Contents

1 数据结构

1.1 线段树

```
#include<stdio.h>

struct node{
    long long s,p;
};

node tr[1000000];

void pass(long p,long l,long r){
    tr[p].s+=tr[p].p*(r-l+1);
    if (l!=r){
        tr[p*2].p+=tr[p].p;
        tr[p*2+1].p+=tr[p].p;
    }
    tr[p].p=0;
}

void insert(long p,long l,long r,long a,long b,long long c){
    if (l==a&&r==b){
        tr[p].p+=c;
        return;
    }
    pass(p,l,r);
    tr[p].s+=(b-a+1)*c;
    int m=(l+r)/2;
    if (b<=m) insert(p*2,l,m,a,b,c);
    else if (a>m) insert(p*2+1,m+1,r,a,b,c);
    else {
        insert(p*2,l,m,a,m,c);
        insert(p*2+1,m+1,r,m+1,b,c);
    }
}

long long get(long p,long l,long r,long a,long b){
    pass(p,l,r);
    if (l==a&&r==b) return tr[p].s;
    long m=(l+r)/2;
    if (b<=m) return get(p*2,l,m,a,b);
    else if (a>m) return get(p*2+1,m+1,r,a,b);
    else {
        long long x1=get(p*2,l,m,a,m);
        long long x2=get(p*2+1,m+1,r,m+1,b);
        return x1+x2;
    }
}

int main(){
    freopen("3468.in","r",stdin);
    freopen("3468.out","w",stdout);
    long n,m,a,b,i;
    long long c;
    char ch;
    scanf("%ld%ld",&n,&m);
    for (i=1;i<=n;i++){
        scanf("%lld",&c);
        insert(1,1,n,i,i,c);
    }
    for (i=1;i<=m;i++){
        scanf("\n%c",&ch);
        if (ch=='Q') {
            scanf("%ld%ld",&a,&b);
            printf("%lld\n",get(1,1,n,a,b));
        }else {
            scanf("%ld%ld%lld",&a,&b,&c);
            insert(1,1,n,a,b,c);
        }
    }
    return 0;
}
//poj 3468
```

1.2 树状数组

```
#define lowbit(x) ((x)&(-(x)))
int sum(int *a,int x){
  int s=0;
  for(;x;x-=lowbit(x))s+=a[x];
  return s;
}
void update(int *a,int x,int w){
  for(;x<=n;x+=lowbit(x))a[x]+=w;
}
```

1.3 并查集

```
#include<stdio.h>

    long fa[100000];

int gf(int u){
```

```c
    if (u==fa[u]) return u;
    return fa[u]=gf(fa[u]);
}
void main(){
    long i,n,m,u,v,flag=0;
    scanf("%ld%ld",&n,&m);
    for (i=1;i<=n;i++) fa[i]=i;
    for (i=1;i<=m;i++){
        scanf("%ld%ld",&u,&v);
        if (gf(u)!=gf(v))
            fa[gf(u)]=gf(v);
        else {
            flag=1;
            break;
        }
    }
    printf("%ld\n",flag);
    (flag==1)?printf("false"):printf("true");
}
```

1.4 树链剖分

本质是树的分治。

size[u]=以u为根的子树的节点个数。令v为u的儿子中Size最大的一个，则边[u→v]为重边，其余边为轻边。

重路径：全部由重边组成的路径。

易证：每个点到根的路径上不超过logN条轻边和logN条重路径。

对于每条重路径，建立一棵线段树，可以用logN的时间求解。对于轻边直接处理。

【问题一】：需要维护的信息。

用2个dfs维护出以下信息：

fa[u]:u的父亲

deep[u]:u的深度

sz[u]:以u为根的子树节点个数

bel[u]:u属于哪条重路径

po[u]:u在所在重路径中的位置

hd[u]:u所在重路径的头

lh[hd[u]]:u所在重路径的尾在重路径中的位置

接着建立线段树并维护以下答案信息：

ma[nod]:tr[nod]这条链中的最大值

sum[nod]:链的和

□□还有很多看题目而定

【问题二】：如何简便的建立多棵线段树。

利用这些信息去建立很多线段树，如果将线段树们放到一个数组里维护，代码会简短很多。这样insert的时候就不是像普通线段树一样从1开始，而是从bel[u]开始，后面都相同，inc(path),然后tr[path]就是原先的一半什么的，都跟普通线段树一样处理，就是开始不从1开始。

【问题三】：如何找到x到y的路径。

普通法：查询x→lca(x,y)和y→lca(x,y)分别的答案，再合并。查询的方法是看hd[x]与hd[lca(x,y)]相等了没有，没有的话就查询整段线段树tr[bel[x]],再x=fa[hd[x]];相等了就只查那一段。

优化版：发现可以不用可以求lca就找到lca。做法是把x,y像上述方法一样不断往上提，但不是提到lca，而是hd谁矮提谁，最后必然会到同一条链上，并且高的那个就是lca。此部分看代码更易理解。

【问题四】：当权值在边上时，轻边不在线段树里怎么判。

只需要在每次一条重路径查完时看看重路径的头与其父亲的连边即可。

```pascal
原题：ZJOI2008 树的统计 Count bzoj题号1036
{$M 50000000}
uses math;
const oo=maxlongint>>2;
var
  s:string[5];
  ch:char;
  e:array[0..500000]of record v,n:longint; end;
  sz,bel,fa,po,hd,deep,ma,sum,h,lh,a:array[-1..500000]of longint;
  tr:array[0..500000,0..1]of longint;
  tot,path,n,m,i,j,u,v,x,y,ans,su:longint;

procedure add(u,v:longint);
begin
  inc(tot);
  e[tot].v:=v;
  e[tot].n:=h[u];
  h[u]:=tot;
end;

procedure dfs1(u:longint);
var y,v:longint;
begin
  sz[u]:=1;
  y:=h[u];
  while y>0 do begin
        v:=e[y].v;
        if v<>fa[u] then begin
                deep[v]:=deep[u]+1;
                fa[v]:=u;
                dfs1(v);
                inc(sz[u],sz[v]);
        end;
        y:=e[y].n;
  end;
```

```
end;

procedure dfs2(u,head,len:longint);
var y,v,ma:longint;
begin
  bel[u]:=path;
  lh[head]:=len;
  po[u]:=len;
  hd[u]:=head;
  ma:=-1; y:=h[u];
  while y>0 do begin
        v:=e[y].v;
        if (fa[u]<>v)and(sz[v]>sz[ma])then ma:=v;    //so sz:array[-1..]
        y:=e[y].n;
  end;
  if ma>0 then dfs2(ma,head,len+1);

  y:=h[u];
  while y>0 do begin
        v:=e[y].v;
        if (fa[u]<>v)and(v<>ma) then begin
                inc(path);
                dfs2(v,v,1);
        end;
        y:=e[y].n;
  end;
end;

procedure insert(nod,l,r,p,c:longint);
var mi:longint;
begin
  if l=r then begin
        ma[nod]:=c; sum[nod]:=c;
        exit;
  end;
  mi:=(l+r)>>1;
  if tr[nod,0]=0 then begin inc(path); tr[nod,0]:=path; end;
  if tr[nod,1]=0 then begin inc(path); tr[nod,1]:=path; end;
  if p<=mi then insert(tr[nod,0],l,mi,p,c) else insert(tr[nod,1],mi+1,r,p,c);
  sum[nod]:=sum[tr[nod,0]]+sum[tr[nod,1]];
  ma[nod]:=max(ma[tr[nod,0]] , ma[tr[nod,1]]);
end;

procedure ask(nod,l,r,a,b,k:longint);
var mi:longint;
begin
  if (l=a)and(r=b) then begin
        if k=1 then ans:=max(ans,ma[nod]) else inc(su,sum[nod]);
        exit;
  end;
  mi:=(l+r)>>1;
  if b<=mi then ask(tr[nod,0],l,mi,a,b,k)
  else if a>mi then ask(tr[nod,1],mi+1,r,a,b,k)
  else begin
        ask(tr[nod,0],l,mi,a,mi,k);
        ask(tr[nod,1],mi+1,r,mi+1,b,k);
  end;
end;

procedure get(x,y,k:longint);
begin
  ans:=-oo; su:=0;
  while hd[x]<>hd[y] do begin
        if deep[hd[x]]>deep[hd[y]] then begin
                ask(bel[x],1,lh[hd[x]],1,po[x],k);
                x:=fa[hd[x]];
        end
        else begin
                ask(bel[y],1,lh[hd[y]],1,po[y],k);
                y:=fa[hd[y]];
        end;
  end;
  ask(bel[x],1,lh[hd[x]],min(po[x],po[y]),max(po[x],po[y]),k);
  if k=1 then writeln(ans) else writeln(su);
end;


begin
assign(input,'count.in'); reset(input);
assign(output,'count.out'); rewrite(output);
  readln(n);
  for i:=1 to n-1 do begin
        readln(u,v);
        add(u,v); add(v,u);
  end;
  for i:=1 to n do read(a[i]); readln;
  dfs1(1); dfs2(1,1,1);
  for i:=1 to n do insert(bel[i],1,lh[hd[i]],po[i],a[i]);

  readln(m);
  for i:=1 to m do begin
        read(s);
        if s='QMAX ' then begin
                readln(x,y);
                get(x,y,1);
        end;
```

```
        if s='QSUM ' then begin
                readln(x,y);
                get(x,y,2);
        end;
        if s='CHANG' then begin
                readln(ch,x,y);
                insert(bel[x],1,lh[hd[x]],po[x],y);
        end;
  end;
close(output);
end.
```

1.5 树的分治

```
对于一个点，边只有经过此点和未经过此点2类。对于未经过的递归处理。经过的通过dfs可以解决。若选取重心求解则满足最多只有logn层，这样效率是nlogn，但dfs后的值需排序，于是n(logn)^2.

做法：
每次用2个dfs(getsize,getroot)找一棵树的重心，标记重心已访问，再从重心开始dfs(getdist)，求出未访问的点到重心的距离，存起来。然后做完了加到答案里。再去dfs(doit)重心切开后剩下子树的。一共4个dfs。
统计时对于未走最短路的路径只需每次一个v出来的先剪掉。
```

```
//poj1741 楼教男人八题tree
var
  n,k,i,u,v,l,et,top,pt,ans:longint;
  vis:array[0..20000]of boolean;
  q,z,h,dis,size:array[0..20000]of longint;
  e:array[0..100000]of record v,n,l:longint; end;

procedure swap(var a,b:longint);
var c:longint;
begin
  c:=a; a:=b; b:=c;
end;

procedure sort(l,r:longint);
var
  i,j,x:longint;
begin
  i:=l; j:=r;
  x:=q[(l+r)>>1];
  repeat
    while q[i]<x do inc(i);
    while x<q[j] do dec(j);
    if not(i>j) then begin
      swap(q[i],q[j]);
      inc(i); dec(j);
    end;
  until i>j;
  if l<j then sort(l,j);
  if i<r then sort(i,r);
end;

procedure add(u,v,l:longint);
begin
      inc(et);
      e[et].v:=v;
      e[et].l:=l;
      e[et].n:=h[u];
      h[u]:=et;
end;

procedure dfs(u:longint);                //get dist
var   y,v:longint;
begin
      vis[u]:=true;
      inc(top); q[top]:=dis[u];
      y:=h[u];
      while y>0 do begin
            v:=e[y].v;
            if not vis[v] then begin
                  dis[v]:=dis[u]+e[y].l;
                  dfs(v);
            end;
            y:=e[y].n;
      end;
      vis[u]:=false;
end;

function calc(l,r:longint):longint;
begin
      calc:=0;
      sort(l,r);
      while l<=r do begin
            while (q[r]+q[l]>k)and(r>l) do dec(r);
            if r>l then inc(calc,r-l);
            inc(l);
      end;
end;

procedure gs(u:longint);                       //get size
var   v,y:longint;
begin
      vis[u]:=true; size[u]:=1;
```

```
    y:=h[u];
    while y>0 do begin
        v:=e[y].v;
        if not vis[v] then begin
            gs(v);
            inc(size[u],size[v]);
        end;
        y:=e[y].n;
    end;
    vis[u]:=false;
end;

procedure gr(var p:longint);                        //get root
var   u,l,r,y,v,i:longint;
      flag:boolean;
begin
    l:=0; r:=1;
    z[1]:=p; vis[p]:=true;
    while l<r do begin
        inc(l); u:=z[l];
        flag:=size[u]*2>=size[p];
        y:=h[u];
        while y>0 do begin
            v:=e[y].v;
            if not vis[v] then begin
                if size[v]*2>size[p] then flag:=false;
                inc(r); vis[v]:=true; z[r]:=v;
            end;
            y:=e[y].n;
        end;
        if flag then begin p:=u; break; end;
    end;
    for i:=1 to r do vis[z[i]]:=false;
end;

procedure doit(u:longint);
var   y,v:longint;
begin
    gs(u); if size[u]=1 then begin vis[u]:=true; exit; end;
    gr(u); vis[u]:=true;
    pt:=1; top:=1; q[1]:=0;
    y:=h[u];
    while y>0 do begin
        v:=e[y].v;
        if not vis[v] then begin
            dis[v]:=e[y].l;
            dfs(v);
            dec(ans,calc(pt+1,top));
            pt:=top;
        end;
        y:=e[y].n;
    end;
    inc(ans,calc(1,top));
//-------------------------------------calc ans
    y:=h[u];
    while y>0 do begin
        if not vis[e[y].v] then doit(e[y].v);
        y:=e[y].n;
    end;
end;

begin
assign(input,'1741.in'); reset(input);
assign(output,'1741.out'); rewrite(output);
while true do begin
    readln(n,k);
    ans:=0; et:=0;
    fillchar(h,sizeof(h),0);
    fillchar(vis,sizeof(vis),0);

    if (n=0)and(k=0) then break;
    for i:=1 to n-1 do begin
        readln(u,v,l);
        add(u,v,l);
        add(v,u,l);
    end;
    doit(1);
    writeln(ans);
end;
close(output);
end.
```

1.6 可持久化线段树

给每段前缀s[1..i]都建一棵线段树，这棵线段树存的是数字在序列中出现次数，或一段数字中每个数字在序列中出现次数之和。

这n棵线段树的结构就完全一样的，不同的只是每个节点上的数值不同而已。

发现s[1..i+1]跟s[1..i]相比只有s[i+1]这个数一直到根的logn个节点要+1，而其他部分跟s[1..i]是一样的，那么去访问的时候直接访问前一棵树的这个部分即可。于是每加一个点新增logn个节点，n棵线段树节点数只用nlogn。

