Hotel Business

Raushan Prakhar

→ Maintain Register

Expanding 1000 rooms

Room No.	Occupied
1	F
2	F
3	F
4	F
5	F

① Room nos. <1-1000>

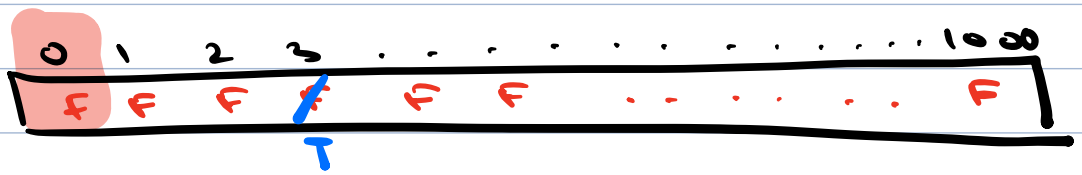
array → idx → room no.

→ value → occupancy status

bool isOccupied [1001]

Occupied → T

else → F



Status can be checked in O(1)

Update status → O(1)

② Room no. <1-n>

bool isOccupied [n+1]

idx
0 → n

③ Pandemic hit → business went down

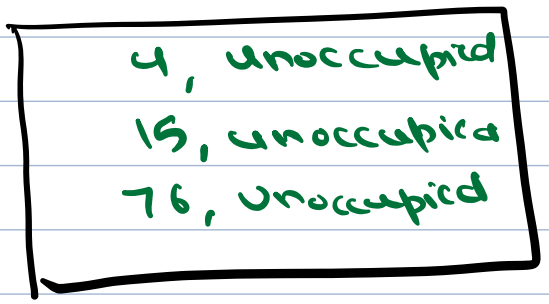
↓

1000 lucky nos. in range [1-10^9]

< 4, 15, 76, ...>

1000 ids are used of 10^9

// Hashmap is a data structure that store <key, value> pairs



Room, occupancy status
↓ key ↓ value

In Hashmap, Tc of search is O(1) and Sc is O(n)
↳ no. of entries

Hashmap <key, value>
↓ unique ↓ can be anything

// If we want to store only keys →
Hashset <key>
↳ key → unique
search ↓ O(1) in Hashset

// value - can be anything
// key - <String, int, long, float, . . . >
↓ Any primitive data type

↳ a list can't be kept as a key

1) Store population of every country

unique

Hashmap < key , value >

key → Country.

value → Population

int → 10^9

long → 10^{18}

Hashmap < String , long >

2) No. of states in each country

Hashmap < key , value >

key → Country

value → no. of states

Hashmap < String , Int >

3) For every country, we want to store all state names

Hashmap < key , value >

key → Country

value → List of state names

Hashmap <String , List <String >>

hm [key] → return value

4) For every country, store population of each state.

