1031. Maximum Sum of Two Non-Overlapping Subarrays

Given an integer array `nums` and two integers `firstLen` and `secondLen`, return *the maximum sum of elements in two non-overlapping **subarrays** with lengths* `firstLen` *and* `secondLen`.

The array with length `firstLen` could occur before or after the array with length `secondLen`, but they have to be non-overlapping.

A **subarray** is a **contiguous** part of an array.

Example 1:

```
Input: nums = [0,6,5,2,2,5,1,9,4], firstLen = 1, secondLen = 2
Output: 20
Explanation: One choice of subarrays is [9] with length 1, and [6,5] with
length 2.
```

Example 2:

```
Input: nums = [3,8,1,3,2,1,8,9,0], firstLen = 3, secondLen = 2
Output: 29
Explanation: One choice of subarrays is [3,8,1] with length 3, and [8,9]
with length 2.
```

Example 3:

```
Input: nums = [2,1,5,6,0,9,5,0,3,8], firstLen = 4, secondLen = 3
Output: 31
Explanation: One choice of subarrays is [5,6,0,9] with length 4, and [3,8]
with length 3.
```

Constraints:

- `1 <= firstLen, secondLen <= 1000`
- `2 <= firstLen + secondLen <= 1000`
- `firstLen + secondLen <= nums.length <= 1000`
- `0 <= nums[i] <= 1000`

```
class Solution:
    def maxSumTwoNoOverlap(self, nums: List[int], firstLen: int, secondLen:
```

```python
int) -> int:
        dp1 = [0]*len(nums)
        dp2 = [0]*len(nums)
        prefix = 0

        for i in range(len(nums)):
            if i<firstLen:
                prefix+=nums[i]
                dp1[i] = prefix
            else:
                prefix+=nums[i]-nums[i-firstLen]
                dp1[i] = max(dp1[i-1],prefix)



        prefix = 0
        for j in range(len(nums)-1,-1,-1):
            if j+secondLen>=len(nums):
                prefix+=nums[j]
                dp2[j] = prefix
            else:
                prefix+=nums[j]-nums[j+secondLen]
                dp2[j] = max(dp2[j+1],prefix)



        ans = 0
        for i in range(firstLen-1,len(nums)-secondLen):
            ans = max(ans,dp1[i]+dp2[i+1])







        prefix = 0
        for i in range(len(nums)):
            if i<secondLen:
                prefix+=nums[i]
                dp1[i] = prefix
            else:
                prefix+=nums[i]-nums[i-secondLen]
                dp1[i] = max(dp1[i-1],prefix)



        prefix = 0
        for j in range(len(nums)-1,-1,-1):
            if j+firstLen>=len(nums):
                prefix+=nums[j]
                dp2[j] = prefix
            else:
                prefix+=nums[j]-nums[j+firstLen]
                dp2[j] = max(dp2[j+1],prefix)
        for i in range(secondLen-1,len(nums)-firstLen):
            ans = max(ans,dp1[i]+dp2[i+1])
        return ans
```