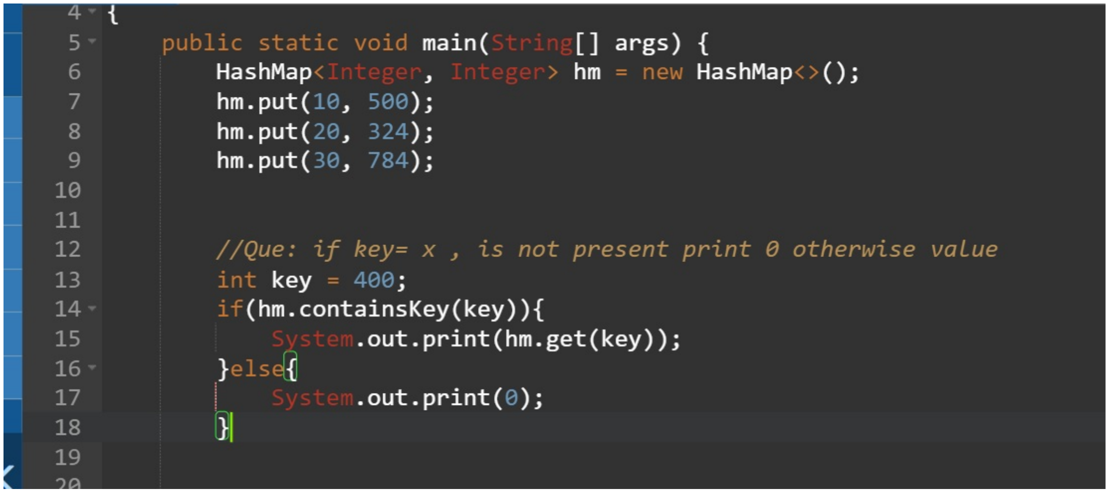
hm.get(5);

10	20
5	16



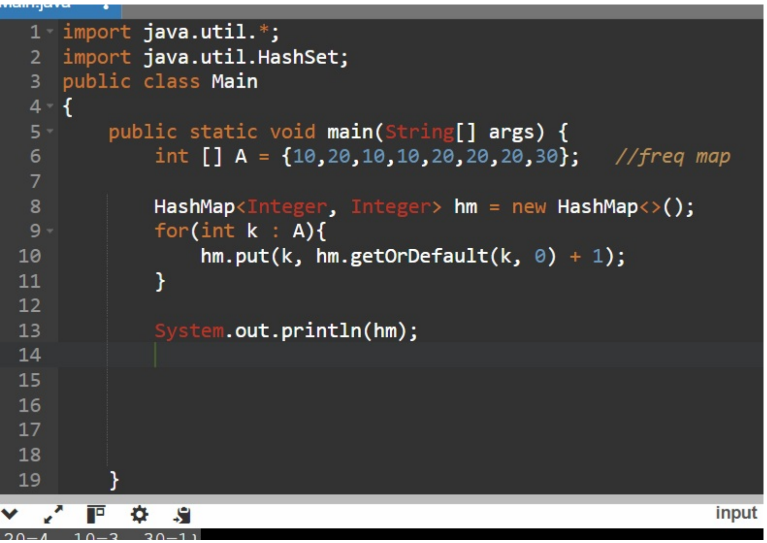
```
{
    public static void main(String[] args) {
        HashMap<Integer, Integer> hm = new HashMap<>();
        hm.put(10, 500);
        hm.put(20, 324);
        hm.put(30, 784);


        //Que: if key= x , is not present print 0 otherwise value
        int key = 400;
        if(hm.containsKey(key)){
            System.out.print(hm.get(key));
        }else{
            System.out.print(0);
        }

```

10 20 20 20 10 20 30 10

k



```java
import java.util.*;
import java.util.HashSet;
public class Main
{
    public static void main(String[] args) {
        int [] A = {10,20,10,10,20,20,20,30};    //freq map

        HashMap<Integer, Integer> hm = new HashMap<>();
        for(int k : A){
            hm.put(k, hm.getOrDefault(k, 0) + 1);
        }

        System.out.println(hm);




    }
```

```
{20=4, 10=3, 30=1}
```

10	~~1~~ ~~2~~ 3
20	4
30	1

1679. Max Number of K-Sum Pairs

You are given an integer array `nums` and an integer `k`.

In one operation, you can pick two numbers from the array whose sum equals `k` and remove them from the array.

