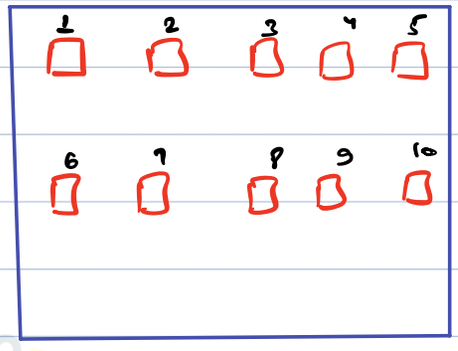
Today's agenda

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

// HashMap Intro

① Mohit:



true : occupied
false : empty

boolean n1 :

n2 :

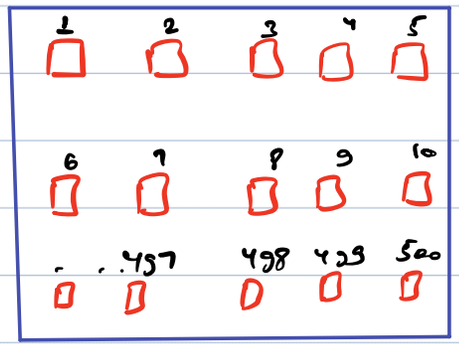
n3 : false (true)

.

.

n : 10 :

② Adarsh



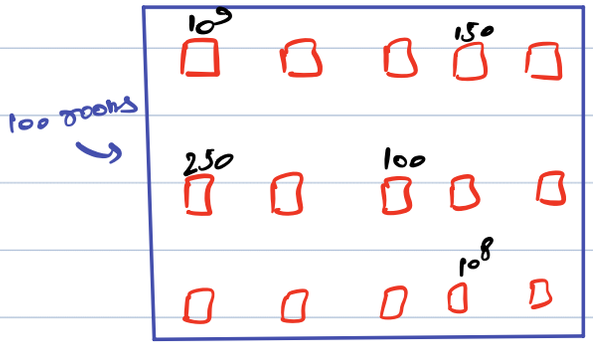
boolean arr[501]

arr[500] : true

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

(iii) Cranesham

→ {1, 10^9}

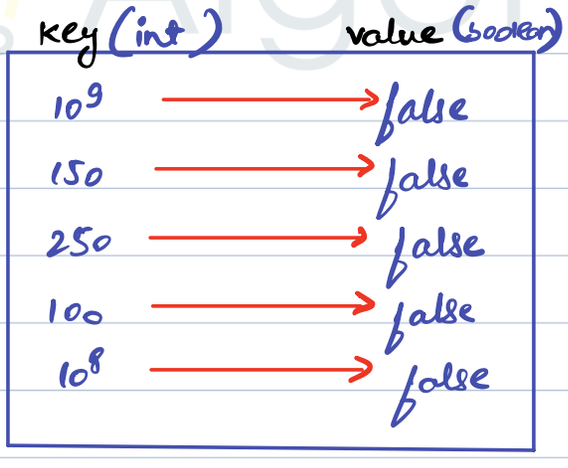


boolean arr[10^9+1]

100 rooms → 10^9 size

↳ Hashmap solves the issue

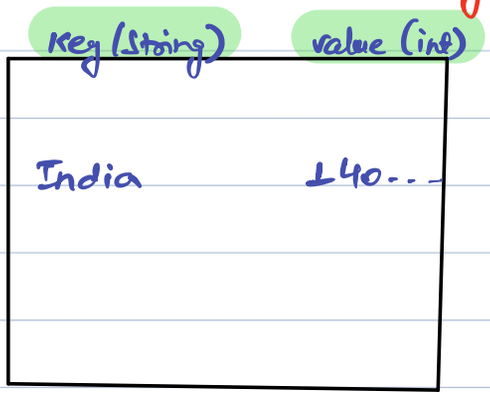
100 size + 0.99



↳ (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: String
value: String

facts:

(i) Keys can be only of following data type:

wrapper class

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

Not applicable: arrays, objects, null

(ii) values can be of any type.

Syntax:

key type ↙ value type ↙ name

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

//add

```
hm.put(10, 50);
hm.put(20, 60);
hm.put(30, 60);
hm.put(20, 70); → replace
```

T.C : O(1)

hm

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

//get

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

T.C: O(1)

//size

```
hm.size()
```

// Contains key → 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)

int arr[n];

→ for(int i=0; i<n; i++){

System.out.println(arr[i]);

}

int arr[n];

for(int val: arr){

System.out.println(val);

}

iterating on keys

↳ for(int key: hm.keySet()){

}

Q) find frequency

Ls 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 }

//idea:

Ls iterate and count for every query.

