Hashing [redacted]

- 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.
  load factor n/m

ASSUMING simple uniform hashing:

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

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
← French 'hacher' = chop up
← Old French 'hache' = axe
(cf. English 'hatchet')
← Vulcan 'la'ash' = axe

Universal hashing:

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

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

- 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

$$E[\#\text{ keys colliding in a slot}] \leq 1 + \underbrace{\alpha}_{n/m}$$

Proof: - consider keys $k_1, k_2, \dots, k_n$

- let $I_{i,j} = \begin{cases} 1 & \text{if } h(k_i) = h(k_j) \\ 0 & \text{else} \end{cases}$ $\boxed{\text{INDICATOR RANDOM VARIABLE}}$

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

$\square$

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

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 $a_{\text{not } d}$

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

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

$$= 1/m \qquad \square$$

Another universal hash family: [CLRS]

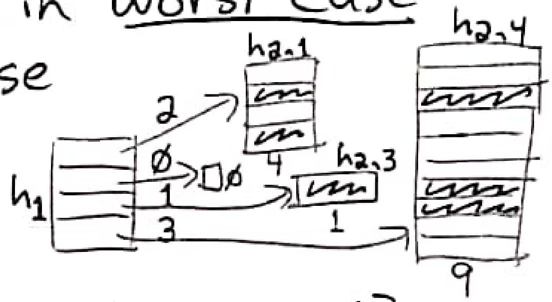
- choose prime $p \geq u$ (once)
- $h_{ab}(k) = [(a \cdot k + b) \bmod p] \bmod m$
- $\mathcal{H} = \{ h_{ab} \mid a, b \in \{0, 1, \ldots, u-1\} \}$

Static dictionary problem: given n keys to store in table, support $\text{Search}(k)$.

no collisions

Perfect hashing: [Fredman, Komlós, Szemerédi 1984]
- 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 $h_1 : \{0, 1, \ldots, u-1\} \to \{0, 1, \ldots, m-1\}$
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 $j \in \{0, 1, \ldots, m-1\}$:
- let $l_j = \#$ items in slot $j = |\{i \mid h(k_i) = j\}|$
- pick $h_{2,j} : \{0, 1, \ldots, u-1\} \to \{0, 1, \ldots, m_j\}$
from a universal hash family
for $l_j^2 \le m_j \le O(l_j^2)$ (e.g. nearby prime)
- replace chain in ① slot j with hashing-with-chaining using $h_{2,j}$

Space $= O(n + \sum_{j=0}^{m-1} l_j^2)$
- to guarantee space $= O(n)$:

constant to be chosen

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

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

- to guarantee $= O(1)$:

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

⇒ no collisions at second level!

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

(2.5): $$\Pr_{h_{2,j}}\{h_{2,j}(k_i) = h_{2,j}(k_{i'}) \text{ for some } i \neq i'\}$$
$$\leq \sum_{i \neq i'} \Pr_{h_{2,j}}\{h_{2,j}(k_i) = h_{2,j}(k_{i'})\} \quad \leftarrow \text{Union Bound}$$
$$\leq \binom{l_j}{2} \cdot 1/l_j^2 \quad \leftarrow \text{by universality}$$
$$< 1/2 \qquad \text{(Birthday Paradox)}$$

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

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

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

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

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

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

$= O(n)$ because $m = \Theta(n)$

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

$\leq 1/2$ 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.