Hash Tables

Weiss, Chapter #5

Hash Tables
• Constant time accesses!
• A hash table is an array of some fixed size, usually a prime number.
• General idea:
  \[ \text{key space (e.g., integers, strings)} \rightarrow \text{hash func.} \rightarrow \text{hash table} \rightarrow \text{TableSize –1} \]

Example
• key space = integers
• TableSize = 10
  \[ h(K) = K \mod 10 \]
  • Insert: 7, 18, 41, 34
  • How do we find them?

Another Example
• key space = integers
• TableSize = 6
  \[ h(K) = K \mod 6 \]
  • Insert: 7, 18, 41, 34
  • How do we find them?

Hash Functions
1. simple/fast to compute,
2. Avoid collisions
3. have keys distributed evenly among cells.

Perfect Hash function:
Sample Hash Functions:

- key space = strings
- $S = s_0 s_1 s_2 \ldots s_{k-1}$

1. $h(s) = s_0 \mod \text{TableSize}$
2. $h(s) = \left( \sum_{i=0}^{k-1} s_i \right) \mod \text{TableSize}$
3. $h(s) = \left( \sum_{i=0}^{k-1} s_i \cdot 37^i \right) \mod \text{TableSize}$

Collision Resolution

**Collision:** when two keys map to the same location in the hash table.

Two ways to resolve collisions:
1. Separate Chaining
2. Open Addressing (linear probing, quadratic probing, double hashing)

Separate Chaining

- **Separate chaining:** All keys that map to the same hash value are kept in a list (or "bucket").

How big should the hash table be?

- For Separate Chaining:

Open Addressing

- **Linear Probing:** after checking spot $h(k)$, try spot $h(k)+1$, if that is full, try $h(k)+2$, then $h(k)+3$, etc.
Open Addressing
Examine cells in the order:
\[ h_0(k), h_1(k), h_2(k), \ldots \]
where: \[ h_i(k) = (\text{hash}(k) + f(i)) \mod \text{TableSize} \]

Linear Probing
\[ f(i) = i \]
• After searching spot \( \text{hash}(k) \) in the array, look in \( \text{hash}(k) + 1, \text{hash}(k) + 2, \text{etc.} \)

Quadratic Probing
\[ f(i) = i^2 \]
• After searching spot \( \text{hash}(k) \), look in the 1st, 4th, 9th, etc. spots after \( \text{hash}(k) \).
• Less likely to encounter primary clustering.

Double Hashing
\[ f(i) = i \cdot \text{hash}_2(k) \]
• Items that hash to the same location with \( \text{hash}(k) \) won’t have the same probe for \( \text{hash}_2(k) \).

Rehashing
**Idea**: When the table gets too full, create a bigger table and hash all the items from the original table into the new table.
• When to rehash?
  – half full (\( \lambda = 0.5 \))
  – when an insertion fails
  – some other threshold
• Cost of rehashing

Hash Table example
\[ \begin{array}{c|c|c}
\hline
0 & \text{Insert:} & 89 \\
9 & 18 & 49 \\
8 & 58 & 69 \\
7 & & \\
6 & & \\
5 & & \\
4 & & \\
3 & & \\
2 & & \\
1 & & \\
\hline
\end{array} \]
+ marking deleted items
+ choice of table size