动态规划入门

信息竞赛组

广州大学附属中学

2024 年 7 月

目录

动态规划

- 动态规划（DP）不是某种具体算法，而是一种思想。

动态规划

- 动态规划（DP）不是某种具体算法，而是一种思想。
- 核心在于：把大问题转化为小问题，利用小问题的解推断出大问题的解。

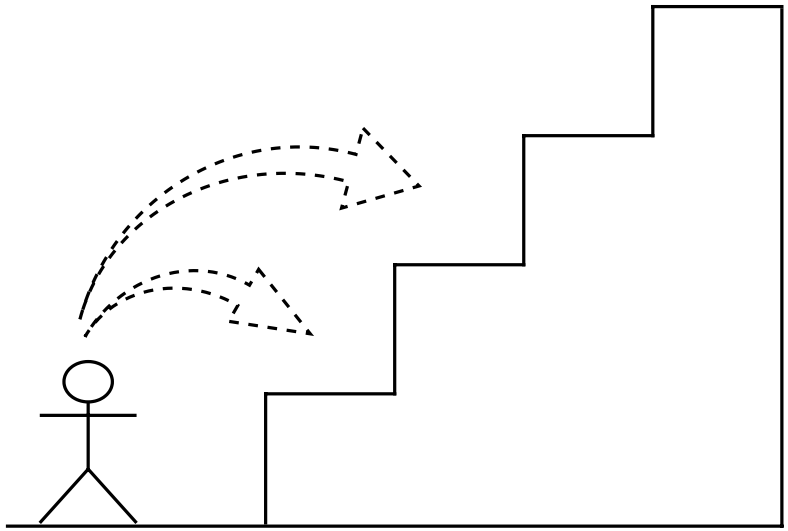
动态规划

- 动态规划（DP）不是某种具体算法，而是一种思想。
- 核心在于：把大问题转化为小问题，利用小问题的解推断出大问题的解。
- 带着这种思想，我们来学习 DP.

目录

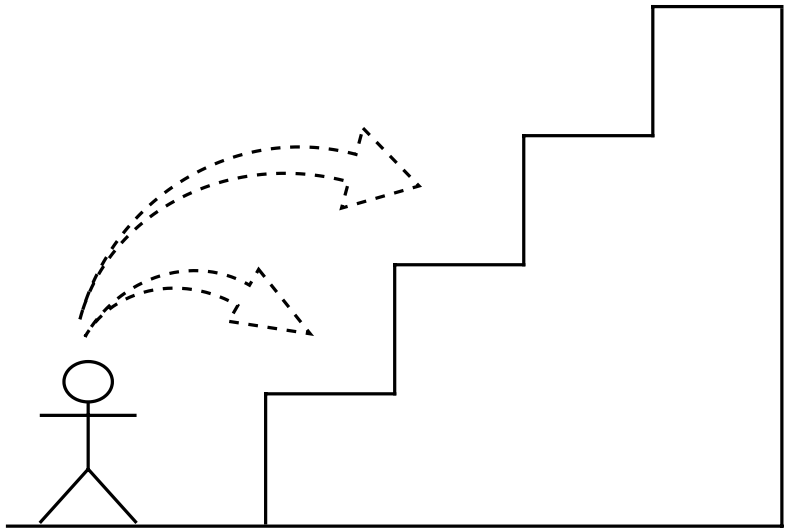
爬楼梯问题

- 有一个 n 层级高的楼梯，每次只能上 1 步或者 2 步台阶，从下往上走，一共多少种走法？



爬楼梯问题

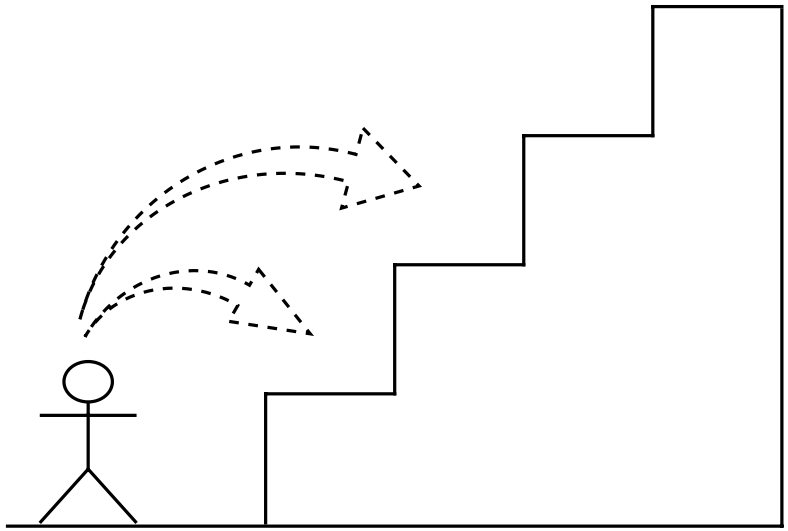
- 有一个 n 层级高的楼梯，每次只能上 1 步或者 2 步台阶，从下往上走，一共多少种走法？



