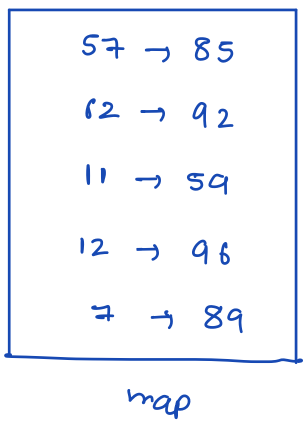
Hashmap Introduction → key - value pairs

Id	marks
57	85
62	92
11	59
12	96
7	89



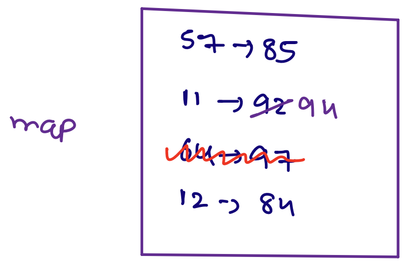
```
HashMap < Integer, Integer > map = new HashMap<>();
              ↓        ↓
             key     value
```

Country vs Population

key → String (country name)

value → Long (no. of people)

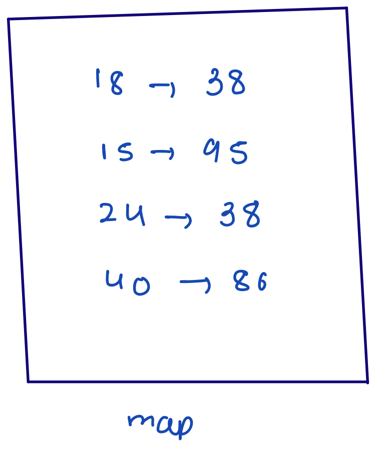
```
HashMap < String, Long > map = new HashMap<>();
```



```
void   main (  ) {

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

     map. put  (57, 85);                    map. put (key, value)

     map - put  (11,  92);

      map . put   (64, 97);

       map . put    (12,  84);

       map . put    (11,  94);


      sopln ( map. get (11)) ;          94

       sopln  ( map. get  (22)) ;        null


        sopln ( map. containsKey ( 12)) ;         true

        sopln (  map. containsKey (24)) ;         false

          map. remove (29) ;    (nothing)

            map . remove ( 64);

           sopln (map. size ( )) ;     3
```

put ( ), get ( ), containsKey( ), remove ( ), size( )

↳ O(1)



i) keys can't be duplicate

ii) Order of Insertion is not maintained

Q.1 **Count frequency**

Given an array and Q queries, find how many times a particular element is coming in array.

A : 2 1 2 3 1 5 4 2 1

Queries

ele	freq
2	3
1	3
9	0
5	1

idea1 : Go on every query, and travel the entire array to find freq. of that element.

TC : O(Q*N)

idea2 : Creating a frequency map

A : 2 1 2 3 1 5 4 2 1

$2 \rightarrow 3$
$1 \rightarrow 3$
$3 \rightarrow 1$
$5 \rightarrow 1$
$4 \rightarrow 1$

map

ele vs freq

A : 2 1 2 3 1 5 4 2 1

2 →	$\cancel{1}\ \cancel{2}\ 3$
1 →	$\cancel{1}\ \cancel{2}\ 3$
3 →	1
5 →	1
4 →	1

map

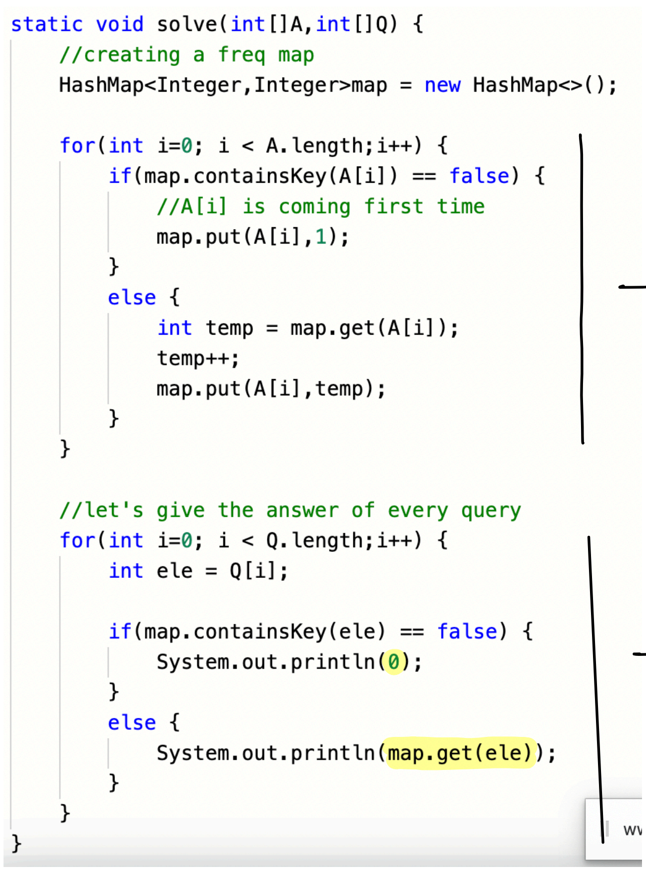
(ele vs freq)

