ACM 笔记

TJU ACM

tmeteorj

版本	时间	修订人
V1.0	**2012.12.04**	**tmeteorj**
V2.0	**2013.11.08**	**tmeteorj**

/**********/

图论

/**********/

/**************************************/

定理

/**************************************/

二部图中:

1、Konig 定理:最大匹配数等于最小覆盖数(顶点属于最小覆盖集,或与最小覆盖集中的某个顶点直接相连)

2、最大独立集等于最小边覆盖数(最小路径覆盖,注意路径是否可重,可重则还要求传递闭包)等于图的点数减去最大匹配数（点数相等时）

3、顶点的最大度数等于最小边染色数:
简证,二部图中没有奇环,对于欧环,可以用两种颜色不断重复来染色,所以只要保证最大度数顶点连接的边染不同色即可.

4.最大团等于补图的最大独立集。(建边时注意将集合分为二分图)

最大流最小割定理

1、网络流图中的最大流等于最小割.

平面图性质

1.欧拉公式）如果一个连通的平面图有 n 个点，m 条边和 f 个面，那么 f=m-n+2

2.每个平面图 G 都有一个与其对偶的平面图 G*
G*中的每个点对应 G 中的一个面
对于 G 中的每条边 e
e 属于两个面 f1、f2，加入边(f1*， f2*)
e 只属于一个面 f，加入回边(f*， f*)

a.G 的面数等于 G*的点数，G*的点数等于 G 的面数，G 与 G*边数相同

b.G*中的环对应 G 中的割一一对应

利用最短路求最小割

对于一个 s-t 平面图，我们对其进行如下改造：
1.连接 s 和 t，得到一个附加面
2.求该图的对偶图 G*，令附加面对应的点为 s*，无界面对应的点为 t*
3.删去 s*和 t*之间的边
一条从 s*到 t*的路径，就对应了一个 s-t 割！
更进一步，如果我们令每条边的长度等于它的容量，那么最小割的容量就等于最短路的长度！

最少添加边问题

1. 最少添加有向边使的原图为强连通图的数量等于原图缩点后出度为 0 的点数和入度为 0 的点数的个数的最大值
2.添加最少边使原图成为双连通图的数量等于原图缩点后度数为 1 的点加 1 除以 2.(无向图也要缩点)

/**************************************/

割点割边

/**************************************/

```
/**
init:
    memset(vis,false,sizeof(vis));
    memset(cut,false,sizeof(cut));
    memset(bridge,false,sizeof(bridge));
    dfs(root,root,0);
**/
bool  vis[N];
int  dep[N],low[N];
void  dfs(int  now,int  fa,int  deep){
      vis[now]=true;
      dep[now]=low[now]=deep;
      int  tot=0;
      for(int  i=head[now];i!=-1;i=edge[i].nxt){
            int  to=edge[i].to;
            if(to==fa)continue;

if(vis[to])low[now]=min(low[now],dep[to]);
            else{
                  dfs(to,now,deep+1);
                  tot++;
                  low[now]=min(low[now],low[to]);

if((now==root&&tot>1)||(now!=root&&low[to]>=de
p[now]))cut[now]=true;

if(low[to]>dep[now])bridge[i]=bridge[i^1]=true
;
            }
      }
}
```

/**************************************/

树的分治

/**************************************/

```
/**
问：求树中所有路径长度小于 len 的点对个数
init:
    mark <-- false
    dfs(root)
**/
bool  mark[N];
int  temp[N],cnt,size[N];
int  n,len;
void  getsize(int  x,int  fa)  {  //以 x 为根的子树的
结点数目
      size[x]=1;
      for(int  i=head[x];  i!=-1;  i=edge[i].next)
{
            int  t=edge[i].to;
            if(mark[t]||t==fa)
                  continue;
            getsize(t,x);
            size[x]+=size[t];
      }
}
int  bestroot(int  x)  {  //找树的重心
      getsize(x,x);
      int  half=size[x]>>1;
      while(1)  {
            int  id=x,mmax=0;
            for(int  i=head[x];  i!=-1;
i=edge[i].next)  {
                  int  t=edge[i].to;
                  if(!mark[t]&&mmax<size[t])
                        mmax=size[id=t];
            }
            if(mmax<=half)
                  break;
            size[x]-=size[id];
            size[id]+=size[x];
            x=id;
      }
      return  x;
}
void  getdist(int  now,int  fa,int  d)  {  //当前节
点到'根'的距离
      if(d>len)return;
      temp[cnt++]=d;
      for(int  t,i=head[now];  i!=-1;
i=edge[i].next)  {
            t=edge[i].to;
            if(!mark[t]&&t!=fa)
                  getdist(t,now,d+edge[i].cost);
      }
}
int  dfs(int  now)  {
      now=bestroot(now);
      mark[now]=true;
      int  ret=0;
      for(int  t,i=head[now];  i!=-1;
i=edge[i].next)  {
            t=edge[i].to;
            if(!mark[t])
                  ret+=dfs(t);
      }
      cnt=0;
      getdist(now,now,0);
      sort(temp,temp+cnt);
      for(int  ll=0,rr=cnt-1;  ll<rr;)  {
            if(temp[ll]+temp[rr]<=len)
                  ret+=rr-ll,ll++;
            else
                  rr--;
      }
      for(int  t,limit,i=head[now];  i!=-1;
i=edge[i].next)  {
            t=edge[i].to;
            if(mark[t])
                  continue;
            cnt=0;
            limit=len-2*edge[i].cost;
            getdist(t,t,0);
            sort(temp,temp+cnt);
            for(int  ll=0,rr=cnt-1;  ll<rr;)  {
```

```
                if(temp[ll]+temp[rr]<=limit)
                    ret-=rr-ll,ll++;
                else
                    rr--;
            }
        }
        mark[now]=false;
        return ret;
}
/**************************************/
```

无向图的最大匹配

```
/**************************************/
#include<cstdio>
#include<cstring>
#include<cmath>
#include<algorithm>
#include<queue>
using namespace std;
#define MAXE 250*250*2
#define MAXN 250
#define SET(a,b) memset(a,b,sizeof(a))
deque<int> Q;
//g[i][j]存放关系图：i,j是否有边,match[i]存放 i 所匹配的点
int g[MAXN][MAXN];
bool inque[MAXN],inblossom[MAXN];
int match[MAXN],pre[MAXN],base[MAXN];
//找公共祖先
int findancestor(int u,int v) {
    bool inpath[MAXN]= {false};
    while(1) {
        u=base[u];
        inpath[u]=true;
        if(match[u]==-1)break;
        u=pre[match[u]];
    }
    while(1) {
        v=base[v];
        if(inpath[v])return v;
        v=pre[match[v]];
    }
}
//压缩花
void reset(int u,int anc) {
    while(u!=anc) {
        int v=match[u];
        inblossom[base[u]]=1;
        inblossom[base[v]]=1;
        v=pre[v];
        if(base[v]!=anc)pre[v]=match[u];
        u=v;
    }
}
void contract(int u,int v,int n) {
    int anc=findancestor(u,v);
    //SET(inblossom,0);
    memset(inblossom,0,sizeof(inblossom));
    reset(u,anc);
    reset(v,anc);
    if(base[u]!=anc)pre[u]=v;
    if(base[v]!=anc)pre[v]=u;
    for(int i=1; i<=n; i++)
        if(inblossom[base[i]]) {
            base[i]=anc;
            if(!inque[i]) {
                Q.push_back(i);
                inque[i]=1;
            }
        }
}
bool dfs(int S,int n) {
    for(int i=0; i<=n;
i++)pre[i]=-1,inque[i]=0,base[i]=i;
    Q.clear();
    Q.push_back(S);
    inque[S]=1;
    while(!Q.empty()) {
        int u=Q.front();
        Q.pop_front();
        for(int v=1; v<=n; v++) {

if(g[u][v]&&base[v]!=base[u]&&match[u]!=v) {

if(v==S||(match[v]!=-1&&pre[match[v]]!=-1))con
tract(u,v,n);
            else if(pre[v]==-1) {
                pre[v]=u;

if(match[v]!=-1)Q.push_back(match[v]),inque[ma
tch[v]]=1;
                else {
                    u=v;
                    while(u!=-1) {
                        v=pre[u];
                        int
w=match[v];
                        match[u]=v;
                        match[v]=u;
                        u=w;
                    }
                    return true;
                }
            }
        }
    }
    return false;
}
int maxmatch(int n) {
    int res=0;
    memset(match,-1,sizeof(match));
    for(i=1; i<=n; i++) {
        if(match[i]==-1&&dfs(i,n)) {
            res++;
        }
    }
    return res;
}
/**************************************/
```