查询s[x,y]的时候，只需要看s[1..x-1]和s[1..y]，每个节点的差值就是s[x,y]中nlogn个数段在序列中的出现次数。

一般情况下要离散。

```
//poj2104
var
      n,m,i,x,y,k,tot:longint;
      tr:array[0..3000000] of record n,l,r:longint; end;
      a,p,pp:array[0..200000]of longint;

procedure swap(var a,b:longint);
var c:longint;
begin
  c:=a; a:=b; b:=c;
end;

procedure sort(l,r:longint);
var
  i,j,x:longint;
begin
  i:=l; j:=r;
  x:=a[(l+r)>>1];
  repeat
    while a[i]<x do inc(i);
    while x<a[j] do dec(j);
    if not(i>j) then begin
      swap(a[i],a[j]);
      swap(p[i],p[j]);
      inc(i); dec(j);
    end;
  until i>j;
  if l<j then sort(l,j);
  if i<r then sort(i,r);
end;

procedure insert(x:longint);
var   p,q,l,r,m:longint;
begin
      l:=1; r:=n;
      p:=i; q:=i-1;
      while true do begin
            tr[p].n:=tr[q].n+1;
            if l=r then break;
            m:=(l+r)>>1;
            if x<=m then begin
                  tr[p].r:=tr[q].r;
                  inc(tot); tr[p].l:=tot;
                  p:=tot; q:=tr[q].l;
                  r:=m;
            end
            else begin
                  tr[p].l:=tr[q].l;
                  inc(tot); tr[p].r:=tot;
                  p:=tot; q:=tr[q].r;
                  l:=m+1;
            end;
      end;
end;

function kfnd(x,y,k,l,r:longint):longint;
var   t,m:longint;
begin
      t:=tr[tr[y].l].n-tr[tr[x].l].n; m:=(l+r)>>1;
      if l=r then begin writeln(a[l]); exit; end;
      if k<=t then kfnd(tr[x].l,tr[y].l,k,l,m)
      else kfnd(tr[x].r,tr[y].r,k-t,m+1,r);
end;

begin
assign(input,'chair.in'); reset(input);
assign(output,'chair.out'); rewrite(output);
      readln(n,m);
      tot:=n;
      for i:=1 to n do begin read(a[i]); p[i]:=i; end; readln;
      sort(1,n);
      for i:=1 to n do pp[p[i]]:=i;
      for i:=1 to n do insert(pp[i]);
      for i:=1 to m do begin readln(x,y,k); kfnd(x-1,y,k,1,n); end;
close(output);
end.
```

1.7 可并堆

```
将两个堆合并，且合并后仍满足堆的性质，并快速实现堆的功能，这就是可合并堆。

左偏树的原理是不仅满足key值，还要满足dis[lson[a]]<dis[rson[a]]，可以得到最大的左偏树是满二叉树。于是层数小于logn，操作效率是nlogn的。
删除最小(大)的节点：取走树根，并将左右儿子合并。
加入一个点：将一个点与原左偏树合并。

具体请看：左偏树的特点及其应用.doc

随机堆的加入删除等操作与左偏树完全相同，体现在程序上就是就是只有merge函数的不同。
随机堆即randam出0或1，若0则与左儿子并，1则右儿子。由于是基于随机的，层数大约也是logn层，且很难出数据卡掉他。在大多数据下跑的比左偏树快(因为左偏树向左偏的，在右边还空的时候左边可能已经有达到logn层的链了)，从程序上看也好记一些。
```

```
但若需删除其中的某个点(非最大最小点)：
左偏树, 删除P且将左右儿子合并, 得到新的子树。此时再不断向上调整。调整伪代码：
while q ≠ NULL do
    If dist(left(q)) < dist(right(q)) Then
        swap(left(q), right(q))
    If dist(right(q))+1 = dist(q) Then
        Exit Procedure
    dist(q) ← dist(right(q))+1
    p ← q
    q ← parent(q)
End

对于随机堆就更方便了，将左右儿子合并来替代p即可。
```

1.7.1 随机堆

```
int rand(){return t=t^1;}
int merge(int x,int y){
    if (!x || !y) return x+y;
    if (key[x]>key[y]) swap(x,y);
    if (rand()) lson[x]=merge(lson[x],y);
    else rson[x]=merge(rson[x],y);
    return x;
}
```

1.7.2 左偏树

```
//bzoj 1455: 罗马游戏
var
      i,n,m,a,b,t,f1,f2,f3:longint;
      fa,lson,rson,key,dis:array[0..2000000]of longint;
      kd:array[0..2000000]of boolean;
      ch:char;

function gf(a:longint):longint;
begin
      if fa[a]=a then exit(a);
      fa[a]:=gf(fa[a]); exit(fa[a]);
end;
procedure swap(var a,b:longint);
var c:longint; begin c:=a; a:=b; b:=c; end;
function rand:boolean;
begin inc(t); exit(t and 1=1); end;

{function merge(x,y:longint):longint;        //这是随机堆
begin
      if (x=0)or(y=0) then exit(x+y);
      if key[x]>key[y] then swap(x,y);
      if rand then lson[x]:=merge(lson[x],y)
             else rson[x]:=merge(rson[x],y);
      exit(x);
end;}

function merge(x,y:longint):longint;          //左偏树
begin
      if (x=0)or(y=0) then exit(x+y);
      if key[x]>key[y] then swap(x,y);
      lson[x]:=merge(lson[x],y);
      if dis[rson[x]]>dis[lson[x]] then swap(lson[x],rson[x]);
      if rson[x]=0 then dis[x]:=0
                   else dis[x]:=dis[rson[x]]+1;
      exit(x);
end;

begin
assign(input,'1455.in'); reset(input);
assign(output,'1455.out'); rewrite(output);
      readln(n);
      for i:=1 to n do begin
            fa[i]:=i;
            read(key[i]);
      end;
      readln;
      readln(m);
      for i:=1 to m do begin
            read(ch);
            if ch='M' then begin
                  readln(a,b);
                  f1:=gf(a); f2:=gf(b);
                  if kd[a] or kd[b] or (f1=f2) then continue;
                  f3:=merge(f1,f2);
                  fa[f1]:=f3; fa[f2]:=f3;
            end
            else if ch='K' then begin
                  readln(a);
                  if kd[a] then begin writeln(0); continue; end;
                  a:=gf(a);
                  kd[a]:=true;
                  writeln(key[a]);
                  fa[a]:=merge(lson[a],rson[a]);
                  fa[fa[a]]:=fa[a];
            end;
      end;
close(output);
end.
```

1.8 平衡树

注意每个子过程中p,q的含义，都有可能为父或子。

断线连线的顺序为：下上中

tup中只tget(q)的原因是p还要继续上转。

pass标记下传，tget信息上传。为了得到完整的信息，应当在找点的时候就把标记下传，只有极少数情况下不必，懒得动脑子怕想错就直接下传吧，反正效率几乎相等。而任何一次改变树的结构都要对相应点进行tget,从儿子那里得到新的信息。

pass和tget的原则是处理后信息只读此点即可知晓。

注意在pass过程中，add什么的不仅加到num里，还要加到max。

注意在tget过程中还要先把左右儿子给pass了，因为信息从左右儿子中传出来的。

2种类型的kfnd （kfnd即select)

2种类型的build

搞出一段区间[x,y],即kfnd到x-1对应的u和y+1对应的v，u转到根，v转到u.r，则此区间就是v.l，打标记什么的像线段树一样对待他。

在x后插入一段区间，就把x转到root,x+1转到root.r,于是root.l.r必然空着，把他们插进去吧。

如果要操作[1,k]或者[k,n]时，会发现没有l-1或r+1,只要自己添加这两个点就可以了，注意添加的两个点的信息也要初始化，不能就0放在那里，要保证不能影响答案，具体看题。

在空间可能不够的情况下，要回收空间，即删掉的部分要用起来。于是把删掉的点的序号存到队列里，新增点不要直接inc(tot),如果队列里有东西就拿出来用。

合并splay:把小的一个个拆下来插到大的里去，看似效率低下，但是把一片森林并成一棵树竟然均摊NlogN。我不会证，但这是事实。

```
//0:左儿子的标号   1：右儿子的标号     2：父亲标号    3..9：本节点的值
//0:lch;1:rch;2:fat;3:v;4:x;5:num;6:snum;7:sumv;8:sumx;9:sumv*x;
const
        maxn=30000;
var
        b:array[0..maxn,0..9] of int64;
        a:array[0..maxn,1..2] of int64;
        n,p,q,trs:int64;
        i:longint;
        ans:int64;

function sort(l,r:int64):int64;
var     x,i,j:int64;
        t:array[1..2] of int64;
begin
        i:=l;j:=r;x:=a[(l+r) shr 1,2];
        repeat
                while a[i,2]<x do inc(i);
                while a[j,2]>x do dec(j);
                if i<=j then begin
                        t:=a[i];
                        a[i]:=a[j];
                        a[j]:=t;
                        inc(i);
                        dec(j);
                end;
        until i>j;
        if l<j then sort(l,j);
        if i<r then sort(i,r);
end;

function tget(p:int64):int64;           //更新，维护题目中的需要的信息
var     l,r:int64;
begin
        l:=b[p,0];r:=b[p,1];
        b[p,6]:=b[l,6]+b[r,6]+b[p,5];
        b[p,7]:=b[l,7]+b[r,7]+b[p,3];
        b[p,8]:=b[l,8]+b[r,8]+b[p,4];
        b[p,9]:=b[l,9]+b[r,9]+b[p,3]*b[p,4];
end;

function build(p,v,x,d:int64):int64;        //p:父节点  v,x:节点信息  d:是父亲的左或右节点
begin
        inc(trs);
        b[trs,2]:=p;
        b[trs,3]:=v;
        b[trs,4]:=x;
        b[trs,5]:=1;
        tget(trs);
        b[p,d]:=trs;
        exit(trs);
end;

function ir(p:int64):int64;                 //p是其父亲的左节点则返回0，右节点返回1
begin
        //exit(ord(b[b[p,2],1]=p));         作用相同
        if b[b[p,2],0]=p then exit(0)
                else exit(1);
```

```
end;

function rtup(p:int64):int64;    //p：本节点  q：父节点
var     q,x,y:int64;
begin
        q:=b[p,2];x:=ir(p);y:=x xor 1;

        b[q,x]:=b[p,y];        //q的x方向孩子为p的y方向孩子
        if b[p,y]<>0 then b[b[p,y],2]:=q;     //十分重要，否则超级跟0节点会被转走

        b[p,2]:=b[q,2];        //将q脱离顶点，p顶替
        b[b[q,2],ir(q)]:=p;

        b[p,y]:=q;             //将p成为q的父亲
        b[q,2]:=p;

        tget(q);
end;

procedure sply(p:int64);              //将p转到根
begin
        if p=0 then exit;
        while b[p,2]<>0 do begin
                if b[b[p,2],2]<>0 then
                        if ir(p)=ir(b[p,2]) then rtup(b[p,2])
                                else rtup(p);
                rtup(p);
        end;
        tget(p);
end;

function kfnd(p,v:int64):int64;
var     q:int64;
begin
        if b[p,3]>v then
                if b[p,0]=0 then exit(-1)
                        else exit(kfnd(b[p,0],v))
        else begin
                q:=0;                   //重点。。必须要有q。。本过程应按需编写，
                  但q必不可少，否则有可能输出无解的答案。。
                if b[p,1]<>0 then q:=kfnd(b[p,1],v);
                if q<=0 then exit(p)
                        else exit(q);
        end;
end;

function tins(p,v,x:int64):int64;    //插入一个点
begin
        if (v<b[p,3])or(b[p,3]=v)and(b[p,4]<=x) then
                if b[p,0]=0 then exit(build(p,v,x,0))
                        else exit(tins(b[p,0],v,x))
        else
                if b[p,1]=0 then exit(build(p,v,x,1))
                        else exit(tins(b[p,1],v,x));
        tget(p);
end;

begin
        read(n);
        for i:=1 to n do read(a[i,1],a[i,2]);
        sort(1,n);
        for i:=2 to n do begin
                if i=2 then build(0,a[i-1,1],a[i-1,2],1)
                else begin
                        p:=tins(b[0,1],a[i-1,1],a[i-1,2]);
                        sply(p);
                end;
                p:=kfnd(b[0,1],a[i,1]);
                if p=-1 then begin
                        p:=b[0,1];
                        ans:=b[p,7]*a[i,2]-b[p,9]+ans;
                        continue;
                end else if p=0 then begin
                        p:=b[0,1];
                        ans:=a[i,1]*a[i,2]*b[p,6]-a[i,1]*b[p,8]+ans;
                        continue;
                end;
                sply(p);
                q:=b[p,1];
                ans:=b[q,7]*a[i,2]-b[q,9]+ans;
                while b[q,0]<>0 do q:=b[q,0];
                sply(q);
                ans:=a[i,1]*a[i,2]*b[p,6]-a[i,1]*b[p,8]+ans;
        end;
        writeln(ans);
end.

poj1990
```

```
//1:left 2:right 3:father 4:num 5:change 6:sum 7:maxl 8:maxr 9:maxm 10:size 11:turn
const oo=maxlongint>>2; mo=510000;
var
  c:string[7];
  ch:char;
  p,i,j,n,m,x,y,k,ll,rr:longint;
  qq,a:array[0..1000000]of longint;
  b:array[0..1000000,1..11]of longint;
```

```
procedure swap(var a,b:longint);
var c:longint;
begin c:=a; a:=b; b:=c; end;

function max(a,b:longint):longint;
begin if a>b then exit(a); exit(b); end;

function ir(p:longint):longint;
begin
  if b[b[p,3],1]=p then exit(1); exit(2);
end;

procedure pass(p:longint);
var k:longint;
begin
  if p=0 then exit;
  if b[p,11]=1 then begin
      swap(b[p,1],b[p,2]);
      swap(b[p,7],b[p,8]);
      if b[p,1]<>0 then b[b[p,1],11]:=1-b[b[p,1],11];
      if b[p,2]<>0 then b[b[p,2],11]:=1-b[b[p,2],11];
      b[p,11]:=0;
  end;
  if b[p,5]<>-oo then begin
      k:=b[p,5]; b[p,5]:=-oo;
      b[p,4]:=k;
      b[p,6]:=k*b[p,10];
      if b[p,1]<>0 then b[b[p,1],5]:=k;
      if b[p,2]<>0 then b[b[p,2],5]:=k;
      b[p,9]:=max(k,b[p,6]);
      if k>0 then k:=b[p,6] else k:=0;     //b[p,9]=k?
      b[p,7]:=k; b[p,8]:=k;
  end;
end;

procedure tget(p:longint);
var l,r:longint;
begin
  if p=0 then exit;
  l:=b[p,1]; r:=b[p,2];
  pass(l); pass(r);
  b[p,6]:=b[l,6]+b[r,6]+b[p,4];
  b[p,7]:=max(b[l,7],b[l,6]+b[p,4]+b[r,7]);
  b[p,8]:=max(b[r,8],b[r,6]+b[p,4]+b[l,8]);
  b[p,9]:=max(max(b[l,9],b[r,9]), b[l,8]+b[p,4]+b[r,7]);
  b[p,10]:=b[l,10]+b[r,10]+1;
end;

procedure tup(p:longint);
var q,x,y:longint;
begin
  q:=b[p,3]; x:=ir(p); y:=3-x;
  b[q,x]:=b[p,y];
  if b[p,y]<>0 then b[b[p,y],3]:=q;

  b[p,3]:=b[q,3];
  b[b[q,3],ir(q)]:=p;

  b[q,3]:=p;
  b[p,y]:=q;
  tget(q);
end;

procedure splay(p,v:longint);
begin
  if p=v then exit;
  while b[p,3]<>v do begin
      if b[b[p,3],3]<>v then
            if ir(p)=ir(b[p,3]) then tup(b[p,3]) else tup(p);
      tup(p);
  end;
  tget(p);
end;

function kfnd(p,k:longint):longint;
begin
  pass(p);
  if k=b[b[p,1],10]+1 then exit(p);
  if k<b[b[p,1],10]+1 then exit(kfnd(b[p,1],k))
                      else exit(kfnd(b[p,2],k-b[b[p,1],10]-1));
end;

procedure build(p,l,r,d:longint);
var q,mi,i:longint;
begin
  if l>r then exit;
  mi:=(l+r)>>1;
  q:=qq[ll]; ll:=(ll mod mo)+1;
  for i:=1 to 11 do b[q,i]:=0;
  b[q,5]:=-oo;
  b[q,3]:=p; b[p,d]:=q;
  b[q,4]:=a[mi];
  build(q,l,mi-1,1);
  build(q,mi+1,r,2);
  tget(q);
```

```
end;

function getxy(x,y:longint):longint;
var u,v:longint;
begin
  u:=kfnd(b[0,2],x-1);
  v:=kfnd(b[0,2],y+1);
  splay(u,0); splay(v,b[0,2]);
  exit(b[b[b[0,2],2],1]);
end;

procedure kong(p:longint);
begin
  rr:=(rr mod mo)+1;
  qq[rr]:=p;
  if b[p,1]<>0 then kong(b[p,1]);
  if b[p,2]<>0 then kong(b[p,2]);
end;

begin
assign(input,'sequence.in'); reset(input);
assign(output,'sequence.out'); rewrite(output);
  readln(n,m);
  for i:=2 to n+1 do read(a[i]); readln;
  a[1]:=-oo; a[n+2]:=-oo;
  for i:=1 to mo do qq[i]:=i;
  ll:=1; rr:=mo; b[0,9]:=-oo;
  build(0,1,n+2,2);

  for j:=1 to m do begin
      read(c);
      if c='INSERT ' then begin
                  read(x,y); inc(x);
                  getxy(x+1,x);
                  for i:=1 to y do read(a[i]); readln;
                  build(b[b[0,2],2],1,y,1);
                  tget(b[b[0,2],2]); tget(b[0,2]);
      end;
      if c='DELETE ' then begin
                  readln(x,y); inc(x); y:=x+y-1;
                  p:=getxy(x,y);
                  b[b[p,3],ir(p)]:=0;
                  tget(b[b[0,2],2]); tget(b[0,2]);
                  kong(p);
      end;
      if c='MAKE-SA' then begin
                  read(ch); read(ch);
                  readln(x,y,k); inc(x); y:=x+y-1;
                  p:=getxy(x,y);
                  b[p,5]:=k;
                  pass(p);
                  tget(b[b[0,2],2]); tget(b[0,2]);
      end;
      if c='REVERSE' then begin
                  readln(x,y); inc(x); y:=x+y-1;
                  p:=getxy(x,y);
                  b[p,11]:=1-b[p,11];
                  pass(p);
                  tget(b[b[0,2],2]); tget(b[0,2]);
      end;
      if c='GET-SUM' then begin
                  readln(x,y); inc(x); y:=x+y-1;
                  p:=getxy(x,y);
                  writeln(b[p,6]);
      end;
      if c='MAX-SUM' then begin
                  readln;
                  writeln(b[b[0,2],9]);
      end;
  end;
close(output);
end.
```

