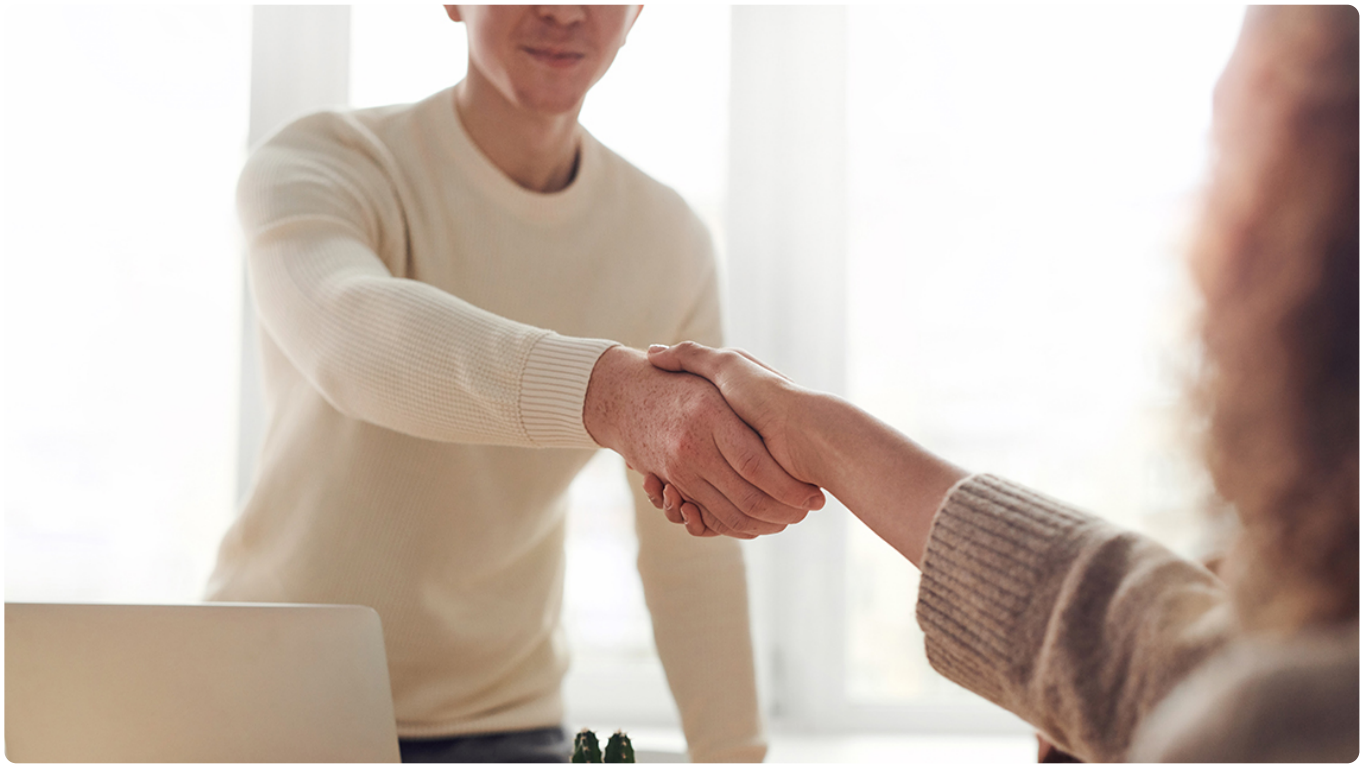
05｜面试即正义第一期：什么样的问题应该使用动态规划？

2020-09-23 卢誉声

动态规划面试宝典 进入课程 >



讲述：卢誉声
时长 15:27 大小 14.16M

你好，我是卢誉声。

作为“初识动态规划”模块的最后一课，今天我们不谈具体的解决方案了，我们来聊聊面试相关的话题，做个总结，也为我们后面的深入学习打下一个良好的基础。

那说起动态规划，我不知道你有没有这样的困扰，在掌握了一些基础算法和数据结构之后，碰到一些较为复杂的问题还是无从下手，面试时自然也是胆战心惊。如果我说动态规划是个玄幻的问题其实也不为过。究其原因，我觉得可以归因于这样两点：

