1. 题目
2. 题干

给定一个经过编码的字符串，返回它解码后的字符串。

编码规则为: k[encoded\_string]，表示其中方括号内部的 encoded\_string 正好重复 k 次。注意 k 保证为正整数。

你可以认为输入字符串总是有效的；输入字符串中没有额外的空格，且输入的方括号总是符合格式要求的。

此外，你可以认为原始数据不包含数字，所有的数字只表示重复的次数 k ，例如不会出现像 3a 或 2[4] 的输入。

1. 示例

示例 1：

输入：s = "3[a]2[bc]"

输出："aaabcbc"

示例 2：

输入：s = "3[a2[c]]"

输出："accaccacc"

示例 3：

输入：s = "2[abc]3[cd]ef"

输出："abcabccdcdcdef"

1. 题解
2. 思路

对于这种字符串中包含左右括号的题目，第一反应是用栈来解决。当出现括号嵌套的情况，就可以一层一层地展开括号内的内容，复制多份后重新入栈。

如果当前的字符是数字，则解析出一个数字入栈；

如果当前的字符是字母或者左括号，直接入栈；

如果当前的字符是右括号，则说明一个括号内的字符串已经全部入栈了，可以将这些字符出栈，直到碰见左括号。由于栈的特性，出栈后的字符串需要反转，然后取出栈顶的数字，该数字表示这个字符串应该复制的次数，然后将复制后的字符串构造到一个新的字符串中，重新入栈。

1. 代码

Java：

class Solution {

    int ptr = 0;

    public String decodeString(String s) {

        //用栈的思路来解决

        //新建一个链表

        LinkedList<String> stk = new LinkedList<String>();

        //遍历一遍s即可解决

        while(ptr < s.length()){

            char cur = s.charAt(ptr);

            //当前字符是数时，判断数字字符的个数，组成数字入栈

            if(Character.isDigit(cur)){

                //获取一个数字入栈

                String digits = getDigits(s);

                stk.addLast(digits);

            }

            //当前字符是字母或者左括号，直接入栈

            else if(Character.isLetter(cur) || cur =='['){

                stk.addLast(Character.toString(cur));

                ptr++;

            }

            //当前字符是右括号

            else{

                ptr++;

                //用新的LinkedList保存出栈的字符串

                LinkedList<String> sub = new LinkedList<>();

                //出栈，直到遇到左括号

                while(!"[".equals(stk.peekLast())){

                    sub.addLast(stk.removeLast());

                }

                //反转sub

                Collections.reverse(sub);

                //将左括号出栈

                stk.removeLast();

                //此时的栈顶为数字,出栈

                int repTime = Integer.parseInt(stk.removeLast());

                //复制repTime次字符串

                StringBuilder t = new StringBuilder();

                String o = getString(sub);

                while(repTime-- > 0){

                    t.append(o);

                }

                //再次入栈

                stk.addLast(t.toString());

            }

        }

        return getString(stk);

    }

    public String getDigits(String s){

        //多位数字直接用StringBuilder返回toString即可

        StringBuilder ret = new StringBuilder();

        while(Character.isDigit(s.charAt(ptr))){

            ret.append(s.charAt(ptr++));

        }

        return ret.toString();

    }

    public String getString(LinkedList<String> v){

        StringBuilder ret = new StringBuilder();

        for(String s : v){

            ret.append(s);

        }

        return ret.toString();

    }

}