280. Wiggle Sort (/problems/wiggle-sort/)

March 5, 2016 | 7.2K views

Given an unsorted array `nums`, reorder it **in-place** such that `nums[0] <= nums[1] >= nums[2] <= nums[3]...`.

For example, given `nums = [3, 5, 2, 1, 6, 4]`, one possible answer is `[1, 6, 2, 5, 3, 4]`.

Solution

Approach #1 (Sorting) [Accepted]

The obvious solution is to just sort the array first, then swap elements pair-wise starting from the second element. For example:

```
    [1, 2, 3, 4, 5, 6]
        ↑  ↑  ↑  ↑
        swap  swap

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

```
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;
}
```

Complexity analysis

- Time complexity : $O(n \log n)$. The entire algorithm is dominated by the sorting step, which costs $O(n \log n)$ time to sort n elements.
- Space complexity : $O(1)$. Space depends on the sorting implementation which, usually, costs $O(1)$ auxiliary space if `heapsort` is used.

Approach #2 (One-pass Swap) [Accepted]

Intuitively, we should be able to reorder it in one-pass. As we iterate through the array, we compare the current element to its next element and if the order is incorrect, we swap them.

```
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;
    }
}
```

We could shorten the code further by compacting the condition to a single line. Also observe the boolean value of `less` actually depends on whether the index is even or odd.

```
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);
        }
    }
}
```

Here is another amazing solution by @StefanPochmann who came up with originally here (https://leetcode.com/discuss/57113/java-o-n-solution?show=57192#a57192).

```
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);
        }
    }
}
```

Complexity analysis

- Time complexity : $O(n)$. In the worst case we swap at most $\frac{n}{2}$ times. An example input is `[2,1,3,1,4,1]` .
- Space complexity : $O(1)$.

Rate this article: (/ratings/107/12/?return=/articles/wiggle-sort/) (/ratings/107/12/?return=/articles/wiggle-sort/) (/ra

Previous (/articles/house-robber/) Next (/articles/3sum-smaller/)

Join the conversation

Login to Reply

Y

Yaohua628 commented 8 hours ago

(https://discuss.leetcode.com/user/yaohua628)

Another simple no-repeatness condition is:
(((nums[i] - nums[i - 1]) * ((i % 2) - 0.5)) < 0)
Just for fun..

A (https://discuss.leetcode.com/user/azimbabu)

azimbabu commented 4 months ago

For the second approach, it said in the worst case we swap at most n/2 times. But for the example input, number of swaps seems to be n-1.


(https://discuss.leetcode.com/user/manuelp)

ManuelP commented 7 months ago

@rbacevedo (https://discuss.leetcode.com/uid/311917) No you didn't. Don't lie.

R (https://discuss.leetcode.com/user/rbacevedo)

rbacevedo commented 7 months ago

I literally did it that way and it says Time Limit exceded :/

X (https://discuss.leetcode.com/user/xuan18222)

xuan18222 commented last year

@hieu.trinh (https://discuss.leetcode.com/uid/554) You are saying Wiggle Sort II. See the comments of ashiqimran and of 1337c0d3r.

H (https://discuss.leetcode.com/user/hieu-trinh)

hieu.trinh commented last year

Thanks for the analysis. I have a question, I tried to run all your solutions with this test case [1,2,2,1,2,1,1,1,1,3,2,2] but they produce the in correct result.
Ideally, it should show the result as [1, 3, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2] but it does not. Can you comment on this test case?

A (https://discuss.leetcode.com/user/ashiqimran)

ashiqimran commented 2 years ago

@1337c0d3r (https://discuss.leetcode.com/uid/1) Thanks :)


(https://discuss.leetcode.com/user/1337c0d3r)

1337c0d3r commented 2 years ago

@ashiqimran (https://discuss.leetcode.com/uid/20) You can take a look at the highest voted answers on Wiggle Sort II here (https://leetcode.com/discuss/questions/oj/wiggle-sort-ii?sort=votes).

A (https://discuss.leetcode.com/user/ashiqimran)

ashiqimran commented 2 years ago

Yeah, Actually I am having trouble to solve it in place. Could you help me on Wiggle Sort II?


(https://discuss.leetcode.com/user/1337c0d3r)

1337c0d3r commented 2 years ago

@ashiqimran (https://discuss.leetcode.com/uid/20) Yes that is a totally different problem, which is in Wiggle Sort II - https://leetcode.com/problems/wiggle-sort-ii/ (https://leetcode.com/problems/wiggle-sort-ii/)

View original thread (https://discuss.leetcode.com/topic/39)

Load more comments...