Return *the maximum number of operations you can perform on the array.*

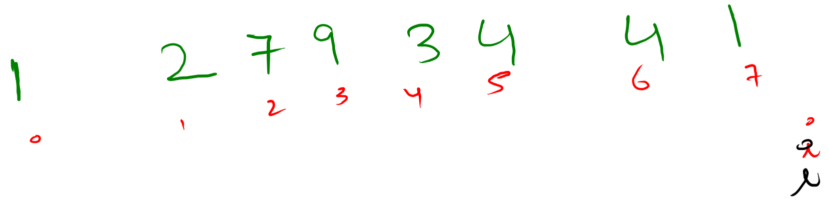
```
Input: nums = [1,2,3,4], k = 5
Output: 2
```

ans = ~~0~~ ~~1~~ ~~2~~ 3

ans = 3

k = 5

k - A[i] = rem



1	~~1~~ 0
2	~~1~~ 0
7	1
4	~~1~~ 0

1 → rem → 4

2 → rem → 3

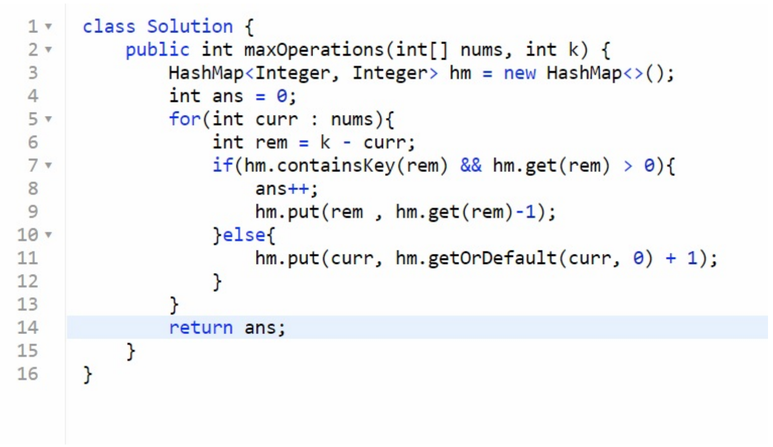
7 → rem → -2

9 → rem → -4

3 → rem → 2

4 → rem → 1

1 → rem → 4



```java
class Solution {
    public int maxOperations(int[] nums, int k) {
        HashMap<Integer, Integer> hm = new HashMap<>();
        int ans = 0;
        for(int curr : nums){
            int rem = k - curr;
            if(hm.containsKey(rem) && hm.get(rem) > 0){
                ans++;
                hm.put(rem , hm.get(rem)-1);
            }else{
                hm.put(curr, hm.getOrDefault(curr, 0) + 1);
            }
        }
        return ans;
    }
}
```

k = 6

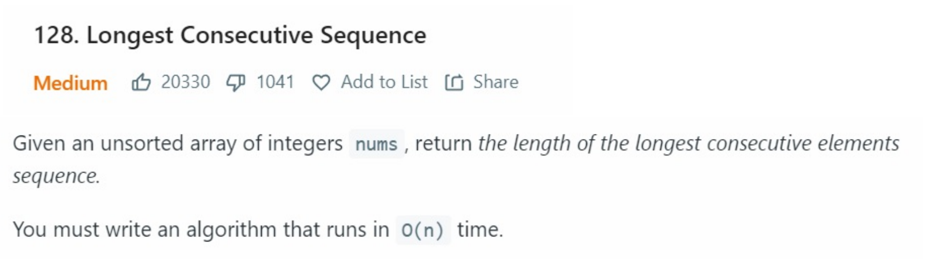
~~3~~ 1 ~~4~~ ~~2~~ ~~3~~ 3

curr

ans = ~~0~~ ~~1~~
2

rem = 3.

