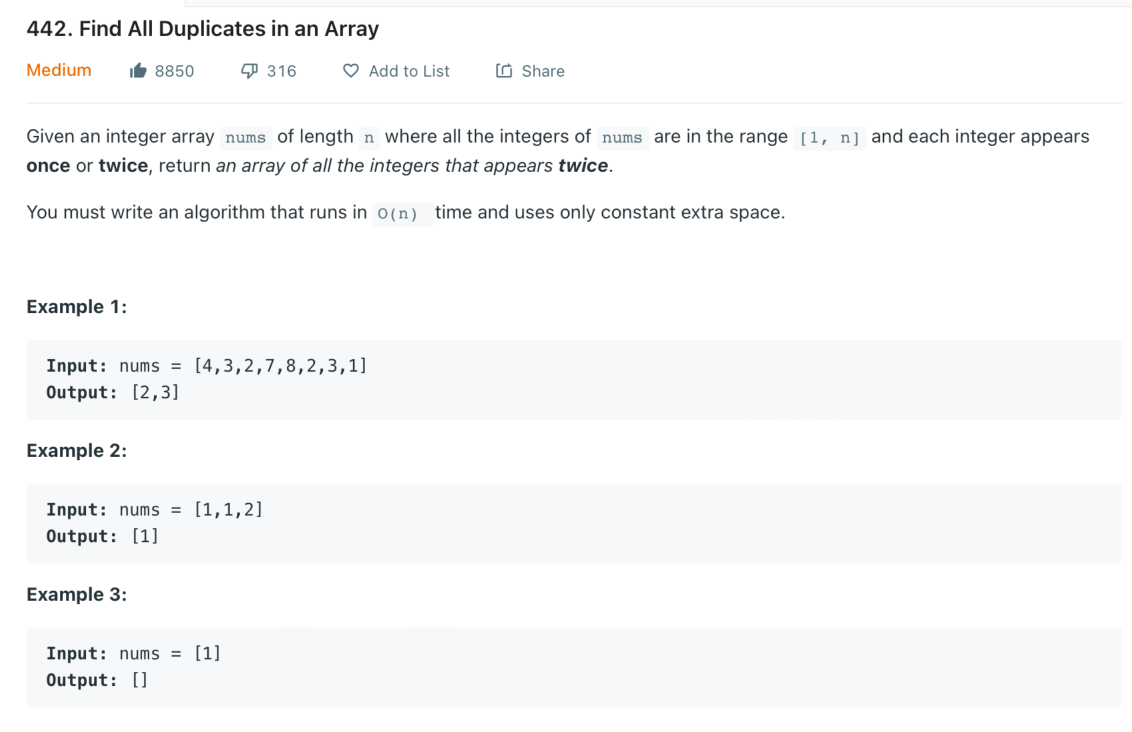
Find All Duplicates in an Array



442. Find All Duplicates in an Array

Given an integer array `nums` of length `n` where all the integers of `nums` are in the range `[1, n]` and each integer appears **once** or **twice**, return *an array of all the integers that appears* ***twice***.

You must write an algorithm that runs in `O(n)` time and uses only constant extra space.

Example 1:

```
Input: nums = [4,3,2,7,8,2,3,1]
Output: [2,3]
```

Example 2:

```
Input: nums = [1,1,2]
Output: [1]
```

Example 3:

```
Input: nums = [1]
Output: []
```

Brute force

→ Sort the array and traverse by comparing previous element if they are same then store in ans.

```
ans : { } , Sort (arr)
loop (i : 1 → n)
    check ( arr[i-1] == arr[i])
        insert ( arr[i])
```

T.C - $O(n\log n) + O(n)$

S.C - $O(n/2)$

Better approach

Using unordered map :-

→ store all element in unordered map with their frequency and who ever have freq more than 1

```
unordered map mpp , ans
for (i : 0 → n)
        mpp[arr[i]]++
for ( auto it : mpp)
        if (mpp.second > 1) ans.push (mpp.first)
}
return ans
```

T. C - O(n)

S. C - O(n)

Optimal approach

In constraint it is given that the element in Array will be less than equal to size of Array

