Hashing 1

Content

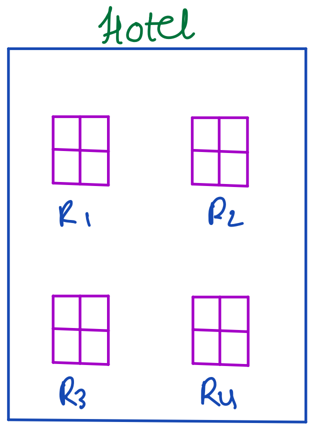
⟶ Introduction {Hotel Example}

⟶ Hash function + Collision + Chaining

⟶ Min equal pair distance.

⟶ Longest sub with 0 sum

⟶ Longest consecutive sequence.



R_1	:	Occ
R_2	:	Vac
R_3	:	Occ

R_1 — 3 fans
1 bal
2 AC

→ what is the no. of AC , balcony , fans .

→ AC , fans , Balcony

R_1 : 2 AC

key : value

$A = \{ 2 \quad 3 \quad 5 \quad 5 \quad 2 \quad 4 \quad 2 \}$

$$\begin{array}{ccccccccc} freq = \{ & 0 & 0 & 3 & 1 & 1 & 2 & \} \\ & 0 & 1 & 2 & 3 & 4 & 5 & \end{array}$$

⇒ key = index = value of A_i
value = count of A_i

i : $A[i]$

⇒ $\{100\}$

waste of space

$$\underbrace{[\qquad\qquad\qquad\qquad]}_{101}$$

size

$\{10^9\} \longrightarrow$ MLE

Problems with using freq array

If A[i] is 10^9, this will result in MLE.

hmap

hmap.put (10^9, 1)

Max size of freq array — 10^6

1 ——————————————— 10^9

1 —————————— 10^6

$$\{1 \text{ to } 100\} \xrightarrow[\%\,10]{} \{0 \text{ to } 9\}$$

⇒ It maps a key to a location, to convert the key into the index location

HMAP uses ⇒ Hash function

⇒ $f(x) = 2*x$ 1 → 2

$f(x) = \sin(x)$

⋮

$f(x) = x^2$ .....

⇒ Hash function $f(x) \% M$

$A = \{ 10 , 20 , 30 , 27 \}$

$M = 17$

Our own hash function

$x \longrightarrow x \% M \quad x \% 17$

$10 \longrightarrow 10$

$20 \longrightarrow 3$

$30 \longrightarrow 13$

$27 \longrightarrow 10$

$\Longrightarrow$

$3 \longrightarrow 20$

$10 \longrightarrow 10 , 27 \longrightarrow$ collision.

$13 \longrightarrow 30$

Collision is when you get same hash index for different values.

Pigeon Hole Principle

N — Holes
N+1 — Pigeons

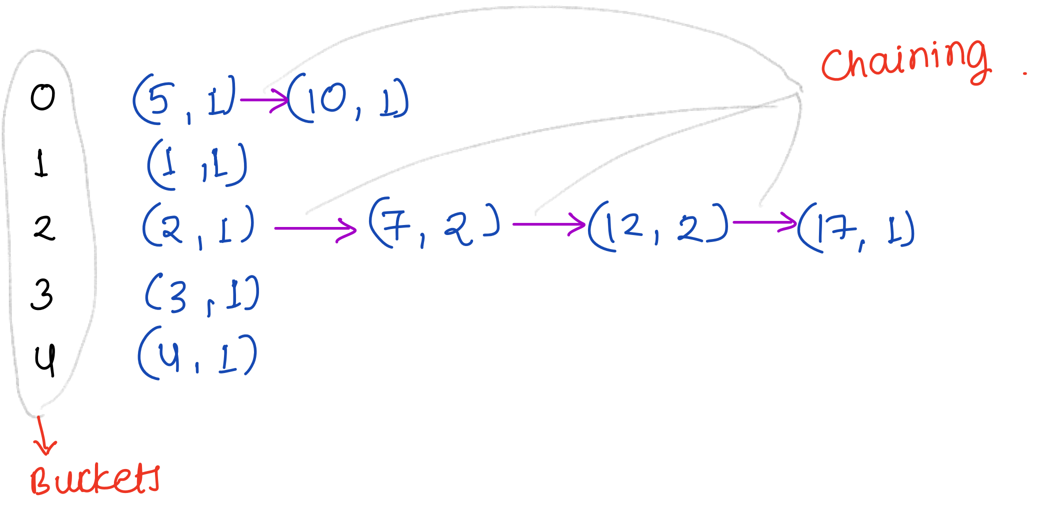
There are N holes and N+1 pigeons.
There will be atleast 1 hole with more than 1 pigeon.

