lab8nb

August 2, 2024

Given a positive integer array nums and an integer k. We need to find the length of the longest subarray that has a sum less than or equal to k.

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

k = 5

Task 1:

Write the Naive solution: Consider all possible subarrays using a nested loop. Return the one with sum <= k and largest length.

[ ]:
```
def length_subarray(nums, k):
    result = 0
    for start in range(len(nums)):
        sum = 0
        for end in range(start, len(nums)):
            sum += nums[end]
            if sum <= k:
                result = max(result, end - start + 1)
    return result
```

[ ]:
```
nums = [3, 2, 1, 3, 1, 1]
k = 5
result = length_subarray(nums, k)
print(result)
```

```
3
```

Task 2: (Optional)

Without looking at my code, try to write the sliding window solution

[ ]:
```
def length_subarray_win(nums, k):
    result = 0
    sum = 0
    left_index = 0

    for right_index in range(len(nums)):
        sum += nums[right_index]
```

```
        while sum > k:
            sum -= nums[left_index]
            left_index += 1
        result = max(result, right_index - left_index + 1)

    return result
```

[ ]:
```
nums = [3, 2, 1, 3, 1, 1]
k = 5
result_win = length_subarray_win(nums, k)
print(result_win)
```

```
3
```

Given two strings s and t, return true if s is a subsequence of t, or false otherwise. A subsequence of a string is a sequence of characters that can be obtained by deleting some of the characters from the original string, while maintaining the relative order of the remaining characters. For example, “ace” is a subsequence of “abcde” while “aec” is not.

[ ]:
```
def subsequence(t, s):
    result_bool = False
    pt2 = 0
    for pt1 in range(len(t)):
        if t[pt1] == s[pt2]:
            pt2 += 1
            if pt2 == (len(s)):
                return True
    return False
```

[ ]:
```
t = "abcde"
s1 = "ace"
s2 = "aec"

print(subsequence(t, s1))
print(subsequence(t, s2))
```

```
True
False
```

[ ]: