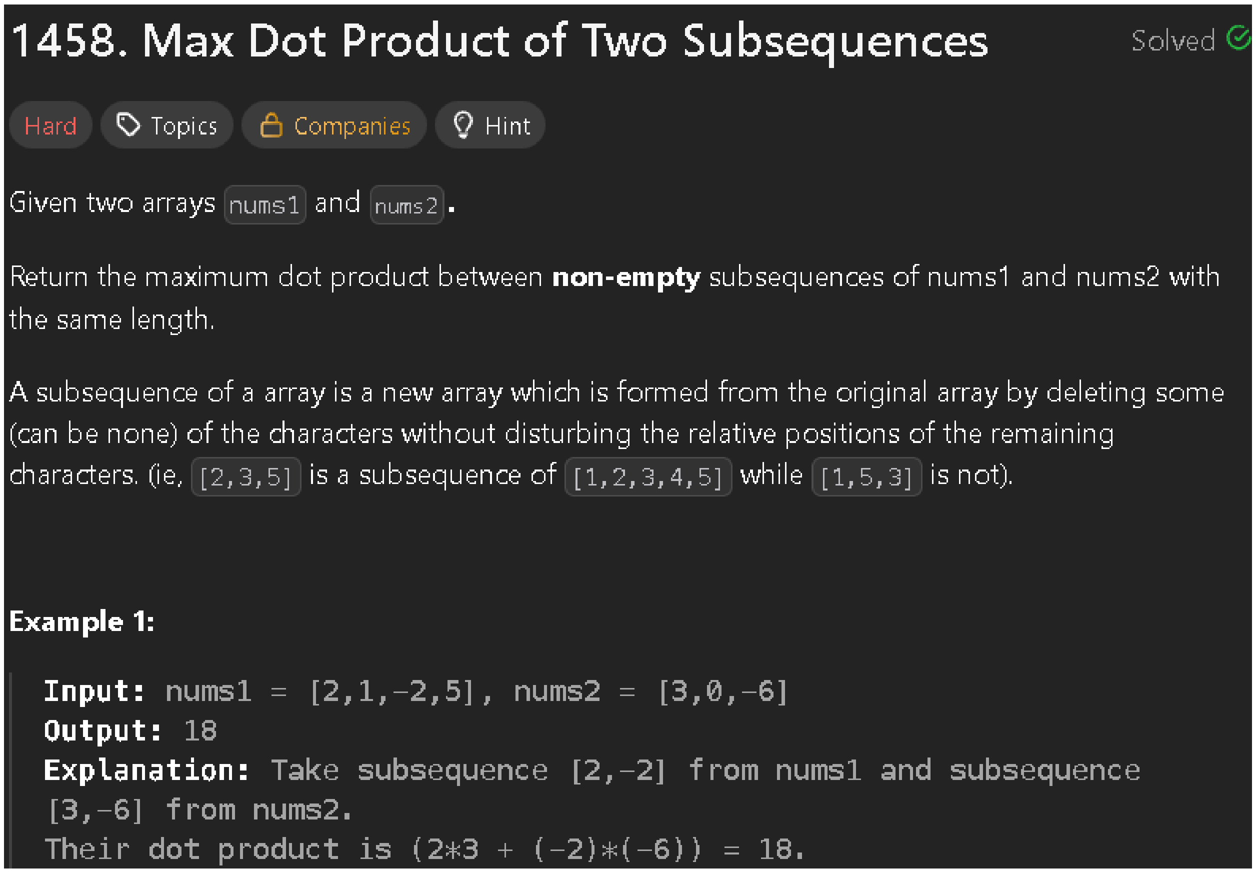


1458. Max Dot Product of Two Subsequences

Solved

Hard Topics Companies Hint

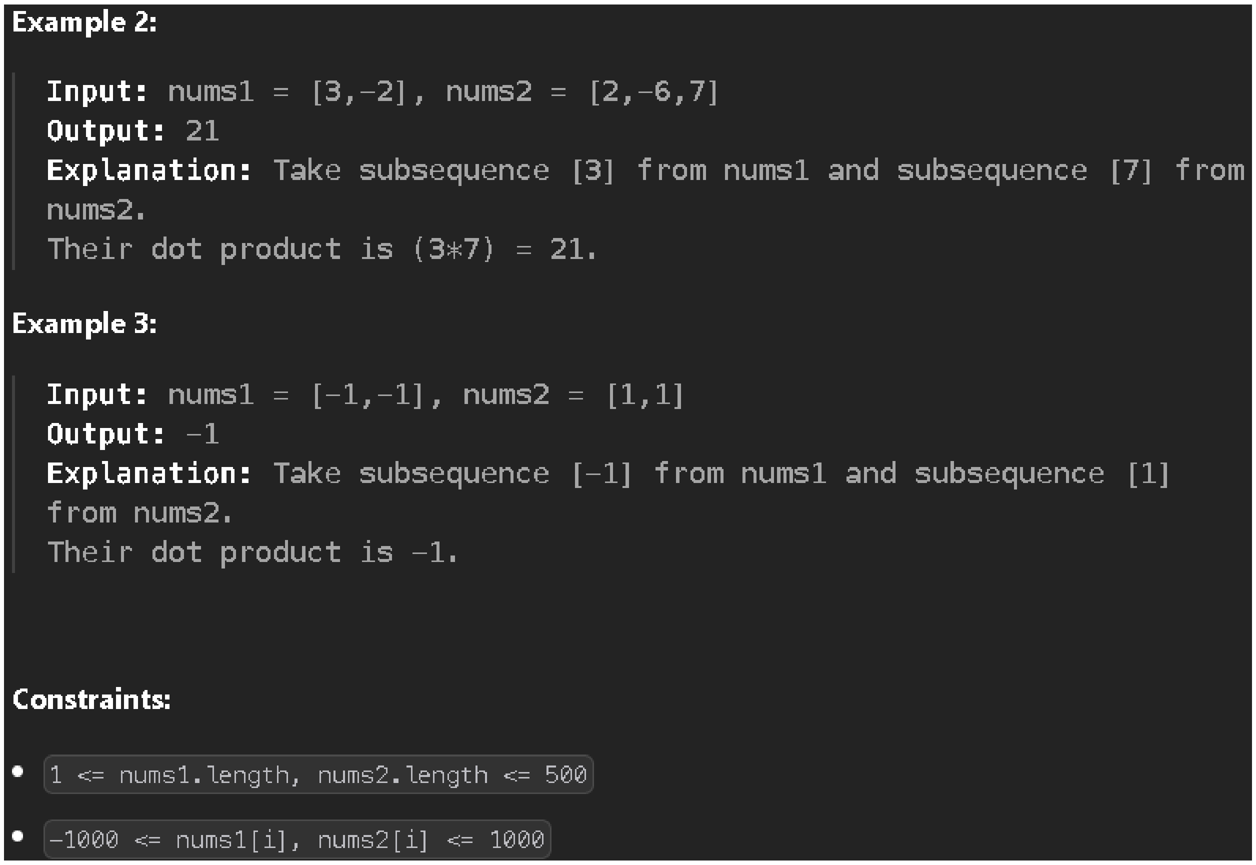
Given two arrays `nums1` and `nums2`.

Return the maximum dot product between **non-empty** subsequences of nums1 and nums2 with the same length.

A subsequence of a array is a new array which is formed from the original array by deleting some (can be none) of the characters without disturbing the relative positions of the remaining characters. (ie, `[2,3,5]` is a subsequence of `[1,2,3,4,5]` while `[1,5,3]` is not).

Example 1:

```
Input: nums1 = [2,1,-2,5], nums2 = [3,0,-6]
Output: 18
Explanation: Take subsequence [2,-2] from nums1 and subsequence
[3,-6] from nums2.
Their dot product is (2*3 + (-2)*(-6)) = 18.
```


Example 2:

```
Input: nums1 = [3,-2], nums2 = [2,-6,7]
Output: 21
Explanation: Take subsequence [3] from nums1 and subsequence [7] from
nums2.
Their dot product is (3*7) = 21.
```

Example 3:

```
Input: nums1 = [-1,-1], nums2 = [1,1]
Output: -1
Explanation: Take subsequence [-1] from nums1 and subsequence [1]
from nums2.
Their dot product is -1.
```

Constraints:

- `1 <= nums1.length, nums2.length <= 500`
- `-1000 <= nums1[i], nums2[i] <= 1000`

Python:

```
class Solution:
    def maxDotProduct(self, a, b):
