

1486. 数组异或操作

已解答 

简单

 相关标签

 相关企业

 提示

Aa

给你两个整数，`n` 和 `start`。

数组 `nums` 定义为：`nums[i] = start + 2*i`（下标从 0 开始）且 `n == nums.length`。

请返回 `nums` 中所有元素按位异或（**XOR**）后得到的结果。

更快速的答案：

```
class Solution {
public:
    int sumXor(int x) {
        if (x % 4 == 0) {
            return x;
        }
        if (x % 4 == 1) {
            return 1;
        }
        if (x % 4 == 2) {
            return x + 1;
        }
        return 0;
    }

    int xorOperation(int n, int start) {
        int s = start >> 1, e = n & start & 1;
        int ret = sumXor(s - 1) ^ sumXor(s + n - 1);
        return ret << 1 | e;
    }
};
```

我的答案

```
class Solution {
public:
    int xorOperation(int n, int start) {
        int i,ans=0;
        for(i=0;i<n;++i)
        {
            ans^=(start+2*i);
        };
        return ans;
    }
};
```

给你一个 **非严格递增排列** 的数组 `nums`，请你 **原地** 删除重复出现的元素，使每个元素 **只出现一次**，返回删除后数组的新长度。元素的 **相对顺序** 应该保持 **一致**。然后返回 `nums` 中唯一元素的个数。

考虑 `nums` 的唯一元素的数量为 `k`，你需要做以下事情确保你的题解可以被通过：

- 更改数组 `nums`，使 `nums` 的前 `k` 个元素包含唯一元素，并按照它们最初在 `nums` 中出现的顺序排列。`nums` 的其余元素与 `nums` 的大小不重要。
- 返回 `k`。

我的答案：

```
int nums[10] = {0, 0, 1, 1, 1, 2, 2, 3, 3, 4};
int nums2[3] = { 1, 1, 2 };
int length = sizeof(nums) / sizeof(nums[0]);
int length2 = sizeof(nums2) / sizeof(nums2[0]);
int u = 0;
int expnums[10];
for (int i = 0; i < length - 1; i++)
{
    if (nums[length - 1] == nums[length - 2])
    {
        if (nums[i] != nums[i + 1])
        {
            expnums[u] = nums[i];
            u++;
        }
    }
    else
    {
        if (nums[i] != nums[i + 1])
        {
            expnums[u] = nums[i];
            u++;
        }
        expnums[u] = nums[length-1];
    }
}

cout << "数组中大小: " << u + 1 << endl;
cout << "组成为: " << endl;
for (int k = 0; k < u+1 ; k++)
{
    cout << expnums[k] << endl;
}
```

双指针解法:

```
class Solution {
    public int removeDuplicates(int[] nums) {
        int n = nums.length;
        if (n == 0) {
            return 0;
        }
        int fast = 1, slow = 1;
        while (fast < n) {
            if (nums[fast] != nums[fast - 1]) {
                nums[slow] = nums[fast];
                ++slow;
            }
            ++fast;
        }
        return slow;
    }
}
```

给你一个整数数组 `nums` 。

如果一组数字 (i, j) 满足 `nums[i] == nums[j]` 且 $i < j$ ，就可以认为这是一组 **好数对**。

返回好数对的数目。

我的答案：

```
1 class Solution {
2 public:
3     int numIdenticalPairs(vector<int>& nums) {
4         int j,u=0;
5         j=nums.size();
6         for(int i=0;i<j;i++)
7         {
8             for(int k=i+1;k<j;k++)
9             {
10                 if(nums[i]==nums[k])
11                 {
12                     u++;
13                 }
14             };
15         };
16         return u;
17     }
```

用哈希表统计每个数在序列中出现的次数，假设数字 k 在序列中出现的次数为 v ，那么满足题目中所说的 `nums[i] = nums[j] = k (i < j)` 的 (i, j) 的数量就是 $\frac{v(v-1)}{2}$ ，即 k 这个数值对答案的贡献是 $\frac{v(v-1)}{2}$ 。我们只需要把所有数值的贡献相加，即可得到答案。

代码

C++ | Java | Python3

```
class Solution {
public:
    int numIdenticalPairs(vector<int>& nums) {
        unordered_map<int, int> m;
        for (int num: nums) {
            ++m[num];
        }

        int ans = 0;
        for (const auto &[k, v]: m) {
            ans += v * (v - 1) / 2;
        }

        return ans;
    }
};
```

给定一个整数数组 `nums` 和一个整数目标值 `target`，请你在该数组中找出 **和为目标值 `target`** 的那 **两个** 整数，并返回它们的数组下标。

