



OLYMPIAD IN INFORMATICS

信息学奥赛试题是如何炼成的？

合肥信息学辅导站 张金苗



合肥OI人
www.hfoier.cn



引言

- 想当年国际赛IOI1994“数塔问题”可是难倒了一批当年的大牛，现在可是很多人学习动态规划第一个例子；全国联赛NOIP2006“能量项链”可是全国赛NOI1995“石子合并”与大学计算机教材上经典动态规划例子“矩阵相乘”的有机结合。
- 难道是当年的那些题目太“菜”了吗？当然不是，因为我们不能用现在的眼光看那些时过境迁的事物，青少年信息学奥赛也是在不断变化与发展中的，把握信息学奥赛题目变化与发展规律，就能够站在一个比较高的高度，解决好较难的问题。



- 同时，把握信息学奥赛题目的变化与发展，也是剖析、揣摩了出题者的想法和一般思路的过程，这对于学习者和教学者都是有益的，这又就是另一种角度的“**换位思考**”，也是更高层次的学习与教学。
- 以下通过一系列由浅入深、在各级各类信息学竞赛中出现的真题，阐述信息学奥林匹克竞赛“变化与发展”这个永恒主题，通过如下讨论，也不难看出，信息学奥赛试题是如何演变与炼就的。



Contents



题目原型

算法拓展

增加决策多样性

增加控制点

增加线程

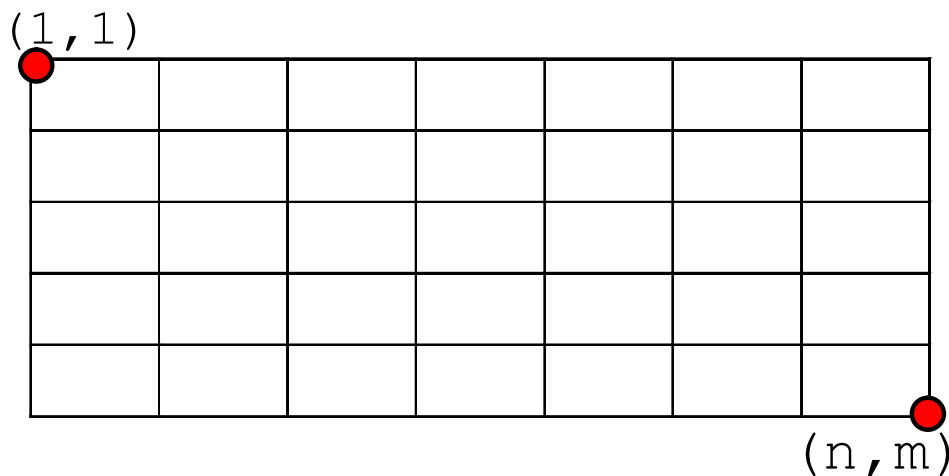
增加数学调料



一、题目原型

● [题1] 棋盘路径

- 有一个 $n*m$ 的棋盘，左上角为 $(1,1)$ ，右上角为 (n,m) 。有一颗棋子，初始状态为 $(1,1)$ ，该棋子只能向右走或者向下走，问改棋子从 $(1,1)$ 到 (n,m) 一共有几条路径？（中国象棋，走边）

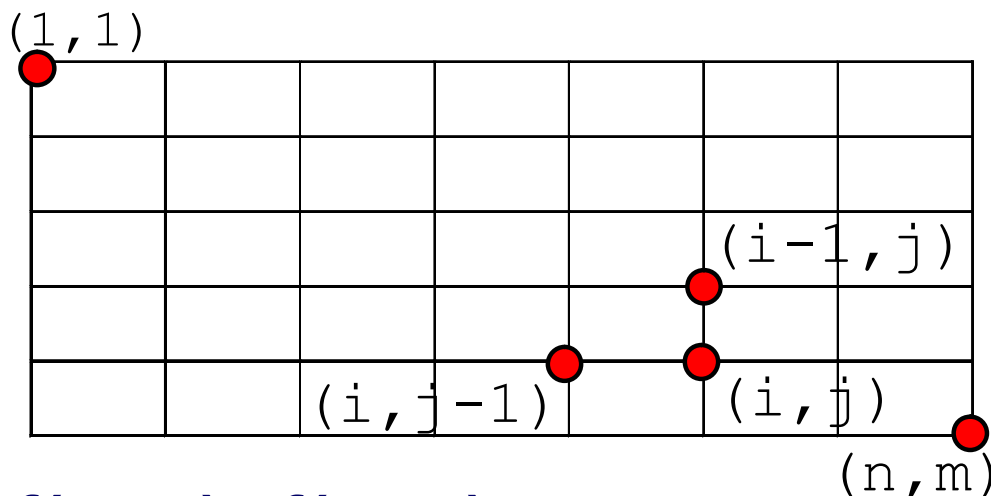


- 输入：两个整数 n 和 m
- 输出：一个数，路径总数。



[解题思路]

