以文本方式查看主题

- **株洲教育科研论坛** (http://bbs.zzer.org/index.asp)
-- **『信息学奥赛』** (http://bbs.zzer.org/list.asp?boardid=9)
---- **NOIP2004 普及组解题报告** (http://bbs.zzer.org/dispbbs.asp?boardid=9&id=348)

-- 作者：xielj
-- 发布时间：2005-1-8 0:00:24

-- NOIP2004 普及组解题报告

第 1 题 不高兴的津津

【问题描述】
津津上初中了。妈妈认为津津应该更加用功学习，所以津津除了上学之外，还要参加妈妈为她报名的各科复习班。另外每周妈妈还会送她去学习朗诵、舞蹈和钢琴。但是津津如果一天上课超过八个小时就会不高兴，而且上得越久就会越不高兴。假设津津不会因为其它事不高兴，并且她的不高兴不会持续到第二天。请你帮忙检查一下津津下周的日程安排，看看下周她会不会不高兴；如果会的话，哪天最不高兴。
【输入文件】
输入文件 unhappy.in 包括七行数据，分别表示周一到周日的日程安排。每行包括两个小于 10 的非负整数，用空格隔开，分别表示津津在学校上课的时间和妈妈安排她上课的时间。
【输出文件】
输出文件 unhappy.out 包括一行，这一行只包含一个数字。如果不会不高兴则输出 0，如果会则输出最不高兴的是周几（用 1, 2, 3, 4, 5, 6, 7 分别表示周一，周二，周三，周四，周五，周六，周日）。如果有两天或两天以上不高兴的程度相当，则输出时间最靠前的一天。
【样例输入】
5 3
6 2
7 2
5 3
5 4
0 4
0 6
【样例输出】
3

解题方法：采用模拟。
当天不高兴程度=当天学校上课时间+当天妈妈安排上课时间
注意点 1：题目的要求的是哪一天最不高兴，不是求不高兴程度。

2：上课时间>8 才不高兴，若<=8 就不会不高兴。

3：满足(上课时间>8 条件)前提的数据中取时间最长的，

若最高存在多个则取靠前的天数。

4：若没有一天达到(上课时间>8 条件)，就输出 0。

解题程序：

```
program unhappy;
const f1=\'unhappy.in\'; f2=\'unhappy.out\';
var i,a,b,c,d:shortint；
begin
  assign (input,f1);
  assign (output,f2);
  reset (input);
  rewrite (output);
    d:=0; c:=0;                    初始设置。
    for i:=1 to 7 do               星期一到星期日。
     begin
       readln (input,a,b);         读入当天学校和妈妈安排的上课时间。
       if (a+b>8)and(a+b>c) then    学校和妈妈安排的上课时间>8，且>原有最多上课时间。
        begin
          c:=a+b;                  记录最不高兴那天的上课时间。
          d:=i;                    记录哪天最不高兴。
        end;
     end;
    writeln (output,d);            输出是第几天最不高兴。
  close (input);
  close (output);
end.
```

-- 作者：xielj

-- 发布时间：2005-1-8 0:10:57

--

第二题　　花生采摘

问题描述】

鲁宾逊先生有一只宠物猴，名叫多多。这天，他们两个正沿着乡间小路散步，突然发现路边的告示牌上贴着一张小小的纸条：“欢迎免费品尝我种的花生！——熊字”。

鲁宾逊先生和多多都很开心，因为花生正是他们的最爱。在告示牌背后，路边真的有一块花生田，花生植株整齐地排列成矩形网格（如图 1）。有经验的多多一眼就能看出，每棵花生植株下的花生有多少。为了训练多多的算术，鲁宾逊先生说：“你先找出花生最多的植株，去采摘它的花生；然后再找出剩下的植株里花生最多的，去采摘它的花生；依此类推，不过你一定要在我限定的时间内回到路边。”

我们假定多多在每个单位时间内，可以做下列四件事情中的一件：
1) 从路边跳到最靠近路边（即第一行）的某棵花生植株；
2) 从一棵植株跳到前后左右与之相邻的另一棵植株；
3) 采摘一棵植株下的花生；
4) 从最靠近路边（即第一行）的某棵花生植株跳回路边。
现在给定一块花生田的大小和花生的分布，请问在限定时间内，多多最多可以采到多少个花生？注意可能只有部分植株下面长有花生，假设这些植株下的花生个数各不相同。
例如在图 2 所示的花生田里，只有位于(2, 5), (3, 7), (4, 2), (5, 4)的植株下长有花生，个数分别为 13, 7, 15, 9。沿着图示的路线，多多在 21 个单位时间内，最多可以采到 37 个花生。