2 图论

2.1 2set

```
若i与j矛盾，则选i就必选j'，选j就必选i'，（但是选了i'未必选j，也可以选j'，所以是单向边），于是连边i-->j'和j-->i'。可以发现这样构图是反向对称的（这部分的证明请看资料）。
这样构图后如果i与i'可以互达，说明选i必选i'，选i'必选i，所以不成立（这一步的做法是缩点看在不在同一强连通分量中）。
如果任意i与i'都不能互达，缩完点后就成了个有向无环拓扑图。
```

```
int n,x[N],y[N],flag,q[N],top,low[N],dfn[N],cc,f[N],ff[N],po[N];
vector<int> e[N];
void add(int u,int v){e[u].PB(v);}

void tarjan(int u){
    low[u]=dfn[u]=++cc;
    q[++top]=u;
    f[u]=ff[u]=true;
    int v;
    For(i,e[u].size()){
        v=e[u][i];
```

```
        if (!f[v]){
            tarjan(v);
            upmin(low[u],low[v]);
        }
        else if (ff[v]) upmin(low[u],dfn[v]);
    }

    if (dfn[u]==low[u]) do{
        v=q[top--];
        ff[v]=false;
        po[v]=u;
    } while (u!=v);
}

void main(){
    freopen("3207.in","r",stdin);
    freopen("3207.out","w",stdout);
    flag=true;
    cin>>n>>n;
    rep(i,1,n){
        cin>>x[i]>>y[i];
        if (x[i]>y[i]) swap(x[i],y[i]);
    }
    rep(i,1,n-1) rep(j,i+1,n)
        if (!( (y[i]<=x[j]) || (y[j]<=x[i]) || ((x[i]<=x[j])&&(y[j]<=y[i])) || ((x[i]>=x[j])&&(y[j]>=y[i])) )){
            add(i,j+n); add(j,i+n);
            add(i+n,j); add(j+n,i);
        }
    rep(i,1,2*n) if (!f[i]) tarjan(i);
    rep(i,1,n) if (po[i]==po[i+n]) flag=false;
    flag?cout<<"panda is telling the truth..."<<endl:cout<<"the evil panda is lying again"<<endl;
}
```

2.2 欧拉路

```
/*
定义：
欧拉路径：每条边都经过且只经过一边
欧拉回路：欧拉路径的起点终点相同

判断是否是欧拉回路：
无向图：每个顶点的度数为偶数
有向图：每个顶点的入度出度相等
混合图：将无向边拆成2条有向边，用有向边的方法做

求路径：
1.若此点无边，将此点加入队列。将栈顶元素d出栈，做d。
2.若此点有边，将此点入栈。任选一边，此边的终点为d，做d。删除此边。
将队列倒序输出即为欧拉回路路径。

输出边序就按删边的次序来。
输出字典序
  若是点序，只需在add之前对边排序，关键字是这条边的终点。
  若是边序，这个时候边表方便，先输出小的。
*/
void oula(int u){
    For(i,e[u].size())if (e[u][i].f){
        e[u][i].f=0;
        oula(e[u][i].v);
    }
    q[++top]=u;
}
```

2.3 LCA

2.3.1 倍增

```
void dfs(int u,int d){
    if (d) rep(i,1,20) fa[u][i]=fa[fa[u][i-1]][i-1];
    deep[u]=d;
    For(i,e[u].size()){
        fa[e[u][i]][0]=u;
        dfs(e[u][i],d+1);
    }
}

int main(){
    cin>>n;
    rep(i,1,n-1){
        cin>>u>>v;
        e[u].PB(v);
    }
    dfs(1,0);
    cin>>a>>b;
    if (deep[a]<deep[b]) swap(a,b);
    d=deep[a]-deep[b];
    per(i,20,0) if (d>=g[i]){
        d-=g[i];
        a=fa[a][i];
    }
    per(i,20,0) if (fa[a][i]!=fa[b][i]){
        a=fa[a][i];
        b=fa[b][i];
```

```
    }
    if (a!=b) a=fa[a][0];
    cout<<a<<endl;
}
```

```
procedure bfs;
var     l,r,y,u,v,i,j:longint;
begin
        l:=0; r:=1;
        qq[1]:=1; dp[1]:=1;
        while l<r do begin
                inc(l); u:=qq[l];
                y:=h[u];
                while y>0 do begin
                        v:=e[y].v;
                        if f[u,0]<>v then begin
                                dp[v]:=dp[u]+1;
                                f[v,0]:=u;
                                inc(r); qq[r]:=v;
                        end;
                        y:=e[y].n;
                end;
        end;

        for i:=1 to n+1 do
        for j:=1 to lg do f[qq[i],j]:=f[f[qq[i],j-1],j-1];
end;

function lca(a,b:longint):longint;
var     i,tmp:longint;
begin
        if dp[a]<dp[b] then begin tmp:=a; a:=b; b:=tmp; end;
        for i:=lg downto 0 do if dp[f[a,i]]>=dp[b] then a:=f[a,i];
        for i:=lg downto 0 do if f[a,i]<>f[b,i] then begin a:=f[a,i]; b:=f[b,i]; end
           ;
        if a<>b then exit(f[a,0]) else exit(a);
end;
```

2.3.2 tarjan

```
void tarjan(int u){
    For(i,e[u].size()){
        int v=e[u][i];
        tarjan(v);
        fa[v]=u;
    }
    done[u]=1;
    For(i,qe[u].size()){
        int v=qe[u][i];
        if (done[v]) lca[u][v]=gf(v);
        else qe[v].push_back(u);
    }
}

int main(){
    cin>>n>>m;
    rep(i,1,n-1){
        cin>>u>>v;
        e[u].push_back(v);
        e[v].push_back(u);
    }
    rep(i,1,n) fa[i]=i,done[i]=0;
    rep(i,1,m){
        cin>>u>>v;
        qe[u].push_back(v);
    }
    tarjan(1);
}
```

2.4 RMQ

2.4.1 ST

```
uses math;
var
  a:array[1..200000] of longint;
  f:array[1..200000,0..19] of longint;
  d:array[0..19] of longint;
  n,i,j,m,k,l,r:longint;

begin
  readln(n);
  for i:=1 to n do read(a[i]);

  d[0]:=1;
  for i:=1 to 19 do d[i]:=d[i-1]*2;            // d[i]=2^i
   //----------------------
  for i:=1 to n do f[i,0]:=a[i];               // ready
  for j:=1 to trunc(ln(n)/ln(2)) do            // ln(n)/ln(2)=log2(n)
    for i:=1 to n-d[j]+1 do
      f[i,j]:=max(f[i,j-1],f[i+d[j-1],j-1]);
   //---------------------
  readln(m);
  for i:=1 to m do begin
        readln(l,r);
        k:=trunc(ln(r-l+1)/ln(2));             // 2^k<=r-l+1 but 2*2^k>r-l+1
```

```
        writeln(max(f[l,k],f[r-d[k]+1,k]));
  end;
end.
```

2.4.2 线性

```
var
  i,n,top,tot,l,r,y,m:longint;
  fa,a,q,ans,h:array[0..100000]of longint;
  e:array[0..250000]of record v,n,xu:longint; end;

function gf(u:longint):longint;
begin
  if fa[u]=u then exit(u);
  fa[u]:=gf(fa[u]);
  exit(fa[u]);
end;

procedure add(u,v,xu:longint);
begin
  inc(tot);
  e[tot].v:=v;
  e[tot].xu:=xu;
  e[tot].n:=h[u];
  h[u]:=tot;
end;

begin
  readln(n,m);
  for i:=1 to n do readln(a[i]);
  for i:=1 to m do begin
        readln(l,r);
        add(r,l,i);
  end;
  for i:=1 to n do fa[i]:=i;

  for i:=1 to n do begin
        while (top>0)and(a[q[top]]<=a[i]) do begin   //以最大值为例，若要求最小值，只需将此处<=改成>=，程序的其他地方无需变动
                fa[q[top]]:=i;
                dec(top);
        end;
        inc(top);
        q[top]:=i;
  //---------------------------
        y:=h[i];
        while y>0 do begin
                ans[e[y].xu]:=a[gf(e[y].v)];
                y:=e[y].n;
            end;
  end;
  for i:=1 to tot do writeln(ans[i]);
end.
```

2.5 最短路

2.5.1 SPFA

```
int n,q[N],dist[N],f[N];
struct edge{int v,l;};
vector<edge> e[N];

void spfa(int uu){
    For(i,n+2) dist[i]=oo,f[i]=0;
    int l=0,r=1,u,v;
    q[1]=uu; dist[uu]=0;
    while (l!=r){
        l=(l%n)+1;
        f[u=q[l]]=false;
        For(i,e[u].size()){
            v=e[u][i].v;
            if (dist[v]>dist[u]+e[u][i].l){
                dist[v]=dist[u]+e[u][i].l;
                if (!f[v]){
                    r=(r%n)+1;
                    f[q[r]=v]=true;
                }
            }
        }
    }
}
```

2.5.2 dijkstra

```
int n,m,u,v,c,tot,d,dd,dist[N],num[N],lnk[N],heap[N],done[N];
struct edge{int v,l;};
vector<edge> e[N];

void up(int p){
    while(p>1 && heap[p]<heap[p>>1]){
        swap(heap[p],heap[p>>1]);
        swap(lnk[num[p]],lnk[num[p>>1]]);
        swap(num[p],num[p>>1]);
        p>>=1;
    }
```

```
}
void down(int p){
    while(p<=(tot>>1)){
        int k=p<<1;
        if (k<tot && heap[k]>heap[k+1]) k++;
        if (heap[p]<=heap[k]) return;
        swap(heap[p],heap[k]);
        swap(lnk[num[p]],lnk[num[k]]);
        swap(num[p],num[k]);
        p=k;
    }
}

int main(){
//freopen("test.in","r",stdin);
while(cin>>n>>m,n&&m){
    rep(i,1,n) e[i].clear();
    rep(i,1,m){
        cin>>u>>v>>c;
        e[u].push_back(edge(v,c));
        e[v].push_back(edge(u,c));
    }
    rep(i,1,n) dist[i]=heap[i]=oo;
    rep(i,1,n) done[i]=0;
    dist[1]=heap[1]=0;
    tot=n;
    rep(i,1,n) num[i]=lnk[i]=i;//num[i]:堆中标号为i的点在图中是哪个点 lnk[i]:图中标号为i的点在堆中的标号是多少
    rep(i,1,n){
        done[u=num[1]]=1;
        dist[u]=d=heap[1];
        heap[1]=heap[tot]; num[1]=num[tot]; lnk[num[tot--]]=1;
        down(1);
        For(j,e[u].size()){
            v=e[u][j].v; dd=d+e[u][j].l;
            if (!done[v] && heap[lnk[v]]>dd){
                heap[lnk[v]]=dd;
                up(lnk[v]);
            }
        }
    }
    cout<<dist[n]<<endl;
}
}
```

2.6 最小生成树

2.6.1 prim

2.6.2 kruskal

```
int n,m,ans,fa[N];
struct edge{int u,v,l;}e[M];
int main(){
    cin>>n>>m;
    ans=0;
    rep(i,1,m) cin>>e[i].u>>e[i].v>>e[i].l;
    sort(e+1,e+m+1);
    rep(i,1,n) fa[i]=i;
    rep(i,1,m) if (gf(e[i].u)!=gf(e[i].v)){
        fa[gf(e[i].v)]=gf(e[i].u);
        ans+=e[i].l;
    }
    cout<<ans<<endl;
}
```

