给定一个只包含整数的有序数组，每个元素都会出现两次，唯有一个数只会出现一次，找出这个数。

示例 1:

输入： `[1,1,2,3,3,4,4,8,8]`
输出： `2`

示例 2:

输入： `[3,3,7,7,10,11,11]`
输出： `10`

注意: 您的方案应该在 O(log n)时间复杂度和 O(1)空间复杂度中运行。

思路，该题看类型肯定是二分查找的变种。但如何进行二分判断想了半天。最后想到方法，因为数组只包含一个唯一数，其他都重复，那么其数组长度必为奇数个，那么我们选取数组中点，将数组分为左右，中三个部分，然后判断中点是否是唯一数，不是对其左右进行判断，若是其等于左边，那么判断左右两端的长度，若是为奇数，那么左边的数组必不包含唯一数（因为若是包含，因为其长度去掉1后为偶数，若是再有一个元素为1，那么必还有一个元素唯一，那么违反了其只有一个唯一数的约束）若是为偶数，其右端必不包含唯一数，每次剔除偶数个数组，留下的数组依旧为奇数个，动态的锁定到唯一的数。

代码

```
class Solution {
public:
    int singleNonDuplicate(vector<int>& nums) {
        int i=0,j=nums.size()-1;
        while(i<j){
            int wz=(i+j)/2;
            if(nums[wz-1]!=nums[wz]&&nums[wz+1]!=nums[wz]){
                return nums[wz];
            }
            int flag=(wz-i)%2;
            if(nums[wz]==nums[wz-1]){
                if(flag==0){
                    j=wz-2;
                }else{
                    i=wz+1;
                }
```

```
            }else{
                if(flag==0){
                    i=wz+2;
                }else{
                    j=wz-1;
                }
            }
        }
        return nums[i];
    }
};
```