9th Jan 2023

FIRST:-

Majority Vote

Easy Accuracy: **38.34%** Submissions: **3K+** Points: **2**

You are given a list of integers nums where each number represents a vote to a candidate. Return the ids of the candidates that have greater than **n/3** votes, If there's not a majority vote, return -1.

```
Example 1:
Input:
n = 11
nums = [2, 1, 5, 5, 5, 5, 6, 6, 6, 6, 6]
Output:
[5,6]
Explanation:
5 and 6 occur more n/3 times.
```

```
Example 2:
Input:
n=5
nums = [1,2,3,4,5]
Output:
[-1]
```

Your Task:

You don't need to read input or print anything. Your task is to complete the function **Solve()** which takes a integer n denoting a number of element and a list of numbers and return the list of number which occur more than n/3 time.

Expected Time Complexity: O(n)
Expected Space Complexity: O(1)

Constraint:

$1 <= n <= 5 * 10^4$

$-10^9 <= nums[i] <= 10^9$

CODE SECTION:-

```
vector<int> Solve(int n, vector<int>& arr) {
        // Code here

        vector<int>v;
        int num1=0,num2=0,c1=0,c2=0;

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

            if(num1==arr[i]) c1++;
            else if(num2 == arr[i]) c2++;
            else if(c1==0) { num1 = arr[i]; c1=1;}
            else if( c2==0 ) { num2 = arr[i]; c2=1; }
            else {
                c1--;
                c2--;
            }
        }
        c1=0;c2=0;
        for(int i=0;i<n;i++){
            if(arr[i]==num1) c1++;
            if(arr[i]==num2) c2++;
        }
        if(c1>n/3)v.push_back(num1);
        if(c2>n/3)v.push_back(num2);
        if(v.size()==0){
            return {-1};
        }
        return v;
    }
```

HELP SECTION:-

1. WE can'nt have more than two majority element in an array.
2. Create two variable for two majority elements and two for their count.
3. Traverser the array and see if the majority elements found then increase their frequency otherwise check c1 or c2 is zero or not if zero then assign num1 as array[i] if c2 is zero then assign num2=arr[i].
4. Traverser the array again and check the frequency of the num1 and num2 if they are satisfying the condition then return otherwise return -1.

SECOND:-

Last modified ball :: Easy

Samwell laid out **N** bowls in a straight line and put a few marbles randomly in each bowl, i^th bowl has **A[i]** marbles. A bowl can never have more than 9 marbles at a time. A bowl can have zero marbles. Now Samwells friend adds one more marble to the last bowl, after this addition all the bowls must still be aligned with the rules mentioned above. Adding a marble follows the same rules as of addition with carryover. You are given the initial list of the number of marbles in each bowl find the position of the bowl which was last modified. It is guaranteed that there is at least one bowl which has at least one space left.

Note: Consider one-based indexing.

```
Input:
N = 4
A[] = {3, 1, 4, 5}
Output:
4
Explanation:
The last bowl has 5 marbels, we can just
add the marbel here.
```

Example 2:

```
Input:
N = 3
A[] = {1, 9, 9}
Output:
1
Explanation:
When we add the marbel to last bowl we
have to move one marbel to 2nd bowl,
to add the marbel in 2nd bowl we have
to move one marbel to 1st bowl.
Hence the last modified bowl is 1.
```

Your Task:
You don't need to read input or print anything. Your task is to complete the function **solve( )** which takes **N** and **A[ ]** as input parameters and returns the position of the last modified bowl.

Constraints:
1 ≤ N ≤ 10^5
0 ≤ A[i] ≤ 9

CODE SECTION:-

```
int solve(int N, vector<int> A) {
        // code here

        if(N==1){
            return 1;
        }

        if(A[N-1]!=9){
            return N;
        }

        else{
            for(int i=N-2;i>=0;i--) {
                if(A[i]!=9){
                    return i+1;
                }
            }
        }
    }
```

THIRD:-

Longest consecutive subsequence :: Medium

Given an array of positive integers. Find the length of the longest sub-sequence such that elements in the subsequence are consecutive integers, the **consecutive numbers can be in any order.**

Example 1:

```
Input:
N = 7
a[] = {2,6,1,9,4,5,3}
Output:
6
Explanation:
The consecutive numbers here
are 1, 2, 3, 4, 5, 6. These 6
numbers form the longest consecutive
subsquence.
```

Example 2:

```
Input:
N = 7
a[] = {1,9,3,10,4,20,2}
Output:
4
Explanation:
1, 2, 3, 4 is the longest
consecutive subsequence.
```

Your Task:
You don't need to read input or print anything. Your task is to complete the function **findLongestConseqSubseq()** which takes the array arr[] and the size of the array as inputs and returns the length of the longest subsequence of consecutive integers.

Expected Time Complexity: O(N).
Expected Auxiliary Space: O(N).

Constraints:

$1 <= N <= 10^5$

$0 <= a[i] <= 10^5$

CODE SECTION:-

```
int findLongestConseqSubseq(int arr[], int N)
   {
     //Your code here
     sort(arr,arr+N);

     int count=0;
     int max=0;
     int neww;
     for(int i=1;i<N;i++){
       if(arr[i]==arr[i-1]+1){
           count++;
           if(count>max){
               max=count;
           }
       }
       else if(arr[i]==arr[i-1]){
       }
       else{
           count=0;
       }
     }
     return max+1;
   }
```

DONE FOR TODAY