Hashmap < key , value >

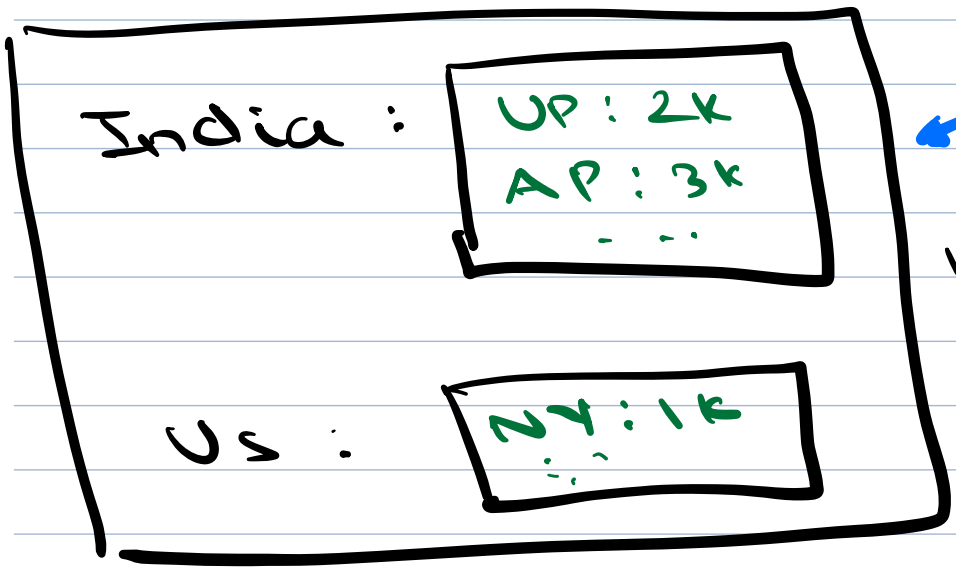
key → Country

value → Population of each state

Hashmap < String , Hashmap < String, Long > > hm

key value
State Population

Hashmap < String, Long >



	Java	C++	Python
Hashmap	Hashmap	unordered_map	dict
Hashset	Hashset	unordered_set	set

JS	C#
map	dictionary
set	hashset

Hashmap <K, V>

Hashset <K>

Ind → 2K
Us → 3K

Hashmap <key, value>

insert <key, value>
search <key>
remove <key>
update <key, new value>

size() → no. of keys in hashmap

All operations in O(1) (avg) for a single operation

Hashset <key>

insert (key)
search (key)
remove (key)

size() → no of keys in hashset

update (key) → remove (key)
↓
insert (new key)

1. Find Frequency of numbers

Given N array elements & Q queries. For each query, find frequency of given element in that query.

N = 10 ar[10] = <2, 6, 3, 8, 2, 8, 2, 3, 8, 10>

Query = 4

2 : 3
8 : 3
3 : 2
5 : 0

Idea: For every query, iterate and get count

TC: O(QN)
SC: O(1)

Iterate the array once and store info in a better manner

Frequency of elements

Hashmap <K, V>

array element — frequency

Hashmap <int, int> hm

HM
elem, freq

2, ~~1~~ ~~2~~ 3
6, 1
3, ~~1~~ 2
8, ~~1~~ ~~2~~ 3
10, 1

```
Hashmap <int, int> hm

for (i = 0 ; i < N ; i++) {          → N
    if (hm.search (A[i]) == true) {
        hm [A[i]] ++
    }
    else {
        hm.insert (A[i], 1)
    }
}

for (i = 0 ; i < Q ; i++) {          → Q
    int elem = q[i]
    if (hm.search (elem) == true) {
        print (hm [elem])
    }
    else {
        print (0)
    }
}
```

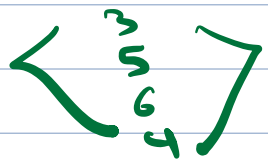
TC: O(N + Q)

SC: O(N)

10:35

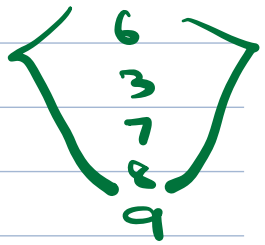
2. Given N array elements, find no. of distinct elements.

Ex 1 ar[5] : <3, 5, 6, 5, 4>
+1 +1 +1 +1



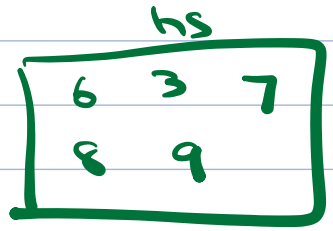
ans = 4

Ex 2 ar[7] : <6, 3, 7, 3, 8, 6, 9>
+1 +1 +1 +1 +1



ans = 5

Insert all elements in hashset.
ans = hs.size()




```
Hashset <int> hs
for (int i=0 ; i<N ; i++){
    hs.insert(A[i])
}
print (hs.size())
```

1 el → O(1)
N el → N * O(1)
TC : O(N)
SC : O(N)

Q. Does the array contains duplicate?

If all elements are unique
hs.size() == N

die

hs.size() < N

3. Given N array elements, check if there exists a pair (i, j) such that ar[i] + ar[j] = k and i != j (i,j) = (j,i)

	0	1	2	3	4	5	6	7	8
ar[] :	8	9	1	-2	4	5	11	-6	4

		i	j	
K : 6	True	2	5	ar[2] + ar[5] = 1+5=6
K : 22	False			
K : 8	True	4	8	ar[4] + ar[8] = 4+4=8

