Hash Tables

- A dynamic data structure that supports only
  - Search
  - Insert
  - Delete
- Typical implementations have $O(1)$ average case performance, but $\Theta(n)$-time in the worst case
- Suppose we want to represent a set of n items from some universe U
- A potentially efficient way of doing this is with a hash table of size m

$$m << |U|$$

- A hash function is any function

$$h : U \rightarrow [0 \ldots m-1]$$

assume $h(x)$ is computed in constant time

- To check for some item x
  - compute $h(x)$
  - check the corresponding cell in the hash table
- Two items x and y collide if the hash to the same value

$$h(x) = h(y) \qquad x \neq y$$

Collisions

- chaining
  each item in the hash table is a linked list
- linear probing
  put colliding item in the next empty cell
- Quadratic probing
- In general the cost of an operation on key x is $O(\ell(x))$-time where $\ell(x)$ is the number of distinct keys in the set that hash to $h(x)$
- Find a hash function that minimize $\ell(x)$

Chaining

- in chaining all items that hash to the same location are stored in a linked list.
- For some table T we define the load factor α of T as $\frac{n}{m}$
- The worst case behavior is $\Theta(n)$
- Average case depends on how effectively ~~of~~ our hash function distributes the items
- for now assume Simple Uniform Hashing
  - any key is equally likely to hash into any of the m slots.

- For $j = [0 \ldots m-1]$ let the length of that chain be n_j

$$n = n_0 + n_1 + n_2 + \ldots + n_{m-1}$$

- The $E[n_j] = \frac{n}{m}$

Theorem

An unsuccessful search will run in $\Theta(1 + \frac{n}{m})$ on average

Proof

- under simple uniform hashing a key k that is not in the table is equally likely to hash into any of the m slots.

- The expected time to search for k is just the length of the chain at $T[h(k)]$ or $n_{h(k)}$ $E[n_{h(k)}] = \frac{n}{m}$
- Including the time to compute $h(k)$ the total time required is $\Theta(1 + \frac{n}{m})$

Successful Search

Theorem:
A successful search will run in $\Theta(1+\frac{n}{m})$

Proof:
- the probability that some chain contains the target item is proportional to the number of items in the chain.

- assume the element being searched for is equally likely to be any of the n elements
- to find item x we must examine 1 plus the number of items that appear before x in x's list.
- Since new elements are placed a the beginning of the list, all elements that appear before x were inserted after x
- The expected number of elements is the average over the n elements x of 1 + number of items inserted after x that hash into x's list.

- Let x_i be the i^{th} element inserted into the table for $i = [1 \ldots n]$
- For items x_i and x_j define an indicator random variable X_{ij} where $X_{ij} = 1$ iff $h(x_i)$ and $h(x_j)$ are the same
- $Pr(h(x_i) = h(x_j)) = \frac{1}{m}$ if ~~$x_i \neq$~~ $i \neq j$

$$E[X_{ij}] = Pr(X_{ij} = 1) = \frac{1}{m}$$

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

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

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

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

$$= 1 + \frac{1}{nm}\sum_{k=0}^{n-1}k$$

$$= 1 + \frac{1}{nm}\cdot\frac{n(n-1)}{2}$$

$$= 1 + \frac{n-1}{2m}$$

$$= 1 + \frac{n}{2m} - \frac{1}{2m} \in \Theta\left(1+\frac{n}{m}\right)$$

- If the number of slots in the hash table is at least proportional to the number of elements were storing

$n \in O(m)$ then

$$\frac{n}{m} \in \frac{O(m)}{m} \in O(1)$$

Thus search takes constant time on average.