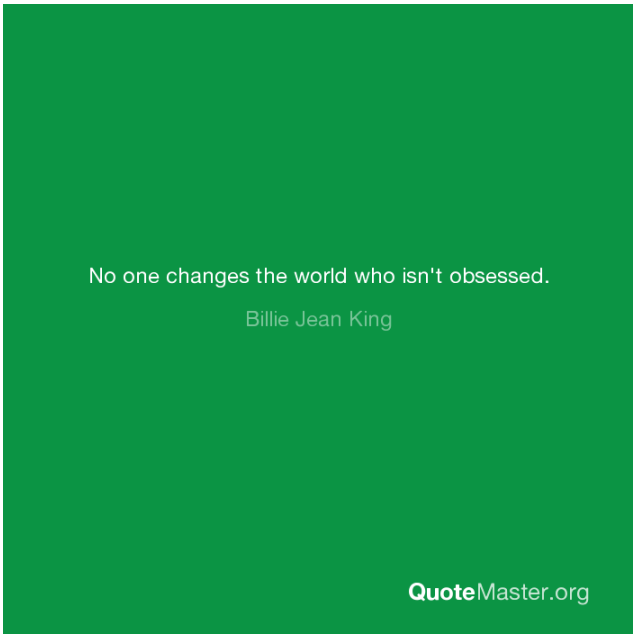
HASHMAP 1



Good Evening

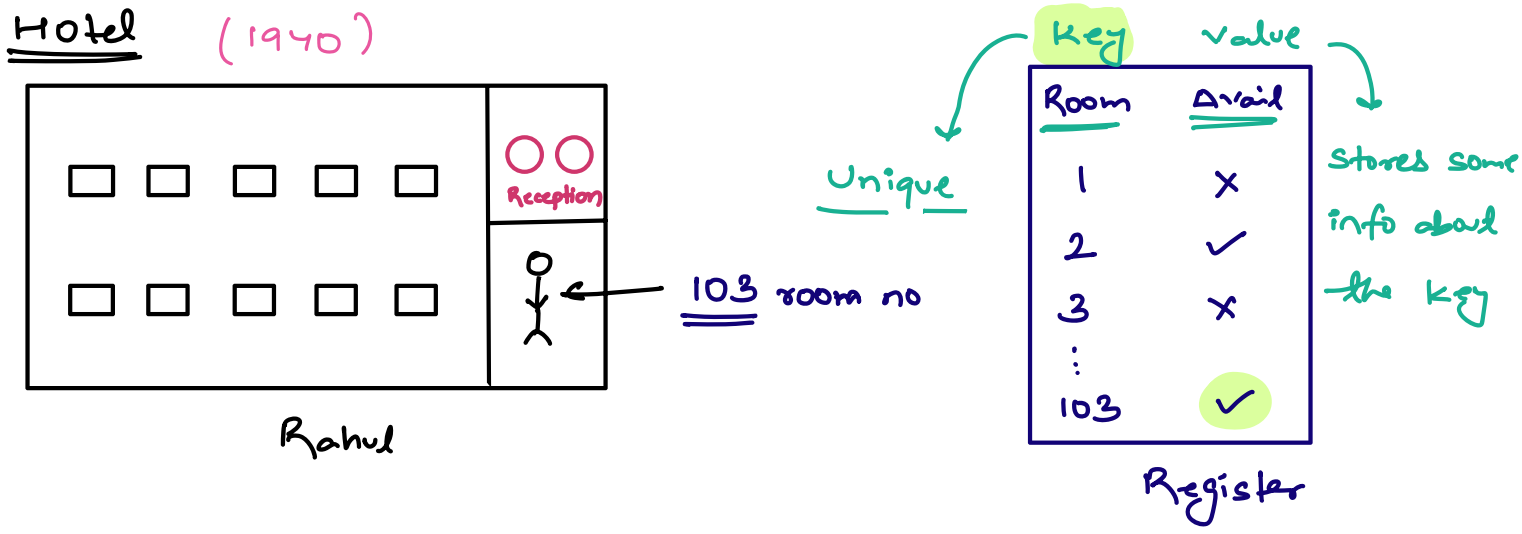
Today's Agenda

* Understanding HashMap

Questions
* Closest Duplicates
* Longest chain of consecutive elements
* Longest subarray with sum = 0

→ Array of lists

HashMap → DS which stores the info in key-value pair



Key (Room no) → People , price , Availability , AC

Values

Functionalities in HM

* Search for an ele
* Insert an ele
* Remove an ele
* Size of map
* Update an ele

TC: O(1) avg

* Store frequency of arr ele

A = { 3 2 1 2 5 1 5 }
0 1 2 3 4 5 6

freqarr =

0	1	2	3	4	5
0	2	2	1	0	2

→ Simplest hashmap

```
for (int i = 0; i<n; i++) {
    int val = A[i];
    freqarr[val]++;
}
```

TC: O(n)

SC: O(Range of ele)
O(max of the arr + 1)

* Range of elements = [1 10^9]

→ Can we create a freqarr? No

↓

Memory limit Exceeded

Array size = 10^6 or 10^7
= 40 MB

Array size = 10^9 → 4 GB ✗

* Limited Memory = M

Do we have anything which can limit the range of ele?

