【热点·难点·赛点】

动态规划是信息学奥林匹克竞赛最为热门的一个命题方向，它是解决最优化问题的一种十分有效的方法，几乎在每一次竞赛中都会涉及到动态规划方面的试题，这是由于动态规划方法解决了搜索中的大量冗余的出现，提高了算法的时间效率。动态规划方法的难点在于如何将一个实际问题转化为一个动态规划模型，并进一步确定动态规划的几个基本要素：阶段的划分、状态的描述方式和状态之间的转移关系，即状态转移方程的确定。一般说来，动态规划方法所需要的时间复杂度为问题的规模（n）的平方级（$o(n^2)$）,因此在大多数情况下对动态规划方法的时间的优化考虑得不是很多，由于动态规划方法中需要保留很多状态变量而需要很大的空间需求，所以实际问题中更多地考虑其空间的优化方法。

【解题方法例析】

1、 最短路径问题。图7-1中给出了一个地图，地图中每个顶点代表一个城市，两个城市间的连线代表道路，连线上的数值代表道路的长度。现在，想从城市A到达城市E，怎样走路程最短，最短路程的长度是多少？

【分析】把从A到E的全过程分成四个阶段，用k表示阶段变量，第1阶段有一个初始状态A，两条可供选择的支路AB1，AB2；第2阶段有两个初始状态B1，B2，B1有三条可供选择的支路，B2有两条可供选择的支路……。用$d_k(x_k,x_{k+1})$表示在第K阶段由初始状态x_K到下阶段的初始状态x_{K+1}的路径距离，$f_k(x_k)$表示从第K

图7-1 地图

阶段的x_k到终点E的最短距离，利用倒推方法求解A到E的最短距离。具体计算过程如下：

S1：K=4，有：$f_4(D1)=3$, $f_4(D2)=4$, $f_4(D3)=3$

S2：K=3，有：$F_3(C1)=\min\{d_3(C1,D1)+f_4(D1), d_3(C1,D2)+f_4(D2)\}=\min\{8,10\}=8$

$F_3(C2)=d_3(C2,D1)+f_4(D1)=5+3=8$

$F_3(C3)=d_3(C3,D3)+f_4(D3)=8+3=11$

$F_3(C4)=d_3(C4,D3)+f_4(D3)=3+3=6$

S3：k=2，有：F_2（B1）$=\min\{d_2(B1,C1)+f_3(C1), d_2(B1,C2)+f_3(C2), d_2(B1,C3)+f_3(C3)\}$

$=\min\{9,12,14\}=9$

F_2（B2）$=\min\{d_2(B2,C2)+f_3(C2), d_2(B2,C4)+f_3(C4)\}=\min\{16,10\}=10$

S4：k=1，有：$f_1(A)=\min\{d_1(A,B1)+f_1(B1), d_1(A,B2)+f_1(B2)\}=\min\{13,14\}=13$

因此由A点到E点的全过程的最短路径为A->B2->C4->D3->E。最短路程长度为13。

从以上过程可以看出，每个阶段中，都求出本阶段的各个初始状态到过程终点E的最短径和最短距离，当逆序倒推到过程起点A时，便得到了全过程的最短路径及最短距离，同时附带得到了一组最优结果（即各阶段的各状态到终点E的最优结果）。

在上例的多阶段决策问题中，各个阶段采取的决策，一般来说是与时间有关的，决策依赖于当前状态，又随即引起状态的转移，一个决策序列就是在变化的状态中产生出来的，故 有“动态”的含义，称这种解决多阶段决策最优化问题的方法为动态规划方法。

知识提示：什么样的“多阶段决策问题”才可以采用动态规划的方法求解？

一般说来，能够采用动态规划方法求解的问题必须满足最优化原理和无后效性原则。

（1）动态规划的最优化原理：作为整个过程的最优策略具有：无论过去的状态和决策如何，对前面的决策所形成的状态而言，余下的诸决策必须构成最优策略的特质。

可以通俗地理解为子问题的局部最优将导致整个问题的全局最优，即问题具有最优子结构的性质，也就是说一个问题的最优解只取决于其子问题的最优解，非最优解对问题的求解没有影响。在例题7-1最短路径问题中，A到E的最优路径上的任一点到终点E的路径也必然该点到终点E的一条最优路径，问题满足最优化原理。