- 暴力模拟：指数级复杂度。

爬楼梯问题

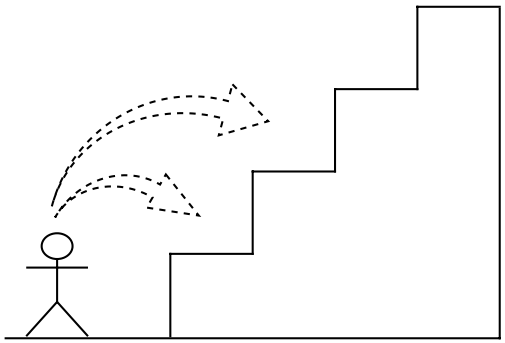
- 有一个 n 层级高的楼梯，每次只能上 1 步或者 2 步台阶，从下往上走，一共多少种走法？



- 暴力模拟：指数级复杂度。
- 考虑更优的算法。如果以 $f[i]$ 表示“从 0 级走到 i 级的方案数”，假设 $f[1], f[2] \ldots f[n-1]$ 全都已知，如何利用这些信息推出 $f[n]$？

爬楼梯问题

- 考虑：走到第 10 层台阶的上一步位置有几种情况？



两种情况：$8 \to 10$ 或 $9 \to 10$；

$$f[10] = f[8] + f[9]$$
$$f[9] = f[8] + f[7]$$

- 走到 $f[n]$，要么是从 $n-1$ 级走上来的，要么是从 $n-2$ 级来的。依据加法原理

$$f[n] = f[n-1] + f[n-2]$$

这就是这个问题的递推式。

- 初始化：当走上第 1 级台阶，$f[1] = 1$，当走上第 2 级台阶，$f[2] = 2$

目录

硬币问题

- 今天你手上有无限的面值为 $1, 5, 11$ 元的硬币。

硬币问题

- 今天你手上有无限的面值为 $1, 5, 11$ 元的硬币。
- 给定 n, 问: 至少用多少枚硬币, 可以恰好凑出 n 元?

硬币问题

- 今天你手上有无限的面值为 $1, 5, 11$ 元的硬币。
- 给定 n, 问: 至少用多少枚硬币, 可以恰好凑出 n 元?
- 例如:

硬币问题

- 今天你手上有无限的面值为 $1, 5, 11$ 元的硬币。
- 给定 n, 问: 至少用多少枚硬币, 可以恰好凑出 n 元?
- 例如:
  - $n = 15$ 时答案是 3, 构造方法为 $5 + 5 + 5$

硬币问题

- 今天你手上有无限的面值为 $1, 5, 11$ 元的硬币。
- 给定 n, 问: 至少用多少枚硬币, 可以恰好凑出 n 元?
- 例如:
  - $n = 15$ 时答案是 3 , 构造方法为 $5 + 5 + 5$
  - $n = 12$ 时答案是 2 , 构造方法为 $11 + 1$

硬币问题

- 用 $f[i]$ 记录“凑出 i 元所需要的硬币数”。那么答案显然就是 $f[n]$。

硬币问题

- 用 $f[i]$ 记录“凑出 i 元所需要的硬币数”。那么答案显然就是 $f[n]$。
- 如何求出 f 数组呢？$f[i]$ 等于什么呢？

硬币问题

- 用 $f[i]$ 记录“凑出 i 元所需要的硬币数”。那么答案显然就是 $f[n]$。
- 如何求出 f 数组呢？$f[i]$ 等于什么呢？
- 提示：注意思考“$f[i]$ 从哪里来”。

硬币问题

- 考虑一个具体的例子：凑出 15 元。

硬币问题

- 考虑一个具体的例子：凑出 15 元。
- 为了凑出 15 元，我们最开始的时候，可以使用哪枚硬币？

硬币问题

- 考虑一个具体的例子：凑出 15 元。
- 为了凑出 15 元，我们最开始的时候，可以使用哪枚硬币？
  - 假设用了 1 元硬币，那么接下来要凑出 14 元。共 $1+4=5$ 枚

硬币问题

- 考虑一个具体的例子：凑出 15 元。
- 为了凑出 15 元，我们最开始的时候，可以使用哪枚硬币？
  - 假设用了 1 元硬币，那么接下来要凑出 14 元。共 $1+4=5$ 枚
  - 假设用了 5 元硬币，那么接下来要凑出 10 元。共 $1+2=3$ 枚

硬币问题

- 考虑一个具体的例子：凑出 15 元。
- 为了凑出 15 元，我们最开始的时候，可以使用哪枚硬币？
  - 假设用了 1 元硬币，那么接下来要凑出 14 元。共 $1+4=5$ 枚
  - 假设用了 5 元硬币，那么接下来要凑出 10 元。共 $1+2=3$ 枚
  - 假设用了 11 元硬币，那么接下来要凑出 4 元。共 $1+4=5$ 枚