BF : Go to all pairs (i,j) and check if their sum = k

	0	1	2	3	4
A :	3	2	6	8	4

Pairs	j=0	j=1	2	3	4
i=0	0,0	0,1	0,2	0,3	0,4
i=1	1,0	1,1	1,2	1,3	1,4
i=2	2,0	2,1	2,2	2,3	2,4
i=3	3,0	3,1	3,2	3,3	3,4
i=4	4,0	4,1	4,2	4,3	4,4

i	j
0	1 → N-1
1	2 → N-1
2	3 → N-1
3	4 → N-1

```
for ( i=0 ; i<N ; i++) {
    for (j= i+1 ; j<N ; j++) {
        if (a[i] + a[j] == k)
            return true
    }
}
return false
```

TC: $O(N^2)$

SC: $O(1)$

(wrong)

Optimized : Put all elements in hashset. Iterate the array, pick an element, find its partner in hashset

	0	1	2	3	4	5	6	7	8	
A[9]:	8	9	2	-2	4	5	11	-6	4	K=9
	↓	↓	↓	↓						
	1	0	7	11	return true					
	X	X	X	✓						

$a + b = k$

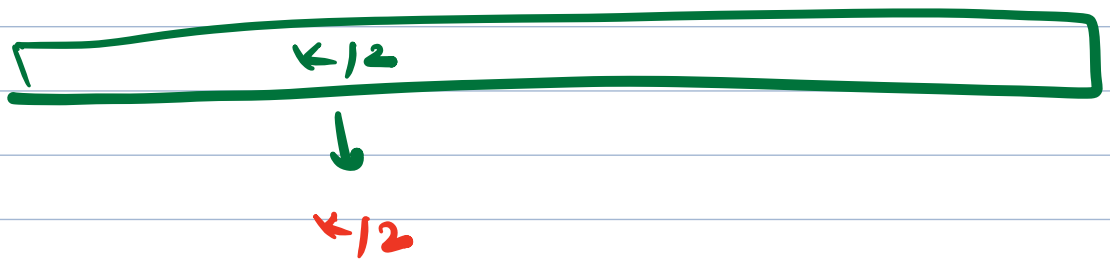
$b = k - a$

Hashset

8	9	2
-2	4	5
11	-6	

	0	1	2	3	4	5	6	7	8	
A[9]:	8	9	2	-2	4	5	11	-6	4	K=4
	↓	↓	↓							
	-4	-5	2							
	X	X	✓							

returns true (wrong)



Optimized : Insert elements in hashset on the go

(Correct)

	0	1	2	3	4	5	6	7	8	
A[] :	8	9	2	-2	4	5	11	-6	4	K=9
	↓	↓	↓	↓	↓	↓				
	1	0	7	11	5	4				
	X	X	X	X	X	✓				

return true

HS: 8 9 2 -2 4

A[i] → Partner 0 → i-1

	0	1	2	3	4	5	6	7	8	
A[] :	8	9	2	-2	4	5	11	-6	4	K=4
	↓	↓	↓							
	-4	-5	2							
	X	X	X							

HS: 8 9 2

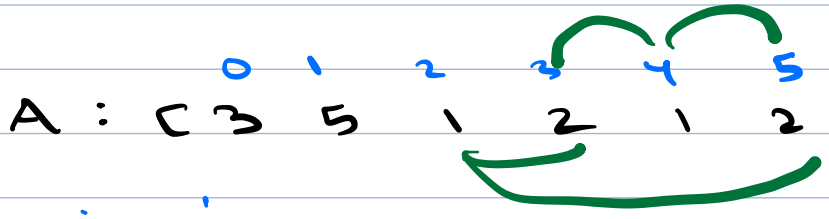
```
bool checkSum(int A[], int N, int K){
    hashset<int> hs
    for(i=0 ; i<N ; i++){
        int partner = K-a[i]
        if(hs.search(partner) == true)
            return true
        hs.insert(A[i])
    }
    return false
}
```