【输入文件】
输入文件 peanuts.in 的第一行包括三个整数，M, N 和 K，用空格隔开；表示花生田的大小为 M * N（1 <= M, N <= 20），多多采花生的限定时间为 K（0 <= K <= 1000）个单位时间。接下来的 M 行，每行包括 N 个非负整数，也用空格隔开；第 i + 1 行的第 j 个整数 Pij（0 <= Pij <= 500）表示花生田里植株(i, j)下花生的数目，0 表示该植株下没有花生。
【输出文件】
输出文件 peanuts.out 包括一行，这一行只包含一个整数，即在限定时间内，多多最多可以采到花生的个数。
【样例输入 1】
6 7 21
0 0 0 0 0 0 0
0 0 0 0 13 0 0
0 0 0 0 0 0 7
0 15 0 0 0 0 0
0 0 0 9 0 0 0
0 0 0 0 0 0 0
【样例输出 1】
37
【样例输入 2】
6 7 20
0 0 0 0 0 0 0
0 0 0 0 13 0 0
0 0 0 0 0 0 7
0 15 0 0 0 0 0
0 0 0 9 0 0 0
0 0 0 0 0 0 0
【样例输出 2】
28
解题方法：“你先找出花生最多的植株，去采摘它的花生；然后再找出剩下的植株里花生最多的，去采摘它的花生；依此类推，不过你一定要在我限定的时间内回到路边。”题目中的这个条件已经暗示我们

要采用贪心算法，故先对数据进行从大到小的排序。而且必须在规定的时间内回到路边(用模拟)。由于去摘花生必须从路边进入花生田和从花生田出来，所以我们可以先减去 2 个单位时间。当然进入哪一列的花生田由包含最多花生的植株所在列决定，因为要采用贪心嘛。即出发点已经在第一行的 X 列中。由于必须在规定时间内回到路边，我们在前往目标前要先判断(去采摘目标花生的时间)+(采摘那目标花生所用的 1 单位时间)+(从目标所在地往第一行的时间)<=(剩下的单位时间)。若条件不满足就停止，若满足就继续采摘。起点到目标的距离=ABS(起点所在行-目标所在行)+ABS(起点所在列-目标所在列)。

解题程序：

```
program peanuts;
const f1=\'peanuts10.in\'; f2=\'peanuts.out\';
var c,i,j,m,n,k:integer;  M 行数 N 列数 K 单位时间
    x:array[1..20,1..20]of integer;读入的数据
    y,z:array[1..400]of integer;按数量大小排列的花生植株的行和列。Y 为行 Z 为列
    sum:longint;  总采摘的花生数量
procedure xp(var a,b:integer); 交换 A 和 B
begin
  c:=a;
  a:=b;
  b:=c;
end;
begin
  assign (input,f1);
  assign (output,f2);
  reset (input);
  rewrite (output);
  readln (input,m,n,k); 读入行、列、时间单位
  for i:=1 to m do
   for j:=1 to n do
    begin
      read (input,x[i,j]); 读入数据
      y[(i-1)*n+j]:=i;   记录数据所在行
      z[(i-1)*n+j]:=j;   记录数据所在列
    end;
  for i:=1 to m*n do
   for j:=i+1 to m*n do
   if x[y,z]<x[y[j],z[j]] then 因数据不大，用冒泡排序，将花生数量按从大到小排列(排序时是交换其的行和列，不是排序数量)
    begin
      xp(y,y[j]); 花生数量比 I 大的 J，交换 I 和 J 的行
      xp(z,z[j]); 花生数量比 I 大的 J，交换 I 和 J 的列
    end;
  k:=k-2; c:=1; sum:=0; i:=1; j:=z[1]; 减去 2 个单位时间 C 表示目标是第 X 大的花生植株 I 表示当前所在的行 J 标示当前所在的列
  while (abs(y[c]-i)+abs(z[c]-j)+1+(y[c]-1)<=k) do 判断现在的单位时间是否足够到目标，并采摘，且回
```

