Why Hashing?

How is hashmap implement?

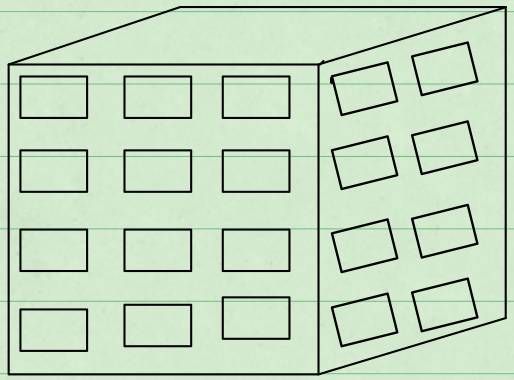
Problem Patterns.

Linked list
Trees
self BBST.

Two volunteers

Afnan Akshay

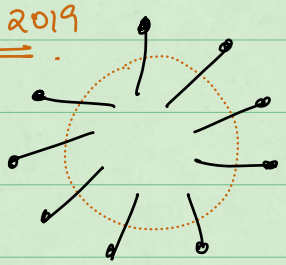
1000 rooms.



Register

Room No.	Availability
1	A
2	O
⋮	
1000	O

Dec 2019



NUMEROLOGIST

↳ $1 - 10^9$

1000 Rooms {38, 42, 56, 1, 7, ...

Before Covid.

Array
Size : 1000

bool arr [1000]
⇓ 1001

arr[i] → F → occupied
arr[i] → T → unoccupied.

After Covid

$1 - 10^9$

bool arr [10^9+1]

Rooms → 1000

1 space → 1 Byte.

1000 " → 1000 Bytes.

10^9 Bytes = 1GB.

Advantages

① Access
- update O(1)
- lookup O(1)

Requirements

① No space should get wasted
② lookup O(1)
③ update O(1)

1st → 4s
2nd → 1s
3rd → 1s
4th → 3s

12/6 → 2s.

O(1)

Array
1st → 2s
2nd → 2s
3rd → 2s

5th → 1S
6th → 2S

Hashmap

Room No. Availability

<Key, value>

RoomNo → int

availability → bool

Hashmap < int, bool >

Que. Country Name → Population

Hashmap < Name, population >

Name → String

population → long

Que. For every country, we want to store all the state names.

Hashmap < country name, state names >

country name → String

state names → list

Hashmap < String , List < String >>

Q. For a country, store population of each state

Hashmap < country_name , population of every state >

String

Hashmap < String , int >

Hashmap < String , Hashmap < String , int >>

Value : can be anything

Key : datatype : primitive

- int
- long
- double
- string
- float
- bool

Can we have duplicate keys? → NO!

Hashmap

HashSet	Set	unordered_set	set
Java	Python	C++	JS / Ruby
HashMap	Dictionary	unordered_map	Map

C → move on

Hashmap Functionalities

O(1)

1. insert (key, value)
2. get (key)
3. delete (key)
4. update (key, value)
5. size () // no. of keys
6. isPresent / contains (key)

Que. Given an array of size N & Q queries.

Every query → x (integer)

Return frequency of x in array

A : {2, 6, 3, 8, 2, 8, 2, 3, 8}

Q : 10

2 → 3
8 → 3
6 → 1
3 → 2
⋮

Hashmap

2	→	3
6	→	1
3	→	2
8	→	3

Brute Force

Q times:
1. → iterate on array [O(N)]
2.
3.

TC → O(N * Q)

SC → O(1)

Hashmap

Hashmap< value, freq > mp ;

int, int

```
for(i = 0; i < N; i++) {
    if( A[i] is present in map) {
        update ( A[i], mp[A[i]] + 1 )
    }
    else {
        insert ( A[i], 1)
    }
}
```

For each query → O(1)

Q queries → Q x 1

O(Q)

creation of map

TC → O(N) + O(Q) = O(N+Q)

SC → O(N)

→ for answering queries

Q2. Given an array, count the no. of distinct ele in

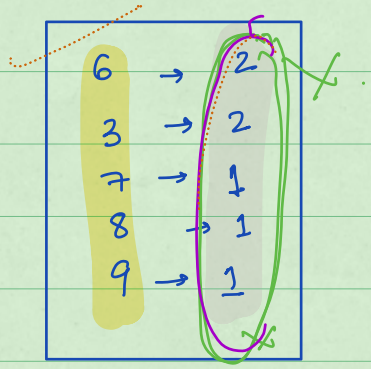
= the array.