//Pseudo code

```
void PrintFrequency (int arr[N], int queries[M]){

    for (int i=0; i<M; i++){
        int val = queries[i];
        int count = 0;
        for (int j=0; j<n; j++){
            if (arr[j] == val){
                count++;
            }
        }
        System.out.println(count);
    }
}
```

T.C: O(M*N)

```
void PrintFrequency (int arr[n], int queries[m]){
    for (int i=0; i<m; i++){
        int val = queries[i];
        int count = 0;
        for (int j=0; j<n; j++){
            if (arr[j] == val){
                count++;
            }
        }
        System.out.println(count);
    }
}
```

Eg: 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 }

val = 8 count = 0

3

AlgoPrep

Idea 2

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 }

hm: Integer(key) → Integer(val) → frequency

Integer(key)	Integer(val) → frequency
2	3
6	2
3	2
8	3
10	1

3 3 2 0

T.C: O(N) + O(M)

//Pseudo Code

```
void PrintFrequency (int arr[N], int queries[M]){

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

T.C: O(N+M)

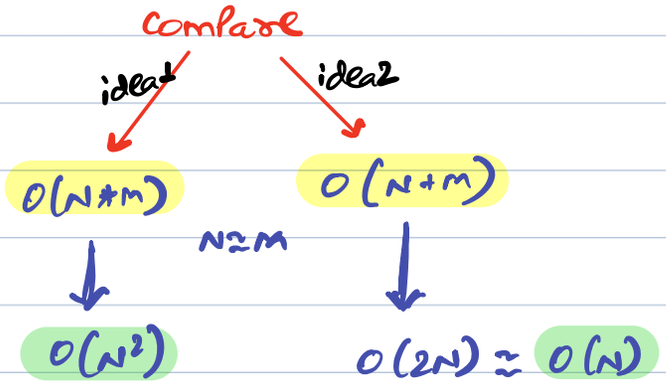
arr[11]: { 2 6 3 8 2 8 2 3 8 10 6 }
(0 1 2 3 4 5 6 7 8 9 10)

```
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);
    }
}
```

hm

Integer (key)	→	Integer (val)
2	—	~~X~~ 3
6	—	~~X~~ 2
3	—	~~X~~ 2
8	—	~~X~~ 3
10	—	1

i	arr[i]	hm.containsKey(arr[i])	temp
0	2	F	
1	6	F	
2	3	F	
3	8	F	
4	2	T	1
5	8	T	1
6	2	T	2



Break till 9:50 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}

hm

5	x2
4	x2
3	1
6	x2
7	1

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

//Pseudo Code

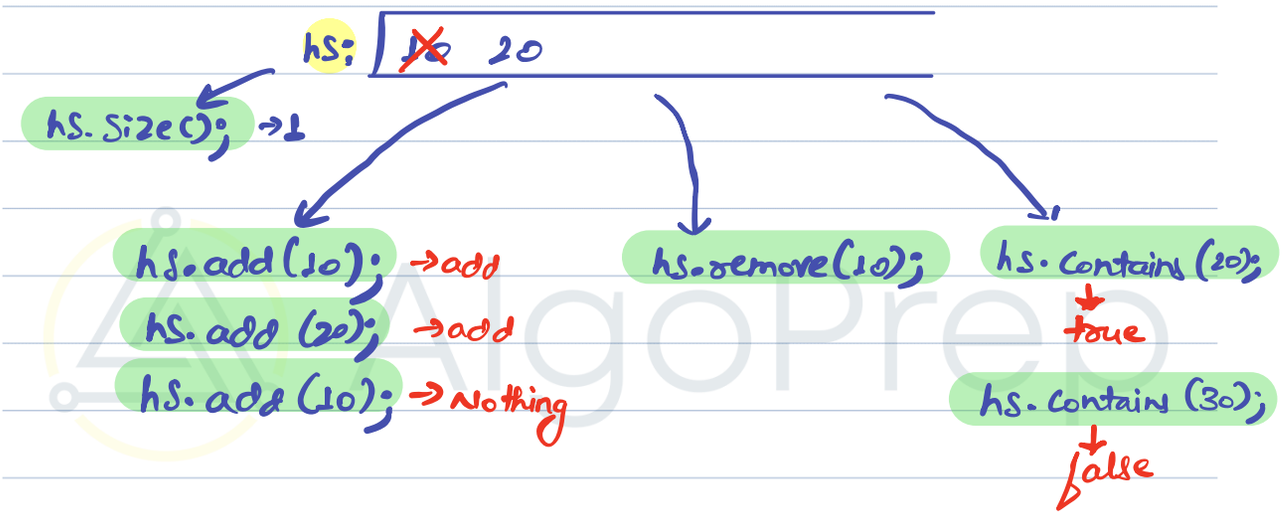
```
P S   int firstNonRepeating(int arr[N]){

    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);
        }
    }

T.C: O(2N)
 ≈O(N)

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

    return -1;
}
```

// HashSet

↳ only the key Part of hashmap

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

↳ random order



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}

hs: | 10 20 |

↳ ans = hs.size()

//Pseudo code

```
P S int distinctElements (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

//ideal

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

T.C: O(N²)

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	b is present?
8	3	NO
9	2	NO
1	10	NO
-2	13	NO
4	7	yes → true

(II) arr[10]:

	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 → true

//idea3

Ls insert all elements of array in your hashmap with frequency.

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

hm

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 at same index

//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);
        }
    }

T.C: O(N)

    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 && hm.containsKey (b) == true){
            return true;
        }
    }

    return false;
```

3

i

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

hm

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

```
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;
    }
}
```

hm.containsKey(b)==true

a+b = -4

a	b	is b Present
8	-12	NO
9	-13	NO
1	-5	NO
-2	-2	NO, same index
4	-8	yes → true