吉林省 2012 信息学冬令营测试解题报告

第二试

2012 年 1 月 19 日 13:00-16:00

（请选手务必仔细阅读本页内容）

一. 题目概况

中文题目名称	数列	排队	树形图计数
英文题目名称	seq	lineup	count
可执行文件名	seq	lineup	count
输入文件名	seq.in	lineup.in	count.in
输出文件名	seq.out	lineup.out	count.out
每个测试点时限	2 秒	1 秒	1 秒
测试点数目	10	10	10
每个测试点分值	10	10	10
附加样例文件	有	有	有
题目类型	传统	传统	传统

二. 提交源程序文件名

对于 pascal 语言	seq.pas	lineup.pas	count.pas
对于 C 语言	seq.c	lineup.c	count.c
对于 C++语言	seq.cpp	lineup.cpp	count.cpp

三. 编译命令（不包含任何优化开关）

对于 pascal 语言	fpc seq.pas	fpc lineup.pas	fpc count.pas
对于 C 语言	gcc –o seq seq.c -lm	gcc –o lineup lineup.c -lm	gcc –o count count.c -lm
对于 C++语言	g++ -o seq seq.cpp -lm	g++ -o lineup lineup.cpp -lm	g++ -o count count.cpp -lm

四. 运行内存限制

内存上限	128M	128M	128M	128M

五. 注意事项

1、文件名（程序名和输入输出文件名）必须使用小写。

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 1.9GHz，内存 1G，上述时限以此配置为准。各省在自测时可根据具体配置调整时限。

1. 数列

(seq.pas/c/cpp)

【问题描述】

给你一个长度为 N 的正整数序列，如果一个连续的子序列，子序列的和能够被 K 整除，那么就视此子序列合法，求原序列包括多少个合法的连续子序列？

对于一个长度为 8 的序列：2, 1, 2, 1, 1, 2, 1, 2 。当 K=4 时，答案为 6，子序列是位置 1->位置 8,2->4,2->7,3->5,4->6,5->7。

【输入】

第一行：T，表示数据组数

对于每组数据：

第一行:2 个数，K，N

第二行：N 个数，表示这个序列

【输出】

共 T 行，每行一个数表示答案

【输入输出样例】

seq.in	seq.out
2 7 3 1 2 3 4 8 2 1 2 1 1 2 1 2	0 6

【数据范围】

30%数据满足：1<=T<=10，1<=N,K<=1,000

100%数据满足：1<=T<=20，1<=N<=50,000，1<=K<=1,000,000，序列的每个数<=1,000,000,000

【分析】

用 S[i]表示前 i 个数之和对 K 的余数，S[i]=(S[i-1]+a[i])mod K。如果(I,j)是一对合法序列即第 i 到第 j 个数之和是 K 的倍数，由于 a[i]+a[i+1]+…+a[j]=s[j]-s[i-1],因此 S[j]=S[i-1]。

用 num[i]记录前缀和中除以 k 的余数为 i 的个数。所以我们只要从左向右扫一次，当扫到 a[i] 时，计算出 S[i], 接下来只要执行 inc(ans,num[s[i]]);inc(num[s[i]])，当然别忘了一开始记录 num[0]=1。

时间复杂度为 O(T*N)。

程序如下：

```
var n,t,k,i,j,x,pre,ans:longint;
    num:array[0..1000000]of longint;
begin
  assign(input,'seq.in');
  reset(input);
  assign(output,'seq.out');
  rewrite(output);
  readln(t);
  for i:=1 to t do begin
    fillchar(num,sizeof(num),0);
    num[0]:=1;
    readln(k,n);
    pre:=0;
    ans:=0;
    for j:=1 to n do begin
      read(x);
      pre:=(pre+x)mod k;
      inc(ans,num[pre]);
      inc(num[pre]);
    end;
    readln;
    writeln(ans);
  end;
  close(input);
  close(output);
end.
```

2．排队

(lineup.pas/c/cpp)

【问题描述】

N 个正整数排成一排，每次操作允许你从中删除一个数再把它插入到任意位置。问最少需要几次操作可以把这 N 个数排成从小到大的序列。

【输入】

第一行输入 N(N<=100,000)表示数的个数。

第二行 N 个用空格隔开的正整数(每个数不超过 1,000,000)

【输出】

输出一个数表示最少需要的操作数

【输入输出样例 1】

lineup.in	lineup.out
3 1 2 3	0

【输入输出样例 2】

lineup.in	lineup.out
5 10 30 20 30 10	2

【输入输出样例 3】

lineup.in	lineup.out
6 1 1 1 2 3 1	1

【数据范围】

50%的数据满足:N<=5,000;

