Hashing

Dictionary w/o ordering

- Insert (S, x) : $S := S \cup \{x\}$
- Delete (S, x) : $S := S \setminus \{x\}$

} dynamic set

- Search (S, k) : $:= \begin{cases} x & \text{if } x \in S,\ x.key = k \\ nil & \text{otherwise} \end{cases}$

Example: Phone company

Task: Find caller based on number

Customer record

678 5321	} key
Paul	} satellite data
Egger	

	Linked list	RB-tree	direct access table (array)
Insert	O(1)	O(log n)	O(1)
Delete	O(1)	O(log n)	O(1)
Search	O(n)	O(log n)	O(1)

↑ Add after expl.

Direct access table

$U = \{0, \ldots, m-1\}$ universe of keys

Assume: $x \neq y \Rightarrow x.key \neq y.key$, f.a. $x, y \in S$

(distinct elements have distinct keys)

Array $T[0..m-1]$ represents dyn. set S:

$$T[k] := \begin{cases} x & \text{if } x.key = k \\ nil & \text{otherwise} \end{cases}$$

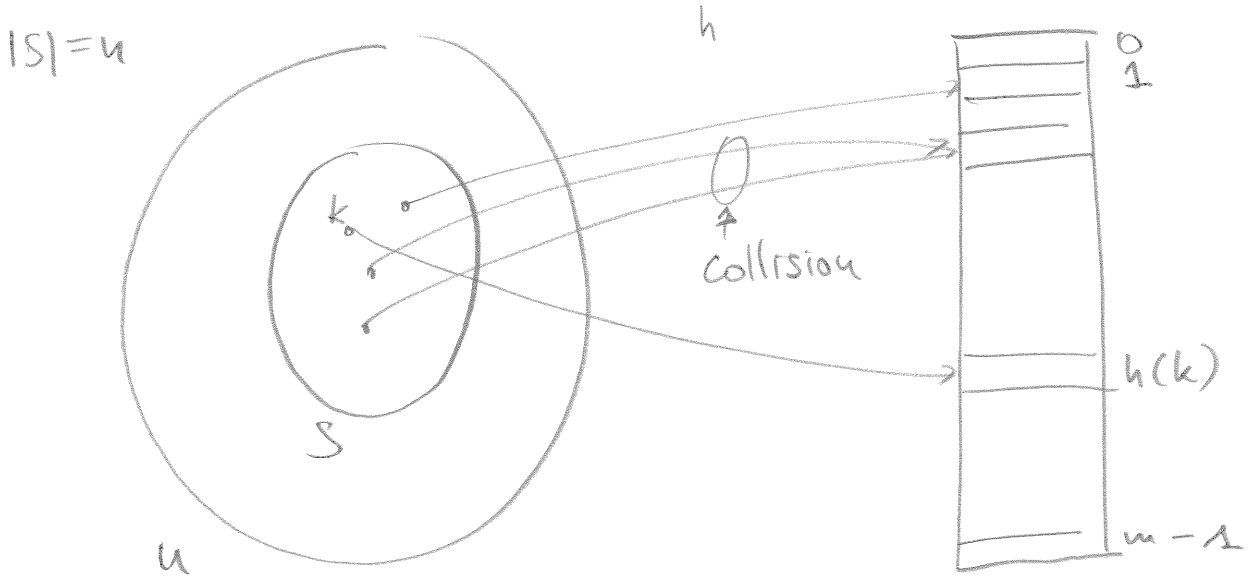
Ops: $O(1)$ time (+)

Bad if $m \gg n$: many empty slots (−)

Can we keep the good side and get rid of the bad?