无向图的最大环

```
/**************************************/
//cost:直接路径长度；dist:间接路径长度
int mincircle(){
    int ans=inf;
    for(int k=1;k<=n;k++){
        for(int i=1;i<k;i++){
            for(int j=i+1;j<k;j++){

ans=min(ans,dist[i][j]+cost[i][k]+cost[k][j]);
            }
        }
        for(int i=1;i<=n;i++){
            for(int j=1;j<=n;j++){

dist[i][j]=min(dist[i][j],dist[i][k]+dist[k][j
]);
            }
        }
    }
    return ans;
}

/**************************************/
```

Dinic 最大流 O（V^2*E）

```
/**************************************/
int dep[N],stk[N],cur[N],pre[N];
ll Dinic(int n,int s,int t) {
    int i,j,k,top,rear,front;
    ll res,flow=0;
    while(1) {
        memset(dep,-1,sizeof(dep));
        dep[stk[front=0]=s]=0;
        rear=1;
        for(; front!=rear;) {
            for(i=stk[front++],j=head[i];
j!=-1; j=edge[j].next) {

if(edge[j].cap>0ll&&dep[k=edge[j].y]==-1) {
                    dep[k]=dep[i]+1;
                    stk[rear++]=k;
                    if(k==t) {
                        front=rear;
                        break;
                    }
                }
            }
        }
```

```
            if(dep[t]==-1)
                break;
            memcpy(cur,head,sizeof(head));
            for(top=0,i=s;;) {
                if(i==t) {
                    for(res=inff,k=0; k<top; k++)
{
                        if(res>edge[stk[k]].cap)
{

res=edge[stk[j=k]].cap;
                        }
                    }
                    flow+=res;
                    for(k=0; k<top; k++) {
                        edge[stk[k]].cap-=res;

edge[stk[k]^1].cap+=res;
                    }
                    i=edge[stk[top=j]].x;
                } else {
                    for(j=cur[i]; j!=-1;
j=cur[i]=edge[j].next)

if(edge[j].cap>0&&dep[edge[j].y]==dep[edge[j].
x]+1)
                            break;
                    if(j!=-1) {
                        stk[top++]=j;
                        i=edge[j].y;
                    } else {
                        if(top==0)
                            break;
                        dep[i]=-1;
                        i=edge[stk[--top]].x;
                    }
                }
            }
        }
        return flow;
}
```

/**************************************/

Sap 最大流 O（V*E*logF

/**************************************/

```
int src,sink,h[V],cur[V],num[V],n;
int findpath(int x,int flow) {
    if(x==sink)return flow;
    int f=flow;
    for(int i=head[x]; i!=-1; i=edge[i].nxt) {
        if(edge[i].cap&&h[edge[i].y]+1==h[x])
{
            int
d=findpath(edge[i].y,f<edge[i].cap?f:edge[i].c
ap);
            edge[i].cap-=d;
            edge[i^1].cap+=d;
            f-=d;
            if(h[src]==n||!f)return flow-f;
        }
    }
    int minh=n;
    for(int i=head[x]=cur[x]; i!=-1;
i=edge[i].nxt) {

if(edge[i].cap&&h[edge[i].y]+1<minh)minh=h[edg
e[i].y]+1;
    }
    if(num[h[x]]-1==0)h[src]=n;
    else h[x]=minh;
    return flow-f;
}
int Sap() {
    memcpy(cur,head,sizeof(head));
    memset(h,0,sizeof(h));
    memset(num,0,sizeof(num));
    num[0]=n;
    int ans=0;
    while(h[src]<n)ans+=findpath(src,inf);
    return ans;
}
```

/**************************************/

ZKW_Cost_Flow

/**************************************/

```
//不能有负权边，对于最终流量较大，而费用取值范围不大的
图，或者是增广路径比较短的图（如二分图），zkw 算法都会比
较快.
struct ZKW_flow {
    int st, ed, ecnt, n;
    int head[MAXN];
    int cap[MAXE], cost[MAXE], to[MAXE],
next[MAXE];
    void init() {
        memset(head, 0, sizeof(head));
        ecnt = 2;
    }
    void addEdge(int u, int v, int cc, int ww)
{
        cap[ecnt] = cc;
        cost[ecnt] = ww;
        to[ecnt] = v;
        next[ecnt] = head[u];
        head[u] = ecnt++;
        cap[ecnt] = 0;
        cost[ecnt] = -ww;
        to[ecnt] = u;
        next[ecnt] = head[v];
        head[v] = ecnt++;
    }
    int dis[MAXN];
    void SPFA() {
        for(int i = 1; i <= n; ++i) dis[i]
= INF;
        priority_queue<pair<int, int> > Q;
        dis[st] = 0;
        Q.push(make_pair(0, st));
        while(!Q.empty()) {
            int u = Q.top().second, d =
-Q.top().first;
            Q.pop();
            if(dis[u] != d) continue;
            for(int p = head[u]; p; p = next[p])
{
                int &v = to[p];
                if(cap[p] && dis[v] > d +
cost[p]) {
                    dis[v] = d + cost[p];

Q.push(make_pair(-dis[v], v));
                }
            }
        }
        for(int i = 1; i <= n; ++i) dis[i]
= dis[ed] - dis[i];
    }
    int minCost, maxFlow;
    bool use[MAXN];
    int add_flow(int u, int flow) {
        if(u == ed) {
            maxFlow += flow;
            minCost += dis[st] * flow;
            return flow;
        }
        use[u] = true;
        int now = flow;
        for(int p = head[u]; p; p = next[p])
{
            int &v = to[p];
            if(cap[p] && !use[v] && dis[u] ==
dis[v] + cost[p]) {
                int tmp = add_flow(v, min(now,
cap[p]));
                cap[p] -= tmp;
                cap[p^1] += tmp;
                now -= tmp;
                if(!now) break;
            }
        }
        return flow - now;
    }
    bool modify_label() {
        int d = INF;
        for(int u = 1; u <= n; ++u) if(use[u])
            for(int p = head[u]; p; p =
next[p]) {
                int &v = to[p];
                if(cap[p] && !use[v]) d
= min(d, dis[v] + cost[p] - dis[u]);
            }
        if(d == INF) return false;
```

```
            for(int i = 1; i <= n; ++i) if(use[i])
dis[i] += d;
            return true;
      }

      int min_cost_flow(int ss, int tt, int nn)
{
            st = ss, ed = tt, n = nn;
            minCost = maxFlow = 0;
            SPFA();
            while(true) {
                  while(true) {
                        for(int i = 1; i <= n; ++i)
use[i] = 0;
                        if(!add_flow(st, INF))
break;
                  }
                  if(!modify_label()) break;
            }
            return minCost;
      }
} G;
```

/**************************************/

平面图网络流

/**************************************/

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<vector>
#include<cmath>
#define MP make_pair
#define PB push_back
#define X first
#define Y second
using namespace std;
typedef long long ll;
const int MAX=300005,INF=0x3f3f3f3f;
const ll INFC=1LL<<60;
const double eps=1e-8;
int head_d[MAX],head_f[MAX],nc_d,nc_f;//最短路径&最大流
int S_f,T_f,S_e,T_e,n_f,m_f,T_d,S_d,n_d;//流的S,T以及与之关联的边和最短路的S,T
int pres[MAX];//每条边右方代表的最短路中的点
ll dist[MAX];//最短路径长度
bool in_q[MAX];
int LS[MAX],front,rear;
struct point{
      int x,y;
      void read(){
            scanf("%d%d",&x,&y);
      }
      bool operator<(const point &ne)const{
            return x<ne.x;
      }
      point(){}
      point(int _x,int _y){
            x=_x;
            y=_y;
      }
}po[MAX];
struct EdgeDist {
      int to,nxt;
      ll cost;
} edge_d[MAX*3];
struct EdgeFlow {
      int pa,pb,nxt;
      ll cost;
} edge_f[MAX*3];
struct PointAngle{
      double ang;
      int a,b,e;
      bool operator<(const PointAngle &ne)const{
            return ang<ne.ang;
      }
      PointAngle(){}
      PointAngle(int _a,int _b,int _e){
            a=_a;
            b=_b;
            e=_e;
ang=atan2((double)(po[a].y-po[b].y),(double)(po[a].x-po[b].x));
      }
};
vector<PointAngle> in_e[MAX];//入边编号
void add_d(int a,int b,ll c) {
      edge_d[nc_d].to=b;
      edge_d[nc_d].cost=c;
      edge_d[nc_d].nxt=head_d[a];
      head_d[a]=nc_d++;
}
void add_f(int a,int b,ll c){
      edge_f[nc_f].pa=a;
      edge_f[nc_f].pb=b;
      edge_f[nc_f].cost=c;
      edge_f[nc_f].nxt=-1;
      in_e[b].PB(PointAngle(a,b,nc_f));
      nc_f++;
}
void init(){
      memset(head_d,-1,sizeof(head_d));
      memset(head_f,-1,sizeof(head_f));
      for(int i=0;i<MAX;i++)in_e[i].clear();
      nc_d=nc_f=0;
}
void work(int id){
      int si=in_e[id].size();
      sort(in_e[id].begin(),in_e[id].end());
      for(int i=0;i<si-1;i++){