100%的数据满足：N<=100,000

【分析】

本题要求最少操作次数，我们可以考虑计算最多不用动的数的个数，从而转变成计算最长不下降子序列的长度 Len，再用 N-Len 就是答案。

方法一：

对于 50%的数据，设 F[I]表示以第 A[i]结尾的最长不下降子序列的长度，考虑 A[i]前一个数得到以下转移方程：

$$F[I]=\begin{cases}1 & I=1\\ \text{Max}\{F[J]+1,1\}\ \text{其中}\ J\ \text{满足}(1<=J<=I-1)AND(A[J]<=A[I]) & I>1\end{cases}$$

时间复杂度为 $O(N^2)$，预计得分：50 分。

方法一程序如下：

```
uses math;
var n,i,ans,num,j:longint;
   a,f:array[0..100000]of longint;
begin
   assign(input,'lineup.in');
   reset(input);
   assign(output,'lineup.out');
   rewrite(output);
   readln(n);
   for i:=1 to n do read(a[i]);
   num:=1;
   f[1]:=1;
   for i:=2 to n do begin
     f[i]:=1;
     for j:=1 to i-1 do begin
       if a[j]<=a[i] then f[i]:=max(f[i],f[j]+1);
     end;
     num:=max(num,f[i]);
   end;
   writeln(n-num);
   close(input);
   close(output);
end.
```

方法二：

贪心法。不下降子序列的最后一个数越小越好。记录 b[i]表示目前已经计算出来的长度为 i 的不下降子序列的最后一个元素的最小值。数组 b 是有序的。依次处理 a1,a2..,a[n]，对于 a[i],二分求出在 b[]数组中插入的位置 p，则 F[i]=p，同时把 b[p]改为 a[i]。

时间复杂度为 O(nlgn)。预计得分：100 分。

方法二程序如下：

```
var n,i,ans,num,l,r,m:longint;
    a,b:array[0..100000]of longint;
begin
    assign(input,'lineup.in');
    reset(input);
    assign(output,'lineup.out');
    rewrite(output);
    readln(n);
    for i:=1 to n do read(a[i]);
    num:=1;
    b[1]:=a[1];
    for i:=2 to n do begin
      if a[i]>=b[num] then begin
         inc(num);
         b[num]:=a[i];
         continue;
      end;
      l:=1;
      r:=num;
      while l<=r do begin
        m:=(l+r)shr 1;
        if b[m]<=a[i] then l:=m+1
        else r:=m-1;
      end;
      b[l]:=a[i];
    end;
    writeln(n-num);
    close(input);
    close(output);
end.
```

3. 树形图计数

(count.pas/c/cpp)

【问题描述】

小 k 同学最近正在研究最小树形图问题。所谓树形图，是指有向图的一棵有根的生成树，其中树的每一条边的指向恰好都是从根指向叶结点的方向。现在小 k 在纸上画了一个图，他想让你帮忙数一下这个图有多少棵树形图，树形图必须包括所有点。

【输入】

第 1 行输入 1 个正整数：n，表示图中点的个数

第 2~n+1 行每行输入 n 个字符，描述了这个图的邻接矩阵。第 i+1 行第 j 个字符如果是 0 则表示没有从 i 连向 j 的有向边，1 表示有一条从 i 到 j 的有向边。

【输出】

输出 1 行 1 个整数，表示这个有向图的树形图个数。

【输入输出样例】

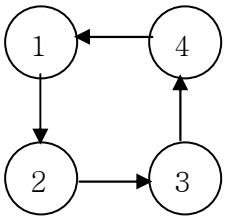
count.in	count.out
4 0100 0010 0001 1000	4

【数据范围】

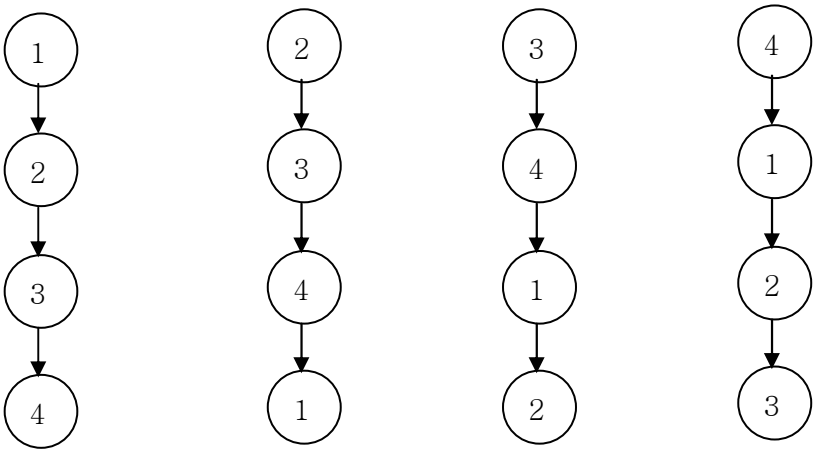
对于 100%的数据，n<=8。

【样例解释】

原图为：



树形图为以下 4 种：



【分析】

方法一：

树形图本质是一棵树，N 个结点的树有 N-1 条边，除了根结点以外每个结点有且只有一个父结点。我们可以用 DFS 来搜选取哪些边，注意在选取边(a,b)的时候要注意两个问题：

