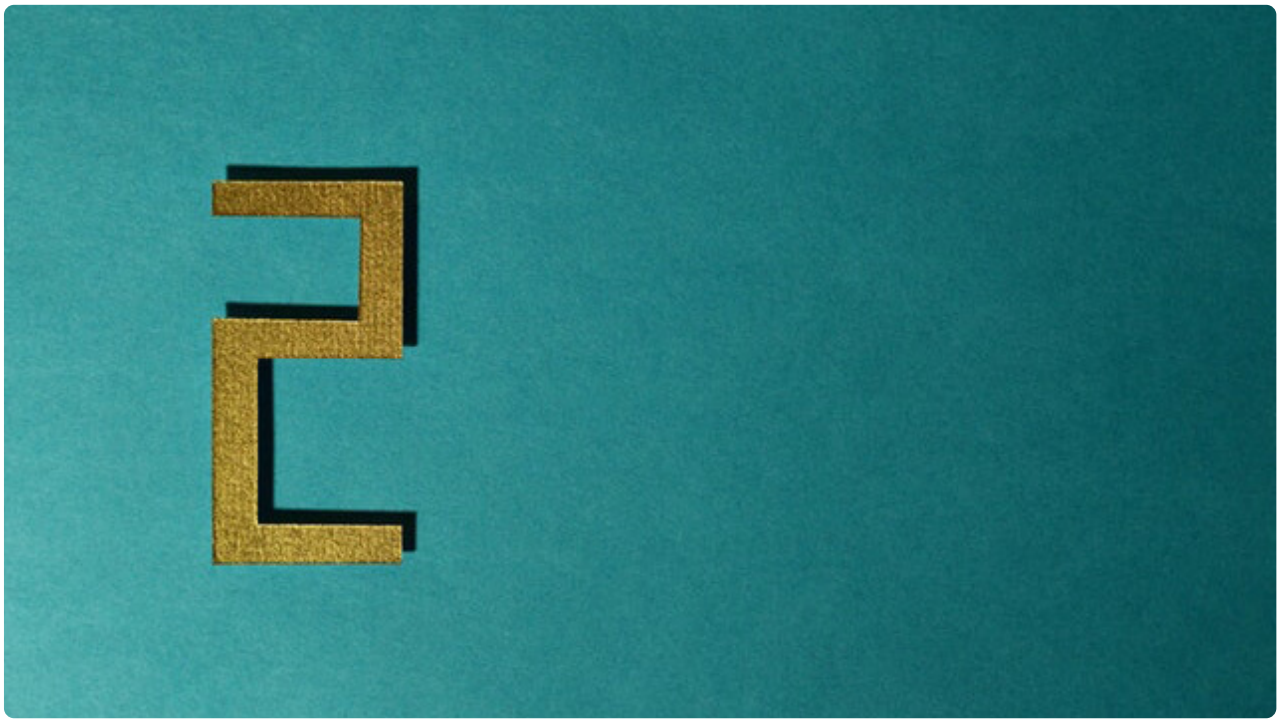
28 摆动排序

更新时间：2019-09-16 10:31:37



> 今天应做的事没有做，明天再早也是耽误了。
>
> ——裴斯泰洛齐

刷题内容

> 难度: **Medium**
>
> 题目链接：https://leetcode-cn.com/problems/wiggle-sort/ (会员题)
>
> 题目链接：https://leetcode-cn.com/problems/wiggle-sort-ii/

题目描述

给定一个无序数组，在原数组中按照
1、nums[0] <= nums[1] >= nums[2] <= nums[3]....
2、nums[0] < nums[1] > nums[2] < nums[3]....
这种次序进行排序，有无限多种排序方式，输出一种即可

例子:

输入: nums = [3,5,2,1,6,4]
输出: 其中一种输出 [3,5,1,6,2,4]

解题方案

思路1 时间复杂度: O(NlgN) 空间复杂度: O(N)

以下的次序1和次序2分别指代我们wiggle sort和wiggle sort 2需要我们返回的次序种类

观察排序的次序，我们可得次序1是次序2的特例。只要我们构造出数组2，肯定也能满足数组1，我们尝试通过构造数组来达到次序2。

假设nums有序，将nums拆成两个数组：(设n为数组nums的大小)

1：[nums[n-1],nums[n-2],…,nums[n/2+1]]

2：[nums[n/2],nums[n/2-1],…,nums[0]]