- 除左边界和上边界上的点的路径，为其上面点的路径同左边点路径之和。



- 递推公式: $f(i,j)=f(i-1,j)+f(i,j-1)$
- 边界条件: $f(1,1)=1$
- 程序框架
 - $f[1,1]:=1;$
 - For $i:=1$ to n do
 - for $j:=1$ to n do
 - if $i*j \neq 1$ then $f[i,j]:= f(i-1,j)+f(i,j-1);$



Contents



题目原型

算法拓展

增加决策多样性

增加控制点

增加线程

增加数学调料



二、算法拓展

- 上题就是一个单纯的二维递推；增加策略选择就成为简单动态规划题了。

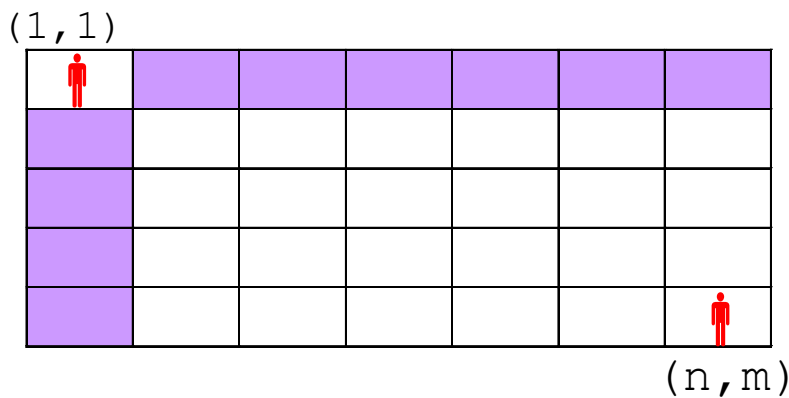
- [题2]最小伤害**

- 小可可站在一个 $N \times N$ 的方阵中最左上角的格子里。他可以从一个格子走到它右边和下边的格子里。每一个格子都有一个伤害值。他想在受伤害最小的情况下走到方阵的最右下角。
- 输入数据：第一行输入一个正整数 n 。以下 n 行描述该矩阵。矩阵中的数保证是不超过1000的正整数。
- 输出数据：输出最小伤害值。
- 样例输入：
 - 3
 - 1 3 3
 - 2 2 2
 - 3 1 2
- 样例输出：
 - 8
- 数据规模：
 - $n \leq 1000$



[解题思路]

- 很简单的DP，但需要注意边界的预处理。
- 状态转移方程：
 - $f(i,j)=\min\{f(i-1,j),f(i,j-1)\}+f(i,j)$;
- 对边界的不同处理方法方面，大致有两种解法（以下代码状态复用本身的值的数组，没有为状态单独开辟存储空间）：
 - 方法一：预先处理好左边界和上边界
 - `a:array[1..1000,1..1000] of longint;`



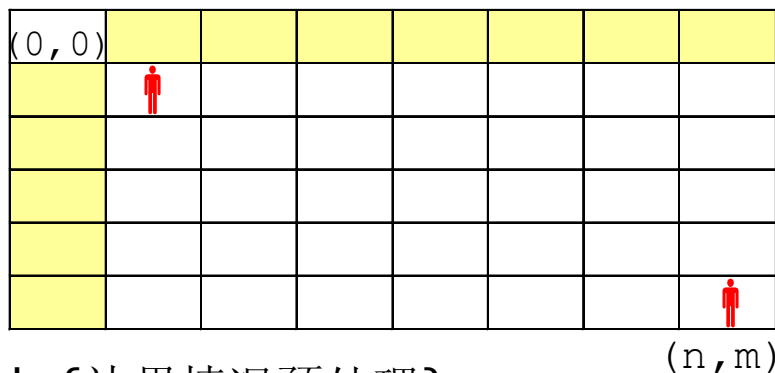


- for $i:=2$ to n do
- begin
- $a[i,1]:=a[i,1]+a[i-1,1]$; {左边界处理}
- $a[1,i]:=a[1,i]+a[1,i-1]$; {上边界处理}
- end;
- for $i:=2$ to n do {DP主体部分}
- for $j:=2$ to n do
- $a(i,j)=\min\{a(i-1,j),a(i,j-1)\}+a(i,j)$;



