题目描述：

Given an array of integers where 1 ≤ a[i] ≤ *n* (*n* = size of array), some elements appear twice and others appear once.

Find all the elements of [1, *n*] inclusive that do not appear in this array.

Could you do it without extra space and in O(*n*) runtime? You may assume the returned list does not count as extra space.

Example:

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

Output:
[5,6]
```

自己的解题思路：

先把元素放进一个HashSet，第二次循环时寻找元素，不在的就加入返回列表中 结果：accepted，但是空间复杂度超出了限定

题解代码：

主要思路就是：见过x这个数字，则把index=x-1这个位置上的数字置为负数，第二次循环时，看哪个是位置是正数，则这个位置所代表的数字一定是没见过的

```
public List<Integer> findDisappearedNumbers(int[] nums) {
        List<Integer> ret = new ArrayList<Integer>();

        for(int i = 0; i < nums.length; i++) {
            int val = Math.abs(nums[i]) - 1;
            if(nums[val] > 0) {
                nums[val] = -nums[val];
            }
        }

        for(int i = 0; i < nums.length; i++) {
            if(nums[i] > 0) {
                ret.add(i+1);
            }
        }
        return ret;
    }
```