edge_f[in_e[id][i].e].nxt=in_e[id][i+1].e^1;
      }

edge_f[in_e[id][si-1].e].nxt=in_e[id][0].e^1;
}
void make_point(int e,int id){
      for(int
i=e;i!=-1&&pres[i]==-1;i=edge_f[i].nxt)
            pres[i]=id;
}
ll spfa(int src,int sink){
      for(int i=0;i<n_d;i++)dist[i]=INFC;
      memset(in_q,false,sizeof(in_q));
      dist[src]=0;
      front=0;
      rear=1;
      in_q[src]=true;
      int now,to;
      ll cost;
      while(front!=rear){
            in_q[now=LS[front++]]=false;
            if(front==MAX)front=0;
            for(int
i=head_d[now];i!=-1;i=edge_d[i].nxt){
                  to=edge_d[i].to;
                  cost=edge_d[i].cost+dist[now];
                  if(dist[to]>cost){
                        dist[to]=cost;
                        if(!in_q[to]){

in_q[LS[rear++]=to]=true;
                              if(rear==MAX)rear=0;
                        }
                  }
            }
      }
      return dist[sink];
}
int main() {
      int Test,a,b;
      ll c;
      for(scanf("%d",&Test); Test; Test--) {
            scanf("%d%d",&n_f,&m_f);
            S_f=T_f=1;
            for(int i=1; i<=n_f; i++) {
                  po[i].read();
                  if(po[i]<po[S_f])S_f=i;
                  if(po[T_f]<po[i])T_f=i;
            }
            init();
            for(int i=0; i<m_f; i++) {
                  scanf("%d%d%I64d",&a,&b,&c);
                  if(a==b)continue;
                  add_f(a,b,c);
                  add_f(b,a,c);
```

```
            }
            point
L(po[S_f].x-1,po[S_f].y),R(po[T_f].x+1,po[T_f]
.y);
            po[0]=L;
            po[n_f+1]=R;
            S_e=nc_f;
            add_f(0,S_f,INFC);
            add_f(S_f,0,INFC);
            T_e=nc_f;
            add_f(n_f+1,T_f,INFC);
            add_f(T_f,n_f+1,INFC);
            for(int i=1;i<=n_f;i++)work(i);
            memset(pres,-1,sizeof(pres));
            make_point(S_e,0);
            make_point(T_e,1);
            S_d=0;
            T_d=1;
            n_d=2;
            for(int i=0;i<nc_f;i++){
                 if(pres[i]==-1){
                      make_point(i,n_d);
                      n_d++;
                 }
            }
            for(int i=0;i<S_e;i+=2){

add_d(pres[i],pres[i^1],edge_f[i].cost);

add_d(pres[i^1],pres[i],edge_f[i].cost);
            }
            printf("%I64d\n",spfa(S_d,T_d));
       }
       return 0;
}
```

/**************************************/

混合图的欧拉回路

/**************************************/

```
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
int
d[300],ne,head[300],cur[300],ps[300],dep[300];
struct enode
{
    int x,y,nxt,c;
}bf[20000];
const int inf=1<<29;
void add(int x,int y,int c)
{
    bf[ne].x=x;bf[ne].y=y;bf[ne].c=c;
    bf[ne].nxt=head[x];head[x]=ne++;
    bf[ne].x=y;bf[ne].y=x;bf[ne].c=0;
    bf[ne].nxt=head[y];head[y]=ne++;
}
int flow(int n,int s,int t)
{
    int tr,res=0,i,j,k,f,r,top;
    while(1)
    {
        memset(dep,-1,sizeof(dep));
        for(f=dep[ps[0]=s]=0,r=1;f!=r;)
        {

for(i=ps[f++],j=head[i];j;j=bf[j].nxt)
            {
                if(bf[j].c&&-1==dep[k=bf[j].y])
                {
                    dep[k]=dep[i]+1;ps[r++]=k;
                    if(k==t)
                    {
                        f=r;break;
                    }
                }
            }
        }
        if(-1==dep[t])
            break;
        memcpy(cur,head,n*sizeof(int));
        for(i=s,top=0;;)
        {
            if(i==t)
            {
                for(k=0,tr=inf;k<top;++k)
                {
                    if(bf[ps[k]].c<tr)
                    {
                        tr=bf[ps[f=k]].c;
                    }
                }
                for(k=0;k<top;k++)

bf[ps[k]].c-=tr,bf[ps[k]^1].c+=tr;
                res+=tr;i=bf[ps[top=f]].x;
            }

for(j=cur[i];cur[i];j=cur[i]=bf[cur[i]].nxt)
            {

if(bf[j].c&&dep[i]+1==dep[bf[j].y])
                    break;
            }
            if(cur[i])
            {
                ps[top++]=cur[i];
                i=bf[cur[i]].y;
            }
            else
            {
                if(0==top)
                    break;
                dep[i]=-1;i=bf[ps[--top]].x;
            }
        }
    }
    return res;
}
int main()
{
    int num;
    scanf("%d",&num);
    while(num--)
    {
        int n,m,a,b,s,i;
        scanf("%d%d",&n,&m);
        memset(d,0,sizeof(d));
        ne=2;
        memset(head,0,sizeof(head));
        while(m--)
        {
            scanf("%d%d%d",&a,&b,&s);
            if(s==1)
            {
                d[a]--;
                d[b]++;
            }
            else
            {
                add(a,b,1);
                d[a]--;d[b]++;
            }
        }
        bool flag=true;
        int re=0;
        for(i=1;i<=n;i++)
            if(d[i]<0)
            {
                if((-d[i])%2!=0)
                {
                    flag=false;break;
                }
                else
                {
                    add(0,i,-d[i]/2);
                    re+=-d[i]/2;
                }
            }
            else
            {
                if(d[i]%2!=0)
                {
                    flag=false;break;
                }
                else
                {
                    add(i,n+1,d[i]/2);
                }
            }
        if(flag&&re==flow(n+2,0,n+1))
            printf("possible\n");
```

```
        else
            printf("impossible\n");
    }
    return 0;
}
```

/**************************************/

最大权闭合图

/**************************************/

```
//对于权值大于 0 的点 add（S，v，w），对于点权小于 0 的点
add（v,T,-w），对于原图中其它边
add(u,v,inf),add(v,u,inf)
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
typedef long long LL;
const int N=6000,M=60000;
const LL inf=111<<60;
int head[N],nc;
struct edge
{
    int x,y,next;
    LL cap;
} edge[M*3];
void add(int x,int y,LL cap)
{
    edge[nc].x=x;
    edge[nc].y=y;
    edge[nc].cap=cap;
    edge[nc].next=head[x];
    head[x]=nc++;
    edge[nc].x=y;
    edge[nc].y=x;
    edge[nc].cap=0;
    edge[nc].next=head[y];
    head[y]=nc++;
}
int num[N],h[N],S,T,n,m;
LL findpath(int x,LL flow)
{
    if(x==T)
        return flow;
    LL res=flow;
    int pos=n-1;
    for(int i=head[x]; i!=-1; i=edge[i].next)
    {
        int y=edge[i].y;
        if(h[x]==h[y]+1&&edge[i].cap>0)
        {
            LL
tp=findpath(y,min(edge[i].cap,res));
            res-=tp;
            edge[i].cap-=tp;
            edge[i^1].cap+=tp;
            if(!res||h[S]==n)
                return flow-res;
        }
        if(edge[i].cap>0&&h[y]<pos)
            pos=h[y];
    }
    if(res==flow)
    {
        num[h[x]]--;
        if(num[h[x]]==0)
        {
            h[S]=n;
            return flow-res;
        }
        h[x]=pos+1;
        num[h[x]]++;
    }
    return flow-res;
}
void Sap()
{
    memset(h,0,sizeof(h));
    memset(num,0,sizeof(num));
    LL ans=0;
    while(h[S]!=n)
        ans+=findpath(S,inf);
    return ;
}
bool vis[N];
LL aa[N];
int dfs(int now,LL &val)
{
    int cnt=1;
    vis[now]=true;
    val+=aa[now];
    for(int i=head[now]; i!=-1; i=edge[i].next)
    {
        if(!vis[edge[i].y]&&edge[i].cap>0)
            cnt+=dfs(edge[i].y,val);
    }
    return cnt;
}
int main()
{
    while(scanf("%d%d",&n,&m)!=EOF)
    {
        memset(head,-1,sizeof(head));
        nc=0;
        S=0;T=n+1;
        for(int i=1;i<=n;i++)
        {
            scanf("%lld",&aa[i]);
            if(aa[i]>0)
                add(S,i,aa[i]);
            else if(aa[i]<0)
                add(i,T,-aa[i]);
        }
        for(int i=0,a,b;i<m;i++)
        {
            scanf("%d%d",&a,&b);
            add(a,b,inf);
        }
        n=T+1;
        Sap();
        aa[S]=0;
        memset(vis,false,sizeof(vis));
        int nn;
        LL ans=0;
        nn=dfs(S,ans)-1;
        printf("%d %lld\n",nn,ans);
    }
    return 0;
}
```

/**************************************/

最大密度子图

/**************************************/

