动态规划解决鸡蛋下落问题

原创 冯晓妍 LOA算法学习笔记 2021-02-15 20:36

01 从分而治之到动态规划

分而治之方法是将一个问题分解为若干个相互独立的子问题，然后用递归的方式求解子问题，最后将子问题的解结合起来从而得到原问题的解。与该方法不同的是，动态规划方法是将一个问题分解为若干个彼此非独立的子问题，亦即，几个子问题之间分享若干个子问题，也称重叠子问题。应用动态规划方法求解一个优化问题通常包括如下几个步骤：

（1）刻画最优解的结构

（2）给出最优值的递归表达式

（3）计算最优值

（4）根据所得最优值，确定最优解

可以看出，由于采用分而治之方法设计的算法重复求解了相同的子问题，所以其所作的一些运算是不必要的。而采用动态规划方法设计的算法求解每一个子问题仅一次，然后将所得解存贮在一个表中，这样就避免了遇到同一个子问题的时候，需要再花时间求出它的解。

02 鸡蛋下落问题

我们在算法分析中学习动态规划，除了勤加思考原理，更应该多多解决具体的问题。下面与大家分享一个经典的动态规划问题——鸡蛋下落问题：

给定eggs个鸡蛋和一栋从1到floors层高的建筑。假设存在楼层F，满足0≤F≤floors，任何从高于F的楼层落下的鸡蛋都会碎，从F楼层或比它低的楼层落下的鸡蛋都不会破。每次实验，如果你还有完整的鸡蛋，你可以把一个鸡蛋从任意楼层k（1<k≤floors）扔下，每个蛋就像现实中的鸡蛋一样，只能碎一次，但如果鸡蛋是完整的，则可以拿来重复进行实验。

首先分而治之并不适用于这个问题，因为如果根据二分法来选择实验楼层，无法保证有足够的鸡蛋来做实验。接下来根据动态规划方法求解优化问题的一般步骤来尝试解决这个问题。

1. 刻画最优解的结构

假设用**dp[i][j]**（0<i≤eggs,0<j≤floors）表示**子问题**：用i个鸡蛋在一栋j层高的楼寻找目标层更需要的最少实验次数，则 **dp[eggs][floors]**记录了原始问题，eggs个鸡蛋在一栋floors层高的楼寻找目标层更需要的最少实验次数，的答案。

(1)当**鸡蛋个数i大于楼层数j时**

即使减少一个鸡蛋，最优解仍然不变，因此可以把问题转化成楼层数相同、鸡蛋个数减一的子问题dp[i-1][j]。

(2) 当**鸡蛋个数i小于楼层数j时**

假设我们用一个鸡蛋在第层做实验，存在两种可能，**鸡蛋碎了**或者**鸡蛋没碎**。

鸡蛋碎了：j层可能是目标层，但是我们还需要在k-1层做实验来验证。此时dp[i][j]=dp[i-1][j-1]。

鸡蛋没碎：j层必然不是目标层，此时我们还需要经过至少dp[i][j-k]次实验才能找到目标层。举例来说，比如求用3个鸡蛋在7层高的楼做实验需要的最少次数，在第4层进行了实验，鸡蛋没碎，那么第1、2、3、4层都不需要再进行实验，问题就转化为求用3个鸡蛋在3层高的楼做实验需要的最少次数dp[3][3]+1。

由于存在 鸡蛋碎了 和 鸡蛋没碎 两种可能情况，我们需要取两种情况中的最大值才能保证有足够的次数完成所有实验。

我们在上面只讨论出了在第k层进行一次实验后还需要的最少实验次数，但是在第k层做实验并不一定会使总实验次数最少，因此我们还需要遍历0<k≤j，来寻找对于j层高的楼使总实验次数最少的中间层K。

2. 给出最优值的递归表达式

由上述最优解的结构可以得到下面的式子：

$$dp[1][j] = j$$
$$dp[0][0] = 0$$
$$dp[i][j] = 1 + \min_{1 \le k \le j}\{max\ (dp[i-1][k-1], dp[i][j-k])\}$$

由此可以归纳得出这么一个信息：如果要求出dp[i][j]，那么一定先求出它的2j子问题，进行一次**决策**：选择一个楼层K进行一次实验，使得总的实验次数最少，即当前考查的子问题中，应该在第K层进行第一次实验。

把dp[i][j]称为问题的**状态**，而把上面的式子称作**状态转移方程**，它把状态dp[i][j]转移为

$$1 + \min_{1 \le k \le j}\{max\ (dp[i-1][k-1], dp[i][j-k])\}$$

可以发现，状态dp[i][j]只与二维数组/动态规划表中左上方的状态有关，即从dp[1][ ]和dp[ ][0]组成的边界出发，递推地求得dp[eggs][floors]。

3. 计算最优值

解决该问题的主要代码如下

```
if (eggs > floors) {
    table[eggs][floors] = table[eggs - 1][floor];
} else {
    let min = floors;
    for (let floor = 1; f <= floors; floor += 1) {
        const max = Math.max(
            table[eggs - 1][floor - 1], // egg breaks
            table[eggs][floors - floor] // egg didn't break
        );
        min = Math.min(min, max);
    }
    table[eggs][floors] = 1 + min;
}
```

4. 计算最优值根据所得最优值，确定最优解

最优解是鸡蛋下落实验的具体步骤，由于每一次状态转移都需要一个最优的实验楼层K，我们可以再用一个二维数组opt[i][j]记录每一次状态转移最优的K，然后取第一个K=opt[eggs][floors]，然后继续往前查看opt[eggs-1][K-1],opt[eggs-1][floors-K]的值，但是由于存在鸡蛋碎了或者没碎两种状态，所以最优步骤需要分类讨论，欢迎感兴趣的同学们自行尝试。

尼玛侃轻武