**[Maximum Sum of Two Non-Overlapping Subarrays](https://leetcode.com/problems/maximum-sum-of-two-non-overlapping-subarrays/)**

Given an array A of non-negative integers, return the maximum sum of elements in two non-overlapping (contiguous) subarrays, which have lengths L and M.  (For clarification, the L-length subarray could occur before or after the M-length subarray.)

Formally, return the largest V for which V = (A[i] + A[i+1] + ... + A[i+L-1]) + (A[j] + A[j+1] + ... + A[j+M-1]) and either:

* 0 <= i < i + L - 1 < j < j + M - 1 < A.length, **or**
* 0 <= j < j + M - 1 < i < i + L - 1 < A.length.

public int maxSumTwoNoOverlap(int[] A, int L, int M) {

for (int i = 1; i < A.length; ++i)

A[i] += A[i - 1];

int res = A[L + M - 1], Lmax = A[L - 1], Mmax = A[M - 1];

for (int i = L + M; i < A.length; ++i) {

Lmax = Math.max(Lmax, A[i - M] - A[i - L - M]);

Mmax = Math.max(Mmax, A[i - L] - A[i - L - M]);

res = Math.max(res, Math.max(Lmax + A[i] - A[i - M], Mmax + A[i] - A[i - L]));

}

return res;

}

{3, 8, 1, 4, 2, 7, 22, 9} L=3, M=2

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 3 | 8 | 1 | 4 | 2 | 7 | 22 | 9 |

Cumulative Sum (CM)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 3 | 8 | 1 | 4 | 2 | 7 | 22 | 9 |
| 3 | 11 | 12 | 16 | 18 | 25 | 47 | 56 |

LMax : L contagious elements taken first

MMax: M contagious element taken first

Max: Maximum sum for two non-overlapping sub-array

i=0 {Here if you see Lmax and Mmax they will look overlapping but Max is already been make sure that they are not overlapping}

LMAx = CM[L-1] = CM[2] = 12 [ 3,8,1]

MMax = CM[M-1] = CM[1] = 11 [3,8]

Max = CM[L+M-1] = CM[4] = 18 { [3,8,1] & [4,2] }

I=5

Next L Elements are [8,1,4]=13 and earlier [3,8,1] = 12 => 13

Next M elements are [8,1]= 9 and earlier [3,8] = 11 => 11

Max = {L elements + M elements , M elements + L elements } = {13 + {25-16}, 11+{25-12} }

={21, 24 } = 24

Since [8,1,4]=13 Gave maximum as compare to old [3,8,1]

And [3,8]=11 Gave maximum as compare to new [8,1].

Now M elements after [8,1,4] are [2,7]=9

And Next L elements after [8,1] are [4,2,7]=13

I=6

Next L elements are [1,4,2] = 8 and earlier [8,1,4] = 13 => 13

Next M elements are [1,4] = 5 and earlier [3,8] = 11 => 11

Now M elements after [1,4,2] are [7,22]= 29

And L elements after [1,4] are [2,7,22]=31

Max = {L elements + M elements , M elements + L elements } = {13 + {47-18}, 11 + {47-16} } = = {13 + 29, 11 + 31 } = 42

Since [1,4,2]=8 did not give maximum sum as compare to old [8,1,4]=13

And [1,4]=5 did not give maximum sum as compare to old [3,8]=11

I =7

L-> [4,2,7]=13 ; old [8,1,4]=13 => 13

M -> [4,2]=7 , old [3,8]=11 => 11

Max = { 13 + 31, 11 + 38 } = 49

L-> Next M Elements [22,9]

M-> Next L elements [7,22,9]

….

Ans= 49