Taking advantage of constrain we can mark element as visited element by using their value like

$$\begin{array}{ccc} 0 & 1 & 2 \\ [\,2, & 3, & 1\,] \end{array}$$

i ↗

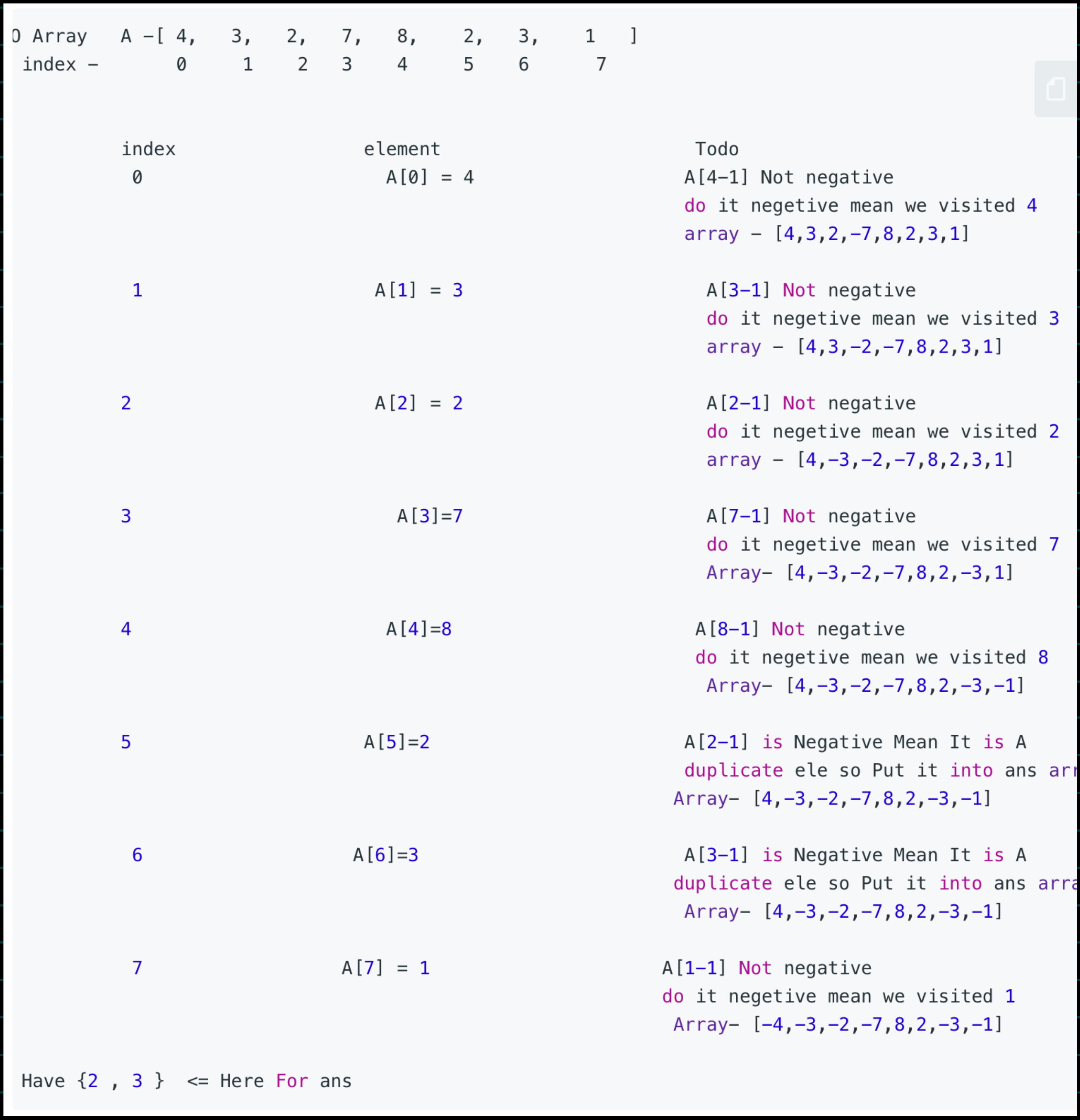
$a[2-1] = -a[2-1]$

$a[1] = -a[1]$

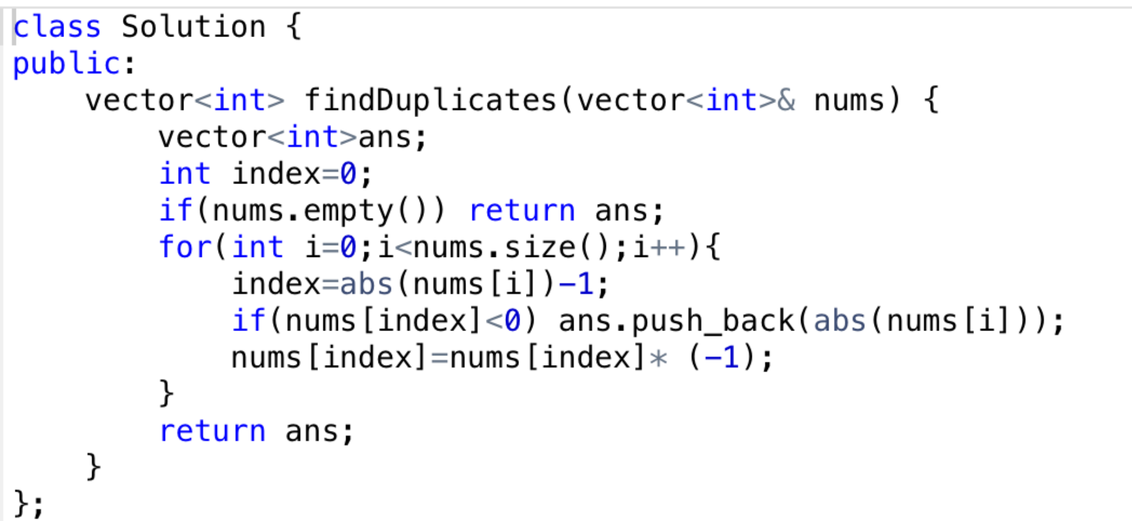
$a[1] = -3$

$[\,2, -3, 1\,]$

If we encounter 2 again then we will get -ve element and insert that elem in ans.



```
0 Array   A -[ 4,   3,   2,   7,   8,    2,   3,    1   ]
 index -       0     1    2   3    4     5    6      7


          index                 element                       Todo
           0                      A[0] = 4                   A[4-1] Not negative
                                                             do it negetive mean we visited 4
                                                             array - [4,3,2,-7,8,2,3,1]

           1                     A[1] = 3                      A[3-1] Not negative
                                                               do it negetive mean we visited 3
                                                               array - [4,3,-2,-7,8,2,3,1]

          2                      A[2] = 2                      A[2-1] Not negative
                                                               do it negetive mean we visited 2
                                                               array - [4,-3,-2,-7,8,2,3,1]

          3                        A[3]=7                      A[7-1] Not negative
                                                               do it negetive mean we visited 7
