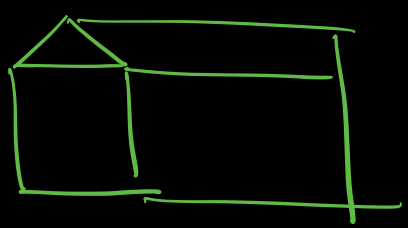
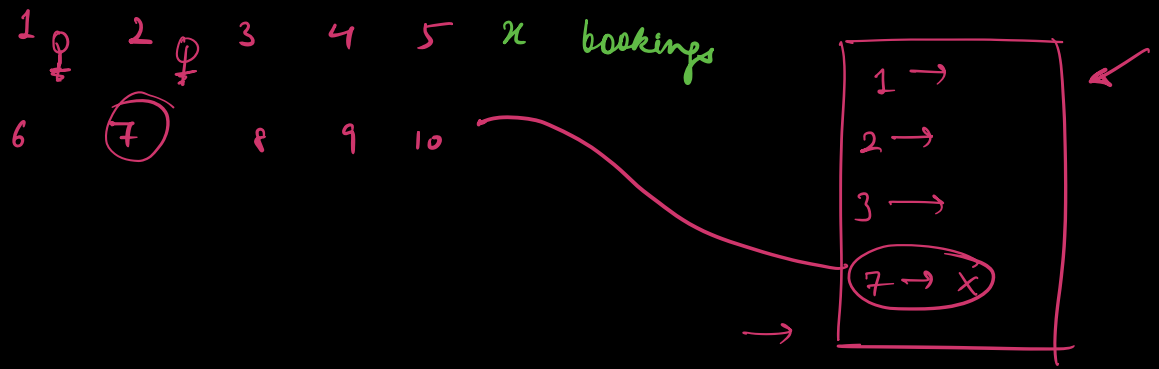
Agenda

1. Why one more DS ?
2. Implementation
3. Python specific
4. Problems



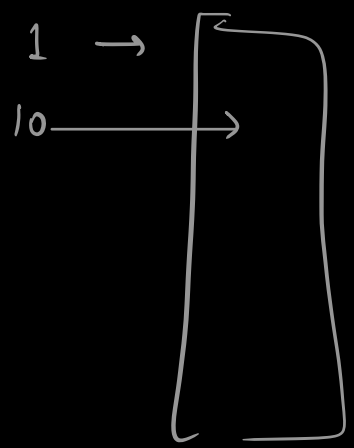
10 rooms in hotel

1000 rooms



COVID → Baba

new room nos



10^{10}

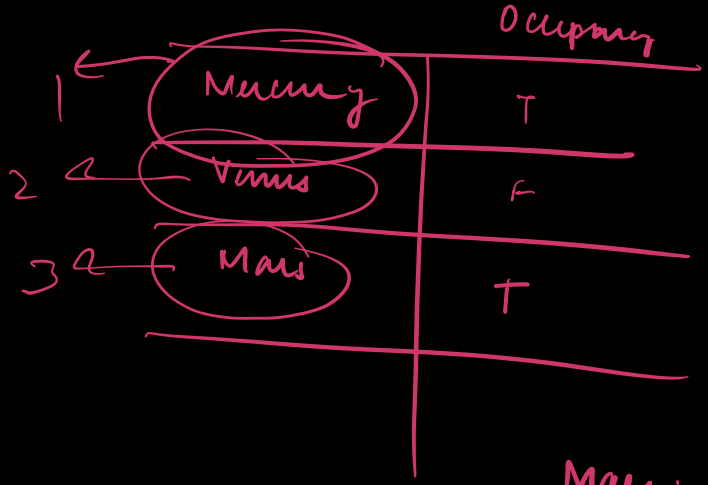
(Indexes to occupance pairs)

