



第五周 (3.1-3.7)

📅 StartDate	@2021/03/01 → 2021/03/07
✔ Status	已完成
☰ 内容	动态规划（一）、期中
☰ 总结	
↗ 题目	

期中总结

- Array 数组
- Linked List 链表
- Stack 栈
- Queue 队列
- HashTable 哈希表
- Set Map
- Tree 二叉树
- BST 二叉搜索树
- Search 查询
- Recursion 递归
- DFS 广度优先搜索
- BFS 深度优先搜索
- Divide & Conquer 分治
- Backtracking 回溯
- Greedy 贪心
- Binary Search 二叉查找
- DP 动态规划

Array 数组
Linked List 链表 (动画)
Stack 栈
Queue 队列
HashTable 哈希表 (动画)
Set、Map
Tree 二叉树
BST 二叉搜索树 (动画)

Search 查询
Recursion 递归 (动画)
DFS 深度优先搜索
BFS 广度优先搜索
Divide & Conquer 分治
Backtracking 回溯
Greedy 贪心
Binary Search 二叉查找

动态规划 (一)

1. 动态规划的原理讲解

动态规划 Dynamic Programming



1. Wiki 定义：
https://en.wikipedia.org/wiki/Dynamic_programming

2. “Simplifying a complicated problem by breaking it down into simpler sub-problems”
(in a recursive manner)

3. Divide & Conquer + Optimal substructure
分治 + 最优子结构

理解动态规划的关键点是什么？有哪些误区？

- 有无最优的子结构。所谓最优子结构：推导出的第n步的值，是前面几个值的最佳值的简单累加，或者最大值，或者最小值。
- 找到重复子问题
- 求最优状态，淘汰次优状态（降低后面处理的复杂度）。

2. DP例题解析：

- Fibonacci数列

自顶向下的编程思路：递归+记忆化搜索

自底向上的编程思路：直接从初始值开始进行循环（动态递推）

- 路径计数

- 65. 不同路径

题目描述 评论(916) 题解(1001) 提交记录 Java

63. 不同路径 II

难度 中等 448 收藏 分享 切换为英文 接收动态反馈

一个机器人位于一个 $m \times n$ 网格的左上角（起始点在下图中标记为“Start”）。
机器人每次只能向下或者向右移动一步。机器人试图达到网格的右下角（在下图中标记为“Finish”）。
现在考虑网格中有障碍物。那么从左上角到右下角将会有多少条不同的路径？

示例 1:

网格中的障碍物和空位置分别用 1 和 0 来表示。

```
class Solution {
    public int uniquePathsWithObstacles(int[][] obstacleGrid) {
        int m = obstacleGrid.length;
        int n = obstacleGrid[0].length;
        int[] dp = new int[n + 1];
        dp[1] = 1;

        for(int i = 0; i < m; i++) {
            for(int j = 1; j <= n; j++) {
                if (obstacleGrid[i][j - 1] == 1) // 障碍物
                    dp[j] = 0;
                else
                    dp[j] = dp[j] + dp[j - 1]; // dp[i-1][j] + dp[i][j-1]
            }
        }
        return dp[n];
    }
}
```

您上次编辑到这里，代码已从您浏览器本地的临时存储中恢复了 [还原默认代码模版](#)

- 1143.最长子序列

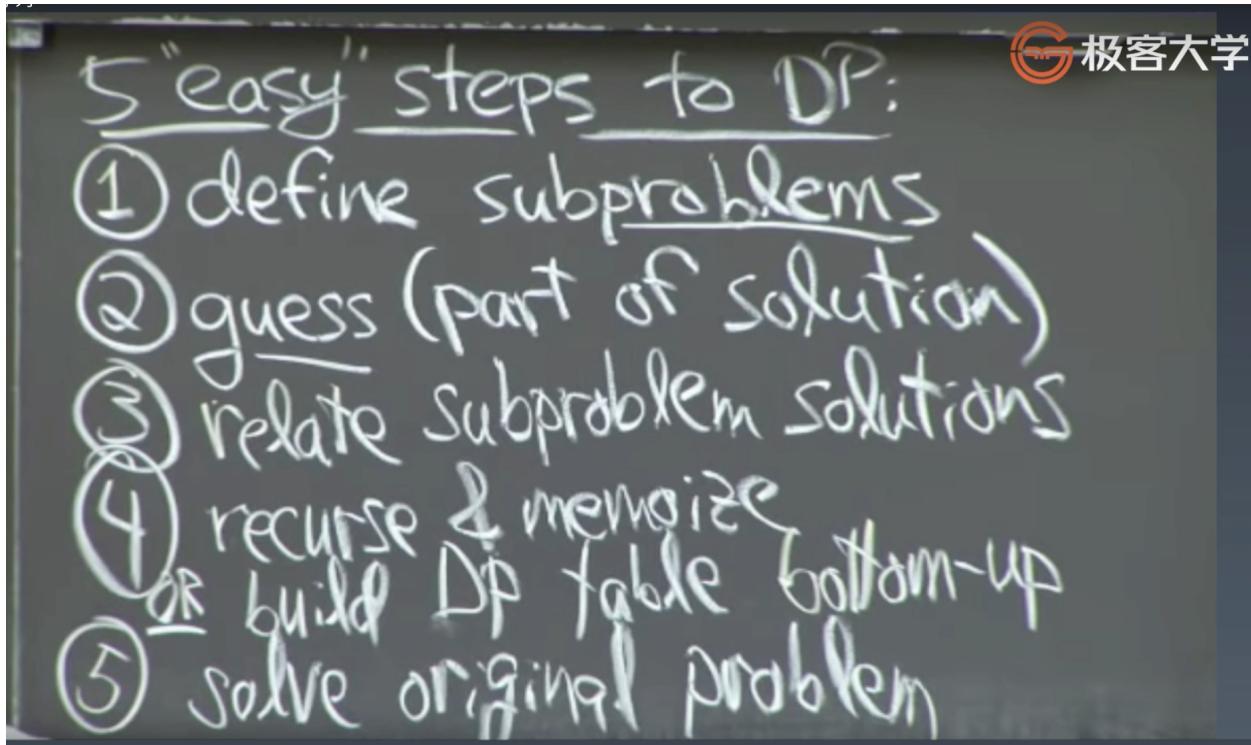
动态规划关键点



1. 最优子结构 $\text{opt}[n] = \text{best_of}(\text{opt}[n-1], \text{opt}[n-2], \dots)$
2. 储存中间状态: $\text{opt}[i]$
3. 递推公式 (美其名曰: 状态转移方程或者 DP 方程)

Fib: $\text{opt}[i] = \text{opt}[i-1] + \text{opt}[i-2]$

二维路径: $\text{opt}[i,j] = \text{opt}[i+1][j] + \text{opt}[i][j+1]$ (且判断 $a[i,j]$ 是否空地)



MIT 动态规划课程最短路径算法