\[ = 1 + \frac{1}{n} \sum_{i=1}^{n} \sum_{j=i+1}^{n} E[X_{ij}] \]

\[ = 1 + \frac{1}{mn} \sum_{i=1}^{n} (n-i) \]

\[ = 1 + \frac{1}{mn} \sum_{i=1}^{n-1} i \]

\[ = 1 + \frac{1}{mn} \frac{n(n-1)}{2} \]

\[ = 1 + \frac{n}{2m} - \frac{1}{2m} \]

\[ = 1 + \frac{\alpha}{2} - \frac{\alpha}{2n} = O\left(1 + \alpha \right) \]
Open Addressing
Elements are stored in the hash table itself.
Max $\alpha = 1$
Saves space

Hash function now has form $h(k, i)$.

\begin{align*}
  k &= \text{key} \\
  i &= \text{probe number} \ (0 \to m - 1) \\
  h(k, i) &= (h'(k) + h_2(k, i)) \mod m
\end{align*}
Probing

**Linear**

\[ h(k, i) = (h'(k) + i) \mod m \]

**Quadratic**

\[ h(k, i) = (h'(k) + c_1 i + c_2 i^2) \mod m \]

**Double Hashing**

\[ h(k, i) = (h'(k) + h_2(i)) \mod m \]
Assuming uniform hashing:

Worst-case

- Insert: $O(n)$
- Find: $O(n)$
- Delete: $O(n)$

Expected number of probes in an unsuccessful search: $\frac{1}{1-\alpha}$

In a successful search: $\frac{1}{\alpha \ln \frac{1}{1-\alpha}}$