（2）动态规划的无后效性原则：第阶段的状态一旦确定，就不受这之后决策的影响。也就是说，“未来与过去无关”，当前的状态是此前历史的一个完整总结，此前的历史只能通过当前的状态去影响过程未来的演变。具体地说，如果一个问题被划分各个阶段之后，阶段I中的状态只能由阶段I+1中的状态通过状态转移方程得来，与I-1及其他状态没有关系，特别是与未发生的状态没有关系，这就是无后效性。

2、数字三角形问题。图7-2示出了一个数字三角形宝塔。数字三角形中的数字为不超过100的整数。现规定从最顶层走到最底层，每一步可沿左斜线向下或右斜线向下走。

任务一：假设一角形行数≤10，键盘输入一个确定的整数值M，编程确定是否存在一条路径，使得沿着该路径所经过的数字的总和恰为M,若存在则给出所有路径，若不存在，则输出“NO Answer!”字样。

任务二：假设一角形行数≤100，编程求解从最顶层走到最底层的一条路径，使得沿着该路径所经过的数字的总和最大，输出最大值。

输入数据：中文件输入数据，任务一中文件第一行是三角形的行数N和整数值M。以后的N行分别是从最顶层到最底层的每一层中的数字。任务二中文件数据格式同任务一，只是第一行中没有整数值M。在例子中任务的文件数据表示如下：

```
5 28
7
3 8
8 1 0
2 7 4 4
4 5 2 6 5
```

输出：输出路径和最大值或“No Answer!”字样。

4 5 2 6 5

图**7-2** 数字三角形

【分析】对于这一问题，很容易想到用枚举的方法去解决，即列举出所有路径并记录每一条路径所经过的数字总和，然后判断数字总和是否等于给定的整数值M或寻找出最大的数字总和，这一想法很直观，而且对于任务一，由于数字三角形的行数不大（<=10），因此其枚举量不是很大，应该能够实现，但对于任务二，如果用枚举的方法，当三角形的行数等于100时，其枚举量之大是可想而知的，显然，枚举法对于任务二的求解并不适用。其实，只要对对任务二稍加分析，就可以得出一个结论：

如果得到一条由顶至底的某处的一条最佳路径，那么对于该路径上的每一个中间点来说，由顶至该中间点的路径所经过的数字和也为最大。因此该问题是一个典型的多阶段决策最优化的问题。算法设计与分析如下：

对于任务一，合理地确认枚举的方法，可以优化问题的解法。由于从塔顶到底层每次都只有两种走法，即左或右。设“0”表示左，“1”表示右，对于层数为N的数字塔，从顶到底的一种走法可用一个N-1位的二进制数表示。如例中二进制数字串1 0 1 1，其对应的路径应该是：8→1→4→6。这样就可以用一个N-1位的二进制数来模拟走法和确定解的范围。穷举出从0到2 n-1个十进制数所对应的N-1位二进制串对应的路径中的数字总和，判定其是否等于M而求得问题的解。

对于任务二，采用动态规划中的顺推解法。按三角形的行划分阶段。若行数为n, 则可把问题看作一个n-1个阶段的决策问题。从始点出发，依顺向求出第一阶段、第二阶段，……，第n-1阶段中各决策点至始点的最佳路径，最终求出始点到终点的最佳路径。

