

TODAY: Hashing

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

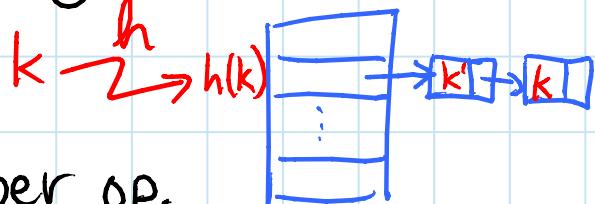
Dictionary problem: Abstract Data Type (ADT)
maintain set of items, each with a key.
Subject to

- 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
& by van Emde Boas

$O(\lg n)$,
 $O(\lg \lg n)$

Hashing (verb from 'hache' = hatchet, Germanic)

- 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:

$$\Pr_{k_1 \neq k_2} \{ h(k_1) = h(k_2) \} = 1/m \quad \text{what you'd expect if totally uniform}$$

- requires assuming input keys are random
- only works in average case
(like Basic Quicksort)

We will remove this unreasonable assumption.

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

$\hookrightarrow n/m$

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

INDICATOR
RANDOM
VARIABLE

$$\begin{aligned}
E[\# \text{ keys colliding with } k_i] &= E\left[\sum_{j=1}^n I_{i,j}\right] \\
&= \sum_{j=1}^n E[I_{i,j}] \quad \leftarrow \text{linearity of expect.} \\
&= \sum_{j \neq i} E[I_{i,j}] + E[I_{i,i}] \\
&= \underbrace{\Pr\{I_{i,j}\}}_{= \Pr\{h(k_i) = h(k_j)\}} \quad \leftarrow \text{indicator rand. var.} \\
&\quad \leftarrow \text{def. of } I_i \\
&= \frac{1}{m} \quad \leftarrow \text{universality} \\
&= n/m + 1
\end{aligned}$$

□

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

Do universal hash families exist? YES:

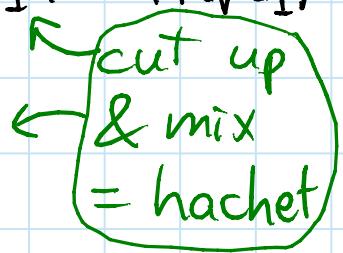
$\mathcal{H} = \{\text{all hash functions}$

$h: \{0, 1, \dots, u-1\} \rightarrow \{0, 1, \dots, n-1\}\}$ is universal

... but this is useless:

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

Dot-product hash family:

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

- $\mathcal{H} = \{ha \mid a \in \{0, 1, \dots, u-1\}\}$

- storing $ha \in \mathcal{H}$ 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 $ha(k)$ takes $O(1)$ time
[$O(\lg u)$ 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 $a \not\in d = \{0, 1, \dots, r-1\} \setminus \{d\}$

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

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

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

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

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

m prime $\Rightarrow \mathbb{Z}_m$ has mult. inverses

$$= \mathbb{E}_{a \not\in d} \left[\Pr_{a_d} \{ a_d = f(k, k', a \not\in d) \} \right]$$

because a_d is independent from $a \not\in d$

$$\left(= \sum_x \Pr_{a \not\in d} \{ a \not\in d = x \} \Pr_{a_d} \{ a_d = f(k, k', x) \} \right)$$

$$= \mathbb{E}_{a \not\in d} [1/m]$$

$$= 1/m$$

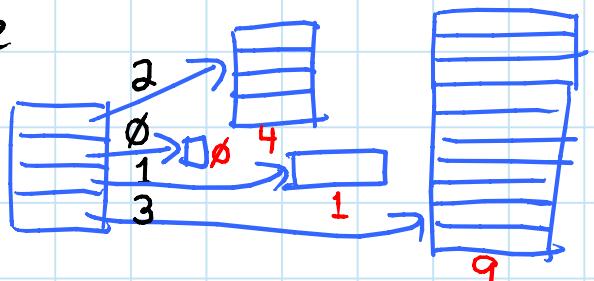
□

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

Perfect hashing: [Fredman, Komlós, Szemerédi 1984]

- polynomial build time w.h.p.
- $O(1)$ time for Search_n in worst case
- $O(n)$ space in worst case

Idea: 2-level hashing



- ① pick $h_1 : \{0, 1, \dots, u-1\} \rightarrow \{0, 1, \dots, m-1\}$ 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 $l_j = \# \text{ items in slot } j = |\{i \mid h(k_i) = j\}|$
 - pick $h_{2,j} : \{0, 1, \dots, u-1\} \rightarrow \{0, 1, \dots, m_j\}$ from a universal hash family for $l_j^2 \leq m_j \leq O(l_j^2)$ (e.g. nearby prime)
 - replace chain in slot j with hashing-with-chaining using $h_{2,j}$

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

- to guarantee $\text{space} = O(n)$:

- ⑤ if $\sum_{j=0}^{m-1} l_j^2 > cn$ then redo step ①

\nearrow constant to be chosen

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

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

⇒ no collisions at second level!

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

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

⇒ each trial is like a coin flip, tails \Rightarrow OK
 $\Rightarrow E[\# \text{ trials}] \leq 2$
& $\# \text{ trials} = O(\lg n)$ w.h.p. (by Lecture 8)

- each trial $O(n)$ time (at most)
 - must do this for each j
- $\Rightarrow O(n^2 \lg n)$ time w.h.p.

$$\textcircled{1.5}: E\left[\sum_{j=0}^{m-1} l_j^2\right] = E\left[\sum_{i=1}^n \sum_{i'=1}^n I_{i,i'}\right]$$

indicator rand. var. = $\begin{cases} 1 & \text{if } h_1(k_i) = h_1(k_{i'}) \\ 0 & \text{else} \end{cases}$

$$= \sum_{i=1}^n \sum_{i'=1}^n E[I_{i,i'}] \quad (\text{linearity of expect.})$$

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

$$= n + 2 \binom{n}{2} \cdot \frac{1}{m}$$

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

$$\Pr_{h_1} \left\{ \sum_{j=0}^{m-1} l_j^2 \geq c \cdot n \right\} \leq \frac{E\left[\sum_{j=0}^{m-1} l_j^2\right]}{c \cdot n} \quad \left. \begin{array}{l} \text{Markov} \\ \text{inequality} \end{array} \right\}$$

$\leq 1/2$ for suff. large const. c

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

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