10^0 10^1 10^2 10^3 . — —

Room No.	Index	Occupied
10^0	0	T
10^1	1	F
10^2	2	T
10^3	3	T

Baba -2 → Name Rooms after Stars, Planets

- Mercury
- Venus
- Mars
- Jupiter



Mapping room names to numbers

Just number indexes that could be mapped to values

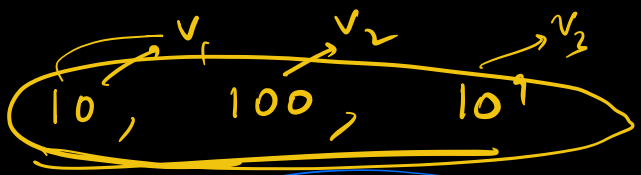
```
l[0]
l[1] = 'a'
```

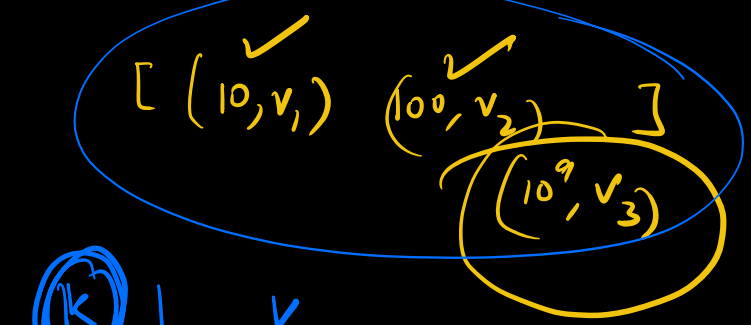
Map keys to some values

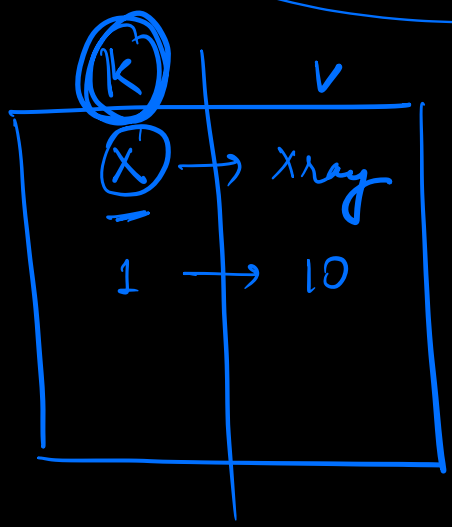
Magic_DS['key'] → value

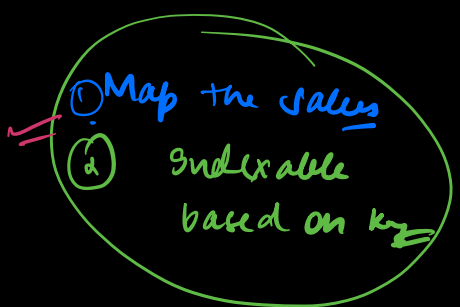
Hash Maps

- → Space efficient solution to store K-V pair
- → Fast access of data ✓
- → Check if a key is present or not
- → Insert K, V pair
- → Update





