吉林省 2012 信息学冬令营测试解题报告

第一试

2012 年 1 月 19 日 8:30-11:30

（请选手务必仔细阅读本页内容）

一. 题目概况

中文题目名称	删边	取数	超车
英文题目名称	edge	choice	overtaking
可执行文件名	edge	choice	overtaking
输入文件名	edge.in	choice.in	overtaking.in
输出文件名	edge.out	choice.out	overtaking.out
每个测试点时限	1 秒	1 秒	2 秒
测试点数目	10	20	10
每个测试点分值	10	5	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	edge.pas	choice.pas	overtaking.pas
对于 C 语言	edge.c	choice.c	overtaking.c
对于 C++语言	edge.cpp	choice.cpp	overtaking.cpp

三. 编译命令（不包含任何优化开关）

对于 pascal 语言	fpc edge.pas	fpc choice.pas	fpc overtaking.pas
对于 C 语言	gcc –o edge edge.c -lm	gcc –o choice choice.c -lm	gcc –o overtaking overtaking.c -lm
对于 C++语言	g++ -o edge edge.cpp -lm	g++ -o choice choice.cpp -lm	g++ -o overtaking overtaking.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. 删边

(edge.pas/c/cpp)

【问题描述】

给你一个 N 个点 M 条边的无向连通图，问最多可以删除多少条边还能保持图连通。

【输入】

第一行输入两个用一个空格的整数 N 和 M(2<=N<=100,N-1<=M<=N*(N-1)/2)

接下来 M 行每行两个整数 x 和 y(1<=x,y<=N,x<>y)，表示点 x 和点 y 之间有一条边直接相连。

输入保证边不会重复出现并保证图是连通图。

【输出】

输出一个数表示最多可以删除的边数。

【输入输出样例】

edge.in	edge.out
4 6 1 2 2 3 3 4 1 4 2 4 1 3	3

【分析】这是一个送分题。题目给定的是 N 个点 M 条边的无向连通图，要保持连通最少需要N-1条边,而且原图保证连通,所以一定存在N-1条边保证图连通,所以最多删除M-(N-1)条边即可。

程序如下：

```
var n,m:integer;
begin
   assign(input,'edge.in');
   reset(input);
   assign(output,'edge.out');
   rewrite(output);
   readln(n,m);
   writeln(m-n+1);
   close(input);
   close(output);
end.
```

2. 取数

(choice.pas/c/cpp)

【问题描述】

从键盘上输入 3 个自然数 N、K 和 M（N<=1000000,K<=N/2），从 1，2，....，N 中任取 K 个数，要求所取的 K 个数中，任意两个数不能相差 1。编程求有多少种取法。

如：N=6 ，K=3,，从 1，2，3，4，5，6 中取 3 个数，任意两个数不能相差 1，取法如下:（1 3 5） （1 3 6） （1 4 6） （2 4 6）共有 4 种取法。

提示:（1 3 5）和（3 1 5）属于一种取法。

【输入】

一行，N、K 和 M，中间用空格隔开。

【输出】

一行，取法的种数对 M 的余数。

【输入输出样例】

choice.in	**choice.out**
6 3 1000003	4

【数据范围】

50%的数据满足:N<=100;

70%的数据满足:N<=1000;

85%的数据满足：N<=1000000,M=1000003;

另外的 15%的数据满足：N<=1000000,M= 2147483647。

【分析】

方法一：

对于50%的数据，我们可以定义F[I,J]表示从1到I中选择J个不连续数并且I一定要选的方案数。考虑I前一个要选的数进行状态转移：

$$F[I,J]=\begin{cases} 1 & J=1(\text{边界条件}) \\ \sum F[K,J-1](2*J-3<=K<=I-2) & J>1 \end{cases}$$

时间复杂度为$O(N^2*K)$。预计得分：50分。

方法一程序如下：

```
var n,k,m,i,j,ans,l:longint;
   f:array[0..1000,0..1000]of int64;
begin
   assign(input,'choice.in');
   reset(input);
   assign(output,'choice.out');
   rewrite(output);
   readln(n,k,m);
   for i:=1 to n do f[i,1]:=1;
   for j:=2 to k do begin
     for i:=2*j-1 to n do begin
       for l:=i-2 downto 2*j-3 do
       f[i,j]:=(f[i,j]+f[l,j-1])mod m;
     end;
   end;
   ans:=0;
   for i:=2*k-1 to n do ans:=(ans+f[i,k]) mod m;
   writeln(ans);
   close(input);
   close(output);
end.
```

