* how to resolve collision during insertion ?
  ① fixed (unordered) array per bucket
     [can still overflow]
  ② linked list per bucket
     chaining : don't want long chains
  ③ other data structure [usually called secondary] per bucket
     [more complicated]
  ④ use other empty buckets
     open addressing

* open addressing
  a[key] = value
  x = a[key]

  ① insert (key, value) to check $h_0(key) = h(key)$
  ② if fail, insert to check $h_1(key)$
  ③ if fail, insert to check $h_2(key)$
  ⋮
  (m-1) if fail, insert to check $h_m(key)$
  (m) declare failure
     not found

* (A) linear probing :

$$h_i(key) = (h_{i-1}(key) + 1) \% K$$
$$= (h_0(key) + i) \% K$$

primary clustering

$m = K-1$

(B) quadratic probing :

$$h_i(key) = (h_0(key) + i^2) \% K$$

secondary clustering

$m = ?$

(C) double hashing :

$$h_i(key) = (h_0(key) + i\,\tilde{h}(key)) \% K$$

$\tilde{h}(key) > 0$

$m = ?$

* hash table of K entries after n keys

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

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

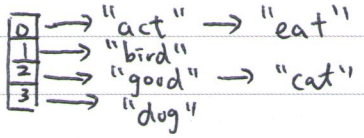
hash non-uniform $\Rightarrow$ $\frac{n}{K_{eff}}$ large

* 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 ($\frac{n}{K} > \theta$)
  - 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 $O(K) + O(\frac{n}{K})$

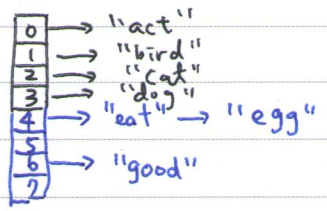
e.g. hashing w/ chaining of length 2

h(key) = (key[0] - 'a') % K



insert "egg"

naive



lazy (directory extension)