硬币问题

- 考虑一个具体的例子：凑出 15 元。
- 为了凑出 15 元，我们最开始的时候，可以使用哪枚硬币？
  - 假设用了 1 元硬币，那么接下来要凑出 14 元。共 $1+4=5$ 枚
  - 假设用了 5 元硬币，那么接下来要凑出 10 元。共 $1+2=3$ 枚
  - 假设用了 11 元硬币，那么接下来要凑出 4 元。共 $1+4=5$ 枚
- 这三种方案，当然是选代价最低的，所以我们在这一次决策中，选择了 5 元硬币。

硬币问题

- 现在再来看，$f[i]$ 是“凑出 i 元需要的硬币数”，它等于什么？

硬币问题

- 现在再来看，$f[i]$ 是“凑出 i 元需要的硬币数”，它等于什么？
- 可供选择的决策方案如下：

硬币问题

- 现在再来看，$f[i]$ 是“凑出 i 元需要的硬币数”，它等于什么？
- 可供选择的决策方案如下：
  - 先用一个 1 元硬币，代价 $1 + f[i-1]$

硬币问题

- 现在再来看，$f[i]$ 是“凑出 i 元需要的硬币数”，它等于什么？
- 可供选择的决策方案如下：
  - 先用一个 1 元硬币，代价 $1 + f[i-1]$
  - 先用一个 5 元硬币，代价 $1 + f[i-5]$

硬币问题

- 现在再来看，$f[i]$ 是“凑出 i 元需要的硬币数”，它等于什么？
- 可供选择的决策方案如下：
  - 先用一个 1 元硬币，代价 $1 + f[i-1]$
  - 先用一个 5 元硬币，代价 $1 + f[i-5]$
  - 先用一个 11 元硬币，代价 $1 + f[i-11]$

硬币问题

- 现在再来看，$f[i]$ 是“凑出 i 元需要的硬币数”，它等于什么？
- 可供选择的决策方案如下：
  - 先用一个 1 元硬币，代价 $1 + f[i-1]$
  - 先用一个 5 元硬币，代价 $1 + f[i-5]$
  - 先用一个 11 元硬币，代价 $1 + f[i-11]$
- 在上述方案里面，选择代价最低的就行！

硬币问题

- 现在再来看，$f[i]$ 是“凑出 i 元需要的硬币数”，它等于什么？
- 可供选择的决策方案如下：
  - 先用一个 1 元硬币，代价 $1 + f[i-1]$
  - 先用一个 5 元硬币，代价 $1 + f[i-5]$
  - 先用一个 11 元硬币，代价 $1 + f[i-11]$
- 在上述方案里面，选择代价最低的就行！
- 所以有

$$f[i] = \min \begin{cases} 1 + f[i-1] \\ 1 + f[i-5] \\ 1 + f[i-11] \end{cases}$$

目录

初步总结：状态

- 我们用“大事化小，小事化了”的思想，解决了上楼梯问题和硬币问题。

初步总结：状态

- 我们用“大事化小，小事化了”的思想，解决了上楼梯问题和硬币问题。
- 大事能转化成小事，是因为大事和小事都有一样的形式：

初步总结：状态

- 我们用“大事化小，小事化了”的思想，解决了上楼梯问题和硬币问题。
- 大事能转化成小事，是因为大事和小事都有一样的形式：
- 上楼梯问题

初步总结：状态

- 我们用“大事化小，小事化了”的思想，解决了上楼梯问题和硬币问题。
- 大事能转化成小事，是因为大事和小事都有一样的形式：
- 上楼梯问题
  - 大问题：爬上 n 级有多少种方案

初步总结：状态

- 我们用“大事化小，小事化了”的思想，解决了上楼梯问题和硬币问题。
- 大事能转化成小事，是因为大事和小事都有一样的形式：
- 上楼梯问题
  - 大问题：爬上 n 级有多少种方案
  - 小问题：爬上 $n-1$ 级有多少种方案、爬上 $n-2$ 级有多少种方案

初步总结：状态

- 我们用“大事化小，小事化了”的思想，解决了上楼梯问题和硬币问题。
- 大事能转化成小事，是因为大事和小事都有一样的形式：
- 上楼梯问题
  - 大问题：爬上 n 级有多少种方案
  - 小问题：爬上 $n-1$ 级有多少种方案、爬上 $n-2$ 级有多少种方案
  - 它们都是**爬上 $\times\times$ 级有多少种方案**这一类问题。

初步总结：状态

- 我们用“大事化小，小事化了”的思想，解决了上楼梯问题和硬币问题。
- 大事能转化成小事，是因为大事和小事都有一样的形式：
- 上楼梯问题
  - 大问题：爬上 n 级有多少种方案
  - 小问题：爬上 $n-1$ 级有多少种方案、爬上 $n-2$ 级有多少种方案
  - 它们都是**爬上 $\times\times$ 级有多少种方案**这一类问题。
- 硬币问题