到第一行。

```
    begin
     k:=k-(abs(y[c]-i)+abs(z[c]-j)+1); 单位时间减去到目标所用时间及采摘花生的 1 单位时间。
     if x[y[c],z[c]]=0 then break; 目标的植株的花生数量为 0 时，再摘也没意思了，就退出
     sum:=sum+x[y[c],z[c]]; 到达目标，累加采摘数量
     i:=y[c]; 当前所在行更替为目标所在行 就是移动到目标啦，就要以目标为起点
     j:=z[c]; 当前所在列更替为目标所在列
     inc(c); 目标更替为下一棵植株
     if c>m*n then break; 目标>总棵数———退出
    end;
  writeln (output,sum); 输出总采摘数量
  close (input);
  close (output);
end.
```

-- 作者：xielj
-- 发布时间：2005-1-8 0:12:42

--
第三题　　FBI 树

【问题描述】
我们可以把由“0”和“1”组成的字符串分为三类：全“0”串称为 B 串，全“1”串称为 I 串，既含“0”又含“1”的串则称为 F 串。
FBI 树是一种二叉树，它的结点类型也包括 F 结点，B 结点和 I 结点三种。由一个长度为 2N 的“01”串 S 可以构造出一棵 FBI 树 T，递归的构造方法如下：
1)T 的根结点为 R，其类型与串 S 的类型相同 3) ；
2)若串 S 的长度大于 1，将串 S 从中间分开，分为等长的左右子串 S1 和 S2；由左子串 S1 构造 R 的左子树 T1，7) 由右子串 S2 构造 R 的右子树 T2。
现在给定一个长度为 2N 的“01”串，请用上述构造方法构造出一棵 FBI 树，并输出它的后序遍历序列。
【输入文件】
输入文件 fbi.in 的第一行是一个整数 N（0 <= N <= 10），第二行是一个长度为 2N 的“01”串。
【输出文件】
输出文件 fbi.out 包括一行，这一行只包含一个字符串，即 FBI 树的后序遍历序列。
【样例输入】
3
10001011
【样例输出】
IBFBBBFIBFIIIFF
【数据规模】
对于 40%的数据，N <= 2；

对于全部的数据，N <= 10。

解题方法： 由于 2 的 10 次方=1024，若选择字符串就回超出，故选择用数组，1 表示 B 2 表示 I 3 表示 F。先将读入的\'0\'转换为 1，将\'1\'转换为 2。

以输入数据为第一层开始构造完整的 FBI 树，遵循此规则

1：两个数据结合，若其中有 F(表示为 3)，结合结果为 F(表示为 3)

2：两个数据结合，其中不存在 F(表示为 3)

(1)两数据相同，结果为此数据

(2)两数据不同，结果为 F(表示为 3)

构造好 FBI 数后，以最上层开始进行后序遍历(先左子树再右子树最后根)。一边遍历一边输出

解题程序：

```
program fbi;
const f1=\'fbi.in\'; f2=\'fbi.out\';
      s=\'BIF\';
var i,j,k,n:integer;
   x:array[1..11,1..1024]of 1..3; 储存完整的 FBI 树，X[1]为 FBI 树的最底层，依次类推。
   z:array[1..11]of integer;     储存完整的 FBI 树长度，Z[1]为 FBI 树的最底层的长度，依次类推。
   e:char;                       用来读入数据。
procedure xp(c,b:integer);       后序遍历 FBI 树，C 为当前层次，B 为当前层次的第 X 个(就是列数啦)。
begin
  if c>1 then 1 是最底层，只要不是最底层就可以继续遍历下一层。FBI 树是满 2 叉树，只要当前节点不在最底层，它肯定有左右子树。
   begin
    xp(c-1,b*2-1);              遍历当前节点的左子树。
    xp(c-1,b*2);                遍历当前节点的右子树。
   end;
  write (output,s[x[c,b]]);     输出当前节点。
end;
begin
 assign (input,f1);
 assign (output,f2);
 reset (input);
 rewrite (output);
 readln (input,n);
 for i:=1 to n+1 do z:=0;     初始化，将层长度清为 0。
 while not seekeof do          只要文件还有数据就继续读入。
  begin
   read (input,e);             读入临时字符 E。
   inc(z[1]);                  第 1 层(最底层)长度+1，因为又有一个新数据读入。
    j:=ord(e)-48;              将字符转化为数字 J。
```

```
  if j=0 then x[1,z[1]]:=1 else x[1,z[1]]:=2; 例如：读入 1 个 0，就是读入 1 个 B，但程序中 B 表示为 1，故步读入 0 记录为 1。
  end;                                        读入 1 个 1，就是读入 1 个 I，但程序中 I 表示为 2，故步读入 1 记录为 2。
 i:=2;                          第一层(最底层)从文件读入了，现在要开始构造完整的 FBI 树，初始目标是第 2 层。
     while z[i-1]>1 do 只要底下一层长度>2 就可以构造新的一层，因为是 FBI 树是 2 叉树嘛。
  begin
   for j:=1 to z[i-1] div 2 do   以目标层的下一层构造目标层，目标层共可以得到其下一层长度/2 的 FBI 树。
    begin
     inc (z); 目标层长度+1
     if (x[i-1,j*2-1]=3)or(x[i-1,j*2]=3) then x[i,z]:=3 两个数据中存在 F(表示为 3)，构造成的数据就是 F。
      else begin
       if x[i-1,j*2-1]=x[i-1,j*2] then x[i,z]:=x[i-1,j*2] 两个数据相同，构造数据就是这两个数据的值。
        else x[i,z]:=3; 两个数据不同，构造数据为 F。
       end;
    end;
   inc (i);   目标层构造完毕，将目标向上移一层。
  end;
  xp(n+1,1);  以最顶层的第 1 列开始遍历。
 close (input);
 close (output);
end.
```

