冬季训练课程之动态规划

Edit by AntiheroChen

2012 年 2 月 1 日

Abstract

这篇文章，目的在于带领各位同学走近动态规划（Dynamic Programming[1]）算法。

这份讲义会涉及到：问题引入和学习建议；一些基本的概念；动态规划使用的条件和局限；一些基本的动态规划思路；一些经典的动态规划模型；一些推荐的习题。

在各种编程竞赛，动态规划以善于体现选手思维水平而备受重视；在工程和研究中，动态规划作为一项极其 有效的工具方法，也展现了它的威力。对于*ACM*的选手，无论从竞赛角度还是锻炼自己思维的角度而言， 都是占据了相当的重要性。而且，由于动态规划的灵活多变和较少的代码量[2]，也广受选手们乃至程序设计师们的追捧和喜爱。

希望，首先，你们能爱上这个算法；然后，才是用好它。

Then just enjoy it.

Keywords:动态规划;汗水和灵感;思维和兴趣;ACM\ICPC

[1]简称"DP"。

[2]这是针对绝大多数问题而言 ，并非绝对，比如插头DP问题。

Contents

1 序

动态规划（Dynamic Programming）是一种在数学和计算机科学中使用的，用于求解包含重叠子问题的最优化问题的方法。

与其说动态规划是一种算法，不如说是一种思想。用直接的话来解释可以这么说：在解空间内，**为了避免对同一状态点重复计算，而记忆每一状态所需求出的答案**，使多次使用该次状态点的答案的时候可以只计算一次的一种数学方法。

例题一 AntiheroChen（小A）住在遥远的四川，每次回四川都是一个煎熬的过程，因为从他所就读的大学 — 哈尔滨工业大学所在的哈市，没有直达四川火车。 所以，他不得不选择通过一些车站进行中转。这时候就浮现出来一个问题，可以选择的中转城市如此之多（小A高超的抢票能力保证了他的 备选方案如此之多），他必须首先确定从哪个车站中转，保证能尽快回家。 为此，他还搞到了一份路线图**Figure 1**来辅助他分析和选择。

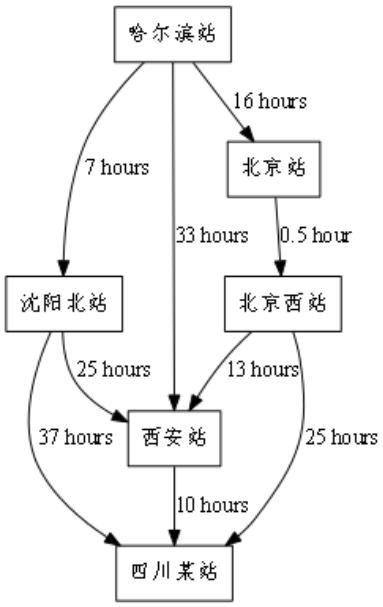

Figure 1: 火车路线图

这个问题中，按照一般的搜索的思维：从起点出发，到终点结束，计算出一条路的时间，更新最佳答案，重复直到计算过所有路线。

经过简单分析，我们不难发现，到**四川的最短路肯定是由西安、沈阳和北京西三站的最短路中选出来的**。而**西安的最短路 肯定是由到北京西、沈阳和哈尔滨的最短路中选出来的**。换句话说，要求到四川的最短路，我们可以先求到西安、沈阳和北京西的最短路； 同理，要求到西安的最短路，我们可以先求到北京西、沈阳和哈尔滨的最短路。

更重要的，通过这样的思路，我们把原来的问题[3]分解成了很多子问题[4]， 只要我们按顺序解决了这些小问题并记录答案，我们的原问题就得到了解决。

那么这个只是解题和分析思路，我们还需要考虑是否这个方法就更高效? 为什么这个方法更高效?

