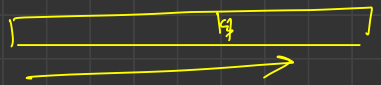
Hashing :

hashing is a technique used for searching.

① Linear Search



Searching
TC: O(N)

② Binary Search
↳ In sorted array.
searching O(log N).

③ Hashing ⟶ searching $\frac{TC}{O(1)}$

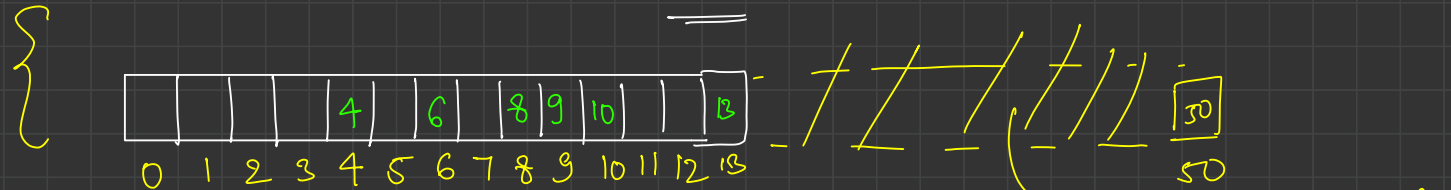
ele → (8, 9, 13, 6, 4, 10), 50

↳ data structure

↳ store this elements

when try to search any of them,

then TC : O(1)

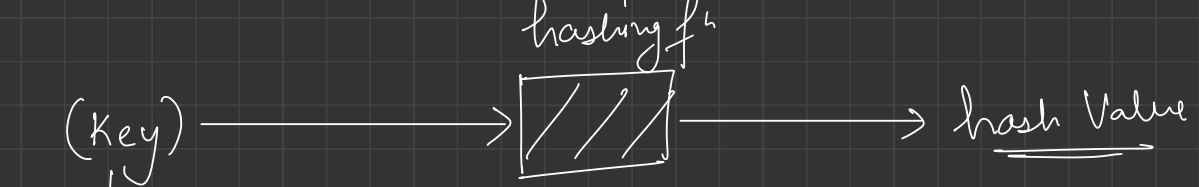


search(8) ⇝ { if (arr[8] != null) return true; } → O(1)

search(2) ⇝ else return false → O(1)

(very high memory is required to make this soln work)

To overcome this issue of high memory, hashing techniques were used!



eg Key = "pipik"
↓ ↓
assci add them

↓
the things we (have to store)

hashing fn

Step1 $g(x) = K$ — key → integer value.

↓
string, float, character, etc

Step2 $h(K) = y$ → generating hash value

↳ passing integer value

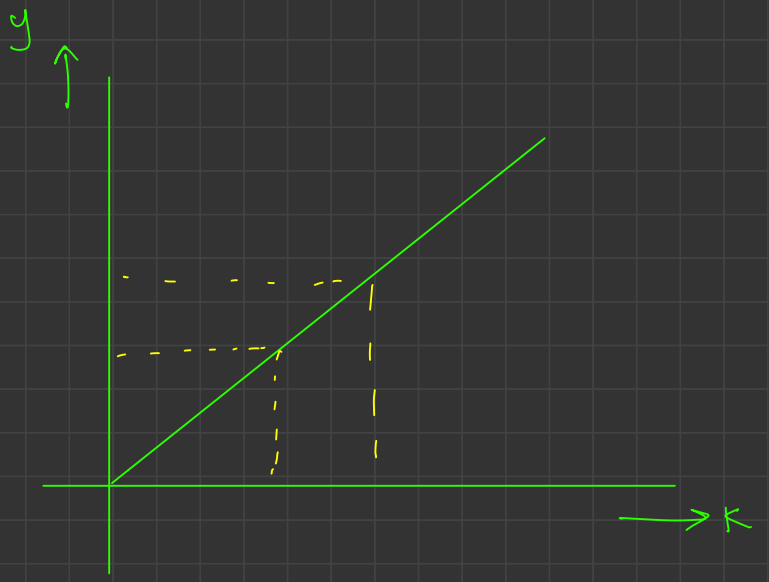
Key Space

8
3
13
6
4
10

hashing fn

$h(K) = K$

$h(8) = 8$
$h(3) = 3$
$h(13) = 13$
$h(6) = 6$
$h(4) = 4$
$h(10) = 10$



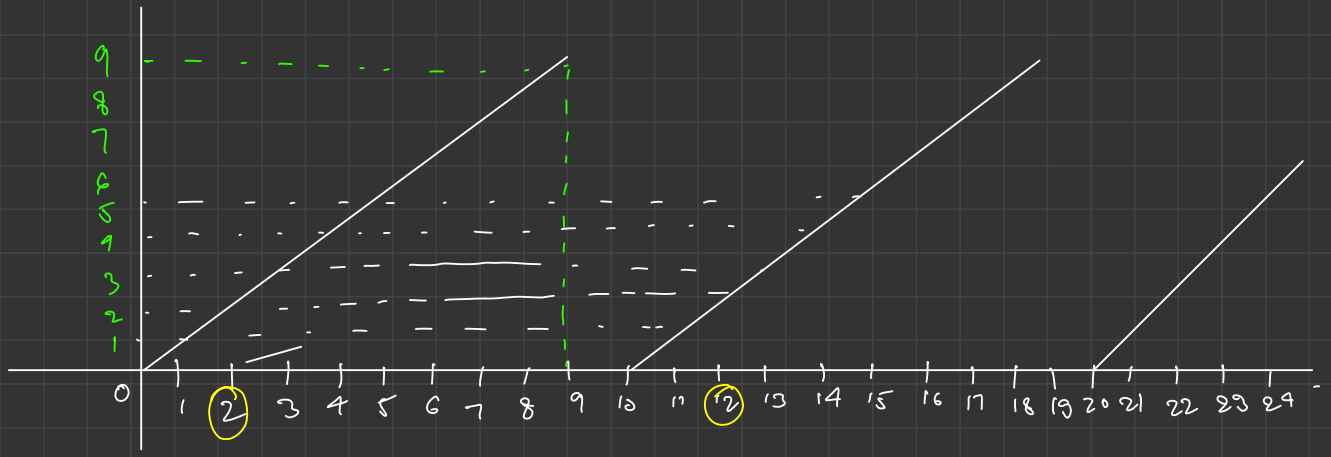
for every value of K we have a unique y

(one to one mapping)

