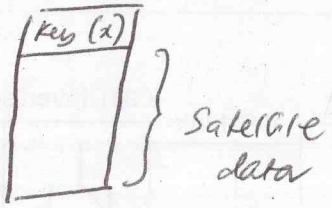
Symbol-table

Table S holding n records

Each record x



Operations

- Insert (S, x) — $S \leftarrow S \cup \{x\}$ } Dynamic
- Delete (S, x) — $S \leftarrow S - \{x\}$ }
- Search (S, k) — Returns x s.t key $(x) = k$ or returns nil if no such k

Direct Access Table

Suppose keys are drawn from $U = \{0, 1, \ldots, m-1\}$

Keys are distinct (Assumption)

Set up an array $T[0, \ldots m-1]$ to represent the dynamic set S

$$T[k] = \begin{cases} x & \text{if } x \in S \text{ \& key}(x) = k \\ \text{nil} & \text{otherwise} \end{cases}$$

Ops take $O(1)$ time

- m can be huge → huge table

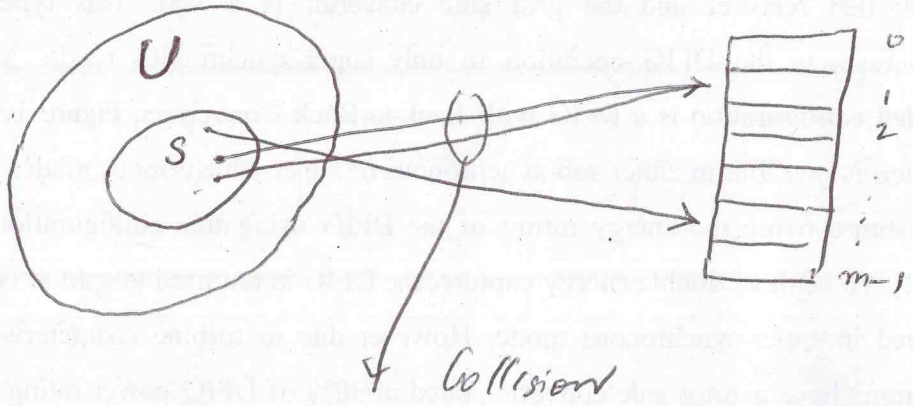
  Think of 64 bit values but S is small

  $m = 2^{64}$

- Think of keys as character strings → sparse table

Hashing

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

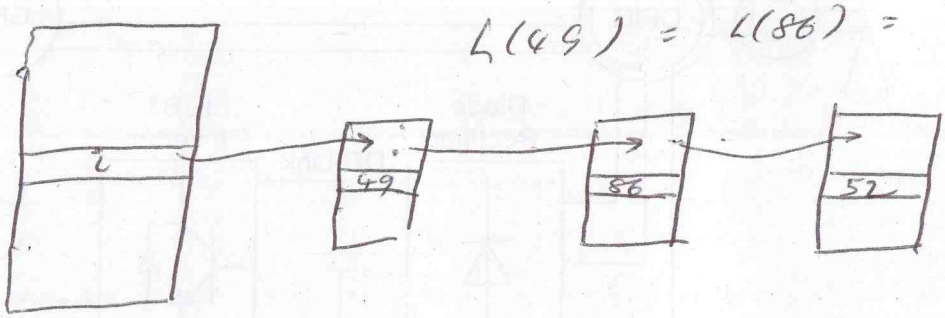


when a record to be inserted maps to an already occupied slot, a <u>collision</u> occurs.

Resolving Collisions by Chaining

<u>Idea</u>: Link records in the same slot into a list

Eg $h(49) = h(86) = h(52) = i$



<u>Analysis</u>

<u>Worst Case</u>: Every key hashes to same slot

Access time $\Theta(n)$ time if $|S| = n$

<u>Average case</u>: assumption of <u>simple uniform hashing</u>

- Each key $k \in S$ is equally likely to hashed to any slot in T independent of where other keys are hashed

- Odds that two keys hash to some slot $\rightarrow 1/m$

  one key hash to slot is $\rightarrow 1/m$

Define The load factor of a hash table with n keys and m slots to be

$\alpha = n/m$ = # average keys per slot

Expected unsuccessful search time

$= \Theta(1+\alpha)$

↑ hash & access slot ↑ search list

Exp search time $= \Theta(1)$ if $\alpha = O(1)$

(i.e.) $n = O(m)$

Work: running time depends on "load factor"

Choosing a hash function

- Should distribute keys uniformly into slots
- Regularity into key distributions should not affect uniformity

Division method

$h(k) = k \bmod m$

- Don't pick m with a small divisor d

Ex: $d = 2$ & all keys even

$\Rightarrow$ only even slots used

Ex: $m = 2^r$ hash does not depend on all bits of k

Good heuristic: Pick m to be a prime not too close to a power of 2 or 10.

MAD

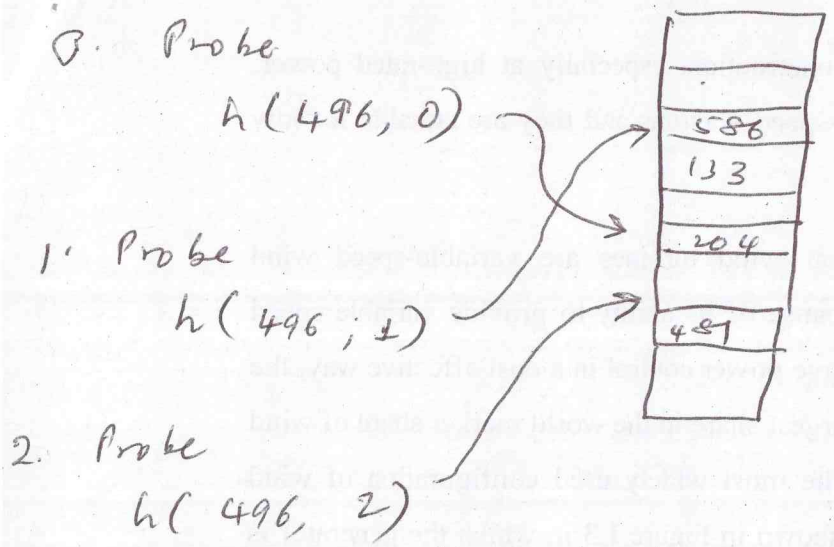
$$h(k) = ak + b \bmod N$$

N → prime $a \bmod N \neq 0$

a and b random.

Resolving collisions by open addressing

- Probe table systematically until empty slot is found
- $h : U \times \{0, 1, \ldots, m-1\} \rightarrow \{0, 1, \ldots m-1\}$ (slot)
  - U ↓ universe of keys
  - $\{0, 1, \ldots, m-1\}$ → probe #
- Prob seq should be permutation
- Table may fill up ($n \leq m$)
- Deletion is hard



Search : same same probe seq

- successful finds record
- unsuccessful finds nil

Probe seq

- Linear $h(k, i) = (h(k, 0) + i) \bmod m$
- primary clustering - long runs of filled slots
- Quadratic
- $h(k, i) = (h_1(k) + i\, h_2(k)) \bmod m$
- Pick $m = 2^r$ and $h_2(k)$ odd