2.7 最小树形图

```
可以认为是最小生成树的有向边版本。

有固定根的最小树形图求法O(VE)：
消除自环，显然自环不在最小树形图中。
判定是否存在最小树形图，以根为起点DFS一遍即可。

cost为最小树形图总权值。A为最短弧集合(有向)

对所有除源点外的节点v，找到一条以v为终点的边e，e是v的所有入边中最小的边。(记pre[v]为e的起点，minl[v]为e的权值。此时e就在A中了。)
若A中有环就把环缩成一个点，重新构造A。无则inc(cost,A的权值和)，输出cost。
缩点和A的构造：
假设环中的点为集合vi，缩成点v。对所有不在环上的点u：
dist[u,v]=min{ dist[u,vi]-minl[vi] }
dist[v,u]=min{ dist[vi,u]          }
pre[u]=v  (if pre[u]=vi)
另外inc(cost,minl[vi])

找环O(V)，收缩O(E)，总复杂度O(VE)。


对于不固定根的最小树形图，新加一个点，和每个点连权相同的边，这个权大于原图所有边权的和，这样这个图固定跟的最小树形图和原图不固定跟的最小树形图就是对应的了。

课题：PKU3164
```

```

补充构造A的正确性：
出边不变，入边的权要减去minl[vi]?
对于新图中的最小树形图T，设指向v的边为e。将v展开以后，e指向了一个环。假设原先e是指向v1的，这个时候我们将环上指向v1的边minl[v1]删除，这样就得到了原图中的一个树形图。我们会发现，如果新图中e的权w'(e)是原图中e的权w(e)减去minl[v1]权的话，那么在我们删除掉minl[v1]，并且将e恢复为原图状态的时候，这个树形图的权仍然是新图树形图的权加环的权，而这个权值正是最小树形图的权值。所以在展开节点之后，我们得到的仍然是最小树形图。逐步展开所有的人工节点，就会得到初始图的最小树形图了。
```

```cpp
//hdu49666
struct node{
    int u,v,l;
}e[10000];
const int oo=10000000;
int nn,m,mm,x,c1,c2,l1,l2,cost,sum[1000],minl[1000],pre[1000],id[1000],vis[1000];

inline void add(int u,int v,int l){
    e[++m].u=u;
    e[m].v=v;
    e[m].l=l;
}

int zhuliu(int root,int n,int m) {
    int u,v,cost=0;
    while(1){
        pre[root]=0;
        rep(i,1,n) minl[i]=oo;
        rep(i,1,m){
            u=e[i].u; v=e[i].v;
            if (e[i].l<minl[v] && u!=v){
                pre[v]=u;
                minl[v]=e[i].l;
            }
        }
        rep(i,1,n) if (i!=root && minl[i]==oo) return -1;
        int nn=minl[root]=0;
        rep(i,0,n) id[i]=vis[i]=-1;

        rep(i,1,n){
            cost+=minl[i];
            for(v=i;v && vis[v]==-1;v=pre[v]) vis[v]=i;
            if (v && vis[v]==i){
                id[v]=++nn;
                for(u=pre[v];u!=v;u=pre[u]) id[u]=nn;
            }
        }
        if (!nn) break;
        rep(i,1,n) if(id[i]==-1) id[i]=++nn;
        int j=1;
        rep(i,1,m){
            v=e[i].v;
            e[j].u=id[e[i].u];
            e[j].v=id[e[i].v];
            if (e[j].u!=e[j].v){
                e[j].l=e[i].l-minl[v];
                j++;
            }
        }
        m=j-1; n=nn;
        root=id[root];
    }
    return cost;
}

int main(){
    while (1){
        scanf("%d%d",&nn,&mm);
        if (nn==0 && mm==0) break;
        sum[0]=1; m=0;
        rep(i,1,nn){
            scanf("%d",&x); x++;
            sum[i]=sum[i-1]+x;
            rep(j,1,x-1) add(sum[i-1]+j+1,sum[i-1]+j,0);
            add(1,sum[i-1]+1,0);
        }
        rep(i,1,mm){
            scanf("%d%d%d%d%d",&c1,&l1,&c2,&l2,&cost);
            add(sum[c1-1]+l1+1,sum[c2-1]+l2+1,cost);
        }
         printf("%d\n",zhuliu(1,sum[nn],m));
    }
}
```

2.8 有向图强连通分量

```cpp
void tarjan(int u){
    low[u]=dfn[u]=++cc;
    q[++top]=u;
    instack[u]=true;
    int v;
    For(i,e[u].size()){
        v=e[u][i];
        if (!dfn[v]){
            tarjan(v);
            upmin(low[u],low[v]);
```

```
        }
        else if (instack[v]) upmin(low[u],dfn[v]);
    }

    if (dfn[u]==low[u]) do{
        v=q[top--];
        instack[v]=false;
        //v被缩到u中, 更新信息, 如po[v]=u;
    } while (u!=v);
}

int main(){
    rep(i,1,n) if (!dfn[i]) tarjan(i);
}
```

```
procedure tarjan(uu:longint);
var
  v,top,i,tt:longint;
  flag:boolean;
begin
  top:=1; d[top]:=uu;
  while top>0 do begin
        u:=d[top];
        flag:=true;
        if dfn[u]=0 then begin
                inc(cc); dfn[u]:=cc; low[u]:=cc;
                inc(tot); q[tot]:=u;
                f[u]:=true; ff[u]:=true;
                y[top]:=h[u];
        end
        else begin low[u]:=min(low[u],low[e[y[top]].v]); y[top]:=e[y[top]].n; end;
        while y[top]>0 do begin
                v:=e[y[top]].v;
                if not f[v] then begin
                        inc(top);
                        d[top]:=v;
                        flag:=false;
                        break;
                end
                else if ff[v] then low[u]:=min(low[u],dfn[v]);
                y[top]:=e[y[top]].n;
        end;
        if flag then begin
              if dfn[u]=low[u] then begin
                      tt:=0;
                      repeat
                              v:=q[tot];
                              dec(tot);
                              ff[v]:=false;
                              {here you can give the information to u}
                      until u=v;
              end;
              dec(top);
        end;
  end;
end;

begin
  for i:=1 to n do if dfn[i]=0 then tarjan(i);
end.
```

2.9 哈密顿回路

哈密顿回路：每个点都经过一次但不重复，最后才回到的起始点的回路。显然回路边数=n。这是个npc问题。特殊哈密尔顿回路：

在任意一个无向图中，只要满足任意两个不同的点p和q，D(p) + D(q) > N，这个图上就一定存在哈密尔顿回路，同时存在一个时间复杂度为O(N2)的算法可以构造出这条回路。其中D(p)表示p点在图中的度。

解法：

任取一个点，视为长度为1的链。

对于一条长度为m的链p1→p2□→pm

1)若剩下的点中存在点q有边q→p1或pm→q就加进来

2)否则说明p1与pm相连的点都在p1~pm内，但二者度数都大于n/2，根据抽屉原理必存在(p[1]-p[k+1]且p[k]-p[m])，于是将其改造成环。此时从环上任引一边至剩余点即可。

裸题：sgu122

```
//sgu122
var
      u,uu,v,y,n,m,et,g,i:integer;
      q,qq:array[0..10000]of integer;
      f,h:array[0..10000]of boolean;
      e:array[0..1010,0..1010]of boolean;
      flag:boolean;

begin
assign(input,'122.in'); reset(input);
assign(output,'122.out'); rewrite(output);
      readln(n);
      for u:=1 to n do begin
            while not eoln do begin
                  read(v);
                  e[u,v]:=true; e[v,u]:=true;
            end;
            readln;
            h[u]:=true;
      end;

```

```
    f[1]:=true; q[1]:=1; m:=1;
    for g:=1 to n-1 do begin
        flag:=false;
        u:=q[m];
        for v:=1 to n do if h[u] and e[u,v] and not f[v] then begin
            inc(m); q[m]:=v; f[v]:=true;
            flag:=true; break;
        end;
        if flag then continue;
        u:=q[1];
        for v:=1 to n do if h[u] and e[u,v] and not f[v] then begin
            inc(m); for i:=m downto 2 do q[i]:=q[i-1];
            q[1]:=v; f[v]:=true;
            flag:=true; break;
        end;
        if flag then continue;

        for uu:=2 to m-1 do if not e[q[1],q[m]] and e[q[m],q[uu]] and e[q[1],q[uu+1]] then begin
            for i:=1 to uu do qq[i]:=q[i];
            for i:=uu+1 to m do qq[m-i+uu+1]:=q[i];
            for i:=1 to m do q[i]:=qq[i];
            break;
        end;

        for uu:=2 to m-1 do if not flag then begin
            u:=q[uu]; if not h[u] then continue;
            for v:=1 to n do if e[u,v] and not f[v] then begin
                for i:=uu to m do qq[i-uu+2]:=q[i];
                for i:=1 to uu-1 do qq[m-uu+2+i]:=q[i];
                inc(m);
                for i:=2 to m do q[i]:=qq[i];
                q[1]:=v; f[v]:=true;
                flag:=true; break;
            end;
            if not flag then h[u]:=false;
        end;
    end;

    for uu:=2 to m-1 do if not e[q[1],q[m]] and e[q[m],q[uu]] and e[q[1],q[uu+1]] then begin
            for i:=1 to uu do qq[i]:=q[i];
            for i:=uu+1 to m do qq[m-i+uu+1]:=q[i];
            for i:=1 to m do q[i]:=qq[i];
            break;
        end;
    for u:=1 to m do if q[u]=1 then begin
        for i:=u to m do write(q[i],' ');
        for i:=1 to u-1 do write(q[i],' ');
        //for i:=u downto 1 do write(q[i],' ');
        //for i:=m downto u+1 do write(q[i],' ');
        writeln(1);
    end;
close(output);
end.
```

2.10 关键路径

```
带权有向图，AOE网。
AOE网的性质：
一个顶点发生了，说明之前的活动都已完成，之后的活动可以开始。(不冲突的点可以同时发生)

★【要求的值】：
1.完成整个工程需要多少时间？
2.哪些活动是关键活动，会影响到工程进度？(因此只有优化关键活动才能缩短工程时间)

记：
ve[i]表示i点的最早发生时间。vl[i]表示i点的最迟发生时间。      //early, last
ee[i]表示i边的最早发生时间。el[i]表示i边的最迟发生时间。

因此得出：
ve[1]=0;
ve[v]=max{ve[u]+d[u,v]}(u为v的所有父亲)              //d[u,v]表示边<u,v>的长，可用边表
vl[n]=ve[n]                                         //显然的
vl[u]=min{vl[v]-d[u,v]}(v为u的所有孩子)
对于y边连接着u和v
ee[y]=ve[u];  el[y]=vl[v]-d[u,v]

关键路径：
完成工程的最短时间是从开始点到完成点的最长路径长度，此路径就称之为关键路径。(回答问题一)(可能不止一条)

关键活动：
若ee[y]=el[y]则y为关键边。
若ve[i]=vl[i]则i为关键点。


算法总结：
1.输入e条弧，建立AOE网.
2.得出拓扑序，若无法拓扑说明有环，问题不可解，halt。
3.初始化ve[1]=0,按拓扑序更新ve
4.初始化vl[n]=ve[n]，按逆拓扑序更新vl                  //无需重求，逆序做即可(自己证)
5.求出ee[y]和el[y]
```

```
6.输出关键路径

算法简化：
两个关键点之间必为关键边，关键边的两头必然为2个关键点。                //自己想
因此若无要求可以简化第5条。不必求ee[y]与el[y]。
直接这样：for i:=1 to n do if ve[a[i]]=vl[a[i]] then write(a[i],'->');   //a是拓扑序列

算法代码：
var
  i,j,n,m,u,v,l:longint;
  a,vl,ve,into:array[0..1000]of longint;
  d:array[0..1000,0..1000]of longint;
function toper:boolean;
var
  i,j,k:longint;
begin
  for i:=1 to n do for j:=1 to n do if d[i,j]>=0 then inc(into[j]);
  for i:=1 to n do begin
        for j:=1 to n+1 do if into[j]=0 then break;
        if j=n+1 then exit(false);
        a[i]:=j; into[j]:=-1;
        for k:=1 to n do if d[j,k]>=0 then dec(into[k]);
  end;
  exit(true);
end;

begin
assign(input,'guan.in'); reset(input);
assign(output,'guan.out'); rewrite(output);
  readln(n,m);
  fillchar(d,sizeof(d),200);
  for i:=1 to m do begin
        readln(u,v,l);
        d[u,v]:=l;
  end;
  if not toper then begin writeln('no solution!'); close(output); halt; end;
  ve[1]:=0;
  for i:=2 to n do
    for j:=1 to i-1 do if (d[a[j],a[i]]>=0)and(ve[a[i]]<ve[a[j]]+d[a[j],a[i]]) then
    ve[a[i]]:=ve[a[j]]+d[a[j],a[i]];
  fillchar(vl,sizeof(vl),100);
  vl[n]:=ve[n];
  for i:=n-1 downto 1 do
    for j:=i+1 to n do if (d[a[i],a[j]]>=0)and(vl[a[i]]>vl[a[j]]-d[a[i],a[j]]) then
    vl[a[i]]:=vl[a[j]]-d[a[i],a[j]];

  writeln(ve[n]);
  for i:=1 to n-1 do if ve[i]=vl[i] then write(a[i],'->');    //这里输出的不是关键路径。有肯能：1->2->4,1->3->4这个长度相等都是，而这里会输成1->2->3->4具体自己解决
  writeln(a[n]);
close(output);
end.
```

2.11 割点割边

```
int n,ans,u,v,low[N],dfn[N],cut[N],rt_son,sign;
vector<int> e[N];
void dfs(int u){
    low[u]=dfn[u]=++sign;
    For(i,e[u].size()){
        int v=e[u][i];
        if (!dfn[v]){
            dfs(v);
            if (u==1) rt_son++;
            upmin(low[u],low[v]);
            if (low[v]>=dfn[u]) cut[u]=1;
            if (low[v]>dfn[u]) uv是割边;
        }
        else upmin(low[u],dfn[v]);
    }
}

int main(){
    ans=sign=rt_son=0;
    rep(i,1,n) dfn[i]=cut[i]=0;
    dfs(1);
    if (rt_son<2) cut[1]=0;//有两个子树的根是割点
    rep(i,1,n) ans+=cut[i];
    cout<<ans<<endl;
}
```

2.12 二分图匹配

```
//记link[i]表示右列i目前连着的点，没有时连0
int find(int x){
    rep(i,1,m) if (b[x][i] && !cover[i]){
        int v=link[i];
        link[i]=x; cover[i]=1;
        if (!v || find(v)) return 1;
        link[i]=v;
    }
    return 0;
}
```

```
//调用时：
rep(i,1,n){
    memset(cover,0,sizeof(cover));
    find(i);
}
//ans:
rep(i,1,m) if (link[i]) ans++;
```

