CCF 全国信息学奥林匹克联赛（NOIP2013）复赛

普及组

（请选手务必仔细阅读本页内容）

一．题目概况

中文题目名称	计数问题	表达式求值	小朋友的数字	车站分级
英文题目与子目录名	count	expr	number	level
可执行文件名	count	expr	number	level
输入文件名	count.in	expr.in	number.in	level.in
输出文件名	count.out	expr.out	number.out	level.out
每个测试点时限	1 秒	1 秒	1 秒	1 秒
测试点数目	10	10	10	10
每个测试点分值	10	10	10	10
附加样例文件	有	有	有	有
结果比较方式	全文比较（过滤行末空格及文末回车）			
题目类型	传统	传统	传统	传统
运行内存上限	128M	128M	128M	128M

二．提交源程序文件名

对于 C++语言	count.cpp	expr.cpp	number.cpp	level.cpp
对于 C 语言	count.c	expr.c	number.c	level.c
对于 pascal 语言	count.pas	expr.pas	number.pas	level.pas

三．编译命令（不包含任何优化开关）

对于 C++语言	g++ -o count count.cpp -lm	g++ -o expr expr.cpp –lm	g++ -o number number.cpp -lm	g++ -o level level.cpp -lm
对于 C 语言	gcc -o count count.c -lm	gcc -o expr expr.c -lm	gcc-o number number.c -lm	gcc -o level level.c -lm
对于 pascal 语言	fpc count.pas	fpc expr.pas	fpc number.pas	fpc level.pas

注意事项：

1、文件名（程序名和输入输出文件名）必须使用英文小写。

2、C/C++中函数 main()的返回值类型必须是 int，程序正常结束时的返回值必须是 0。

3、全国统一评测时采用的机器配置为：CPU AMD Athlon(tm) 64x2 Dual Core CPU 5200+，2.71GHz，内存 2G，上述时限以此配置为准。

4、只提供 Linux 格式附加样例文件。

5、特别提醒：评测在 NOI Linux 下进行。

1．记数问题

(count.cpp/c/pas)

【问题描述】

试计算在区间 1 到 n 的所有整数中，数字 x（$0 \le x \le 9$）共出现了多少次？例如，在 1 到 11 中，即在 1、2、3、4、5、6、7、8、9、10、11 中，数字 1 出现了 4 次。

【输入】

输入文件名为 count.in。

输入共 1 行，包含 2 个整数 n、x，之间用一个空格隔开。

【输出】

输出文件名为 count.out。

输出共 1 行，包含一个整数，表示 x 出现的次数。

【输入输出样例】

count.in	count.out
11 1	4

【数据说明】

对于 100%的数据，$1 \le n \le 1{,}000{,}000$，$0 \le x \le 9$。

2．表达式求值

(expr.cpp/c/pas)

【问题描述】

给定一个只包含加法和乘法的算术表达式，请你编程计算表达式的值。

【输入】

输入文件为 expr.in。

输入仅有一行，为需要你计算的表达式，表达式中只包含数字、加法运算符“+”和乘法运算符“*”，且没有括号，所有参与运算的数字均为 0 到 2^{31}-1 之间的整数。输入数据保证这一行只有 0~ 9、+、*这 12 种字符。

【输出】

输出文件名为 expr.out。

输出只有一行，包含一个整数，表示这个表达式的值。**注意：当答案长度多于 4 位时，请只输出最后 4 位，前导 0 不输出。**

【输入输出样例 1】

expr.in	expr.out
1+1*3+4	8

【输入输出样例 2】

expr.in	expr.out
1+1234567890*1	7891

【输入输出样例 3】

expr.in	expr.out
1+1000000003*1	4

【输入输出样例说明】

样例 1 计算的结果为 8，直接输出 8。

样例 2 计算的结果为 1234567891，输出后 4 位，即 7891。

样例 3 计算的结果为 1000000004，输出后 4 位，即 4。

【数据范围】

对于 30%的数据，0≤表达式中加法运算符和乘法运算符的总数≤100；

对于 80%的数据，0≤表达式中加法运算符和乘法运算符的总数≤1000；

对于 100%的数据，0≤表达式中加法运算符和乘法运算符的总数≤100000。

3．小朋友的数字

(number.cpp/c/pas)