分析思路如下：我们从每条连边考虑。

- 对于一般搜索思路，显然，每个边会计算一次以上。
- 对于后一种方法而言，我们每个边只计算了一次[5]。

如果你足够耐心，可以计算出来，前一种方法一共会调用12次边的数据；后一种方法只会调用9次。对于足够复杂的图来说，后者方法的 效率会更加突出。

为什么，后一种方法会如此高效?

因为，我们记录了到每个点的最短路，避免了多次计算到某个点的路径，达到了减少冗余的效果。 实际上，只要你记录了每个点的最优值，无论你用BFS还是DFS，亦或是利用邻接表进行枚举，都能达到相同的复杂度。

接着，根据这道题的解题思路，我们归纳下，DP方法能够使用的前提：

- 无后效性：保证调用更新的关系是偏序关系，即后求出来的状态的答案不会对它的前提状态造成影响，否则当前 最优这个性质无法得到有效的保证。

[3]求到四川的最短路的问题。

[4]求到各站的最短路的问题。

[5]比如:西安至四川线，只在用到西安的最短路来更新 到四川的最短路时被调用。

- 最优子结构：保证了原问题可以归纳成规模更小的子问题。反映在这道题上，体现为求到四川的最短路可以通过求规模更小的到达 其它地点的最短路的问题。

动态规划算法其实并没有固定的一个代码，而是在特定模型下的一种解题思路。打个比方，你给果树 喷撒农药，用什么器具不是最重要的，重要的是装的是什么药。所以学习动态规划要求我们多思考，而不是盲目堆砌代码量，盲目堆砌 刷题数。

思考：

- 如果**例题一**中出现了环路[6]，DP能否继续使用?
- 如果拓展模型，允许负边出现，又如何?
- 在拓展模型的基础下，又允许负边出现。DP一定会出错，需要什么条件?
- 结合以上几题的思考，重新回顾DP使用的前提。

[6]即：从一地出发可以回到该地。

2 一些基本概念

动态规划有一些通用的术语[7]，我这里大概介绍一下。

首先是DP适用前提，在决策模型下，要求满足[8]：1.无后效性。2.最优子结构。

然后是描述DP过程的一些术语[9]，如状态和决策：

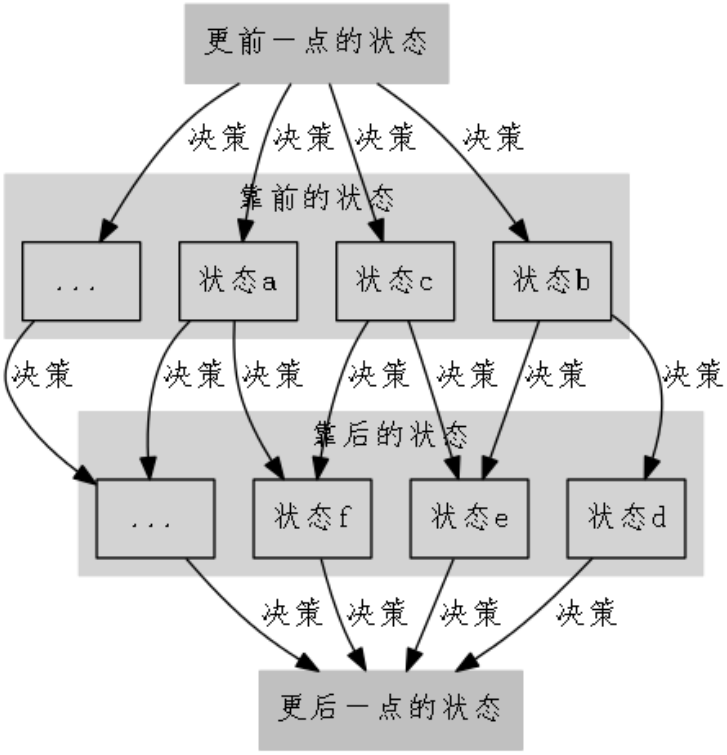

Figure 2: 决策和状态

而用方程形式对决策和状态进行表述则称为**状态方程**。一般而言，状态方程就能成功地表述对于待解问题的动态规划解法，采用方程的方式进行描述不但简练准确，而且贴近代码实现，是广大竞赛选手表述动态规划思路的首要选择。同时需要注意， 状态方程要求表述转移公式和边界条件。

[7]便于学术交流，沟通解题思路等等，不用特别注意。

[8]请结合例题和练习题进行理解!

