<https://leetcode.com/problems/find-all-good-indices/description/>

You are given a 0-indexed integer array nums of size n and a positive integer k.

We call an index i in the range k <= i < n - k good if the following conditions are satisfied:

The k elements that are just before the index i are in non-increasing order.

The k elements that are just after the index i are in non-decreasing order.

Return an array of all good indices sorted in increasing order.

**Example 1:**

**Input:** nums = [2,1,1,1,3,4,1], k = 2

**Output:** [2,3]

**Explanation:** There are two good indices in the array:

- Index 2. The subarray [2,1] is in non-increasing order, and the subarray [1,3] is in non-decreasing order.

- Index 3. The subarray [1,1] is in non-increasing order, and the subarray [3,4] is in non-decreasing order.

Note that the index 4 is not good because [4,1] is not non-decreasing.

**Example 2:**

**Input:** nums = [2,1,1,2], k = 2

**Output:** []

**Explanation:** There are no good indices in this array.

**Constraints:**

n == nums.length

3 <= n <= 10^5

1 <= nums[i] <= 10^6

1 <= k <= n / 2

**Attempt 1: 2024-08-28**

**Solution 1: LIS (30 min)**

**因为需求略有不同导致和 Leetcode 2100 有一点区别：**

**1.We have to initialize the count in dp array for all numbers by 1**

**2.The non-increasing / non-decreasing subsequence for index 'i' calculate start from 'i - 1' and 'i + 1', but not 'i'**

class Solution {

    public List<Integer> goodIndices(int[] nums, int k) {

        int n = nums.length;

        List<Integer> result = new ArrayList<>();

        int[] nonIncreasing = new int[n];

        int[] nonDecreasing = new int[n];

        // Must initialize all dp value as 1 since each

        // number itself is considered as a non-increasing

        // or non-decreasing order

        // Test out by:

        // Input: nums = [2,1,1,1,3,4,1], k = 2

        // Output: [], Expect Output: [2,3]

        Arrays.fill(nonIncreasing, 1);

        Arrays.fill(nonDecreasing, 1);

        // For non-increasing

        for(int i = 1; i < n; i++) {

            if(nums[i] <= nums[i - 1]) {

                nonIncreasing[i] = nonIncreasing[i - 1] + 1;

            }

            // No need reset nonIncreasing[i] to 0 since

            // default as 0 when nums[i] < nums[i - 1]

        }

        // For non-decreasing

        for(int i = n - 2; i >= 0; i--) {

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

                nonDecreasing[i] = nonDecreasing[i + 1] + 1;

            }

            // No need reset nonDecreasing[i] to 0 since

            // default as 0 when nums[i] > nums[i + 1]

        }

        // Find index is good when:

        // k elements "before" index i is non-increasing,

        // "before" means calculate subsequence length start from i - 1,

        // the result stored in dp array as dp[i - 1],

        // i.e if the dp[i - 1] size is >= k

        // k elements after index i is non-decreasing,

        // "after" means calculate subsequence length start from i + 1,

        // the result stored in dp array as dp[i + 1],

        // i.e if the dp[i + 1] size is >= k

        for(int i = k; i < n - k; i++) {

            if(nonIncreasing[i - 1] >= k && nonDecreasing[i + 1] >= k) {

                result.add(i);

            }

        }

        return result;

    }

}

**Refer to**

<https://leetcode.com/problems/find-all-good-indices/solutions/2620565/dp-c-java-python-intuition/comments/1617702>

vector<int> goodIndices(vector<int>& a, int k) {

int n = size(a);

vector<int> dp1(n,1);

vector<int> dp2(n,1);

vector<int> ans;

//storing the length of contiguous non-increasing subarray in dp1

for (int i = 1; i < n; i++)

{

if (a[i-1] >= a[i])

dp1[i] = dp1[i-1] + 1;

}

//storing the length of contiguous non-decreasing subarray in dp2

for (int i = n-2; i >= 0; i--)

{

if (a[i] <= a[i+1])

dp2[i] = dp2[i+1] + 1;

}

//index is good when :

//k elements before index i is non-increasing, i.e if the dp[i-1] size is >= k

//k elements after index i is non-decreasing, i.e if the dp[i+1] >= k

//checking conditions of good index in the range of all possible good index

for (int i = k; i < n-k; i++)

{

if (dp1[i-1] >= k && dp2[i+1] >= k)

ans.push\_back(i);

}

return ans;

}

**Refer to**

[L2100.Find Good Days to Rob the Bank (Ref.L1671,L300)](note://WEBe446de09772f09c6c7e67ae34948f3f8)

[L300.Longest Increasing Subsequence](note://20C954956F6E4AA6A3575EA3146A1FCA)

[L1671.Minimum Number of Removals to Make Mountain Array (Ref.L300)](note://A717168457D646DF8CE8A9054DD61C12)