TC: O(N)

SC: O(N)

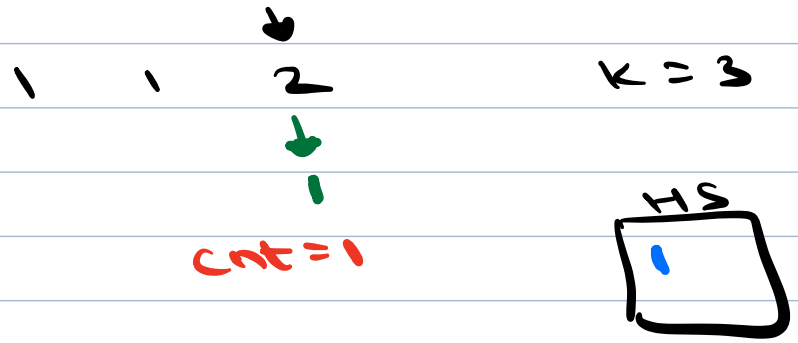
Given N array elements, counts pairs (i, j) such that a[i] + a[j] = K

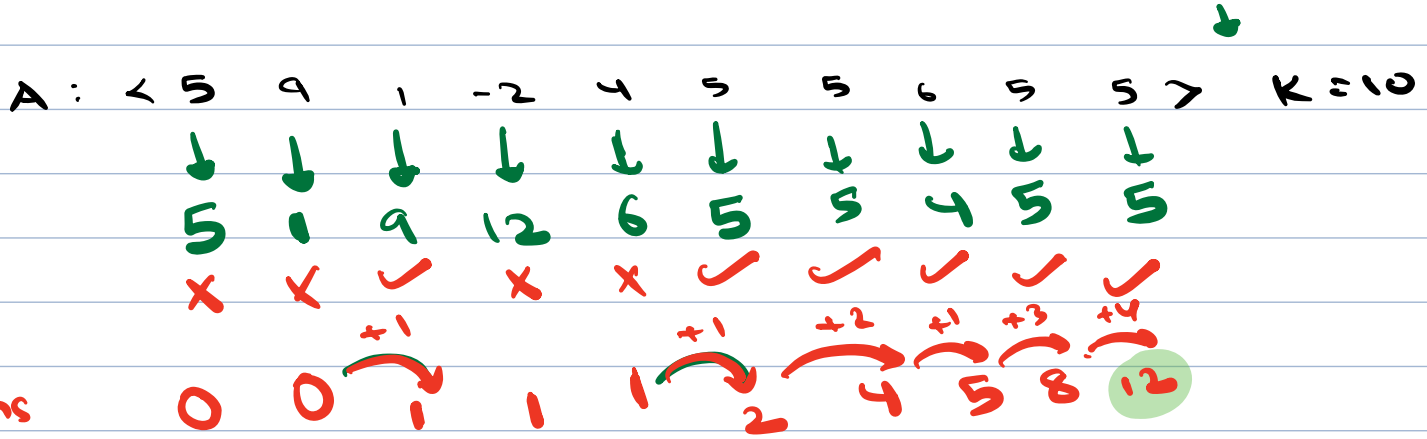
① (i != j) ② (i, j) = (j, i)



i	j
2	3
2	5
4	3
4	5

ans = 4





9, 1
5, 5
5, 5
5, 5
4, 6

HM
elem, freq
5, ~~1 2 3~~ X 5
9, 1
1, 1
-2, 1
4, 1

6, 1

```
// int a[], int n, int K

hashmap <int, int> hm
int ans = 0
for (i = 0 ; i < N ; i++) {
    int partner = K - a[i]
    if (hm.search(partner) == true) {
        ans = ans + hm[partner]
    }
    if (hm.search(A[i]) == true) {
        hm[A[i]]++
    }
    else
        hm.insert(A[i], 1)
}
return ans
```

TC : O(N)

SC : O(N)

Fri (15 Nov) Hashing

Mon (18 Nov) —

Wed (20 Nov) —

Fri (22 Nov) Break

Mon (18 Nov) Break

Wed (20 Nov) —

Fri (22 Nov) —