Modulo operator

Hash Function

Provides us the key within our memory limit

$h(x) = x \% M$

- x: original key
- $x \% M$: New key which is going to be in our limited memory

$arr[] = \{10, 20, 30, 27\}$ $M = 17$

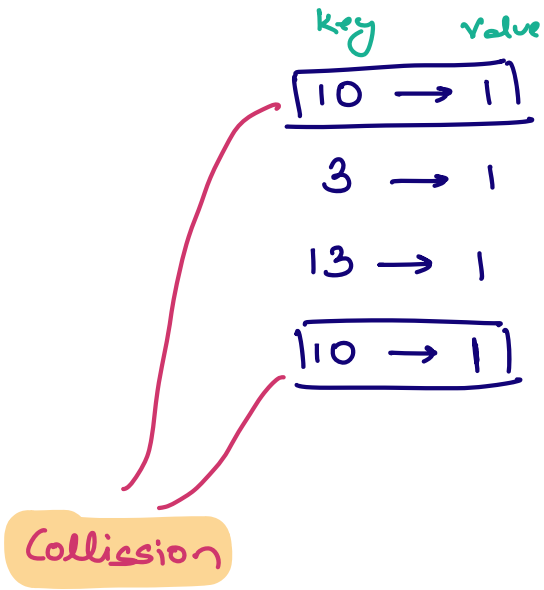
$h(x) = x \% 17$

$h(x) = 10 \% 17 = 10$

$h(20) = 20 \% 17 = 3$

$h(30) = 30 \% 17 = 13$

$h(27) = 27 \% 17 = 10$



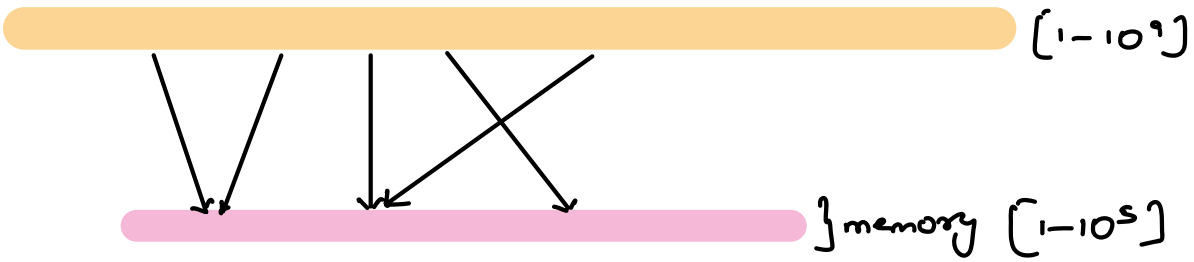
Is it possible to avoid collision? NO

(limited memory)

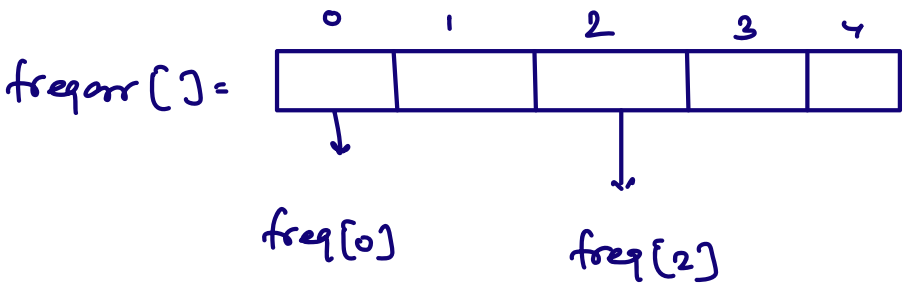
(Pigeon hole principle)

Pigeon hole principle → N holes & (N+1) pigeons & you want to put every pigeon in a hole.

