A. insert(key, value) $\Rightarrow$ A[h(key)] = value

v = A. retrieve(key) $\Rightarrow$ v = A[h(key)]

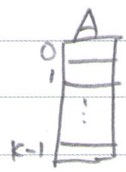
bucket array   hash function

* if {keys} $\xLeftrightarrow[\text{one-to-one}]{}$ {0, 1, 2, ⋯, K−1}

"perfect hashing"

* what if not ?

e.g. # keys > K



$\Rightarrow$ two keys of the same h(·)

$\Rightarrow$ collision !

* e.g. h(str) = str[0] − 'a' ; 26 possibilities

h("act") = h("apple")

$h(str) = \sum_i (str[i] - 'a')$ $\boxed{2^{32}}$? possibilities

not uniform

$h(str) = \sum_i (str[i] - 'a') \% K$ K possibilities

"stop" "pots" "spot"

$\Rightarrow$ easy collision

* how to hash

h(key) = compress (Hashcode(key))

- sum [not using position info]
- polynomial (like shift-left)
- cyclic shift

- division (% K)
- MAD ((a·□+b) % K)

* hash table of K entries
  after n keys

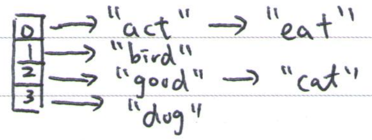
if $\frac{n}{K}$ large $\Rightarrow$ hash won't work
load factor

hash non-uniform $\Rightarrow$ $\frac{n}{K_{eff}}$ large

* idea: increase K when $\frac{n}{K}$ large
* naive
  1. set $K^{new} = 2K$
  2. change $h(key)$ to range $\{0, \cdots, 2K-1\}$
  3. rebuild w/ $O(n)$ if insert is $O(1)$
  - cannot do often $(\frac{n}{K} > \theta)$
  - long waiting

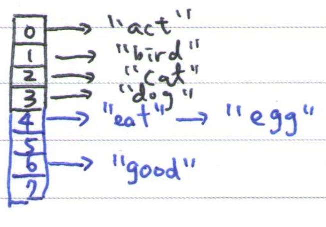
* lazy approach
  1. set $K^{new} = 2K$ (use one more bit of $h(\cdot)$)
  2. change $h(key)$
  3. rebuild only the overflow entry $O(K) + O(\frac{n}{K})$

e.g. hashing w/ chaining of length 2
$h(key) = (key[0] - \text{'a'}) \% K$

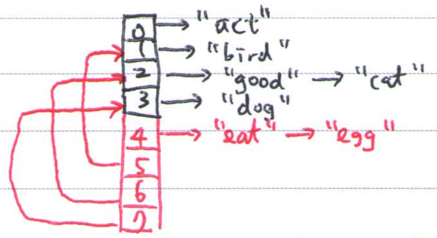


insert "egg"

naive



lazy (directory extension)



* hash table of K entries
after n keys

if $\frac{n}{K}$ large $\Rightarrow$ hash won't work

$\frac{n}{K}$: load factor

hash non-uniform $\Rightarrow$ $\frac{n}{K_{eff}}$ large

* idea: increase K when $\frac{n}{K}$ large

* naive

① set $K^{new} = 2K$

② change h(key) to range $\{0, \cdots, 2K-1\}$

③ rebuild w/ $O(n)$ if insert is $O(1)$

- cannot do often $\left(\frac{n}{K} > \theta\right)$
- long waiting

* lazy approach

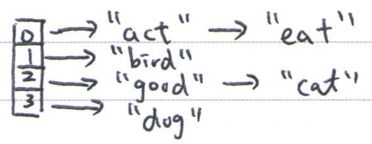
① set $K^{new} = 2K$ (use one more bit of $h(\cdot)$)

② change h(key)

③ rebuild only the overflow entry $O(K) + O\left(\frac{n}{K}\right)$

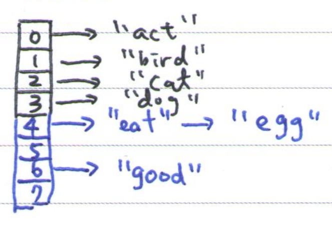
e.g. hashing w/ chaining of length 2

$h(\text{key}) = (\text{key}[0] - \text{'a'}) \% K$

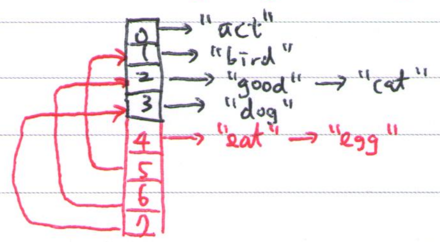


insert "egg"

naive



lazy (directory extension)



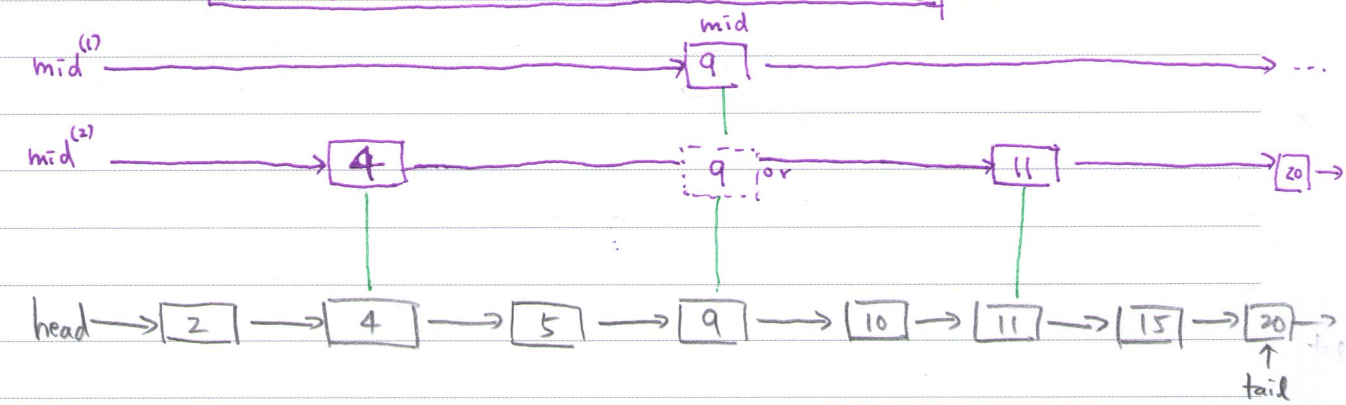
* hashing : extending array to do map/dictionary

* how to extend list to do map/dictionary
  - unordered : fast insertion , slow search
  - ordered : slow insertion , slow search

* why slow search ? sequential

  can we do binary search on ordered linked lists ?

  YES if mid node can be found quickly



* skip list = list + ... + list of quad + list of mid

search for 10
* (head , tail) = (2, 20) * mid = 9
* (head , tail) = (9 , 20) * mid = 11
* (head , tail) = (9, 11) * mid = 10 found !

search for 14

(2, 20)

(9, 20)

(11, 20)

(11, 15) fail !

* but how to insert fast ? "cannot work if too strict"

  probabilistic : a node "survives" to the upper list w/ prob 1/2