**Problem #3105: Longest Strictly Increasing or Strictly Decreasing Subarray. (Easy)**

<https://leetcode.com/problems/longest-strictly-increasing-or-strictly-decreasing-subarray/description/>

**My Solution:**

1. Set increasing = False..
2. Let maxLen be 1
3. Initialize length to 0.
4. First consider increasing subarrays.

Iterate through nums.

If nums at index I is less than that at index I -1 (i.e the current value is less than previous value), if increasing is False, then length is 2, the set increasing to True. , otherwise if increasing is True, then increment length by 1.

Update maxLen to be the maximum of maxLen and length.

If nums at index I is not less than that at index I -1, then set increasing to False.

1. Now consider decreasing subarrays.

Iterate through nums.

If nums at index I is greater than that at index I -1 (i.e the current value is greater than previous value), if decreasing is False, then length is 2, the set decreasing to True. , otherwise if decreasing is True, then increment length by 1.

Update maxLen to be the maximum of maxLen and length.

If nums at index I is not greater than that at index I -1, then set deccreasing to False.

1. Finally return maxLen.

class Solution:

def longestMonotonicSubarray(self, nums: List[int]) -> int:

increasing = False

maxLen = 1

length = 0

# Consider increasing subarrays

for i in range(1, len(nums)):

if nums[i - 1] < nums[i]:

if not increasing:

length = 2

increasing = True

else: # increasing is True

length += 1

maxLen = max(maxLen, length)

else:

increasing = False

# Consider decreasing subarrays

decreasing = False

for i in range(1, len(nums)):

if nums[i - 1] > nums[i]:

if not decreasing:

length = 2

decreasing = True

else: # decreasing is True

length += 1

maxLen = max(maxLen, length)

else:

decreasing = False

return maxLen