- 方法二：左边、上边各扩展一行，置上一个较大的数

- a:array[0..1000,0..1000] of longint;



- for i:=1 to n do{边界情况预处理}
 - begin
 - a[i,0]:=1000001;
 - a[0,i]:=1000001;
 - end;
 - for i:=1 to n do{DP主体部分}
 - for j:=1 to n do
 - if (i=1)and(j=1)then a[i,j]:=a[i,j]
 - else a[i,j]:=min(a[i-1,j],a[i,j-1])+a[i,j];



Contents

题目原型

算法拓展

增加决策多样性

增加控制点

增加线程

增加数学调料





三、增加决策多样性

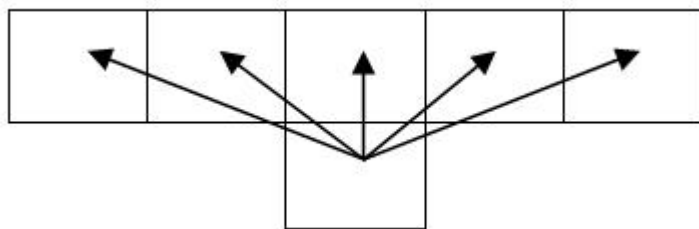
- 题2中的决策只是二选一，如果增加决策的难度，多选一，又会怎样呢？
- [题3] 取数
- 在一个 $n*m$ 的方格中（ m 为奇数），放置有 $n*m$ 个数，如下图($6*7$)所示：

16	4	3	12	6	0	3
4	-5	6	7	0	0	-2
6	0	-1	-2	3	6	8
5	3	4	0	0	-2	7
-1	7	4	0	7	-5	6
0	-1	3	4	12	4	2
			人			



[解题思路]

- 方格中间的下方有一人，此人可按照五个方向前进但不能越出方格：



- 人每走过一个方格必须取此方格中的数。小可可请你编程找到一条从底到顶的路径，使其数相加之和为最大。
- 输入：在输入文件中：
 - 第一行： n, m ($n, m \leq 1000$)；
 - 第二行： m 个数，数字 (< 10000) 之间用空格隔开
 - 第三行： m 个数
 -
 - 第 $n+1$ 行： m 个数
- 输出：取得的相加之和为最大的数



- 根据题意，本题的预处理比较好的方法是向左右各扩充两行（0行、-1行、1001行、1002行），因为是求最大的值，并且方格中可以有负数的情况，所以预处理需要置比较小的负数。由于是5选1，所以需要用一个循环来实现。
- for i:=1 to n do
 - for j:=1 to m do read(a[i,j]);
 - for i:=1 to m do f[1,i]:=a[1,i];
 - for i:=2 to n do
 - for j:=1 to m do
 - **for k:=j-2 to j+2 do f[i,j]:=max(f[i,j],a[i,j]+f[i-1,k]);**
 - max:=-maxlongint;
 - for i:=(m div 2+1)-2 to (m div 2+1)+2 do
 - if f[n,i]>max then max:=f[n,i];



