6.046	Lecture 8	Mar. 5, 2015

TODAY: Hashing
- review:
  - dictionaries
  - chaining
  - simple uniform
- universal hashing
  - why (useful)
  - how
- perfect hashing
  - how
  - why (it works)

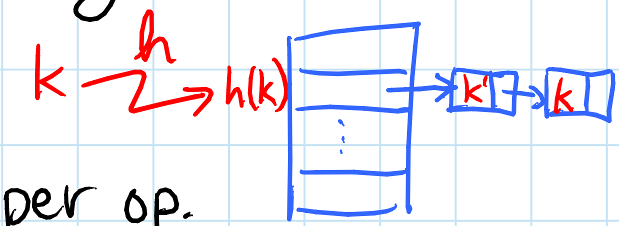
Dictionary problem: Abstract Data Type (ADT)

maintain set of items, each with a key, subject to
- insert(item): add item to set
- delete(item): remove item from set
- search(key): return item with key if it exists

- assume items have distinct keys (or that inserting new one clobbers old)
- easier than predecessor/successor problem solved by AVL/2-3 trees/skip lists $O(\lg n)$
  & by van Emde Boas $O(\lg\lg u)$

Hashing [6.006]

- goal: $O(1)$ time per operation & $O(n)$ space
- u = # keys over all possible items
- n = # keys/items currently in table
- m = # slots in table
- hashing with chaining achieves $\Theta(1+\alpha)$ time per op.



$\uparrow$ load factor n/m

ASSUMING simple uniform hashing:

$$\Pr_{k_1 \neq k_2}\{h(k_1) = h(k_2)\} = 1/m$$

$\leftarrow$ what you'd expect if totally uniform

- requires assuming input keys are random
- only works in average case

(like Basic Quicksort) :(

We will remove this unreasonable assumption.

Etymology:

English 'hash' (1650s) = cut into small pieces
$\leftarrow$ French 'hacher' = chop up
$\leftarrow$ Old French 'hache' = axe
(cf. English 'hatchet')
$\leftarrow$ Vulcan 'la'ash' = axe

Universal hashing:

- choose a random hash function h from $\mathcal{H}$
- require $\mathcal{H}$ to be a universal hash family:

$$\Pr_{h \in \mathcal{H}} \{h(k) = h(k')\} \le \frac{1}{m} \text{ for all } k \ne k'$$

- now just assuming h is random
- no assumption about input keys

(like Randomized Quicksort)

Theorem: for n arbitrary distinct keys & for random $h \in \mathcal{H}$, & $\mathcal{H}$ universal

$$E[\text{\# keys colliding in a slot}] \le 1 + \alpha$$

$\hookrightarrow n/m$

Proof:
- consider keys $k_1, k_2, \ldots, k_n$
- let $I_{i,j} = \begin{cases} 1 & \text{if } h(k_i) = h(k_j) \\ 0 & \text{else} \end{cases}$

$\boxed{\text{INDICATOR RANDOM VARIABLE}}$

$$\begin{aligned}
E[\text{\# keys hashing to same slot as } k_i] &= E\left[\sum_{j=1}^{n} I_{i,j}\right] \\
&= \sum_{j=1}^{n} E[I_{i,j}] \quad \leftarrow \text{linearity of expectation} \\
&= \sum_{j \ne i} \underbrace{E[I_{i,j}]}_{\substack{= \Pr\{I_{i,j}=1\} \leftarrow \text{indicator random var.} \\ = \Pr\{h(k_i)=h(k_j)\} \leftarrow \text{def. of } I_{i,j} \\ \le 1/m \leftarrow \text{universality}}} + E[I_{i,i}] \\
&\le n/m + 1 \qquad \square
\end{aligned}$$

$\Rightarrow$ Insert, Delete, Search cost $O(1+\alpha)$ expected.