```
/*
对原图中每一个点 add(S,v,U),add(v,T,U+2g-dv),U 为边的总数，dv 为点 v 的度数，g 为 2 分的估计值。对于原图其它边，add(u,v,1),add(v,u,1)。
   如果是带边权（正数）的图，则将结点的度改为它所连的边的权值之和，最大容量上限 U 也变成了所有边权总和，原图中每条边的容量改为该边的权值。
   如果带点权（可正可负）且带边权（正数），最大密度为点权之和加上边权之和除以点的个数，点的度数定义仍按边权的定义做，但是每个通向汇点的边的容量改为 U+2*(g-pv)+dv，pv 为点权，U 为所有点权的绝对值之和加上所有边权之和
*/
#include<cstdio>
#include<cstring>
#include<cstdlib>
#include<cmath>
#include<algorithm>
using namespace std;
const int N=400,M=5000;
const double inf=1e10,eps=1e-6;
int head[N],nc,du[N];
struct data
{
    int x,y;
}po[M];
struct edge
{
    int x,y,next;
    double cap;
} edge[M*3];
void add(int x,int y,double cap)
{
    edge[nc].x=x;
    edge[nc].y=y;
    edge[nc].cap=cap;
    edge[nc].next=head[x];
    head[x]=nc++;
    edge[nc].x=y;
    edge[nc].y=x;
    edge[nc].cap=0;
```

```
    edge[nc].next=head[y];
    head[y]=nc++;
}
int num[N],h[N],S,T,n,m;
double findpath(int x,double flow)
{
    if(x==T)
        return flow;
    double res=flow;
    int pos=n-1;
    for(int i=head[x]; i!=-1; i=edge[i].next)
    {
        int y=edge[i].y;
        if(h[x]==h[y]+1&&edge[i].cap>eps)
        {
            double
tp=findpath(y,min(edge[i].cap,res));
            res-=tp;
            edge[i].cap-=tp;
            edge[i^1].cap+=tp;
            if(res<eps||h[S]==n)
                return flow-res;
        }
        if(edge[i].cap>eps&&h[y]<pos)
            pos=h[y];
    }
    if(abs(res-flow)<eps)
    {
        num[h[x]]--;
        if(num[h[x]]==0)
        {
            h[S]=n;
            return flow-res;
        }
        h[x]=pos+1;
        num[h[x]]++;
    }
    return flow-res;
}
double Sap(double x)
{
    memset(head,-1,sizeof(head));
    nc=0;
    for(int i=0;i<m;i++)
    {
        add(po[i].x,po[i].y,1.0);
        add(po[i].y,po[i].x,1.0);
    }
    for(int i=1;i<T;i++)
    {
        add(S,i,(double)m);
        add(i,T,(double)m+2*x-(double)du[i]);
    }
    double ans=0;
    memset(h,0,sizeof(h));
    memset(num,0,sizeof(num));
    while(h[S]!=n)
        ans+=findpath(S,(T-1)*m);
    return (T-1.0)*m-ans;
}
bool vis[N];
int dfs(int now)
{
    int cnt=1;
    vis[now]=true;
    for(int i=head[now];i!=-1;i=edge[i].next)
    {
        if(!vis[edge[i].y]&&edge[i].cap>eps)
        {
            cnt+=dfs(edge[i].y);
        }
    }
    return cnt;
}
int main()
{
    while(scanf("%d%d",&n,&m)!=EOF)
    {
        memset(du,0,sizeof(du));
        S=0;T=n+1;n=T+1;
        for(int i=0; i<m; i++)
        {
            scanf("%d%d",&po[i].x,&po[i].y);
            du[po[i].x]++;du[po[i].y]++;
        }
        if(m==0)
        {
            printf("1\n1\n");
            continue;
        }
        double ll=0,rr=(double)m,mid;
        while(rr-ll>1.0/(T-1.0)/(T-1.0))
        {
            mid=(ll+rr)/2.0;
            double tp=Sap(mid);
            if(abs(tp)<eps)
                rr=mid;
            else
                ll=mid;
        }
        memset(vis,false,sizeof(vis));
        int ans=dfs(S)-1;
        if(ans==0)
        {
            Sap(ll);
            memset(vis,false,sizeof(vis));
            ans=dfs(S)-1;
        }
        printf("%d\n",ans);
        for(int i=1;i<T;i++)
            if(vis[i])
                printf("%d\n",i);
    }
    return 0;
}
```

/**************************************/

最小边割集

/**************************************/

```
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
const int N=600,M=125000;
const int inf=1<<29;
int n,m;
bool vis[N],combine[N];
int dist[N],path[N][N];
int mincut()
{
    int i,j,k,t,la,tp,mm,id,ans;
    ans=inf;
    memset(combine,false,sizeof(combine));
    for(i=n;i>1;i--)
    {
        memset(dist,0,sizeof(dist));
        memset(vis,false,sizeof(vis));
        for(j=0;j<i;j++)
        {
            mm=id=-1;
            for(k=0;k<n;k++)
            {

if(!vis[k]&&!combine[k]&&dist[k]>mm)
                    mm=dist[id=k];
            }
            if(id==-1)
                return 0;
            if(j==i-2)
                la=id;
            vis[id]=true;
            for(t=0;t<n;t++)
            {
                if(!vis[t]&&!combine[t])
                {
                    dist[t]+=path[id][t];
                }
            }
        }
        ans=min(dist[id],ans);
        combine[id]=true;
        for(j=0;j<n;j++)
        {

path[j][la]=path[la][j]+=path[j][id];
        }
    }
    return ans;
}
int main()
{
    while(scanf("%d%d",&n,&m)!=EOF)
    {
        memset(path,0,sizeof(path));
        for(int i=0; i<m; i++)
```

```
        {
            int a,b,c;
            scanf("%d%d%d",&a,&b,&c);
            path[a][b]+=c;
            path[b][a]+=c;
        }
        printf("%d\n",mincut());
    }
    return 0;
}
```

/**************************************/

二分图最小权覆盖集

/**************************************/