方法二：

方法一可以通过前缀和优化到O(N*K)。另外我们可以重新定义F[I,J]表示从1到I中选择J个不连续数的方案数。通过考虑I选还是不选来进行状态转移。

1. 如果不选I：则方案数为F[I-1,J]；
2. 如果选I:由于不能选相邻两个数，所以I-1不能选，则剩余的J-1个数只能在1到I-2中选，即F[I-2,J-1]；

综上可得：

$$F[I,J]=\begin{cases} I & J=1(\text{边界条件}) \\ F[I-1,J]+F[I-2,J-1] & J>1 \end{cases}$$

时间复杂度为 O(N*K)。预计得分：70 分。

方法二程序如下：

```
var n,k,m,i,j:longint;
    f:array[0..1000,0..1000]of int64;
begin
    assign(input,'choice.in');
    reset(input);
    assign(output,'choice.out');
    rewrite(output);
    readln(n,k,m);
    for i:=1 to n do f[i,1]:=i;
    for j:=2 to k do begin
      for i:=2*j-1 to n do
        f[i,j]:=(f[i-1,j]+f[i-2,j-1])mod m;
    end;
    writeln(f[n,k]);
    close(input);
    close(output);
end.
```

方法三：

通过打表可以发现规律，当然我们也可以这样来考虑，为了保证所选 K 个数不连续，我们可以考虑先从 N-(K-1)个数中选择 K 个数出来，这样选出来的是不能保证不连续的，但我们可以把该方案调整成合法方案，只要把第 I(1<=I<=K)个数每个数加 I-1，这样每个方案就一一对应于一个合法方案。所以答案为 C(N-K+1,K)。

如题目中样例，我们可以认为是 C(4,3)，从 1 到 4 中选 3 个数的方案跟从 1 到 6 中选 3 个不连续的方案是一一对应的，用上面的方法得到以下对应关系：

（1，2，3）⇔ (1,3,5)

（1，2，4）⇔ (1,3,6)

（1，3，4）⇔ (1,4,6)

（2，3，4）⇔ (2,4,6)

注意 85% 的数据 N<=1000000,M=1000003，根据上面分析，答案 Ans=(N-K+1)*(N-K)*…*(N-2*K+2)/(K*(K-1)*…*1)mod M，我们可以先把分子即 (N-K+1)*(N-K)*…*(N-2*K+2)mod M 的值计算出来记为 a，同样把分母即 K*(K-1)*…*1 mod M 的值计算出来记为 b，由于这里 M=1000003 是一个素数，所以 Ans 的答案是唯一的。我们可以枚举 Ans 再判断 Ans*b mod M=a 是否成立即可。

时间复杂度为 O(N+M)，预计得分 85 分。

方法三程序如下：

```
var n,k,m,i,ans:longint;
    a,b:int64;
begin
```

```
    assign(input,'choice.in');
    reset(input);
    assign(output,'choice.out');
    rewrite(output);
    readln(n,k,m);
    a:=1;b:=1;
    for i:=1 to k do begin
      a:=a*(n-2*k+1+i) mod m;
      b:=b*i mod m;
    end;
    for ans:=1 to m-1 do
      if b*ans mod m=a then break;
    writeln(ans);
    close(input);
    close(output);
end.
```

方法四：

计算 C(N-K+1,K)mod M,跟方法三同样先计算出分子分母对 M 的余数 a 和 b，根据 x=(x/y)*y 可知：a=(Ans*b)mod m，再转化成 Ans*b+m*P=a，其中 b,m 和 a 为已知，并且 m 为素数，这样就变成我们熟悉的 a*x+b*y=c，用扩展 GCD 来求。

