# 题目

给定一个整数数组 nums，其中恰好有两个元素只出现一次，其余所有元素均出现两次。找出只出现一次的那两个元素。

示例 :

输入: [1,2,1,3,2,5]

输出: [3,5]

注意：

结果输出的顺序并不重要，对于上面的例子， [5, 3] 也是正确答案。

你的算法应该具有线性时间复杂度。你能否仅使用常数空间复杂度来实现？

**类似题目：**剑指offer 56-1

# 分析

## 方法一：哈希表

**思路：**

**代码：**

class Solution {

public:

vector<int> singleNumber(vector<int>& nums) {

unordered\_map<int,int> mp;

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

{

mp[nums[i]]++;

}

unordered\_map<int,int>::iterator it = mp.begin();

vector<int> res;

for(; it!=mp.end(); it++)

{

if(it->second==1)

res.push\_back(it->first);

}

return res;

}

};

## 方法二：位运算

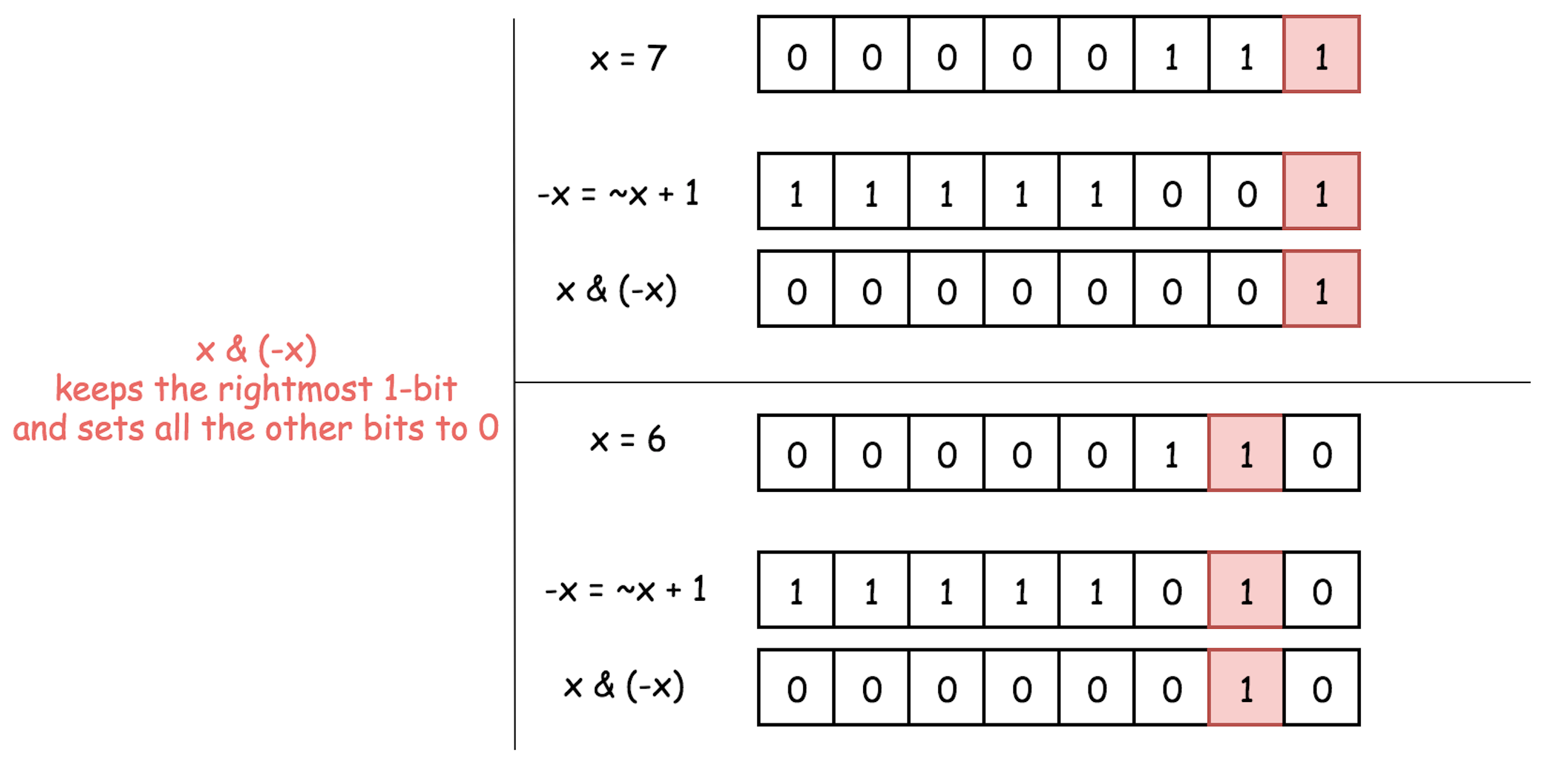
**思路：**

将使用两个按位技巧：

使用异或运算可以帮助我们消除出现两次的数字；我们计算bitmask ^= x，则bitmask留下的就是出现奇数次的位。