2.13 网络流

2.13.1 上下界

```
问题类型：
一个图，给出每条边的上下界的流量限制，求[可行流]or[最大流]or[最小流]

做法：
1.连接T->S，下界0上界oo(或者按题目要求的其他值)，原题转换为无源汇点的XX流。

2.将每条边拆解成2条边，下界都为0，a边上界=原边下界，b边上界=原边上界-下界，这样a+b的
  流量=原边流量。称a为必要弧，现在必须要求a全满。当然T->S的那条不用分。

3.添加X与Y点，进一步拆分。对于每条a边u->v，变成u->X->Y->v,流量不变。

4.此时擦去X->Y的所有边，这样Y变成源，X变成汇，做最大流，此时的最大流满足1.必要弧为满2
  .满足容量上界限制3.保持流平衡。

5.将边u->X和Y->v还原成a边，这样便得到一个满足上下界限制的可行流。求可行流问题解决。此
  可行流流量可以O(1)得到，即T->S的流量。

注意！！原问题可行流存在无解情况，即4得到的最大流不能把源(或汇)相连的弧饱和，即得到
  的最大流!=所有必要弧的和。

6.去掉必要弧，对剩下的图做最大流，再重新把必要弧加回去就是最大流了。最小流只需改成对
  剩下的图以T为源，S为汇做最大流即可。实现方法是删除X与Y,删除T->S的边，做最大流，得
  流量，再加上刚才得到的可行流流量即可。
```

```
//bzoj2502: 清理雪道  2011福建集训

void add(int u,int v,int l){
    e[++tot].v=v;
    e[tot].l=l;
    e[tot].n=h[u];
    e[tot].b=tot+1;
    h[u]=tot;

    e[++tot].v=u;
    e[tot].l=0;
    e[tot].n=h[v];
    e[tot].b=tot-1;
    h[v]=tot;
}

int sap(int u,int flow){
    if (u==tt) return flow;
    int sapp=0;
    for (int y=h[u];y;y=e[y].n){
        int v=e[y].v;
        if (v<=up && g[u]==g[v]+1 && e[y].l){
            int ff=sap(v,min(flow-sapp,e[y].l));
            sapp+=ff;
            e[y].l-=ff;
            e[e[y].b].l+=ff;
            if (sap==flow) return sapp;
        }
    }
    if (g[ss]==up) return sapp;
    if (--vg[g[u]]==0) g[ss]=up;
    vg[++g[u]]++;
}

int main(){
freopen("2502.in","r",stdin);
freopen("2502.out","w",stdout);
    cin>>n;
    ss=0; tt=n+1; x=n+2; y=n+3;
    rep(u,1,n){
        add(ss,u,oo);
        add(u,tt,oo);
        cin>>m;
        rep(i,1,m){
            cin>>v;
            add(u,v,oo);
            add(u,x,1);
            add(y,v,1);
        }
    }
    add(tt,ss,oo);

    ss=y; tt=x; up=n+3;
    g[ss]=1; vg[1]=1; vg[0]=oo;
    while (g[ss]<up) sap(ss,oo);
    ans=e[tot].l;

    e[tot].l=e[tot-1].l=0;
    ss=n+1; tt=0; up=n+1;
    For(i,n+10) g[i]=vg[i]=0;
```

```
    g[ss]=1; vg[1]=1; vg[0]=oo;
    while (g[ss]<up) ans-=sap(ss,oo);

    cout<<ans<<endl;
}
```

2.13.2 平面图最小割

```
1.建立原图
2.添加一条S->T的边，画在最外面，边权无穷大
3.以面为点，对每条边左右两边的点建边，新边权=老边权
4.以S->T为一边的点作为S，无边界的作为T
5.求最短路，即最小割，即最大流
```

2.13.3 费用流

```
int d[],h[],q[],f[],ans,tot,ss,tt;
struct edge{int l,v,n,c;}e[];
struct PRE{int v,e;}pre[];
int spfa(){
    int l,r,u,v,y,k;
    rep(i,ss,tt){
        d[i]=oo;
        f[i]=pre[i].e=0;
        pre[i].v=-1;
    }
    l=0,r=1;
    q[1]=ss; d[ss]=0;
    while(l<r){
        u=q[++l]; f[u]=0;
        for(y=h[u];y;y=e[y].n){
            v=e[y].v;
            if (e[y].l>0 && relax(u,y,v) && !f[v]){
                f[v]=1;
                q[++r]=v;
            }
        }
    }
    if (d[tt]>=oo) return 0;
    for(k=oo,v=tt;pre[v].v!=-1;v=pre[v].v) upmin(k,e[pre[v].e].l);
    for(v=tt;pre[v].v!=-1;v=pre[v].v){
        y=pre[v].e;
        ans+=e[y].c*c;
        e[y].l-=k; e[y^1].l+=k;
    }
    return 1;
}

int main(){
    //建边，tot=1
    while (spfa()){}
    cout<<ans<<endl;
}
```

2.13.4 最小路径覆盖

```
void dfs(int u){
    int y,v;
    cout<<u<<' ';
    for(y=h[u];e[y].l!=0;y=e[y].n);
    v=e[y].v;
    if (v==0) return;
    dfs(v-n);
}

int main(){
    ss=0,tt=2*n+1;
    //建边，tot=1
    ans=n;
    g[ss]=1; vg[1]=1; vg[0]=oo;
    while(g[ss]<tt) ans-=sap(ss,oo);

    rep(u,n+1,2*n){
        for(y=h[u];e[y].v!=tt;y=e[y].n);
        if (e[y].l==1){
            dfs(u-n);
            cout<<endl;
        }
    }
    cout<<ans<<endl;
}
```

2.13.5 tips

```
网络流中为了限制点的访问次数，可以将每个点分割成2个点<i.a>和<i.b>。然后在这2个点之间
  连上界为访问次数的边。欲将i,j连边即将<i.b>与<j.a>相连。建立二分图去做。
如果涉及到点权，可以转化为边权，若是不行，也可按此法分割点，然后连上上界无穷大，费用
  为点权的边，而题目中的其他限制则是用上界控制，而费用为0。做最小（大）费用最大流即
  可。

有些问题叫你求最大值，但有一些限制，比如相邻的不能取，那么建立二分图，将相邻的连边，
  即将那些题目说不能的都连上，然后求出最小割，这样这些边都被割开了，两个集合分开
  了，不会产生任何矛盾，剩下的就都取了。接着ans就是全部的值-最小割，而最小割=最大
  流，只要求最大流就行了。
```

```
算法就是将所有矛盾关系化为二分图中的边，然后最大流=最小割，再用所有关系的和-最小割，保证了矛盾关系不被建立。
当然最小覆盖点集就是最小割了，因为二分图中一对矛盾关系中只能取一个，于是S->x=a[x],y->T=a[y],x->y=oo;这样得到的就是最小的那条边，最后减的也是他。

【1】
求：割边最少的割集中的权值最小的割

每条边的权值w=w+maxw*m
做一遍最大流

原因：这样边权变换后多一条边一定亏



【2】
求：权值最小的割集中割边最少的割

方法1：每条边的权值w=w*(m+1)+1
这样得到最大流=maxflow div (m+1),最少割边数=maxflow mod (m+1)

原因：如果原先两个割都是最小割，那么求出的最大流相等，但边权变换后只有边数小的才是最小割了。
乘(m+1)是为了保证边数叠加后依然是余数，不至于影响求最小割的结果.

方法2：建图，得到最大流后，图中边若满流，说明该边是最小割上的边
再建图，原则：满流的边改为容量为1的边，删除未满流的边，然后最大流即答案

原因：两个结论：
1.满流边一定是某个最小割的边
2.最小割集的边集中任取一个割一定也满足最小割



【3】
求：权值最小的割集中费用最少的割

由【2】的方法2得到启发。最大流后取出满流边，以费用为容量做一次最大流。

原因同【2】

推广：【2】是【3】的特例，即【2】的费用为1



【4】
求：费用最少的割集中权值最小的割

做法同【3】，将费用看做权值，权值看做费用。真相是就是两个关键字。


【5】
求：三个关键字最小的割

不会


【6】
求：权值最小的割集中割边最少的割集中字典序最小的割

是【5】的特例。解题报告：http://hi.baidu.com/mengyun1993/blog/item/c30d193c9a85932870cf6cda.html
```

3 数学

3.1 快速幂

```
int power(int a,int b,int mo){
    int k=a,ret=1;
    for(;b;b>>=1){
        if(b&1) ret=ret*k%mo;
        k=k*k%mo;
    }
    return ret;
}
```

3.2 线性筛法

```
void shai(int n){
    rep(i,1,n) f[i]=i;//f[i]表示i的最小质因数
    rep(i,2,n){
        if (f[i]==i) a.push_back(i);//a是素数表
        for(int j=0;j<a.size() && a[j]*i<=n;j++){
            f[a[j]*i]=a[j];
            if (i%a[j]==0) break;
        }
    }
}
```

3.3 容斥

```
给出N个整数A1, A2, □□, AN, 统计满足i<j且Ai和Aj互质的二元组(i, j)的数量。
【【题解】】:
把每个数分解质因数, 比如分解后有2,3,5, 那么给记录有2的+1, 有3的+1, 有5的+1, 有2又有3的+1□□
```

```
一个处理就是w就是记录这个的数组。那么很显然2记在w[2]里，3记在w[3]里，又有2又有3的就可以记在w[2*3]，就是w[6]里。这样一一对应了。
现在假如加入a[i]=3*5*7(省略了多次幂，比如(3^2)*(5^4)*(7^3)也当做3*5*7处理)，那么i之前的数中不与a[i]互质的数就有w[3]+w[5]+w[7]-w[15]-w[21]-w[35]+w[105]个，就是上面的容斥了。
做法是枚举长度为3的二进制，100表示3,010表示5,001表示7,101表示21□□然后再计算这个二进制中有多少1，odd一下判断是+还是-就好了。
int main(){
    shai(1000000);
    g[0]=1; rep(i,1,30) g[i]=g[i-1]*2;
    cin>>n;
    rep(i,1,n){
        cin>>x;
        ans+=i-1;
        now.clear();
        for(;x>1;x/=f[x]) if (now[now.size()-1]!=f[x]) now.push_back(f[x]);
        rep(opt,1,g[now.size()]-1){
            num=1; cnt=k=0;
            for(x=opt;x;x>>=1,k++)if (x&1){
                num*=now[k];
                cnt++;
            }
            if (cnt&1) ans-=w[num]++;
            else ans+=w[num]++;
        }
    }
    cout<<ans<<endl;
}
```

3.4 拓扑

```
int tp(){
    int l=0,r=0;
    rep(i,1,n) if (!ru[i]) q[++r]=i;
    while(l<r){
        int u=q[++l];
        For(i,e[u].size()){
            int v=e[u][i];
            if ((--ru[v])==0) q[++r]=v;
        }
    }
    return r==n;
}
判断图中是否有环，如果有环那么途中会剩余一些点和边的。
(判断图中是否有负权环的是Bellman-ford)
有唯一拓扑序当且仅当任何时刻入度=0，出度>0的点只有一个。
```

3.5 斜率优化

```
够陡，小的赢；很平，大的赢。于是i若不在凸壳上，于是向上论陡比不过i+1,向下论平还是比不过i+1，于是可剔除。
若i,j在凸壳上(i>j)，则j的优势是向下更平，i的优势是向上更陡。举例：a>i>j>b，则j可打败b但i不一定能打败b，j可以更优；i可打败a但j不一定，i可以更优。所以凸壳上的点都是有意义的。

（P.S.此处用以说明的图像是(5,0)-->(3,1)-->(2,5)-->(1,100)这样的方向）
```

3.6 polya

```
这个看小黑书最好。（P245）
心得：

【burnside引理】
方案数=每种置换下 不变的染色方案数 的平均值。
即L=(1/|G|)*sigma(c(f))
效率O(nsp)，n是所有方案数，超级大。s是置换数，p是格子数。

【polya定理】
定理：如果用k种颜色给有限集S着色，那么对于一个置换f，在该置换下不变的置换方案数c(f)=k^(m(f))，其中m(f)=置换f的循环节数。
代入burnside得到L=(1/|G|)*sigma(k^m(f))
效率O(sp)

【优化】

旋转置换：
给一串n个珠子染色，共有m种颜色，求本质不同的方案数。
解法：
转0格,转1格□□转n-1格，其中循环数分别为gcd(i,n)，所以方案数=(∑m^(gcd(i,n)))/n。
当然枚举i是会T的，考虑gcd(i,n)=k,则i=x*k,n=y*k,且x<=y且gcd(x,y)=1。
于是枚举n的约数k,y=n/k，求比y小的与y互质的x的个数，而这个问题就是欧拉函数。

还有很多其他优化

【黑书上的题】
一串n个珠子黑白染色，从中选出m个染成黑色，问本质不同方案数。可旋转+翻转。
```

3.7 初等数论

```
int gcd(int a,int b){
    return b?gcd(b,a%b):a;
}
```

```
//a和b的最小的正线性组合ax+by=gcd(a,b),求x,y.
int exgcd(int a,int b){
    if (!b){
        x=1,y=0;
        return a;
    }
    int gcd=exgcd(b,a%b);
    //int x1=x; x=y;
    //y=x1-a/b*y;
    swap(x,y);
    y-=a/b*x;
    return gcd;
}

//【解不定方程ax+by=c】
int bd(int a,int b,int c){
    int d=exgcd(a,b);
    if (c%d) return 0;
    x*=c/d; y*=c/d;
    return 1;
}
/*若ax+by=c有解，则d|c (d=gcd(a,b)) {这个性质很重要}。
所以若d不是c的约数，那么ax+by=c一定无解。
当d|c时，先用Euclid求出gcd(a,b)=d=ax'+by'的x'和y'，则x=x'*c/d, y=y'*c/d。

若ax+by=c有解，则有无数个解，即x+=b*c*k/d,y-=a*c*k/d, k取整数。
*/

//【计算同余方程ax≡b(mod n)】
int ty(int a,int b,int mo){return bd(a,mo,b);}//x即x

//【逆元】
int ny(int a,int mo){return bd(a,mo,1);}//x即a'
//还有费马法可做，待补

//【中国剩余定理】
x≡c1(mod b1)
x≡c2(mod b2)
……
x≡cn(mod bn)
求满足条件的x
int x,y,b1,b2,c1,c2,n,flag;
int exgcd();
int bd();
int main(){
    cin>>n;
    cin>>c1>>b1;
    rep(i,2,n){
        cin>>c2>>b2;
        if (!bd(b1,b2,c2-c1)) flag=1;
        x%=b2/d;
        c1+=x*b1;
        b1*=b2/d;
        c1=(c1+b1)%b1;
    }
    cout<<flag?-1:c1<<endl;
}

【费马小定理】：如果p是素数，a是小于p的正整数，那么a^(p-1) mod p =1。
应用：
于是随机几个a:=random(n-2)+2 来判断p是否满足。（最好是素数）（记得用快速幂）
或者用前7个素数(2, 3, 5, 7, 11, 13和17)可以搞定10e14以内的所有数。。
或者用【2,7,61】，可以搞定10e9以内的所有数。。

【欧拉函数】
n为正整数，φ(n)=不超过n且与n互质的正整数个数。

函数值：
φ(n)=n(1-1/p1)(1-1/p2)…(1-1/px),其中p1,p2…px为n的所有质因数,每种质因数只一个。
φ(1)=1。
例如12=2*2*3,φ(12)=12*(1-1/2)*(1-1/3)=4

常见性质：
φ(mn)=φ(m)φ(n)
当n为奇数时，φ(2n)=φ(n)
```

