# 题目

给你一个下标从 0 开始的整数数组 nums 。根据下述规则重排 nums 中的值：

按 非递增 顺序排列 nums 奇数下标 上的所有值。

举个例子，如果排序前 nums = [4,1,2,3] ，对奇数下标的值排序后变为 [4,3,2,1] 。奇数下标 1 和 3 的值按照非递增顺序重排。

按 非递减 顺序排列 nums 偶数下标 上的所有值。

举个例子，如果排序前 nums = [4,1,2,3] ，对偶数下标的值排序后变为 [2,1,4,3] 。偶数下标 0 和 2 的值按照非递减顺序重排。

返回重排 nums 的值之后形成的数组。

示例 1：

输入：nums = [4,1,2,3]

输出：[2,3,4,1]

解释：

首先，按非递增顺序重排奇数下标（1 和 3）的值。

所以，nums 从 [4,1,2,3] 变为 [4,3,2,1] 。

然后，按非递减顺序重排偶数下标（0 和 2）的值。

所以，nums 从 [4,1,2,3] 变为 [2,3,4,1] 。

因此，重排之后形成的数组是 [2,3,4,1] 。

示例 2：

输入：nums = [2,1]

输出：[2,1]

解释：

由于只有一个奇数下标和一个偶数下标，所以不会发生重排。

形成的结果数组是 [2,1] ，和初始数组一样。

提示：

1 <= nums.length <= 100

1 <= nums[i] <= 100

# 分析

## 方法一：模拟

思路：

代码：

class Solution {

public:

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

int n = nums.size();

vector<int> even, odd;

// 分别将偶数下标和奇数下标的值存储到两个数组中

for (int i = 0; i < n; ++i) {

if (i % 2 == 0) {

even.push\_back(nums[i]);

} else {

odd.push\_back(nums[i]);

}

}

// 对偶数下标的值进行非递减排序

sort(even.begin(), even.end());

// 对奇数下标的值进行非递增排序

sort(odd.begin(), odd.end(), greater<int>());

// 将排序后的值按顺序放回原数组的对应位置

for (int i = 0, j = 0, k = 0; i < n; ++i) {

if (i % 2 == 0) {

nums[i] = even[j++];

} else {

nums[i] = odd[k++];

}

}

return nums;

}

};