→ atleast one hole with more than 1 pigeon



03 Handle the collision → Chaining

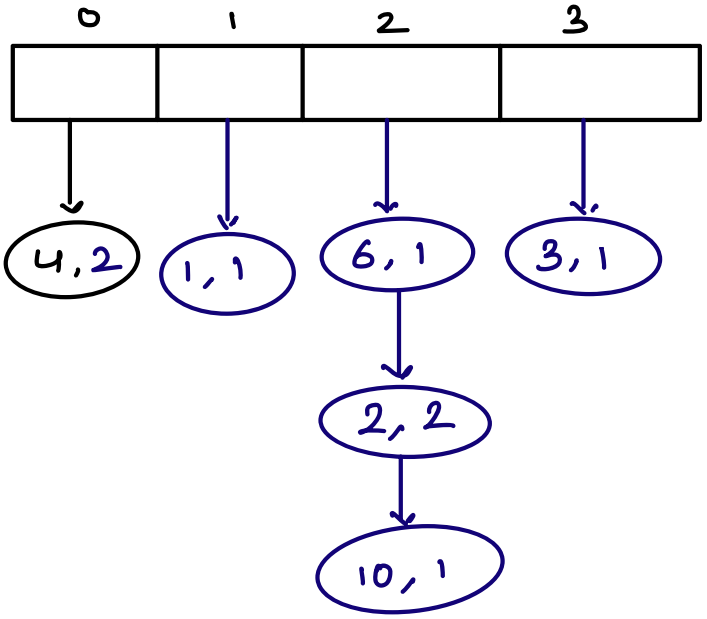


* Instead of storing the freq[i].

store a list containing A[i], freq[A[i]]

A = { 3 6 4 1 2 10 2 4 } M=4

freqarr =



Loading factor = 2

As soon as we go beyond LF, rehashing

Disadvantage of Chaining

* Searching for an ele eg!- 3 ⇒ TC: O(1)

Worst = TC: O(n)

* Hashset
  ↳ store the keys

Q Check if array contains duplicates or not?

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

Key - value

HashMap

4 → 1
1 → 2 X → Contains duplicates
2 → 1
3 → 1

Hashset (Unique)

1, 2, 3, 4

if (hs.size() < n) → duplicate

Closest duplicates

Q → Given an integer array of size N. Find pair (i,j) such that $j > i$ and $A[i] == A[j]$ && $j - i$ is minimum

arr[ ] =

2	4	5	6	-1	2	5	4	3	7	3	2
0	1	2	3	4	5	6	7	8	9	10	11

Ans = 10 - 8 = 2

Brute force → Consider all pairs,

→ find all duplicates pairs, calculate the distance & update the ans accordingly

TC : $O(n^2)$

SC : $O(1)$

arr[ ] = { x .... x .... x ... x }

Conclusion → Should try to store the index of closest equal element.

arr[ ] = { x (7) .... x (5) .... x (10) ..... x }

0 7 12 22

x = ∅ 7 ~~12~~ 22

ans = ∞

ans = ∞ vs 7 - 0 = 7

ans = 7 vs 12-7 = 5

ans = 5 vs 22-12 = 5

arr = {1, 2, 3, 6, 2, 1}

0 1 2 3 4 5

* HashMap

Key → Distinct ele

Value → indexes

Key	Value
1	~~0~~ 5
2	~~1~~ 4
3	2
6	3

ans = ∞

ans = min(∞ vs 4-1) = 3

ans = min(3 vs 5-0) = 3

```
HM<I, I> map = new HM<>();

int ans = ∞

for (i=0; i<n; i++){
    int ele = A[i];

    if (map.containskey(ele) == false){
        map.put(ele, i);
    }
```

```
        else {
            int lo = map.get(ele)
            ans = min(ans, i - lo);
            map.put(ele, i);
        }
    }
    return ans;
```

TC : O(n)
SC : O(n)

10:16 pm → 10:26 pm

02. Given an array of size N. Find the length of longest sequence of consecutive elements.

Eg :- {100 4 3 6 10 20 11 5 101}

Sequence = 100, 101 → 2
3, 4, 5, 6 → 4
10, 11 → 2
20 → 1

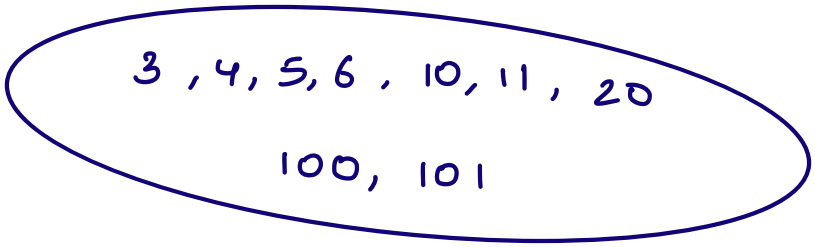
Ans = 4

Idea 1 → Sort the array

3, 4, 5, 6 | 10, 11 | 20 | 100, 101
4 | 2 | 1 | 2

TC : O(nlogn)
SC : O(1)