4 字符串

4.1 KMP

```
//a:文本串；b:模式串;n=a的长度;m=b的长度;ab必须从1开始
void getnext(){
    int j=0;
    next[1]=0;
    rep(i,2,m){
        while (j && b[j+1]!=b[i]) j=next[j];
        if (b[j+1]==b[i]) j++;
        next[i]=j;
    }
}

void kmp(){
    int j=0; ans=0;
    rep(i,1,n){
        while (j && a[i]!=b[j+1]) j=next[j];
```

```
        if (a[i]==b[j+1]) j++;
        if (j==m){
            ans++;
            cout<<i-m+1<<endl;
            j=next[j];
        }
    }
}
```

4.2 exKMP

```
//A[i]=T与T(i,m)的最长公共前缀, 2<=i<=m。
//B[i]=T与S(i,n)的最长公共前缀, 1<=i<=n。
void getA(){
    int j,k,len;//len:能匹配到的最远距离;len=k+A[k]-1
    for(j=0;t[1+j]==t[2+j]&&2+j<=m;j++);
    a[2]=j; k=2;
    rep(i,3,m){
        len=k+a[k]-1; j=len-i+1;
        if (a[i-k+1]<j) a[i]=a[i-k+1];
        else{
            upmax(j,0);
            for(;t[1+j]==t[i+j]&&i+j<=m;j++);
            a[i]=j; k=i;
        }
    }
}

void getB(){
    int j,k,len;
    for(j=0;t[1+j]==s[1+j]&&1+j<=m&&1+j<=n;j++);
    b[1]=j; k=1;
    rep(i,2,n){
        len=k+b[k]-1; j=len-i+1;
        if (a[i-k+1]<j) b[i]=a[i-k+1];
        else{
            upmax(j,0);
            for(;t[1+j]==s[i+j]&&1+j<=m&&i+j<=n;j++);
            b[i]=j; k=i;
        }
    }
}
```

4.3 AC 自动机

```
{hdu3065
题意：给n个模式串，一个母串，问每个模式串在母串中出现多少次。
1.模式串只包含大写字母，母串中包含各种奇葩字符。
2.空间被卡了。完全可以造出跟hdu2222一样的32M开不下的数据，还好数据比较人性化。
3.坑爹的！！！！多组数据木有声明啊有木有！！！！
}
const maxN=290000;
var
      ni,i,n,tot,len:longint;
      tr:array[0..maxN,'A'..'Z']of longint;     //trie树
      s:array[0..2000500]of char;               //模式串或母串
      qu,nt,vis:array[0..maxN]of longint;   //qu:bfs序, nt:next, vis:访问次数
      wd,lwd:array[0..2000]of longint;      //单词位置, 单词长度
      wds:array[0..2000,0..100]of char;     //记录单词, 此题输出要求, 不必理会

procedure reads;                //读入, 平常读入就好, 这里是因为此题会发生某些奇葩字
   符也出现的情况
var   ch:char;
begin
      len:=0;
      while not eoln do begin
            read(ch); if (ch<'A')or(ch>'Z') then break;
            inc(len); s[len]:=ch;
      end;
end;

procedure insert;               //插入单词, 建立trie树
var   p,i:longint;
begin
      p:=0;
      for i:=1 to len do begin
            if tr[p,s[i]]=0 then begin
                  inc(tot);
                  tr[p,s[i]]:=tot;
            end;
            p:=tr[p,s[i]];
      end;
      wd[ni]:=p;
end;

procedure build;                //bfs求next函数
var   l,r,u,v,q:longint;
      c:char;
begin
      l:=0; r:=1;
      qu[1]:=0;
      while l<r do begin
            inc(l);
            u:=qu[l];
```

```
                for c:='A' to 'Z' do if tr[u,c]>0 then begin
                        q:=nt[u]; v:=tr[u,c];
                        while (tr[q,c]=0)and(q<>0) do q:=nt[q];
                        if u=0 then nt[v]:=0 else nt[v]:=tr[q,c]; //注意要特判
                        inc(r); qu[r]:=v;
                end;
        end;
end;

procedure dos;                  //匹配
var   p,i:longint;
begin
        p:=0;
        for i:=1 to len do begin
                while (tr[p,s[i]]=0)and(p<>0) do p:=nt[p];
                p:=tr[p,s[i]]; inc(vis[p]);
        end;
end;

begin
assign(input,'3065.in'); reset(input);
assign(output,'3065.out'); rewrite(output);
while not eof do begin          //坑爹的多组数据
fillchar(nt,sizeof(nt),0);
fillchar(vis,sizeof(vis),0);
fillchar(tr,sizeof(tr),0);
tot:=0;

        readln(n);
        for ni:=1 to n do begin
                reads; readln;
                insert;
                lwd[ni]:=len;           //不必理会
                for i:=1 to len do wds[ni,i]:=s[i]; //不必理会
        end;
        build;
        while not eoln do begin
                reads;
                if len>0 then dos;
        end;
        for i:=tot+1 downto 2 do inc(vis[nt[qu[i]]],vis[qu[i]]);  //统计

        for ni:=1 to n do begin                 //奇葩的输出
                if vis[wd[ni]]=0 then continue;
                for i:=1 to lwd[ni] do write(wds[ni,i]);
                writeln(': ',vis[wd[ni]]);
        end;
end;
close(output);
end.
```

4.4 后缀数组

```
const maxl=100;
var height,sa,tmp,tax,rank:array[0..maxl*2]of longint;
t:array['@'..'z']of longint;
s:array[0..maxl]of char;
i,j,l,o,b:longint;
ch:char;

function max(a,b:longint):longint;
begin if a<b then exit(b); exit(a); end;

procedure sort(u:longint);
begin
  for j:=1 to l do tax[j]:=0;
  for j:=1 to l do inc(tax[rank[sa[j]+u]]);
  for j:=1 to l do inc(tax[j],tax[j-1]);
  for j:=l downto 1 do begin
    tmp[tax[rank[sa[j]+u]]]:=sa[j];
    dec(tax[rank[sa[j]+u]]);
  end;
  sa:=tmp;
end;

begin
assign(input,'1.in'); reset(input);
  while not eoln do begin inc(l); read(s[l]) end;
  for i:=1 to l do t[s[i]]:=1;
  for ch:='A'to 'z' do if t[ch]=1 then t[ch]:=t[pred(ch)]+1 else t[ch]:=t[pred(ch)];
  for i:=1 to l do rank[i]:=t[s[i]];
  for i:=1 to trunc(ln(2*l-1)/ln(2)) do begin
    o:=1 shl (i-1);
    for j:=1 to l do sa[j]:=j;
    sort(o);
    sort(0);
    for j:=1 to l do begin
      tmp[sa[j]]:=tmp[sa[j-1]];
      if(rank[sa[j]+o]<>rank[sa[j-1]+o]) or (rank[sa[j]]<>rank[sa[j-1]])
      then inc(tmp[sa[j]]);
    end;
    rank:=tmp;
  end;
  for i:=1 to l do if rank[i]<>l then begin
    b:=rank[i];
```

```
    o:=sa[b+1];
    while s[height[b]+i]=s[height[b]+o] do inc(height[b]);
    height[rank[i+1]]:=max(height[b]-1,0);
  end;
end.
```

4.5 后缀自动机

```
#include<iostream>
#include<cstdio>
#include<cstring>
using namespace std;
const int N = 10005;
char ch[N];
struct SAM{
    struct node{
        int ch[26];
        int f, len, u;
        void init() {
            f = -1, len = 0;
            memset(ch,-1,sizeof ch);
        }
    };
    node a[N<<1];
    int tot, last;
    void init() {
        tot = 0, last = 0;
        a[tot++].init();
    }
    int newnode() {
        a[tot].init();
        return tot++;
    }
    void add(int c,int v) {
        int u, p, q, r;
        p = newnode();
        a[p].u = v;
        u = last;
        a[p].len = a[last].len + 1;
        for ( ; u != -1 && a[u].ch[c] == -1; u = a[u].f) a[u].ch[c] = p;
        if (u == -1) a[p].f = 0;
        else {
            q = a[u].ch[c];
            if (a[u].len + 1 == a[q].len) a[p].f = q;
            else {
                r = newnode();
                a[r] = a[q];
                a[r].len = a[u].len + 1;
                a[p].f = a[q].f = r;
                for ( ; u != -1 && a[u].ch[c] == q; u = a[u].f) a[u].ch[c] = r;
            }
        }
        last = p;
    }
} sam;
int a[11] = {0, 1, 0, 1, 2, 0, 1, 0};
int main() {
    int T, i, j, l, p;
    sam.init();
    for (i = 0; i < 8; i++) sam.add(a[i], i);
    l = sam.tot;
    cout<<l<<endl;
    for (i = 0; i < l; i++)
        for (j = 0; j < 26; j++)
            if (sam.a[i].ch[j] != -1) printf("%d %d %d\n",i, j, sam.a[i].ch[j]);
    for (i = 0; i < l; i++) printf("%d %d\n",sam.a[i].f,sam.a[i].u);
    /*scanf("%d",&T);
    while (T--) {
        sam.init();
        scanf("%s",ch);
        l = strlen(ch);
        for (i = 0; i < l; i++) sam.add(ch[i] - 'a');
        for (i = 0; i < l; i++) sam.add(ch[i] - 'a');
        p = 0;
        for (i = 0; i < l; i++)
            for (j = 0; j < 26; j++)
                if (sam.a[p].ch[j] != -1) {
                    p = sam.a[p].ch[j];
                    break;
                }
        printf("%d\n", sam.a[p].len - l + 1);
    }*/
    return 0;
}
```

5 计算几何

5.1 计算几何

```
我们只考虑点与向量的运算，线等用参数方程代替。优越性在于不必考虑斜率问题。

向量的产生：
有2个点A(x1,y1),B(x2,y2),则向量A→B=(x2-x1,y2-y1)

```

```
直线的表示：
直线用一个线上的点P与一个线上的向量a表示，参数方程：(x,y)=P+t·a  （t∈R）
射线或线段只需限制t的范围即可。

向量的运算：
【模长】：|x,y|=sqrt(x^2+y^2)

【加】：(x1,y1)+(x2,y2)=(x1+x2,y1,y2)

【减】：(x1,y1)-(x2,y2)=(x1-x2,y1-y2)

【数乘】：t*(x,y)=(tx,ty)

【点积】：(x1,y1)·(x2,y2) = x1x2+y1y2 = |x1,y1||x2,y2|cosθ
    作用1：判断a是否在b的左右180°内，若是则值>0
    作用2：判断a,b是否垂直，若是则值=0
    作用3：【投影】设a在b上的投影为a’，则a’=(b/|b|)*(a·b/|b|)

【叉积】：(x1,y1)×(x2,y2) = |x1 y1| = x1y2-x2y1 = |x1,y1||x2,y2|sinθ
                            |x2 y2|

    作用1：判断a,b是否平行，若是则值=0
    作用2：计算面积。得到的值为a,b构成平行四边形的面积，亦为a,b构成三角形的面积的2倍。这里的面积都为有向面积，方向根据右手螺旋判定，a×b=k，若k>0则a→b逆时针，否则顺时针（从小于180的那个角转）。
    作用2的延伸：判断a是否在b的左边或右边。

【三维叉积】：(x1,y1,z1)×(x2,y2,z2)=(y1z2-y2z1,z1x2-z2x1,x1y2-x2y1)
    作用1：得到一个新向量c。a×b的方向遵循右手螺旋定则，从a转到b(<180)，大拇指的方向即为c的方向，同时与a,b垂直。
    作用2：c的重要性质是垂直ab平面，建系，求法向量什么的都很有用。
    作用3：判断a,b是否平行，若是则c的模长=0
    作用4：计算面积。|c|为a,b构成平行四边形的面积，亦为a,b构成三角形的面积的2倍。

【旋转】：
    将(x,y)逆时针旋转k°到(x’,y’)
    x’=x*cosk-y*sink
    y’=x*sink+y*cosk
    作用举例：重建坐标系
    有一个平面ax+by+cz=0，想要以该平面为xOy面重构坐标系。则(a,b,c)为法向量。
    先绕z轴把法向量转到yOz上，即在xOy平面上把(a,b)转到(0,sqrt(a^2+b^2)),则cosk=b/sqrt(a^2+b^2),每个点都用这个转一下，这对四个象限都有效请放心使用。然后法向量变成了(0,sqrt(a^2+b^2),c)，绕x轴再来一遍即可。

【三维绕轴旋转】
绕(0,0,0)-(a,b,c)逆时针旋转k°
[x’,y’,z’,1]=[x,y,z,1]*
[a*a*(1-ck)+ck, a*b*(1-ck)+c*sk, a*c*(1-ck)-b*sk, 0,
 a*b*(1-ck)-c*sk, b*b*(1-ck)+ck, b*c*(1-ck)+a*sk, 0,
 a*c*(1-ck)+b*sk, b*c*(1-ck)-a*sk, c*c*(1-ck)+ck, 0,
 0, 0, 0, 1]

【求直线交点】
    求直线line1[A(点)、a(向量)]，line2[P、b]的交点。
    设方程A+a·t1=P+b·t2=交点D
    设B(点)=A+a,Q=P+b
    则t1=[ (Px-Ax)·(Qy-Py)-(Py-Ay)·(Qx-Px) ] / [ (Bx-Ax)·(Qy-Py)-(By-Ay)·(Qx-Px) ]
    =cross(A,P,P,Q) / cross(A,B,P,Q)
    代入A+a·t1即得D。
    注意！在做这些操作之前先判断是否平行，若平行无交点会被零除！！

【二维化】
    在三维中若某些操作在一个平面内，而空间中不好想象，就把它摊到平面上，搞完了弄回到三维中去。


【未整理部分】
若 P × Q > 0 , 则P在Q的顺时针方向。
若 P × Q < 0 , 则P在Q的逆时针方向。
若 P × Q = 0 , 则P与Q共线，但可能同向也可能反向。

直线用两点存，可以小过程造向量

判断点P在AB上：PA×PB=0且P在AB为对角顶点的矩阵内。
注意x,y都要判，理由是水平或垂直。

判断点P在多边形内：
从P点水平向左射出一条线，枚举每条边，判断与多边形交点个数，若为奇数在多边形内，偶数在外。
为特殊情况制定规则：
1.水平边不考虑。
2.若交于某边顶点，且另一点在上方，计数。
3.若交于某边顶点，且另一点在下方，不计数。
伪代码如下：
 count ← 0;
以P为端点，作从右向左的射线L;
for 多边形的每条边s do
if P在边s上then return true;
if s不是水平的then
if s的一个端点在L上
if 该端点是s两端点中纵坐标较大的端点
then count ← count+1
 else if s和L相交then count ← count+1;
if count mod 2 = 1 then return true;
 else return false;
其中做射线L的方法是：设P’的纵坐标和P相同，横坐标为正无穷大（很大的一个正数），则P和P
```

```
  '就确定了射线L。
判断点是否在多边形中的这个算法的时间复杂度为O(n)。

判断线段是否在多边形内：
在多边形内必须满足：
1.两个点在多边形内
2.线段与每条l的交点必须是线段或l的端点上
3.给交点们排序，每相邻两点间的线段的中点在多边形内
由于交点不会很多，所以还是O(n)的。

点P到直线的距离：
直线上一点A,垂足P',l1=AP,l2=AP'
叉积法：abs(x1y2-x2y1)/sqrt(x2*x2+y2*y2)
点积法：sqrt( (x1*x1+y1*y1) - sqr(x1y1+x2y2)/(x2*x2+y2*y2) )

点P到直线的垂足：
P点+line的向量，旋转90°，做交点。

const double eps = 1e-8;
int dcmp(double c) {
    return c < -eps ? -1 : c > eps;
}
struct Point {
    double x,y;
    Point () {}
    Point (double x, double y) : x(x), y(y) {}
    Point operator + (const Point &a)const{ return Point(x+a.x,y+a.y); }
    double len() { return sqrt(x * x + y * y); }
    double len2() { return x * x + y * y; }
    void read() { scanf("%lf%lf", &x, &y); }
};
inline Point operator - (Point a, Point b) { return Point(a.x - b.x, a.y - b.y); }
inline Point operator + (Point a, Point b) { return Point(a.x + b.x, a.y + b.y); }
inline Point operator * (Point a, double b) { return Point(a.x * b, a.y * b); }
inline Point operator / (Point a, double b) { return Point(a.x / b, a.y / b); }
inline double operator % (Point a, Point b) { return (a.x * b.x + a.y * b.y); }
inline double operator * (Point a, Point b) { return a.x * b.y - a.y * b.x; }
inline double xmul(Point a, Point b, Point c) { return (b - a) * (c - a); }
```