初步总结：状态

- 我们用“大事化小，小事化了”的思想，解决了上楼梯问题和硬币问题。
- 大事能转化成小事，是因为大事和小事都有一样的形式：
- 上楼梯问题
  - 大问题：爬上 n 级有多少种方案
  - 小问题：爬上 $n-1$ 级有多少种方案、爬上 $n-2$ 级有多少种方案
  - 它们都是**爬上 $\times\times$ 级有多少种方案**这一类问题。
- 硬币问题
  - 大问题：凑出 n 元钱的最少硬币数

初步总结：状态

- 我们用“大事化小，小事化了”的思想，解决了上楼梯问题和硬币问题。
- 大事能转化成小事，是因为大事和小事都有一样的形式：
- 上楼梯问题
  - 大问题：爬上 n 级有多少种方案
  - 小问题：爬上 $n-1$ 级有多少种方案、爬上 $n-2$ 级有多少种方案
  - 它们都是**爬上 $\times\times$ 级有多少种方案**这一类问题。
- 硬币问题
  - 大问题：凑出 n 元钱的最少硬币数
  - 小问题：凑出 $n-1, n-5, n-11$ 元的最少硬币数

初步总结：状态

- 我们用“大事化小，小事化了”的思想，解决了上楼梯问题和硬币问题。
- 大事能转化成小事，是因为大事和小事都有一样的形式：
- 上楼梯问题
  - 大问题：爬上 n 级有多少种方案
  - 小问题：爬上 $n-1$ 级有多少种方案、爬上 $n-2$ 级有多少种方案
  - 它们都是**爬上 $\times\times$ 级有多少种方案**这一类问题。
- 硬币问题
  - 大问题：凑出 n 元钱的最少硬币数
  - 小问题：凑出 $n-1, n-5, n-11$ 元的最少硬币数
  - 它们都是**凑出 $\times\times$ 元钱所需最少硬币数**这一类问题。

初步总结：状态

- 可见，只有大问题和小问题拥有相同的形式，才能考虑大事化小。

初步总结：状态

- 可见，只有大问题和小问题拥有相同的形式，才能考虑大事化小。
- 如果满足这个要求，那么我们遇到的每个问题，都可以很简洁地表达。

初步总结：状态

- 可见，只有大问题和小问题拥有相同的形式，才能考虑大事化小。
- 如果满足这个要求，那么我们遇到的每个问题，都可以很简洁地表达。
- 我们把可能遇到的每种局面称为状态。

初步总结：状态

- 可见，只有大问题和小问题拥有相同的形式，才能考虑大事化小。
- 如果满足这个要求，那么我们遇到的每个问题，都可以很简洁地表达。
- 我们把可能遇到的每种局面称为**状态**。
- 硬币问题中，要表达“我们需要凑出 n 元钱”这个局面，可以设计状态：“$f[i]$ 表示凑出 i 元用的最少硬币数”。

初步总结：状态

- 可见，只有大问题和小问题拥有相同的形式，才能考虑大事化小。
- 如果满足这个要求，那么我们遇到的每个问题，都可以很简洁地表达。
- 我们把可能遇到的每种局面称为**状态**。
- 硬币问题中，要表达“我们需要凑出 n 元钱”这个局面，可以设计状态：“$f[i]$ 表示凑出 i 元用的最少硬币数”。
- 上楼梯问题中，设计状态：“$f[i]$ 表示走上 i 级的方案数”。

初步总结：状态

- 可见，只有大问题和小问题拥有相同的形式，才能考虑大事化小。
- 如果满足这个要求，那么我们遇到的每个问题，都可以很简洁地表达。
- 我们把可能遇到的每种局面称为**状态**。
- 硬币问题中，要表达“我们需要凑出 n 元钱”这个局面，可以设计状态：“$f[i]$ 表示凑出 i 元用的最少硬币数”。
- 上楼梯问题中，设计状态：“$f[i]$ 表示走上 i 级的方案数”。
- 设计完状态之后，只要能利用**小状态的解求出大状态的解**，就可以动手把题目做出来！

目录

最长上升子序列问题

- 一个数的序列 B，当 $b_1 < b_2 < ... < b_{|B|}$ 的时候，我们称这个序列是上升的。

最长上升子序列问题

- 一个数的序列 B，当 $b_1 < b_2 < ... < b_{|B|}$ 的时候，我们称这个序列是上升的。
- 对于给定的一个序列 $a_1, a_2, ..., a_N$，我们可以得到一些上升的子序列 $a_{i_1}, a_{i_2}, ..., a_{i_K}$，其中 $1 \leq i_1 < i_2 < \cdots < i_K \leq N$。

最长上升子序列问题

- 一个数的序列 B，当 $b_1 < b_2 < ... < b_{|B|}$ 的时候，我们称这个序列是上升的。
- 对于给定的一个序列 $a_1, a_2, ..., a_N$，我们可以得到一些**上升的子序列** $a_{i_1}, a_{i_2}, ..., a_{i_K}$，其中 $1 \leq i_1 < i_2 < \cdots < i_K \leq N$。
- 比如，对于序列 $[1, 7, 3, 5, 9, 4, 8]$，它的上升子序列就有 $[1, 7]$、$[3, 4, 8]$、$[1, 3, 5, 8]$ 等等。这些上升子序列中最长的长度是 4。