Our goal while designing a hash map
1> minimize the collision
2> How to handle collision

$A = \{ 2 \quad 5 \quad 10 \quad 7 \quad 3 \quad 4 \quad 7 \quad 12 \quad 17 \quad 12 \quad \overset{\downarrow}{1} \}$

$M = 5$

$H(x) = x \% 5$



⟹ Load factor. — % of my HM space filled.

20 elements → max limit.

→ we stored 11 elements.

→ $\frac{11}{20}$ → X % of hashmap is filled

→ It doubles its size and recalculates all the values.

Java implementation of Hmap load factor is 75%

How is insertion, deletion in Hmap O(1)?

step 1> Calculate hash {O(1)}

step 2> Go through chain of elements at the hash index

↓

In the worst case every element goes to the same hash index O(N)

On average the TC of insertion, deletion in Hmap O(1)

TODO — Learn about internal working of a dynamic array.

$\{ V_1 \quad V_2 \quad V_3 \quad V_4 \quad V_5 \quad V_6 \quad V_7 \quad V_8 \}$

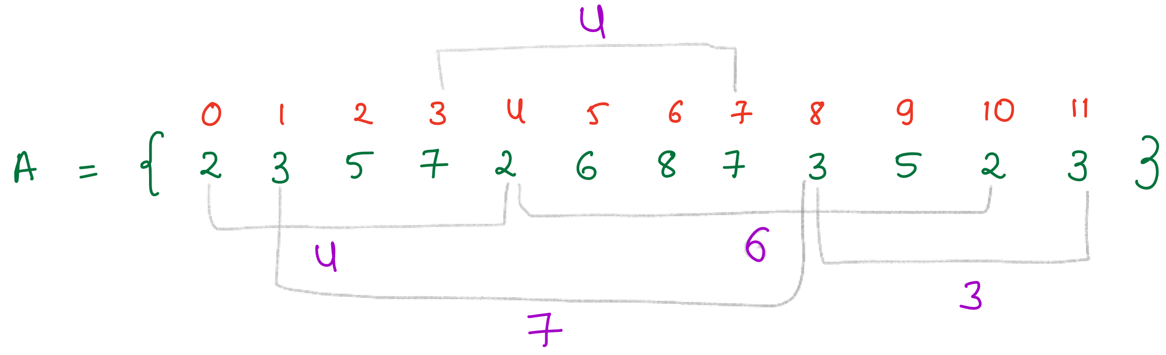
50 %　　size　10　　20

0	V_1	V_5
1	V_2	V_6
2	V_3	
3	V_4	

0	V_3
1	V_6
2	V_1
3	V_2
4	V_8
5	V_4
6	V_5
7	V_7
8	

Q> Given an integer array A, find any pair (i, j) such that

$i != j$ and $A_i == A_j$ and $|i-j|$ is min



$$A = \left\{ \begin{array}{cccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ 2 & 3 & 5 & 7 & 2 & 6 & 8 & 7 & 3 & 5 & 2 & 3 \end{array} \right\}$$

output = 3

Bruteforce

TC : $O(N^2)$
SC : $O(1)$

```
f i 0 to n-1
    f j 0 to n-1
        i != j  and  A_i == A_j
            ans = min (ans , |i-j|)
```

Idea 2 **Use Hash Map**

	0	1	2	3	4	5	6	7	8	9	10	11	
A = {	2	3	5	7	2	6	8	7	3	5	2	3	}

HMAP

key		value
2	:	~~0~~ ~~4~~ 10
3	:	~~1~~ ~~8~~ 11
5	:	~~2~~ 9
7	:	~~3~~ 7
6	:	5
8	:	6

ans = 4 − 0 = 4
= 7 − 3 = 4
= 8 − 1 = 7
= 9 − 2 = 7
= 10 − 4 = 6
= 11 − 8 = 3

Pseudocode

TC: O(N)
SC: O(N)

```
int pairMin ( A[] ) {
        hmap     // init in your own language.
        ans  =  ∞

        for ( i = 0 , i< n , i++ ) {
                val = A[i]
                if ( val is present in hmap ) {
                        pi = // get val from hmap
                           = hmap.get(val)
                        ans = min (ans , i - pi )
                }
                //Always update hmap
                hmap.put( val , i )
        }
    return ans .
```

