Nested List Weight Sum

Given a nested list of integers, return the sum of all integers in the list weighted by their depth.

Each element is either an integer, or a list -- whose elements may also be integers or other lists.

Example 1:
Given the list `[[1,1],2,[1,1]]`, return **10**. (four 1's at depth 2, one 2 at depth 1)

Example 2:
Given the list `[1,[4,[6]]]`, return **27**. (one 1 at depth 1, one 4 at depth 2, and one 6 at depth 3; 1 + 4*2 + 6*3 = 27)

Solution 1

```
public int depthSum(List<NestedInteger> nestedList) {
    return helper(nestedList, 1);
}

private int helper(List<NestedInteger> list, int depth)
{
    int ret = 0;
    for (NestedInteger e: list)
    {
        ret += e.isInteger()? e.getInteger() * depth: helper(e.getList(), depth +
1);
    }
    return ret;
}
```

written by larrywang2014 original link here

Solution 2

```
public int depthSum(List<NestedInteger> nestedList) {
    if(nestedList == null){
        return 0;
    }

    int sum = 0;
    int level = 1;

    Queue<NestedInteger> queue = new LinkedList<NestedInteger>(nestedList);
    while(queue.size() > 0){
        int size = queue.size();

        for(int i = 0; i < size; i++){
            NestedInteger ni = queue.poll();

            if(ni.isInteger()){
                sum += ni.getInteger() * level;
            }else{
                queue.addAll(ni.getList());
            }
        }

        level++;
    }

    return sum;
}
```

written by lop original link here

Solution 3

c++:

```
class Solution {
    private:
        int DFS(vector<NestedInteger>& nestedList, int depth){
            int n = (int)nestedList.size();
            int sum = 0;
            for(int i=0;i<n;i++){
                if(nestedList[i].isInteger()){
                    sum += nestedList[i].getInteger()*depth;
                }
                else{
                    sum += DFS(nestedList[i].getList(),depth+1);
                }
            }
            return sum;
        }
    public:
        int depthSum(vector<NestedInteger>& nestedList) {
            return DFS(nestedList, 1);
        }
};
```

Python:

```
class Solution(object):
    def depthSum(self, nestedList):
        """
        :type nestedList: List[NestedInteger]
        :rtype: int
        """
        def DFS(nestedList, depth):
            temp_sum = 0
            for member in nestedList:
                if member.isInteger():
                    temp_sum += member.getInteger() * depth
                else:
                    temp_sum += DFS(member.getList(),depth+1)
            return temp_sum
        return DFS(nestedList,1)
```

Javascript:

```
/**
 * @param {NestedInteger[]} nestedList
 * @return {number}
 */
var dfs = function(nestedList,depth){
    var sum = 0;
    var n = nestedList.length;
    for(var i=0; i<n;i++){
        if(nestedList[i].isInteger()){
            sum += nestedList[i].getInteger() * depth;
        }
        else{
            sum += dfs(nestedList[i].getList(),depth+1);
        }
    }
    return sum;
};
var depthSum = function(nestedList) {
  return dfs(nestedList,1);
};
```

written by sxycwzwzq original link here