not useful

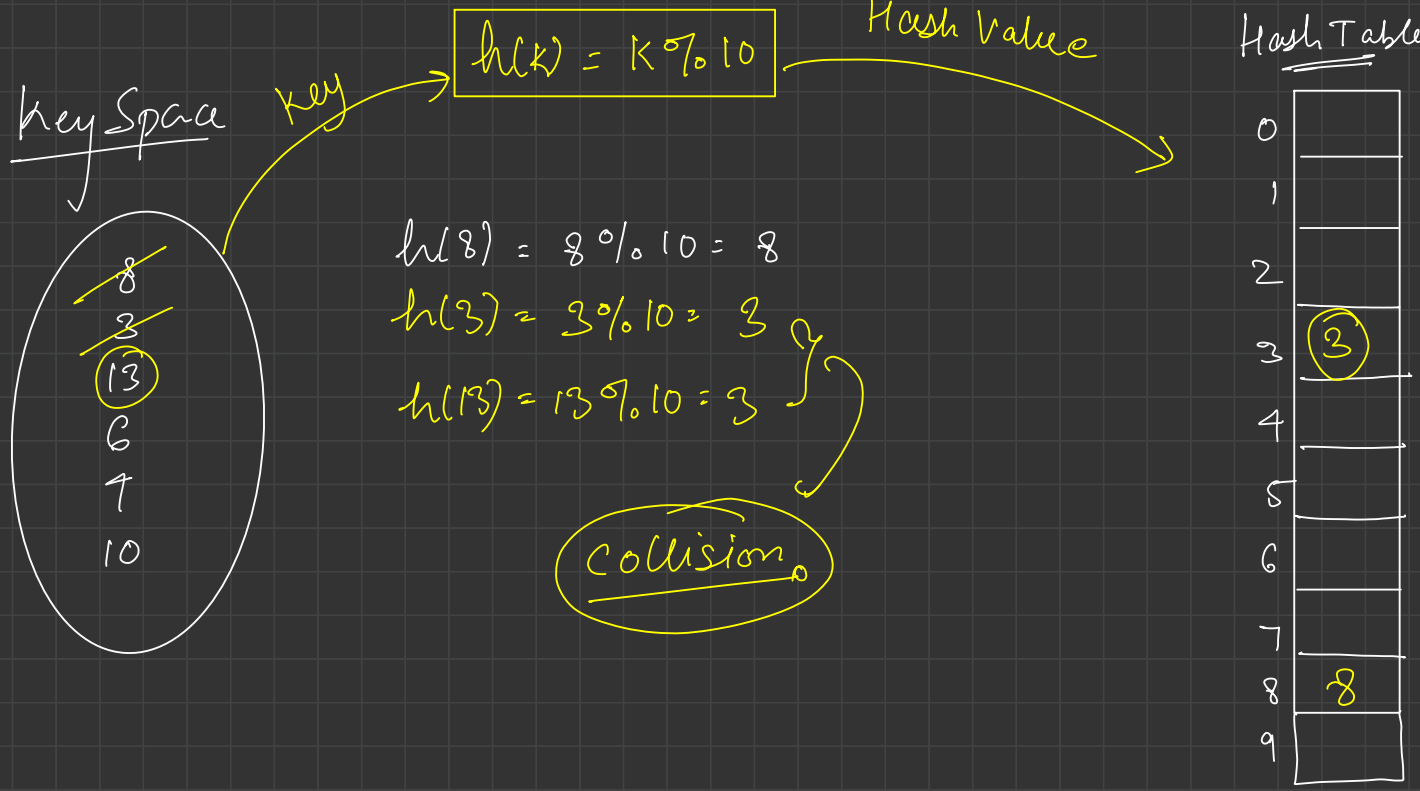
Many to one Relation

$$h(k) = k \% 10$$



$h(2) = 2 \% 10 = 2$ → hash value

$h(12) = 12 \% 10 = 2$ → hash value


Key Space
Key
h(k) = k%10
Hash Value
Hash Table
8
3
13
6
4
10
h(8) = 8%10 = 8
h(3) = 3%10 = 3
h(13) = 13%10 = 3
Collision
0
1
2
3
4
5
6
7
8
9
3
8

Methods to remove Collision

- open Hashing
  - → chaining
- closed Hashing
  - → linear Probing
  - → Quadratic Probing

