1.面试题 03.04. 化栈为队

题目链接

```java
Stack<Integer> inStack = new Stack();
Stack<Integer> outStack = new Stack();

/**
 * Initialize your data structure here.
 */
public MyQueue() {

}

/**
 * Push element x to the back of queue.
 */
public void push(int x) {
    inStack.push(x);
}

public void transfer() {
    if (!outStack.isEmpty()) return;
    while (!inStack.isEmpty()) {
        outStack.push(inStack.pop());
    }
}

/**
 * Removes the element from in front of queue and returns that element.
 */
public int pop() {
    transfer();
    return outStack.pop();
}

/**
 * Get the front element.
 */
public int peek() {
    transfer();
    return outStack.peek();
}

/**
 * Returns whether the queue is empty.
 */
public boolean empty() {
    return outStack.empty() && inStack.empty();
}
```

2.LeetCode 682 棒球比赛

题目链接

```java
public int calPoints(String[] ops) {
    Stack<Integer> stack = new Stack<>();
    for (String op : ops) {
        if (op.equals("+")) {
            int a = stack.pop();
            int b = stack.peek() + a;
            stack.push(a);
            stack.push(b);
        } else if (op.equals("D")) {
            stack.push(2 * stack.peek());
        } else if (op.equals("C")) {
            stack.pop();
        } else {
            stack.push(Integer.parseInt(op));
        }
    }
    int result = 0;
    for (Integer integer : stack) {
        result += integer;
    }
    return result;
}
```

3.LeetCode 844. 比较含退格的字符串

题目链接

```java
public boolean backspaceCompare(String S, String T) {
    return backspace(S).equals(backspace(T));
}

public String backspace(String str) {
    StringBuilder stringBuilder = new StringBuilder();
    for (int i = 0; i < str.length(); i++) {
        char ch = str.charAt(i);
        if (ch == '#') {
            if (stringBuilder.length() > 0)
                stringBuilder.deleteCharAt(stringBuilder.length() - 1);
        } else {
            stringBuilder.append(ch);
        }
    }
    return stringBuilder.toString();
}
```

双指针法

```java
ic static boolean backspaceCompare(String S, String T) {
int countS = 0, countT = 0;
int i = S.length() - 1, j = T.length() - 1;
while (i >= 0 || j >= 0) {
    while (i >= 0) {
        if (S.charAt(i) == '#') {
            countS++;
            i--;
        } else if (countS > 0) {
            i--;
            countS--;
        } else {
            break;
        }
    }
    while (j >= 0) {
        if (T.charAt(j) == '#') {
            countT++;
            j--;
        } else if (countT > 0) {
            j--;
            countT--;
        } else {
            break;
        }
    }
    if (i >= 0 && j >= 0) {
        if (S.charAt(i) != T.charAt(j)) return false;
    } else {
        if (i >= 0 || j >= 0) return false;
    }
    i--;
    j--;
}
return true;
```

4.LeetCode 946 验证栈序列

题目链接

```java
public boolean validateStackSequences(int[] pushed, int[] popped) {
    Stack<Integer> stack = new Stack<>();
    int j = 0;
    for (int n : pushed) {
        stack.push(n);
        while (!stack.isEmpty() && j < popped.length && stack.peek() ==
ed[j]) {
            stack.pop();
            j++;
        }
    }
    return stack.isEmpty();
}
```

5.LeetCode 20 有效的括号

题目链接

```java
public boolean isValid(String s) {
    HashMap<Character, Character> map = new HashMap<>();
    map.put(')', '(');
    map.put(']', '[');
    map.put('}', '{');
    Stack<Character> stack = new Stack<>();
    for (int i = 0; i < s.length(); i++) {
        switch (s.charAt(i)) {
            case '(':
            case '[':
            case '{':
                stack.push(s.charAt(i));
                break;
            case ')':
            case ']':
            case '}':
                if (stack.isEmpty() || map.get(s.charAt(i)) != stack.peek())
rn false;
                stack.pop();
                break;
        }
    }
    return stack.isEmpty();
}
```

6.LeetCode 1021 删除最外层的括号

题目链接

```java
public String removeOuterParentheses(String S) {
    StringBuilder stringBuilder = new StringBuilder();
    for (int i = 0, pre = 0, count = 0; i < S.length(); i++) {
        if (S.charAt(i) == '(') {
            count++;
        } else {
            count--;
        }
        if (count != 0) continue;
        stringBuilder.append(S.substring(pre + 1, i));
        pre = i + 1;
    }
    return stringBuilder.toString();
}
```

进阶

```java
s Solution {
public String removeOuterParentheses(String S) {
    StringBuilder sb = new StringBuilder();
    int level = 0;
    for (char c : S.toCharArray()) {
        if (c == ')') --level;
        if (level >= 1) sb.append(c);
        if (c == '(') ++level;
    }
    return sb.toString();
}

```

7.LeetCode 1249. 移除无效的括号

题目链接

船长思路

```java
 public String minRemoveToMakeValid(String s) {
        char[] t = new char[s.length()];
        char[] ans = new char[s.length()];
        int tlen = 0;
        for (int i = 0, cnt = 0; i < s.length(); i++) {
            if (s.charAt(i) != ')') {
                if (s.charAt(i) == '(') cnt++;
                t[tlen++] = s.charAt(i);
            } else {
                if (cnt == 0) continue;
                cnt--;
                t[tlen++] = ')';
            }
        }
        int ansHead = tlen;
        for (int i = tlen - 1, cnt = 0; i >= 0; i--) {
            if (t[i] != '(') {
                if (t[i] == ')') cnt++;
                ans[--ansHead] = t[i];
            } else {
                if (cnt == 0) continue;
                cnt--;
                ans[--ansHead] = '(';
            }
        }
        return new String(ans).trim();
    }
```

