**寻找出现两次的数**

在一个存放着数字的数组里，有一个数字出现了2次，其余的都出现了3次，如何找到那个出现了2次的数字？

这是在知乎上看到的一个很有趣的问题。看到这题我立刻就想到了另一个经典的问题，即在一堆数据里，有一个数字出现了奇数次，其他数据都出现了偶数次，如何找到那个出现了奇数次的数据？一个正常人的第一反应九成九是哈希，但是事实上这个问题存在一个非常简单、而且时间复杂度非常低的方法，即把所有的数据异或起来，最后的结果就是我们要找的数据。

这个方法乍看也许很玄妙，但其实原理很简单，即两个相同的数异或的结果是0。把所有的数据异或起来，出现了偶数次的数据都在自己和自己异或的过程中湮灭了，于是最后只剩下那个出现了奇数次的数据站在那儿茕茕孑立。

但是如果某个数组里只有一个数出现了为偶数的2次，其余的都出现了为奇数的3次应该怎么办？我一开始囿于成见想了很久怎么用异或来做这件事情，最终结果是恍然悟出我在做一件多么愚蠢的事情——异或的结果只能让我们想要找的那个数变得如羚羊挂角，无迹可求。想要不借助于哈希这种无聊的办法而得到这题的答案，必须要使用一种运算让一个数一但出现了3次，就会自我抵消掉。

下面的这个办法不是我想出来的，而是知乎上的一位牛人所给出的。即把所有的数据当作一个3进制数，然后在每一位上，把所有的数据这一位的3进制数加起来，乘以2再对3取模，最后得到的3进制数的值（也就是对应的10进制数）就是我们需要的答案。

这个巧妙算法的原理同样也很简单。一个数字如果出现了3次，那么把3个它加起来一定是3的倍数，即对3取模为0。但是如果一股脑把所有数据加起来再对3取模，我们只能得到0，1，2三种值，明显离我们的答案存在的可能性相差甚远。所以这个方法又采取了另一个巧妙的思想即把所有数据看作3进制数，然后在每一位上利用这个原理来处理。这和之前我们提到的那个题目用二进制运算异或有异曲同工之妙。

下面给出本人用VC写的一个比较笨拙的方法：

// findtwicenumber.cpp : 定义控制台应用程序的入口点。

#include "stdafx.h"

#include <iostream>

#include <vector>

#include <windows.h>

#include <math.h>

using namespace std;

int final\_bit(int &dec)

{

int finalBit = dec % 3;

dec /= 3;

return finalBit;

}

int plus(vector<int> &vec)

{

int sumValue = 0;

for(vector<int>::size\_type i = 0; i < vec.size(); i++)

{

sumValue += final\_bit(vec[i]);

cout << vec[i]<<" ";

}

cout << endl;

return (sumValue \* 2 % 3);

}

bool is\_all\_zero(const vector<int> &vec)

{

bool zero\_flag = true;

for(vector<int>::size\_type i = 0; i < vec.size(); i++)

{

if(vec[i] != 0)

{

zero\_flag = false;//不全为0

break;

}

}

return zero\_flag;

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

int array[14] = {13, 404, 64, 404, 64, 13, 64, 404, 13, 25, 25, 76, 76, 25};

vector<int> vec(array, array+14);

int result = 0;

int m = 0;

while(!is\_all\_zero(vec))

{

result += ( plus(vec) \* pow(3.0, m));

m++;

}

cout << result <<endl;

system("pause");

return 0;

}

事实上在这个问题上我们还能更进一步地思考，如果一陀数据里只有一个数出现了偶数次，其余的数都出现了奇数次，如何找到那个出现了偶数次的数据？