Idea 2 → HashSet



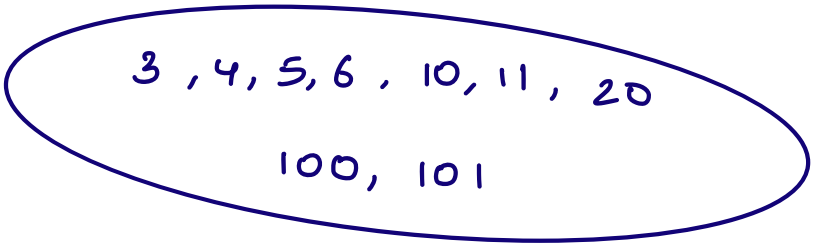
Eg:- {100 4 3 6 10 20 11 5 101}

	length
100 → 101	2
4 → 5 → 6	3
3 → 4 → 5 → 6	4 → ans
6	1
10 → 11	2
20	1
11	1
5 → 6	2
101	1

TC: $O(n^2)$
SC: $O(n)$

x x-1

Obs → If x-1 is present, can x be the starting point? No



Eg :- {100 4 3 6 10 20 11 5 101}

100 → 101 length = 2

4

3 → 4 → 5 → 6 length = 4

6

10 → 11 length = 2

20 length = 1

11

5

101

TC : O(2n) ≈ O(n)

```
HashSet<I> set;

for(i=0; i<n; i++){
    set.add(A[i]);
}
```

TC: O(n)

SC: O(n)

Iterate on hashset

```
for (i=0; i<n; i++) {
    int x = A[i];
    if (set.contains(x-1) == false)
        chain = 1
        y = x+1
        while (set.contains(y))
            chain++;
            y = y+1
        ans = max(ans, chain);
}
return ans;
```

arr = {6, 6, 6, 6, 7, 8, 9}

hashset = {6, 7, 8, 9}

6 → 7 → 8 → 9

6 → 7 → 8 → 9

6 → 7 → 8 → 9

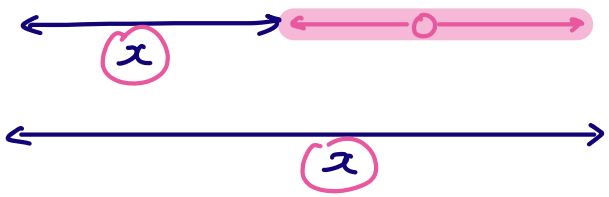
6 → 7 → 8 → 9

Q Given an array of Integers. Find the length of **longest subarr with sum = 0**

arr[ ] =	2	2	1	-3	4	3	1	-8	6	-2	1
	0	1	2	3	4	5	6	7	8	9	10

Ans = 7

pf[ ] =	2	4	5	2	6	9	10	2	8	6	7
	0	1	2	3	4	5	6	7	8	9	10



Elements are repeating in pfsum, then the subarr sum b/w them is 0

01. Insert the subarrsum in hashmap & if the sum appears again → update the ans but we are not going to update its value.

arr[ ] =	2	2	1	-3	4	3	1	-8	6	-2	1
	0	1	2	3	4	5	6	7	8	9	10

pf[ ] =	2	4	5	2	6	9	10	2	8	6	7
	0	1	2	3	4	5	6	7	8	9	10

⇒ Optimise this to O(1)

HashMap Keys → Subarray sums

Values → index

Key	Value
2	0
4	1
5	2
6	4
9	5
10	6
8	8
7	10

ans = 0

ans = max(0 vs 3-0) = 3

ans = max(3 vs 7-0) = 7

ans = max(7 vs 9-4) = 7

longest length of subarray sum = 0

array = {2, -2}
0 1

Pfsum = {2, 0}
0 1

HM

0 → -1

2 → 0

1-(-1) = 2

01. if (Pfsum[i] == 0) {
    len = i-0+1
}

3

02. Add (0, -1) in our HM prior to all iterations

arr = { 2, 4, -6, 3, -3 }

pfsum = { 2, 6, 0, 3, 0 }
0 1 2 3 4

os = 4 - (-1) = 4+1

subarr sum → I , indexes → I

```
HashMap <I, I> map;
map.add(0, -1);
int sum = 0
for (i=0; i<n; i++){
    sum += A[i];
    if (map.containskey(sum) == true){
        len = max(len, i - map.get(sum))
    }
    else {
        map.put(sum, i);
    }
}
return len;
```

TC : O(n)

SC : O(n)