# 题目

给定一个非负整数数组 A，A中一半整数是奇数，一半整数是偶数。

对数组进行排序，以便当 A[i]为奇数时，i 也是奇数；当 A[i]为偶数时，i也是偶数。

**示例：**

输入：[4,2,5,7]

输出：[4,5,2,7]

解释：[4,7,2,5]，[2,5,4,7]，[2,7,4,5] 也会被接受。

**提示：**

2 <= A.length <= 20000

A.length % 2 == 0

0 <= A[i] <= 1000

# 分析

## 方法一：两次遍历

思路：第一次遍历寻找奇数，第二次遍历寻找偶数

## 方法二：双指针

我们可能会被面试官要求写出一种不需要开辟额外空间的解法。

在这个问题里面，一旦所有偶数都放在了正确的位置上，那么所有奇数也一定都在正确的位子上。所以只需要关注A[0], A[2], A[4], ...都正确就可以了。

将数组分成两个部分，分别是偶数部分even = A[0], A[2], A[4], ... 和奇数部分odd = A[1], A[3], A[5], ...。定义两个指针i和j, 每次循环都需要保证偶数部分中下标i之前的位置全是偶数，奇数部分中下标j之前的位置全是奇数。

**算法**

让偶数部分下标i之前的所有数都是偶数。为了实现这个目标，把奇数部分作为暂存区，不断增加指向奇数部分的指针，直到找到一个偶数，然后交换指针i，j所指的数。

**代码：**

class Solution {

public:

vector<int> sortArrayByParityII(vector<int>& A) {

int even = 0,odd = 1,temp;

while(even < A.size()-1 && odd < A.size()){

while(even < A.size()-1 && 0 == A[even]%2)

//先判断长度然后判断数据是否为偶数，不能颠倒

even += 2;

while(odd < A.size() && 1 == A[odd]%2)

odd += 2;

if(even < A.size()-1 && odd < A.size()){

temp = A[even];

A[even] = A[odd];

A[odd] = temp;

//不申请内存，调换位置即可

}

}

return A;

}

};