                                                               Array- [4,-3,-2,-7,8,2,-3,1]

          4                       A[4]=8                      A[8-1] Not negative
                                                              do it negetive mean we visited 8
                                                               Array- [4,-3,-2,-7,8,2,-3,-1]

          5                     A[5]=2                       A[2-1] is Negative Mean It is A
                                                             duplicate ele so Put it into ans arr
                                                            Array- [4,-3,-2,-7,8,2,-3,-1]

           6                   A[6]=3                        A[3-1] is Negative Mean It is A
                                                            duplicate ele so Put it into ans arra
                                                             Array- [4,-3,-2,-7,8,2,-3,-1]

           7                  A[7] = 1                     A[1-1] Not negative
                                                           do it negetive mean we visited 1
                                                            Array- [-4,-3,-2,-7,8,2,-3,-1]

 Have {2 , 3 }  <= Here For ans
```



```
class Solution {
public:
    vector<int> findDuplicates(vector<int>& nums) {
        vector<int>ans;
        int index=0;
        if(nums.empty()) return ans;
        for(int i=0;i<nums.size();i++){
            index=abs(nums[i])-1;
            if(nums[index]<0) ans.push_back(abs(nums[i]));
            nums[index]=nums[index]* (-1);
        }
        return ans;
    }
};
```