【问题描述】

有 n 个小朋友排成一列。每个小朋友手上都有一个数字，这个数字可正可负。规定每个小朋友的特征值等于排在他前面（包括他本人）的小朋友中连续若干个（最少有一个）小朋友手上的数字之和的最大值。

作为这些小朋友的老师，你需要给每个小朋友一个分数，分数是这样规定的：第一个小朋友的分数是他的特征值，其它小朋友的分数为排在他前面的所有小朋友中（不包括他本人），小朋友分数加上其特征值的最大值。

请计算所有小朋友分数的最大值，**输出时保持最大值的符号，将其绝对值对 p 取模后输出。**

【输入】

输入文件为 number.in。

第一行包含两个正整数 n、p，之间用一个空格隔开。

第二行包含 n 个数，每两个整数之间用一个空格隔开，表示每个小朋友手上的数字。

【输出】

输出文件名为 number.out。

输出只有一行，包含一个整数，表示最大分数对 p 取模的结果。

【输入输出样例 1】

number.in	number.out
5 997 1 2 3 4 5	21

【输入输出样例说明】

小朋友的特征值分别为 1、3、6、10、15，分数分别为 1、2、5、11、21，最大值 21 对 997 的模是 21。

【输入输出样例 2】

number.in	number.out
5 7 -1 -1 -1 -1 -1	-1

【输入输出样例说明】

小朋友的特征值分别为-1、-1、-1、-1、-1，分数分别为-1、-2、-2、-2、-2，最大值-1 对 7 的模为-1，输出-1。

【数据范围】

对于 50%的数据，$1 \le n \le 1{,}000$，$1 \le p \le 1{,}000$所有数字的绝对值不超过 1000；

对于 100%的数据，$1 \le n \le 1{,}000{,}000$，$1 \le p \le 10^9$，其他数字的绝对值均不超过 10^9。

4．车站分级

(level.cpp/c/pas)

【问题描述】

一条单向的铁路线上，依次有编号为 1, 2, …, n 的 n 个火车站。每个火车站都有一个级别，最低为 1 级。现有若干趟车次在这条线路上行驶，每一趟都满足如下要求：如果这趟车次停靠了火车站 x，则始发站、终点站之间所有级别大于等于火车站 x 的都必须停靠。（注意：起始站和终点站自然也算作事先已知需要停靠的站点）

例如，下表是 5 趟车次的运行情况。其中，前 4 趟车次均满足要求，而第 5 趟车次由于停靠了 3 号火车站（2 级）却未停靠途经的 6 号火车站（亦为 2 级）而不满足要求。

车站编号	1		2		3		4		5		6		7		8		9
车站级别 / 车次	3		1		2		1		3		2		1		1		3
1	始	→	→	→	停	→	→	→	停	→	终						
2					始	→	→	→	停	→	终						
3	始	→	→	→	→	→	→	→	停	→	→	→	→	→	→	→	终
4							始	→	停	→	停	→	停	→	停	→	终
5					始	→	→	→	停	→	→	→	→	→	→	→	终

现有 m 趟车次的运行情况（全部满足要求），试推算这 n 个火车站至少分为几个不同的级别。

【输入】

输入文件为 level.in。

第一行包含 2 个正整数 n, m，用一个空格隔开。

第 $i+1$ 行（$1 \le i \le m$）中，首先是一个正整数 s_i（$2 \le s_i \le n$），表示第 i 趟车次有 s_i 个停靠站；接下来有 s_i 个正整数，表示所有停靠站的编号，从小到大排列。每两个数之间用一个空格隔开。输入保证所有的车次都满足要求。

