Hash Tables

Given a set of possible keys U, such that $|U| = u$ and a table of m entries, a **Hash function** h is a mapping from U to $M = \{1, ..., m\}$.

A collision occurs when two hashed elements have $h(x) = h(y)$.

Definition 1. *A hash function $h : U \rightarrow M$ is* **perfect** *for a set S if it causes no collisions for pairs in S.*

For any given S such that $|S| \leq m$ there is a perfect hash function.

For any S such that $|S| > m$ there is **no** perfect hash function.

If $|U| > m$ there is no perfect hashing function for all $S \subset U$, s.t. $|S| = m$.

Chaining

$h(.)$ - hash function.

A table $T[1..n]$ such that $T[k]$ is a pointer to a linked list of all the elements hashed to $T[k]$.

Insert k: add k to the linked list $T[h(k)]$.

Search/delete k: search ($+$ delete) in $T[h(k)]$.

The cost is proportional to the length of the link lists.

Hash Functions

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

$h(k) = (ak + b) \; mod \; m$,

$H = \{h(k) \mid 1 \leq a \leq m - 1, \; 0 \leq b \leq m - 1\}$

If m not a prime, let $p > m$ be a prime

$$h(k) = ((ax + b) \; mod \; p) \; mod \; m$$

Analysis of Hashing with Chaining

Let n be the number of keys stored in the table.

The **load factor** $\alpha = \frac{n}{m}$.

Worst case insert time either $O(1)$ or $O(n)$.

Worst case search/delete time $O(n)$.

For simple probabilistic analysis:

Simple Uniform Assumption: Keys are hashed to uniformly random and independent locations.

Assume that $h(.)$ is computed in $O(1)$ time.

Theorem 1. *In a hash table in which collisions are resolved by chaining, under the assumption of simple uniform hashing,*

1. *An unsuccessful search takes* $\Theta(1 + \alpha)$ *expected time.*

2. *A successful search takes* $\Theta(1 + \alpha)$ *expected time.*

Proof.

(1) The expected time of an unsuccessful search is the average length of a list, plus the time to compute $h(.)$ which is $O(1 + \alpha)$.

(2) We assume that the key being searched is equally likely any on the n keys in the tables.

Assume that a key is inserted at the head of the link list.

If the key we are searching was the i-th key to be inserted to the table, The expected number of elements in front of that key in its linked list is $\frac{n-i}{m}$.

The expected search time is

$$\frac{1}{n}\sum_{i=1}^{n}\left(1+\frac{n-i}{m}\right) \tag{1}$$

$$= 1+\frac{1}{nm}\sum_{i=1}^{n}(n-i) \tag{2}$$

$$= 1+\frac{1}{nm}\frac{n(n-1)}{2} = 1+\frac{\alpha}{2}+\frac{1}{2m} \tag{3}$$

□

Universal Hash Functions

Definition 2. *A family H of hash functions from U to M is* **2-universal** *if for all $x, y \in U$, such that $x \neq y$, and for a randomly chosen function h from H*

$$Pr(h(x) = h(y)) \ \leq \ \frac{1}{m}.$$

Let H be the set of all functions from U to M, then H is 2-universal.

Problem: There are u^m functions from U to M - requires $m \log u$ bits to choose, represent and store as a table.

Theorem 2. *Assuming that we hash n keys to a table of size m, $n \leq m$, using a hash function chosen at random from a 2-universal family of hash functions. The expected number of collisions of a given key is less than 1.*

Proof. Let $\delta(x, y, h) = 1$ iff $h(x) = h(y)$, else 0.

By definition for a given pair of keys x and y. $E[\delta(x, y, h)] = 1/m$.

There are $n - 1$ other keys in the table thus the expected number of collisions with a given key x is $(n - 1)/m$. $\square$

Theorem 3. *For any sequence of r operations, such that there are never more than s elements in the table, the expected total work is:*

$$r(1 + \frac{s}{m}).$$

Proof.

Let $\delta(x, y, h) = 1$ iff $h(x) = h(y)$, else 0.

Assume that when we insert (or delete) the element x while the set S is in the table. The time to insert (delete) key x is

$$1 + C(x, S)$$

where

$$C(x, S) = \sum_{y \in S} \delta(x, y, h).$$

$$E[C(x,S)] \;=\; \frac{1}{|H|}\sum_{h\in H}\sum_{y\in S}\delta(x,y,h) \;=$$

$$\frac{1}{|H|}\sum_{y\in S}\sum_{h\in H}\delta(x,y,h) \leq \frac{1}{|H|}\sum_{y\in S}\frac{|H|}{m} = \frac{|S|}{m}.$$

□

Constructing 2-universal hash functions

Let m be a prime number.

Let $(x_0, ..., x_r)$ be the binary representation of a key x.