时间复杂度为 O(N)。预计得分：100 分。

方法四程序如下：

```
var n,k,m,i:longint;
    a,b,c,x,y:int64;
procedure gcd(a,b,c:int64;var x,y:int64);
  var x1,y1:int64;
  begin
    if b=0 then begin
      x:=c div a;
      y:=0;
      exit;
    end;
    gcd(b,a mod b,c,x1,y1);
    x:=y1;
    y:=x1-a div b*y1;
  end;
begin
    assign(input,'choice.in');
    reset(input);
    assign(output,'choice'+ch+'.out');
    rewrite(output);
```

```
    readln(n,k,m);
    a:=1;c:=1;
    for i:=1 to k do begin
      a:=a*i mod m;
      c:=c*(n-2*k+1+i) mod m;
    end;
    b:=m;
    gcd(a,b,c,x,y);
    x:=(x mod b+b)mod b;
    writeln(x);
    close(input);
    close(output);
end.
```

3. 超车

(overtaking.pas/c/cpp)

【问题描述】

jzabc 除了对多米诺骨牌感兴趣外，对赛车也很感兴趣。上个周末他观看了一场赛车比赛。他总是能想出许多稀奇的问题。某一时刻，他看到有 n 辆车（总是匀速行驶）在同一直线上，并且处在一个无限长度的直道上，而且 n 辆车有严格的先后之分。他通过特殊的器材测出了每一辆车的速度。那么问题出现了，如果有两辆车 A 车和 B 车，A 车在 B 车的后面，并且 A 车的速度比 B 车的快，那么经过一段时间后，A 车一定会超过 B 车。我们称之为一次超车。那么他想请你帮忙计算超车总数。我们记车道起点的坐标为 0。没有两辆车的坐标相同。

【输入】

第一行，一个数 n，车辆总数。以下 n 行为 n 两辆车的信息

第二行至第 n+1 行，每行有两个正整数 x，y，x 和 y 之间有一个空格。x 为车的坐标，y 为车的速度（ 0<x，y<=1,000,000,000）。

【输出】

一个整数表示超车总数。

【输入输出样例】

overtaking.in	overtaking.out
2 5 6 2 8	1

【数据范围】

20%的数据，n<=300；

50%的数据，n<=3000；

100%的数据，n<=300000

【分析】

分析什么时候才会发生超车，肯定是后面的车如果比前面的车速度快的话，那么这两辆车之间一定会发生一次超车，这样就转变成计算有多少对车满足后面的车速比前面的车速快。

按照 x 值从小到大排序，再利用归并排序求逆序对即可。

时间复杂度为 O(NlgN)。预计得分：100 分。

程序如下：
```
uses math;
const maxn=300100;
var a:array[0..maxn,1..2]of longint;
    t:array[1..maxn]of longint;
    n,i,ans:longint;
procedure qsort(l,r:longint);
  var i,j,t:longint;
  begin
    if l>=r then exit;
    i:=l;
    j:=r;
    t:=a[(i+j)shr 1,1];
    repeat
      while a[i,1]<t do inc(i);
      while a[j,1]>t do dec(j);
      if i<=j then begin
         a[0]:=a[i];
         a[i]:=a[j];
         a[j]:=a[0];
         inc(i);
         dec(j)
      end;    ;
    until i>j;
    qsort(l,j);
    qsort(i,r);
  end;
function count(l,r:longint):int64;
  var m,i,j,k:longint;
  begin
    if l=r then exit(0);
    m:=(l+r)shr 1;
    count:=count(l,m)+count(m+1,r);
    i:=l;j:=m+1;k:=i;
    while (i<=m)or(j<=r) do begin
      if i>m then begin
```

```
        t[k]:=a[j,2];
        inc(k);inc(j);continue;
      end;
      if j>r then begin
        t[k]:=a[i,2];
        inc(k);inc(i);continue;
      end;
      if a[i,2]>a[j,2] then begin
        inc(count,m-i+1);
        t[k]:=a[j,2];
        inc(k);inc(j);
      end else begin
        t[k]:=a[i,2];
        inc(k);inc(i);
      end;
    end;
    for k:=l to r do a[k,2]:=t[k];
  end;
begin
  assign(input,'overtaking.in');
  reset(input);
  assign(output,'overtaking.out');
  rewrite(output);
  readln(n);
  for i:=1 to n do readln(a[i,1],a[i,2]);
  qsort(1,n);
  writeln(count(1,n));
  close(input);
  close(output);
end.
```