3	~~1~~ ~~0~~ 1
1	1
4	0



128. Longest Consecutive Sequence

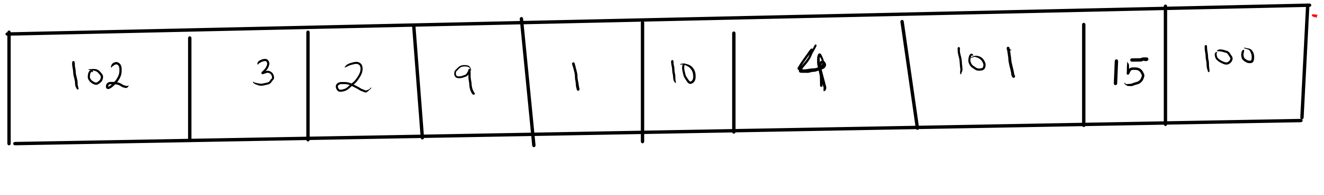
Medium 20330 1041 Add to List Share

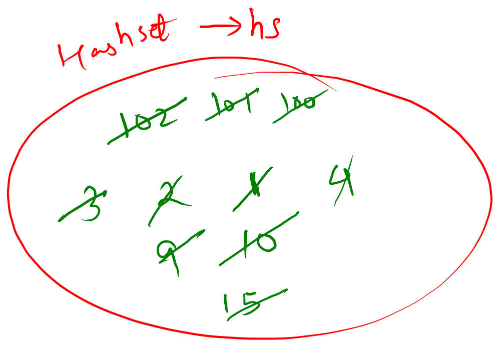
Given an unsorted array of integers `nums`, return *the length of the longest consecutive elements sequence.*

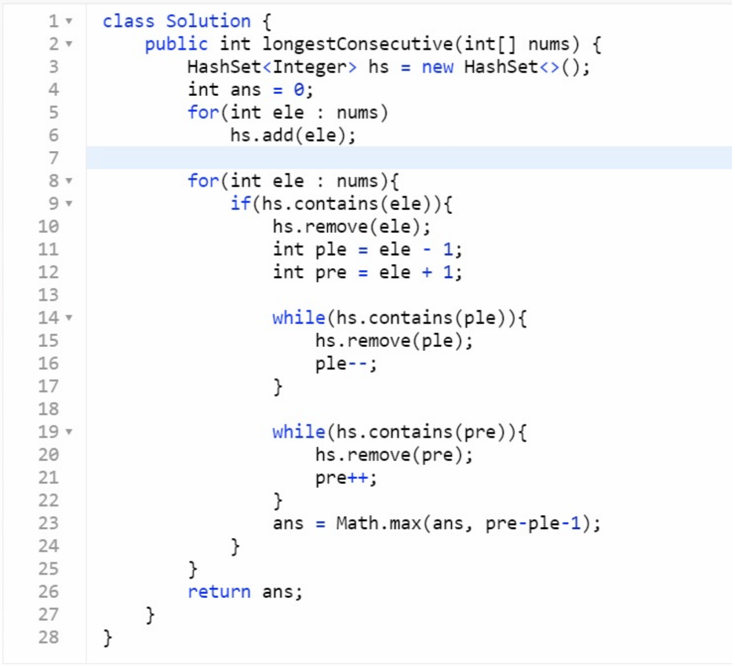
You must write an algorithm that runs in `O(n)` time.

ans = 4

ans = ~~0~~ ~~3~~ 4







```java
class Solution {
    public int longestConsecutive(int[] nums) {
        HashSet<Integer> hs = new HashSet<>();
        int ans = 0;
        for(int ele : nums)
            hs.add(ele);

        for(int ele : nums){
            if(hs.contains(ele)){
                hs.remove(ele);
                int ple = ele - 1;
                int pre = ele + 1;

                while(hs.contains(ple)){
                    hs.remove(ple);
                    ple--;
                }

                while(hs.contains(pre)){
                    hs.remove(pre);
                    pre++;
                }
                ans = Math.max(ans, pre-ple-1);
            }
        }
        return ans;
    }
}
```

12 3 5 2 7 6 11 4

~~ele~~ ~~ele~~

ele = 3

ple = ~~2~~ 1

pre = ~~4~~ ~~5~~ ~~6~~ ~~7~~ 8

ans = ~~0~~ ~~2~~ 6

$l=2$
$r=9$

$$2 \quad \boxed{3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8} \quad 9$$

$l \qquad\qquad\qquad\qquad\qquad\qquad r$

count = 8

$\boxed{r-l+1} = 2$

$r-l-1$

$9-2=7 \leftarrow$

$r-l\pm 1$

$r-l$

$4 \quad 5 \quad 6 \quad 7$

$l \qquad\qquad r$

$r-l$

$7-4=3$

3 1 2 7 8

e

$\boxed{ans = \cancel{0}\ 3}$

7 8

ple = $\cancel{7}\ \cancel{1}\ 0$,

pre = 4,,

0 $\boxed{1\ 2\ 3}$ 4

curr = 4 - 0 - 1

= $\textcircled{3}$