[9]基于决策模型下的术语，实际上还有其他实现方式，ex，记忆化搜索。

在程序中，用于储存各部分状态所求答案的数组，称为**状态数组**[10]。

3 一些经典的模型

下面是一些经典的动态规划模型，不要求各位同学强行背记，但要求各位同学积极思考，随同讲义的思路，领会动态规划的一些基础概念和构思 的思路。请先完成基础模型的阅读，和一些基础练习。学有余力的同学可以认真完成下思考题，必然大有裨益!

3.1 方格取数

例题二 [11] 如图，在一个方格(N x M)中，从左上角开始取数，每次取数，只能在上次所取位置的右方或下方取数，直到取到右下角的 数字为止。问：如何最大化所取到数字之和?

Table 1: 方格取数问题

0	0	0	0	0	0	0	0
0	0	13	0	0	6	0	0
0	0	0	0	7	0	0	0
0	0	0	14	0	0	0	0
0	21	0	0	0	4	0	0
0	0	15	0	0	0	0	0
0	14	0	0	0	0	0	0
0	0	0	0	0	0	0	0

分析： 首先我们设$f[i,j]$表示在取到第i行第j列时取到的最大数字。发现$f[i,j]$只与$f[i-1,j]$和$f[i,j-1]$有关； 因为这个点的最优值只能由它的

[10]并不一定是以数组的形式储存。

[11]此题可以用DP解决，但并非仅能，本章例题均是如此，不要局限思维。

上边或左边的点进行更新。容易得出：

$$f[i,j] = Matrix[i,j] + \begin{cases} 0 & i = 0 and j = 0 \\ max(0, f[i-1,j]) & i \neq 0 and j = 0 \\ max(0, f[i,j-1]) & i = 0 and j \neq 0 \\ max(f[i-1,j], f[i,j-1]) & i \neq 0 and j \neq 0 \end{cases}$$

为了简便起见[12]，一般记作：

$$f[i,j] = Matrix[i,j] + max(f[i-1,j], f[i,j-1]);$$

根据状态转移方程，不难实现其代码。同时，也可以分析出时间复杂度为$O(n^2)$,空间复杂度也为$O(n^2)$。

回顾解题过程，可以发现，动态规划解题的一般思路：

- 分析题意，建立模型。根据模型写出**状态数组，明确每一维的含义！**
- 根据题目给出的关系，确立**状态方程**。
- 分析题目得出**边界条件**和**初值**。
- 分析得出算法的复杂度，如果可以接受，便进行代码实现。

思考：如果可以重复取两次（第一次取过之后的格子会变成0），该如何求解?

对应题目：HOJ 1058。

[12]如果没有特殊说明，以后都是只写出简便公式。

3.2 最长公共子序列

例题三 问题描述，给定两个子序列A和B，求出这两个字符串最长的公共子序列[13]的长度，如果要求你给出该最长公共子序列又该如何处理?

