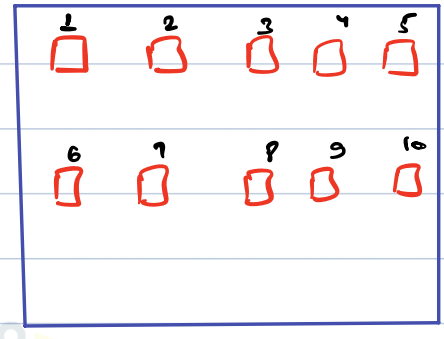
Today's agenda

- HashMap Intro
- frequency of each query
- first non-repeating elements
- HashSet
  - Number of distinct elements
  - Pair Sum == K → O(N)

// Hashmap Intro

① Aditya

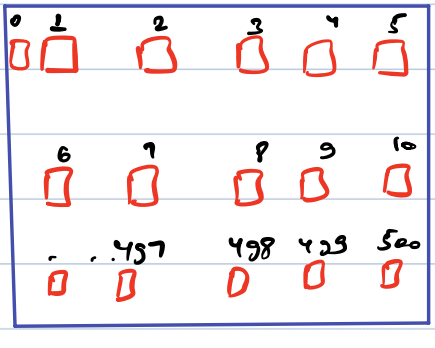


True → occupied
false → empty

boolean

n1 = true
n2 = false
n3 = . .
n4 = . .
.
.
n10 = . .

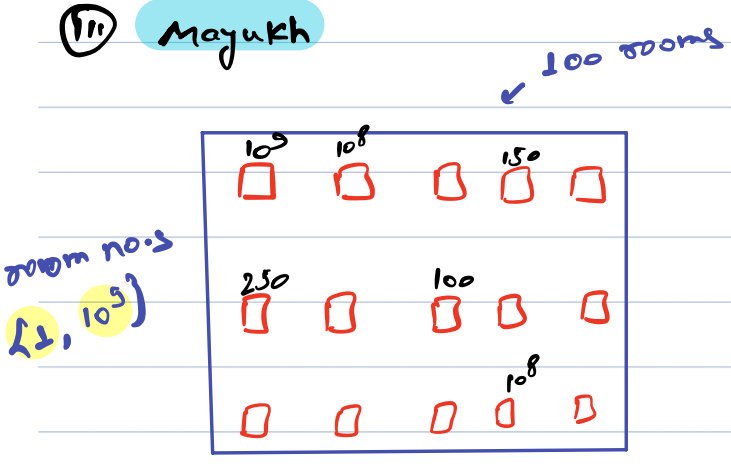
② Ajay




```
boolean arr[501];

if (arr[350] == false){
    Print ("given");
    arr[350] = true;
}
```

(11) Mayukh



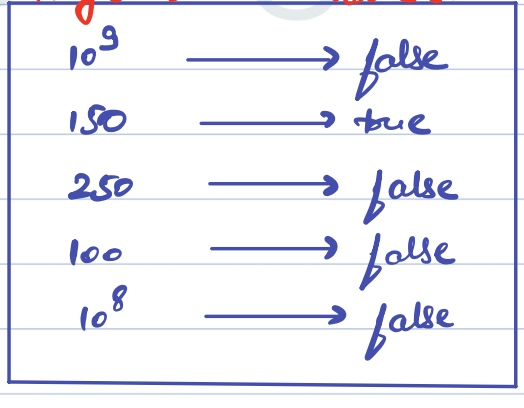
boolean arr[10^9+1]

100 rooms:

↓

To represent 100 rooms: 10^9+1

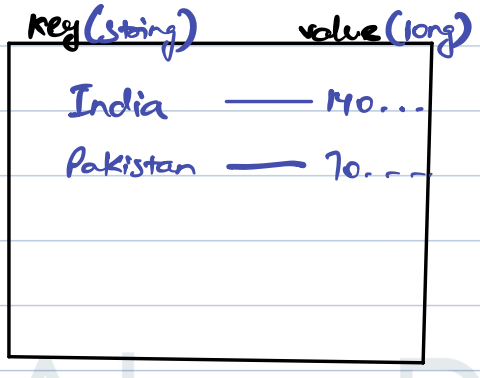
↳ Hashmap solves the issue



↳ (key, value) Pair

Q) Store Population of every country:

Key: Country name: String
value: Population: int/long



Q) Store All the School with their Principal name.

key: School name: String
value: Principal name: String

facts:

(i) Keys can be only of following type:

(or wrapper class)

int	→	Integer
long	→	Long
boolean	→	Boolean
char	→	Character
double	→	Double
String	→	String

Not allowed: arrays, objects, null

(ii) values can be of any type.

