# 385. Mini Parser

## 题目描述：https://leetcode.com/problems/mini-parser/

> 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.
> ```

## 解题思路：

按照栈模式从上到下依次处理

## 代码：

```
/**
 * // 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 s
ingle 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 t
o it.
 *     void add(const NestedInteger &ni);
 *
 *     // Return the nested list that this NestedInteger holds, if it holds a nest
ed list
 *     // The result is undefined if this NestedInteger holds a single integer
 *     const vector<NestedInteger> &getList() const;
 * };
 */
class Solution {
public:
    NestedInteger deserialize(string s) {
        stack<NestedInteger> st;
        vector<NestedInteger> v;
        int n = 0;
        int sign = 1;
        int valid = false;
        for(int i = 0; i < s.size(); i++) {
            if(isdigit(s[i])) {
                valid = true;
                n = n * 10 + s[i] - '0';
            }
            else if(s[i] == '-') {
                sign = -1;
            }
            else if(s[i] == '[') {
                NestedInteger ni;
                st.push(ni);
            }
            else if(s[i] == ']' || s[i] == ',') {
                if(valid) {
                    NestedInteger ni(n*sign);
                    st.top().add(ni);
                    n = 0; sign = 1; valid = false;
                }
                if(s[i] == ']' && st.size() > 1) {
                    NestedInteger ni = st.top();
```

```
                    st.pop();
                    st.top().add(ni);
                }
            }
        }
        if(st.empty()) {
            NestedInteger ni(n*sign);
            st.push(ni);
        }
        return st.top();
    }
};
```