NOI分区联赛 - 2000年第六届提高组试题解析

注意：解析和源程序均为OIBH站长刘汝佳所写，疏漏在所难免，但至少程序均通过了比赛时使用的测试数据，
所以还是可以一看。

第一题：

大家对正数进制的转换应该比较熟悉吧！（不会的看我的《循序渐进》）
负数进制一样。每次取的余数保证在0~-m-1之间。（例如m=-16,则余数应该在0~15）就可以直接输出。
所以用系统的“mod”运算符的时候必须注意检查是不是在该范围（可能在m+1~0），否则就调整。
调整的方法是：

```
if 余数<0 then
begin
  余数=余数-m;
  商=商+1;
end;
```

程序见附件。

第二题：
很明显的动态规划。
令d[i,j,k]为第i个数字到第j个数字加k个乘号所能够达到的最大值。
状态转移方程是：
d[i,j,k]=max{num[i,t]*d[t+1,j,k-1]}（枚举t，满足i<=t<=j-1）
注意到状态转移的时候总是使k减小（或不变）的，所以把k作为阶段来递推（节约空间）。
在每个状态中，1=j-i越来越小，所以从1=0递推到1=n
即：

```
for k:=1 to c do
  for 1:=0 to n do
    for i:=1 to n do
      递推d[i,j,k]
```

显然，用两个数组记录d[i,j,k]和d[i,j,k-1],就可以只用两维：d[i,j]
于是算法框架就是(请对照我的源程序)：

```
初始化d1[i,j]

for k:=1 to c do
begin
  for 1:=0 to n do
    for i:=1 to n do
      用d1[i,j](k-1阶段)递推d2[i,j]（k阶段）
  d1:=d2; {节省空间，因为递推的时候只与上个阶段有关，故只保留相邻两个阶段}
end;
```

高精度乘法的方法我不是用的模拟手算的过程（这个大家会做吧），而用了类似
多项式乘法的方法，因为我觉得这种写法很好记！(大家仔细看看我的mu1过程)

程序见附件。

第三题：
搜索。数据很小，因此回溯就可以了。程序先预处理,建立矩阵add[i,j]
即第j个串连在第i个串之后能增加的长度。0代表j不能增加i的后面。
一个需要注意的地方：
计算add[i,j]的时候要计算最大可能值！
例如：
ABABABAB和ABABABC就是5,不是1！
现在没有问题了吧。为了方便和直观，我采用递归实现回溯搜索。程序见附件。

第四题：
有同学搜索第一个人，拣了以后第二个人用动态规划，一定能得最优解，但时间效率不大高。
有同学采用贪心，即用动态规划算出第一个人最大能拣的数，再在剩下的数中用动态规划。
反例如下：
1 9 1
0 0 0
1 9 1
第一次是：
1->9->9->1
第二次是：1
和是21
但显然可以两次把所有的数拣完(22)。

本题是典型的多维动态规划，很象IOI93的第四题,另一个算法是网络流，很象IOI97第一题，
这里我只分析前者。这道题目的简单之处是阶段很好划分（对角线），这种方法我就不介绍了，
因为很多地方都有介绍。这里讲一种怪一点的动态规划^_^

容易想到的思路是：
令d[x1,y1,x2,y2]表示甲在x1,y1,乙在x2,y2以后（包括取走(x1,y1)）的过程中可以获得的最大和。
则d[1,1,1,1]就是原问题的解。

但是是否能取数和另一个人是否事先已经到过该格子有关，我们需要设计一种走的方法，使得只根据x1,y1,
x2,y2就能判断一些关键的格子是否已经到达过。这样，问题才具有无后效性。

为此，我们把路线作如下处理：
1)当两个人路线有交叉的时候，改成等效的，不交叉的。
如下图，1代表第一个人，2代表第二个人。X代表相遇点。

```
X111
2  1
222X2
   12
   12
   1X1
    2X
```

变成：
```
X222
1  2
111X2
   12
   12
   1X2
    1X
```

反正让2走右边就行了。

2)现在1在第y1列，2在第y2列，让1和2分别向右走,到达yy1和yy2列，然后向下走一格
这样如果yy1 < y2,便是分别取走第y1~yy1,y2~yy2列数，否则路线有重复，就取走y1~yy2的数。
为了方便连续取数，我用了一个sum[x,y1,y2]的数组，就是第x行的y1~y2的数。
请看我的程序中的相应部分。

这样，所有的走法都可以转换成上述的，具有无后效性的结构了。

由于递推是从d[x1,y1,x2,y2]到d[x1+1,y1',x2+1,y2'],而总有x1=x2=x,所以可以把状态节省为：d[y1,y2]
而把x(当前行)作为阶段来递推：

```
for x:=n-1 downto 1 do
begin
  for y1:=1 to n do
    for y2:=y1 to n do
      枚举y1'和y2'作为新的y1和y2,注意保证y1' >= y1, y2' >= y2(题目规定), y1' <= y2'(刚才的分析),递推d[y1,y2]
  d1:=d2; {只记录相邻两个状态}
end;
```

边界是什么呢？当然是从第n列的值了，就是sum[n,y1,n](从y1取到n)

说了这么多，真不知道我说清楚了没有（ 好象是没有:-P ），如果有什么不明确的地方，请大家在论坛上提出来。
一句话，就是两个人一起，一行一行往下走，每一行可以一次向右走若干步，但是要保证2在1的右边。

注意最后输出的是d2[1,1]。如果输出d1[1,1],n=1会得到0。（为什么？自己想啊）

现在程序就不难写了吧。程序见附件：

测试数据见附件。