分析： 设f[i,j]表示A字符串的前i个字符所组成的字符串与B字符串的前j个字符组成的字符串的最长公共子序列的长度。 我们从1开始，依次枚举i，j。分以下两种情况处理:

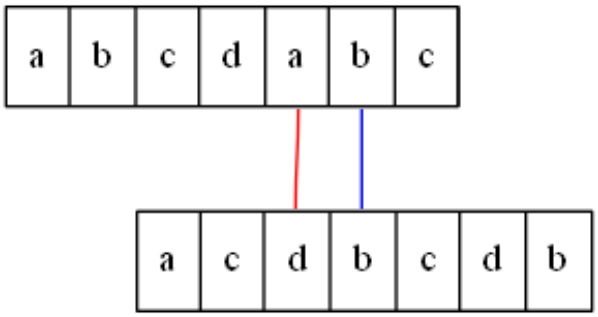

Figure 3: 当A[i]等于B[j]

此时，$f[i,j] = f[i-1,j-1] + 1$;

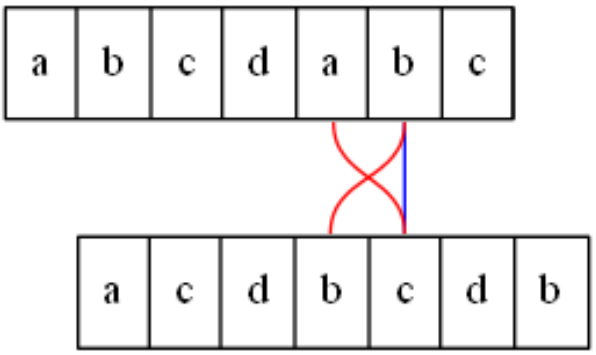

Figure 4: 当$A[i]$不等于$B[j]$

此时，$f[i,j] = max(f[i-1,j], f[i,j-1])$。

时间复杂度和时间复杂度都是$O(n^2)$。

[13]子序列：在保持原有成员顺序不变 的情况下，去掉一些字符得到的新字符串。

思考：如何证明该方法的正确性? 试试用所给概念和方法重新描述和推导下本题的DP方法。如果让你给出该最长子序列， 这里需要加上什么内容?

对应题目：HOJ 1227 1316

3.3 最长上升子序列

例题四 对于一个整数序列A，求出它的最长的升序子序列[14]。

分析：设$f[i]$表示原数列的前i个元素所组成的序列的最长升序数列的长度。用j枚举小于i的所有值，当$A[j] < A[i]$时， $f[i] = max(f[i], f[j]+1)$。

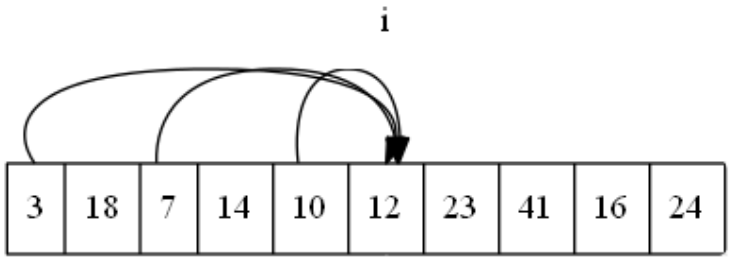

Figure 5: $f[i]$的可转移点

时间复杂度$O(n^2)$，空间复杂度是$O(n)$。

思考：如果$f[i]$代表该子序列长为i时，最小的末尾数字。那么这个DP方法该如何解? 它的空间复杂度和时间复杂度，你能 约束到哪个地步?

对应题目：HOJ 1288

3.4 01背包问题

例题五 问题描述：你有一个背包可以装下一些物品，每个物品的价值不同。然而，你的背包容量 有限，不能装下所有物品，每个物品的体积也不一样。你想最大化所装物品的价值和。

[14]保证子序列的成员依次增大。

分析：初步模型：$f[i,j]$表示只取前i种物品时,体积为j的背包可以容纳的物品的最大价值和。

考虑对于当前的物体i，它的价格为$price[i]$，价值为$value[i]$。此时，初始化$f[i][j] = f[i][j-1]$。 如果我们一定要把物品i塞进背包，得到的价值为 $f[i-1][j-price[i]] + value[i]$；或者干脆不要装这个物品，得到的价值为$f[i-1][j]$。

