394. Decode String

题目描述：https://leetcode.com/problems/decode-string/

> 给定一个字符串，按照如下翻译decode：
> k[encoded*string] = encoded*string encoded*string encoded*string ... encoded_string (共k个)
> 例如：

```
s = "3[a]2[bc]", return "aaabcbc".
s = "3[a2[c]]", return "accaccacc".
s = "2[abc]3[cd]ef", return "abcabccdcdcdef"
```

解题思路：

1. 递归dfs
2. stack

代码**dfs**自己写的：

```
class Solution {
public:
    string decodeString(string s) {
        int k = 0;
        string tmp;
        string res;
        for(int i = 0; i < s.size(); i++) {
            while(i < s.size() && s[i] >= '0' && s[i] <= '9') {
                k = k * 10 + s[i++] - '0';
            }
            int j;
            if(s[i] == '[') {
                int l = 0;
                for(j = i+1; j < s.size(); j++) {
                    if(s[j] == '[') {
                        l++;
                    }
                    else if(s[j] == ']') {
                        if(l == 0) {
                            string str = s.substr(i+1, j-i-1);
                            tmp = decodeString(str);
                            break;
                        }
                        l--;
                    }
                }
                while(k-- > 0) {
                    res = res + tmp;
                }
                i = j;
                k = 0;
            }
            else if(k == 0) {
                res = res + s[i];
            }
        }
        return res;
    }

};
```

代码dfs简洁版：

不推荐不写了

代码stack版本：

```
class Solution {
public:
    string decodeString(string s) {
        stack<int> nums;
        stack<string> strs;
        string res, tmp;
        int n = 0;
        int k = s.size();
        for(int i = 0; i < k; i++) {
            while(isdigit(s[i])) {
                n = n * 10 + s[i++] - '0';
            }
            if(s[i] == '[') {
                nums.push(n);
                strs.push(tmp);
                tmp.clear();
                n = 0;
            }
            else if(s[i] == ']') {
                n = nums.top();
                nums.pop();
                while(n-- > 0) {
                    strs.top().append(tmp);
                }
                tmp = strs.top();
                strs.pop();
                n = 0;
            }
            else {
                tmp = tmp + s[i];
            }
        }
        return strs.empty()?tmp:strs.top();
    }
};
```