A: [7, 3, 2, 1, 3, 7, 0]
0 1 2 3 4 5 6

— QUIZ —

A: [6, 3, 7, 3, 8, 6, 9]

Hashmap < val, freq >
int int

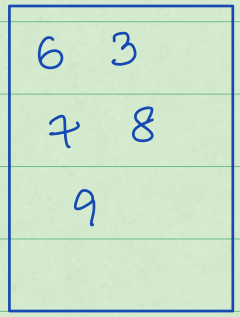


✓
① Create Hashmap
② Return size of Hashmap

Set / HashSet.

stores only keys
unique keys.

Return size



set<int> s;

```
set<int> s;
for(i=0; i<N; i++){
    s.insert(A[i])
}
return s.size()
```

TC → O(N)
SC → O(N)

Amazon | LinkedIn

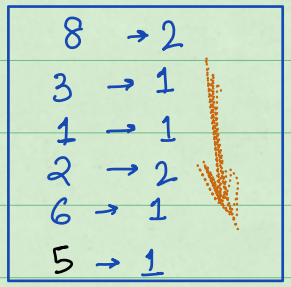
Q. Given an array of size N, find first non repeating ele in the array

A: {8, 2, 8, 3, 1, 2, 6, 5}

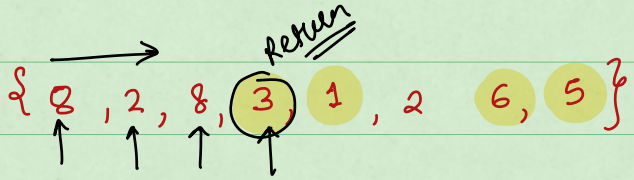
ans.

A: {1, 2, 3, 1, 2, 5}

ans.



order is not maintained in HashMap.



Steps

1. Store freq of ele in map
2. Iterate over array & if any ele has freq 1, return it

TC → O(N) + O(N) = O(N)

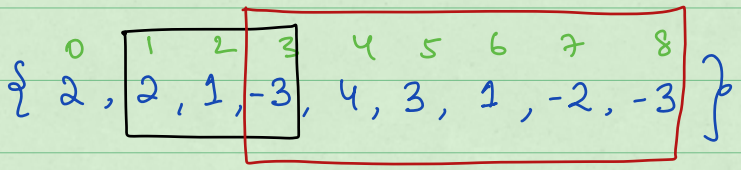
SC → O(N)

Break: 10:25

Que. Directi | Amazon | Myntra | Flipkart

Given an array, check if there exists a subarray with sum = 0

subarray ⇓ continuous part of the array



Create all the subarrays.
Check if sum = 0 or not

TC → O(N³)
SC → O(1)

```
for( i=0; i<N; i++) {
    for( j=i ; j<N; j++ ){
        sum=0
        for( k=i; k<=j ; k++) {
            sum += A[k]
        }
        if (sum == 0)
            return true
    }
}
return false.
```

create pf[]

```
for( i=0; i<N; i++) {
    for( j=i ; j<N; j++ ){
        sum = Pf[j] - Pf[i-1]
```

TC → O(N²)
SC → O(N)

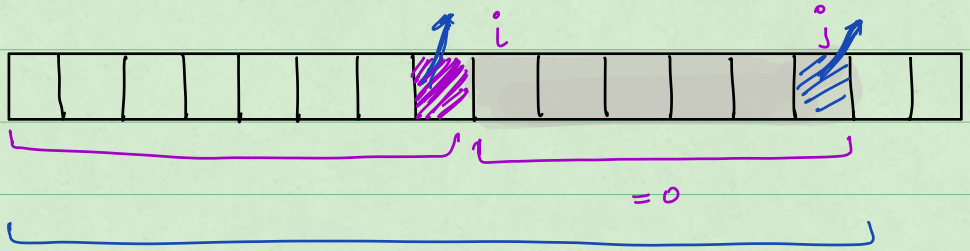
```
if (sum == 0)
    return true.
}
return false.
```