Do universal hash families exist? YES:

$\mathcal{H} = \{$all hash functions
$h: \{0, 1, \ldots, u-1\} \to \{0, 1, \ldots, m-1\}\}$ is universal

... but this is useless:

- storing h takes $\lg(m^u) = u \lg m$ bits $\gg n$
  ~ just like direct map table (big array)
- would need to precompute u values
  $\Rightarrow \Omega(u)$ time, possibly ω(# operations)

Dot-product hash family:

- assume m is prime (find nearby prime)
- assume $u = m^r$ for integer r (round up else)
- view keys in base m: $k = \langle k_0, k_1, \ldots, k_{r-1} \rangle$
- for key $a = \langle a_0, a_1, \ldots, a_{r-1} \rangle$
  define $h_a(k) = a \cdot k \bmod m$
  $= \sum_{i=0}^{r-1} a_i \cdot k_i \bmod m$

  dot product

  cut up & mix = hachet

- $\mathcal{H} = \{h_a \mid a \in \{0, 1, \ldots, u-1\}\}$

- storing $h_a \in \mathcal{H}$ requires just storing 1 key, a
- word RAM model: manipulating $O(1)$ machine words takes $O(1)$ time, & "objects of interest" (here: keys) fit in a machine word

$\Rightarrow$ computing $h_a(k)$ takes $O(1)$ time
[$O(\log_m u)$ using just $+$ & $\cdot$ ~ can you do better?]

Theorem: dot-product hash family $\mathcal{H}$ is universal

Proof: take any two keys $k \neq k'$

$\Rightarrow$ differ in some digit, say $k_d \neq k'_d$

— let $\underline{\text{not } d} = \{0, 1, \ldots, r-1\} \setminus \{d\}$

$$\Pr_a \{ h_a(k) = h_a(k') \}$$

$$= \Pr_a \left\{ \sum_{i=0}^{r-1} a_i \cdot k_i = \sum_{i=0}^{r-1} a_i \cdot k'_i \pmod m \right\}$$

$$= \Pr_a \left\{ \sum_{i \neq d} a_i \cdot k_i + a_d \cdot k_d = \sum_{i \neq d} a_i \cdot k'_i + a_d k'_d \pmod m \right\}$$

$$= \Pr_a \left\{ \sum_{i \neq d} a_i (k_i - k'_i) + a_d (k_d - k'_d) = 0 \pmod m \right\}$$

$$= \Pr_a \left\{ a_d = -\underbrace{(k_d - k'_d)^{-1}}_{m \text{ prime} \Rightarrow \mathbb{Z}_m \text{ has multiplicative inverses}} \sum_{i \neq d} a_i (k_i - k'_i) \pmod m \right\}$$

$$= \mathop{E}_{a_{\text{not } d}} \left[\Pr_{a_d} \{ a_d = f(k, k', a_{\text{not } d}) \} \right]$$

because a_d is independent from $a_{\text{not } d}$

$$\left(= \sum_x \Pr\{a_{\text{not } d} = x\} \Pr_{a_d}\{a_d = f(k, k', x)\} \right)$$

$$= \mathop{E}_{a_{\text{not } d}} [1/m]$$

$$= 1/m$$

$\square$

Another universal hash family: [CLRS]

- choose prime $p \geq u$ (once)
- $h_{ab}(k) = [(a \cdot k + b) \bmod p] \bmod m$
- $\mathcal{H} = \{ h_{ab} \mid a, b \in \{0, 1, \ldots, u-1\} \}$

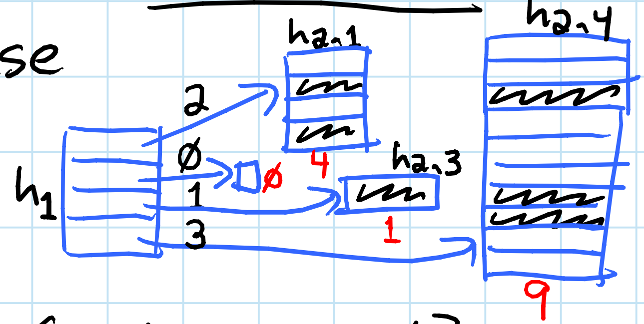
Static dictionary problem: given n keys to store in table, support Search(k)

Perfect hashing: [Fredman, Komlós, Szemerédi 1984]

→ no collisions

- polynomial build time w.h.p. (nearly linear)
- $O(1)$ time for Search, in worst case
- $O(n)$ space in worst case