【输出】

输出文件为 level.out。

输出只有一行，包含一个正整数，即 n 个火车站最少划分的级别数。

【输入输出样例】

level.in	level.out
9 2 4 1 3 5 6 3 3 5 6	2
9 3 4 1 3 5 6 3 3 5 6 3 1 5 9	3

【数据范围】

对于 20%的数据，$1 \le n, m \le 10$；

对于 50%的数据，$1 \le n, m \le 100$；

对于 100%的数据，$1 \le n, m \le 1000$。

NOIp2013 普及组
解题报告

By 绍兴文理学院附中 任轩笛

计数问题

（count.pas/c/cpp）

Time Limit:1000Ms Memory Limit:131072K

算法一

这是一题送分题。

初看这道题，你想到了什么？对了，NOIp2010 普及组《数字统计》。几乎就是一模一样的题目。一看数据范围，n 不是很大，直接上线性复杂度的扫描算法。

算法具体就是对于每一个数字转成字符串后扫描字符串，统计数字个数即可。

PASCAL 代码见此。

时间效率 *O(N)*
空间效率 *O(1)*

算法二

如果对时间效率实在不放心，也可以用数学方法来完成。但相对复杂了一点。通常 NOIp 普及组的第一题用不着太高科技的算法。

时间效率 *O(1)*
空间效率 *O(1)*

代码_算法一

```
var x,i,j,k,l,m,n,Ans:Longint;
    s:Ansistring;
Begin
Assign(input,'count.in');
Assign(output,'count.out');
Reset(input);
rewrite(output);
Readln(n,x);
For i:=1 to N do
  Begin
  str(i,s);
  For j:=1 to Length(s) do
    If ord(s[j])-48=x then Inc(Ans);
  End;
writeln(Ans);
Close(input);
Close(output);
End.
```

表达式求值

（expr.pas/c/cpp）

Time Limit:1000Ms Memory Limit:131072K

算法一

表达式求值的题目已经堪称经典了。如果读者尝试过 NOIp2005 提高组《等价表达式》，那么这道题目其实非常轻松。

对于一般的（甚至更复杂的）表达式求值的题目，我们一般采用如下方法：

1、设两个栈：符号栈与数字栈；

2、扫描表达式，遇到数字则进栈，遇到符号则转第 3 步，扫描完毕转第 4 步；

3、遇到符号：首先将符号栈中优先级比当前符号大的都弹出，每次取出数字栈顶两个元素，求值后压入数字栈。然后将当前符号压入符号栈。转第 2 步；

4、扫描完毕后数字栈顶便是所求的值。

对于本题，只要输出最后 4 位，由于只包含“+”和“*”，因此可以边处理边 mod，甚至根本不用设置栈，直接先算出所有“*”出的值，然后依次相加即可，只不过没有设置栈的方法简单。

PASCAL 代码见此。

时间效率 *O(Len)* *//其中 Len 为表达式的位数*

空间效率 *O(Len)*

代码_算法一

```
var i,j,k,l,m,n,ToTm,ToTf:Longint;
    s,Ans:Ansistring;
    Tmp,Tmp1,Tmp2:Int64;
    Dis:array[' '..'@'] of Longint;
    Sm:array[0..200000] of Int64;
    sf:array[0..200000] of char;

Function Math(c:char):Boolean;
begin
Exit((c>='0')and(c<='9'));
End;

Procedure Doit(S:Ansistring);
var i,j,k:Longint;
    ch:char;
Begin
i:=1;
While i<=Length(s) do
  Begin
  If Math(s[i]) then
    Begin
    Tmp:=0;
    While Math(s[i]) do
      Begin
      Tmp:=Tmp*10+ord(s[i])-48;
      Inc(i);
      End;
    Inc(ToTm);Sm[ToTM]:=Tmp mod 10000;
    End Else
      Begin
      ch:=s[i];
      While Dis[sf[ToTf]]<=Dis[ch] do
        Begin
        Tmp1:=Sm[ToTm];
        Tmp2:=Sm[ToTm-1];
        If Sf[ToTf]='+' then
          Tmp:=Tmp1+Tmp2 Else Tmp:=Tmp1*Tmp2;
        Sm[ToTm-1]:=Tmp mod 10000;
        Sm[ToTm]:=0;
        Dec(ToTm);
        Sf[Totf]:=' ';
        Dec(ToTf);
        End;
```

```
      Inc(ToTf);
      Sf[ToTf]:=ch;
      Inc(i);
      End;
  End;
End;

Begin
Assign(input,'expr.in');
Assign(output,'expr.out');
Reset(input);
Rewrite(output);
Readln(s);
S:=S+'@';
Dis['*']:=1;
Dis['+']:=2;
Dis['@']:=3;
Dis[' ']:=4;
Sf[0]:=' ';
Doit(S);
sm[1]:=sm[1] mod 10000;
str(Sm[1],Ans);
Writeln(Ans);
Close(input);
Close(output);
End.
```