- 你对动态规划相关问题的套路和思想还没有完全掌握；
- 你没有系统地总结过究竟有哪些问题可以用动态规划解决。

知己知彼，你想把动态规划作为你的面试武器之一，就得足够了解它；而应对面试，总结、归类问题其实是个不错的选择，这在我们刷题的时候其实也能感觉得到。

那么今天，我们就针对以上两点，系统地谈一谈究竟什么样的问题可以用动态规划来解。相信这节课过后，你就能有针对性地攻克难关了，无论是面试还是工程实践都能做到有的放矢。

动态规划是一种思想

动态规划算法，这种叫法我想你应该经常听说。嗯，从道理上讲这么说我觉得也没错，首先动态规划它不是数据结构，这一点毋庸置疑，并且严格意义上来说它就是一种算法。但更加准确或者更加贴切的提法应该是说动态规划是一种思想。

那什么是思想？算法和思想又有什么区别呢？

一般来说，我们都会把算法和数据结构放一起来讲，这是因为它们之间密切相关，而算法也往往是在特定数据结构的基础之上对解题方案的一种严谨的总结。

比如说，在一个乱序数组的基础上进行排序，这里的数据结构指的是什么呢？很显然是数组，而算法则是所谓的排序。至于排序算法，你可以考虑使用简单的冒泡排序或效率更高的快速排序方法等等来解决问题。

没错，你应该也感觉到了，算法是一种简单的经验总结和套路。那什么是思想呢？相较于算法，思想更多的是指导你我来解决问题。

比如说，在解决一个复杂问题的时候，我们可以先将问题简化，先解决简单的问题，再解决难的问题，那么这就是一种指导解决问题的思想。另外，我们常说的分治也是一种简单的思想，当然它在诸如归并排序或递归算法当中会常常被提及。

而动态规划就是这样一个指导我们解决问题的思想：**你需要利用已经计算好的结果来推导你的计算，即大规模问题的结果是由小规模问题的结果运算得来的。**这句话对于你充分理解动态规划的基本原理十分重要，希望你能记下来。

简单理解的话，你可以这样认为：**算法是一种经验总结，而思想则是用来指导我们解决问题的。**

既然动态规划是一种思想，那它实际上就是一个比较抽象的概念了，也很难和实际的问题关联起来。所以说，弄清楚什么样的问题可以使用动态规划来解，就显得十分重要了。

动态规划问题的典型特点

在前几课中，我们已经基本了解了动态规划的基本概念，从贪心算法、暴力递归再到最后的动态规划解法，我们比较完美地解决了提出的问题。在这个过程中，你有没有想过这些问题为什么可以用动态规划来解决？是别人这么做我也要这么做吗？别人的经验又是从何而来？

事实上，动态规划是运筹学上的一种最优化方法，只不过在算法问题上应用广泛。接下来我们就深挖一层，看看动归问题所具备的一些特点。

求"最"优解问题（最大值和最小值）

除非你碰到的问题是简单到找出一个数组中最大的值这样，对这种问题来说，你可以对数组进行排序，然后取数组头或尾部的元素，如果觉得麻烦，你也可以直接遍历得到最值。不然的话，你就得考虑使用动态规划来解决这个问题了。这样的问题一般都会让你求最大子数组、求最长递增子数组、求最长递增子序列或求最长公共子串、子序列等等。不知道你发现没有，这些问题里都包含一个“最”字，如果出现了这个字，那么你就该警惕它是否是动归问题。那具体怎么判断呢？

既然是要求最值，不妨先想一下核心问题是什么。其实在真的解决最值问题的时候，你应该按照这样的思考顺序来解决问题：

优先考虑使用贪心算法的可能性；

然后是暴力递归进行穷举（但这里的数据规模不大）；

还是不行呢？选择动态规划！

你也看到了，求解动态规划的核心问题其实就是穷举。那么因为我们要求最值，就肯定要把所有可行的答案穷举出来，然后在其中找最值就好了嘛。你看，动态规划也不过如此，就两个字：**穷举**。

当然了，动态规划问题也不会这么简单了事，我们还需要考虑待解决的问题是否存在重叠子问题、最优子结构等特性。我们已经在前面的课程中讲清楚了重叠子问题，而对于最优

子结构，我会在后续的内容中继续给你讲清楚。

清楚了特点，那根据我的经验，绝大多数面试者其实还是很难在第一时间针对具体问题采取明确的行动：这个问题到底该不该用动态规划思想来解呢？

我在这里列出几道常见的经典面试题，如果你遇到它们了，不要犹豫，使用动态规划来解。这样不仅目标明确，而且会在面试时给面试官留下极为深刻的印象（这一讲我们只是分析下题目，后面的课程中会涵盖所有问题的解法，并归纳总结解题套路）。