Idea: 2-level hashing



① pick $h_1 : \{0, 1, \dots, u-1\} \to \{0, 1, \dots, m-1\}$
from a universal hash family
for $m = \Theta(n)$ (e.g. nearby prime)
- hash all items with chaining using h_1

② for each slot $j \in \{0, 1, \dots, m-1\}$:
- let ℓ_j = # items in slot j = $|\{i \mid h(k_i) = j\}|$
- pick $h_{2,j} : \{0, 1, \dots, u-1\} \to \{0, 1, \dots, m_j\}$
  from a universal hash family
  for $\ell_j^2 \le m_j \le O(\ell_j^2)$ (e.g. nearby prime)
- replace chain in ① slot j with hashing-with-chaining using $h_{2,j}$

Space $= O(n + \sum_{j=0}^{m-1} \ell_j^2)$

- to guarantee space $= O(n)$:

(1.5) if $\sum_{j=0}^{m-1} \ell_j^2 > c\,n$ then redo step ①

→ constant to be chosen

Search time $= O(1)$ for first table (h_1)
$+ O(\text{max chain size in second table})$

- to guarantee $= O(1)$:

$\to$ ANY collision

(2.5) while $h_{2,j}(k_i) = h_{2,j}(k_{i'})$ for any $i \neq i', j$:
repick $h_{2,j}$ & rehash those l_j items

$\Rightarrow$ no collisions at second level!

Build time: (1) & (2) are $O(n)$. (1.5) & (2.5)?

(2.5): $\Pr_{h_{2,j}}\{h_{2,j}(k_i) = h_{2,j}(k_{i'}) \text{ for some } i \neq i'\}$

$$\leq \underbrace{\sum_{i \neq i'}}_{} \underbrace{\Pr_{h_{2,j}}\{h_{2,j}(k_i) = h_{2,j}(k_{i'})\}}_{\text{by universality}} \quad \leftarrow \text{Union Bound}$$

$$\leq \binom{l_j}{2} \cdot 1/l_j^2$$

$< 1/2$ (Birthday Paradox)

$\Rightarrow$ each trial is like a coin flip, tails $\Rightarrow$ OK
$\Rightarrow E[\#\text{ trials}] \leq 2$
& #trials $= O(\lg n)$ w.h.p. (by Lecture 7)

- Chernoff bound $\Rightarrow l_j = O(\lg n)$ w.h.p.
$\Rightarrow$ each trial $O(\lg n)$ time (also obviously $O(n)$)
- must do this for each j
$\Rightarrow O(n \lg^2 n)$ time w.h.p. (or obviously $O(n^2 \lg n)$)

$$(1.5): \quad E\left[\sum_{j=0}^{m-1} \ell_j^2\right] = E\left[\sum_{i=1}^{n} \sum_{i'=1}^{n} \underbrace{I_{i,i'}}_{\text{indicator rand. var.} = \begin{cases} 1 \text{ if } h_1(k_i) = h_1(k_{i'}) \\ 0 \text{ else} \end{cases}}\right]$$

$$= \sum_{i=1}^{n} \sum_{i'=1}^{n} E[I_{i,i'}] \quad \leftarrow \text{linearity of expectation}$$

$$= \sum_{i=1}^{n} E[I_{i,i}] + 2 \sum_{i \neq i'} E[I_{i,i'}]$$

$$\leq \quad n \quad + 2\binom{n}{2} \cdot 1/m \quad \leftarrow \text{universality}$$

$$= O(n) \quad \text{because} \quad m = \Theta(n)$$

$$\Pr_{h_1}\left\{\sum_{j=0}^{m-1} \ell_j^2 \geq c \cdot n\right\} \leq \frac{E\left[\sum_{j=0}^{m-1} \ell_j^2\right]}{c \cdot n} \quad \Big\} \text{Markov inequality}$$

$$\leq 1/2 \quad \text{for suff. large const. } c$$

$\Rightarrow E[\#\text{ trials}] \leq 2$

& # trials $= O(\lg n)$ w.h.p.

$\Rightarrow$ (1) & (1.5) take $O(n \lg n)$ w.h.p.