状态方程便如此容易的得到：$f[i][j] = max(f[i][j-1], f[i-1][j], f[i-1][j-price[i]]+value[i])$。时间 复杂度为$O(n*m)$，空间复杂度为$O(n*m)$。

这里主要的思路是：用不同维数完整地表示出当前状态，调用已经求出的状态来更新当前状态。只要保证了问题 可以用动态规划求解，这种思路能帮助你解决绝大数的动态规划问题。巧妙地改良维数来改善对于状态的表示，是优化动态规 划最直接的方法。

思考：如果j是从高到低枚举的，那么可以省去状态方程的第一维，为什么?

对应题目：HOJ 1388 1485 2196，拓展阅读《背包九讲》。

3.5 最大子段和

例题六 问题描述：对于一个有序数列A，求它的一个由一些连续的元素所组成的子段[15] ，最大化这个子段所有元素的和。

分析：用表示$f[i]$以第i个元素结尾的最大子段和。抓住子段必须连续的这个性质，发现以第i元素结尾的最优子段： 要么只包含i元素；要么包含i元素和以i-1元素结尾的最优子段。我们取其较大者即可。

另一种思路是：对于以i元素结尾的子段，当前它只包含元素i，如果继续向前延伸能够得到积极影响，则继续向前延伸；否则不向前延伸。

于是得到：$f[i] = A[i] + max(f[i-1], 0)$。

时间复杂度为$O(n)$，空间复杂度为$O(n)$。如果滚动存储，时间复杂度可以降低到$O(1)$。

最大子段和，常常作为求解一些中高等难度的问题一个预处理或者中间过程出现，要求思路清晰，代码简单，不能出错。

[15]至少包含一个元素

思考：如何，快速而简便地证明该DP方法的正确性? 两种思路的不同点在哪? 它们的优点分别有哪些? 你更喜欢哪一个? 对应题目：HOJ 1760

3.6 整数的划分

例题七 对于一个正整数n，把它划分成几个正整数值和的形式，共有多少种划分方法。

分析：这里的难点在于，如何去掉重复方案。比如$11 = 7 + 4$和$11 = 4 + 7$都是同一种划分。 初步模型：用$f[i]$表示整数i划分的方法数，在我们枚举$f[j](j < i)$值来更新$f[i]$的时候，发现几乎无法避免重复方案重复地 被计算[16]。这说明：**一维的状态数组无法充分地表示动态规划 所需的状态的性质**，那么我们需要增加维数来更加细致地区分每个状态。

一个可行的思路是：我们发现原数所划分成的数字之间存在着一个关系——大小关系，这是个偏序关系。如果我们顺着偏序关系进行枚举，就可以避免重复。换句话说，我们确保当前枚举数字比之前枚举的都要大[17]。

那么新的模型是：$f[i, j]$表示整数i划分成不超过j的整数的方案总数。那么我们需要，所有小于j的整数对i-j进行划分的方案总数，来更新 当前状态：

$$f[i, j] = \sum_{k=1}^{j} f[i - j][k]$$

通过递推可以简化为：

$$f[i, j] = f[i, j - 1] + f[i - j, j];$$

另一种思路是：$f[i, j]$表示整数i划分成不超过j的整数的方案总数。$f[i, j]$包含了两种情况：

- 一定包含j的划分，其方案数为$f[i - j, j]$;
- 一定不包含j的划分，其方案数为$f[i, j - 1]$。

[16]如果组合数学功底过硬，可以试图直接求出通解的公式，:-)。

[17]同理，取小也可以。

得到的方程也是上面的方程。

最后，时间复杂度为$O(n^2)$；空间复杂度为$O(n^2)$。

思考：试用背包问题的眼光看待这个问题，重新进行一次构思。如果要求至少划分成k份，又该如何解决? 体会状态方程维数 对于动态规划模型的意义。

对应题目：1402

3.7 总结建模的两种常用思路

综合上面六道例题，我们可以总结出两大常见的DP动机[2]。主要是从子问题的结构来下手：

- 具有“包含”关系的最优子结构。
- 具有“阶段”关系的最优子结构[18]。

由于黑书[19]对此已经有详细的叙述和经典而不陈旧的例子，所以请移步至 黑书进行进一步的学习，如果对此你有更多的精力和兴趣的话。

思考：你能对前面所提及的各种思路进行分类么?

