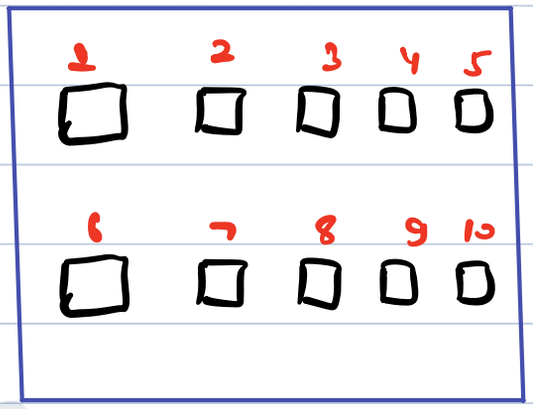
Today's agenda

- HashMap Intro
- frequency of each query
- first non-repeating elements
- HashSet
- Number of distinct elements
- Pair Sum == K → Two Sum

HashMap Intro

① Ajay



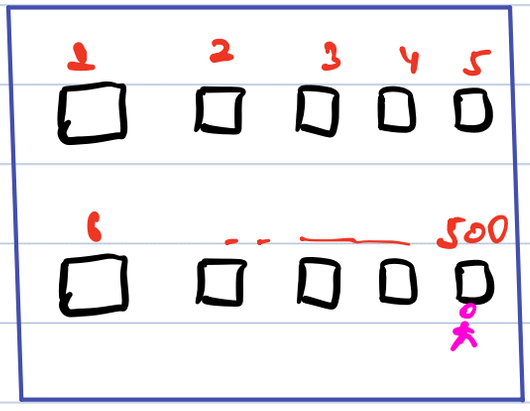
boolean b1 : true (occupied)

b2 : false (available)

.
.
.

b10 : true

② Arun



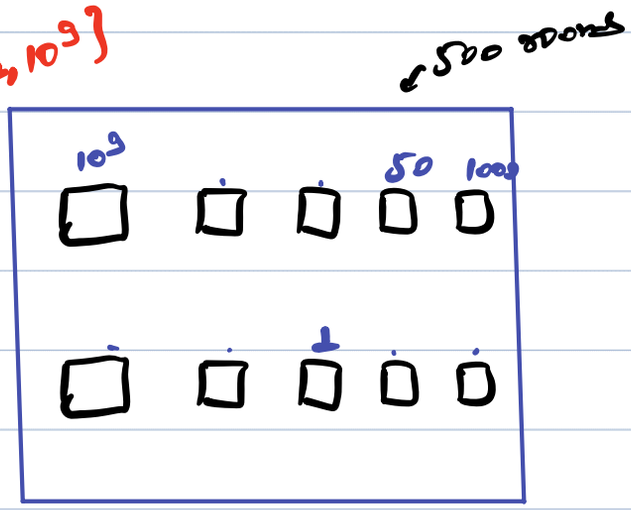
boolean arr[501];

+ 500 index will be present

arr[500] = true;

(11) Jayesh

$[1, 10^9]$

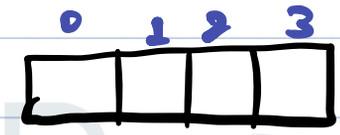


boolean arr[10^9+1];

↓

arr[50] = true

↳ space wastage



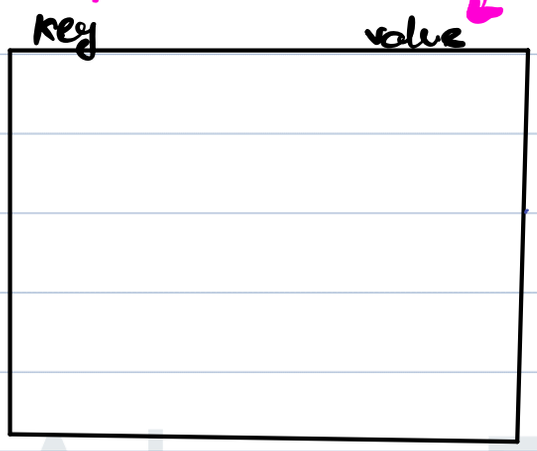
↳ HashMap solves this.