$sum(i,j) = Pf[j] - Pf[i-1]$

Assume: $sum(i,j) = 0$

$0 = Pf[j] - Pf[i-1]$

$Pf[j] = Pf[i-1]$



$Pf[j] = sum(0,j)$

$Pf[j] = sum(0,i-1) + sum(i,j)$

$Pf[j] = Pf[i-1] + sum(i,j)$

same

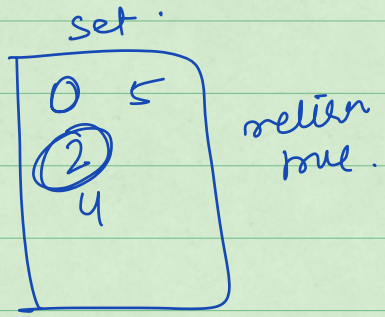
Conclusion: If there exists a sum = 0 two values in array of prefix sum will be same

OR

if you find two repeating values in the prefix array, then sum b/w those two values = 0

A: {2, 2, 1, -3, 4, 3, 1, -2, -3}

Pf: {2, 4, 5, 2, 6, 9, 10, 8, 5}



return true.

Steps: ① Create Pf array

② use set / map.

Edges Cases

A: {1, 2, 0, 8}

{1} {2} {0} {8}

{1,2} . . . .

Pf: {1, 3, 3, 8}

A: {0, 3, 2, 8}

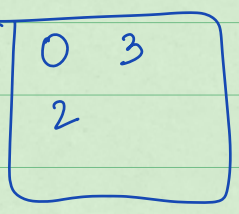
→ return true

Pf: {0, 3, 5, 13}

if 0 is present in given array answer will always be true.

A: {3, -1, -2, 4} →

Pf: {3, 2, 0, 4}



① // check if 0 is present in prefix array.

② insert 0 by yourself in set

Q. Find the length of longest subarray with sum = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12
A[]:	3	3	4	-5	-2	2	1	-3	3	-1	5	-4	-1

ans = 10

	0	1	2	3	4	5	6	7	8	9	10	11	12
Pf[]:	3	6	10	5	3	5	6	3	6	5	10	6	5

Observation:

Find diff. b/w first index & last index of repeated ele.

Hashmap<val, first occurrence index>

Hashmap<val, last occurrence index>

Hashmap<val, list<ind>>

	0	1	2	3	4	5	6	7	8	9	10	11	12
A[]:	3	3	4	-5	-2	2	1	-3	3	-1	5	-4	-1

ans = 10

	0	1	2	3	4	5	6	7	8	9	10	11	12
Pf[]:	3	6	10	5	3	5	6	3	6	5	10	6	5

3	→	0
6	→	1
10	→	2

			ans
3 : 4	4 − 0	= 4	4
5 : 5	5 − 3	= 2	4
6 : 6	6 − 1	= 5	5
			7

5 → 3

3 : 7	+−0	= +	1
6 : 8	8 − 1	= 7	7
5 : 9	9 − 3	= 6	7
10 : 10	10 − 2	= 8	8
6 : 11	11 − 1	= 10	10
5 : 12	12 − 3	= 9	10

Steps

① Pf[ ]

② Hashmap<int, int> mp

```
ans = 0
for(i=0; i<N; i++) {
    if(A[i] is present) {
        ans = max(ans, i - mp[A[i]])
    }
    else {
        mp.insert(A[i], i)
    }
}
return ans
```

	-1	0	1	2	3	4
ar[ ] :		4	-3	-1	2	-2
pf[ ] :	0	4	1	0	2	0

ans

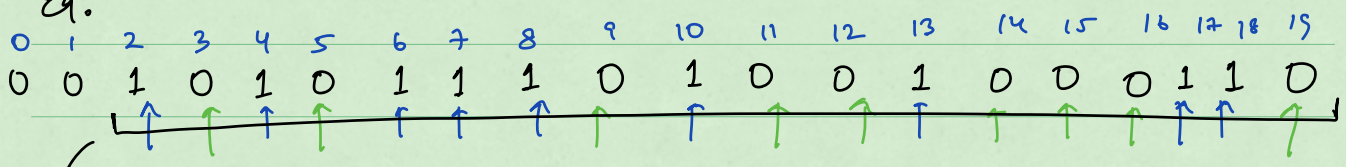
key	value
0	-1
4	0
1	1

0 : 2 → 2 − (−1) = 3

0 : 4 → 4 − (−1) = 5

Ques: Given an array containing only 1s & 0s. Find max length subarray which contains equal 0's & 1's.

Eg:



Replace 0 with -1

Left with 1 & -1

max len subarray having sum 0.