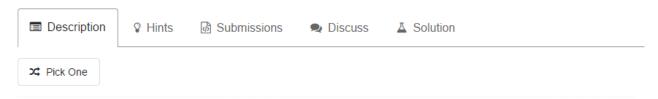
## 137. Single Number II



Given a **non-empty** array of integers, every element appears *three* times except for one, which appears exactly once. Find that single one.

## Note:

Your algorithm should have a linear runtime complexity. Could you implement it without using extra memory?

## Example 1:

Input: [2,2,3,2]
Output: 3

## Example 2:

Input: [0,1,0,1,0,1,99]

Output: 99

```
* 本道题目是利用hashmap的方式,如果value值超过了1,则把它从
* hashmap移动到filter中。
public class L137 {
     public int singleNumber(int[] nums) {
         if(nums == null || nums.length == 0)
             return -1;
         if(nums != null && nums.length == 1)
             return nums[0];
         Map<Integer, Integer> maps = new HashMap<>();
         List<Integer> filter = new ArrayList<Integer>();
         for(int i = 0; i < nums.length; i++) {</pre>
             if(i == 0){
                 maps.put(nums[i], 1);
             }else {
               if(!filter.contains(nums[i])) {
                   if(maps.get(nums[i]) == null) {
                       maps.put(nums[i], 1);
                   } else {
                       maps.put(nums[i], maps.get(nums[i]) + 1);
                   if(maps.get(nums[i]) > 1) {
                       maps.remove(nums[i]);
                       filter.add(nums[i]);
               }
           }
         }
          * maps.keySet()是一个集合(set),.toArray,变成一个数组,类型为object,
          * 加个new Integer[0],则变成一个int类型的数组,maps.keySet().
          * toArray(new Integer[0])[0]为取出第一个元素
         return maps.keySet().toArray(new Integer[0])[0];
     }
```