```
//如果权值累加方式为乘积，可以将权值改为 log（w），最后在 exp（W）还原。最小点权覆盖集的补图为最大点权独立集
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
const int N=300,M=5000;
const int inf=1<<29;
int head[N],nc;
struct edge
{
    int x,y,next,cap;
} edge[M*4];
void add(int x,int y,int cap)
{
    edge[nc].x=x;
    edge[nc].y=y;
    edge[nc].cap=cap;
    edge[nc].next=head[x];
    head[x]=nc++;
    edge[nc].x=y;
    edge[nc].y=x;
    edge[nc].cap=0;
    edge[nc].next=head[y];
    head[y]=nc++;
}
int num[N],h[N],S,T,n,m;
int findpath(int x,int flow)
{
    if(x==T)
        return flow;
    int res=flow,pos=n-1;
    for(int i=head[x]; i!=-1; i=edge[i].next)
    {
        int y=edge[i].y;
        if(h[x]==h[y]+1&&edge[i].cap>0)
        {
            int
tp=findpath(y,min(edge[i].cap,res));
            res-=tp;
            edge[i].cap-=tp;
            edge[i^1].cap+=tp;
            if(!res||h[S]==n)
                return flow-res;
        }
        if(edge[i].cap>0&&h[y]<pos)
            pos=h[y];
    }
    if(res==flow)
    {
        num[h[x]]--;
        if(num[h[x]]==0)
        {
            h[S]=n;
            return flow-res;
        }
        h[x]=pos+1;
        num[h[x]]++;
    }
    return flow-res;
}
int Sap()
{
    memset(h,0,sizeof(h));
    memset(num,0,sizeof(num));
    int ans=0;
    num[0]=n;
    while(h[S]!=n)
        ans+=findpath(S,inf);
    return ans;
}
bool vis[N],mark[N];
void dfs(int now)
{
    vis[now]=true;
    for(int i=head[now]; i!=-1; i=edge[i].next)
    {
        if(!vis[edge[i].y]&&edge[i].cap>0)
            dfs(edge[i].y);
    }
}
int main()
{
    while(scanf("%d%d",&n,&m)!=EOF)
    {
        memset(head,-1,sizeof(head));
        nc=0;
        S=0,T=2*n+1;
        for(int i=1,w;i<=n;i++)
        {
            scanf("%d",&w);
            add(i+n,T,w);
        }
        for(int i=1,w;i<=n;i++)
        {
            scanf("%d",&w);
            add(S,i,w);
        }
        for(int i=0,a,b;i<m;i++)
        {
            scanf("%d%d",&a,&b);
            add(a,b+n,inf);
        }
        int nn=n;
        n=T+1;
        printf("%d\n",Sap());
        memset(vis,false,sizeof(vis));
        memset(mark,false,sizeof(mark));
        dfs(S);
        int top=0;
        for(int i=head[S];i!=-1;i=edge[i].next)
        {
            if(!vis[edge[i].y])
                top++,mark[edge[i].y]=true;
        }
        for(int i=head[T];i!=-1;i=edge[i].next)
        {
            if(vis[edge[i].y])
                top++,mark[edge[i].y]=true;
        }
        printf("%d\n",top);
        for(int i=1;i<=nn;i++)
        {
            if(mark[i])
                printf("%d -\n",i);
            if(mark[i+nn])
                printf("%d +\n",i);
        }
    }
    return 0;
}
```

/**************************************/

最优比例生成树

/**************************************/

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<cmath>
#include<cstdlib>
using namespace std;
const double esp = 0.00001;
const int MAXN =1010;
const double DINF = 1000000000.0;
struct Point
{
    int x,y,z;
} points[MAXN];
int N;
bool vi[MAXN];
double dist[MAXN];
int pre[MAXN];

double cal(int a,int b)
{
    return
sqrt((points[a].x-points[b].x)*(points[a].x-po
ints[b].x)+(points[a].y-points[b].y)*(points[a
].y-points[b].y)+0.0);
}
```

```
double prim(double x)
{
    memset(vi,false,sizeof(vi));
    for(int i=2; i<=N; i++)
    {

dist[i]=abs(points[1].z-points[i].z)-cal(1,i)*
x;
        pre[i]=1;
    }
    dist[1]=0;
    vi[1]=true;
    double cost=0,len=0;
    for(int i=1; i<N; i++)
    {
        double Min=DINF;
        int u;
        for(int j=2; j<=N; j++)
            if(!vi[j]&&Min>dist[j])
            {
                Min=dist[j];
                u=j;
            }
        vi[u]=1;
        cost+=abs(points[pre[u]].z-points[u].z);
        len+=cal(pre[u],u);
        for(int j=2; j<=N; j++)
        {
            double
val=abs(points[u].z-points[j].z)-cal(u,j)*x;
            if(!vi[j] && dist[j]>val)
            {
                dist[j]=val;
                pre[j]=u;
            }
        }
    }
    return cost/len;
}
int main()
{
    while(scanf("%d",&N),N)
    {
        for(int i=1; i<=N; i++)

scanf("%d%d%d",&points[i].x,&points[i].y,&poin
ts[i].z);
        double a=0,b;
        while(1)
        {
            b=prim(a);
            if(fabs(b-a)<esp)break;
            a=b;
        }
        printf("%.3f\n",b);
    }
    return 0;
}
/**************************************/
```

曼哈顿距离生成树

```
/**************************************/

/*
给定n个点，求曼哈顿最小生成树第K大边的长度
*/
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
const int
N=10005,MAX=5000,M=N*8,inf=0x3f3f3f3f;
inline int lowbit(int x){
    return x&(-x);
}
struct Edge{
    int x,y,cost;
    Edge(){}
    Edge(int _x,int _y,int _c){
        x=_x;
        y=_y;
        cost=_c;
    }
    bool operator<(const Edge &ne)const{
        return cost<ne.cost;
    }
}edge[M];
struct point{
    int x,y,id;
    bool operator<(const point &ne)const{
        return x!=ne.x?x<ne.x:y<ne.y;
    }
}po[N];
struct BITree{
    int a[MAX],b[MAX];
    void init(){
        memset(a,0x3f,sizeof(a));
        memset(b,-1,sizeof(b));
    }
    void updata(int pos,int va,int vb){
        while(pos>0){
            if(a[pos]>va){
                a[pos]=va;
                b[pos]=vb;
            }
            pos-=lowbit(pos);
        }
    }
    int getmin(int pos){
        int id=-1,va=inf;
        while(pos<MAX){
            if(va>a[pos]){
                va=a[pos];
                id=b[pos];
            }
            pos+=lowbit(pos);
        }
        return id;
    }
}BIT;
inline int dist(point a,point b){
    return abs(a.x-b.x)+abs(a.y-b.y);
}
int fa[N];
int Find(int x){
    if(x==fa[x])return x;
    return fa[x]=Find(fa[x]);
}
bool make_set(int x,int y){
    int fx=Find(x),fy=Find(y);
    if(fx==fy)return false;
    fa[fx]=fa[fy]=min(fx,fy);
    return true;
}
int Manhatton_MST(point po[],int n,int K){
    int a[N],b[N],m,tot=0;
    for(int dir=0;dir<4;dir++){
        if(dir==1||dir==3){
            for(int i=0;i<n;i++){
                swap(po[i].x,po[i].y);
            }
        }
        else if(dir==2){
            for(int i=0;i<n;i++){
                po[i].x=-po[i].x;
            }
        }
        sort(po,po+n);
        for(int i=0;i<n;i++){
            a[i]=b[i]=po[i].y-po[i].x;
        }
        sort(b,b+n);
        m=unique(b,b+n)-b;
        BIT.init();
        for(int i=n-1;i>=0;i--){
            int
pos=lower_bound(b,b+m,a[i])-b+1,to;
            to=BIT.getmin(pos);
            if(to!=-1)

edge[tot++]=Edge(po[i].id,po[to].id,dist(po[i]
,po[to]));

BIT.updata(pos,po[i].x+po[i].y,i);
        }
    }
    sort(edge,edge+tot);
    for(int i=0;i<n;i++){
        fa[i]=i;
    }
    for(int i=0,j=0;i<tot;i++){
        if(make_set(edge[i].x,edge[i].y)){
            j++;
            if(j==K)return edge[i].cost;
```

```
            }
        }
}
int main(){
    int n,K;
    while(scanf("%d%d",&n,&K)!=EOF){
        for(int i=0;i<n;i++){
            scanf("%d%d",&po[i].x,&po[i].y);
            po[i].id=i;
        }

printf("%d\n",Manhatton_MST(po,n,n-K));
    }
    return 0;
}
```

