136. Single Number

Easy | 14.1K | 553 | Companies

Given a **non-empty** array of integers `nums`, every element appears *twice* except for one. Find that single one.

You must implement a solution with a linear runtime complexity and use only constant extra space.

Example 1:

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

Example 2:

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

Example 3:

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

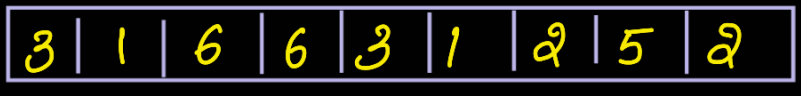
Constraints:

- `1 <= nums.length <= 3 * 10^4`
- `-3 * 10^4 <= nums[i] <= 3 * 10^4`
- Each element in the array appears twice except for one element which appears only once.

Accepted **2.2M** | Submissions **3.1M** | Acceptance Rate **71.0%**

Approach 1:

Using unordered map



unordered_map <int, int > m;

element frequency

Traverse the array and fill the unordered map.

```
for(auto i : nums)
    m[i]++;
```

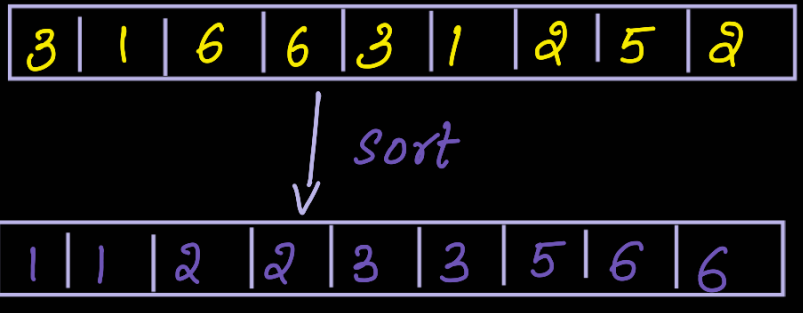
<3, 2>
<1, 2>
<6, 2>
<2, 2>
<5, 1>

Scan this map and return the key whose frequency is 1.

T : O(n)
S : O(n)

Approach 2:

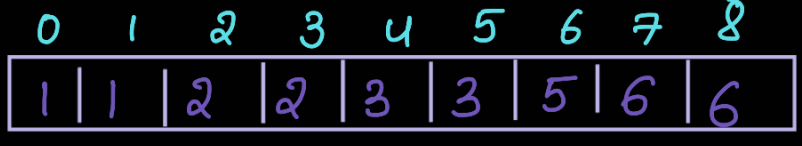
Using sorting



After sorting, start checking from index 1 and see whether

nums[i] == nums[i-1]

if its not then return nums[i-1].



i = 1

nums[1] == nums[0]

i = 3

nums[3] == nums[2]

i = 5

nums[5] == nums[4]

i = 7

nums[7] != nums[6]

So return nums[6].

O(nlogn)

Approach 3:

Using XOR

3	1	6	6	3	1	2	5	2

Single number = 3^1^6^6^3^1^2^5^2

= 5

O(n)