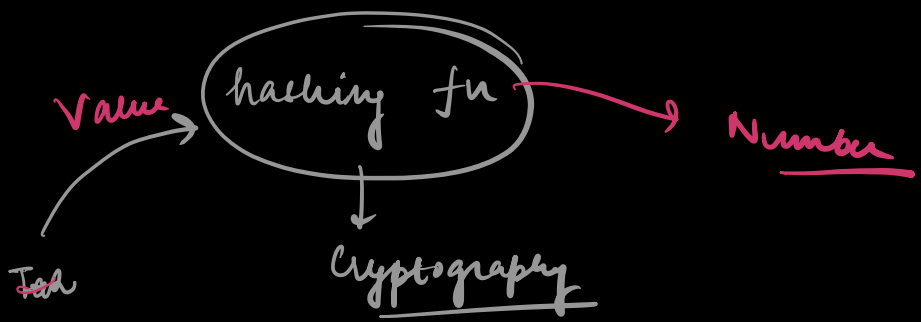




Dictionary

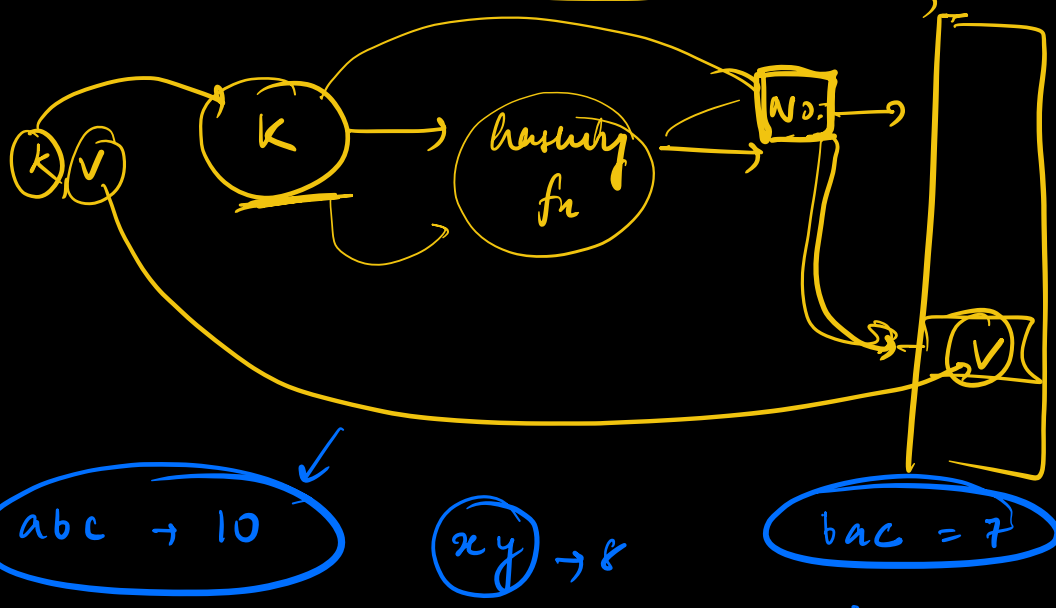
① Mapping

Student Name	Marks
A	10
B	9
C	7
D	8
E	10




```
def hash_string(x):
    ans = 0
    for c in x:
        ans += ord(c) - 97
```