References

[1] 杨靖, 《*DP*基本概念》, 2011.

[2] 刘汝佳,黄亮,《算法艺术与信息学竞赛》, 清华大学出版社, 2004.

[18]广义地讲，后者可以归纳到前者中去。就我们思索思路而言，存在很大区别。

[19]即：《算法艺术与信息学竞赛》。

4 一些要求和推荐习题

在2.1-2.2日，要求大家完成本课件经典模型的对应习题，并上交一份总结。 总结要包含：这两天的学习状态、课件的学习进度以及习题的完成效果。

在2.3-2.4日，要求完成以下习题，并于2.4日参加当日的DP综合测试题目，时间为3个小时，题量预计为6道[20]。 如果没有能在比赛期间AK，要求赛后完成比赛题目，并且上交第二份动态规划课程的总结。

推荐习题：以下习题并不作硬性要求，请根据自己的学习情况酌情安排。

HOJ1058 Number Triangles

HOJ1447 Compromise

HOJ1603 Brackets Sequence

HOJ1249 Optimal Array Multiplication Sequence

HOJ1714 Minimax Triangulation

HOJ1005 Fast Food

如果有同学仍然“吃不饱”，请阅读黑书相关章节。

5 进阶技巧（不做要求且尚未就位）

本内容均大大超出寒假集训难度，这里提出只作为名词上的了解。

会在后续版本的本讲义中放出，敬请期待。

5.1 状态压缩

尚未制作，敬请期待。

5.2 改善模型与视角

尚未制作，敬请期待。

[20]DP代码量不会太大，不要担心。

5.3 树结构上的动态规划

尚未制作，敬请期待。

5.4 按位的动态规划

尚未制作，敬请期待。

5.5 斜率优化

尚未制作，敬请期待。

5.6 平行四边形优化

尚未制作，敬请期待。

5.7 利用高级数据结构优化算法

尚未制作，敬请期待。

5.8 插头动态规划问题

尚未制作，敬请期待。

6 跋

本文经典模型一节大量参考了杨靖学长一文。实际上，学长之作已然足够作为此次寒假讲稿，只是本人奔着练习科技文献排版和完善寒假动态规划讲稿的目地制作了本文。

本文在讲稿上主要注重两点：1.减少文字描述，增加图像。2.每句尽量短。嘴上说起来容易，实际做起来比较困难，作为一个初涉排版的学生，掂量细节，以及把握页面结构上还存在很多问题；画图着实费了不少功夫，一些思维导图的软件并不能最好地完成我们所想要的效果，特别是 对中文环境的表现。

各位同学且将就着看，我会不断维护和拓冲这个课件。如果你们坚持到了暑假集训，我或许也已经完成了这个课件的**进阶技巧**这一部分。 然后你们会再次接触到这个讲稿[21]，从而学习到一些更加有趣的进阶技巧。

或许有一天，你投入了很多精力在动态规划的学习上，然后发现工作学习中用到的并非想象中那么多。但是，你不会后悔，你知道在学习动态规划 的这条路上，你各方面的能力已经提升了很多。搞ACM也好，学习算法也好，我们并非单纯为知识性掌握这些方法而学习。要知道以后的工作永远 是开卷考试，你的优势不该是寄托在某些知识上，而是在自己的能力上。

到最后，如果你们对于这个课件的改良有任何意见和建议，请发送邮件到AntiheroChen@gmail.com来与我联系。

谢谢!

[21]当然课件名称也会随之改变