Chaining

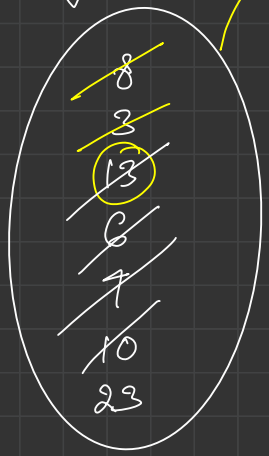
Key Space

Key

$h(k) = k \% 10$

Hash value

Hash Table



$h(8) = 8 \% 10 = 8$

$h(3) = 3 \% 10 = 3$

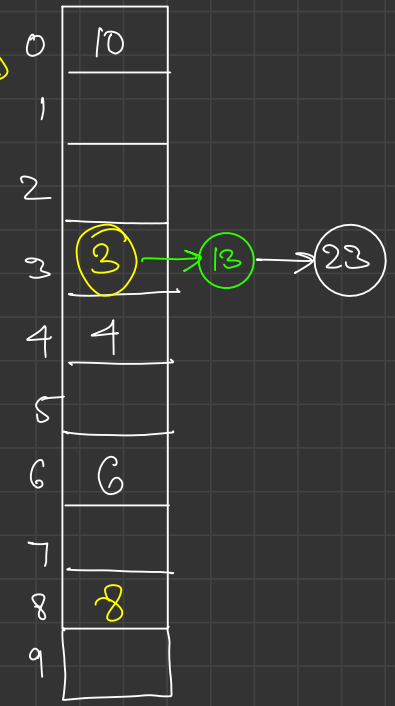
$h(13) = 13 \% 10 = 3$

$h(6) = 6 \% 10 = 6$

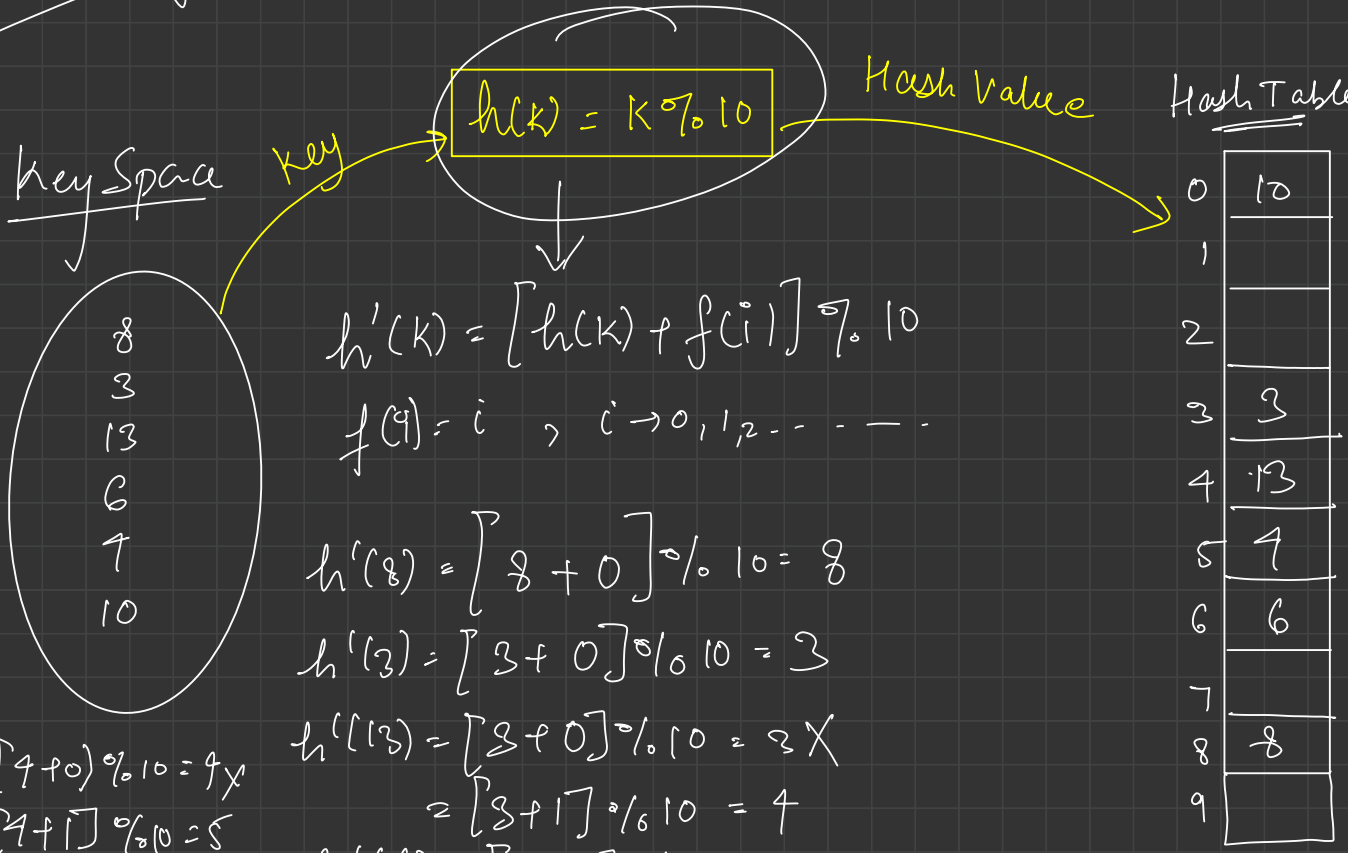
$h(4) = 4 \% 10 = 4$

$h(10) = 10 \% 10 = 0$

$h(23) = 23 \% 10 = 3$



Linear Probing



$h'(k) = [h(k) + f(i)] \% 10$

$f(i) = i \;,\; i \to 0, 1, 2 \ldots$

$h'(8) = [8 + 0] \% 10 = 8$

$h'(3) = [3 + 0] \% 10 = 3$

$h'(13) = [3 + 0] \% 10 = 3$ X

$= [3 + 1] \% 10 = 4$

$h'(6) = [6 + 0] \% 10 = 6$

$h'(4) = [4 + 0] \% 10 = 4$ X

$= [4 + 1] \% 10 = 5$

$h'(10) = [0 + 0] \% 10 = 0$

Quadratic Probing

Key Space

3
13
23

Key → $h(k) = k \% 10$ → Hash Value → Hash Table

$h'(k) = [h(k) + f(i)] \% 10$

$f(i) = i^2$, $i = 0, 1, 2, 3, 4 \ldots$

$h'(3) = [3 + 0] \% 10 = 3$

$h'(13) = [3 + 0] \% 10 = 3$ X

$= [3 + 1] \% 10 = 4$

$h'(23) = [3 + 4] \% 10 = 7$

0	
1	
2	
3	3
4	13
5	
6	
7	23
8	
9	

O(1)