abc
97+98+99



abc → 10

xy → 8

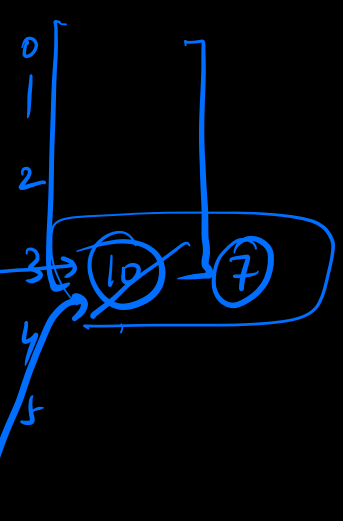
bac = 7

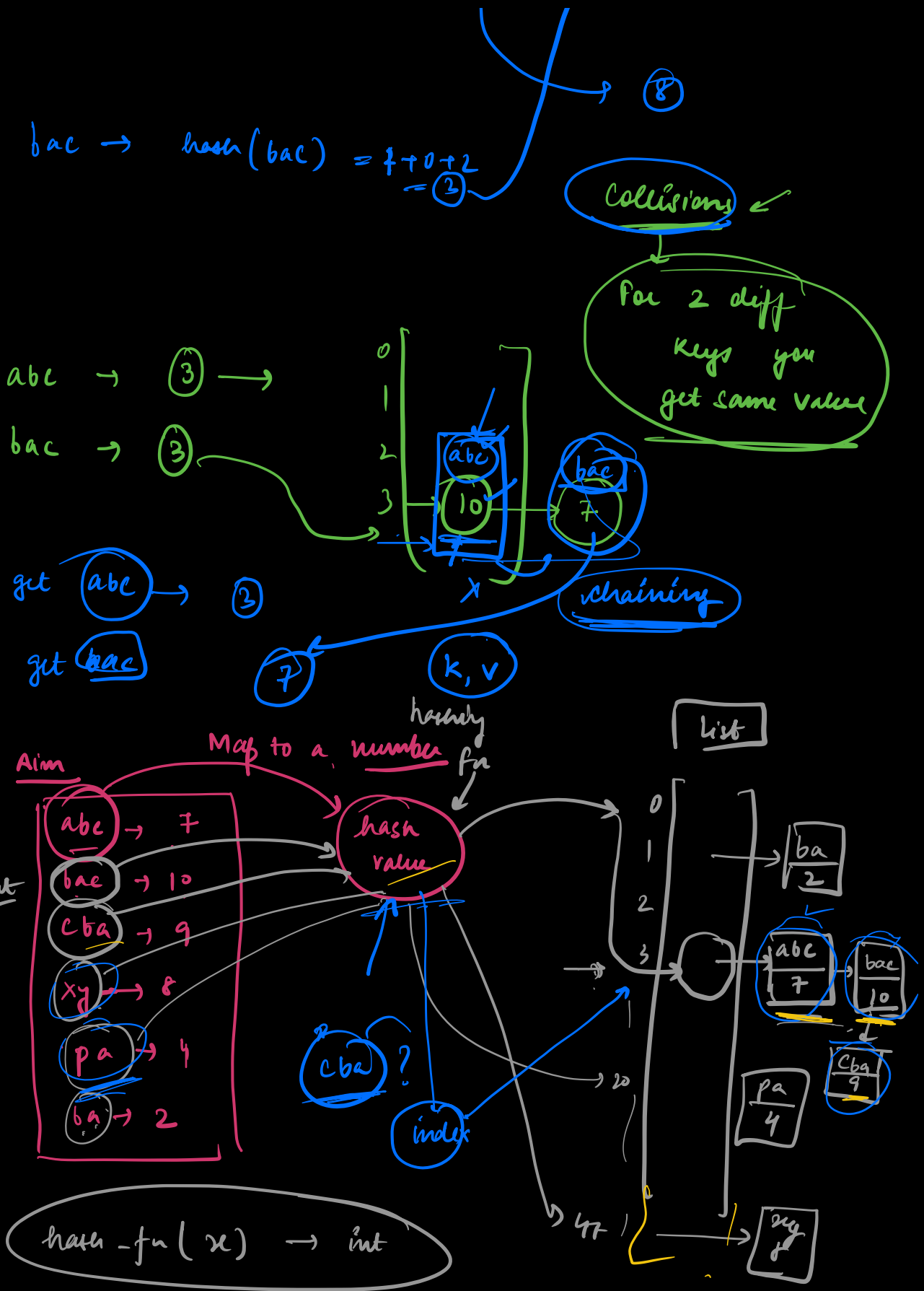
abc → hash(abc) → 3

0+1+2

xy → hash(xy) → 23+24

= 47




bac → hash(bac) = 7+0+2 = 3
8
Collisions
For 2 diff keys you get same value
abc → 3 →
bac → 3
0
1
2
3
abc
10
bac
7
get abc → 3
get bac
7
k, v
chaining
Aim
Map to a number
hashing fn
Input
abc → 7
bac → 10
cba → 9
xy → 8
pa → 4
ba → 2
hash value
cba ?
index
20
47
List
ba 2
abc 7
bac 10
cba 9
pa 4
xy 8
hash_fn(x) → int

Need → space efficient
Get in O(1) time
→ improving H.F → unique

Size of list on right depends on the range of hash values!