设：f k（Uk）为从第k阶段中的点Uk至三角形顶点有一条最佳路径，该路径所经过的数字的总和最大，f k（Uk）表示为这个数字和；

由于每一次决策有两个选择，或沿左斜线向下，或沿右斜线向下，因此设：

Uk1为k-1阶段中某点Uk沿左斜线向下的点；

Uk2为k-1阶段中某点Uk沿右斜线向下的点；

d k（Uk1）为k阶段中Uk1的数字；d k（Uk2）为k阶段中Uk2的数字。

因而可写出顺推关系式（状态转移方程）为：

f k(Uk)=max｛f k-1(Uk)＋d k(Uk1)，f k-1(Uk)＋d k(Uk2)｝(k= 1 , 2 , 3 ,……, n）

f 0(U0)= 0

经过一次顺推，便可分别求出由顶至底N个数的N条路径，在这N条路径所经过的N个数字和中，最大值即为正确答案。

3、花店橱窗布置问题（IOI99试题）。假设想以最美观的方式布置花店的橱窗，有F束花，每束花的品种都不一样，同时，至少有同样数量的花瓶，被按顺序摆成一行，花瓶的位置是固定的，并从左到右，从1到V顺序编号，V是花瓶的数目，编号为1的花瓶在最左边，编号为V的花瓶在最右边，花束可以移动，并且每束花用1到F的整数唯一标识，标识花束的整数决定了花束在花瓶中列的顺序，即如果I＜J，则花束I必须放在花束J左边的花瓶中。例如，假设杜鹃花的标识数为1，秋海棠的标识数为2，康乃馨的标识数为3，所有的花束在放入花瓶时必须保持其标识数的顺序，即：杜鹃花必须放在秋海棠左边的花瓶中，秋海棠必须放在康乃馨左边的花瓶中。如果花瓶的数目大于花束的数目,则多余的花瓶必须空，即每个花瓶中只能放一束花。

每一个花瓶的形状和颜色也不相同，因此，当各个花瓶中放入不同的花束时，会产生不同的美学效果，并以美学值（一个整数）来表示，空置花瓶的美学值为0。在上述例子中，花瓶与花束的不同搭配所具有的美学值，可以用如下表格表示：

	花瓶1	花瓶2	花瓶3	花瓶4	花瓶5
杜鹃花	7	23	-5	-24	16
秋海棠	5	21	-4	10	23
康乃馨	-21	5	-4	-20	20

根据表格，杜鹃花放在花瓶2中，会显得非常好看，但若放在花瓶4中则显得很难看。

为取得最佳美学效果，必须在保持花束顺序的前提下，使花的摆放取得最大的美学值，如果具有最大美学值的摆放方式不止一种，则输出任何一种方案即可。题中数据满足下面条件：$1\le F\le 100$，$F\le V\le 100$，$-50\le A_{IJ}\le 50$，其中A_{IJ}是花束I摆放在花瓶J中的美学值。输入整数F，V和矩阵（A_{IJ}），输出最大美学值和每束花摆放在各个花瓶中的花瓶编号。

【分析】问题实际就是给定F束花和V个花瓶，以及各束花放到不同花瓶中的美学值，需要你找出一种摆放的方案，使得在满足编号小的花放进编号小的花瓶中的条件下，美学值达到最大。

（1）将问题进行转化，找出问题的原型。首先，看一下上述题目的样例数据表格。

将摆放方案的要求用表格表现出来，则摆放方案需要满足：每行选且只选一个数（花瓶）；摆放方案的相邻两行中，下面一行的花瓶编号要大于上面一行的花瓶编号两个条件。这时可将问题转化为：给定一个数字表格，要求编程计算从顶行至底行的一条路径，使得这条路径所经过的数字总和最大（要求每行选且仅选一个数字）。同时，路径中相邻两行的数字，必须保证下一行数字的列数大于上一行数字的列数。

看到经过转化后的问题，发现问题与上例的数学三角形问题十分相似，数字三角形问题的题意是：

给定一个数字三角形，要求编程计算从顶至底的一条路径，使得路径所经过的数字总和最大（要求每行选且仅选一个数字）。同时，路径中相邻两行的数字，必须保证下一行数字的列数，与上一行数字的列数相等或者等于上一行数字的列数加1。

上例中已经知道：数字三角形中的经过数字之和最大的最佳路径，路径的每个中间点到最底层的路径必然也是最优的，可以用动态规划方法求解，对于“花店橱窗布置”问题经过转化后，也可采取同样的方法得出本题同样符合最优性原理。因此，可以对此题采用动态规划的方法。

（2）对问题原型动态规划方法的修改。“数字三角形”问题的动态规划方法为：已知它是用行数来划分阶段。假设用a[i,j]表示三角形第i行的第j个数字，用p[i,j]表示从最底层到a[i,j]这个数字的最佳路径（路径经过的数字总和最大）的数字和，易得问题的动态转移方程为：p[n+1,j]=0　　(1≤j≤n+1)

p[i,j]=max{p[i+1,j],p[i+1,j+1]}+a[i,j]　(1≤j≤i≤n，其中n为总行数)

分析两题的不同之处，就在于对路径的要求上。如果用path[i]表示路径中第i行的数字编号，那么两题对路径的要求就是：“数字三角形”要求path[i]≤path[i+1]≤path[i]+1，而本题则要求path[i+1]>path[i]。

