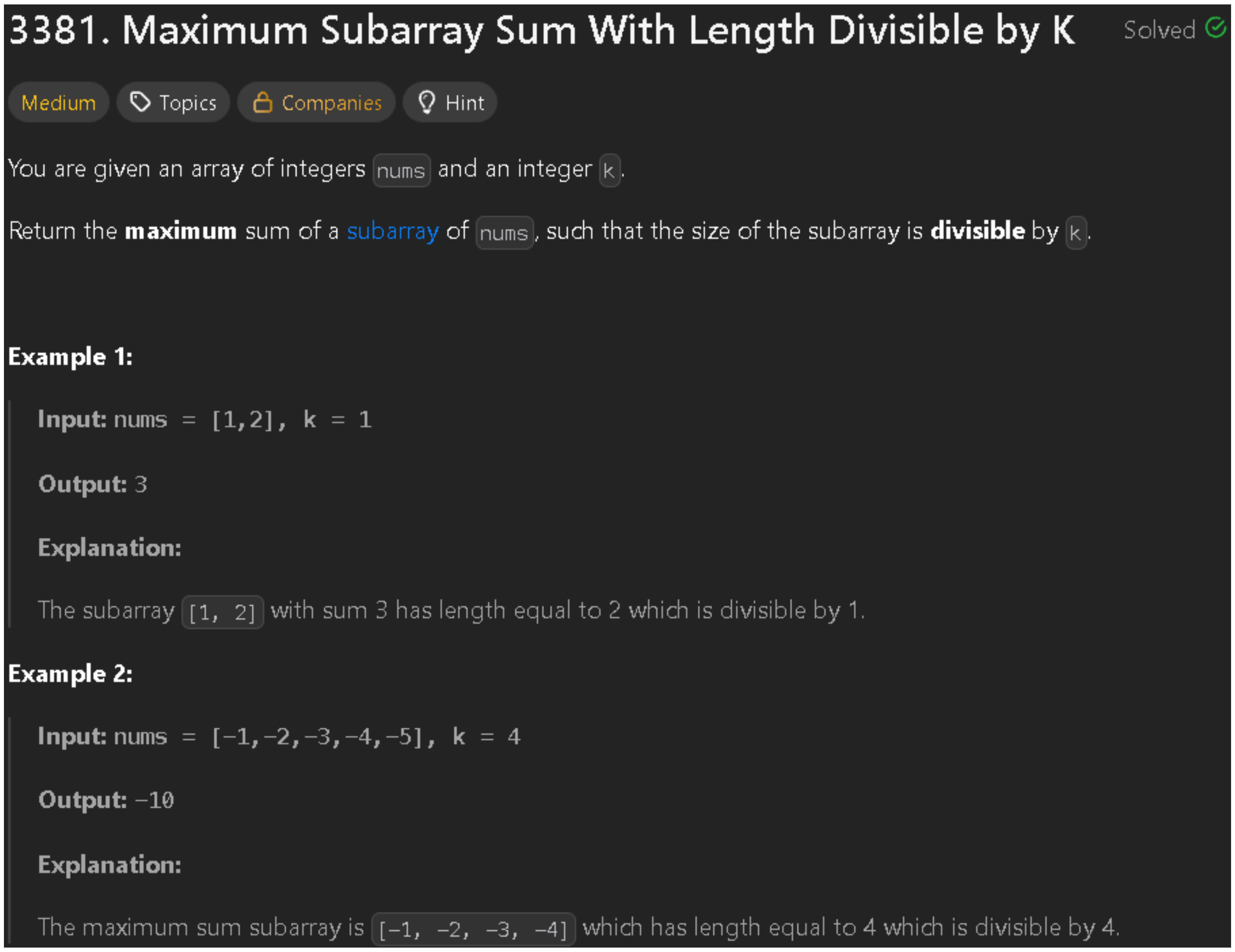


# 3381. Maximum Subarray Sum With Length Divisible by K

Solved ✓

Medium | Topics | Companies | Hint

You are given an array of integers `nums` and an integer `k`.

Return the **maximum** sum of a subarray of `nums`, such that the size of the subarray is **divisible** by `k`.

**Example 1:**

**Input:** nums = [1,2], k = 1

**Output:** 3

**Explanation:**

The subarray `[1, 2]` with sum 3 has length equal to 2 which is divisible by 1.