Contents

题目原型

算法拓展

增加决策多样性

增加控制点

增加线程

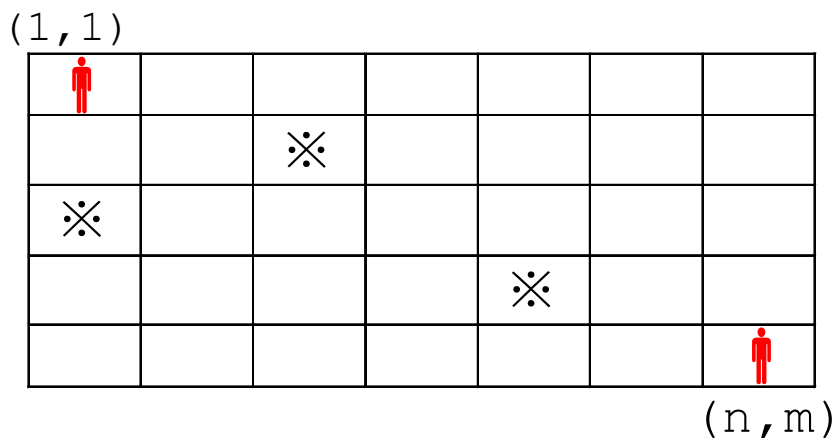
增加数学调料





四、增加控制点

- 如果在题2的基础上添加一些不能到达的控制点，又需要怎样处理呢？
- [题4] 最小花费
 - 现在有一个 $n \times m$ 的矩形棋盘，每个格子上面都有一个非负整数。你拥有一枚棋子，在游戏开始的时候，你的棋子位于左上角的方格内。游戏的规则很简单：每一次，你可以将棋子向右或向下移动一格，当棋子到达右下角的方格时，游戏结束。同时，你必须保证你的棋子通过的路径是花费最小的。一条路径的花费就是这条路径上所有格子上的数字的和。有一点需要说明的是，被标记为0的格子是不可以走到的。





[解题思路]

- 针对控制点的解决方法有两种（边界处理采用扩充边界，置较大的数来解决）：
- （一）通过判断“0”来解决
 - for i:=1 to n do
 - for j:=1 to m do
 - if (i>1)and(j>1)and(a[i,j] >0) then begin
 - if (a[i,j-1]=0)and(a[i-1,j]=0)then a[i,j]:=0;
 - if (a[i,j-1]=0)and(a[i-1,j]>0)then inc(a[i,j],a[i-1,j]);
 - if a[i,j-1]>0)and(a[i-1,j]=0)then inc(a[i,j],a[i,j-1]);
 - if (a[i,j-1]>0)and(a[i-1,j]>0)then
 - if a[i,j-1]<a[i-1,j] then inc(a[i,j],a[i,j-1]) else inc(a[i,j],a[i-1,j]);
 - end; {如果是a[1,1]或者是a[i,j]=0的情况，则无需操作，保留原值作为状态值}
- （二）同边界处理类似，将“0”的位置替换为一个较大值，使得其最终不可能被取到。
 - 这种处理方法，将控制点的处理同扩充边界的方法结合起来，程序实现会简单一些，值得推荐。



Contents

题目原型

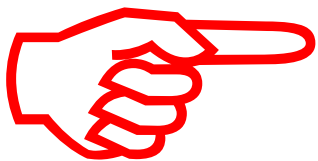
算法拓展

增加决策多样性

增加控制点

增加线程

增加数学调料





五、增加线程

● [题5] 方格取数

- 设有 $n*n$ 的方格图 ($N \leq 8$)，我们将其中的某些方格中填入正整数，而其他的方格中则放入数字 0。如下图所示（见样例）：

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

- 某人从图的左上角的 A 点出发，可以向下行走，也可以向右走，直到到达右下角的 B 点。在走过的路上，它可以取走方格中的数（取走后的方格中将变为数字 0）。此人从 A 点到 B 点共走两次，试找出 2 条这样的路径，使得取得的数之和最大。



[解题思路]

如图1所示样例，图2按最优路径走一次后，余下的两个数4和6就不可能同时取到了，而按图3中的非最优路径走一次后却能取得全部的数，原因在于两次走法之间的协作是十分重要的，而图2中的走法并不能考虑全局，因此这一算法只能算是“局部动归、整体贪心”而已。那么此时是否就无法采用动归策略呢？

1	5	2		4	
		7			
		2			
				8	
				9	
		6			

(图 1)

1	5	2		4	
		7			
		2			
				8	
				9	
		6			

(图 2)

1	5	2		4	
		7			
		2			
				8	
				9	
		6			

(图 3)



- 实际上本题也可以动态规划解决，只是状态转移方程更复杂。题2在只走一次的情况时，只需考虑一个人到达某个格子 (i,j) 的情况，得出 $s[i,j]=\max(s[i-1,j],s[i,j-1])+a[i,j]$ ，现在考虑两个人同时从起点出发，则需考虑两个人到达任意两个格子 $(i1,j1)$ 与 $(i2,j2)$ 的情况，显然要到达这两个格子，其前一状态必为 $(i1-1,j1),(i2-1,j2)$ ； $(i1-1,j1),(i2,j2-1)$ ； $(i1,j1-1),(i2-1,j2)$ ； $(i1,j1-1),(i2,j2-1)$ 四种情况之一，这样就需要用4维数组表示某一状态。
- 设 $T=\max(s[i1-1,j1,i2-1,j2],s[i1-1,j1,i2,j2-1],s[i1,j1-1,i2-1,j2],s[i1,j1-1,i2,j2-1])$,

$$S[i1,j1,i2,j2]=\begin{cases} 0 & (\text{当 } i1=0 \text{ 或 } j1=0 \text{ 或 } i2=0 \text{ 或 } j2=0) \\ T+a[i1,j1] & (\text{当 } i1,j1,i2,j2 \text{ 均不为零, 且 } i1=i2, j1=j2) \\ T+a[i1,j1]+a[i2,j2] & (\text{当 } i1,j1,i2,j2 \text{ 均不为零, 且 } (i1 \neq i2 \text{ 或 } j1 \neq j2)) \end{cases}$$