Hashing

Hash function maps keys "randomly" into slots of table T

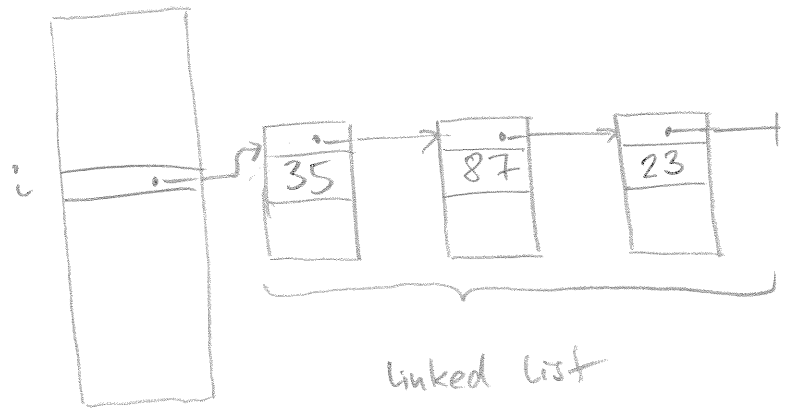


2 Questions:

- h?
- collisions?

Collision Resolution by Chaining

$h(35) = h(87) = h(23) = i$



Analysis:

- Worst case: all keys of S hash to same slot

  Time : $\Theta(n)$ for $n = |S|$

- Avg case: Assume

  Simple Uniform Hashing:
  - all slots are equally likely for each key
  - hash values for any two keys are independent of each other

  $\Rightarrow \quad P(h(k_1) = h(k_2)) = \frac{1}{m}$

  $P(h(k) = i) = \frac{1}{m}$ $\quad 0 \le i < m$ fixed

Assume also: T has m slots, n elements (= keys)

$\Rightarrow$ Expected search time

- unsuccessful: $\Theta(1) + \Theta\left(\frac{n}{m}\right)$
- successful: $\Theta(1) + \Theta\left(\frac{1}{2} \cdot \frac{n}{m}\right)$

hash & list access

list search

Def^n: Load factor of T is $\alpha := \frac{n}{m}$

$\Rightarrow$ Expected search time $= \Theta(1+\alpha)$

Thus, $\alpha = O(1) \Rightarrow$ search time $= O(1)$

Note: $\alpha = O(1) \Leftrightarrow \underbrace{n = O(m)}$

elements controlled by # slots

Hash function: h should

- distribute keys uniformly over slots
- still be uniform, if keys have pattern (e.g., all keys even)

Division method:

$$h(k) = k \bmod m$$

Examples

- $m = 2^r$
  $\Rightarrow$ $h(k)$ depends only on last r bits of k

- $m = 2m_0$
  all k's even $\Rightarrow$ $h(k)$ always even

- $m = q \cdot p$

$$k = p, 2p, 3p, \ldots, q \cdot p, (q+1) \cdot p$$
$$h(k) = p, 2p, 3p, \ldots, 0, p$$

$\Rightarrow$ short cycles for multiples of p (or q)

Better: choose m as prime

Downside: division can be costly

Multiplication method:

$m = 2^r$, $w =$ #bits in computer word

(e.g., $w = 32, 64$)

A odd integer, $2^{w-1} < A < 2^w$

$\Rightarrow 2^{-1} < A \cdot 2^{-w} < 1$

$\bar{A}$:"

Means: First and last bit of A are 1

$$h(k) := (k \cdot A \mod 2^w) \text{ rsh } (w-r)$$

rsh ⇝ bit-wise right shift

- A should not be close to 2^{w-1} or 2^w
- Fast method: mult mod 2^w faster than division

(mult mod 2^w: ignore higher order bits)

rsh fast

Example: $m = 8 = 2^3$, $w = 7$

$$
\begin{array}{rl}
. 1011001 & = A \\
1101011 & = k \quad (= 107_{10}) \\
\hline
\underbrace{1001010}_{\text{ignore, due to mod } 2^w}.\underbrace{0}\underbrace{1101011}_{\text{rsh } (w-r)} & \\
\qquad\qquad\quad | & \\
\qquad\quad h(k) &
\end{array}
$$

