# 第三周 - 递归与栈


第三周 - 递归与栈
- 栈的基础知识
- 经典面试题
  - 栈的基本操作
    - LeetCode #面试题03.04 化栈为队
    - LeetCode #682 棒球比赛
    - LeetCode #844 比较含退格的字符串
    - LeetCode #946 验证栈序列
  - 栈结构扩展应用
    - LeetCode #20 有效的括号
    - LeetCode #1021 删除最外层的括号
    - LeetCode #1249 移除无效的括号
    - LeetCode #145 二叉树的后序遍历
    - LeetCode #331 验证二叉树的前序序列化
    - LeetCode #227 基本计算器II
  - 智力发散题
    - LeetCode #636 函数的独占时间
    - LeetCode #1124 表现良好的最长时间段


## 栈的基础知识

栈是一种“先进后出”（FILO, First In Last Out）的数据结构。

## 经典面试题

### 栈的基本操作

#### LeetCode #面试题03.04 化栈为队

利用两个栈来实现，一个输入栈、一个输出栈。

输入栈用于读入数据。当需要输出元素时，若输出栈为空，则将输入栈的所有元素推送到输出栈，然后取栈顶元素；若输出栈非空，则输出栈顶即可。

输出栈的作用是对已经被反转的序列进行二次反转。对此感到困惑的同学可以画图模拟一下。

### LeetCode #682 棒球比赛

### LeetCode #844 比较含退格的字符串

按照题意实现即可。

### LeetCode #946 验证栈序列

被出栈的元素只有两种可能：即将入栈的元素 和 当前栈顶的元素。

只需要关注出栈序列，分类讨论后模拟即可。

## 栈结构扩展应用

### LeetCode #20 有效的括号

结论1：在任意一个位置上，左括号数量 $\geq$ 右括号数量

结论2：在最后一个位置上，左括号数量 $==$ 右括号数量

根据上述两个结论，程序中只需要记录左括号和右括号的数量即可。

一对括号可以等价为一个完整的事件。左括号可以看作事件的开始、右括号可以看作事件的结束。而括号序列可以看作事件与事件之间的完全包含关系。

栈可以处理具有**完全包含关系**的问题。

### LeetCode #1021 删除最外层的括号

左括号和右括号差值为0时，代表这一串括号序列是独立的，可以被单独分解出来。

### LeetCode #1249 移除无效的括号

可以被匹配的括号都是有效的，而其他的括号都需要被删除。

### LeetCode #145 二叉树的后序遍历

递归做法比较简单，在这里介绍一下基于迭代的做法。

技巧是使用两个栈，一个数据栈，一个状态栈。将“遍历左子树”，“遍历右子树”和“访问根节点”三个步骤分别用状态码表示，枚举状态转移过程，使用有限状态自动机（FSM, Finite State Machine）的模型来模拟递归过程。

### LeetCode #331 验证二叉树的前序序列化

思路1：每次拆掉一个“数字、#、#”的节点（即叶子结点），最后树上的全部节点都会被拆光（即只剩一个“#”），能拆光的序列就是合法序列。

思路2：初始状态有一个坑。每多增加一个数字节点，会在占掉一个坑后，产生两个坑，每多增加一个#，会减少一个坑。合法的二叉树前序遍历最后会刚好用完所有的坑。

### LeetCode #227 基本计算器II

思路1：找到式子中优先级最低的运算符，然后递归分治运算两侧的子式即可。

思路2：使用操作数栈和操作符栈辅助计算，当操作符栈遇到更低优先级的操作符时，需要将之前更高级别的操作符对应的结果计算出来。

对于有括号的情况，左括号相当于提高了内部全部运算符的优先级，当遇到右括号的时候需要将匹配的括号间的内容全部计算出来。

可以通过加一个特殊操作符的处理技巧，来额外处理结尾的数字。

## 智力发散题

### LeetCode #636 函数的独占时间

本质就是一道模拟题，画一个线段图能显著地辅助理解。任务开始时进栈，上一个任务暂停执行；任务完成时出栈，恢复上一个任务的执行。

## LeetCode #1124 表现良好的最长时间段

把表现“良好”记为+1，把表现“不好”记为-1，将原序列转化为正负1的序列，原问题转化为求转化后序列的最长一段连续子序列，使得子序列的和大于0。

在这里引入“前缀和”的技巧。前缀和数组的第n项，是原数组前n项的和。

记原数组为a，那么前缀和$\mathrm{prefix}[n] = \sum_{i=1}^{n} \mathrm{a}[i]$。这样就可以将“区间和”转化为“前缀和”之差来计算。前缀和可以视情况补一个前导0，表示前0项之和，即不取任何元素的情况。

在本题中，a数组中的元素只有-1和1，因此前缀和prefix的变化一定是连续的。我们记录下前缀和中，每一个前缀和第一次出现的位置，它对应的位置一定是从该前缀和出发的最优解。

我们以$\mathrm{f}(n)$表示以$n$结尾的序列的最大长度，$\mathrm{pos}(n)$表示前缀和$n$第一次出现的位置。那么有$\mathrm{f}(n) = \mathrm{f}(n-1) + \mathrm{pos}(n) - \mathrm{pos}(n-1)$。