动态规划 (Dynamic Programming, DP)

Tan Yiqing

2025 年 11 月 4 日

1 特点

动态规划作为算法的重要基础之一，它代表一类算法，其主要特点包括：

- 问题可以分解为较小规划的同类问题。
- 子问题多次重复出现。

它的核心概念是将大问题分成多个小问题，然后解决各个小问题，每个小问题只解决一次。

2 实现方法

动态规划通常有两种实现方法：

- 从上到下：和递归类似，不同的是存储中间计算的结果，避免多次解决同一个小问题。
- 从下到上：把有关的子问题一个一个解决，最后计算出要的结果。

3 解题思路

动态规划的解题思路一般包括以下几个步骤：

- 确定 dp 数组 (dp table) 及下标的含义。(这里的 dp 就是子问题的结果)
- 确定递推公式。
- 初始化 dp 数组。
- 确定遍历顺序。
- 举例推导 dp 数组。

4 debug 方法

找问题的最好方式就是把 dp 数组打印出来，看一下究竟是不是按照自己的思路推导的。
做动态规划的题目，写代码之前一定要把状态转移在 dp 数组中的具体情况模拟一遍做到心中有数，确定最后推导出的结果是什么。然后编写相应的代码，如果代码没通过就打印 dp 数组，看看和自己预先推导的哪里不一样。如果打印结果和自己预先模拟推导的结果是一样的，那么就是自己的递推公式、dp 数组的初始化或者遍历顺序有问题了。如果打印结果和自己预先模拟推导的结果不一样，那么就是代码的实现细节有问题。

5 例子

5.1 斐波那契数列

很简单，主要是用于实践五个步骤。

1. 确定 dp 数组及下标的含义：dp[i] 表示第 i 个斐波那契数。
2. 确定递推公式：dp[i] = dp[i-1] + dp[i-2]
3. 初始化 dp 数组：dp[0] = 0, dp[1] = 1
4. 确定遍历顺序：从小到大遍历。
5. 举例推导 dp 数组：写出前十项：[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]

C++ 代码实现：

```
int fib(int n) {
    if (n <= 1) return n;
    vector<int> dp(n + 1);
    dp[0] = 0;
    dp[1] = 1;
    for (int i = 2; i <= n; ++i) {
        dp[i] = dp[i - 1] + dp[i - 2];
    }
    return dp[n];
}
```

事实上我们没必要记录整个 dp 数组，只需要记录前两个数即可，空间复杂度可以优化到 O(1)。

```
int fib(int n) {
    if (n <= 1) return n;
    int a = 0, b = 1;
    for (int i = 2; i <= n; ++i) {
        int c = a + b;
        a = b;
        b = c;      //滚动数组,只维护两个数字，不存储整个数组。
    }
    return b;
}
```

当然这个也可以用从

5.2 最长递增子序列

很暴力的算法就是找出所有的子序列，再看哪些递增，再看哪个最长，时间复杂度是指数级别的。动态规划的思路是：

5.3 找零问题