**Justin\_DepthfirstSearch\_0339. Nested List Weight Sum**

**Concept:**

查看是否為數字，如果是則將 數字\*depth 加入總和裡

**Code:**

# """

# 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 depthSum(self, nestedList):

return self.add\_sum(nestedList,1)

def add\_sum(self,nestedList,depth):

result = 0

for i in nestedList:

if i.isInteger():

result += i.getInteger() \* depth

else:

result += self.add\_sum(i.getList(), depth+1)

return result