

PromptScratchpadSolutionsVideo Explanation

Difficulty: Category: RecursionSuccessful Submissions: 46,822+

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)`

Hints

Hint 1

Try using recursion to solve this problem.

Hint 2

Initialize the product sum of the "special" array to 0. Then, iterate through all of the array's elements; if you come across a number, add it to the product sum; if you come across another "special" array, recursively call the productSum function on it and add the returned value to the product sum. How will you handle multiplying the product sums at a given level of depth?

Hint 3

Have the productSum function take in a second parameter: the multiplier, initialized to 1. Whenever you recursively call the productSum function, pass in the multiplier incremented by 1.

Optimal Space & Time Complexity

O(n) time | O(d) space - where n is the total number of elements in the array, including sub-elements, and d is the greatest depth of "special" arrays in the array

Your Solutions

Solution 1Solution 2Solution 3

✓

```
1 def productSum(array, depth = 1):
2     # Write your code here.
3     Sum = 0
4     for element in array:
5         if type(element) is list:
6             Sum+= productSum(element, depth + 1)
7         else:
8             Sum+= element
9     return Sum * depth
10
11 # Kunal Wadhwa
12
13 # https://github.com/kunal5042
14 # https://leetcode.com/kunal5042/
15 # https://www.hackerrank.com/kunalwadhwa_cs
16 # https://www.linkedin.com/in/kunal5042/
17
```

TestsQuick TestSandbox

Test Case 1

```
1 {
2   "array": [5, 2, [7, -1], 3, [6, [-13, 8], 4]]
3 }
```

Test Case 2

```
1 {
2   "array": [1, 2, 3, 4, 5]
3 }
```

Test Case 3

```
1 {
2   "array": [1, 2, [3], 4, 5]
3 }
```

Test Case 4

```
1 {
2   "array": [
3     [1, 2],
4     3,
5     [4, 5]
6   ]
7 }
```

Test Case 5

```
1 {
2   "array": [
3     [
```

Custom OutputRaw OutputSubmit Code

Yay, your code passed all the test cases!

6 / 6 test cases passed.

✓ Test Case 1 passed!

✓ Test Case 2 passed!

✓ Test Case 3 passed!

✓ Test Case 4 passed!

✓ Test Case 5 passed!

✓ Test Case 6 passed!