Approach 1: Brute Force

1. Check for every subarray by reversing it.
2. If the array becomes sorted for any subarray being reversed, return true.
3. If no such subarray

### **Pseudocode**

For i = 0 to n:

For j = 0 to i:

reverse the subarray arr[i:j]

if the array becomes sorted:

return true

If no such subarray exist:

return false

**Code:**

for i in range(n):

for j in range(i+1, n+1):

# generate reversed array (reverse i to j)

rev\_arr = arr[0:i] + arr[i:j][::-1] + arr[j:]

flag = True

# check if reversed array sorted

for k in range(1,n):

if rev\_arr[k] < rev\_arr[k-1]:

flag = False

break

if flag:

return True

return False

Approach 2: Optimized

1. It is known that
2. Iterate the array from start until mismatched found, call this front

[1,2,3,5,4] here front = 3

1. Iterate backwards until mismatch found, call this back

[1,2,3,5,4] here back = 4

1. Iterate from back to front and check if array is sorted

**Code**:

n = len(arr)

front = None

sorted\_arr = sorted(arr)

for i in range(1,n):

if arr[i] != sorted\_arr[i]:

front = i

break

# if front not updated, array already sorted

if not front:

return True

for i in range(n,-1, -1):

if arr[i] != sorted\_arr[i]:

back = i

break

# check if array between front and back sorted

for i in range(back, front, -1):

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

return False

return True