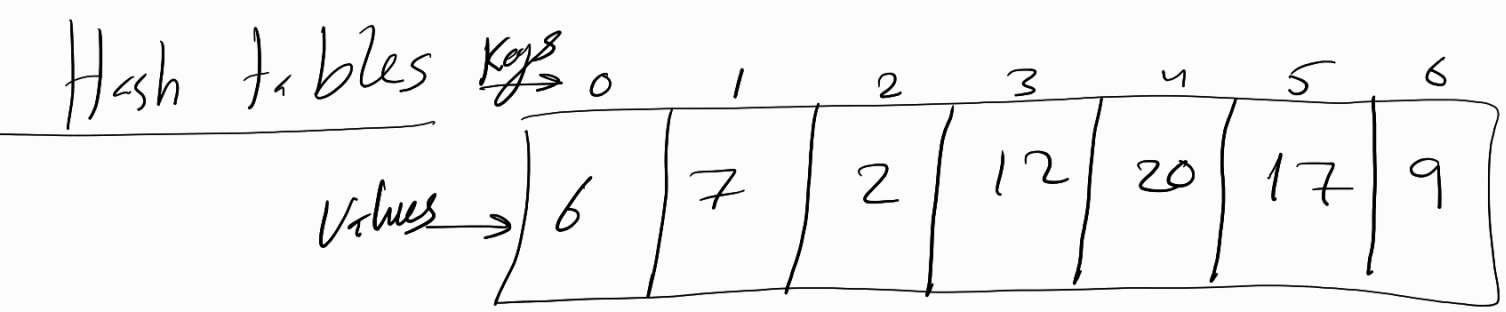
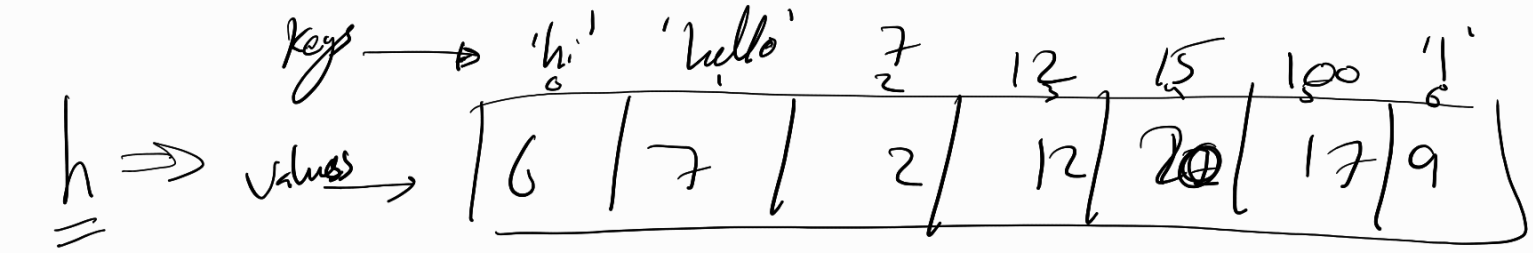
Hash tables





h('hello') ⇒ 7

(key, value) ⇒ Python dict, C++ map, JS object, Json

---

hash table store (key, value)

↓

hash function

↓

index

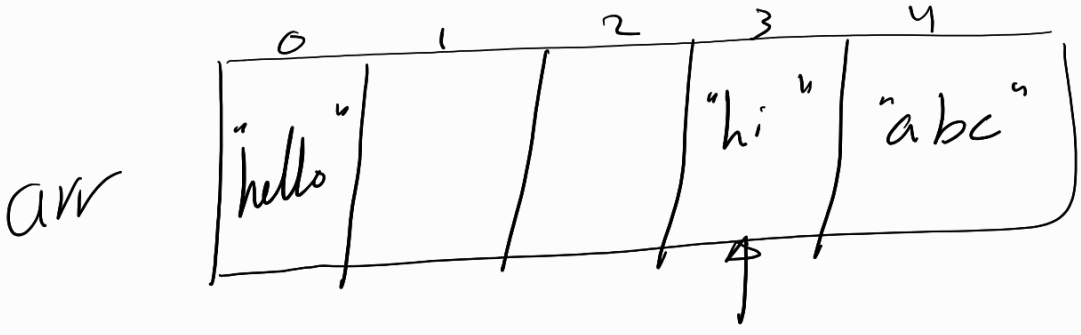
array[ ]

