Problem 364: Nested List Weight Sum II

Problem Information

Difficulty: Medium
Acceptance Rate: 66.16%
Paid Only: Yes
Tags: Stack, 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**. Let `maxDepth` be the **maximum depth** of any integer.

The **weight** of an integer is `maxDepth - (the depth of the integer) + 1`.

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

Example 1:

![](https://assets.leetcode.com/uploads/2021/03/27/nestedlistweightsumiiex1.png)

Input: nestedList = [[1,1],2,[1,1]] **Output:** 8 **Explanation:** Four 1's with a weight of 1, one 2 with a weight of 2. 1*1 + 1*1 + 2*2 + 1*1 + 1*1 = 8

Example 2:

![](https://assets.leetcode.com/uploads/2021/03/27/nestedlistweightsumiiex2.png)

Input: nestedList = [1,[4,[6]]] **Output:** 17 **Explanation:** One 1 at depth 3, one 4 at depth 2, and one 6 at depth 1. 1*3 + 4*2 + 6*1 = 17

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`. * There are no empty lists.

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 depthSumInverse(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
* // Return null 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
* // Return empty list if this NestedInteger holds a single integer
* public List<NestedInteger> getList();
* }
*/
class Solution {
```

```
public int depthSumInverse(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
```

```
# Return None 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
# Return None if this NestedInteger holds a single integer
# :rtype List[NestedInteger]
# """

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