小朋友的数字

（number.pas/c/cpp）

Time Limit:1000Ms Memory Limit:131072K

算法一

首先是暴力算法。

第一种是完全按照题意模拟，枚举首尾，然后累加中间一段，取最大值。如果你愿意把累加中间段的这一重循环省略掉，加上前缀和优化即可。不过意义不大。

时间效率 $O(N^4)$ *//完全朴素*
$O(N^3)$ *//加上前缀和优化*
空间效率 $O(N)$

算法二

做题量稍大的同学估计读完题就能发现这是个（几乎是裸的）最大子段和问题。对于最大子段和问题，我们有 $O(N)$的算法。

具体的做法是这样的：当前要求第 I 位及之前的最大子段和，如果第（I-1）位及之前的最大子段和大于 0，则显然这一位取了也未尝不可（不会减少），也就是当前这一位和前面一段连接起来。否则的话，就新开一段——把前面的最大子段和改成 0 以后继续往下扫描。

如果你一定要说这是 DP 也可以，DP 方程为

$$f[i]=Max\{A[i],f[i-1]+A[i]\}$$

其中 $f[i]$表示前 i 个数的最大子段和。和上面的算法思路是一样的。

附 PASCAL 代码。

时间效率 $O(N)$
空间效率 $O(N)$

算法三

上述算法能拿到 80 分。关键在于算到后来已经超出了 int64 的范围。怎么办呢？

你可以写个高精度，不过要分正负讨论，比较麻烦，这里不再赘述。

有一个方法可以较好解决这个问题。想一想，除了第 1 个小朋友以外，后面的小朋友的分数值肯定是单调不递减的。那么我们不用把值都记录下来，可以一边处理，一边 mod。如果碰到可以更新的（当前这个小朋友的分数值大于 0），根本不用考虑是否比 Max 要大。（肯定比 Max 要大）因此直接用这个小朋友分数值 mod P 的值更新 Max 即可。至于第 1 个小朋友，再分类讨论一下即可。

附 PASCAL 代码。

时间效率 $O(N)$
空间效率 $O(N)$

代码_算法二

```
const oo=1 shl 60;
var i,j,k,l,m,n,P:Longint;
    Ans,Tmp:Int64;
    A,B,C:array[0..1000000+19] of Int64;
Begin
Assign(input,'number.in');
Assign(output,'number.out');
Reset(input);
Rewrite(output);
Readln(N,P);
For i:=1 to N do Read(A[i]);Readln;
Tmp:=0;Ans:=-oo;
For i:=1 to N do
  Begin
  Inc(Tmp,A[i]);
  If Tmp>Ans then Ans:=Tmp;
  If Tmp<0 then Tmp:=0;
  B[i]:=Ans;
  End;
C[1]:=B[1];
Ans:=C[1]+B[1];
For i:=2 to N do
  Begin
  C[i]:=Ans;
  If C[i]+B[i]>Ans then Ans:=C[i]+B[i];
  End;
Ans:=-oo;
For i:=1 to N do If C[i]>Ans then Ans:=C[i];
Writeln(Ans mod P);
Close(input);
Close(output);
End.

