Problem 339: Nested List Weight Sum

Problem Information

Difficulty: Medium
Acceptance Rate: 85.82%
Paid Only: Yes
Tags: Depth-First Search, Breadth-First Search

Problem Description

You are given a nested list of integers `nestedList`. Each element is either an integer or a list whose elements may also be integers or other lists.

The **depth** of an integer is the number of lists that it is inside of. For example, the nested list `[1,[2,2],[[3],2],1]` has each integer's value set to its **depth**.

Return _the sum of each integer in_`nestedList` _multiplied by its**depth**_.

Example 1:

Input: nestedList = [[1,1],2,[1,1]] **Output:** 10 **Explanation:** Four 1's at depth 2, one 2 at depth 1. 1*2 + 1*2 + 2*1 + 1*2 + 1*2 = 10.

Example 2:

Input: nestedList = [1,[4,[6]]] **Output:** 27 **Explanation:** One 1 at depth 1, one 4 at depth 2, and one 6 at depth 3. 1*1 + 4*2 + 6*3 = 27.

Example 3:

Input: nestedList = [0] **Output:** 0

Constraints:

* `1 <= nestedList.length <= 50` * The values of the integers in the nested list is in the range `[-100, 100]`. * The maximum **depth** of any integer is less than or equal to `50`.

Code Snippets

C++:

```
/**
* // This is the interface that allows for creating nested lists.
* // You should not implement it, or speculate about its implementation
* class NestedInteger {
* public:
* // Constructor initializes an empty nested list.
* NestedInteger();
*
* // Constructor initializes a single integer.
* NestedInteger(int value);
*
* // Return true if this NestedInteger holds a single integer, rather than a
nested list.
* bool isInteger() const;
*
* // Return the single integer that this NestedInteger holds, if it holds a
single integer
* // The result is undefined if this NestedInteger holds a nested list
* int getInteger() const;
*
* // Set this NestedInteger to hold a single integer.
* void setInteger(int value);
*
* // Set this NestedInteger to hold a nested list and adds a nested integer
to it.
* void add(const NestedInteger &ni);
*
* // Return the nested list that this NestedInteger holds, if it holds a
nested list
* // The result is undefined if this NestedInteger holds a single integer
* const vector<NestedInteger> &getList() const;
```

```
* };
*/
class Solution {
public:
int depthSum(vector<NestedInteger>& nestedList) {

}
};
```

Java:

```
/**
* // This is the interface that allows for creating nested lists.
* // You should not implement it, or speculate about its implementation
* public interface NestedInteger {
* // Constructor initializes an empty nested list.
* public NestedInteger();
*
* // Constructor initializes a single integer.
* public NestedInteger(int value);
*
* // @return true if this NestedInteger holds a single integer, rather than a
nested list.
* public boolean isInteger();
*
* // @return the single integer that this NestedInteger holds, if it holds a
single integer
* // The result is undefined if this NestedInteger holds a nested list
* public Integer getInteger();
*
* // Set this NestedInteger to hold a single integer.
* public void setInteger(int value);
*
* // Set this NestedInteger to hold a nested list and adds a nested integer
to it.
* public void add(NestedInteger ni);
*
* // @return the nested list that this NestedInteger holds, if it holds a
nested list
* // The result is undefined if this NestedInteger holds a single integer
* public List<NestedInteger> getList();
* }
```

```
*/
class Solution {
public int depthSum(List<NestedInteger> nestedList) {

}
}
```

Python3:

```
# """
# This is the interface that allows for creating nested lists.
# You should not implement it, or speculate about its implementation
# """
#class NestedInteger:
# def __init__(self, value=None):
# """
# If value is not specified, initializes an empty list.
# Otherwise initializes a single integer equal to value.
# """
#
# def isInteger(self):
# """
# @return True if this NestedInteger holds a single integer, rather than a
nested list.
# :rtype bool
# """
#
# def add(self, elem):
# """
# Set this NestedInteger to hold a nested list and adds a nested integer elem
to it.
# :rtype void
# """
#
# def setInteger(self, value):
# """
# Set this NestedInteger to hold a single integer equal to value.
# :rtype void
# """
#
# def getInteger(self):
# """
```

```
# @return the single integer that this NestedInteger holds, if it holds a
single integer
# The result is undefined if this NestedInteger holds a nested list
# :rtype int
# """
#
# def getList(self):
# """
# @return the nested list that this NestedInteger holds, if it holds a nested
list
# The result is undefined if this NestedInteger holds a single integer
# :rtype List[NestedInteger]
# """

class Solution:
def depthSum(self, nestedList: List[NestedInteger]) -> int:
```