HashMap, HashSet

↓

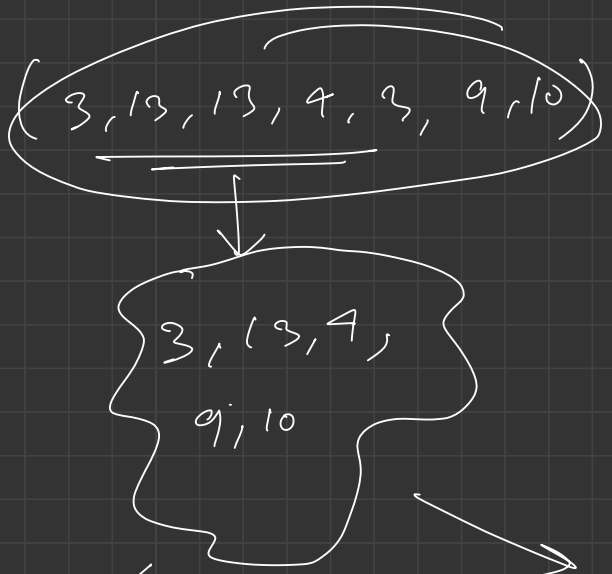
{ Hashing Algo }

TreeMap, TreeSet

↓

{ Red-Black trees }

Sets → Collection of unique entities.



HashSet

→ searching O(1)
→ random order

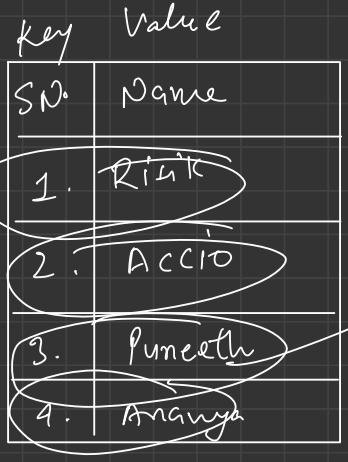
TreeSet

→ searching O(log N)
→ all stored in asc. order

```java
public void add(int key) {
    // Complete the function
    int hashValue = hashFunction(key);

    if (hashTable[hashValue] == null) {
        hashTable[hashValue] = new LinkedList<>();
    }

    if (hashTable[hashValue].indexOf(key) == -1) {
        hashTable[hashValue].add(key);
    }

}
```

HashMap

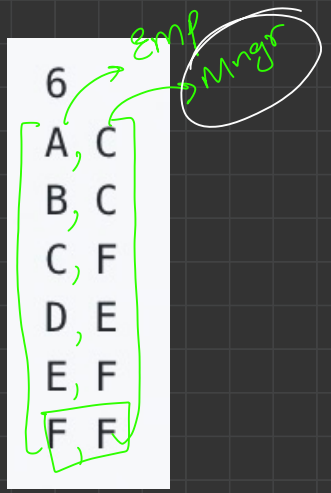
Key-, Value pairs

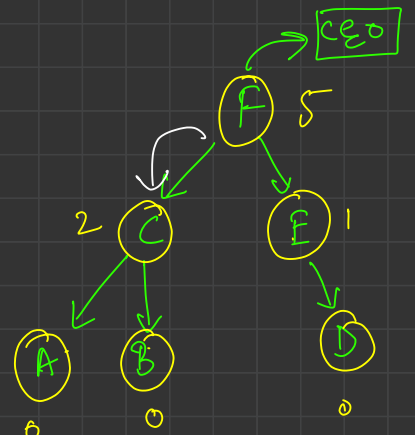


HashMap<Integer, String> map = new HashMap();

Key will be in Random order

↳ searching O(1)

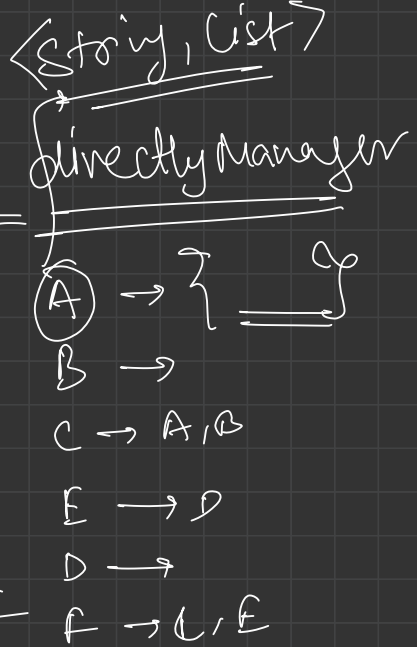




String, int

A → 0
B → 0
C → 2
D → 0
E → 1
F → 5

Sorted ? TreeMap?



```java
int peopleUnderMeIncMe(String manager, HashMap<String, ArrayList<String>> directReportie, Tre
    if (directReportie.containsKey(manager) == false) {
        ans.put(manager, value: 0);
        return 1;
    }

    int cnt = 0;
    for (String emp : directReportie.get(manager)) {
        cnt += peopleUnderMeIncMe(emp, directReportie, ans);
    }

    ans.put(manager, cnt);

    return cnt + 1;
}
```

A → 0
B → 0
C → 2
D → 0
E → 1
F → 5

C → A, B
E → D
F → C, E