Hash Tables

Direct-Address Tables
Hash Functions
Universal Hashing
Chaining
Open Addressing

Direct-Address Tables
Let \( U = \{0, \ldots, m - 1\} \), the set of possible keys.
Use array \( T[0 \ldots m-1] \) as a direct-address table.
Implies 1-1 correspondence between keys and slots.

**Direct-Address-Search** \((T, k)\)

```plaintext
return \( T[k] \)
```

**Direct-Address-Insert** \((T, x)\)

```plaintext
T[x.key] \leftarrow x
```

**Direct-Address-Delete** \((T, k)\)

```plaintext
T[x.key] \leftarrow nil
```

Advantage: operations are \( \Theta(1) \).
Disadvantage: \( \Theta(|U|) \) space required.

Hash Tables
Let \( K \) be the set of keys to be stored.

Goal: use \( \Theta(|K|) \) space and \( \Theta(1) \) time/op.

Idea: Use array \( T[0 \ldots m - 1] \) as a hash table,
and use a \( \Theta(1) \) hash function \( h \), where
\( h: U \rightarrow \{0, \ldots, m-1\} \) maps from keys to slots.

A *collision* is when two keys map to the same slot.
**Good Hash Functions**

Division method: \( h(k) = k \mod m \)
- \( m \) is prime, not close to any \( 2^i \).

Division variation: \( h(k) = (k \mod M) \mod m \)
- \( M \) is prime, << than \( |U| \), not close to any \( 2^i \).
- \( m \) is << than \( M \).

Multiplication method:
- \( h(k) = \lfloor m((kA) \mod 1) \rfloor \)
- \( m \) is a power of 2. \( A = (\sqrt{5} - 1)/2 \)

**Universal Hashing**

Let \( \mathcal{H} \) be a set of hashing functions.
\( \mathcal{H} \) is universal if \( h(k) = h(k') \) with prob. \( 1/m \)
- \( m \) is a prime number.

\( k = \langle k[1], \ldots, k[l] \rangle \), where \( 0 \leq k[i] < m \)
- Assign \( a[i] \leftarrow \text{RANDOM}(0, m-1) \)
- \( h(k) = \left( \sum_{i=1}^{l} a[i] \cdot k[i] \right) \mod m \)

The set of possible functions \( h(k) \) is universal.
- \( h(k) = h(k') \) with prob. \( 1/m \).
- If \( k[i] \neq k'[i] \), \( (a[i] \cdot (k[i] - k'[i])) \mod m \) has equally likely results.

**Horner's Method for Division Hash Function**

If \( k = \langle k[1], \ldots, k[l] \rangle \), and if \( 0 \leq k[i] < r \), then compute hash function by:

\[
h \leftarrow k[1] \mod m \\
\text{for } i \leftarrow 2 \text{ to } l \\
\quad \text{do } h \leftarrow (rh + k[i]) \mod m
\]

**Chaining**

In chaining, slots are linked lists of the elements that hash to that slot, i.e., collisions.

Consider \( m \) slots, \( n \) elts., load factor \( \alpha = n/m \).

- Worst-case: \( \Theta(n) \) if all elts. hash to same slot.
- Best-case: \( \Theta(1 + \alpha) \), each slot has \( \lfloor \alpha \rfloor \) or \( \lceil \alpha \rceil \).

- Average-case: Assume each slot is equally likely.

- Unsuccessful search: \( \Theta(1 + \alpha) \)
  - This is because average slot length = \( \alpha \).
Chaining, Part 2

Successful search: $\Theta(1 + \alpha)$
Before $i$th elt. inserted, avg. length $= (i - 1)/m$.
Expected position of $i$th elt. $= 1 + (i - 1)/m$.

Expected search length is the summation:
\[
\sum_{i=1}^{n} \frac{1}{n} \left(1 + \frac{i - 1}{m}\right) = 1 + \frac{\alpha}{2} - \frac{1}{2m}
\]

Open-Address Hashing

In open addressing, when a collision occurs, probe for an empty slot and insert the new elt. there.

The hash function becomes:
\[h : U \times \{0, \ldots, m-1\} \to \{0, \ldots, m-1\}\]
The probe sequence $\langle h(k, 0), \ldots, h(k, m-1)\rangle$ should include all the slots.

Open-Address Hashing, Part 2

**Hash-Insert** $(T, x)$
\[
\text{for } i \leftarrow 0 \text{ to } m-1 \\
\text{do } j \leftarrow h(x.key, i) \\
\text{if } T[j] = \text{NIL} \\
\text{then } T[j] \leftarrow x \\
\text{return } j
\]
error “hash table overflow”

**Hash-Delete** marks the slot as deleted.
**Hash-Search** must continue past deleted slots.
**Hash-Insert** can put new elts. in deleted slots.

Uniform Hashing Analysis

Uniform hashing assumes each open-address probe-sequence is equally likely.

Unsuccessful Search: $\Theta \left(\frac{1}{1-\alpha}\right)$
Let $p_i = \text{prob. exactly } i \text{ probes find full slots.}$
Let $q_i = \text{prob. first } i \text{ probes find full slots.}$
$q_1 = \frac{n}{m} \Rightarrow q_2 = \left(\frac{n}{m}\right) \left(\frac{n-1}{m}\right) < \alpha^2$
$q_i = \prod_{k=0}^{i-1} \frac{n-k}{m-k} \leq \left(\frac{n}{m}\right)^i = \alpha^i$
Uniform Hashing Analysis, Part 2

Average number of probes is:

\[ 1 + \sum_{i=1}^{n} i p_i = 1 + \sum_{i=1}^{n} q_i \leq \sum_{i=0}^{\infty} \alpha^i = \frac{1}{1 - \alpha} \]

Successful Search: \( \Theta \left( \frac{1}{\alpha} \ln \frac{1}{1 - \alpha} \right) \)

Inserting \( i \)th elt. = unsuccessful search \( i - 1 \) elts.
Average number of probes is:

\[ \sum_{i=1}^{\alpha} \left( \frac{1}{n} \left( \frac{1}{1 - (i-1)/m} \right) \right) \leq \frac{1}{\alpha} \ln \frac{1}{1 - \alpha} \]

Performance of Practical Methods

Quadratic Probing assumes \( m \) is a power of 2.

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

Successful Search: \( \Theta \left( \frac{1}{\alpha} \ln \frac{1}{1 - \alpha} \right) \)

Unsuccessful Search: \( \Theta \left( \frac{1}{1 - \alpha} \right) \)

Double Hashing, \( m \) is prime, \( 1 \leq h_2(k) \leq m - 1 \)

\[ h(k, i) = (h_1(k) + i h_2(k)) \mod m \]

Successful Search: \( \Theta \left( \frac{1}{\alpha} \ln \frac{1}{1 - \alpha} \right) \)

Unsuccessful Search: \( \Theta \left( \frac{1}{1 - \alpha} \right) \)

Performance of Practical Methods

Linear Probing: \( h(k, i) = (h'(k) + i) \mod m \)

Successful Search: \( \Theta \left( \frac{1}{1 - \alpha} \right) \)

Unsuccessful Search: \( \Theta \left( \frac{1}{1 - \alpha} \right) \)

Linear probing suffers from primary clustering, from long runs of occupied slots.

An empty slot preceded by \( i \) full slots gets filled next with probability \( (i + 1)/m \).