HASHING WITH CHAINING

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

- insert(item) (overwrite any existing key)
- delete(item)
- search(key): return item with given key or report doesn't exist

O(lgn) via AVL → O(1)

Python: dict

D[key] ~ search

D[key] = val ~ insert

del D[key] ~ delete

-> item = (key, value)

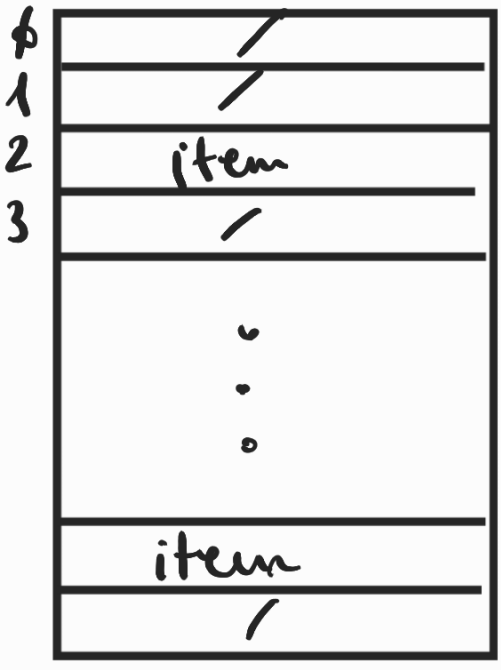
Motivation:

- doc dist
- database
- compilers & interpreters
- network router server
- substring search

- string commonalities
- file synchronization
- cryptography

Simple approach:

Direct-access table



Badness:

1) keys may not be nonneg. integers
2) gigantic memory hog

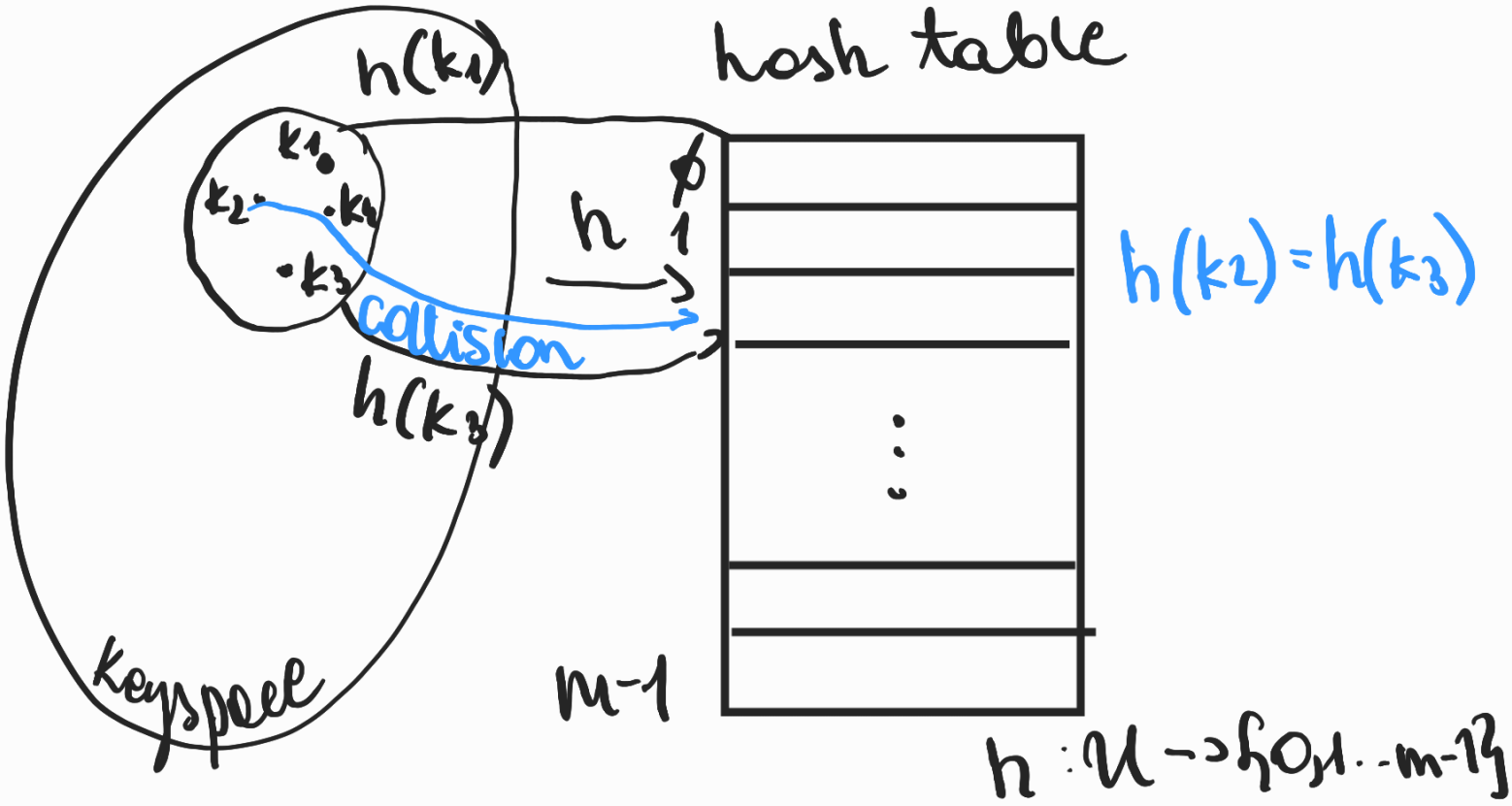
Solution to (1): prehash

- maps keys to nonneg. integers
- in theory keys are finite & discrete (string of bits)

- in Python hash(x) is the prehash of x
- hash('\0B') = hash('\0\0C') = 64
  ideally: hash(x) = hash(y) <=> x = y

Solution to (2): hashing

- reduce universe U of all keys (integers) down to reasonable size m for table



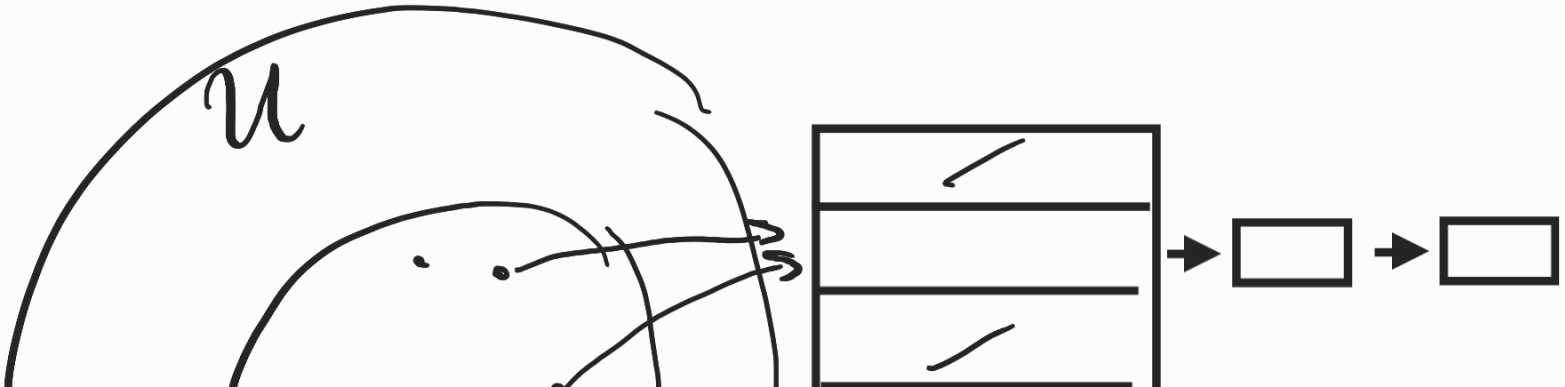
- idea: $m = \Theta(n)$

↓

keys in dictionary

DEALING WITH COLLISIONS

CHAINING: linked list of colliding elements in each slot of hash table



worst case: $\Theta(n)$ ← if we are unlucky with hash function h

Simple uniform hashing:

each key is equally likely to be hashed to any slot of the table, dependent of where other keys hashing.