Equivalent definition: Let $1/2 < \bar{A} < 1$

$$h(k) := \lfloor (k \cdot \bar{A} \bmod 1) * 2^r \rfloor$$

where

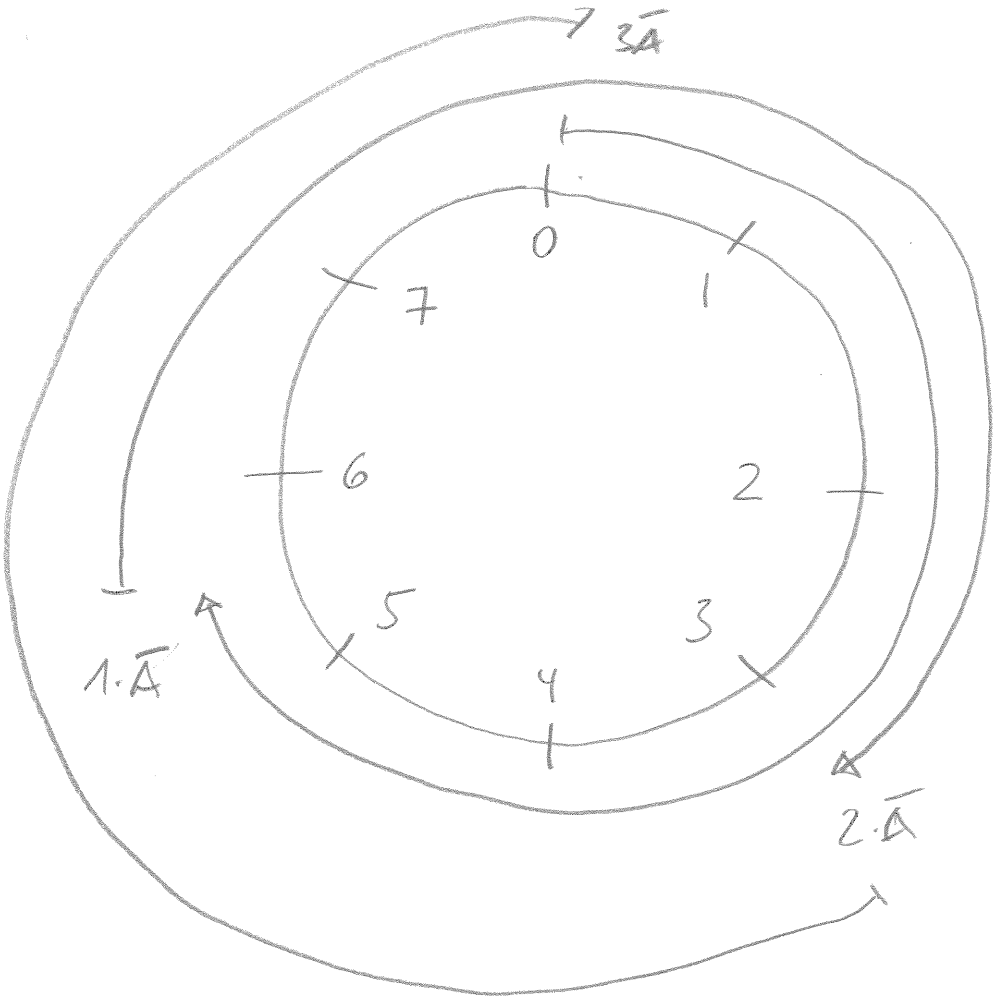
- $x \bmod 1$ = the fractional part of x,
  e.g. $13.28 \bmod 1 = .28$
- $\lfloor y \rfloor$ = floor y, i.e., y rounded down

Cheap: shifts are cheap, mult. & shift

Intuition: $\bar{A} = 0.10111001$

$$\Rightarrow \bar{A} \approx \frac{1}{2} + \frac{1}{8} + \frac{1}{16} = \frac{5.5}{8}$$

Table indices are $0, \ldots, 7 \; (= 2^3 - 1)$



$h(1) = 5$

$h(2) = 3$

$h(3) = 0$

$h(1)$: wheel stops between 5 and 6

$h(2)$: wheel stops between 3 and 4

$h(3)$: wheel stops between 0 and 1