动态规划（Dynamic Programming）

- 动态规划，简称DP
- 是求解最优化问题的一种常用策略

- 通常的使用套路（一步一步优化）

① 暴力递归（自顶向下，出现了重叠子问题）

② 记忆化搜索（自顶向下）

③ 递推（自底向上）

动态规划的常规步骤

■ 动态规划中的“动态”可以理解为是“会变化的状态”

① 定义状态（状态是原问题、子问题的解）

✓比如定义 dp(i) 的含义

② 设置初始状态（边界）

✓比如设置 dp(0) 的值

③ 确定状态转移方程

✓比如确定 dp(i) 和 dp(i – 1) 的关系

动态规划的一些相关概念

- 来自维基百科的解释

Dynamic Programming is a method for solving a complex problem by breaking it down into a collection of simpler subproblems, solving each of those subproblems just once, and storing their solutions.

① 将复杂的原问题拆解成若干个简单的子问题

② 每个子问题仅仅解决1次，并保存它们的解

③ 最后推导出原问题的解

- 可以用动态规划来解决的问题，通常具备2个特点

最优子结构（最优化原理）：通过求解子问题的最优解，可以获得原问题的最优解

无后效性

✓ 某阶段的状态一旦确定，则此后过程的演变不再受此前各状态及决策的影响（未来与过去无关）

✓ 在推导后面阶段的状态时，只关心前面阶段的具体状态值，不关心这个状态是怎么一步步推导出来的

无后效性

(0, 0)				
		(i, j–1)		
	(i–1, j)	(i, j)		
				(4, 4)

- 从起点（0, 0）走到终点（4, 4）一共有多少种走法？只能向右、向下走

- 假设 dp(i, j) 是从（0, 0）走到（i, j）的走法
  - dp(i, 0) = dp(0, j) = 1
  - dp(i, j) = dp(i, j – 1) + dp(i – 1, j)

- 无后效性
  - 推导 dp(i, j) 时只需要用到 dp(i, j – 1)、dp(i – 1, j) 的值
  - 不需要关心 dp(i, j – 1)、dp(i – 1, j) 的值是怎么求出来的

有后效性

(0, 0)				
		(i, j–1)		
	(i–1, j)	(i, j)		
				(4, 4)

- 如果可以向左、向右、向上、向下走，并且同一个格子不能走 2 次

- 有后效性
  - dp(i, j) 下一步要怎么走，还要关心上一步是怎么来的
    - ✓ 也就是还要关心 dp(i, j – 1)、dp(i – 1, j) 是怎么来的?