5.2 最远点对

```
//poj2187 USACO 2003 Fall Beauty Contest
//模板改int
void hull(point a,point b,int l,int r){
    int x=l;
    rep(k,l,r) if (s[x]<s[k] || (s[x]==s[k] && p[x]<p[k])) x=k;
    point y=p[x];
    int i=l-1,j=r+1;
    rep(k,l,r) if ((s[i+1]=(y-a)*(p[k]-a))>0) swap(p[++i],p[k]);
    per(k,r,l) if ((s[j-1]=(b-y)*(p[k]-y))>0) swap(p[--j],p[k]);
    if (l<=i) hull(a,y,l,i);
    bao[++m]=y;
    if (j<=r) hull(y,b,j,r);
}

int main(){
    cin>>n;
    rep(i,1,n) p[i].read();
    rep(i,2,n) if (p[i]<p[1]) swap(p[1],p[i]);
    bao[m=1]=p[1];
    hull(p[1],p[1],2,n);

    j=2; bao[m+1]=bao[1];
    rep(i,1,m){
        while ((bao[j]-bao[i])*(bao[i+1]-bao[i]) < (bao[j+1]-bao[i])*(bao[i+1]-bao[i])) j=(j%m)+1;
        upmax(ans,(bao[j]-bao[i]).len2());
    }
    cout<<ans<<endl;
}
```

5.3 半平面交

```
//[ZJOI2008]掺望塔  BZOJ1038

int jiao(point a,point b,point p,point q){ //l1=a->b,l2=p->q
    double k=(b-a)*(q-p);
    if (dcmp(k)==0) return 0; //平行
    double t=(p-a)*(q-p)/k;
    dd=a+((b-a)*t); //dd为交点
    return eps<t && t<1-eps;
    //交点在端点看不看只用判t=0和t=1算不算。这里判断的是l2所在直线与l1线段是否相交，若二者皆为线段可以反交。
}

int main(){
    cin>>n;
    ans=oo;
    rep(i,1,n) cin>>z[i].x;
    rep(i,1,n) cin>>z[i].y;
    m=4;
    ba[1].x=z[1].x; ba[1].y=-oo; //先搞个大包
    ba[2].x=z[1].x; ba[2].y=oo;
    ba[3].x=z[n].x; ba[3].y=oo;
```

```
    ba[4].x=z[n].x; ba[4].y=-oo;
    rep(i,1,n-1){
        mm=0; ba[m+1]=ba[1];
        rep(j,1,m){
            if ((z[i+1]-z[i])*(ba[j]-z[i])>-eps) bb[++mm]=ba[j];
            if (jiao(ba[j],ba[j+1],z[i],z[i+1])) bb[++mm]=dd;
        }
        m=mm; rep(j,1,m) ba[j]=bb[j];
    }
    //半平面交结束，以下是此题蛋疼输出

}
```

6 DP

6.1 多重背包队列优化

```
void ins(int x,int y){
    while (l<=r && b[r]<=y) r--;
    a[++r]=x; b[r]=y;
}

int main(){
    cin>>n>>m; //n物品数，m背包空间
    rep(i,1,n){
        cin>>s>>v>>t; //s物品体积，v物品价值，t物品数量（0为无限制）
        if (!t || m/s<t) t=m/s;
        For(d,s){
            l=1; r=0;
            rep(j,0,(m-d)/s){
                ins(j,f[j*s+d]-j*v);
                if (a[l]<j-t) l++;
                f[j*s+d]=b[l]+j*v;
            }
        }
    }
    cout<<f[m]<<endl;
}
```

6.2 LIS

```
最长上升子序列

【朴素】
f[i]=max{f[j]}+1(j<i且a[j]<a[i])

【优化】
g[i]=min{a[j]}(f[j]=i)
即g[i]表示长度为i的上升子序列最后一个数最小是多少。
则g[1]<g[2]<...<g[k]

做到f[i]时，在g中找到>=a[i]的第一个g[j]，则f[i]:=j;   //为什么？tip1
由于g[j]>=a[i],所以g[j]:=a[i]

tip1:
因为g[j]第一个>=a[i]，所以g[j-1]<a[i],又g[1]<g[2]<...<g[k]，所以:f[i]:=f[g[j-1]]+1=j
  -1+1=j.寻找的过程可以二分查找，所以效率是nlogn

注意：f[i]表示的是a[i]一定要取的LIS，最后的ans=max{f[i]}(1<=i<=n)

若是最长不下降子序列二分查找的时候就找第一个>a[i]的g。

代码：
  for i:=1 to n do g[i]:=maxlongint;
  f[1]:=1; g[1]:=a[1]; g[0]:=0;
  for i:=2 to n do begin
        l:=0; r:=n;
        while l<=r do begin
                m:=(l+r)>>1;
                if a[i]>g[m] then l:=m+1
                else begin
                        k:=m;
                        r:=m-1;
                end;
        end;
        f[i]:=k; g[k]:=a[i];
  end;
  for i:=2 to n do if f[i-1]>f[i] then f[i]:=f[i-1];
  writeln(f[n]);

推荐：poj1836
```

6.3 树型动规

```
【【树形背包】】
给一棵树，每个点都有自己的价值，但必须取了这个点的父亲才能取这个点，问最大价值。

这里为方便讲述，树的储存为son[i,j]表示i的第j个儿子，儿子总数为son[i,0]
sc[i]表示i的价值
【nc^2】
f[i,j]表示以i为根的子树中选j个点的最大价值。
先给出程序：
```

```
procedure dfs(id:longint);
var
  i,j,k:longint;
begin
  for i:=1 to m do f[id,i]:=sc[id];          //自己一定要学
  for i:=1 to son[id,0] do begin
    dfs(son[id,i]);
    for k:=m downto 1 do                  //01背包DP:把儿子当做物品，一维数组从后往前更新
      for j:=1 to k-1 do                  //用这个儿子中选j个节点去更新自己
        tget(f[son[id,i],j]+f[id,k-j] , f[id,k]);
  end;
end;

【nc】
对于nc^2的过程，我只能说，他太傻了，但是可以借鉴，得到nc做法。
nc^2的做法就像是这样:
第一个儿子的背包                    第二个儿子的背包
                从左边挑x1个      从右边挑x2个   ----------都是枚举，c^2
                     得到本节点x1+x2的值
事实上这是状态不对，导致不能构建良好的背包模型。
f[i,j]表示访问到i点时所有访问过的点的和i的子树中选j个的最大价值。
这样就等于把上述两个表合并成一列，像普通01背包一样就可以了。
这样就成功借鉴01背包的做法，只是物品的顺序是按访问顺序来的。
可是这样不就是直接做01背包，不是不能保证父亲都有了么？
那我按访问顺序来做时干嘛用的，这样的顺序下来有一步操作就能保证，且看代码：

procedure dfs(id:longint);
var i,k,j:longint;
begin
  if id<>0 then f[id,0]:=-maxlongint;
  for i:=1 to son[id,0] do begin
    for j:=1 to m do
      f[son[id,i],j]:=f[id,j-1]+s[son[id,i]];     //【A】
    dfs(son[id,i]);
    for j:=1 to m do
      tget(f[son[id,i],j] , f[id,j]);             //【B】
  end;
end;

这里操作A和操作B可以满足很多条件，极其精妙。
首先A是一个放物品的操作，但是可以看到，对于f[son],这个son是必然要放进去的。再看B，这是当前儿子的状态去更新自己，但更好的说法是以父亲为媒介把状态传递给下一个节点。回到A，由于所有的儿子都是以父亲为媒介，以至于不论怎样，儿子的结果中一定有自己。为什么B操作不会让改变父亲一定在的性质呢？枚举到最下面，叶节点一定必取所有祖先，用叶节点去更新父亲，一定是用所有祖先及叶节点必取的状态去更新除了叶节点，所有祖先必取的状态，不论怎么说，做到谁，谁的祖先们都是一定在的，更新只是把不仅祖先必取，自己也取了的结果给父亲，所以保证了父亲节点必取。
此操作类似sap的做法。


注意一定要从dfs(0)开始，因为sc[1]是在dfs(0)的时候加进去的，所以不能省，还能合并多个森林。

注意一定要初始化，起码f[i,0]=-maxlongint;这样才保证必取祖先,但由于超级根0是不取的，所以f[0,0]=0;如果取的是点值不是边值就f[1,0]也是0总之谁不取，谁就是0.
另一种做法是：有1个不取就inc(m)，然后就可以全部变成-maxlongint了。
```

6.4 动规优化

```
2D/1D问题的状态转移方程可以考虑用四边形不等式优化
就是形如：f[i,j]:=min(f[i,k-1]+f[k,j])+w[i,j](i<k<=j)的方程
但是w[i,j]要是凸的
即对于a<=b<=c<=d，有w[a,c]+w[b,d]<=w[a,d]+w[b,c]
可以把状态转移方程优化为：
f[i,j]:=min(f[i,k-1]+f[k,j])+w[i,j](s[i,j-1]<=i<=s[i+1,j])
其中s[i,j]表示使得f[i,j]取得最优解时的决策变量

1D/1D

f[i]=min(f[k]+w[k,i])(k<i)
若w数组满足w[i,j]+w[i+1,j+1]<=w[i+1,j]+w[i,j+1]则决策单调
反正要对拍 先写个朴素打出决策表 若发现满足决策单调性则
使用一个栈来维护数据，占中的每一个元素保存一个决策的起始位置与终了位置，显然这些位置相互连接且依次递增。当插入一个新的决策时，从后到前扫描栈，对于每一个老决策来说，做这样两件事：
1、  如果在老决策的起点处还是新决策更好，则退栈，全额抛弃老决策，将其区间合并至新决策中，继续扫描下一个决策。
2、  如果在老决策的起点处是老决策好，则转折点必然在这个老决策的区间中；二分查找之，然后新决策进栈，结束。
由于一个决策出栈之后再也不会进入，所以均摊时间为O(1)，但是由于二分查找的存在，所以整个算法的时间复杂度为O(nlogn)。

上面的转移方程中数组w与当前决策i有关 若能分解成独立于i,k的两个值 就能转换成另一个模型f[i]=min(f[k])+a[i](k<i)
但这是无聊的 只需存一个当前最小值的变量即可 但把可选决策的范围限制就可以用线段树优化到nlogn  若可选决策区间单调就转换成了一个经典模型
f[i]=min(f[k])+a[i](b[i]<=k<c[i]) 对于任意i<j 满足b[i]<=b[j] c[i]<=c[j] 则是典型的单调队列；

f[i]=min(a[i]*x[k]+b[i]*y[k])(k<i)
此模型涵盖甚广
f=ax+by->y=-(a/b)x=f/b  即斜率固定 令纵截距取最值 显然最优决策点在凸包上
1.若斜率与加入的决策点横坐标同时满足单调 则是典型的斜率优化(graham-scan)
(事实上不一定是决策点横坐标单调 有可能是纵坐标/横坐标单调等等 只要满足新添加的点必然
```

```
添加在两端即可)
2.无任何限制 平衡二叉树维护凸包与查询最优决策(如何删除点?)(编程复杂度高。。)
(需要维护多个凸面的时候 式子稍微变形 就可以只写一个insert和ask 若是初始很多点 只需要删除操作 那么可以转换成逆序添加操作)
```

7 Others

7.1 高斯消元

```
高斯消元法就是平时用的消元法,用a[i,j]表示第i个方程的xj的系数。详见《高斯消元法.ppt》。
为了使误差减小,每次取xi系数绝对值最大的跟第i行换,再接着做高斯消元。

高斯约当消元法:每次取i行的a[i,j]最大的,记p[i]=j,全行同时除a[i,j],用i行消去其他行,使得a[k,j]=0,这样最后读的时候x[p[i]]=a[i,n+1]。代码短且误差会比高斯消元法小。


精确的高斯消元法:乘上一个数使得系数为整数,当然系数会逐渐变大,记得除以gcd。但仍然不能保证在int64以内,所以只有当题目的数字比较小且要求无误差的情况下用此方法。

解xor方程组:
由xor的独立性易证:
a1*x=b1; a2*x=b2; ==>(a1 xor a2)*x=b1 xor b2
于是选择一行i的第i个系数为1的(如果为0去找一行第i个系数为1的跟第i行换),于是若a[k,i]=1就去消第k行,即a[k,j]:=a[k,j] xor a[i,j] (i<j<=n)。这样可以保证a[k,i]=0。而a[k,i]已经等于0的就不去动他了。
之后总有一天发现全部为0了,若有s个自由元就有2^s组解,用dfs暴力出来就可以了。
这一部分还是看代码吧。
```

7.1.1 高斯消元

```cpp
//m行n+1列,m条方程n个未知数
void gauss(){
    rep(i,1,m){
        int p=1;
        rep(j,1,n) if (abs(a[i][p])<abs(a[i][j])) p=j;
        if (dcmp(a[i][p])==0){
            if (dcmp(a[i][n+1])==0) continue;
            else return 1;
        }
        xp[i]=p;
        double tmp=a[i][p];
        rep(j,1,n+1) a[i][j]/=tmp;
        rep(k,1,m) if (i!=k){
            tmp=a[k][p];
            rep(j,1,n+1) a[k][j]-=tmp*a[i][j];
        }
    }
    rep(i,1,m) x[xp[i]]=a[i][n+1];
    return 0;
}

int main(){
    cin>>n; m=n;
    rep(i,1,m) rep(j,1,n+1) cin>>a[i][j];
    if (gauss()) cout<<"No Solution!"<<endl;
    else rep(i,1,n) cout<<x[i]<<' ';cout<<endl;
}
```