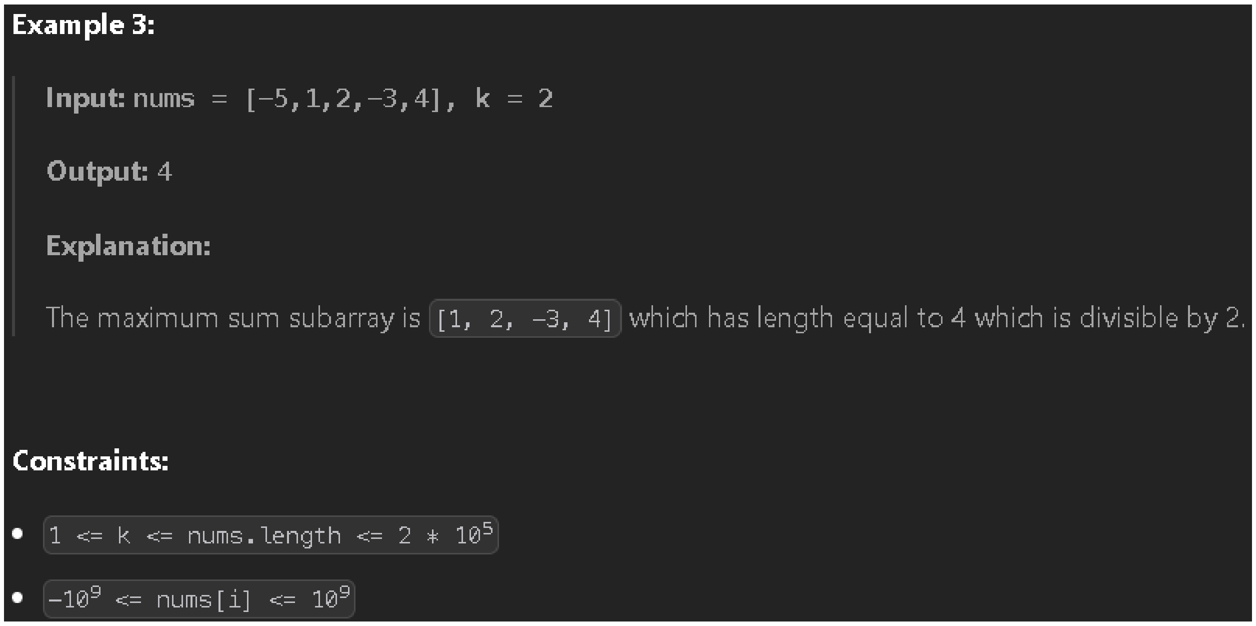
**Example 2:**

**Input:** nums = [-1,-2,-3,-4,-5], k = 4

**Output:** -10

**Explanation:**

The maximum sum subarray is `[-1, -2, -3, -4]` which has length equal to 4 which is divisible by 4.



**Example 3:**

**Input:** nums = [-5,1,2,-3,4], k = 2

**Output:** 4

**Explanation:**

The maximum sum subarray is `[1, 2, -3, 4]` which has length equal to 4 which is divisible by 2.

**Constraints:**

- $1 <= k <= nums.length <= 2 * 10^5$
- $-10^9 <= nums[i] <= 10^9$

## Python:

class Solution:

```
    def maxSubarraySum(self, A: List[int], k: int) -> int:
        prefix = [inf] * k
        prefix[-1] = 0
        res = -inf
        for i, pre in enumerate(accumulate(A)):
            res = max(res, pre - prefix[i % k])
            prefix[i % k] = min(prefix[i % k], pre)
        return res
```

## JavaScript:

```
/**
 * @param {number[]} nums
 * @param {number} k
 * @return {number}
 */
var maxSubarraySum = function(nums, k) {
    const INF = 1e30;
    const minPrefix = new Array(k).fill(INF);
    minPrefix[0] = 0;

    let prefix = 0;
    let answer = -1e30;

    for (let i = 0; i < nums.length; i++) {
        prefix += nums[i];
        const mod = (i + 1) % k;

        if (minPrefix[mod] !== INF) {
            answer = Math.max(answer, prefix - minPrefix[mod]);
        }

        if (prefix < minPrefix[mod]) {
            minPrefix[mod] = prefix;
        }
    }

    return answer;
};
```

## Java:

```
class Solution {
        public long maxSubarraySum(int[] A, int k) {
        long[] prefix = new long[k];
        Arrays.fill(prefix, (long)1e15);
        prefix[k - 1] = 0;
        long res = (long)-1e15, pre = 0;
        for (int i = 0; i < A.length; i++) {
            pre += A[i];
            res = Math.max(res, pre - prefix[i % k]);
            prefix[i % k] = Math.min(prefix[i % k], pre);
        }
        return res;
    }
}
```