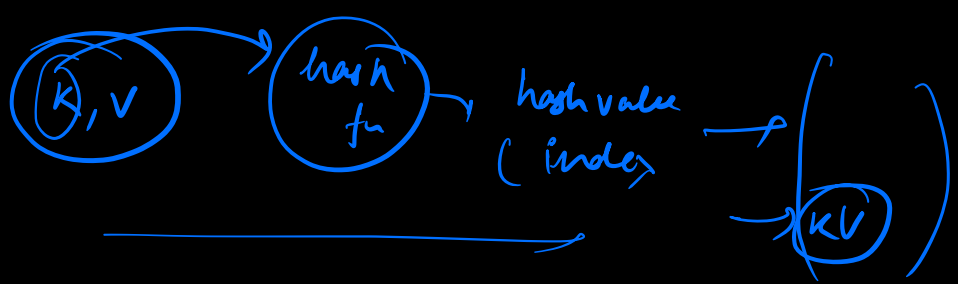
W.C → O(N) → All keys are mapped to same → hash

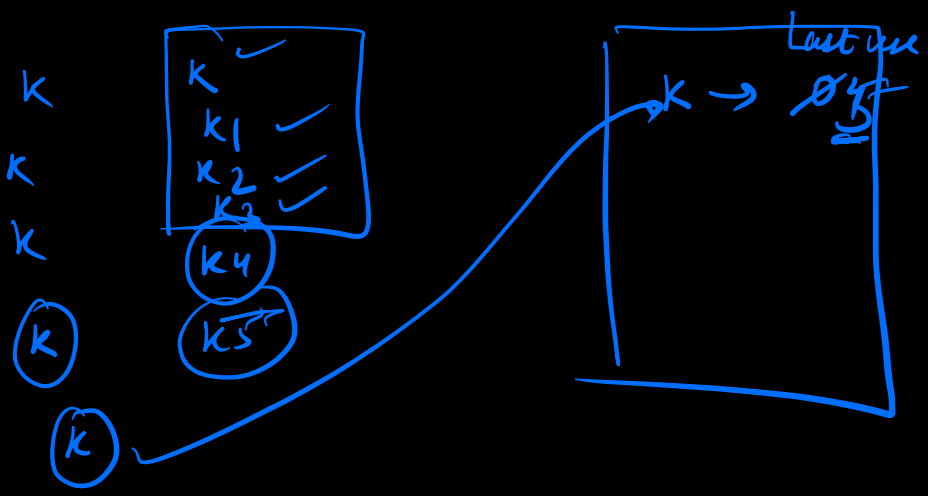
Access / fetch time → O(1)

