# 题目

给你一个无序的数组 nums, 将该数字 原地 重排后使得 nums[0] <= nums[1] >= nums[2] <= nums[3]...。

示例:

输入: nums = [3,5,2,1,6,4]

输出: 一个可能的解答是 [3,5,1,6,2,4]

# 分析

## 解法一：排序

一个显而易见的解法是先将数组排序，再从第二个元素开始逐对交换元素的位置。如：

[1, 2, 3, 4, 5, 6]

↑ ↑ ↑ ↑

swap swap

=> [1, 3, 2, 5, 4, 6]

代码：

public void wiggleSort(int[] nums) {

Arrays.sort(nums);

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

swap(nums, i, i + 1);

}

}

private void swap(int[] nums, int i, int j) {

int temp = nums[i];

nums[i] = nums[j];

nums[j] = temp;

}

复杂度分析

时间复杂度：O(nlogn)。

算法的时间开销由排序过程决定，其时间复杂度为O(nlogn)。

空间复杂度：O(1)。空间复杂度取决于排序的实现，通常而言，如果使用 堆排序，只需要O(1)。

## 方法二：一遍交换

可以只用一遍完成任务。当我们遍历整个数组，比较当前元素与下一个元素。若顺序不正确，则交换之。

代码：

public void wiggleSort(int[] nums) {

boolean less = true;

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

if (less) {

if (nums[i] > nums[i + 1]) {

swap(nums, i, i + 1);

}

} else {

if (nums[i] < nums[i + 1]) {

swap(nums, i, i + 1);

}

}

less = !less;

}

}

我们可以通过将条件压缩到一行来简化代码。注意 less 的布尔值事实上取决于索引的奇偶性。

public void wiggleSort(int[] nums) {

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

if (((i % 2 == 0) && nums[i] > nums[i + 1])

|| ((i % 2 == 1) && nums[i] < nums[i + 1])) {

swap(nums, i, i + 1);

}

}

}

下面是另一个令人惊讶的解法

public void wiggleSort(int[] nums) {

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

if ((i % 2 == 0) == (nums[i] > nums[i + 1])) {

swap(nums, i, i + 1);

}

}

}

复杂度分析

时间复杂度：O(n)。

在最坏的情况下，我们最多交换了n /2次。例如输入为 [2,1,3,1,4,1]。

空间复杂度 : O(1)O(1)。