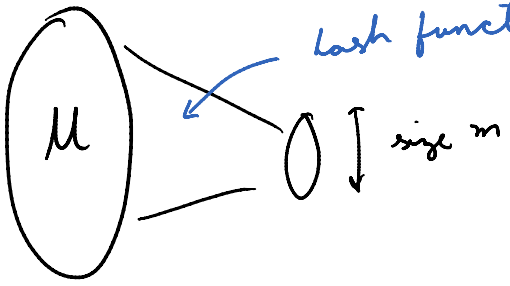
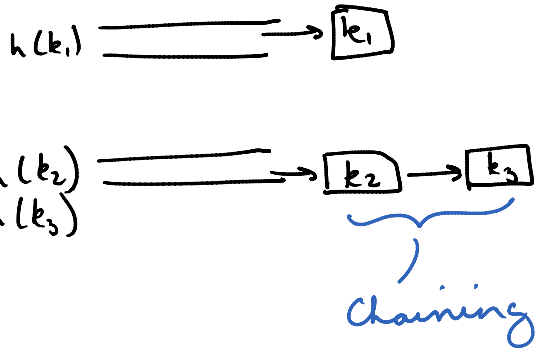
Hashing

Thursday, September 10, 2015
12:02 AM

- Dictionaries need to support search, insertion, deletion.
- Note that if we don't care about size of dictionary, all three can be implemented in $O(1)$ time.
- Question: How? Ans: Direct addressing
- Infeasible if universe of items too large (and number of inserted items is small)
  - Example: set of all IP addresses
-



-



- Collision if for two items i and j, $h(i) = h(j)$
- Need "random looking" hash function to avoid collisions

- Assume n keys inserted.

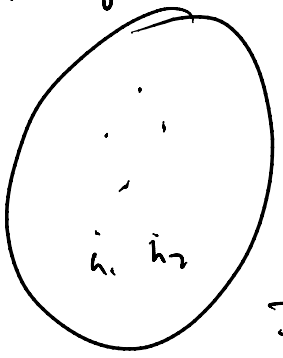
Load factor: $\alpha = \frac{n}{m}$

(average length of a chain)

- ~~Simple uniform hashing assumption~~ (SUHA):

Any item is equally likely to hash into one of the m slots (independently of any other item)

- Under SUHA, a search takes on average $O(1+\alpha)$ time.
- What are good hashing functions?
  - $h(k) = k \bmod m$
    - Prime m is a good choice
  - $h(k) = \lfloor m(kA \bmod 1) \rfloor$
    - Works well if m is a power of 2.
  - Universal hashing
    - Not heuristic but requires randomness
    - Choose uniformly from a collection of hash functions



$\mathcal{H}$ is universal if for any k and $k' \in U$,
$|\{h \in \mathcal{H} : h(k) = h(k')\}| \leq \frac{|\mathcal{H}|}{m}$.

Iow:

$\Pr_{h \in \mathcal{H}}[h(k) = h(k')] \leq \frac{1}{m} \quad \forall k, k'$.

(Would be satisfied if h was the collection of all functions. Why doesn't this work?)

Claim: If $h \in \mathcal{H}$, a universal hash family, then

$\mathbb{E}[n_{h(k)}] \leq \alpha = n/m$.

Pf: Y_k = # of items that hash to the same slot as k.

$$\mathbb{E}[Y_k] = \mathbb{E}\left[\sum_{k' \in T} X_{kk'}\right]$$

$$\leq \frac{n}{m} = \alpha.$$

Claim: Suppose $p > |U|$. For random $a, b \in \mathbb{Z}_p$, $a \neq 0$, let $h_{a,b}(k) = ((ak+b) \bmod p) \bmod m$. Then, $\mathcal{H} = \{h_{a,b}\}$ is universal.

Pf: Let $\varphi_{a,b}(k) = (ak+b) \bmod p$.
Fix any distinct k, l.

Claim: If $(a,b) \neq (a',b')$, then
$$(\varphi_{a,b}(k), \varphi_{a,b}(l)) \neq (\varphi_{a',b'}(k), \varphi_{a',b'}(l))$$

Pf: Let $r = ak+b \bmod p$
$s = al+b \bmod p$

$$a = (r-s)(k-l)^{-1} \bmod p$$
$$b = (r - k(r-s)(k-l)^{-1}) \bmod p$$

unique.

Obs: $\varphi_{a,b}(k) \neq \varphi_{a,b}(l)$ if $k \neq l$.
Each (a,b) maps (k,l) to a different element of $\{(r,s) \in [p]^2 : r \neq s\}$.

- Karp-Robin string matching
  - Pattern string P of length l, target string T

of length n.

- Naive: $O(nl)$
- Idea: Let's hash p and then compare to hashes of substrings of T.
  But again: $O(nl)$ time
- Rolling hash: Suppose alphabet is $[d]$.
  $h(Q) = (d^{l-1}Q_0 + d^{l-2}Q_1 + \cdots + Q_{l-1}) \bmod m$

  But note that:
  $$h(Q^{i+1}) = d(h(Q^i) - d^{l-1}Q_0^i) + Q_{l-1}^{i+1} \bmod m$$
  in $O(1)$ time
- Means whole thing can be done in $O(n)$ time provided load factor is $O(1)$, meaning $m > n$.