Avg Case

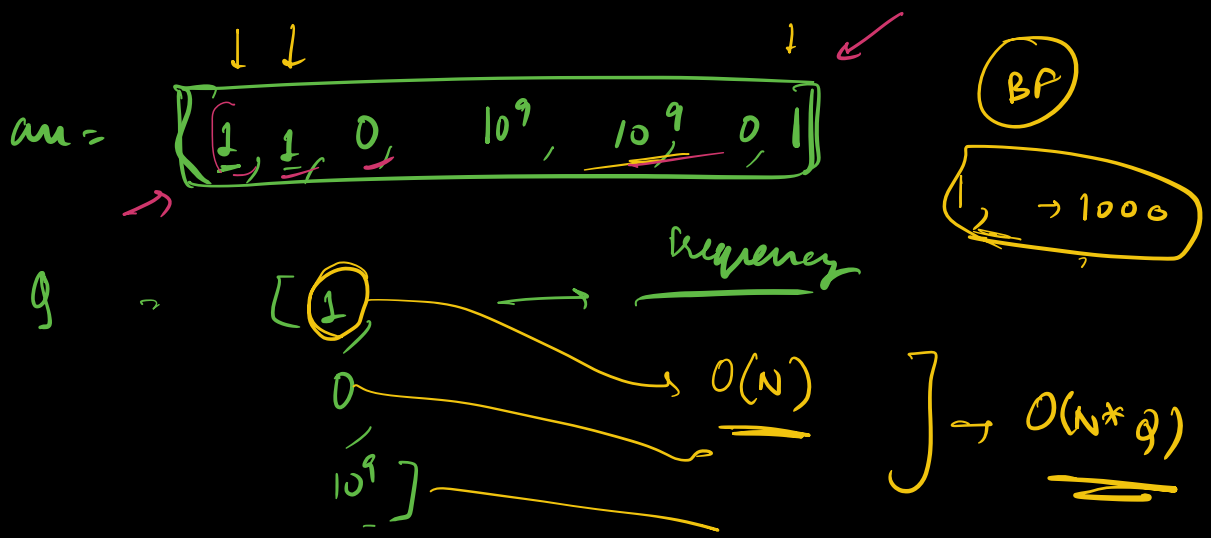
Worst case

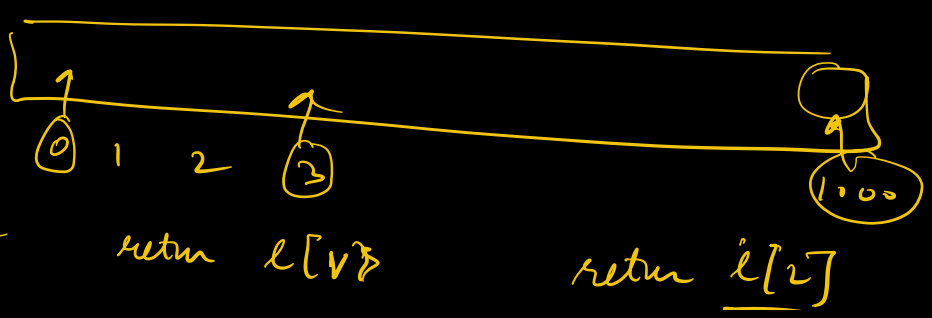
O(N)



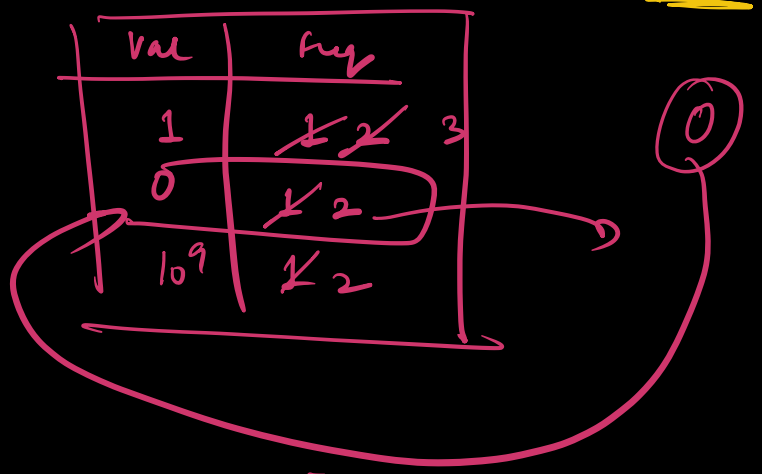


Q Given an array. You have Q queries
Return the count of elements in array
for each query





(2)



return dict[0]

```
def solve(arr, queries)
    freq = {}
    for val in arr:
        if val in freq:
            freq[val] += 1
        else
            freq[value] = 1

    for query in queries:
        if query in freq:
            return freq[query]
        else
            return 0
```

[1]

1000

1

O(N)

O(Q)

O(N+Q)

arr = [1]

