Problem 1031: Maximum Sum of Two Non-Overlapping Subarrays

Problem Information

Difficulty: Medium
Acceptance Rate: 60.62%
Paid Only: No
Tags: Array, Dynamic Programming, Sliding Window

Problem Description

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 [0,3,8] with length 3.

Constraints:

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

Code Snippets

C++:

```
class Solution {
public:
int maxSumTwoNoOverlap(vector<int>& nums, int firstLen, int secondLen) {

}
};
```

Java:

```
class Solution {
public int maxSumTwoNoOverlap(int[] nums, int firstLen, int secondLen) {

}
}
```

Python3:

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