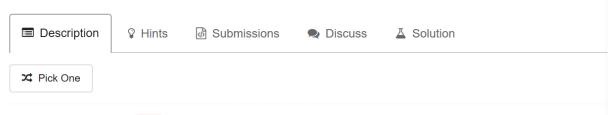
260. Single Number III



Given an array of numbers nums, in which exactly two elements appear only once and all the other elements appear exactly twice the two elements that appear only once.

Example:

```
Input: [1,2,1,3,2,5]
Output: [3,5]
```

Note:

- 1. The order of the result is not important. So in the above example, [5, 3] is also correct.
- 2. Your algorithm should run in linear runtime complexity. Could you implement it using only constant space complexity?

```
public class L260 {
    * 把nums[]中所有数一次进行^运算就能求出结果两个数的异或diff.并且nums[]中两个数
      不相同,所以两个必有某一位是不同的,所以diff必有以为是1.
    * 比如diff中最后一位是1,即可把nums[]中的数分为两组,一组同样最后以为是1,另一组最后一位
    * 不是1。这样result中的两个数一定分在不同组,最后组间进行异或运算即可。
   public int[] singleNumber(int[] nums) {
       int [] result = new int [2];
       int diff = 0;
       for (int num : nums) {
           diff ^= num;
       //nums中两个数不相同,它俩必有某一位不相同,所以diff中必有某一位是1。diff & -diff 得到结果是diff中最后一个1的位置。
       diff &= -diff;
       for(int i = 0; i < nums.length; i ++) {</pre>
           if((nums[i] & diff) == 0) {
              result[0] ^= nums[i];
           }else {
              result[1] ^= nums[i];
       return result;
   }
```

123125 diff= 315 = 011/1 [06] - 30110 diff & - diff = 0110 & 1010 = 0010 以作: 現外記載 4方板間 011 (01) あり もつ