/**************************************/

最小限度生成树

/**************************************/

```
#include<cstdio>
#include<cstring>
#include<map>
#include<algorithm>
#include<string>
using namespace std;
map<string,int> M;
const int inf=1<<29;
const int N=100;
int n;
int dist[N],maxvalue[N],fa[N],g[N][N];
bool vis[N],tree[N][N];
int prime(int root)
{
    int i,j,k,a=0,mm;
    while(1)
    {
        k=-1;mm=inf;
        for(i=1;i<n;i++)
        {
            if(!vis[i]&&dist[i]<mm)
            {
                mm=dist[i];k=i;
            }
        }
        if(k==-1)
            break;
        a+=dist[k];
        vis[k]=true;
        if(k!=root)

maxvalue[k]=max(maxvalue[fa[k]],g[fa[k]][k]);
        tree[fa[k]][k]=tree[k][fa[k]]=true;
        for(i=1;i<n;i++)
        {

if(!vis[i]&&g[k][i]&&g[k][i]<dist[i])
            {
                dist[i]=g[k][i];fa[i]=k;
            }
        }
    }
    return a;
}
void dfs(int root)
{
    int i,now,top=1,stk[N*2],st=0;
    stk[0]=root;
    while(st!=top)
    {
        now=stk[st++];
        vis[now]=true;
        if(st==N*2)
            st=0;
        for(i=1;i<n;i++)
        {
            if(!vis[i]&&tree[now][i])
            {
                fa[i]=now;
                if(now==root)
                    maxvalue[i]=-inf;
                else

maxvalue[i]=max(maxvalue[now],g[now][i]);
                stk[top++]=i;
                if(top==2*N)
                    top=0;
            }
        }
    }
}
int solve(int src,int deg)//src 限度为 deg的最小生成树
{
    int i,j,k,ans=0,mm;
    for(i=1;i<n;i++)
    {
        dist[i]=inf;
        vis[i]=false;
    }
    memset(tree,false,sizeof(tree));
    vis[0]=true;dist[0]=0;
    while(deg)
    {
        k=-1;mm=inf;
        for(i=1;i<n;i++)
        {
            if(!vis[i]&&g[0][i]&&g[0][i]<mm)
            {
                mm=g[0][i];
                k=i;
            }
        }
        if(k==-1)
            break;
        fa[k]=0;
        maxvalue[k]=-inf;
        dist[k]=0;
        ans+=prime(k)+g[0][k];
        deg--;
    }
    while(deg)
    {
        k=-1;mm=inf;
        for(i=1;i<n;i++)
        {
            if(!tree[0][i]&&g[0][i])
            {
                if(mm>g[0][i]-maxvalue[i])
                {
                    mm=g[0][i]-maxvalue[i];
                    k=i;
                }
            }
        }
        if(k==-1||mm>=0)
            break;
        ans+=mm;
        tree[0][k]=tree[k][0]=true;
        int lon=maxvalue[k];
        while(g[fa[k]][k]!=lon)
        {
            k=fa[k];
        }
        tree[fa[k]][k]=tree[k][fa[k]]=false;
        memset(vis,false,sizeof(vis));
        dfs(0);
        deg--;
    }
    return ans;
}
int main()
{
    int m,deg;
    n=1;
    M.clear();
    M["Park"]=0;
    scanf("%d",&m);
    memset(g,0,sizeof(g));
    while(m--)
    {
        char s1[15],s2[15];
        int a,b,di;
        scanf(" %s %s %d",s1,s2,&di);
        if(M.find(s1)==M.end())
            M[s1]=n++;
        if(M.find(s2)==M.end())
            M[s2]=n++;
        a=M[s1];b=M[s2];
        g[a][b]=g[b][a]=di;
    }
    scanf("%d",&deg);
    printf("Total miles
driven: %d\n",solve(0,deg));
```

```
    return 0;
}
/****************************************/
```

次小生成树

```
/****************************************/
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<queue>
using namespace std;
const int N=1005,inf=0x3f3f3f3f;
int
head[N],nc,maxv[N][N],dist[N],pre[N],prd[N],n,
m;
struct Edge{
     int from,to,cost,nxt;
     bool in;
}edge[N*N];
void add(int a,int b,int c){
     edge[nc].from=a;
     edge[nc].to=b;
     edge[nc].nxt=head[a];
     edge[nc].cost=c;
     edge[nc].in=false;
     head[a]=nc++;
}
void init(){
     memset(prd,-1,sizeof(prd));
     memset(pre,-1,sizeof(pre));
     memset(head,-1,sizeof(head));
     memset(maxv,-1,sizeof(maxv));
     memset(dist,0x3f,sizeof(dist));
     nc=0;
}
int prime(){
     dist[1]=0;
     prd[1]=1;
     queue<int> S,T;
     for(int i=1;i<=n;i++)T.push(i);
     int ans=0;
     for(int _=0;_<n;_++){
          int id=-1,mxf=inf;
          for(int i=0,si=T.size();i<si;i++){
               int tmp=T.front();
               T.pop();
               if(dist[tmp]<mxf){
                    if(id!=-1)T.push(id);
                    mxf=dist[id=tmp];
               }
               else{
                    T.push(tmp);
               }
          }
          if(id==-1)break;
          ans+=dist[id];

if(id!=1)edge[pre[id]].in=edge[pre[id]^1].in=t
rue;
          for(int i=0,si=S.size();i<si;i++){
               int tmp=S.front();
               S.pop();

maxv[id][tmp]=maxv[tmp][id]=max(maxv[tmp][prd[
id]],mxf);
               S.push(tmp);
          }
          S.push(id);
          for(int
i=head[id];i!=-1;i=edge[i].nxt){
               int
to=edge[i].to,cost=edge[i].cost;
               if(dist[to]>cost){
                    dist[to]=cost;
                    pre[to]=i;
                    prd[to]=id;
               }
          }
     }
     return ans;
}
int main(){
     //freopen("data.in","r",stdin);
     int T;
     for(scanf("%d",&T);T;T--){
          init();
          scanf("%d%d",&n,&m);
          for(int i=0,a,b,c;i<m;i++){
               scanf("%d%d%d",&a,&b,&c);
               add(a,b,c);
               add(b,a,c);
          }
          int ans=prime();
          bool flag=false;
          for(int i=0;i<nc;i+=2){
               if(edge[i].in)continue;

if(maxv[edge[i].from][edge[i].to]==edge[i].cos
t){
                    flag=true;
                    break;
               }
          }
          if(flag)puts("Not Unique!");
          else printf("%d\n",ans);
     }
     return 0;
}
/****************************************/
```

树链剖分

```
/****************************************/
//0 e c 将 e 边权值改为 c
//1 a b 问 a 到 b 的长度
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
typedef long long ll;
const int N=50005;
struct Edge{
    int to,nxt;
    ll cost;
}edge[N*3];
int head[N],nc,n;
void AddEdge(int a,int b,ll c){

edge[nc].to=b;edge[nc].cost=c;edge[nc].nxt=
head[a];head[a]=nc++;
}
ll line[N*4];
void Insert(int now,int left,int right,int
pos,ll val){
    if(left==pos&&right==pos){
        line[now]=val;
        return ;
    }
    int
mid=(left+right)>>1,lc=now<<1,rc=(now<<1)|1
;
    if(pos<=mid)Insert(lc,left,mid,pos,val);
    else Insert(rc,mid+1,right,pos,val);
    line[now]=line[lc]+line[rc];
}
ll FindSum(int now,int left,int right,int
a,int b){
    if(a>b)return 0ll;
    if(left==a&&right==b){
        return line[now];
    }
    int
mid=(left+right)>>1,lc=now<<1,rc=(now<<1)|1
;
    if(b<=mid)return
FindSum(lc,left,mid,a,b);
    else if(a>mid)return
FindSum(rc,mid+1,right,a,b);
    else return
FindSum(lc,left,mid,a,mid)+FindSum(rc,mid+1
,right,mid+1,b);
}
struct STK_Heavy{//now,fa,e,dep,cost
    int now,fa,e,dep;
    ll cost;
    STK_Heavy(){}
    STK_Heavy(int _n,int _f,int _e,int _d,ll
_c){
        now=_n,fa=_f,e=_e,dep=_d,cost=_c;
```

```
    }
}Hstk[N*2];
struct STK_Create{//now,ac,e
    int now,ac,e;
    STK_Create(){}
    STK_Create(int _n,int _a,int _e){
        now=_n,ac=_a,e=_e;
    }
}Cstk[N*2];
ll cost[N];
int
dep[N],anc[N],pa[N],tid[N],heavy[N],size[N]
,ID;
bool mark[N];
void DFS_Heavy(int root){
    int top=0;
    memset(mark,false,sizeof(mark));

Hstk[top]=STK_Heavy(root,root,head[root],0,
0);
    while(top>=0){
        STK_Heavy elem=Hstk[top];
        if(!mark[elem.now]){
            mark[elem.now]=true;
            size[elem.now]=1;
            heavy[elem.now]=-1;
            pa[elem.now]=elem.fa;
            cost[elem.now]=elem.cost;
            dep[elem.now]=elem.dep;
        }
        if(elem.e==-1){
            if(top){
                size[elem.fa]+=size[elem.now];

if(heavy[elem.fa]==-1||size[heavy[elem.fa]]
<size[elem.now]){
                    heavy[elem.fa]=elem.now;
                }
            }
            top--;
            continue;
        }
        int to=edge[elem.e].to;
        ll cc=edge[elem.e].cost;
        Hstk[top].e=edge[elem.e].nxt;
        if(mark[to])continue;

Hstk[++top]=STK_Heavy(to,elem.now,head[to],
elem.dep+1,cc);
    }
}
void DFS_Create(int root){
    int top=0;

Cstk[0]=STK_Create(root,root,head[root]);
    memset(mark,false,sizeof(mark));
    while(top>=0){
        STK_Create elem=Cstk[top];
        if(!mark[elem.now]){
            mark[elem.now]=true;
            tid[elem.now]=++ID;
            anc[elem.now]=elem.ac;

Insert(1,0,n+1,ID,cost[elem.now]);
            if(heavy[elem.now]!=-1){

Cstk[++top]=STK_Create(heavy[elem.now],elem
.ac,head[heavy[elem.now]]);
                continue;
            }
        }
        if(elem.e==-1){
            top--;
            continue;
        }
        int to=edge[elem.e].to;
        ll cc=edge[elem.e].cost;
        Cstk[top].e=edge[elem.e].nxt;
        if(mark[to])continue;

Cstk[++top]=STK_Create(to,to,head[to]);
    }
}
void Init(){
    memset(head,-1,sizeof(head));
    memset(line,0,sizeof(line));
    nc=1,ID=0;
}
void Change(int e,ll c){
    int a=edge[e*2-1].to,b=edge[e*2].to;
    if(pa[b]==a)swap(a,b);
    Insert(1,0,n+1,tid[a],c);
}
ll Query(int a,int b){
    ll ans=0;
    int left,right;
    while(anc[a]!=anc[b]){
        if(dep[anc[a]]<dep[anc[b]])swap(a,b);
        left=tid[anc[a]],right=tid[a];
        if(left>right)swap(left,right);
        ans+=FindSum(1,0,n+1,left,right);
        a=pa[anc[a]];
    }
    left=tid[a],right=tid[b];
    if(left>right)swap(left,right);
    return ans+FindSum(1,0,n+1,left+1,right);
}
int main(){
    int q;
    while(scanf("%d%d",&n,&q)!=EOF){
        Init();
        int a,b,op;
        ll c;
        for(int i=1;i<n;i++){
            scanf("%d%d%I64d",&a,&b,&c);
            AddEdge(a,b,c);
            AddEdge(b,a,c);
        }
        DFS_Heavy(1);
        DFS_Create(1);
        for(int i=0;i<q;i++){
            scanf("%d",&op);
            if(op==0){
                scanf("%d%I64d",&a,&c);
                Change(a,c);
            }
            else{
                scanf("%d%d",&a,&b);
                printf("%I64d\n",Query(a,b));
            }
        }
    }
    return 0;
}
```

