Problem # 845 : Longest Mountain in Array (Medium)

<https://leetcode.com/problems/longest-mountain-in-array/>

Solution:

1. If the length of A is less than 3, then there is no peak since we need a minimum of 3 elements in the array. So return 0.

2. Initialize res to 0, where res is used to track the length of the longest mountain so far.

3. Iterate through the array to find peaks by considering 3 consecutive elements at a time.

4. Once we find a peak, then initialize left and right pointers to the index of the peak element.

5. Keep moving the left pointer as long as the element at the left pointer is greater than the previous element.

6. Keep moving the right pointer as long as the element at the right pointer is less than the next element.

7. The length of the mountain is the number of elements between the left and right pointer where both are included = (right – left + 1).

8. Update res to be the maximum of current length of the mountain and previous best so far. (i.e. res).

9. Return res.

class Solution:

def longestMountain(self, A: List[int]) -> int:

if len(A) < 3:

return(0)

res = 0

for indx in range(1, len(A)-1):

if A[indx - 1] < A[indx] and A[indx] > A[indx + 1]: # we have a peak

left, right = indx, indx

left = indx

while left > 0 and A[left-1]< A[left]: # go left

left -= 1

right = indx

while right < len(A) - 1 and A[right] > A[right + 1]: # go right

right += 1

res = max(res, right - left + 1)

return(res)