Problem 2164: Sort Even and Odd Indices Independently

Problem Information

Difficulty: Easy
Acceptance Rate: 62.95%
Paid Only: No
Tags: Array, Sorting

Problem Description

You are given a **0-indexed** integer array `nums`. Rearrange the values of `nums` according to the following rules:

1. Sort the values at **odd indices** of `nums` in **non-increasing** order. * For example, if `nums = [4,**_1_** ,2,_**3**_]` before this step, it becomes `[4,_**3**_ ,2,**_1_**]` after. The values at odd indices `1` and `3` are sorted in non-increasing order. 2. Sort the values at **even indices** of `nums` in **non-decreasing** order. * For example, if `nums = [_**4**_ ,1,_**2**_ ,3]` before this step, it becomes `[_**2**_ ,1,_**4**_ ,3]` after. The values at even indices `0` and `2` are sorted in non-decreasing order.

Return _the array formed after rearranging the values of_ `nums`.

Example 1:

Input: nums = [4,1,2,3] **Output:** [2,3,4,1] **Explanation:** First, we sort the values present at odd indices (1 and 3) in non-increasing order. So, nums changes from [4,**_1_** ,2,**_3_**] to [4,_**3**_ ,2,**_1_**]. Next, we sort the values present at even indices (0 and 2) in non-decreasing order. So, nums changes from [_**4**_ ,1,**_2_** ,3] to [_**2**_ ,3,_**4**_ ,1]. Thus, the array formed after rearranging the values is [2,3,4,1].

Example 2:

Input: nums = [2,1] **Output:** [2,1] **Explanation:** Since there is exactly one odd index and one even index, no rearrangement of values takes place. The resultant array formed is [2,1], which is the same as the initial array.

Constraints:

* `1 <= nums.length <= 100` * `1 <= nums[i] <= 100`

Code Snippets

C++:

```
class Solution {
public:
vector<int> sortEvenOdd(vector<int>& nums) {

}
};
```

Java:

```
class Solution {
public int[] sortEvenOdd(int[] nums) {

}
}
```

Python3:

```
class Solution:
def sortEvenOdd(self, nums: List[int]) -> List[int]:
```