394. Decode String

Medium 1242 70 Favorite Share

Given an encoded string, return it's decoded string.

The encoding rule is: `k[encoded_string]`, where the *encoded_string* inside the square brackets is being repeated exactly *k* times. Note that *k* is guaranteed to be a positive integer.

You may assume that the input string is always valid; No extra white spaces, square brackets are well-formed, etc.

Furthermore, you may assume that the original data does not contain any digits and that digits are only for those repeat numbers, *k*. For example, there won't be input like `3a` or `2[4]`.

Examples:

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

Accepted 89,071 | Submissions 202,812

Seen this question in a real interview before? Yes No

Contributor

Companies

Related Topics

```java
package Algorithm;

import java.util.Stack;

public class L394 {
    /*
     * 用栈来解决，每次遇到“[”就把"["之前的res塞进stack里，遇到数字就把数字塞进numStack，每次遇到"]"就把stack.pop()
     * 和numStack.pop()个res连起来，然后作为新的res，等待下次被，塞进栈里面，或者被和stack里的元素连接起来，最后返回res即可*/
    String decodeString(String s) {
        int len = s.length();
        if(len == 0 || s == null)
            return "";
        Stack<Integer> numStack = new Stack<Integer>();   Stack<String> strStack = new Stack<String>();
        StringBuilder res = new StringBuilder();
        for(int i = 0; i < len; i ++) {
            char c = s.charAt(i);
            if(c == '[') {
                strStack.push(res.toString());
                res = new StringBuilder();
            }else if (c == ']') {
                int num = numStack.pop();
                StringBuilder temp = new StringBuilder(strStack.pop());
                for(int j = 0; j < num; j ++) {
                    temp.append(res);
                }
                res = temp;
            }else if(Character.isDigit(c)) {
                int num = 0;
                while(i < len && Character.isDigit(s.charAt(i))) {
                    num = num * 10 + s.charAt(i) - '0';
                    i ++;
                }
                i --;
                numStack.push(num);
            }else {
                res.append(c);
            }
        }
        return res.toString();
    }
}

```