1. 乘积最大子数组

问题：给你一个整数数组 numbers，找出数组中乘积最大的连续子数组（该子数组中至少包含一个数字），返回该子数组的乘积。

复制代码

```
示例1:
输入: [2,7,-2,4]
输出: 14
解释: 子数组 [2,7] 有最大乘积 14。
```

复制代码

```
示例2:
输入: [-5,0,3,-1]
输出: 3
解释: 结果不能为 15, 因为 [-5,3,-1] 不是子数组，是子序列。
```

首先，很明显这个题目当中包含一个“最”字，使用动态规划求解的概率就很大。这个问题的目的就是从数组中寻找一个最大的连续区间，确保这个区间的乘积最大。由于每个连续区间可以划分成两个更小的连续区间，而且大的连续区间的结果是两个小连续区间的乘积，因此这个问题还是求解满足条件的最大值，同样可以进行问题分解，而且属于求最值问题。同时，这个问题与求最大连续子序列和比较相似，唯一的区别就是你需要在这个问题里考虑正负号的问题，其它就相同了。

2. 最长回文子串

问题：给定一个字符串 s，找到 s 中最长的回文子串。你可以假设 s 的最大长度为 1000。

复制代码

```
示例1:
输入: "babad"
输出: "bab"
```

复制代码

```
示例2:
输入: "cbbd"
输出: "bb"
```

这个问题依然包含一个“最”字，同样由于求解的最长回文子串肯定包含一个更短的回文子串，因此我们依然可以使用动态规划来求解这个问题。

3. 最长上升子序列

问题：给定一个无序的整数数组，找到其中最长上升子序列的长度。可能会有多种最长上升子序列的组合，你只需要输出对应的长度即可。

复制代码

```
示例:
输入: [10,9,2,5,3,7,66,18]
输出: 4
解释: 最长的上升子序列是 [2,3,7,66]，它的长度是 4。
```

这个问题依然是一个最优解问题，假设我们要求一个长度为 5 的字符串中的上升自序列，我们只需要知道长度为 4 的字符串最长上升子序列是多长，就可以根据剩下的数字确定最后的结果。

求可行性（True 或 False）

接下来我们再来看另一种可能的动态规划问题。

如果有这样一个问题，让你判断是否存在一条总和为 x 的路径（如果找到了，就是 True；如果找不到，自然就是 False），或者让你判断能否找到一条符合某种条件的路径，那么这

类问题都可以归纳为求可行性问题，并且可以使用动态规划来解。比如我们前面课程中提到的找零钱问题，是不是就很好地说明了这一点？

1. 凑零兑换问题

问题：给你 k 种面值的硬币，面值分别为 c1, c2 ... ck，每种硬币的数量无限，再给一个总金额 amount，问你最少需要几枚硬币凑出这个金额，如果不可能凑出，算法返回 -1 。

复制代码

```
示例1:
输入: c1=1, c2=2, c3=5, c4=7, amount = 15
输出: 3
解释: 11 = 7 + 7 + 1。
```

复制代码

```
示例2:
输入: c1=3, amount =7
输出: -1
解释: 3怎么也凑不到7这个值。
```

这个问题显而易见，如果不可能凑出我们需要的金额（即 amount），最后算法需要返回 -1，否则输出可能的硬币数量。这是一个典型的求可行性的动态规划问题。

2. 字符串交错组成问题

问题：给定三个字符串 s1, s2, s3, 验证 s3 是否是由 s1 和 s2 交错组成的。

复制代码

```
示例1:
输入: s1="aabcc",s2 ="dbbca",s3="aadbbcbcac"
输出: true
解释: 可以交错组成。
```

复制代码

```
示例2:
输入: s1="aabcc",s2="dbbca",s3="aadbbbaccc"
输出: false
```

```
4 解释:无法交错组成。
```

这个问题稍微有点复杂，但是我们依然可以通过子问题的视角，首先求解 s1 中某个长度的子字符串是否由 s2 和 s3 的子字符串交错组成，直到求解整个 s1 的长度为止，也可以看成一个包含子问题的最值问题。

