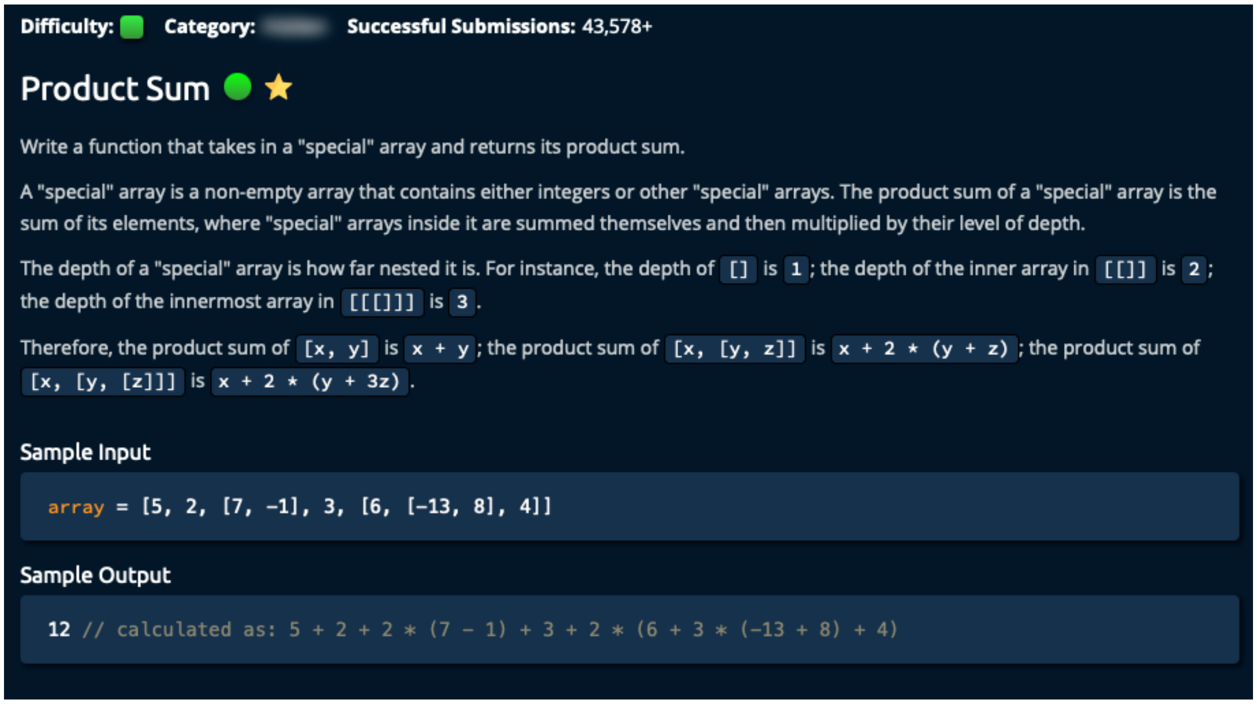
# Code Challenge #9 Product Sum (Easy)



**Difficulty:** 🟩 **Category:** [redacted] **Successful Submissions:** 43,578+

Product Sum ● ★

Write a function that takes in a "special" array and returns its product sum.

A "special" array is a non-empty array that contains either integers or other "special" arrays. The product sum of a "special" array is the sum of its elements, where "special" arrays inside it are summed themselves and then multiplied by their level of depth.

The depth of a "special" array is how far nested it is. For instance, the depth of `[]` is `1`; the depth of the inner array in `[[]]` is `2`; the depth of the innermost array in `[[[]]]` is `3`.

Therefore, the product sum of `[x, y]` is `x + y`; the product sum of `[x, [y, z]]` is `x + 2 * (y + z)`; the product sum of `[x, [y, [z]]]` is `x + 2 * (y + 3z)`.

**Sample Input**

```
array = [5, 2, [7, -1], 3, [6, [-13, 8], 4]]
```

**Sample Output**

```
12 // calculated as: 5 + 2 + 2 * (7 - 1) + 3 + 2 * (6 + 3 * (-13 + 8) + 4)
```

## Solution #1

```
function productSum(array, multiplier = 1) {

  let sum = 0;
for (let element of array){
  if (Array.isArray(element)) {
          sum += productSum(element, multiplier + 1);
  } else {
          sum += element;
  }
  }
          return sum * multiplier;
  }

```

### Explanation

This solution works using a recursive solution where we call the original function productSum on all subarrays in the original array. We solve this problem by creating a function called ProductSum which takes in two arguments called array and multipler with an initial value of 1.  We create a variable called sum which has an initial value of 0. We then create a for loop which loops through each element of the array. We check to see if the element is an array using an object method called Array.isArray(element). If it is we add to the sum the values of that nested array using the recursive call of ProductSum with arguments of element and multipler + 1. If it isn't a nested array we simply add to the sum the element of the array. We finally return the sum times the multipler. *The multiplier would have incremented each time we found a nested array in the original array. This function runs in O(n) time.