Syntax:

```
HashMap<Integer, Integer> hm = new HashMap<>();
```

(Integer = key, Integer = value, hm = name)

hm

Integer	Integer
~~10~~	~~50~~
20	~~60~~ 70
30	60

// add

```
hm.put(key, value);
hm.put(10, 50);
hm.put(20, 60);
hm.put(30, 60);
hm.put(20, 70);
```

T.C: O(1)

→ replace the older value

→ keys are unique in HashMap.

// get

```
hm.get(20); → 70
hm.get(40); → null
```

T.C: O(1)

// size

```
hm.size();
```

T.C: O(1)

// containsKey → check if key exists.

```
hm.containsKey(20); → true
hm.containsKey(40); → false
```

T.C: O(1)

//remove

hm.remove(10); → it will remove (10,50);

T.C: O(1)

// iterate on hashmap

```
int arr[n];
for(int i=0; i<n; i++){
    Print(arr[i]);
}
```

arr = [5|3|1|7]

```
for(int v: arr){
    Print(v);
}
```

iterating on keys:

T.C: O(n)

```
for(int K: hm.keySet()){

}
```

hm

Integer	Integer
10	50
30	60
40	80
60	100
50	90
20	60

ordering in hashmap is Predictable random order.

Q) find frequency

↳ Given N array elements & Q queries. for every query find frequency of element in array.
↳ no. of occurrence

Ex: arr[11]: { 2 6 3 8 2 8 2 3 8 10 6 }
(0 1 2 3 4 5 6 7 8 9 10)

queries[4] = { 2 8 3 5 }
↓ ↓ ↓ ↓
{ 3 3 2 0 }

//idea1

↳ iterate and count for every query.

T.C: O(Q*N)
↳ Q=N
O(N^2)

S.C: O(1)

//idea2

↳ create hashmap of given array.

arr[11]: { 2 6 3 8 2 8 2 3 8 10 6 }
(0 1 2 3 4 5 6 7 8 9 10) i

hm

key (Integer) val	value (Integer) c/freq
2	~~2~~ 3
6	~~1~~ 2
3	~~1~~ 2
8	~~2~~ 3
10	1

queries[4] = { 2 8 3 5 }
↓ ↓ ↓ ↓
3 3 2 0

//Pseudo Code

10 20 20 40 50 60 70 80 90

```
void PointFrequency( int arr[N], int queries[Q]){

    HashMap < Integer, Integer > hm = new HashMap<>();

    for(int i=0; i<N; i++ ) {
        if (hm.containsKey(arr[i]) == true) {
            int temp = hm.get(arr[i]);
            hm.put (arr[i], temp+1);
        }
        else {
            hm.put (arr[i], 1);
        }
    }

    for(int i=0; i<Q; i++ ) {
        if (hm.containsKey (queries[i]) == true){
            Print ( hm.get(queries[i]));
        }
        else {
            Print (0);
        }
    }
}
```

T.C: O(N) + O(Q)
= O(N+Q)
N == Q
= O(2N) = O(N)

S.C: O(N)

Tracing

arr[11]: { 2 6 3 8 2 8 2 3 8 10 6 }
0 1 2 3 4 5 6 7 8 9 10

queries[4] = { 2 8 3 5 }
+3 +3 +2 +0

temp = 2

hm

2	—	x2 3
6	—	x2
3	—	x2
8	—	x2 3
10	—	1

```
for(int i=0; i<n; i++){
    if(hm.containsKey(arr[i]) == true){
        int temp = hm.get(arr[i]);
        hm.put(arr[i], temp+1);
    }
    else{
        hm.put(arr[i], 1);
    }
}
```

```
for(int i=0; i<Q; i++){
    if(hm.containsKey(queries[i]) == true){
        Print(hm.get(queries[i]));
    }
    else{
        Print(0);
    }
}
```

Break till 9:32 Pm

Q) Find the first non-repeating elements
↳return -1 if all the elements are repeating.

Ex1: arr[6]: {1 2 3 1 2 5} → 3

arr[8]: {5 4 4 3 6 7 5 6} → 3

//idea

arr[8]: {5 4 4 3 6 7 5 6}

5 - x2
4 - x2
3 - 1
6 - x2
7 - 1

↳iterate on array and get the first element with freq 1.