1. b 必须满足没有父结点；
2. b 不能和 a 属于同一个子树，

第一个问题可以用 fa[]来记录父结点就可以判断，如果 fa[b]<>0 说明有父结点，则不能选择这条边否则可以选择这条边，第二个问题利用 fa[]来判断是否有共同的祖先，如果有则不能选取这条边否则可以选取，注意第二个问题在判断的时候不能用路径压缩。

可以加上一个优化，如果剩余的边数不够的话就不用再继续搜下去。

方法一程序：

```
var n,i,j,edgenum:integer;
    ch:char;
    ans:longint;
    edge:array[1..64,1..2]of integer;
    fa:array[1..8]of integer;

function find(x:integer):integer;{查找x所在子树的根结点}
  begin
    if fa[x]=0 then exit(x)
    else exit(find(fa[x]));
  end;

procedure dfs(x,y:integer);
  var i,j,fa1,fa2:integer;
  begin
    if x=n then begin
       inc(ans);
       exit;
    end;
    if edgenum-y+1<n-x then exit;{如果边数不够则剪枝}
    dfs(x,y+1);
    if fa[edge[y,2]]=0 then begin{判断是否已经有父结点}
       fa1:=find(edge[y,1]);
       fa2:=find(edge[y,2]);
       if fa1<>fa2 then begin{判断是否在同一个子树中}
          fa[edge[y,2]]:=edge[y,1];
          dfs(x+1,y+1);
          fa[edge[y,2]]:=0;
       end;
    end;
  end;
```

```
begin
  assign(input,'count.in');
  reset(input);
  assign(output,'count.out');
  rewrite(output);
  readln(n);
  edgenum:=0;
  for i:=1 to n do begin
    for j:=1 to n do begin
      read(ch);
      if ch='1' then begin
        inc(edgenum);
        edge[edgenum,1]:=i;
        edge[edgenum,2]:=j;
      end;
    end;
    readln;
  end;
  ans:=0;
  fillchar(fa,sizeof(fa),0);
  dfs(1,1);
  writeln(ans);
  close(input);
  close(output);
end.
```

方法二：

方法一是以搜边，我们可以考虑除了根结点其他结点都有唯一的父结点，这样我们可以在主程序先枚举根结点 root，然后为了避免重复，我们从 1 到 N（除去 root）的顺序来确定它们的父结点，对于结点 x 来说，我们搜索所有指向 x 的边，如何判断 j 点可不可以作为 x 的父结点呢？只要满足 j 不是 x 的子孙即可，为了判断方便我们存储 son[i]表示以 i 为根结点的集合，用二进制压缩存储，所以只要判断(1 shl(j-1))and son[x]是否等于 0 即可，如果等于 0 说明 j 不是 x 的子孙可以作为 x 的父结点，否则不行。一旦确定 j 作为 x 的父结点，就要用 son[x]向上更新，更新程序如下：

```
procedure do1(j,x:integer);
  begin
    while j<>0 do begin
      son[j]:=son[j] or son[x];
      j:=fa[j];
    end;
  end;
```

递归调用返回断点后恢复原指程序如下：

```
procedure do2(j,x:integer);
```

```
  begin
     while j<>0 do begin
       son[j]:=son[j] xor son[x];
       j:=fa[j];
     end;
  end;
```

方法二程序如下：

```
var n,root,i,j:integer;
    ans:longint;
    a:array[1..8,0..8]of integer;
    fa,son:array[1..8]of integer;
    ch:char;
procedure do1(j,x:integer);
  begin
     while j<>0 do begin
       son[j]:=son[j] or son[x];
       j:=fa[j];
     end;
  end;
procedure do2(j,x:integer);
  begin
     while j<>0 do begin
       son[j]:=son[j] xor son[x];
       j:=fa[j];
     end;
  end;

procedure dfs(x:integer);
  var i,j:integer;
  begin
    if x=n+1 then inc(ans)
    else if x=root then dfs(x+1)
    else begin
      for i:=1 to a[x,0] do begin
        j:=a[x,i];
        if (1 shl(j-1)) and son[x]=0 then begin
           do1(j,x);fa[x]:=j;
           dfs(x+1);
           fa[x]:=0;do2(j,x);
        end;
      end;
    end;
  end;
```

```
begin
  assign(input,'count.in');
  reset(input);
  assign(output,'count.out');
  rewrite(output);
  readln(n);
  for i:=1 to n do begin
    for j:=1 to n do begin
      read(ch);
      if ch='1' then begin
        inc(a[j,0]);
        a[j,a[j,0]]:=i;
      end;
    end;
    readln;
  end;
  ans:=0;
  for root:=1 to n do begin
    fillchar(fa,sizeof(fa),0);
    for i:=1 to n do son[i]:=1 shl (i-1);
    dfs(1);
  end;
  writeln(ans);
  close(input);
  close(output);
end.
```