7.1.2 xor 方程组

```pascal
//usaco 09 NOV lights
var
  ans,n,m,i,j,k,x,y:longint;
  flag:boolean;
  a:array[0..500,0..500]of longint;

procedure swap(var a,b:longint);
var c:longint; begin c:=a; a:=b; b:=c; end;

procedure dfs(k,t:longint);
var i:longint;
begin
  if t>=ans then exit;
  if k=0 then begin
        if t<ans then ans:=t;
        exit;
  end;
  if a[k,k]=0 then begin
      //for i:=1 to k-1 do if a[i,k]>0 then a[i,n+1]:=a[i,n+1]xor 0;
        dfs(k-1,t);
      //for i:=1 to k-1 do if a[i,k]>0 then a[i,n+1]:=a[i,n+1]xor 0;
        for i:=1 to k-1 do if a[i,k]>0 then a[i,n+1]:=a[i,n+1]xor 1;
        dfs(k-1,t+1);
        for i:=1 to k-1 do if a[i,k]>0 then a[i,n+1]:=a[i,n+1]xor 1;
  end
  else begin
        if a[k,n+1]=1 then inc(t);
        for i:=1 to k-1 do if a[i,k]>0 then a[i,n+1]:=a[i,n+1]xor a[k,n+1];
        dfs(k-1,t);
        if a[k,n+1]=1 then dec(t);
        for i:=1 to k-1 do if a[i,k]>0 then a[i,n+1]:=a[i,n+1]xor a[k,n+1];
  end;
end;
```

```

begin
assign(input,'xor.in'); reset(input);
assign(output,'xor.out'); rewrite(output);
  readln(n,m);
  for i:=1 to m do begin
        readln(x,y);
        a[x,y]:=1; a[y,x]:=1;
  end;
  for i:=1 to n do begin a[i,i]:=1; a[i,n+1]:=1; end;
  for i:=1 to n do begin
        flag:=false;
        for k:=i to n do if a[k,i]=1 then begin
                for j:=1 to n+1 do swap(a[i,j],a[k,j]);
                flag:=true; break;
        end;

        if not flag then continue;
        for k:=i+1 to n do if a[k,i]=1 then
                for j:=1 to n+1 do a[k,j]:=a[k,j] xor a[i,j];
  end;
  ans:=maxlongint;
  dfs(n,0);
  writeln(ans);
close(output);
end.
```

7.2 博弈论

```
两堆石子 一次可以取其中一堆 或者两堆同时取相同的数目
1 1 2
2 3 5
3 4 7
4 6 10... 首项A(N)=TRUNC(N(SQRT(5)+1/2))

nim
n阶nim
在线求nim方案 （nlog（xi））维护最高位在第K位的所有数字（双向链表）

一排石子堆 每次只能从一堆取任意个放到它右边的那堆里面 不能操作者输
解法 隔位异或 原理用NIM的原理套用即可。 很多题目都是源自这个模型。


许多博弈游戏本质是动态规划 考察单调性（可以通过研究问题本身性质获得，也可以打表找规律）
状态单调：最好能得到两维关系的状态 有一维是单调的（k倍动态减法问题） 或者只有一个特殊值的（许多堆石子，只能取最边上两堆，浙江省选二试）。
决策单调：（k倍动态减法问题）一旦这个点在决策中不起作用 以后的决策中这个也不会起作用 np
【记忆化搜索？】单调是博弈的难点也是非常巧妙的重点。


SG函数适用的游戏：无法做出决策者输
若游戏可以分成若干个小游戏 每次选一个游戏操作 那么这个大游戏的SG值为所有小游戏的SG异或和
SG=MEX(此游戏状态可以操作到的所有子状态)
子游戏有时候不一定很容易划分 有些看似不能分解成子游戏 事实上可以 例子：
浙江09省选一试 game 以及 翻硬币问题 （每次操作非常随意，但事实上均可以分解成每个硬币单独游戏的情况再求异或和）


SG较NP的优势是可以求游戏的和 不少题目还是无法用NP或SG去做 还是自己摸索必胜策略以及必败态(打出SG表找规律)


Anti-SG游戏：做最后一步决策者输
对于任意一个Anti-SG游戏，如果我们规定当局面中所有的单一游戏的SG值为0时，游戏结束，则先手必胜当且仅当：
（1）游戏的SG函数不为0且游戏中某个单一游戏的SG函数大于1；
（2）游戏的SG函数为0且游戏中没有单一游戏的SG函数大于1。

Every-SG 游戏规定，对于还没有结束的单一游戏，游戏者必须
对该游戏进行一步决策；
Every-SG 游戏的其他规则与普通SG 游戏相同

               0     v是终止状态
step(v)=    max(step(u))+1    sg(v)>0   v-->u  sg(u)=0
            min(step(u))+1    sg(v)=0   v-->u
[定理]
对于Every-SG 游戏先手必胜当且仅当单一游戏中最大的step的游戏为先手必胜
```

7.3 陈丹琦分治

```
Procedure Solve(l, r)
If l = r
  Then更新ans, 利用已经计算好的l的最优决策k, 计算f [l]值, Exit
Mid ← (l + r) / 2
Solve(l, mid -1)
  对[l, mid-1]这一段扫描一遍计算出决策的凸线, 由于[mid+1 .. r]这一段以
-a[i] / b[i]的排序在预处理已经完成, 因此只需要扫描一遍更新[mid + 1 .. r]
的最优决策.
Solve(mid+1, r)
  利用[l, mid-1]已排好序的f []值和[mid+1, r]已排好序的f []值归并排序将
[l, r]这一段按f[]值排序.
```

```
  End Procedure
```

7.4 矩阵乘法

```
矩阵加法：c[i,j]:=a[i,j]+b[i,j]

矩阵乘法：

1.行列数:
  设a:n*r; (n行r列) b:r*m 的矩阵 则 c=a*b=n*m;
  j：拼掉中间的r

2.c 的值:
  c[i,j]:=∑a[i,k]*b[k,j]    即  for k:=1 to r do inc(c[i,j],a[i,k]*b[k,j])
  j：c[i,j]就是 a的i行*b的j列

3.求c值的代码:
function mul(a,b:matrix;n,r,m:integer):matrix;
var c:matrix;
    i,j,k:longint;
begin
  fillchar(c,sizeof(c),0);
  for i:=1 to n do
    for j:=1 to m do
      for k:=1 to r do
        inc(c[i,j],a[i,k]*b[k,j])
  mul:=c;
end;

4.运算法则：
  矩阵乘法满足结合律，不满足交换律

5.二分快速幂
  *矩阵乘法不用二分快速幂就毫无意义,使用矩阵乘法的目的就是快速幂加速到log(n)
  *只有方阵才能快速幂，所以做题时要构造方阵
  代码模仿快速幂方法一:
  function qp(a:matrix;n,b:int64):matrix;
  var k:matrix;
  begin
    dec(b);
    k:=a; qp:=a;
    while b>0 do begin
      if b and 1=1 then qp:=mul(qp,k,n,n,n);
      k:=mul(k,k,n,n,n);
      b:=b>>1;
    end;
  end.

  模仿快速幂方法二:
  function mpower(a:matrix;n,r,m,x:longint):matrix;
  var c:matrix;
  begin
    if x=1 then exit(a);
    c:=mpower(a,n,r,m,x>>1);
    c:=mul(c,c,n,r,m);
    if x and 1=1; then mpower:=mul(c,a,n,r,m)
    else exit(c);
  end;
  *一的方法更好，不用多开矩阵，且 b:=b>>1，b可以用高精，二不能做很大的n次方。
```

7.5 求 AmoB+...nAmoB

```
//BZOJ2659 bjwc2012
var
      p,q,ans:int64;

function calc(n,a,b:int64):int64;
var
      r,k:int64;
begin
      if a=0 then exit(0);
      if a>=b then begin
            r:=a mod b;
            k:=a div b;
            exit(k*(n+1)*n div 2+calc(n,r,b));
      end
      else begin
            r:=n*a div b;
            exit(r*n-calc(r,b,a){+r div a});    //r div a是直线上的点，因此题p,q互质，无此点
      end;
end;

begin
assign(input,'2659.in'); reset(input);
assign(output,'2659.out'); rewrite(output);
      readln(p,q);
      inc(ans,calc(p div 2,q,p));
      inc(ans,calc(q div 2,p,q));
      writeln(ans);
close(output);
end.
```

7.6 高精度

```
int bit=10;
struct gao{
    int len,a[10000];
    void jinwei(){
        rep(i,1,len){
            a[i+1]+=a[i]/bit;
            a[i]%=bit;
        }
        while(a[len+1]){
            len++;
            a[len+1]=a[len]/bit;
            a[len]%=bit;
        }
    }
    gao operator *(const gao &u)const{
        gao c;
        c.len=len+u.len-1;
        rep(i,1,len) rep(j,1,u.len) c.a[i+j-1]+=a[i]*u.a[j];
        c.jinwei();
        return c;
    }
    gao operator +(const gao &u)const{
        gao c;
        c.len=max(len,u.len);
        rep(i,1,c.len) c.a[i]=a[i]+u.a[i];
        c.jinwei();
        return c;
    }
    gao operator *(const int &u)const{
        gao c;
        c.len=len;
        rep(i,1,len) c.a[i]=a[i]*u;
        c.jinwei();
        return c;
    }
    bool operator <(const gao &u)const{
        if (len<u.len) return 1;
        if (len==u.len) per(i,len,1){
            if (a[i]<u.a[i]) return 1;
            if (a[i]>u.a[i]) return 0;
        }
        return 0;
    }
    gao operator -(const gao &u)const{
        gao c;
        c.len=len;
        rep(i,1,len) c.a[i]=a[i]-u.a[i];
        rep(i,1,len) if (c.a[i]<0){
            c.a[i]+=bit;
            c.a[i+1]--;
        }
        while (c.len && !c.a[c.len]) c.len--;
        return c;
    }
}
```

7.7 头文件

```
#include <cstdio>
#include <cstring>
#include <queue>
#include <algorithm>
#include <iostream>
#include <sstream>
#include <vector>
#include <string>
#include <map>
#include <set>
#include <cmath>
#define rep(i,a,b) for(int i=a,_b=b;i<=_b;++i)
#define per(i,a,b) for(int i=a,_b=b;i>=_b;--i)
#define For(i,b) for(int i=0,_b=b;i<_b;++i)
#define upmax(a,b) if((a)<(b)) (a)=(b)
#define upmin(a,b) if((a)>(b)) (a)=(b)
#define lowbit(x) (x)&(-(x))
using namespace std;
typedef long long ll;
```

8 Tips

8.1 对拍

```
./make
./pro
./std
while diff pro.out std.out; do
echo "AC"
./make
./pro
./std
done
echo "WA"
```

8.2 class-map

```
/*
// lower_bound()    返回键值>=给定元素的第一个位置
// upper_bound()     返回键值>给定元素的第一个位置
// find(u) 函数返回一个iterator,他的自变量为u
//mp.erase() 括号里为自变量或iterator都可以。
//mp.clear();
C++ Maps是一种关联式容器, 包含"关键字/值"对

begin() 返回指向map头部的迭代器
clear() 删除所有元素
count() 返回指定元素出现的次数
empty() 如果map为空则返回true
end() 返回指向map末尾的迭代器
equal_range() 返回特殊条目的迭代器对
erase() 删除一个元素
find() 查找一个元素
get_allocator() 返回map的配置器
insert() 插入元素
key_comp() 返回比较元素key的函数
lower_bound() 返回键值>=给定元素的第一个位置
max_size() 返回可以容纳的最大元素个数
rbegin() 返回一个指向map尾部的逆向迭代器
rend() 返回一个指向map头部的逆向迭代器
size() 返回map中元素的个数
swap() 交换两个map
upper_bound() 返回键值>给定元素的第一个位置
value_comp() 返回比较元素value的函数
//遍历:
for (cp=mp.begin();cp!=mp.end();cp++)
//mp.size() map中元素个数
*/
```

8.3 FastIo

```
inline int getint()
{
    char c;
    int sig=1,tp=0;
    while (c!='-'&&!isdigit(c)) c=getchar();
    if (c=='-')
    {
        sig=-1; c=getchar();
    }
    while (isdigit(c))
    {
        tp=tp*10+c-'0';
        c=getchar();
    }
    return sig*tp;
}

inline void putint(int x)
{
    if (x<0)
    {
        x=-x; putchar('-');
    }
  if(x>9) putint(x/10);
  putchar(x%10+'0');
}
```

8.4 JavaFastIo

```
import java.util.*;
import java.math.*;
import java.io.*;

public class Main
{
  static public void main(String[] args)
      throws FileNotFoundException
  {
    InputReader in = new InputReader(System.in);
    PrintWriter out = new PrintWriter(System.out);
    while (in.hasNext())
    {
      int x=in.nextInt();
      out.println(x);
    }
    out.close();
  }
}

class InputReader
{
  BufferedReader reader;
  StringTokenizer tokenizer;
  public InputReader(InputStream stream)
  {
    reader = new BufferedReader(new InputStreamReader(stream));
    tokenizer = null;
  }
  public boolean hasNext()
```

```
  {
    while (tokenizer == null || !tokenizer.hasMoreTokens())
    {
      try
      {
        tokenizer = new StringTokenizer(reader.readLine());
      } catch (Exception e)
      {
        return false;
      }
    }
    return tokenizer.hasMoreTokens();
  }
  public String next()
  {
    while (tokenizer == null || !tokenizer.hasMoreTokens())
    {
      try
      {
        tokenizer = new StringTokenizer(reader.readLine());
      } catch (Exception e)
      {
        throw new RuntimeException(e);
      }
    }
    return tokenizer.nextToken();
  }
  public int nextInt()
  {
    return Integer.parseInt(next());
  }
  public Long nextLong()
  {
    return Long.parseLong(next());
  }
  public BigInteger nextBigInteger()
  {
    return new BigInteger(next());
  }
}
```

8.5 javaSample

```
import java.util.*;
import java.math.*;
import java.io.*;

public class Main
{
  static public void main(String[] args)
  {
    Scanner sc=new Scanner(System.in);
    //Scanner sc=new Scanner(new BufferedInputStream(System.in));
    BigInteger u,v;
    BigInteger w[]=new BigInteger[6];
    u=sc.nextBigInteger();
    v=sc.nextBigInteger();
    w[0]=u.add(v);
    w[1]=u.subtract(v);
    w[2]=u.multiply(v);
    w[3]=u.divide(v);
    w[4]=u.remainder(v);
    w[5]=u.gcd(v);
    for (int i=0;i<=5;i++)
      System.out.println(w[i]);
  }
}
```

8.6 tips

```
//扩栈
#pragma comment(linker,"/STACK:102400000,102400000")
//java编译
gedit main.java
javac Main.java
java Main
//开O2
#pragma GCC optimize("O2")
```