Let $\bar{a} = (a_0, ..., a_r)$.

$$h_{\bar{a}}(x) = (\sum_{i=0}^{r} a_i x_i) \ mod \ m.$$

Let

$$H \ = \ \{h_{\bar{a}}(x) \mid a_i \in \{0, ..., m-1\} \ \}.$$

Theorem 4. *H is a family of 2-universal hash functions from U to M.*

Proof.

Fix x, y such that $x \neq y$.

We need to count the number of functions in H (vectors $\bar{a}$) for which

$$h_{\bar{a}}(x) \ = \ h_{\bar{a}}(y)$$

Assume without loss of generality that $x_0 \neq y_0$.

If $h_{\bar{a}}(x) = h_{\bar{a}}(y)$ then

$$a_0(x_0 - y_0) = \sum_{i=1}^{r} a_i(y_i - x_i)$$

Since m is a prime, the arithmetics is in a field, and for each $a_1, ..., a_r$ there is only one value of a_0 that satisfied this equation.

Thus, there are m^r functions in which x and y collide, or the probability is $1/m$. $\square$

Open Addressing

Keys are stored in the table - no pointers.

The hash function has two arguments;

- the key
- the probe number

$$h : U \times \{0, ..., m-1\} \rightarrow \{0, ..., m-1\}.$$

Insert(T,k)

1. $i \leftarrow 0$

2. Repeat

   2.1 $j \leftarrow h(k,i)$
   2.2 If $T[j] = NIL$ then
   2.2.1. $T[j] \leftarrow k$
   2.2.2. RETURN
   2.3 else $i \leftarrow i+1$

3. until $i = m$

4. ERROR: TABLE IS FULL.

Search(T,k)

1. $i \leftarrow 0$

2. Repeat

   2.1 $j \leftarrow h(k, i)$
   2.2 If $T[j] = k$ then RETURN j;
   2.3 $i \leftarrow i + 1$;

3. until $i = m$ or $T[j] = NIL$;

4. Return NIL.

Open Address Hash Functions

Linear Probing:

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

Double Hashing:

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

Analysis of Open Address Hashing

Assume uniform hashing, for a given key k, the probe sequence $h(k,0), h(k,1)....$ is a random permutation on $0, ..., m-1$.

Theorem 5. *For a open address table with load factor* $\alpha = n/m < 1$*, and assuming uniform hashing, the expected number of probes in an unsuccessful search is at most* $\frac{1}{1-\alpha}$*.*

Lemma 1. *Let X be a random variable with values in the Natural numbers $N = \{1, 2, 3, ...\}$, then*

$$E[X] = \sum_{i=1}^{\infty} iPr(X = i) \; = \; \sum_{i=1}^{\infty} Pr(X \geq i).$$

Proof.

$$\begin{aligned} E[X] &= \sum_{i=1}^{\infty} iPr(X = i) \\ &= \sum_{i=1}^{\infty} i(Pr(X \geq i) - Pr(X \geq i + 1)) \\ &= \sum_{i=1}^{\infty} Pr(X \geq i) \end{aligned}$$

□

Proof. Let T be the number of probes in an unsuccessful search.

Let $q_i = Pr(T - 1 \geq i)$, the probability that at least i probes accessed an occupied slot.

$q_1 = \frac{n}{m}$.

$q_2 = (\frac{n}{m})(\frac{n-1}{m-1})$.

For $i \leq n$,

$$\begin{aligned} q_i &= (\frac{n}{m})(\frac{n-1}{m-1}) \cdots \frac{n-i+1}{m-i+1} \\ &\leq (\frac{n}{m})^i \\ &= \alpha^i \end{aligned}$$

For $i > n$, $q_i = 0$.

$$E[T] = 1 + \sum_{i=1}^{n} q_i \leq \frac{1}{1-\alpha}.$$

□

Theorem 6. *The expected number of probes in inserting a new item to a table with load* α *is* $\frac{1}{1-\alpha}$.

Theorem 7. *The expected number of probes in a successful search in an open address table with load factor α is*

$$\frac{1}{\alpha}\ln\frac{1}{1-\alpha}+\frac{1}{\alpha},$$

assuming uniform hashing, and all keys are equally likely to be searched.

Proof.

The expected number of probes in searching for the key that was the $i+1$-th key inserted to the table is

$$\frac{1}{1-\frac{i}{m}}=\frac{m}{m-i}$$

Averaging over all keys

$$\frac{1}{n}\sum_{i=0}^{n-1}\frac{m}{m-i}$$

$$\begin{aligned}
&= \frac{m}{n}\sum_{i=0}^{n-1}\frac{1}{m-i} \\
&= \frac{1}{\alpha}(H_m - H_{m-n}) \\
&\leq \frac{1}{\alpha}(ln\ m + 1 - ln(m-n))) \\
&= \frac{1}{\alpha}(ln\frac{m}{m-n} + 1) \\
&= \frac{1}{\alpha}\ln\frac{1}{1-\alpha} + \frac{1}{\alpha}
\end{aligned}$$

□