给定一个非空整数数组，除了某个元素只出现一次以外，其余每个元素均出现两次。找出那个只出现了一次的元素。

说明：

你的算法应该具有线性时间复杂度。 你可以不使用额外空间来实现吗？

示例 1:

输入: [2,2,1]

输出: 1

示例 2:

输入: [4,1,2,1,2]

输出: 4

===========================================================

class Solution {

public:

int singleNumber(vector<int>& nums) {

// 只有一个的情况

if (nums.size() == 1)

{

return nums[0];

}

// 冒泡排序（从大变小）

for (int i = 0; i < nums.size()-1; ++i)

{

for (int j = nums.size()-1; i < j; --j)

{

if (nums[i] > nums[j])

{

int temp = nums[i];

nums[i] = nums[j];

nums[j] = temp;

}

}

}

// 两个不相同就输出前一个

for (int i = 1; i < nums.size()-1; i = i + 2)

{

if (nums[i] != nums[i - 1])

{

return nums[i - 1];

}

}

}

};

执行出错信息：

Line 3: execution reached the end of a value-returning function without returning a value

最后执行的输入：

[4,1,2,1,2]

===========================================================

class Solution {

public:

int singleNumber(vector<int>& nums) {

if (nums.size() == 1)

{

return nums[0];

}

for (int i = 0; i < nums.size()-1; ++i)

{

for (int j = nums.size()-1; i < j; --j)

{

if (nums[i] > nums[j])

{

int temp = nums[i];

nums[i] = nums[j];

nums[j] = temp;

}

}

}

for (int i = 1; i <= nums.size(); i = i + 2)

{

// 避免溢出（数据数量必为奇数，最坏情况必为i==nums.size()）

if (nums[i] != nums[i - 1] || i == nums.size())

{

return nums[i - 1];

}

}

}

};

16 / 16 个通过测试用例

状态：通过

执行用时：1020 ms

===========================================================

class Solution {

public:

/\*\*

\* @param A : an integer array

\* return : a integer

\*/

int singleNumber(vector<int> &A) {

int x; // You'd better init x with 0. 即x = 0;

for (int i = 0; i < A.size(); i++) {

x ^= A[i];

}

return x;

}

};

因为A XOR A = 0，且XOR运算是可交换的，于是，对于实例{2,1,4,5,2,4,1}就会有这样的结果：

(2^1^4^5^2^4^1) => ((2^2)^(1^1)^(4^4)^(5)) => (0^0^0^5) => 5

就把只出现了一次的元素(其余元素均出现两次)给找出来了！

<https://www.lijinma.com/blog/2014/05/29/amazing-xor/> 感受异或的神奇

a 与 b 交换

a = a ^ b; // a = x ^ y;

b = a ^ b; // b = a ^ y = x ^ y ^ y = x ^ 0 = x;

a = a ^ b; // a = a ^ b = x ^ y ^ x = y;