最长上升子序列问题

- 一个数的序列 B，当 $b_1 < b_2 < ... < b_{|B|}$ 的时候，我们称这个序列是上升的。
- 对于给定的一个序列 $a_1, a_2, ..., a_N$，我们可以得到一些**上升的子序列** $a_{i_1}, a_{i_2}, ..., a_{i_K}$，其中 $1 \leq i_1 < i_2 < \cdots < i_K \leq N$。
- 比如，对于序列 $[1, 7, 3, 5, 9, 4, 8]$，它的上升子序列就有 $[1, 7]$、$[3, 4, 8]$、$[1, 3, 5, 8]$ 等等。这些上升子序列中最长的长度是 4。
- 你的任务，就是对于给定的序列，求出最长上升子序列的长度。

最长上升子序列问题

- 想用大事化小来做这道题，必须先设计状态。

最长上升子序列问题

- 想用大事化小来做这道题，必须先设计状态。
- 如何设计状态，来完整地描述当前遇到的局面？

最长上升子序列问题

- 想用大事化小来做这道题，必须先设计状态。
- 如何设计状态，来完整地描述当前遇到的局面？
- 设计状态：$f[i]$ 表示以 $a[i]$ 结尾的最长上升子序列的长度

最长上升子序列问题

- 想用大事化小来做这道题，必须先设计状态。
- 如何设计状态，来完整地描述当前遇到的局面？
- 设计状态：$f[i]$ 表示以 $a[i]$ 结尾的最长上升子序列的长度
- 那么，答案就是 $f[1], f[2] \ldots f[n]$ 里面的最大值。

最长上升子序列问题

- 想用大事化小来做这道题，必须先设计状态。
- 如何设计状态，来完整地描述当前遇到的局面？
- 设计状态：$f[i]$ 表示以 $a[i]$ 结尾的最长上升子序列的长度
- 那么，答案就是 $f[1], f[2] \ldots f[n]$ 里面的最大值。
- 问题来了，如何求出 f 数组？提示：思考 $f[i]$ 从哪里来。

最长上升子序列问题

- $f[i]$ 表达的是“以 $a[i]$ 结尾的最长上升子序列长度”。这个最长的子序列，一定是把 $a[i]$ 接在某个上升子序列尾部形成的！

最长上升子序列问题

- $f[i]$ 表达的是“以 $a[i]$ 结尾的最长上升子序列长度”。这个最长的子序列，一定是把 $a[i]$ 接在某个上升子序列尾部形成的!
- 序列 $[1, 7, 3, 5, \mathbf{9}, 4, 8]$

最长上升子序列问题

- $f[i]$ 表达的是“以 $a[i]$ 结尾的最长上升子序列长度”。这个最长的子序列，一定是把 $a[i]$ 接在某个上升子序列尾部形成的!
- 序列 $[1, 7, 3, 5, \mathbf{9}, 4, 8]$
- 考虑 f[5]，它的来源有:

最长上升子序列问题

- $f[i]$ 表达的是“以 $a[i]$ 结尾的最长上升子序列长度”。这个最长的子序列，一定是把 $a[i]$ 接在某个上升子序列尾部形成的!
- 序列 $[1, 7, 3, 5, \mathbf{9}, 4, 8]$
- 考虑 f[5]，它的来源有:
  - 自己一个元素作为一个序列。长度为 1 .

最长上升子序列问题

- $f[i]$ 表达的是“以 $a[i]$ 结尾的最长上升子序列长度”。这个最长的子序列，一定是把 $a[i]$ 接在某个上升子序列尾部形成的!
- 序列 $[1, 7, 3, 5, \mathbf{9}, 4, 8]$
- 考虑 f[5]，它的来源有:
  - 自己一个元素作为一个序列。长度为 1 .
  - 接在 a[1] 后面。长度为 $\mathrm{f}[1] + 1 = 2$

最长上升子序列问题

- $f[i]$ 表达的是“以 $a[i]$ 结尾的最长上升子序列长度”。这个最长的子序列，一定是把 $a[i]$ 接在某个上升子序列尾部形成的!
- 序列 $[1, 7, 3, 5, \mathbf{9}, 4, 8]$
- 考虑 f[5]，它的来源有:
  - 自己一个元素作为一个序列。长度为 1 .
  - 接在 a[1] 后面。长度为 $\mathrm{f}[1] + 1 = 2$
  - 接在 a[2] 后面。长度为 $\mathrm{f}[2] + 1 = 3$

最长上升子序列问题

- $f[i]$ 表达的是“以 $a[i]$ 结尾的最长上升子序列长度”。这个最长的子序列，一定是把 $a[i]$ 接在某个上升子序列尾部形成的!
- 序列 $[1, 7, 3, 5, \mathbf{9}, 4, 8]$
- 考虑 f[5]，它的来源有:
  - 自己一个元素作为一个序列。长度为 1 .
  - 接在 a[1] 后面。长度为 $\mathrm{f}[1] + 1 = 2$
  - 接在 a[2] 后面。长度为 $\mathrm{f}[2] + 1 = 3$
  - 接在 a[3] 后面。长度为 $\mathrm{f}[3] + 1 = 3$