O(1)

Hash table consists of :-

1. array : to store data
2. Hash function :- keys → (hash func.) → index.

steps :- - convert key to integer.
- using hash values
hash(5) → 5
hash('hi') → integer.
- extract index from integer.
- integer % size



(4, "abc")
(75, "hello")
(23, "hi")
(13, 27)

```
def h(k):
    return k % 5
```

h(4) = 4
arr[4] = "abc"
h(75) = 0
arr[0] = "hello"
h(23) = 3
arr[3] = "hi"

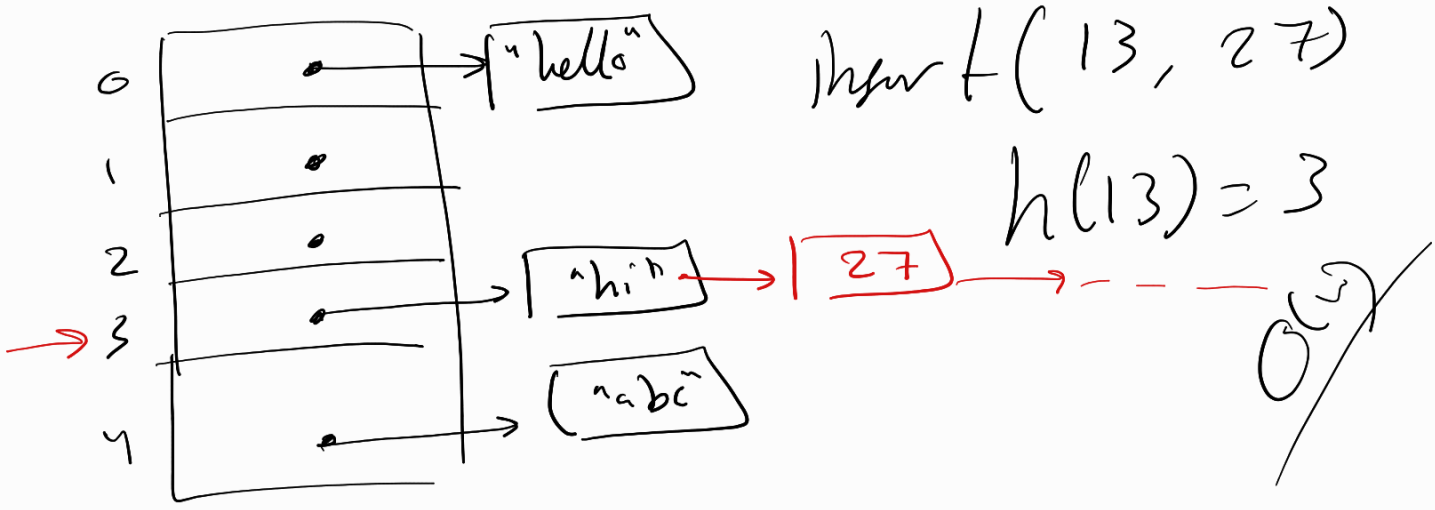
h(13) = 3
arr[3] = 27

---

③ + ④ Collision resolution ⟹ different types of hash table

Buckets

Seperate Chaining



insert(13, 27)

$h(13) = 3$

$O(1)$

hash function Char.

① minimize collisions $h(k) = k \% 10$

$h(k) = k \% 11$ (11 Prime)

$h(1001) = 0$

$h(1011) = 11$

$h(1021) = 9$

$h(1001) = 1$

$h(1011) = 1$

$h(1021) = 1$

② fast to compute.

sum = 0

for each char → ch:

sum += ascii(ch)