```

代码_算法三

```
const oo=1 shl 60;
var i,j,k,l,m,A1,n,P:Longint;
    Ans,Tmp:Int64;
    A:array[0..1000000+19] of Int64;
    Flag:Boolean;
Begin
Assign(input,'number.in');
Assign(output,'number.out');
Reset(input);
Rewrite(output);
Readln(N,P);
For i:=1 to N do Read(A[i]);Readln;
Tmp:=0;Ans:=-oo;
For i:=1 to N do
  Begin
  Inc(Tmp,A[i]);
  If Tmp>Ans then Ans:=Tmp;
  If Tmp<0 then Tmp:=0;
  A[i]:=Ans;
  End;
A1:=A[1];
Ans:=A1+A[1];
For i:=2 to N-1 do
  Begin
  Tmp:=Ans;
  If A[i]>0 then
    Begin
    Ans:=(Tmp+A[i]) mod P;
    If A[i]>Abs(A1) then Flag:=True;
    End;
  End;
If (Not Flag)and(Ans<A1) then Ans:=A1;
Writeln(Ans);
Close(input);
Close(output);
End.
```

车站分级

（level.pas/c/cpp）

Time Limit:1000Ms Memory Limit:131072K

算法一

这题有一点难度。

让我们思考一下，假如一趟火车停靠了了站 L 和站 R，而没有停靠 L 和 R 之间的站 A，意味着什么？站 L 和站 R 的等级肯定比站 A 的等级高。（注意这里是严格的大于）

请读者好好理解一下这句话。这是一个二元关系，我们可以连一条有向边L->A 以及 R->A.

下一步呢？下一步呢？

下一步显然就是求出这整张图上的最长路就可以了。最长路的长度就是车站划分最少的级数。

至于怎么求最长路？DFS？不不，这样效率太低了。我们仔细思考一下，这张图肯定是个有向无环图，有向就不用解释了，无环是因为肯定不存在站 A 的等级比 B 高，站 B 的等级又比 A 高的现象。

得出了这个结论，下一步就很简单了——做一个拓扑排序即可。

注意：不要以为时间效率主要耗费在预处理连边上，对于大部分情况（没有完全重复的），连边的效率是很高的。如果只是交了个裸的拓扑排序上去，大都只能得 80 分。

至于拓扑排序，还有一个优化，不必每一轮都找出入度为 0 的点，这一轮入读为 0 的点，必定是上一轮某些入度为 0 的点连接出来的。

因此，我们可以在上一轮拓扑排序时，就直接处理出下一轮入度为 0 的点。时间效率大大提高。

PASCAL 代码见此。

时间效率 $O(N^2)$

空间效率 $O(N)$

代码_算法一

```
const oo=maxlongint shr 1;
var s,i,j,k,l,m,n,S2,Ans,ss:Longint;
    A:array[0..1000+19,0..1000+19] of Boolean;
    D,Tmp,Tmp2:array[0..1000+19] of Longint;
    Bo:Boolean;
Begin
Assign(input,'level.in');
Assign(output,'level.out');
Reset(input);
Rewrite(output);
Readln(N,M);
For i:=1 to M do
  Begin
  Read(s);
  For j:=1 to s do Read(Tmp[j]);
  s2:=0;
  For j:=1 to s-1 do
    For k:=Tmp[j]+1 to Tmp[j+1]-1 do
      Begin
      Inc(s2);Tmp2[s2]:=k;
      End;
  For j:=1 to s do
    For k:=1 to s2 do
      A[Tmp[j],Tmp2[k]]:=True;
  End;
Fillchar(D,sizeof(D),0);
For i:=1 to N do
  For j:=1 to N do If A[j,i] then Inc(D[i]);
Fillchar(Tmp,sizeof(Tmp),0);
For i:=1 to N do
  If D[i]=0 then Begin Inc(s);Tmp[s]:=i;End;
For Ans:=1 to oo do
  Begin
  For i:=1 to N do If D[i]>0 then Break;
  If D[i]=0 then Break;
  ss:=0;
  For i:=1 to s do
    For j:=1 to N do
      If A[Tmp[i],j] then
        Begin
        A[Tmp[i],j]:=False;Dec(D[j]);
        If D[j]=0 then Begin Inc(ss);Tmp2[ss]:=j;End;
        End;
```

```
  Tmp:=Tmp2;s:=ss;
  End;
Writeln(Ans);
Close(input);
Close(output);
End.
```