求方案总数

除了求最值与可行性之外，求方案总数也是比较常见的一类动态规划问题。比如说给定一个数据结构和限定条件，让你计算出一个方案的所有可能的路径，那么这种问题就属于求方案总数的问题。我在这里介绍几个典型例子，帮助你理解。

1. 硬币组合问题

问题：英国的英镑硬币有 1p, 2p, 5p, 10p, 20p, 50p, £1 (100p), 和 £2 (200p)。比如我们可以用以下方式来组成 2 英镑：1×£1 + 1×50p + 2×20p + 1×5p + 1×2p + 3×1p。问题是一共有多少种方式可以组成 n 英镑? 注意不能有重复，比如 1 英镑 +2 个 50P 和 50P+50P+1 英镑是一样的。

复制代码

```
1 示例1:
2 输入: 2
3 输出: 73682
```

这个问题本质还是求满足条件的组合，只不过这里不需要求出具体的值或者说组合，只需要计算出组合的数量即可。

2. 路径规划问题

问题：一个机器人位于一个 m x n 网格的左上角。机器人每次只能向下或者向右移动一步。机器人试图达到网格的右下角，共有多少路径?

复制代码

```
1 示例1:
2 输入: 2 2
3 输出: 2
```

复制代码

```
示例1:
输入: 3 3
输出: 6
```

这个问题还是一个求满足条件的组合数量的问题，只不过这里的组合变成了路径的组合。我们可以先求出长宽更小的网格中的所有路径，然后再在一个更大的网格内求解更多的组合。这和硬币组合的问题相比没有什么本质区别。

这里有一个规律或者说现象需要强调，那就是求方案总数的动态规划问题一般都指的是求“一个”方案的所有具体形式。如果是求“所有”方案的具体形式，那这种肯定不是动态规划问题，而是使用传统递归来遍历出所有方案的具体形式。

为什么这么说呢？因为你需要把所有情况枚举出来，大多情况下根本就没有重叠子问题给你优化。即便有，你也只能使用备忘录对遍历进行一个简单加速。但本质上，这类问题不是动态规划问题。接下来你就会看到这样的例子，你可以找找区别在哪里。

进一步确认是否为动态规划问题

从前面我所说来看，如果你碰到了求最值、求可行性或者是求方案总数的问题的话，那么这个问题就八九不离十了，你基本可以确定它就需要使用动态规划来解。但这里还有一些极具迷惑性的问题，你需要格外注意。

数据不可排序（Unsortable）

假设我们有一个无序数列，希望求出这个数列中最大的两个数字之和。很多初学者刚刚学完动态规划会走火入魔到看到最优化问题就想用动态规划来求解，嗯，那么这样应该也是可以的吧……不，等等，这个问题不是简单做一个排序或者做一个遍历就可以求解出来了吗？所以学完动态规划后，你一定要注意，遇到这些简单的问题不要把事情变得更复杂了。先考虑一下能不能通过排序来简化问题，如果不能，才极有可能是动态规划问题。还是看个例子。

最小的 k 个数

问题：输入整数数组 arr ，找出其中最小的 k 个数。例如，输入 4、5、1、6、2、7、3、8 这 8 个数字，则最小的 4 个数字是 1、2、3、4。

复制代码

```
示例1:
输入: arr = [3,2,1], k = 2
输出: [1,2] 或者 [2,1]
```

复制代码

```
示例2:
输入: arr = [0,1,2,1], k = 1
输出: [0]
```

我们发现虽然这个问题也是求“最”值，但其实只要通过排序就能解决，所以我们应该用排序、堆等算法或者数据结构来解决，而不应该用动态规划。

数据不可交换（Non-swapable）

还有一类问题，可以归类到我们总结的几类问题里去，但是不存在动态规划要求的重叠子问题（比如经典的八皇后问题），那么这类问题就无法通过动态规划求解。这种情况需要避免被套进去。

全排列

问题：给定一个没有重复数字的序列，返回其所有可能的全排列。

复制代码

```
示例:
输入: [1,2,3]
输出:
[
  [1,2,3],
  [1,3,2],
  [2,1,3],
  [2,3,1],
  [3,1,2],
  [3,2,1]
]