最长上升子序列问题

- $f[i]$ 表达的是“以 $a[i]$ 结尾的最长上升子序列长度”。这个最长的子序列，一定是把 $a[i]$ 接在某个上升子序列尾部形成的!
- 序列 $[1, 7, 3, 5, \mathbf{9}, 4, 8]$
- 考虑 f[5]，它的来源有:
  - 自己一个元素作为一个序列。长度为 1 .
  - 接在 a[1] 后面。长度为 $\mathrm{f}[1] + 1 = 2$
  - 接在 a[2] 后面。长度为 $\mathrm{f}[2] + 1 = 3$
  - 接在 a[3] 后面。长度为 $\mathrm{f}[3] + 1 = 3$
  - 接在 a[4] 后面。长度为 $\mathrm{f}[4] + 1 = 4$

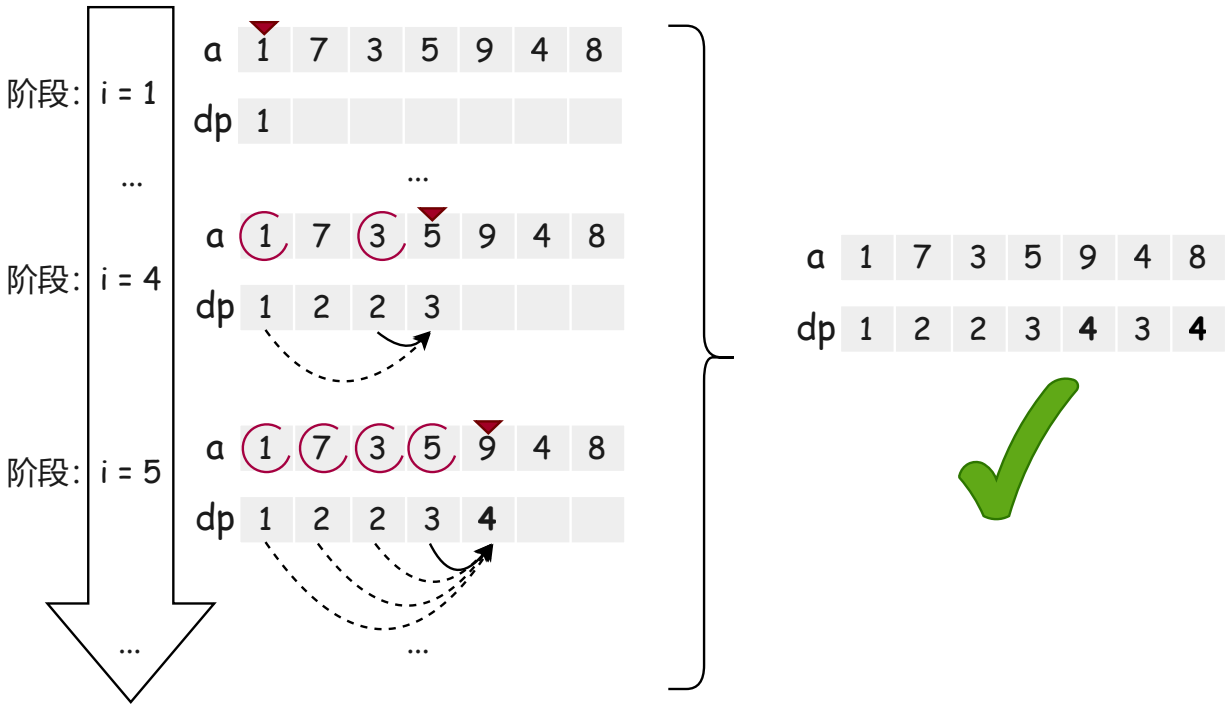
最长上升子序列问题

- 要得到 $f[i]$，只需要看 $a[i]$ 能接在哪些数的后面。也就是：

$$f[i] = \max_{j<i, a[j]<a[i]} \{f[j] + 1\}$$

- 要得到 $f[i]$，只需要看 $a[i]$ 能接在哪些数的后面。也就是：

$$f[i] = \max_{j<i, a[j]<a[i]} \{f[j] + 1\}$$



目录

初步总结：转移

- 在前面三个例题中，我们都是先设计好状态，然后给出了一套用小状态推出大状态解的方法。

初步总结：转移

- 在前面三个例题中，我们都是先设计好状态，然后给出了一套用小状态推出大状态解的方法。
- 从一个状态的解，得知另一个状态的解，我们称之为 状态转移。

初步总结：转移

- 在前面三个例题中，我们都是先设计好状态，然后给出了一套用小状态推出大状态解的方法。
- 从一个状态的解，得知另一个状态的解，我们称之为 **状态转移**。
- 这个转移式子称为 **状态转移方程**。