你可以假设每种输入只会对应一个答案。但是，数组中同一个元素在答案里不能重复出现。

你可以按任意顺序返回答案。

```
1 class Solution {
2 public:
3     vector<int> twoSum(vector<int>& nums, int target) {
4         int n = nums.size();
5         for (int i = 0; i < n; ++i) {
6             for (int j = i + 1; j < n; ++j) {
7                 if (nums[i] + nums[j] == target) {
8                     return {i, j};
9                 }
10            }
11        }
12        return {};
13    }
14 };
```

哈希表解法：

注意到方法一的时间复杂度较高的原因是寻找 `target - x` 的时间复杂度过高。因此，我们需要一种更优秀的方法，能够快速寻找数组中是否存在目标元素。如果存在，我们需要找出它的索引。

使用哈希表，可以将寻找 `target - x` 的时间复杂度降低到从 $O(N)$ 降低到 $O(1)$ 。

这样我们创建一个哈希表，对于每一个 `x`，我们首先查询哈希表中是否存在 `target - x`，然后将 `x` 插入到哈希表中，即可保证不会让 `x` 和自己匹配。

代码


Java | C++ | C | Python3 | Golang


```
class Solution {
public int[] twoSum(int[] nums, int target) {
    Map<Integer, Integer> hashtable = new HashMap<Integer, Integer>();
    for (int i = 0; i < nums.length; ++i) {
        if (hashtable.containsKey(target - nums[i])) {
            return new int[]{hashtable.get(target - nums[i]), i};
        }
        hashtable.put(nums[i], i);
    }
    return new int[0];
}
}
```

9. 回文数

已解答 

简单

 相关标签

 相关企业

 提示

Ax

给你一个整数 `x`，如果 `x` 是一个回文整数，返回 `true`；否则，返回 `false`。

回文数是指正序（从左向右）和倒序（从右向左）读都是一样的整数。

- 例如，121 是回文，而 123 不是。

```
7 using namespace std;
8 int main()
9 {
10     int x = 121;
11     int revertedNumber = 0;
12     if (x < 0 || (x % 10 == 0 && x != 0))
13     {
14         return false;
15     }
16     while (x > revertedNumber)
17     {
18         revertedNumber = revertedNumber * 10 + x % 10;
19         x /= 10;
20     }
21     cout << (x == revertedNumber || x == revertedNumber / 10) << endl;;
22 }
23
```

27. 移除元素

简单

🏷 相关标签

🏢 相关企业

💡 提示

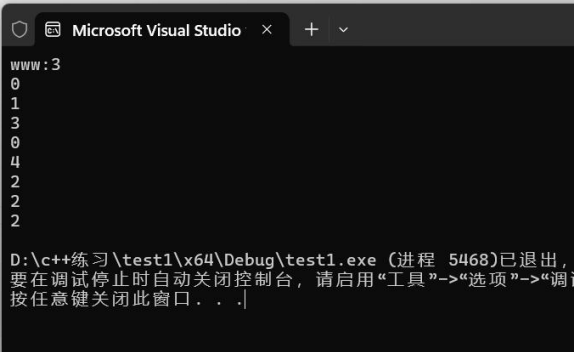
Ax

给你一个数组 `nums` 和一个值 `val`，你需要 **原地** 移除所有数值等于 `val` 的元素。元素的顺序可能发生改变。然后返回 `nums` 中与 `val` 不同的元素的数量。

假设 `nums` 中不等于 `val` 的元素数量为 `k`，要通过此题，您需要执行以下操作：

- 更改 `nums` 数组，使 `nums` 的前 `k` 个元素包含不等于 `val` 的元素。`nums` 的其余元素和 `nums` 的大小并不重要。
- 返回 `k`。

```
int nums[8] = {0,1,2,2,3,0,4,2};
int length = sizeof(nums) / sizeof(nums[0]);
int val = 2, count = 0, j=0;
for (int i = 0; i < length; i++)
{
    if (nums[i] == val)
    {
        count++;
    }
    if (nums[i] != val)
    {
        nums[j] = nums[i];
        j++;
    }
}
cout << "www:" << count << endl;
for (j; j < length; j++)
{
    nums[j] = val;
}
for (int k = 0; k < length; k++)
{
    cout << nums[k] << endl;
}
```



```
Microsoft Visual Studio
www:3
0
1
3
0
4
2
2
2
D:\c++练习\test1\x64\Debug\test1.exe (进程 5468)已退出，
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调
按任意键关闭此窗口...
```