Key (int)	value (boolean)
10^9	false
50	true
1000	true
1	false

we need 500 space for 500 random no.s.

Q) Store Population of every country:

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



Q) Store All the School with their Principal name.

key: Schoolname: String
value: Principal name: String

facts:

(i) Keys can be only of following types:

wrapper class

(i) int → Integer
(ii) long → Long
(iii) boolean → Boolean
(iv) char → Character
(v) double → Double
(vi) String → String

Not allowed: Arrays, ArrayList, 2D Arrays or almost any of data structures.

(ii) Value of hashmap can be of any type.

Syntax:

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

(Integer = Key, String = value, hm = name)

add

```
hm.put(key, value);
```

```
hm.put(10, "Hello");
hm.put(20, "cello");
```


```
hm.put(10, "Kilo");   // update
```

↳ n elements

```
hm.put(30, "cello");   // values can be duplicate
```

↳ T.C of 1 add^{n}: O(1)

//size

```
hm.size();   → 3
```

T.C: O(1)

get

```
hm.get(20);   → cello
hm.get(50);   → null
```

T.C: O(1)

//containsKey

```
hm.containsKey(20);   → true
hm.containsKey(50);   → false
```

T.C: O(1)

//remove

```
hm.remove(10);   // key
```

↳ T.C: O(1)

↳ In hashmap, the order of storing key is not necessarily same as that of adding the key. [unordered]

↳ Predictable

	0	1	2	3	4
arr[5]:	10	20	20	40	50

```
for (int i=0; i<arr.length; i++){
    S.o.p (arr[i]);
}
```

for each loop

v → 10 20 30

```
for (int v: arr){
    S.o.p (v) → 10, 20
}
```

iteration on hashmap

```
for (int v: hm.keySet()){
    v → {30, 10, 20}
    hm.get(v)
}
```

T.C: O(n)

hm

Key (Integer)	Val (String)
30	Cello
10	Kilo
20	Cello

↳ n elements

Input in hashmap

↳ No need

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 no.

T.C: O(Q*N) S.C: O(1)
↳ Q==N
O(N²)

Idea2

↳ Create hashmap for given array. { no. & freq }

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 int int

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

final →

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

//Pseudo code

```
void PointFrequency(int arr[n], int query[Q]){

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

    for(int i=0; i<n; i++) {
        int temp = hm.getOrDefault(arr[i], 0);
        hm.put(arr[i], temp+1);
    }

    for(int i=0; i<Q; i++) {
        int temp = hm.getOrDefault(query[i], 0);
        S.O.P(temp);
    }

}
```

O(2n) = O(n)

n := Q

T.C: O(n+Q)

S.C: O(n)

Q) Find the first non-repeating elements

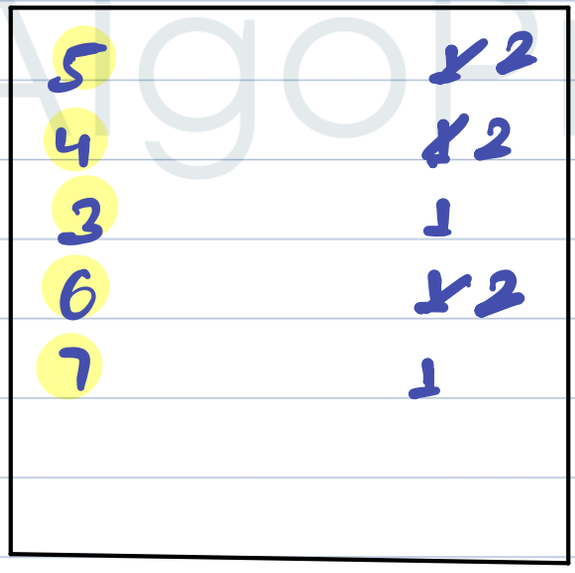
\\return -1 if all the elements are repeating.