小李思路

```java
   public String minRemoveToMakeValid(String s) {
        StringBuilder stringBuilder = new StringBuilder(s);
        Deque<Integer> stack = new LinkedList<>();
        for (int i = 0; i < s.length(); i++) {
            if (s.charAt(i) == '(') {
                stack.push(i);
            } else if (s.charAt(i) == ')') {
                if (!stack.isEmpty() && s.charAt(stack.peek()) == '(') {
                    stack.pop();
                } else stack.push(i);
            }
        }
        for (Integer integer : stack) {
            stringBuilder.deleteCharAt(integer);
        }
        return stringBuilder.toString();
    }
```

8.LeetCode 145. 二叉树的后序遍历

题目链接

迭代法

```java
    public List<Integer> postorderTraversal(TreeNode root) {
        List<Integer> result = new ArrayList<>();
        if (root == null) return result;
        Deque<TreeNode> stack = new LinkedList<>();
        Deque<Integer> statusStack = new LinkedList<>();
        stack.push(root);
        statusStack.push(0);
        while (!stack.isEmpty()) {
            switch (statusStack.pop()) {
                case 0: {
                    statusStack.push(1);
                    if (stack.peek().left != null) {
                        stack.push(stack.peek().left);
                        statusStack.push(0);
                    }
                    break;
                }
                case 1: {
                    statusStack.push(2);
                    if (stack.peek().right != null) {
                        stack.push(stack.peek().right);
                        statusStack.push(0);
                    }
                    break;
                }
                case 2: {
                    result.add(stack.pop().val);
                    break;
                }
            }
        }
        return result;
    }
```

9.LeetCode 331 验证二叉树的前序序列化

题目链接

船长思路

```java
    public boolean isValidSerialization(String preorder) {
        String[] strings = preorder.split(",");
        List<String> list = new ArrayList<>();
        for (int i = 0; i < strings.length; i++) {
            list.add(strings[i]);
            int lastIndex = list.size() - 1;
            while (list.size() >= 3 && list.get(lastIndex).equals("#") &&
                    list.get(lastIndex - 1).equals("#") && !list.get(lastIndex - 2).equals("#")) {
                list.set(lastIndex - 2, "#");
                list.remove(lastIndex);
                list.remove(lastIndex - 1);
                lastIndex = list.size() - 1;
            }
        }
        return list.size() == 1 && list.get(0).equals("#");
    }
```

小李思路

```java
    public boolean isValidSerialization(String preorder) {
        int i = 0;
        int slots = 1;
        int length = preorder.length();
        while (i < length) {
            if (slots == 0) return false;
            if (preorder.charAt(i) == ',') {
                i++;
            } else if (preorder.charAt(i) == '#') {
                slots--;
                i++;
            } else {
                while (i < length && preorder.charAt(i) != ',') {
                    i++;
                }
                slots++;
            }
        }
        return slots == 0;
    }
```

10.LeetCode 227 基本计算器II

题目链接

```
    public int calculate(String s) {
        Deque<Integer> stack = new LinkedList<>();
        char preSign = '+';
        int num = 0;
        int n = s.length();
        for (int i = 0; i < n; i++) {
            // '153' 数字 153
            if (Character.isDigit(s.charAt(i))) {
                num = num * 10 + s.charAt(i) - '0';
            }
            if (!Character.isDigit(s.charAt(i)) && s.charAt(i) != ' ' || i == n
- 1) {
                switch (preSign) {
                    case '+': {
                        stack.push(num);
                        break;
                    }
                    case '-': {
                        stack.push(-num);
                        break;
                    }
                    case '*': {
                        stack.push(stack.pop() * num);
                        break;
                    }
                    case '/': {
                        stack.push(stack.pop() / num);
                        break;
                    }
                }
                preSign = s.charAt(i);
                num = 0;
            }
        }
        int result = 0;
        while (!stack.isEmpty()) {
            result += stack.pop();
        }
        return result;
    }
```

11.LeetCode 636 函数的独占时间

题目链接

```
    class Task {
        int id = 0;
        int time = 0;
        boolean isStart = true;

        Task(String[] split) {
            id = Integer.valueOf(split[0]);
            time = Integer.valueOf(split[2]);
            isStart = split[1].equals("start");
        }
    }

    public int[] exclusiveTime(int n, List<String> logs) {
        Deque<Task> stack = new LinkedList<Task>();
        int[] ans = new int[n];
        for (String log : logs) {
            Task task = new Task(log.split(":"));
            if (task.isStart) {
                stack.push(task);
            } else {
                Task last = stack.pop();
                int duration = task.time - last.time + 1;
                ans[task.id] += duration;
                if (!stack.isEmpty()) {
                    ans[stack.peek().id] -= duration;
                }
            }
        }
        return ans;
    }
```

12.LeetCode 1124. 表现良好的最长时间段

题目链接

```
    public int longestWPI(int[] hours) {
        int sum = 0;
        int res = 0;
        HashMap<Integer, Integer> sumToIndex = new HashMap<>();
        for (int i = 0; i < hours.length; i++) {
            if (hours[i] > 8) {
                sum++;
            } else {
                sum--;
            }
            if (sum > 0) {
                res = i + 1;
            } else {
                if (!sumToIndex.containsKey(sum)) {
                    sumToIndex.put(sum, i);
                }
                if (sumToIndex.containsKey(sum - 1)) {
                    res = Math.max(res, i - sumToIndex.get(sum - 1));
                }
            }
        }
        return res;
    }
```