- **for i1:=1 to n do**
- **for j1:=1 to n do**
- **for i2:=1 to n do**
- **for j2:=1 to n do**
- **begin**
- m:=max(s[i1-1,j1,i2-1,j2],s[i1-1,j1,i2,j2-1],s[i1,j1-1,i2-1,j2],s[i1,j1-1,i2,j2-1]);
- if (i1=i2) and (j1=j2) then s[i1,j1,i2,j2]:=m+a[i1,j1]
- else s[i1,j1,i2,j2]:=m+a[i1,j1]+a[i2,j2];
- **end;**



Contents

题目原型

算法拓展

增加决策多样性

增加控制点

增加线程

增加数学调料





六、增加“数学佐料”

● [题6] 方格游戏

- 现在有一个 $n \times n$ 的正方形棋盘，每个格子上面都有一个非负整数。你拥有一枚棋子，在游戏开始的时候，你的棋子位于左上角的方格内。游戏的规则很简单：每一次，你可以将棋子向右或向下移动一格，当棋子到达右下角的方格时，游戏结束。同时，你必须保证你的棋子通过的路径是花费最小的。一条路径的花费就是这条路径上所有格子上的数字的乘积最后的连续的0的个数。有一点需要说明的是，被标记为0的格子是不可以走到的。
- 输入：第一行为一个整数 n （ $1 \leq n \leq 1000$ ），表示棋盘的大小。下面 n 行每行有 n 个整数，表示每个格子上的数字（均不超过1000000）。
- 我们保证所有的输入数据都至少存在一条左上角到右下角的路径。
- 输出：一个正整数，表示找到的最优路径的费用。



[解题思路]

- 本题有两个亮点：

- (1) “花费”的解读独辟蹊径：一条路径的花费就是这条路径上所有格子上的数字的乘积最后的连续的0的个数。实际上我们只要统计路径上所有格子上的数字的乘积有多少个因子10，由于 $10=5 \times 2$ ，因而只需统计计路径上所有格子上的数字有多少个因子5和2。2和5只有配对成功，才可以产生0，所以最终0的个数是由最少的2或者5决定了。添加如此“数学佐料”，是不是让“烹饪”后的“大餐”更有滋味呢？
- (2) 本题需要统计路径上所有格子上的数字有多少个因子5和2，那么这里“走两遍”问题5的“走两遍”是一回事吗？其实这里是满足“贪心”原则的，证明如下：假如找到某条路径上因子为5的个数最少，为 $S(5)$ ，某条路径上因子为2的个数最少，为 $S(2)$ ，我们选择 $S(2)$ ，因为在产生 $S(2)$ 这条路径上的 $S(5)$ 一定大于 $S(2)$ ；反之亦然。




- 通过以上分析，不难看出一道信息学奥赛试题，是在原型的基础上通过各种知识综合、能力拓展、包装修饰炼成的，问题总是在不断变化与发展中延伸的。通过以上启发，我们也可以将题5变形，题5中0的位置是可以走到的，如果修改为0的位置不可以走到，问题就变得很有意思了，控制点不仅第一遍中就存在，而且走过第一遍后，又会产生新的控制点，这将如何解决呢？留给诸位思考。

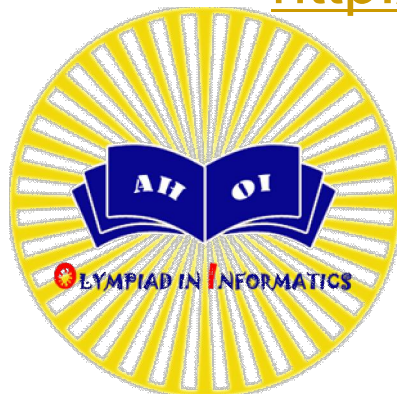


OLYMPIAD IN INFORMATICS

Thank You!

 master0551@126.com

<http://hi.baidu.com/china0551>



QQ在线

与我交谈

合肥OI人

www.hfoier.cn