<https://leetcode.com/problems/monotonic-array/>

An array is **monotonic** if it is either monotone increasing or monotone decreasing.

An array nums is monotone increasing if for all i <= j, nums[i] <= nums[j]. An array nums is monotone decreasing if for all i <= j, nums[i] >= nums[j].

Given an integer array nums, return true *if the given array is monotonic, or* false *otherwise*.

**Example 1:**

Input: nums = [1,2,2,3]

Output: true

**Example 2:**

Input: nums = [6,5,4,4]

Output: true

**Example 3:**

Input: nums = [1,3,2]

Output: false

**Constraints:**

1 <= nums.length <= 10^5

-10^5 <= nums[i] <= 10^5

**Attempt 1: 2023-04-03**

**Solution 1: Two Pass (10 min)**

class Solution {

    public boolean isMonotonic(int[] A) {

        return increasing(A) || decreasing(A);

    }

    public boolean increasing(int[] A) {

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

            if (A[i] > A[i+1]) return false;

        return true;

    }

    public boolean decreasing(int[] A) {

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

            if (A[i] < A[i+1]) return false;

        return true;

    }

}

Time Complexity: O(N), where N is the length of A.

Space Complexity: O(1).

**Refer to**

<https://leetcode.com/problems/monotonic-array/editorial/>

**Approach 1: Two Pass**

**Intuition**An array is *monotonic* if it is monotone increasing, or monotone decreasing. Since

a <= b and

b <= c implies

a <= c, we only need to check adjacent elements to determine if the array is monotone increasing (or decreasing, respectively). We can check each of these properties in one pass.

**Algorithm**To check whether an array

A is monotone increasing, we'll check

A[i] <= A[i+1] for all

i. The check for monotone decreasing is similar.

class Solution {

    public boolean isMonotonic(int[] A) {

        return increasing(A) || decreasing(A);

    }

    public boolean increasing(int[] A) {

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

            if (A[i] > A[i+1]) return false;

        return true;

    }

    public boolean decreasing(int[] A) {

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

            if (A[i] < A[i+1]) return false;

        return true;

    }

}

**Complexity Analysis**

Time Complexity: O(N), where N is the length of A.

Space Complexity: O(1).

**Solution 2: One Pass (10 min)**

class Solution {

    public boolean isMonotonic(int[] nums) {

        int len = nums.length;

        boolean increase = true;

        boolean decrease = true;

        for(int i = 0; i < len - 1; i++) {

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

                increase = false;

            }

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

                decrease = false;

            }

        }

        return increase || decrease;

    }

}

Time Complexity: O(N), where N is the length of A.

Space Complexity: O(1).

**Refer to**

<https://leetcode.com/problems/monotonic-array/editorial/>

**Approach 2: One Pass**

**Intuition**

To perform this check in one pass, we want to handle a stream of comparisons from {−1,0,1}, corresponding to <, ==, or >. For example, with the array [1, 2, 2, 3, 0], we will see the stream (-1, 0, -1, 1).

**Algorithm**

Keep track of store, equal to the first non-zero comparison seen (if it exists.) If we see the opposite comparison, the answer is False.

Otherwise, every comparison was (necessarily) in the set {−1,0}, or every comparison was in the set {0,1}, and therefore the array is monotonic.

class Solution {

    public boolean isMonotonic(int[] A) {

        int store = 0;

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

            int c = Integer.compare(A[i], A[i+1]);

            if (c != 0) {

                if (c != store && store != 0)

                    return false;

                store = c;

            }

        }

        return true;

    }

}

**Complexity Analysis**

Time Complexity: O(N), where N is the length of A.

Space Complexity: O(1).

**Approach 3: One Pass (Simple Variant)**

**Intuition and Algorithm**

To perform this check in one pass, we want to remember if it is monotone increasing or monotone decreasing.It's monotone increasing if there aren't some adjacent values A[i], A[i+1] with A[i] > A[i+1], and similarly for monotone decreasing.If it is either monotone increasing or monotone decreasing, then A is monotonic.

class Solution {

    public boolean isMonotonic(int[] A) {

        boolean increasing = true;

        boolean decreasing = true;

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

            if (A[i] > A[i+1])

                increasing = false;

            if (A[i] < A[i+1])

                decreasing = false;

        }

        return increasing || decreasing;

    }

}

**Complexity Analysis**

Time Complexity: O(N), where N is the length of A.

Space Complexity: O(1).

**Refer to**

[L845.Longest Mountain in Array (Ref.L821)](note://WEB927061d55e7df1331b08a8ea60797df9)