在明确两题的不同之后，就可以对动态规划方程进行修改了。假设用b[i,j]表示美学值表格中第i行的第j个数字，用q[i,j]表示从表格最底层到b[i,j]这个数字的最佳路径（路径经过的数字总和最大）的数字和，修改后的动态规划转移方程为：

q[i,V+1]= -∞ (1≤i≤F+1)

q[F,j]=b[F,j]　(1≤j≤V)

q[i,j]=max{q[i+1,k] (j＜k≤V+1)}+a[i,j]　(1≤i≤F,1≤j≤V)

这样，得出的max{q[1,j]}(1≤j≤V)就是最大的美学值，算法的时间复杂度为$O(FV^2)$。

（3）对算法时间效率的改进。先来看一下这样两个状态的求解方法：

q[i,j]=max{q[i+1,k](j＜k≤V+1)}+a[i,j]　　(1≤i≤F,1≤j≤V)

q[i,j+1]=max{q[i+1,k] (j+1＜k≤V+1)}+a[i,j+1]　(1≤i≤F,1≤j+1≤V)

上面两状态中求max{q[i+1,k]}的过程进行了大量重复的比较。此时对状态的表示稍作修改，用数组t[i,j]=max{q[i,k](j≤k≤V+1)} (1≤i≤F,1≤j≤V)表示新的状态。经过修改后，因为q[i,j]=t[i+1,j+1]+a[i,j]，而t[i,j]=max{t[i,j+1],q[i,j]} (1≤i≤F,1≤j≤V)，所以得出新的状态转移方程：t[i,V+1]=-∞　　(1≤i≤F+1)

t[F,j]=max{t[F,j+1],b[F,j]}　　(1≤j≤V)

t[i,j]=max{t[i,j+1],t[i+1,j+1]+a[i,j]}　(1≤i≤F,1≤j≤V)

这样，得出的最大美学值为t[1,1]，新算法的时间复杂度为O(F×V)，而空间复杂度也为O(F×V)，完全可以满足1≤F≤V≤100的要求。下面给出这一问题的源程序。

```
{$A+,B-,D+,E+,F-,G-,I+,L+,N-,O-,P-,Q-,R-,S+,T-,V+,X+}
{$M 16384,0,655360}
program ex7-6; {花店橱窗布置问题}
const st1='flower.inp';          {输入文件名}
      st2='flower.out';          {输出文件名}
var f,v:integer;                 {f为花束的数量；v为花瓶的数量}
   b:array[1..100,1..100] of shortint;  {b[i,j]为第i束花放进第j个花瓶的美学值}
   t:array[1..101,1..101] of integer;  {t[i,j]为将第i到第f束花放进第j到第v个花瓶所可能得到的最大美学值}
procedure readp;                 {从文件中读入不同花束对应不同花瓶的美学值}
var f1:text;
   i,j:integer;
begin
   assign(f1,st1);reset(f1);
```

```
    readln(f1,f,v);
    for i:=1 to f do
      for j:=1 to v do
        read(f1,b[i,j]);
    close(f1);
end;
procedure main;          {用动态规划对问题求解}
var i,j:integer;
begin
    for i:=1 to f+1 do t[i,v+1]:=-9999;
    for j:=v downto 1 do
      if t[f,j+1]>b[f,j]
        then t[f,j]:=t[f,j+1]
        else t[f,j]:=b[f,j];   {设定动态规划的初始条件，其中-9999即表示负无穷}
    for i:=f-1 downto 1 do
      for j:=v downto 1 do begin
        t[i,j]:=t[i,j+1];
        if t[i+1,j+1]+b[i,j]>t[i,j] then
          t[i,j]:=t[i+1,j+1]+b[i,j];
      end;
end;
procedure print;          {将最佳美学效果和对应方案输出到文件}
var f1:text;
  i,j,p:integer; {i为当前需确定位置的花束编号，p为第i束花应插入的花瓶编号的最小值}
begin
    assign(f1,st2);rewrite(f1);
    writeln(f1,t[1,1]);
    p:=1;
    for i:=1 to f do begin        {用循环依次确定各束花应插入的花瓶}
      j:=p;
      while t[i,j]=t[i,p] do inc(j);
      write(f1,j-1,' ');p:=j;
    end;
    writeln(f1);
    close(f1);
end;
begin
    readp;
    main;
    print;
end.
```