-- 作者：xielj
-- 发布时间：2005-1-8 0:13:16

--
第四题　火星人

【问题描述】
人类终于登上了火星的土地并且见到了神秘的火星人。人类和火星人都无法理解对方的语言，但是我们的科学家发明了一种用数字交流的方法。这种交流方法是这样的，首先，火星人把一个非常大的数字告诉人类科学家，科学家破解这个数字的含义后，再把一个很小的数字加到这个大数上面，把结果告诉火星人，作为人类的回答。
火星人用一种非常简单的方式来表示数字——掰手指。火星人只有一只手，但这只手上有成千上万的手指，这些手指排成一列，分别编号为 1，2，3……。火星人的任意两根手指都能随意交换位置，他们就是通过这方法计数的。
一个火星人用一个人类的手演示了如何用手指计数。如果把五根手指——拇指、食指、中指、无名指和小指分别编号为 1，2，3，4 和 5，当它们按正常顺序排列时，形成了 5 位数 12345，当你交换无名指和小指的位置时，会形成 5 位数 12354，当你把五个手指的顺序完全颠倒时，会形成 54321，在所有能

够形成的 120 个 5 位数中，12345 最小，它表示 1；12354 第二小，它表示 2；54321 最大，它表示 120。下表展示了只有 3 根手指时能够形成的 6 个 3 位数和它们代表的数字：

三进制数 123 132 213 231 312 321
代表的数字 1 2 3 4 5 6

现在你有幸成为了第一个和火星人交流的地球人。一个火星人会让你看他的手指，科学家会告诉你要加上去的很小的数。你的任务是，把火星人用手指表示的数与科学家告诉你的数相加，并根据相加的结果改变火星人手指的排列顺序。输入数据保证这个结果不会超出火星人手指能表示的范围。

【输入文件】

输入文件 martian.in 包括三行，第一行有一个正整数 N，表示火星人手指的数目（1 <= N <= 10000）。第二行是一个正整数 M，表示要加上去的小整数（1 <= M <= 100）。下一行是 1 到 N 这 N 个整数的一个排列，用空格隔开，表示火星人手指的排列顺序。

【输出文件】

输出文件 martian.out 只有一行，这一行含有 N 个整数，表示改变后的火星人手指的排列顺序。每两个相邻的数中间用一个空格分开，不能有多余的空格。

【样例输入】

5
3
1 2 3 4 5

【样例输出】

1 2 4 5 3

【数据规模】

对于 30%的数据，N<=15；
对于 60%的数据，N<=50；
对于全部的数据，N<=10000；

解题方法：

看到题目，我发现会发现科学家要我们加上的数(1-100)并不大，而火星人的手指数目却很大(1-10000)。发现其实(大部分)运算不超过火星人最后 5 手指的运算，故选择进行排列搜索。经分析后发现每次运算(共 M 次)每次从最后一位为起点，向后搜索比它大的数中最小的数，若找不到半个就将搜索起点向前移一位。经过 M 次这样的运算后就能找到结果。

