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: []

Constraints:

- n == nums.length
- 1 <= n <= 10^5
- 1 <= nums[i] <= n
- Each element in nums appears **once** or **twice**.

Approach:

The code finds duplicate elements in the array `nums` by counting occurrences using an auxiliary array `freq`, where `freq[i]` stores the count of `i`. It then collects elements with exactly two occurrences into a list and returns it.

Code:

```
class Solution {
    public List<Integer> findDuplicates(int[] nums) {
        int n=nums.length;
        int[] freq=new int[n+1];
        for(int num:nums)   freq[num]++;
        List<Integer> arr=new ArrayList<Integer>();
        for(int i=0;i<=n;i++){
            if(freq[i]==2)  arr.add(i);
        }
        return arr;
    }
}
```