$O(L)$

③ distribute keys uniformly

✗ $h(k) = floor(k/10) \% 10$

indices [0 ... 9] ✓

$k \Rightarrow [0 \ldots 49] \rightarrow [0 \ldots 4]$

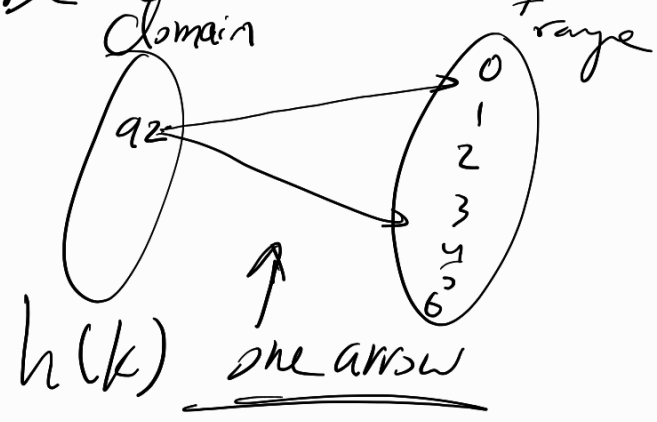
In a hash tbl of size 10

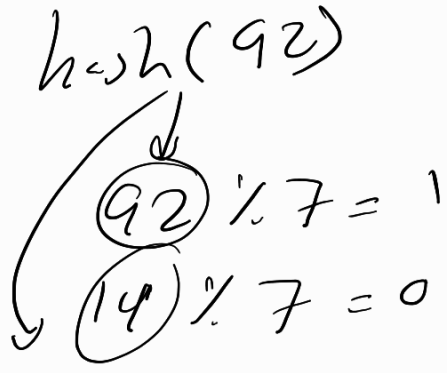
k [0 .. 99]

indices [5 --- 9] → are empty

→ $h(k) = k \% 10$ ✓

④. be deterministic





---

types of hashy:

① seperate chaining
↳ using Linked Lists or dynamic arrays

② open addressing:- Probing:-

ⓐ. Linear probing:

$h(k) \rightarrow$ coll. $\rightarrow h(k)+1$
$+2$
$+3$
⋮
empty slot

ⓑ Quadratic Probing

$h(k) \rightarrow$ coll. $\rightarrow h(k)+1^2$
$+2^2$
$+3^2$
$\% size$

© double probing.

$h_1(k)$ $h_2(k)$ → collision → linear probing

↳ call

③. rehashing.

3.1 using one of the prev. mechanisms

3.2 if the load factor >= threshold

load factor = N/M

N: number of active slots

M: size of array

threshold: 0.7, 0.5, !

then rehash.

⟹ ① create a new array of size next prime > 2M

② insert old values into new array.

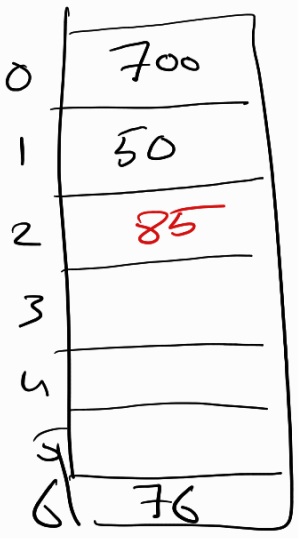
Example # $M = 7$ $h(k) = k \% M$

insert(50) $\Rightarrow$ 1 $\rightarrow$ $1/7 \approx 0.14$

insert(700) $\Rightarrow$ 0 $\rightarrow$ $2/7 \approx 0.28$

insert(76) $\Rightarrow$ 6 $\rightarrow$ $3/7 \approx 0.43$

insert(85) $\Rightarrow$ 1 $\rightarrow$ $4/7 \approx 0.57$



rehashy $\Rightarrow$ $7 * 2 = 14$

New size $M = 17$

insert(700) $\Rightarrow$ $700 \% 17 \rightarrow 3 \rightarrow 1/17 \approx 0.06$

insert(50) $\Rightarrow$ $50 \% 17 \rightarrow 16 \rightarrow 2/17 \approx 0.1$

insert(85) $\Rightarrow$ $85 \% 17 \rightarrow 0 \rightarrow 3/17 \approx 0.17$

insert(76) $\Rightarrow$ $76 \% 17 \rightarrow 8 \rightarrow 4/17 \approx 0.23$

insert(92) $\Rightarrow$ $\rightarrow 7 \rightarrow 5/17 \approx 0.3$

insert(77) $\Rightarrow$ $\rightarrow 5 \rightarrow 6/17 \approx 0.35$

insert(101) $\Rightarrow$ $\rightarrow 16 \rightarrow 7/17 \approx 0.41$