动态规划初步

林东颖

2025 年 2 月 13 日

1 引言

2 引例

3 理论基础

学习指南

- 本专题题目需要同学掌握线性 dp、背包 dp 对应了 $oiwiki$ 里的动态规划基础和背包 dp(习题里的背包 dp 应该会放比较基础的
- https://oi-wiki.org/dp/
- 点击上方链接即可开始学习
- 书写这一份 pdf 对 $oiwiki$ 上动态规划基础部分做了更具体的讲解，以便于同学们更好的学习
- 因为 $oiwiki$ 是一个开源项目，每个书写者所针对的受众并不相同，对于各人来说不一定适配。但把 $oiwiki$ 当做一份知识点大纲当然是合格的，针对特定的一个知识点的学习，同学们可以上网查找更对自己胃口的资料
- 学习过程中有任何疑问可以发到 qq 群共同讨论
- 掌握课程要求的内容之后，学有余力的同学可以继续学习后续知识，acm 协同创新团队欢迎你

洛谷 P1216

题面 (数字三角形)

给定一个 n 行的数字三角形（$n < 1000$），需要找到一条从最高点到底部的路径，使该路径经过数字之和最大。每一步可以走到当前点的左下方或右下方的点。

下面是一个示例数字三角形：

```
        7
      3   8
    8   1   0
  2   7   4   4
4   5   2   6   5
```

在上面的数字三角形中，最优路径的示例是：$7 \to 3 \to 8 \to 7 \to 5$

解法

- 设 $f(i,j)$ 表示从最高点走到第 i 行第 j 列的最大路径和
- 则有： $f(i,j) = max(f(i-1,j-1), f(i-1,j))$
- 最终答案为： $max(f(n,1), f(n,2), \ldots, f(n,n))$

1 引言

2 引例

3 理论基础

几个概念：阶段、状态、决策

- 可以看出我们将整个问题划分为了若干个**阶段**来做 (本题的阶段根据行号来划分
- 每个阶段对应若干个子问题，提取这些子问题的特征 (称之为**状态**
- 所以 $f(i,j)$ 就称做是一个状态。
- 寻找每个状态可能的**决策**, 或者说是各状态间的相互转移方式
- 不妨设 $a(i,j)$ 表示第 i 行第 j 列的数字
- 那么本题中，在 (i,j) 这个位置时可以做出向左下走的决策得到 $f(i,j)+a(i+1,j)$，也可以做出向右下走的决策得到 $f(i,j)+a(i+1,j+1)$
- 只要按照顺序求解每一个阶段的问题即可

几个概念：最优子结构、无后效性、子问题重叠

- 最优子结构：如果一个问题的最优解包含了子问题的最优解，那么称该问题具有最优子结构
- 无后效性：当前状态一旦确定，后续的决策只依赖于当前状态，与如何到达当前状态无关。(就像下棋时，你只需要关心当前棋盘状态，不需要记住之前的每一步怎么走的
- 对于一个问题只有当它满足了上述两个条件，才有办法采用动态规划解决它
- 子问题重叠：在递归分解问题时，大量子问题会被重复计算。动态规划通过“记忆化”避免重复计算
- 例如引例中 $f(i,j)$ 用到了 $f(i-1,j+1)$ 的值，$f(i,j+1)$ 也用到了 $f(i-1,j+1)$ 的值，我们不需要求解两次 $f(i-1,j+1)$ 而是求完之后将值存在数组中
- 动态规划的核心思想正是减少冗余运算

Thanks!