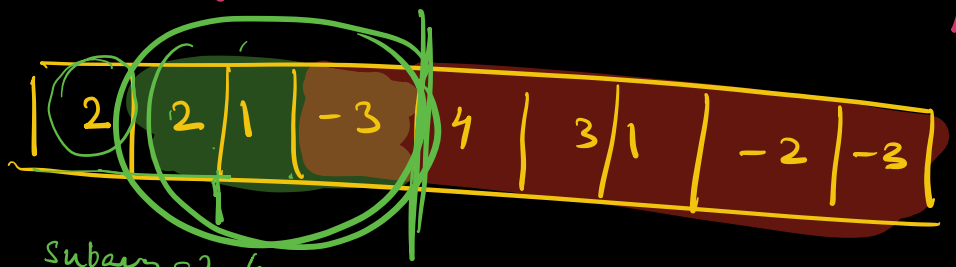
Q = [1, 1, ..., 1] (1000)

arr [1, 2, 3, 1, ..., 3] (1000)

Q → 1

Break → 5 mins

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



Subarray = 2, 4

sum = 4 → 5

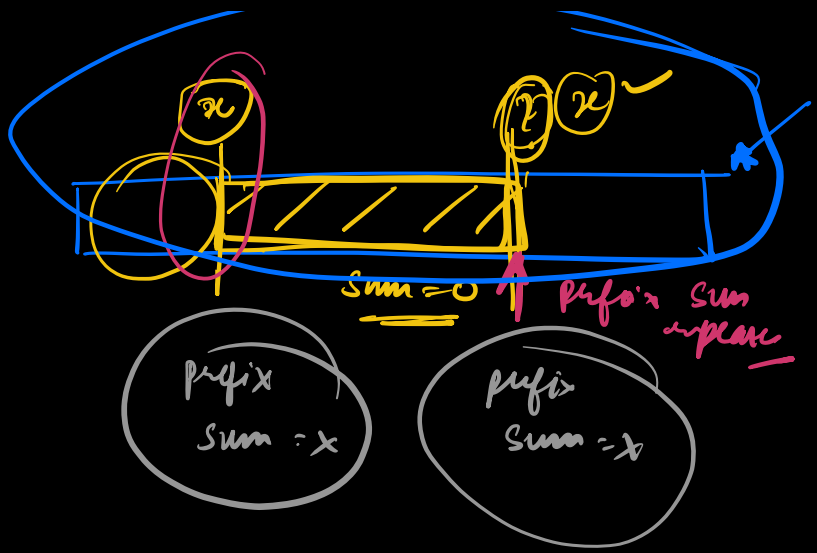
$O(N^2)$
$O(1)$

i → ending index

```
for i in range(N):
    for j in range(0, i+1):
        if pref[i] - pref[j-1] == 0
```