/**************************************/

生成树计数

/**************************************/

```
#include <stdio.h>
#include <vector>
#include <cstring>
#include <cmath>
using namespace std;
const int MOD = 10007;
int a[500][500],g[500][500],ni[MOD];
struct _st {
    int x,y;
}loc[500];
int sqr(int x) {
    return x * x;
}
int dist(int i,int j) {
    return (sqr(loc[i].x - loc[j].x) +
sqr(loc[i].y - loc[j].y));
}
pair<int,int> vec(int x,int y) {
    pair<int,int> tmp;
    tmp.first = loc[y].x - loc[x].x;
    tmp.second = loc[y].y - loc[x].y;
    return tmp;
}
int mat[500][500];
int gcd(int x,int y) {
    if (y == 0) return x;
    return gcd(y,x% y);
```

```
}
int Gauss(int n){
    for (int i = 0;i < n;i++) {
        for (int j = 0;j < n;j++)
            mat[i][j] = (mat[i][j] + MOD) % MOD;
    }
    int col = 0, k;
    int ans = 1;
    //for(int i= 0;i < n; ++i) b[i] = 1ll;
    for(k = 0;k < n && col < n; ++k, ++col){
        if(mat[k][col] == 0){
            for(int i = k + 1;i < n; ++i){
                if(!(mat[i][col] == 0)){
                    for(int j = col;j < n;
++j)swap(mat[k][j], mat[i][j]);
                    ans *= -1;
                    break;
                }
            }
        }
        int x = mat[k][col];
        ans *= x;
        ans %= MOD;
        ans += MOD;
        ans %= MOD;
        for(int i = k + 1;i < n; ++i){
            int y = mat[i][col];
            if(x == 0 || y == 0) continue;
            int d = gcd(abs(x),abs(y)),lcm = abs(x
* y / d);
            int tx = lcm / x,ty = lcm / y;
            for(int j = col;j < n; ++j){
                mat[i][j] = -tx * mat[k][j] + ty *
mat[i][j];
                //printf("!!%dn",mat[i][j]);
                mat[i][j] %= MOD;
                mat[i][j] = (mat[i][j] + MOD) % MOD;
            }
            ans = (ans * ni[ty]) % MOD;
            ans = (ans + MOD) % MOD;
        }
    }
    ans %= MOD;
    ans += MOD;
    ans %= MOD;
    return (int) ans;
}
int x,y;
int egcd(int a,int b) {
    int temp,tempx;
    if (b == 0) {
        x = 1;y = 0;
        return a;
    }
    temp = egcd(b,a % b);
    tempx = x;
    x = y;
    y = tempx - a / b * y;
    return temp;
}
bool cmp(pair<int,int> x,pair<int,int> y) {
    if (x.first * y.second - x.second*y.first ==
0) {
        if ((x.first*y.first > 0)) return true;
        if (x.second *y.second > 0) return true;
        return false;
    }
    return false;
}
int main() {
    for (int i = 1;i < MOD;i++) {
        egcd(i,MOD);
        ni[i] = (x+MOD) % MOD;
    }
    int T;
    for (scanf("%d",&T);T;T--) {
        int n,R;
        scanf("%d%d",&n,&R);
        for (int i = 0;i < n;i++) {
            scanf("%d%d",&loc[i].x,&loc[i].y);
        }
        memset(a,0,sizeof(a));
        memset(g,0,sizeof(g));
        for (int i = 0;i < n;i++) {
            for (int j = 0;j < n;j++) {
                if (dist(i,j) > R * R || i == j)
continue;
                bool flag = true;
                for (int k = 0;k < n;k++) {
                    if (i != k && j != k &&
cmp(vec(i,k), vec(k,j))) {
                        flag = false;
                        break;
                    }
                }
                if (flag) {
                    a[i][j] = 1;
                    g[i][i]++;
                }
            }
        }
        bool hasAns = true;
        for (int i = 0;i < n;i++) {
            if (g[i][i] == 0) hasAns = false;
        }
        if (!hasAns && n != 1) {
            printf("-1n");
            continue;
        }

        for (int i = 0;i < n - 1;i++) {
            for (int j = 0;j < n - 1;j++) {
                mat[i][j] = g[i][j] - a[i][j];
            }
        }

        int ans = Gauss(n-1);
        printf("%dn",ans);

    }
    return 0;
}
```

/**************************************/

树中删除最少边形成 n 连通集

/**************************************/

```
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
int n,m;
const int inf=9999999;
int dp[155][155],tot[155],map[155][155],root;
void dfs(int now)
{
    int to,i,j,k,len=tot[now];
    if(len==0)
    {
        if(now==root)
            dp[now][1]=0;
        else
            dp[now][1]=1;
        return;
    }
    if(now==root)
        dp[now][1]=len;
    else
        dp[now][1]=len+1;
    for(i=0;i<len;i++)
    {
        to=map[now][i];
        dfs(to);
        for(j=m-1;j>=1;j--)
        {
            if(dp[now][j]<inf)
            {
                for(k=m-j;k>=1;k--)
                {
                    if(dp[to][k]<inf)

dp[now][j+k]=min(dp[now][j+k],dp[now][j]+dp[to
][k]-2);
                }
            }
        }
    }
}
int main()
{
    while(scanf("%d%d",&n,&m)!=EOF)
    {
        int i,j,a,b;
        memset(tot,0,sizeof(tot));
        for(i=1;i<=n;i++)
```

```
            for(j=1;j<=m;j++)
                dp[i][j]=inf;
        bool head[155];
        memset(head,true,sizeof(head));
        for(i=0;i<n-1;i++)
        {
            scanf("%d%d",&a,&b);
            head[b]=false;
            map[a][tot[a]++]=b;
        }
        for(i=1;i<=n;i++)
        {
            if(head[i])
            {
                root=i;
                dfs(i);
            }
        }
        int ans=inf;
        for(i=1;i<=n;i++)
        {
            ans=min(ans,dp[i][m]);
        }
        printf("%d\n",ans);
    }
    return 0;
}
```

