Operations

Search \((T, k)\)

Insert \((T, x)\)

Delete \((T, x \text{ or } k)\)

\(T\) - hash table, zero-indexed array

\(x\) - element to insert, delete, search for

\(x, \text{key}\) - key to search for

\(k\) - think of as id #

\(m\) - number of slots (size of array)

\(n\) - number of slots in hash table

\[
\frac{n}{m} = \alpha
\]

load factor

\(h(k)\) - hash function map from keys to slots

Chaining - each slot is a linked list

keys: 2, 3, 5, 7, 11, 13, 17, 19

have 10 slots

\(h(k) = k \mod 10\)

\(n = 8\)

\(m = 10\)

\[\alpha = \frac{8}{10} = \text{average length of list}\]

\[0 + 1 + 1 + 2 + 0 + 1 + 0 + 1 + 1 = 8\]

\[\frac{8}{10} = 0.8\]
Theorem 11.1

In a hash table in which collisions are resolved by chaining, an unsuccessful search takes average-case time $\Theta(1 + \alpha)$, under the assumption of simple uniform hashing.

Assume $h(k)$ is $\Theta(1)$ because keys are small

Note $\alpha$ is average length of linked list

Theorem 11.2

In a hash table in which collisions are resolved by chaining, a successful search takes average-case time $\Theta(1 + \alpha)$, under the assumption of simple uniform hashing.
Chaining

average time \(\Theta(1+x)\)

of dp

suppose \(n-1\) elements

how add another element

assumption of simple uniform hashing

equally likely

average is

implies new element is equally likely
to map to any of the slots

\[
\frac{1+2+2+3+1+2+1+3+6+2}{10} = \frac{8 + 10}{10} = \alpha + 1
\]

If this is so for the \(n\)th element,
the expected length for previous els will be smaller
\[ \frac{n-1}{m} \text{ elements} \]
\[ m \text{ slots} \]
\[ \text{average length} \quad \frac{n-1}{m} \]

So

\[ \text{average length} \]
\[ \text{of modified slot} \]
\[ \text{is} \quad \frac{n-1}{m} + 1 \]

for n'th element

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

so the average length

The expected length of

any search will be

\[ < \alpha + 1 \]

including cost of hash function

average search time is \( \Theta(\alpha + 1) \)

so if \( \alpha \) is also \( \Theta(1) \) \( \text{[(n≈m or n≤m)]} \)

then time is \( \Theta(1) \)
Hash Functions

For example, if we know that the keys are random real numbers $k$ independently and uniformly distributed in the range $0 \leq k < 1$, then the hash function

$$h(k) = \lfloor km \rfloor$$

satisfies the condition of simple uniform hashing.

In the division method for creating hash functions, we map a key $k$ into one of $m$ slots by taking the remainder of $k$ divided by $m$. That is, the hash function is

$$h(k) = k \mod m$$

$m$ should be a prime number. Should not close to a power of 2. Pick a large prime $p$. Any hash $m$. $h(k) = (k \mod p) \mod m$

The multiplication method for creating hash functions operates in two steps. First, we multiply the key $k$ by a constant $A$ in the range $0 < A < 1$ and extract the fractional part of $kA$. Then, we multiply this value by $m$ and take the floor of the result. In short, the hash function is

$$h(k) = \lfloor m (kA \mod 1) \rfloor,$$

where “$kA \mod 1$” means the fractional part of $kA$, that is, $kA - \lfloor kA \rfloor$. 