map.containsKey (A[i])

true / false

true:
```
int t = map.get(A[i]);
t++;
map.put (A[i], t);
```

false:
```
map.put (A[i],1);
```



```java
static void solve(int[]A,int[]Q) {
    //creating a freq map
    HashMap<Integer,Integer>map = new HashMap<>();

    for(int i=0; i < A.length;i++) {
        if(map.containsKey(A[i]) == false) {
            //A[i] is coming first time
            map.put(A[i],1);
        }
        else {
            int temp = map.get(A[i]);
            temp++;
            map.put(A[i],temp);
        }
    }

    //let's give the answer of every query
    for(int i=0; i < Q.length;i++) {
        int ele = Q[i];

        if(map.containsKey(ele) == false) {
            System.out.println(0);
        }
        else {
            System.out.println(map.get(ele));
        }
    }
}
```

$\rightarrow$ N itr

$\rightarrow$ Q itr

A = { 2 1 2 3 1 2 }

Q = { 3 9 2 }

$\uparrow$

$2 \rightarrow \cancel{1}\ \cancel{2}\ 3$

$1 \rightarrow \cancel{1}\ 2$

$3 \rightarrow 1$

map

ans: 1 0 3

TC: O(N+Q)

SC: O(N)

Q·2 Given an array A[ ], find first non repeating element.

A = 2 5 4 5 2 6 ans = 4

A = 4 5 9 4 3 4 ans = 5

A = 2 5 4 5 2 6

2 →	2
5 →	2
4 →	1
6 →	1

map

i) create freq map of A[ ]

ii) travel the array and first ele with freq = 1 is ans.

TC: O(N)

SC: O(N)

code: todo

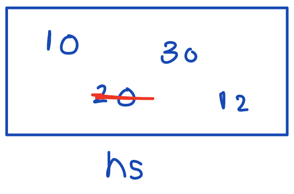
Hashset Intro → It stores only keys

↳ only unique keys (data)

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

$\downarrow$ datatype of key

```
void main() {
    HashSet <Integer> hs = new HashSet<>();
    hs.add(10);
    hs.add(20);
    hs.add(30);
    hs.add(12);

    hs.add(10);        (nothing)
    hs.add(20);        (nothing)

    SOPln( hs.contains(10) );   true
    SOPln( hs.contains(24) );   false

    hs.remove(20);
}
```



# add(), contains(), remove() → O(1)

i) keys can't be duplicate

ii) Order of Insertion is not maintained

Q.3 Given an array A[], find total no. of distinct elements.

A = 3 9 3 4 5 ans = 4

A = 3 3 3 4 4 ans = 2

A = 3 9 3 4 5

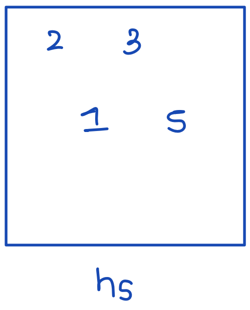
3 → 2
9 → 1
4 → 1
5 → 1

map

i) Idea 1 : use HashMap
- → create freq map of A
- → map.size() is ans

ii) Idea 2 : using HashSet
(I don't worry an ele is coming how many times)

A = 2 3 2 1 5 1



```
int   solve (int [] A) {
    HashSet<Integer> hs = new HashSet<>();
    for(int i=0; i<A.length; i++) {
            hs.add(A[i]);
    }
    return hs.size();
}
```

TC : O(N)

SC : O(N)

