## 339. Nested List Weight Sum

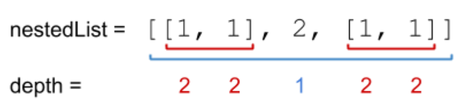
Medium 1017 246 Add to List Share

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**.

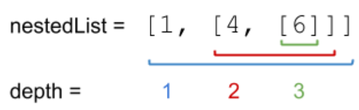
Return *the sum of each integer in* `nestedList` *multiplied by its* ***depth***.

**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`.

Accepted 147,158 | Submissions 186,787

Seen this question in a real interview before? Yes No

Companies

0 ~ 6 months 6 months ~ 1 year 1 year ~ 2 years

Facebook | 28 LinkedIn | 25

Related Topics

Depth-First Search Breadth-First Search

Similar Questions

Nested List Weight Sum II Medium

Array Nesting Medium

Employee Importance Medium

Java Autocomplete

```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 depthSum(List<NestedInteger> nestedList) {
        return dfs(nestedList, 1);
    }

    private int dfs(List<NestedInteger> list, int depth) {
        int total = 0;
        for (NestedInteger nested : list) {
            if (nested.isInteger()) {
                total += nested.getInteger() * depth;
            } else {
                total += dfs(nested.getList(), depth + 1);
            }
        }
        return total;
    }
}


class Solution {
    public int depthSum(List<NestedInteger> nestedList) {
        Queue<NestedInteger> queue = new LinkedList<>();
        queue.addAll(nestedList);

        int depth = 1;
        int total = 0;

        while (!queue.isEmpty()) {
            int size = queue.size();
            for (int i = 0; i < size; i++) {
                NestedInteger nested = queue.poll();
                if (nested.isInteger()) {
                    total += nested.getInteger() * depth;
                } else {
                    queue.addAll(nested.getList());
                }
            }
            depth++;
        }
        return total;
    }
}
```