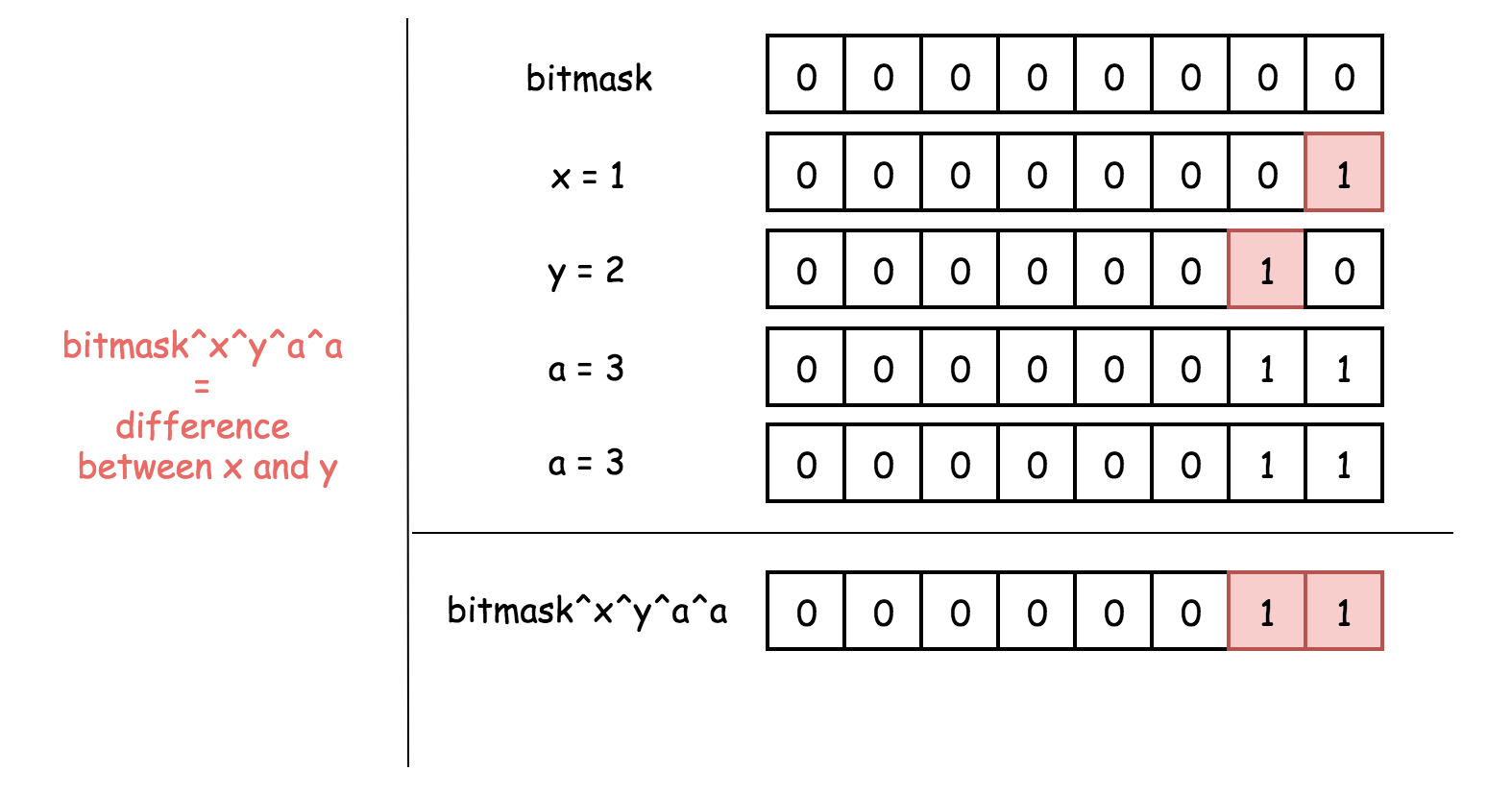
x & (-x) 是保留位中最右边1，且将其余的1设位0的方法。



算法：

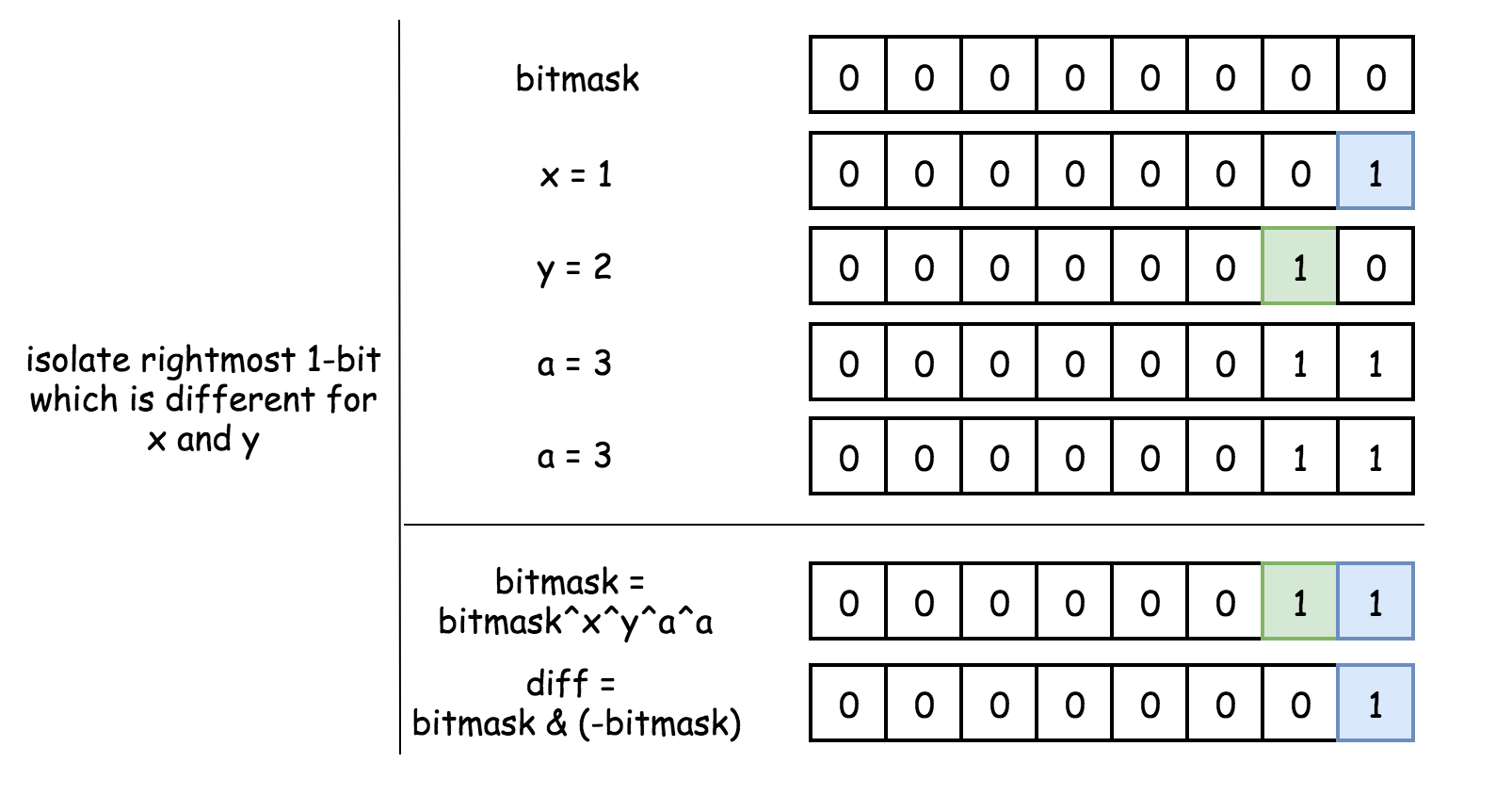
首先计算bitmask ^= x，则bitmask不会保留出现两次数字的值，因为相同数字的异或值为0。

但是bitmask会保留只出现一次的两个数字（x和y）之间的差异。



我们可以直接从bitmask中提取x和y吗？不能，但是我们可以用bitmask作为标记来分离x和y。

我们通过bitmask & (-bitmask) 保留bitmask最右边的1，这个1要么来自x，要么来自y。



当我们找到了x，那么y = bitmask^x。

**代码：**