从递归入手二维动态规划

前置知识:
讲解038-经典递归过程解析、讲解066-从递归入手一维动态规划

本节课:
讲解从递归到二维动态规划的过程
讲解二维动态规划的空间压缩技巧
讲解哪些递归不适合或者说没有必要改成动态规划

下节课:直接从动态规划的定义入手,来见识更多二维动态规划问题

注意:
二维动态规划问题非常多,不仅讲解067、讲解068涉及,整个系列课程会大量涉及
【必备】课程后续会讲背包dp、区间dp、状压dp等等,依然包含大量二维动态规划问题

从递归入手二维动态规划

尝试函数有1个可变参数可以完全决定返回值，进而可以改出1维动态规划表的实现
同理
尝试函数有2个可变参数可以完全决定返回值，那么就可以改出2维动态规划的实现

一维、二维、三维甚至多维动态规划问题，大体过程都是：
写出尝试递归
记忆化搜索(从顶到底的动态规划)
严格位置依赖的动态规划(从底到顶的动态规划)
空间、时间的更多优化

从递归入手二维动态规划

动态规划表的大小：每个可变参数的可能性数量相乘
动态规划方法的时间复杂度：动态规划表的大小 * 每个格子的枚举代价

二维动态规划依然需要去整理 动态规划表的格子之间的依赖关系
找寻依赖关系，往往 通过画图来建立空间感，使其更显而易见
然后依然是 从简单格子填写到复杂格子 的过程，即严格位置依赖的动态规划(从底到顶)

二维动态规划的压缩空间技巧原理不难，会了之后千篇一律
但是不同题目依赖关系不一样，需要 很细心的画图来整理具体题目的依赖关系
最后进行空间压缩的实现

从递归入手二维动态规划

能改成动态规划的递归，统一特征：
决定返回值的可变参数类型往往都比较简单，一般不会比*int*类型更复杂。为什么？

从这个角度，可以解释 带路径的递归（可变参数类型复杂），不适合或者说没有必要改成动态规划
题目2就是说明这一点的

一定要 写出可变参数类型简单（不比*int*类型更复杂），并且 可以完全决定返回值的递归，
保证做到 这些可变参数可以完全代表之前决策过程对后续过程的影响！ 再去改动态规划！

不管几维动态规划
经常从递归的定义出发，避免后续进行很多边界讨论
这需要一定的经验来预知

从递归入手二维动态规划

题目1
最小路径和
给定一个包含非负整数的 m x n 网格 grid
请找出一条从左上角到右下角的路径，使得路径上的数字总和为最小。
说明：每次只能向下或者向右移动一步。
测试链接：https://leetcode.cn/problems/minimum-path-sum/

从递归入手二维动态规划

题目2
单词搜索（无法改成动态规划）
给定一个 m x n 二维字符网格 board 和一个字符串单词 word
如果 word 存在于网格中，返回 true ；否则，返回 false 。
单词必须按照字母顺序，通过相邻的单元格内的字母构成
其中"相邻"单元格是那些水平相邻或垂直相邻的单元格
同一个单元格内的字母不允许被重复使用
测试链接：https://leetcode.cn/problems/word-search/

从递归入手二维动态规划

题目3
最长公共子序列
给定两个字符串text1和text2
返回这两个字符串的最长 公共子序列 的长度
如果不存在公共子序列，返回0
两个字符串的 公共子序列 是这两个字符串所共同拥有的子序列
测试链接 : https://leetcode.cn/problems/longest-common-subsequence/

从递归入手二维动态规划

题目4
最长回文子序列
给你一个字符串 s ，找出其中最长的回文子序列，并返回该序列的长度
测试链接：https://leetcode.cn/problems/longest-palindromic-subsequence/

从递归入手二维动态规划

题目5
节点数为n高度不大于m的二叉树个数
现在有n个节点，计算出有多少个不同结构的二叉树
满足节点个数为n且树的高度不超过m的方案
因为答案很大，所以答案需要模上1000000007后输出
测试链接：https://www.nowcoder.com/practice/aaefe5896cce4204b276e213e725f3ea

左程云

从递归入手二维动态规划

题目6
矩阵中的最长递增路径
给定一个 m x n 整数矩阵 matrix ，找出其中 最长递增路径 的长度
对于每个单元格，你可以往上，下，左，右四个方向移动
不能在对角线方向上移动或移动到边界外（即不允许环绕）
测试链接：https://leetcode.cn/problems/longest-increasing-path-in-a-matrix/