Problem 385: Mini Parser

Problem Information

Difficulty: Medium
Acceptance Rate: 41.43%
Paid Only: No
Tags: String, Stack, Depth-First Search

Problem Description

Given a string s represents the serialization of a nested list, implement a parser to deserialize it and return _the deserialized_ `NestedInteger`.

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

Example 1:

Input: s = "324" **Output:** 324 **Explanation:** You should return a NestedInteger object which contains a single integer 324.

Example 2:

Input: s = "[123,[456,[789]]]" **Output:** [123,[456,[789]]] **Explanation:** Return a NestedInteger object containing a nested list with 2 elements: 1. An integer containing value 123. 2. A nested list containing two elements: i. An integer containing value 456. ii. A nested list with one element: a. An integer containing value 789

Constraints:

* `1 <= s.length <= 5 * 104` * `s` consists of digits, square brackets `"[]"`, negative sign `'-'`, and commas `','`. * `s` is the serialization of valid `NestedInteger`. * All the values in the input are in the range `[-106, 106]`.

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:
NestedInteger deserialize(string s) {

}
};
```

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 NestedInteger deserialize(String s) {

}
}
```

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 deserialize(self, s: str) -> NestedInteger:
```