Python/C++/Java Simple Explanation for O(N) Solution

TLDRAlgos ★ 195 Last Edit: 3 days ago 453 VIEWS

11

Upvote if you like quick and simple explanations!

Join our discord to meet other people prepping for interviews!
https://discord.gg/7wpQQFYC

Intuition
We want differences between adjacent elements in the subsequence to be positive or negative. If we have a choice between multiple elements, we should take the biggest such element in the positive case, and the smallest such element in the negative case.

This suggests we always add a peak or a valley to our subsequence. Note that the endpoints also count as a peak/valley.

Python

```python
class Solution:
    def wiggleMaxLength(self, nums: List[int]) -> int:
        peak = valley = 0
        for i in range(1, len(nums)):
            # current element could be a peak
            if nums[i - 1] < nums[i]:
                peak = valley + 1
            # current element could be a valley
            elif nums[i - 1] > nums[i]:
                valley = peak + 1
        # add 1 because we didn't count the first element
        return 1 + max(peak, valley)
```

C++

```cpp
class Solution {
public:
    int wiggleMaxLength(vector<int>& nums) {
        int peak = 0, valley = 0;
        for (int i = 1; i < nums.size(); i++) {
            // current element could be a peak
            if (nums[i - 1] < nums[i]) {
                peak = valley + 1;
            }
            // current element could be a valley
            else if (nums[i - 1] > nums[i]) {
                valley = peak + 1;
            }
        }
        // add 1 because we didn't count the first element
        return 1 + max(peak, valley);
    }
};
```

Java

```java
class Solution {
    public int wiggleMaxLength(int[] nums) {
        int peak = 0, valley = 0;
        for (int i = 1; i < nums.length; i++) {
            // current element could be a peak
            if (nums[i - 1] < nums[i]) {
                peak = valley + 1;
            }
            // current element could be a valley
            else if (nums[i - 1] > nums[i]) {
                valley = peak + 1;
            }
        }
        // add 1 because we didn't count the first element
        return 1 + Math.max(peak, valley);
    }
}
```

Time Complexity: O(n) - One pass through the array.
Space Complexity: O(1) - Amount of additional space used independent of input size

Upvote and **join our discord** https://discord.gg/7wpQQFYC!

python c++ java cpp greedy approach

ana_2kacer ★ 69 3 days ago

Invalid Discord invite :(

1 Show 2 replies Reply