初步总结：转移

- 在前面三个例题中，我们都是先设计好状态，然后给出了一套用小状态推出大状态解的方法。
- 从一个状态的解，得知另一个状态的解，我们称之为 **状态转移**。
- 这个转移式子称为 **状态转移方程**。
- 硬币问题中，状态转移方程是：

$$f[i] = 1 + \min\{f[i-1], f[i-5], f[i-11]\}$$

初步总结：转移

- 在前面三个例题中，我们都是先设计好状态，然后给出了一套用小状态推出大状态解的方法。
- 从一个状态的解，得知另一个状态的解，我们称之为 **状态转移**。
- 这个转移式子称为 **状态转移方程**。
- 硬币问题中，状态转移方程是：

$$f[i] = 1 + \min\{f[i-1], f[i-5], f[i-11]\}$$

- LIS 问题中，状态转移方程是：

$$f[i] = \max_{j<i, a[j]<a[i]} \{f[j] + 1\}$$

目录

状态和转移

总结刚刚学习的内容。如果我们想用大事化小的思想解决一个问题，我们需要:

状态和转移

总结刚刚学习的内容。如果我们想用大事化小的思想解决一个问题，我们需要:

1. 设计状态。把面临的每一个问题，用状态表达出来。

状态和转移

总结刚刚学习的内容。如果我们想用大事化小的思想解决一个问题，我们需要:

1. 设计状态。把面临的每一个问题，用状态表达出来。
2. 设计转移。写出状态转移方程，从而利用小问题的解推出大问题的解。

状态转移

- 前三个问题中，我们设计转移的时候，考虑的都是“这个局面是从哪过来的"。这是一种常见的思路：当前状态的解未知。需要用已经解决的状态，来推出当前状态的解。

状态转移

- 前三个问题中，我们设计转移的时候，考虑的都是“这个局面是从哪过来的"。这是一种常见的思路：当前状态的解未知。需要用已经解决的状态，来推出当前状态的解。
- DP 还有另一种设计转移的思路：当前状态的解已知。需要利用这个解，去更新它能走到的状态。

状态转移

- 前三个问题中，我们设计转移的时候，考虑的都是“这个局面是从哪过来的"。这是一种常见的思路：当前状态的解未知。需要用已经解决的状态，来推出当前状态的解。
- DP 还有另一种设计转移的思路：当前状态的解已知。需要利用这个解，去更新它能走到的状态。
- 这两种思路，一种是考虑“我从哪里来”，一种是考虑“我到哪里去”。

状态转移

- 如何用“我到哪里去”的转移手段，解决上楼梯问题？

状态转移

- 如何用“我到哪里去”的转移手段，解决上楼梯问题？

$$\begin{aligned} f[i] &\to f[i+1] \\ &\to f[i+2] \end{aligned}$$

状态转移

- 如何用“我到哪里去”的转移手段，解决上楼梯问题？

$$\begin{aligned} f[i] &\to f[i+1] \\ &\to f[i+2] \end{aligned}$$

- 如何用“我到哪里去”的转移手段，解决硬币问题？

状态转移

- 如何用“我到哪里去”的转移手段，解决上楼梯问题？

$$\begin{aligned} f[i] &\to f[i+1] \\ &\to f[i+2] \end{aligned}$$

- 如何用“我到哪里去”的转移手段，解决硬币问题？

$$\begin{aligned} f[i] &\to f[i+1] \\ &\to f[i+5] \\ &\to f[i+11] \end{aligned}$$

状态转移

- 如何用“我到哪里去”的转移手段，解决上楼梯问题？

$$\begin{aligned} f[i] &\to f[i+1] \\ &\to f[i+2] \end{aligned}$$

- 如何用“我到哪里去”的转移手段，解决硬币问题？

$$\begin{aligned} f[i] &\to f[i+1] \\ &\to f[i+5] \\ &\to f[i+11] \end{aligned}$$

- 如何用“我到哪里去”的转移手段，解决 LIS 问题？

状态转移

- 如何用“我到哪里去”的转移手段，解决上楼梯问题？

$$f[i] \to f[i+1]$$
$$\to f[i+2]$$

- 如何用“我到哪里去”的转移手段，解决硬币问题？

$$f[i] \to f[i+1]$$
$$\to f[i+5]$$
$$\to f[i+11]$$

- 如何用“我到哪里去”的转移手段，解决 LIS 问题？

$$f[i] \to f[j]$$

其中 $j > i, a[j] > a[i]$。

状态转移

状态转移有两种方法。

状态转移

状态转移有两种方法。

- **填表法**：对于一个没有求出解的状态，利用能走到它的状态，来得出它的解。(我从哪里来)

状态转移

状态转移有两种方法。

- **填表法**：对于一个没有求出解的状态，利用能走到它的状态，来得出它的解。(我从哪里来)
- **刷表法**：对于一个已经求好了解的状态，拿去更新它能走到的状态。(我到哪里去)

