## 题目描述：

## 一个整型数组里除了两个数字之外，其他的数字都出现了两次。请写程序找出这两个只出现一次的数字。

## 解题思路

首先我们考虑这个问题的一个简单版本：一个数组里除了一个数字之外，其他的数字都出现了两次。请写程序找出这个只出现一次的数字。

 这个题目的突破口在哪里？题目为什么要强调有一个数字出现一次，其他的出现两次？我们想到了异或运算的性质：任何一个数字异或它自己都等于0 。也就是说，如果我们从头到尾依次异或数组中的每一个数字，那么最终的结果刚好是那个只出现一次的数字，因为那些出现两次的数字全部在异或中抵消掉了。

 有了上面简单问题的解决方案之后，我们回到原始的问题。如果能够把原数组分为两个子数组。在每个子数组中，包含一个只出现一次的数字，而其它数字都出现两次。如果能够这样拆分原数组，按照前面的办法就是分别求出这两个只出现一次的数字了。

 我们还是从头到尾依次异或数组中的每一个数字，那么最终得到的结果就是两个只出现一次的数字的异或结果。因为其它数字都出现了两次，在异或中全部抵消掉了。由于这两个数字肯定不一样，那么这个异或结果肯定不为0 ，也就是说在这个结果数字的二进制表示中至少就有一位为1 。我们在结果数字中找到第一个为1 的位的位置，记为第N 位。现在我们以第N 位是不是1 为标准把原数组中的数字分成两个子数组，第一个子数组中每个数字的第N 位都为1 ，而第二个子数组的每个数字的第N 位都为0 。

 现在我们已经把原数组分成了两个子数组，每个子数组都包含一个只出现一次的数字，而其它数字都出现了两次。因此到此为止，所有的问题我们都已经解决。\*/

## 算法描述：

class Solution {

public:

void FindNumsAppearOnce(vector<int> data,int\* num1,int \*num2) {

if (data.size()<2)

return;

int size = data.size();

int temp = data[0];

int i ,index=0;

for (i=1; i<size; i++)

temp = temp^data[i];

if (temp == 0)

return ;

while ((temp&1)!=1){

temp = temp>>1;

index++;

}

\*num1 = \*num2 =0;

for (i=0; i<size; i++){

if (IsBite(data[i],index))

\*num1^=data[i];

else

\*num2^=data[i];

}

}

bool IsBite(int num, int index)

{

num = num >>index;

return (num&1);

}

};