将第一个数组插空到第二个数组中

形成数组：nums[n/2],nums[n-1],nums[n/2-1],nums[n-2],…

则该数组位满足条件的数组

证明这个结论，只需要证明：

1、如果x是nums中的最小值，那么x的个数不会超过n/2+1(n为奇数)，n/2(n为偶数)

2、如果x不是nums中的最小值，那么x的个数不会超过n/2

上面两点的证明很简单，如果个数超过，那么至少会有两个相邻数相等

所以，nums[n/2]<nums[n-1]，nums[n-1]>nums[n/2-1]，。。。

次序2构造完毕，相同的代码，次序1肯定能满足

python beats 64.84 %

```python
class Solution(object):
    def wiggleSort(self, nums):
        nums.sort()
        half = len(nums[::2])
        nums[::2], nums[1::2] = nums[:half][::-1], nums[half:][::-1]
```

c++ beats 64.84 %

```cpp
class Solution {
public:
    void wiggleSort(vector<int>& nums) {
        //复制一个新数组来辅助排序
        vector<int> temp(nums);
        sort(temp.begin(), temp.end());
        int k = temp.size() - 1;
        for (int i = 1;i < nums.size();i += 2) {
            nums[i] = temp[k--];
        }
        for (int i = 0;i < nums.size();i += 2) {
            nums[i] = temp[k--];
        }
    }
};
```

java beats 100%

```
class Solution {
    public void wiggleSort(int[] nums) {
        int[] temp = Arrays.copyOf(nums, nums.length);
        Arrays.sort(temp);
        int k = temp.length - 1;
        for (int i = 1;i < nums.length;i += 2) {
            nums[i] = temp[k--];
        }
        for (int i = 0;i < nums.length;i += 2) {
            nums[i] = temp[k--];
        }
    }
}
```

go beats 74.29%

```
func wiggleSort(nums []int)  {
 temp := make([]int, len(nums))
 copy(temp, nums)
 sort.Ints(temp)
 k := len(temp) - 1
 for i := 1;i < len(nums);i += 2 {
  nums[i] = temp[k]
  k--
 }
 for i := 0;i < len(nums);i += 2 {
  nums[i] = temp[k]
  k--
 }
}
```


思路2 时间复杂度: O(N) 空间复杂度: O(1)

次序1有一种特殊的解法：观察到

1. 如果i是奇数，nums[i] >= nums[i - 1]
2. 如果i是偶数，nums[i] <= nums[i - 1]

所以我们只要遍历一遍数组，把不符合的情况交换一下就行了。

具体来说，

如果nums[i] > nums[i - 1]， 则交换以后肯定有nums[i] <= nums[i - 1]

假设nums[i]>=nums[i-1]，那么nums[i+1]如果大于nums[i]，这两个数需要交换，交换后仍然有nums[i+1]>nums[i-1]

python beats 64.16%

```
class Solution:
    def wiggleSort(self, nums: List[int]) -> None:
        for i in range(1, len(nums)):
            if (i & 1 == 0 and nums[i] > nums[i-1]) or (i & 1 != 0 and nums[i] < nums[i-1]):
                nums[i], nums[i-1] = nums[i-1], nums[i]
```

c++ beats 93.25%

```
class Solution {
public:
    void wiggleSort(vector<int>& nums) {
        for (int i = 0;i < (int)nums.size() - 1;i++) {
            if ((!(i & 1) && nums[i] > nums[i + 1])
                || ((i & 1) && nums[i] < nums[i + 1])) {
                int temp = nums[i];
                nums[i] = nums[i + 1];
                nums[i + 1] = temp;
            }
        }
    }
};
```

java beats 100%

```
class Solution {
    public void wiggleSort(int[] nums) {
        for (int i = 0;i < nums.length - 1;i++) {
            if (((i & 1) == 0 && nums[i] > nums[i + 1])
                || ((i & 1) == 1 && nums[i] < nums[i + 1])) {
                int temp = nums[i];
                nums[i] = nums[i + 1];
                nums[i + 1] = temp;
            }
        }
    }
}
```

go beats 96.55%


```
func wiggleSort(nums []int)  {
    for i := 0;i < len(nums) - 1;i++ {
        if ((i & 1) == 0 && nums[i] > nums[i + 1]) || ((i & 1) == 1 && nums[i] < nums[i + 1]) {
            temp := nums[i]
            nums[i] = nums[i + 1]
            nums[i + 1] = temp
        }
    }
}
```

}

27 搜索旋转排序数组

29 寻找两个有序数组的中位数