```

这个问题虽然是求组合，但没有重叠子问题，更不存在最优化的要求，因此可以使用回溯处理，并不是动态规划的用武之地。

课程总结

今天，我们一起探讨了动态规划问题的本质，更准确或更加严谨地说，动态规划是一种指导我们解决问题的思想。

接着我们列出了辨别一个算法问题是否该使用动态规划来解的五大特点：

1. 求最优解问题（最大值和最小值）；
2. 求可行性（True 或 False）；
3. 求方案总数；
4. 数据结构不可排序（Unsortable）；
5. 算法不可使用交换（Non-swappable）。

如果面试题目出现这些特征，那么在 90% 的情况下你都能断言它就是一个动归问题。

当然了，就像我前面所讲的，你还需要考虑这个问题是否包含重叠子问题与最优子结构，在这个基础之上你就可以 99% 断言它是否为动归问题，并且也顺势找到了大致的解题思路，我会在后面的课程中继续跟你探讨这些问题，彻底解决你的疑惑。

通过上述这几个鲜明的特点，相信你能够在将来迅速地判断出问题是否为动态规划类问题，并使用对应的思想和套路来应对算法或面试问题。

课后思考

1. 除了我在这里列出的动态规划特点以外，你觉得还有哪些类别的问题应该进行归纳总结？能否把你见过的或认为是动态规划的算法留在评论区，并分析一下它们又属于哪些类别。
2. 我在前面提到过子数组与子序列的问题，请你思考一下，这两种情况有什么区别？

欢迎留言和我分享，我会第一时间给你反馈。如果今天的内容让你对动态规划的用法有了进一步的了解，也欢迎把它分享给你身边的朋友，邀请他一起学习！

提建议

上一篇 04 | 动态规划：完美解决硬币找零

下一篇 06 | 0-1背包：动态规划的Hello World

精选留言 (6)

写留言

落曦

2020-10-20

动态规划问题，先看如何进行穷举，再去找重叠子问题以及无后效性，以及最优子结构
通常要求的题目为最优解问题，最大值，最小值所构成的最优方案，方案总数，就是能够实现的方案数量。

作者回复: 总结的很对，顶上去让跟多人看到。

1

Mr.郑先生_

2020-10-07

乘积最大子数组的示例二，输出不应该是3么？

展开

1

猫头鹰爱拿铁

2020-10-09

感觉字符串的动态规划题都比较难。

展开

作者回复: 字符串的动态规划是特定的一类动态规划问题，这类问题在后面的子数组和子序列问题中会详细讲解，并给出求解模板和套路。

建议阅读参考一下。

Scott

2020-09-30

子是不是子数组一般用一维数组保存状态，而子序列一般用二维?

作者回复: 这跟状态参数的定义是有极大关系的。在后面讲解12课动态规划新问题2的时候，你就会发现即便是子数组问题，也需要高维备忘录来存储中间计算的过程。

具体问题具体分析。因此，简单的子数组问题可以认为只需要一维数组作为备忘录就可以解决问题；子序列问题一般需要高维数组作为备忘录解决问题。

但这不是绝对的。在学习了后面的思路分析和解题步骤后，相信你会有新的认识。

Ethan Liu
2020-09-26

老师 全排列那里 数据不可交换是什么意思？与重叠子问题有什么关系？

作者回复: 举个例子，给定一个数组 [2, 3, 1, 6, 7]，如果要求最长上升子串，我们该怎么求呢?

需要做排列组合，找到最长的那个吧？那么如果数据“可交换”，会发生什么？

如果我们把数组重新排序，那么原问题也就不复存在了。所以当原问题可以重新排序时，那么那个问题不用动态规划来解，因为不需要穷举。

比如说，我们还是上面的数组作为输入，但问的问题是数组中最大值是什么？那么你可以先排序，然后取开头或结尾的值，就是你要的答案了。

这里想强调说明的是，并非看到数组或字符串问题都需要使用动态规划来求解。也许朴素的算法能更快得到解。

山茶花
2020-09-23

老师你好，请问后面是否有完全背包问题求第K优解的相关讲解呢？当前我遇到了类似问题，目前使用的是穷举+备忘录法解决的，希望可以有相关讲解

展开

作者回复: 请阅读一下 06 和 07 课针对背包问题的完整阐述。
在讲解完全背包的部分，里面有详细的讲解。