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}



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

// Pseudo code

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

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

    for(int i=0; i<N; i++){
        int temp = hm.getOrDefault(arr[i], 0);
        hm.put(arr[i], temp+1);
    }

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

    return -1;
}
```

T.C: O(N)

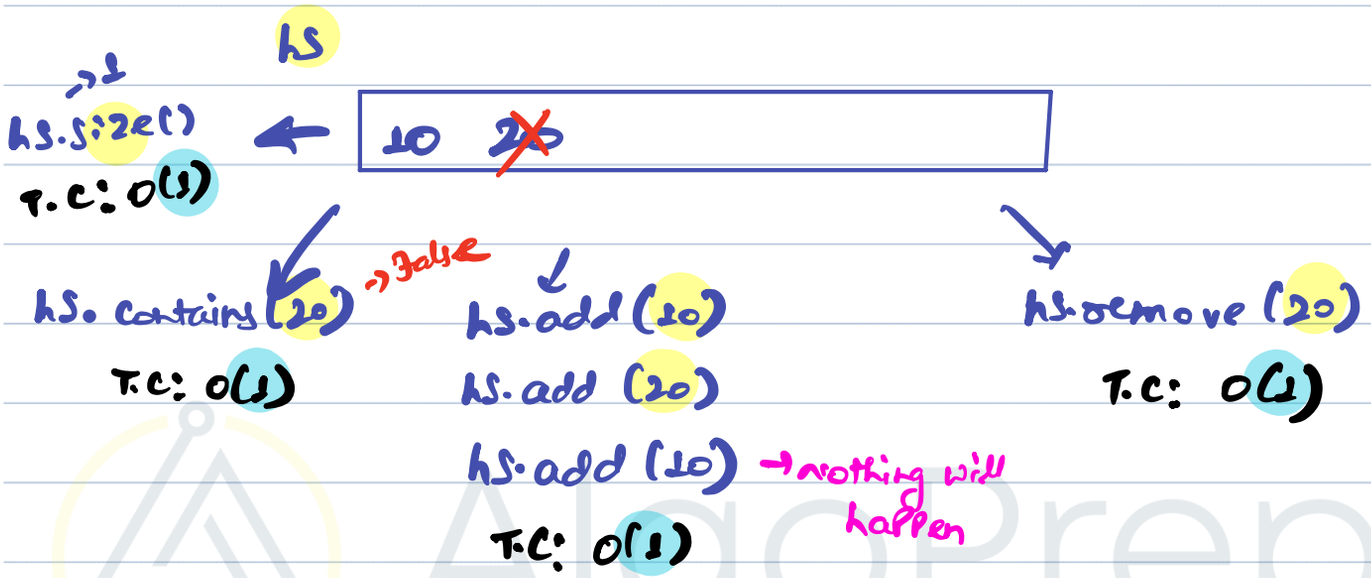
S.C: O(N)

Bolak till 10:45 Pm

// HashSet

↳ only key part of HashMap

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



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

Ex: arr[5] = {4 6 7 6 5} → 4

arr[5] = {10 10 10 20 20} → 2

//idea

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

hs: | 10 20 |

↳ hs.size()

//Pseudo code

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

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

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

    return hs.size();
}
```

T.C: O(N)

S.C: O(N)

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[6] = 4 + 7 = 11 → true

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

K = 22 arr[6] + arr[6] = 11 + 11 = 22 → false (i == j ✗)

Idea 1

↳ Nested loop.

T.C: O(N²)

Idea2 → wrong idea

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

insert all the elements to hashset.

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

k=11

① a+b = 11 b = 11-a

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

	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}

k=18

⑪ a+b = 18

a	b	is b Present?
8	10	NO
9	9	yes ✗✗

Idea 3

Using frequency map.