目录

目录

- 序列型 **DP**的状态是基础中的基础，大部分的动态规划都要用到它，成为一个维。

序列型 DP

- 序列型 **DP**的状态是基础中的基础，大部分的动态规划都要用到它，成为一个维。
- 一般来说，有两种状态的定义：

- 序列型 **DP**的状态是基础中的基础，大部分的动态规划都要用到它，成为一个维。
- 一般来说，有两种状态的定义：
  1. **f[i]** 表示前 i 个元素决策组成的一个状态。

- 序列型 **DP**的状态是基础中的基础，大部分的动态规划都要用到它，成为一个维。
- 一般来说，有两种状态的定义：
  1. **f[i]** 表示前 i 个元素决策组成的一个状态。
  2. **f[i]** 表示用到了第 i 个元素，和其他在 1 到 $i-1$ 间的元素，决策组成的一个状态。

序列型 DP 练习

1. P478 上楼梯问题

序列型 DP 练习

1. P478 上楼梯问题
2. P483 最长上升子序列

序列型 DP 练习

1. P478 上楼梯问题
2. P483 最长上升子序列
3. P471 最长不上升子序列

序列型 DP 练习

1. P478 上楼梯问题
2. P483 最长上升子序列
3. P471 最长不上升子序列
4. P484 合唱队形

序列型 DP 练习

1. P478 上楼梯问题
2. P483 最长上升子序列
3. P471 最长不上升子序列
4. P484 合唱队形
5. P480 排队买票

序列型 DP 练习

1. P478 上楼梯问题
2. P483 最长上升子序列
3. P471 最长不上升子序列
4. P484 合唱队形
5. P480 排队买票
6. P479 小猪过河

序列型 DP 练习

1. P478 上楼梯问题
2. P483 最长上升子序列
3. P471 最长不上升子序列
4. P484 合唱队形
5. P480 排队买票
6. P479 小猪过河
7. P485 最大子段和

目录

- **划分型 DP**用于解决类似序列如何分组的最优问题。

划分型 DP

- **划分型 DP**用于解决类似序列如何分组的最优问题。
- 一般来说，状态的定义：$f[i][j]$ 表示前 i 个元素，分为 j 组的最优状态。

划分型 DP

- **划分型 DP**用于解决类似序列如何分组的最优问题。
- 一般来说，状态的定义：$f[i][j]$ 表示前 i 个元素，分为 j 组的最优状态。
- 状态转移一般枚举最后一组的划分方式进行转移。

划分型 DP

- **划分型 DP**用于解决类似序列如何分组的最优问题。
- 一般来说，状态的定义：$f[i][j]$ 表示前 i 个元素，分为 j 组的最优状态。
- 状态转移一般枚举最后一组的划分方式进行转移。
- 当分组的数量没有限制时，状态可定义为 $f[i]$ 表示前 i 个元素进行分组的最优状态。

划分型 DP 练习

1. P477 乘积最大

划分型 DP 练习

1. P477 乘积最大
2. P481 维修栅栏

划分型 DP 练习

1. P477 乘积最大
2. P481 维修栅栏
3. P482 工作安排

目录

- **棋盘型 DP**用于解决类似在二维棋盘（迷宫）上行走的最优问题。

棋盘型 DP

- **棋盘型 DP**用于解决类似在二维棋盘（迷宫）上行走的最优问题。
- 一般来说，状态的定义：$f[i][j]$ 表示走到 (i,j) 的最优状态。

棋盘型 DP

- **棋盘型 DP**用于解决类似在二维棋盘（迷宫）上行走的最优问题。
- 一般来说，状态的定义：$f[i][j]$ 表示走到 (i,j) 的最优状态。
- 状态的转移一般找哪个位置可以一步走到 (i,j)。

棋盘型 DP 练习

1. P473 数字三角形

棋盘型 DP 练习

1. P473 数字三角形
2. P472 简单传纸条

棋盘型 DP 练习

1. P473 数字三角形
2. P472 简单传纸条
3. P476 过河卒

目录

记忆化搜索

- **记忆化搜索**是一种通过记录已经遍历过的状态的信息，从而避免对同一状态重复遍历的搜索实现方式。

记忆化搜索

- **记忆化搜索**是一种通过记录已经遍历过的状态的信息，从而避免对同一状态重复遍历的搜索实现方式。
- 因为记忆化搜索确保了每个状态只访问一次，它也是一种常见的动态规划实现方式。

记忆化搜索

- **记忆化搜索**是一种通过记录已经遍历过的状态的信息，从而避免对同一状态重复遍历的搜索实现方式。
- 因为记忆化搜索确保了每个状态只访问一次，它也是一种常见的动态规划实现方式。
- 记忆化搜索中的记忆化数组其实就是动态规划中状态的定义。

记忆化搜索练习

1. P381 滑雪

记忆化搜索练习

1. P381 滑雪
2. P474 递归函数

记忆化搜索练习

1. P381 滑雪
2. P474 递归函数
3. P475 雷曼兔