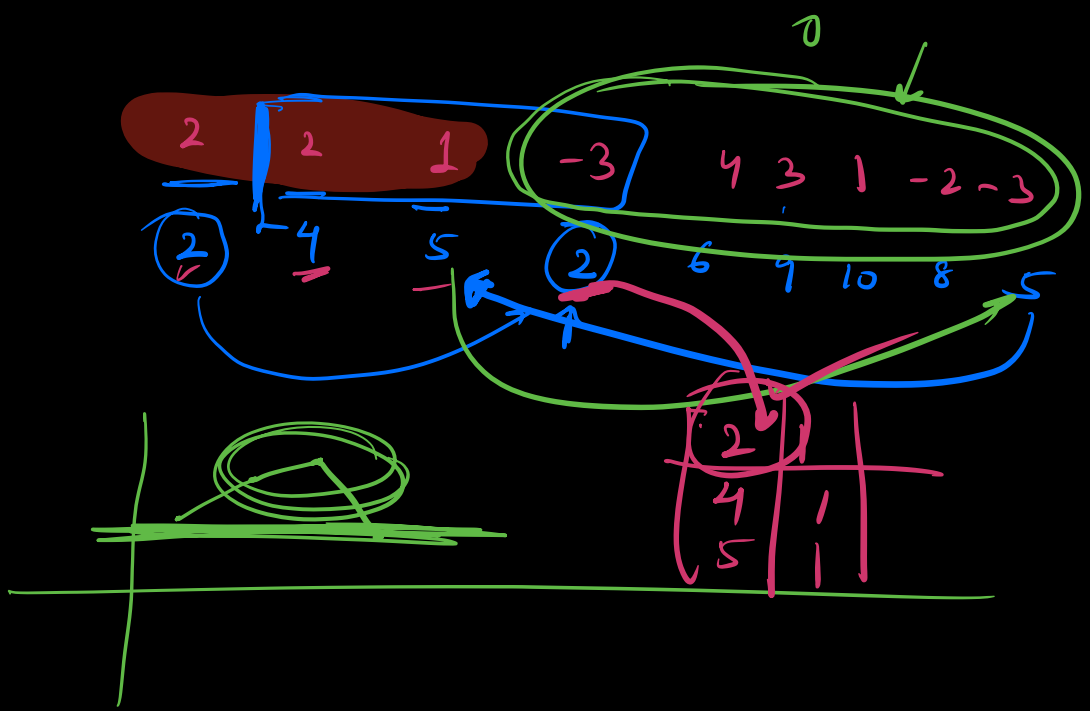
Starting index

At index i is there any subarray ending at index i with sum = 0



Repeats itself ⇒ there exist a subarray with sum 0

a

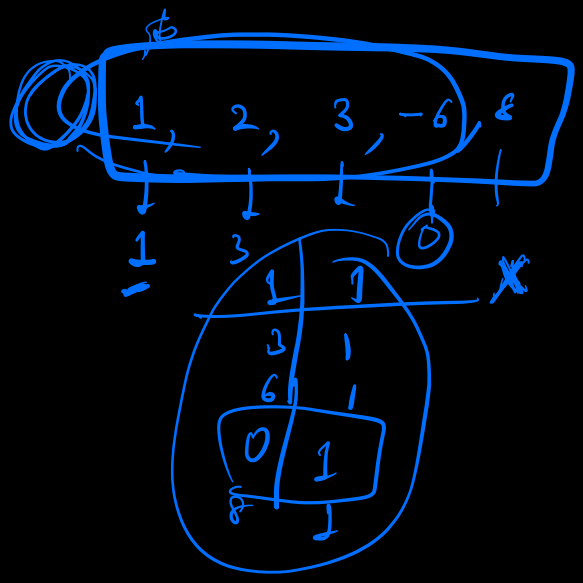


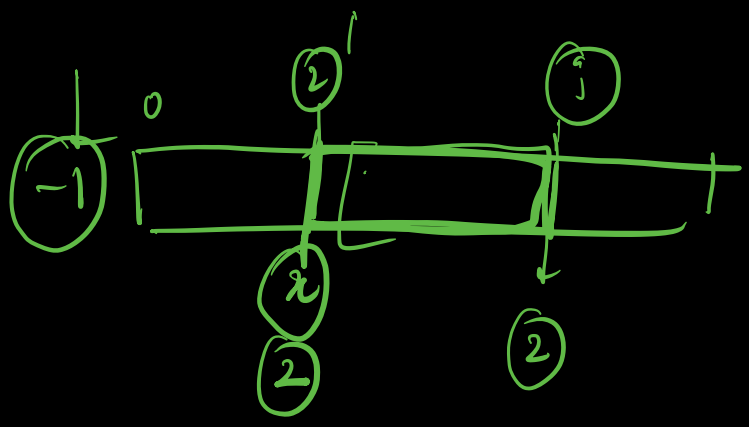
if a prefix sum value is repeated or not

```
dict[0] = -1          dict = {0:-1}
for i in range(n):
    sum += arr[i]
    if sum in dict:
        return True
    dict[sum] = i
return False
```

TC → O(N)

SC → O(N)





dict [2] = i

dic [sum] = [ ]

i+1 → j → subarray

0 → j

dic [0] = -1