例如：1 2 3 4 5 为火星人手指排列顺序 此时 M=3

1 2 3 4 5

搜索到 4(第 4 位)时发现后面 5 比它大(且是比它大中最小的)，4<-->5，将第 4 位以后的数从小到大排列

1 2 3 5 4

搜索到 3(第 3 位)时发现后面 4 比它大(且是比它大中最小的)，3<-->4，将第 3 位以后的数从小到大排列
 1 2 4 3 5
搜索到 3(第 4 位)时发现后面 5 比它大(且是比它大中最小的)，3<-->5，将第 3 位以后的数从小到大排列
 1 2 4 5 3 结果找到啦
解题程序

```
program martian;
const f1=\'martian.in\'; f2=\'martian.out\';
var  x:array[1..10000]of integer; X 为火星人手指排列顺序
     n,m,i,j,k,c,d,a,b:integer;
     bbs:boolean;
procedure xp(var a,b:integer);  交换 A 和 B
begin
  k:=a;
  a:=b;
  b:=k;
end;
begin
   assign (input,f1);
   assign (output,f2);
   reset (input);
   rewrite (output);
    readln (input,n,m);  读入数据，N 为火星人手指数目 M 为科学家要你加上的数。
    for i:=1 to n do read (input,x);  读入火星人手指排列顺序。
    for i:=1 to m do 一共加 M 次，就是进行 M 次转换。
     begin
      j:=n; bbs:=true; J=N 是以最后一位为搜索起点，BBS=TRUE 表明目标尚未找到。
       while bbs=true do 找的目标是：以 起点+1 到 N 中比 X[起点]大的数中最小的一个。
        begin
          c:=30000; 初始设置，只要>10000 就行了，火星人可没有第 10001 个手指呀。
          for k:=j+1 to n do
           if(x[k]>x[j])and(x[k]<c) then 判断是不是比 X[起点]大，且是满足条件中最小的一个。
            begin
              c:=x[k];  记录此数(目标)的大小，以便进行比较。
              d:=k;     记录此数(目标)在排列中的位置。
            end;
          if c<>30000 then bbs:=false else dec(j);  C<>30000 时 表明已经找到这个数。
        end;           (接上一行)ELSE 表明找不到，就转向前一位，以前一位为起点向后搜索。
       xp(x[j],x[d]); 将找到目标时的搜索起点与找到的目标对换。
       for a:=j+1 to n do 将找到目标是的搜索起点后面的数从小到大排列。
         for b:=a+1 to n do if x<x[a] then xp(x,x[a  end;
    for i:=1 to n-1 do write (output,x,\' \'); 输出每两个相邻的数中间用一个空格分开。
    write (output,x[n]); 分 2 次输出是为了解决最后不多输一个空格。
  close (input);
```

```
  close (output);
end.
```

-- 作者：xielj
-- 发布时间：2005-1-8 0:15:09

--
【解题报告】
给出一组置换，快速生成下一个置换
解法 1：
枚举所有可能的排列，对这些排列从大到小排序，然后找到目标排列。代价为 O(N!log(N!))，能处理的数据规模范围是 N<=10。
解法 2：
利用递归将排列变换成对应的数字，时间代价是 N*N，能处理的数据规模范围是 N<=20。
解法 3：
在二的基础上采用高精度，时间代价约为 N*N*N，能处理的数据规模范围是 N<=50。
解法 4：
方法是直接生成下一个置换。我们用 a1,a2,…an 表示序列中的数。如果目前的序列是：
1 4 6 2 9 5 8 7 3
那么下一个置换是：
1 4 6 2 9 7 3 5 8
可以看到 5 之前的数不用动，通过分析可以发现，第一个要置换的数是满足 ai<A(I+1)的最靠右的数，因为它后面的数已经是降序了，交换不会产生更大的置换。
而被换过来的 7 是 5 后面比 5 大的数中最小的一个，然后其他的数从小到大排序就可以生成下一个置换。
反复应用这种贪心启发的时间代价是 M*N*logN。能处理的数据规模范围是 N<=10000。
总结四种解法如下：
解法 方法或要点 时间复杂度 N 的范围 可过数据
解法 1 穷举 O(N!log(N!)) 10 20%
解法 2 递归 N*N 20 30%-40%
解法 3 递归 + 高精度 N*N*N 50 60%
解法 4 贪心 M*N*logN 10000 100%

【数据说明】
所有数据时限均为 1 秒。

ID N 目的
1 5 输入样例
2 3
3 15
4 20 递归的上限
5 20

6 50 高精度的上限

7 100

8 500

9 1000

10 10000

第四题官方的解题报告出来了。