	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 → 1 2	

k = 18

(11)

a + b = 18

a	b	is b Present?
8	10	No
9	9	yes but freq is 1
:		

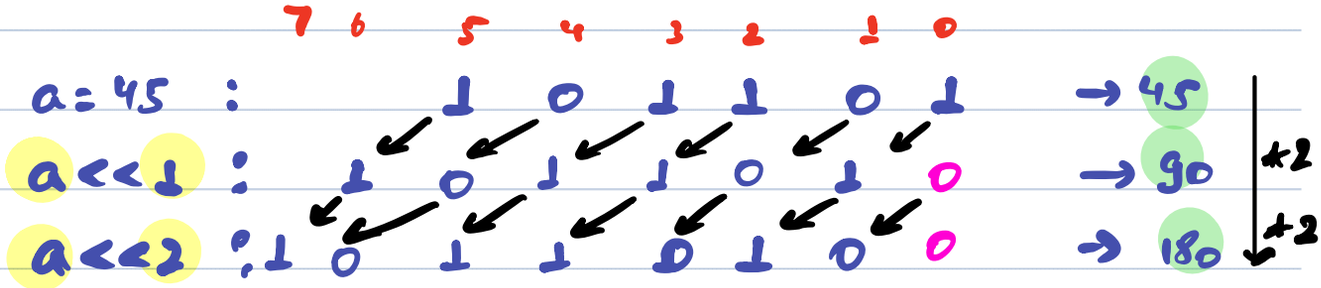
IIPSuedo Code

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

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

    for(int i=0; i<n; i++) {
        int temp = hm.getOrDefault(arr[i], 0);
        hm.put(arr[i], temp+1);
    }

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

        if (hm.containsKey(b)==true && a!=b){
            return true;
        }
        else if (a==b && hm.get(b)>=2){
            return true;
        }
    }

    return false;
}
```

T.C: O(N)

S.C: O(N)

Left shift (<<)



$$a << 1 = a * 2$$

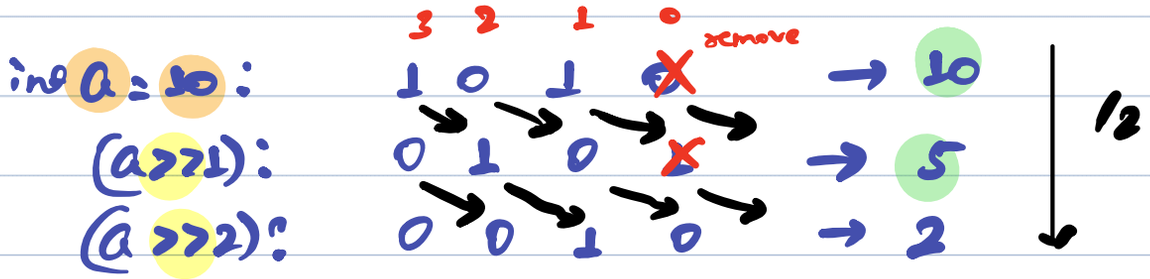
$$a << 2 = a * 2 * 2$$

$$a << 3 = a * 2 * 2 * 2$$

$$a << N = a * [2 * 2 * 2 \ldots 2] = a * 2^N$$
(N times)

$$1 << N \rightarrow 1 * 2^N = 2^N$$

Right Shift (>>)



$$a >> N = \frac{a}{2^N}$$

Code run time ↳ 1 sec

1 sec : 10^8 iterations

Constraints

↳ N: arr.length <= 10^5

0 < arr[i] <= 10^{18} → int / long ?!

↳ $O(N^2)$ → $(10^5)^2 = 10^{10}$ iter. → TLE ✗✗

↳ $O(N\sqrt{N})$ → $10^5 * \sqrt{10^5} = 10^5 * 10^{2.5} = 10^{7.5}$ ✓