Q.4 Given an A[], find if it has any subarray with sum = 0. {Google}

↳ continous part of an array

$$\begin{array}{c|cccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ A: & 2 & 4 & -1 & 3 & -2 & 5 & 1 & 6 \end{array}$$

ans = true

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ A: & 2 & 4 & 3 \end{array}$$

ans = false

idea1 : TC → $O(N^2)$

```
boolean solve (int [] A) {

    int [] ps = prefixSum(A);
    int n= A.length;

    for(int s=0; s<n; s++) {                 } → TC: O(N²)
        for( int e=s; e<n; e++) {
            // sum of subarray from s to e
            - ps[e] - ps[s-1]
            if (sum == 0) {
                return true;
            }
        }
    }
    return false;
}
```

Expected TC: o(n)

$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ A = & 1 & 2 & -3 & 4 \\ PS = & 1 & 3 & 0 & 4 \end{array}$$

↱ sum 0 to i

```
if(ps[i] == 0) {
    return true;
}
```

$$\begin{array}{c|cccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ A: & 2 & 4 & -1 & 3 & -2 & 5 & 1 & 6 \\ PS: & 2 & 6 & 5 & 8 & 6 & 11 & 12 & 18 \end{array}$$

$$\begin{array}{c|cccccc} & 0 & 1 & 2 & 3 & 4 & 5 \\ A: & 4 & 3 & -5 & 1 & 1 & 9 \\ PS: & 4 & 7 & 2 & 3 & 4 & 13 \end{array}$$

i) if ps[i] == 0 then return true

ii) If values in ps[] are repeated then also return true.

	0	1	2	3	4	5	6	7
A :	2	4	-1	3	-2	5	1	6
ps :	2	6	5	8	6	11	12	18

$ps[1] = ps[4]$

$sum(0,1) = sum(0,4)$

$\cancel{sum(0,1)} = \cancel{sum(0,1)} + sum(2,4)$

$sum(2,4) = 0$

```
boolean solve ( int [ ] A ) {

    int [ ] ps = prefixSum(A);

    int n = A.length;

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

    for (int i=0; i<n; i++) {

            if( ps[i] == 0) {
                    return true;
            }
            hs.add (ps[i]);
    }

    if (hs.size() != n) {
            return true;
    }
    else {
            return false;
    }
}
```

$$\begin{array}{c|cccccc} & 0 & 1 & 2 & 3 & 4 & 5 \\ A = & 2 & 6 & 1 & -4 & 3 & 5 \\ PS = & 2 & 8 & 9 & 5 & 8 & 13 \end{array}$$

$$\boxed{\begin{matrix} 2 & 8 & 9 \\ 5 & 13 & \end{matrix}}$$

hs

return true

TC: O(N)

SC: O(N)

Doubts

sum the difference

A = [3 10 5]

	max	min	diff
{ }	0	0	0
{3}	3	3	0
{10}	10	10	0
{5}	5	5	0
{3 5}	5	3	2
{3 10}	10	3	7
{10 5}	10	5	5
{3 10 5}	10	3	7
			21

sum of diff = sum of max's − sum of min's

i) finding smax using contribution technique

A = 3 10 5

	max
{ }	0
{3}	3
{10}	10
{5}	5
{3 5}	5
{3 10}	10
{10 5}	10
{3 10 5}	10

$smax = 3 \times 1 + 10 \times 4 + 5 \times 2$

$= 53$

A = 3 10 5

↓ sort

Arrays.sort (A)

↓

$$\begin{array}{cccc} & 0 & 1 & 2 \\ A = & 3 & 5 & 10 \\ & \downarrow & \downarrow & \downarrow \\ & 1 & 2 & 4 \end{array}$$

A[i] is max in how in subseq.

A[i] is max in 2^i subseq.

```
Arrays.sort (A);
int smax = 0;

for (int i=0; i<n; i++) {

        smax += A[i] * (1<<i);
                        ↳ 2^i
}
```

find smin

	min
{ }	0
{3}	3
{10}	10
{5}	5
{3 5}	3
{3 10}	3
{10 5}	5
{3 10 5}	3

$$\begin{array}{cccc} & 0 & 1 & 2 \\ A = & 3 & 5 & 10 \end{array}$$

A[i] is min in how many subseq.   4   2   1

$smin = 3 \times 4 + 5 \times 2 + 10 \times 1$

$= 32$

write code for smin.

ans = smax − smin