Analysis:

- expected length of chain for n keys and m slots

$$= \underbrace{1/m + 1/m + \dots + 1/m}_{n} = n/m$$

$$\frac{n}{m} = \alpha = \frac{\text{load}}{\text{factor}}$$

$$= \Theta(1) \text{ if } m = \Theta(n)$$

$\Rightarrow$ running time $= O(1 + |chain|)$
$= O(1 + \alpha)$

Hash functions:

(1) division method:

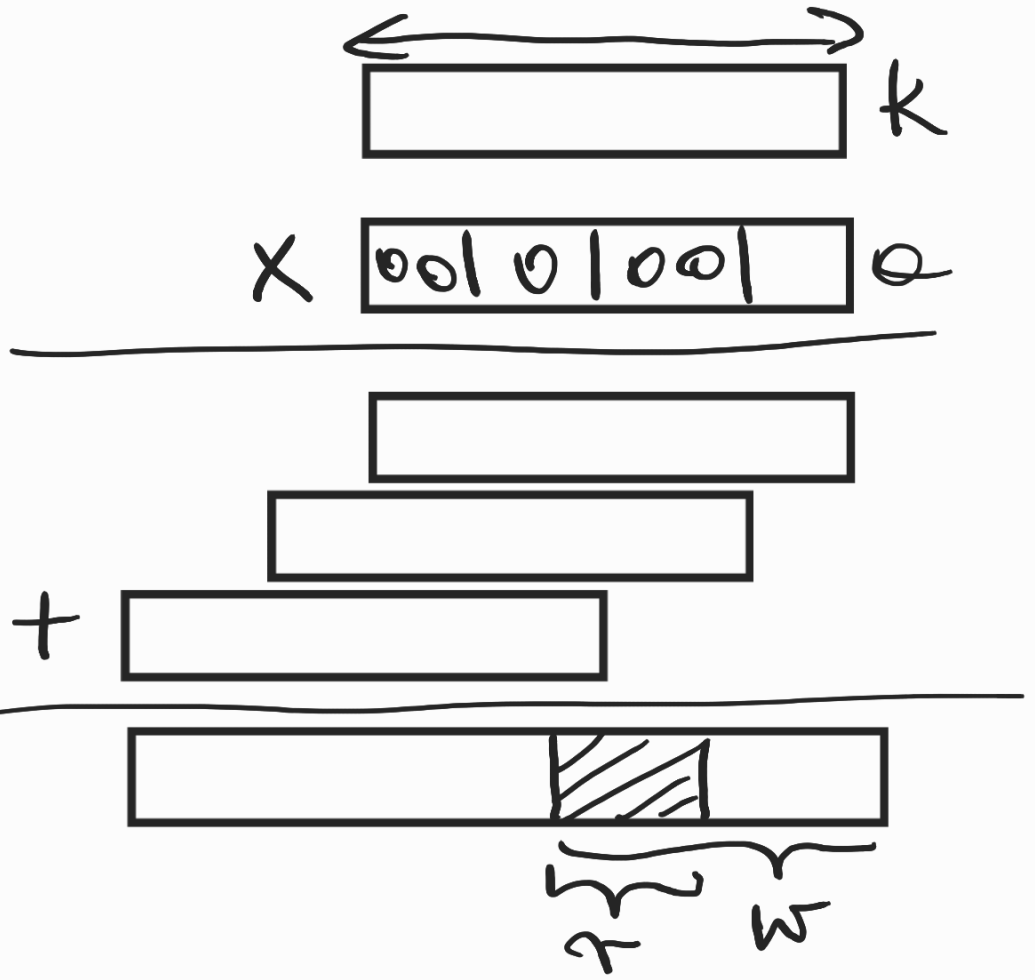
$h(k) = k \bmod m$

(2) multiplication method:

$h(k) = [(a \cdot k) \bmod 2^w] >> (w - r)$

$\hookrightarrow w$ bits

$m = 2^r$



(3) Universal hashing:

$$h(k) = [(ak + b) \bmod p] \bmod m$$

a: random $\in \{0, \ldots p-1\}$

b: random $\in \{0, \ldots p-1\}$

p: prime $> |U|$

for worst-case keys $k_1 \neq k_2$:

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