Break 8:45

Q> Given an integer array, find the length of longest subarray with sum = 0

A = { 1 -2 2 -2 -3 -1 4 -1 }

5

-2 2

-3 -1 4

2 -2 -3 -1 4

2 -1 0 2 3

1 2 2 1 -3 4 3 1 -2 -3 2

whenever you see a question with subarray you should try to think in terms of what ?

A = { 1 -2 2 -2 -3 -1 4 -1 }

pf = { 1 -1 1 -1 -4 -5 -1 -2 }

pf = { 1 -1 1 -1 -4 -5 -1 -2 }

A = { 1 -1 }

pf = { 1 0 }

psum [i] = sum (0 to i)

```
⇒ if (psum [i] == 0)
       length = i+1
```

TODO

TC : O(N)   SC : O(N)

Q> Given an integer array A , find the length of longest chain of consecutive elements.

A = { 100 , 4 , 3 , 6 , 10 , 20 , 11 , 5 , 101 }

↓

100 101 → 2

5 6 → 2

10 11 → 2

4 5 6 → 3

3 4 5 6 → 4

Idea 1> Sort the array and keep checking for the consecutive elements.

Idea 2> Use Hash set / Hash map.

Algo —

1> Put the entire array into Hash set / Hash map

A = { 100, 4, 3, 6, 10, 20, 11, 5, 101 }

↑

100 → 101 → $O(N^2)$ {2}

4 → 5 → 6 → {3}

3 → 4 → 5 → 6 → {4}

6 → {1}

10 → 11 → {2}

20 → {1}

11 → {1}

5 → 6 → {2}

101 → {1}

{ 100, 4, 3, 6, 10, 20, 11, 5, 101 }

100 → 101 → ✓

~~4 → 5 → 6 →~~

3 → 4 → 5 → 6 → ✓

~~6 →~~

10 → 11 → ✓

20 → ✓

~~11 →~~

~~5 → 6~~ →

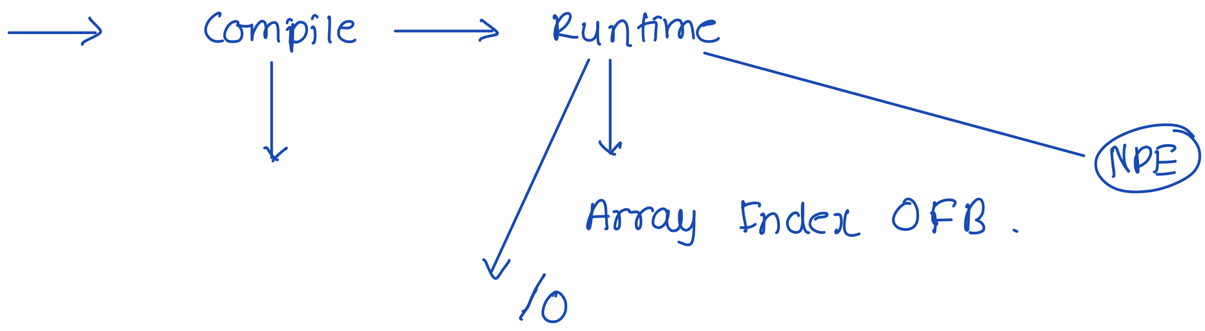
~~101 →~~

Pseudocode

```
int    longest Consecutive ( A[] ) {
            ans  =  0
            set  =      // init in your language
            // put all the elements of A in hashset

            for (i =0 ; i<n ; i++) {
                  val = A[i]
                  prev = A[i] - 1

                  if (prev is not present in set) {
                        start  = val
                        count  = 0

                        while ( start is present in set) {
                              start++
                              count++
                        }

                        ans  =  max (ans , count)
                  }
            }
                                            TC : O(N)
       return  ans                          SC : O(N)
}
```

Doubt session

→ Syntax .... { compile time }.



→ Compile → Runtime

NPE

Array Index OFB.

/0