// Psuedo code

```
int firstnonrepeating (int arr[N]){

    HashMap<Integer, Integer> hm = new HashMap<>();

    for(int i=0; i<N; i++){
        if(hm.containsKey(arr[i]) == true){
            int temp = hm.get(arr[i]);
            hm.put(arr[i], temp+1);
        }
        else{
            hm.put(arr[i], 1);
        }
    }

    for(int i=0; i<N; i++){
        int v = arr[i];
        if(hm.get(v) == 1){
            return v;
        }
    }
    return -1;
}
```

T.C : O(2N) ≈ O(N)

S.C : O(N)

// Hashset

↳ only the key Part of Hashmap.

```
Hashset<Integer> hs = new Hashset<>();
```

↳ random order

hs: 10 20 ~~30~~ 40

hs.size();

3

hs.add(10) → add

hs.add(20) → add

hs.add(30) → add

hs.add(40) → add

hs.add(20) → nothing

T.C: O(1)

hs.remove(30)

T.C: O(1)

hs.contains(20); → true

T.C: O(1)

Q) Given arr[N], find no. of distinct elements.

Ex: arr[5] = {4 6 7 6 5} → ans: 4

arr[5] = {10 10 10 20 20} → ans: 2

Idea

arr[5] = {10 10 10 20 20}
i

hs: | 10 20 |

ans = hs.size();

Pseudo Code

```
P S int distinct elements (int arr[N]){

    HashSet<Integer> hs = new HashSet<>();

    for (int i=0; i<N; i++){
        hs.add(arr[i]);
    }

    return hs.size();

}
```

Q) Pair Sum == K

↳ Given arr[N], check if there exists a Pair(i,j) such that arr[i] + arr[j] == K and (i != j).

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

K=11 → arr[4] + arr[8] ⇒ 4+7 = 11 ✓true

K=6 → arr[0] + arr[3] ⇒ 8 + (-2) = 6 ✓true

K=22 → ~~arr[6] + arr[6]~~ → false

//idea 1

↳ Nested loop, check all Pairs for sum == K.

T.C: $O(N^2)$

Idea 2

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

Step 1: insert all the elements in hashset.

hs: {8 9 1 -2 4 5 11 -6 7}

(i) a + b = 11

a	b	is b Present?
8	3	NO
9	2	NO
1	10	NO
-2	13	NO
4	7	yes -> return true

(ii)

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

hs: {8 9 1 -2 4 5 11 -6 7}

a + b = -4

a	b	is b Present
8	-12	NO
9	-13	NO
1	-5	NO
-2	-2	yes -> return true.

//idea 3

↳ insert all elements of array with frequency of it.

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

8 - 1	11 - 1
9 - 1	-6 - 1
1 - 1	7 - 1
-2 - 1	
4 - 1	
5 - 12	

a + b = -4

a	b	is b Present
8	-12	NO
9	-13	NO
1	-5	NO
-2	-2	yes, but freq is 1 and a == b

//Pseudo code

```
boolean PairSum (int arr[N], int K){

    HashMap < Integer, Integer > hm = new HashMap<>();

    for(int i=0; i<N; i++ ) {
        if (hm.containsKey(arr[i]) == true) {
            int temp = hm.get(arr[i]);
            hm.put (arr[i], temp+1);
        }
        else {
            hm.put(arr[i], 1);
        }
    }

    for (int i=0; i<N; i++ ) {
        int a = arr[i];
        int b = K-a;

        if ( a != b && hm.containsKey(b)==true){
            return true;
        }
        else if (a==b && hm.get(b) > 1) {
            return true;
        }
    }
```

T.C: O(N)

S.C: O(N)

return false;

}

k = -4

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

8 - 1	11 - 1
9 - 1	-6 - 1
1 - 1	7 - 1
-2 - 1	
4 - 1	
5 - 2	

a + b = -4

a	b	is b Present?
8	-12	NO
9	-13	NO
1	-5	NO
-2	-2	yes, but freq 1

```
for (int i=0; i<N; i++) {
    int a = arr[i];
    int b = k-a;

    if (a!=b && hm.containsKey(b)==true){
        return true;
    }
    else if (a==b && hm.get(b) > 1) {
        return true;
    }
}
return false;
```