        n, m = len(a), len(b)
        NEG = -10**9
        dp = [[NEG]*(m+1) for _ in range(n+1)]

        for i in range(1, n+1):
            for j in range(1, m+1):
                take = a[i-1]*b[j-1] + max(0, dp[i-1][j-1])
                dp[i][j] = max(take, dp[i-1][j], dp[i][j-1])

        return dp[n][m]
```

JavaScript:

```
var maxDotProduct = function(a, b) {
    const n = a.length, m = b.length;
    const NEG = -1e9;
    const dp = Array.from({length: n+1}, () => Array(m+1).fill(NEG));

    for(let i=1;i<=n;i++){
        for(let j=1;j<=m;j++){
            let take = a[i-1]*b[j-1] + Math.max(0, dp[i-1][j-1]);
            dp[i][j] = Math.max(take, dp[i-1][j], dp[i][j-1]);
        }
    }
    return dp[n][m];
};
```

Java:

```
class Solution {
    public int maxDotProduct(int[] a, int[] b) {
        int n = a.length, m = b.length;
        int NEG = (int)-1e9;
        int[][] dp = new int[n+1][m+1];

        for(int i=0;i<=n;i++)
            for(int j=0;j<=m;j++)
                dp[i][j] = NEG;

        for(int i=1;i<=n;i++){
            for(int j=1;j<=m;j++){
                int take = a[i-1]*b[j-1] + Math.max(0, dp[i-1][j-1]);
                dp[i][j] = Math.max(take, Math.max(dp[i-1][j], dp[i][j-1]));
            }
        }
        return dp[n][m];
    }
}
```