85. Mini Parser

QuestionEditorial Solution

My Submissions

- Total Accepted: **2154**
- Total Submissions: **8088**
- Difficulty: **Medium**

Given a nested list of integers represented as a string, implement a parser to deserialize it.

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

Note: You may assume that the string is well-formed:

- String is non-empty.
- String does not contain white spaces.
- String contains only digits 0-9, [, - „].

Example 1:

Given s = "324",You should return a NestedInteger object which contains a single integer 324.

Example 2:

Given s = "[123,[456,[789]]]",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.

```
//C++
class Solution {
public:
    NestedInteger deserializeUtil(string s, int& pos) {
        NestedInteger result;

        while(s[pos] != ']' && pos < s.length()) {
            if(s[pos] == '[') {
                result.add(deserializeUtil(s, ++pos));
```

```
                pos++;
                if(pos < s.length() && s[pos] == ',')
                    pos++;
            }
            else {
                string num;
                while(s[pos] != ']' && s[pos] != ',' && pos < s.length())
                    num += s[pos++];

                NestedInteger nestedInt(stoi(num));
                result.add(nestedInt);

                if(s[pos] == ']')
                    return result;

                pos++;
            }
        }
        return result;
    }

    NestedInteger deserialize(string s) {
        int pos = 0;
        if(s[0] == '[')
            return deserializeUtil(s, ++pos);

        return NestedInteger(stoi(s));
    }
};
```