/**************************************/

N 阶完全图生成子图数量

/**************************************/

```
import java.util.*;
import java.math.*;
class Main
{
    static BigInteger cal(int n,int k)
    {
        if(n==0&&k!=0)
            return BigInteger.ZERO;
        else if(k==0||n==k)
            return BigInteger.ONE;
        BigInteger ret=new BigInteger("1");
        for(int i=1;i<=k;i++)
        {

ret=ret.multiply(BigInteger.valueOf(n-i+1)).di
vide(BigInteger.valueOf(i));
        }
        return ret;
    }
    public static void main(String arg[])
    {
        Scanner cin=new Scanner(System.in);
        BigInteger dp[]=new BigInteger[52];
        dp[1]=dp[2]=BigInteger.ONE;
        for(int i=3;i<=50;i++)
        {
            dp[i]=BigInteger.ZERO;
            for(int j=1;j<i;j++)
            {

dp[i]=dp[i].add(dp[j].multiply(cal(i-1,j-1).mu
ltiply(BigInteger.valueOf(2).pow((cal(i-j,2).i
ntValue())))));
            }

dp[i]=BigInteger.valueOf(2).pow(cal(i,2).intVa
lue()).subtract(dp[i]);
        }
        int n;
        while(true)
        {
            n=cin.nextInt();
            if(n==0)
                break;
            else
                System.out.println(dp[n]);
        }
    }
}
```

/**********/

计算几何

/**********/

/**************************************/

公式

/**************************************/

【三角形】：

1. 半周长 P=(a+b+c)/2
2. 面积
S=aHa/2=absin(C)/2=sqrt(P(P-a)(P-b)(P-c))
3. 中线
Ma=sqrt(2(b^2+c^2)-a^2)/2=sqrt(b^2+c^2+2bccos(A))/2
4. 角平分线
Ta=sqrt(bc((b+c)^2-a^2))/(b+c)=2bccos(A/2)/(b+c)
5. 高线
Ha=bsin(C)=csin(B)=sqrt(b^2-((a^2+b^2-c^2)/(2a))^2)
6. 内切圆半径
r=S/P=asin(B/2)sin(C/2)/sin((B+C)/2)

=4Rsin(A/2)sin(B/2)sin(C/2)=sqrt((P-a)(P-b)(P-c)/P)

=Ptan(A/2)tan(B/2)tan(C/2)

7. 外接圆半径
R=abc/(4S)=a/(2sin(A))=b/(2sin(B))=c/(2sin(C))

【四边形】：

D1,D2 为对角线,M 对角线中点连线,A 为对角线夹角

1. a^2+b^2+c^2+d^2=D1^2+D2^2+4M^2
2. S=D1D2sin(A)/2

(以下对圆的内接四边形)

3. ac+bd=D1D2
4. S=sqrt((P-a)(P-b)(P-c)(P-d)),P 为半周长

【正 n 边形】：

R 为外接圆半径,r 为内切圆半径

1. 中心角 A=2PI/n
2. 内角 C=(n-2)PI/n
3. 边长 a=2sqrt(R^2-r^2)=2Rsin(A/2)=2rtan(A/2)
4. 面积
S=nar/2=nr^2tan(A/2)=nR^2sin(A)/2=na^2/(4tan(A/2))

【圆】：

1. 弧长 l=rA
2. 弦长 a=2sqrt(2hr-h^2)=2rsin(A/2)
3. 弓形高
h=r-sqrt(r^2-a^2/4)=r(1-cos(A/2))=atan(A/4)/2
4. 扇形面积 S1=rl/2=r^2A/2
5. 弓形面积 S2=(rl-a(r-h))/2=r^2(A-sin(A))/2

【棱柱】：

1. 体积 V=Ah,A 为底面积,h 为高
2. 侧面积 S=lp,l 为棱长,p 为直截面周长
3. 全面积 T=S+2A

【棱锥】：

1. 体积 V=Ah/3,A 为底面积,h 为高

(以下对正棱锥)

2. 侧面积 S=lp/2,l 为斜高,p 为底面周长
3. 全面积 T=S+A

【棱台】：

1. 体积 V=(A1+A2+sqrt(A1A2))h/3,A1.A2 为上下底面积,h 为高

(以下为正棱台)

2. 侧面积 S=(p1+p2)l/2,p1.p2 为上下底面周长,l 为斜高
3. 全面积 T=S+A1+A2

【圆柱】：

1. 侧面积 S=2PIrh
2. 全面积 T=2PIr(h+r)
3. 体积 V=PIr^2h

【圆锥】：

1. 母线 l=sqrt(h^2+r^2)
2. 侧面积 S=PIrl
3. 全面积 T=PIr(l+r)
4. 体积 V=PIr^2h/3

【圆台】：

1. 母线 l=sqrt(h^2+(r1-r2)^2)
2. 侧面积 S=PI(r1+r2)l
3. 全面积 T=PIr1(l+r1)+PIr2(l+r2)
4. 体积 V=PI(r1^2+r2^2+r1r2)h/3

【球】：
1. 全面积 T=4PIr^2
2. 体积 V=4PIr^3/3
【球台】：
1. 侧面积 S=2PIrh
2. 全面积 T=PI(2rh+r1^2+r2^2)
3. 体积 V=PIh(3(r1^2+r2^2)+h^2)/6
【球扇形】：
1. 全面积 T=PIr(2h+r0),h 为球冠高,r0 为球冠底面半径
2. 体积 V=2PIr^2h/3

/**************************************/

二维计算几何库

/**************************************/

```
#include<cstdlib>
#include<cmath>
#include<cstdio>
#include<algorithm>
#define max(a,b) (((a)>(b))?(a):(b))
#define min(a,b) (((a)>(b))?(b):(a))
#define sign(x) ((x)>eps?1:((x)<-eps?(-1):(0)))
using namespace std;
const int MAXN=1000;
const double eps=1e-8,inf=1e50 ,Pi=acos(-1.0);
struct point {
     double x,y;
     point() {}
     point(double _x,double _y) {
          x=_x;
          y=_y;
     }
     point operator-(const point &ne)const {
          return point(x-ne.x,y-ne.y);
     }
     point operator+(const point ne)const {
          return point(x+ne.x,y+ne.y);
     }
     point operator*(const double t)const{
          return point(x*t,y*t);
     }
     point operator/(const double t)const{
          if(sign(t)==0)exit(1);
          return point(x/t,y/t);
     }
};
struct line{
     point a,b;
     line(){}
     line(point _a,point _b){a=_a;b=_b;}
};
struct line2 {
     double a,b,c;
     line2() {}
     line2(double _a,double _b,double _c) {
          a=_a;
          b=_b;
          c=_c;
     }
};
struct circle {
     point o;
     double r;
     circle(){}
     circle(point _o,double _r){
          o=_o;
          r=_r;
     }
};
struct rectangle{
     point a,b,c,d;
     rectangle(){}
     rectangle(point _a,point _b,point _c,point
_d){
          a=_a;
          b=_b;
          c=_c;
          d=_d;
     }
};
struct polygon{
     point p[MAXN];
     int n;
};
inline double xmult(point a,point b){
     return a.x*b.y-a.y*b.x;
}
inline double xmult(point o,point a,point b) {
     return
(a.x-o.x)*(b.y-o.y)-(b.x-o.x)*(a.y-o.y);
}
inline double xmult(double x1,double y1,double
x2,double y2) {
     return x1*y2-x2*y1;
}
inline double dmult(point o,point a,point b) {
     return
(a.x-o.x)*(b.x-o.x)+(a.y-o.y)*(b.y-o.y);
}
inline double dmult(point a,point b) {
     return a.x*b.x+a.y*b.y;
}
inline double lenth(point a){
     return sqrt(dmult(a,a));
}
inline double dist(point a,point b){
     return lenth(b-a);
}
inline double dist2(point a,point b){
     return dmult(b-a,b-a);
}
//直线一般式转两点式
line toline(double a,double b,double c) {
     if(sign(b)==0)exit(1);
     point A(0,-c/b),B(-c/a,0);
     return line(A,B);
}
//直线两点式转一般式
line2 toline2(point a,point b) {
     double A=b.y-a.y,B=a.x-b.x,C=-B*a.y-A*a.x;
     return line2(A,B,C);
}
//点 p 绕 o 逆时针旋转 alpha
point rotate(point o,point p,double alpha) {
     point tp;
     p.x-=o.x;
     p.y-=o.y;
     tp.x=p.x*cos(alpha)-p.y*sin(alpha)+o.x;
     tp.y=p.y*cos(alpha)+p.x*sin(alpha)+o.y;
     return tp;
}
//向量 u 的倾斜角
double angle(point u) {
     return atan2(u.y,u.x);
}
//oe 与 os 的夹角，夹角正负满足叉积
double angle(point o,point s,point e) {
     point os=s-o,oe=e-o;
     double
bot=sqrt(dmult(os,os)*dmult(oe,oe));
     double top=dmult(os,oe);
     double cosfi=top/bot;
     if (cosfi >= 1.0 ) return 0;
     if (cosfi <= -1.0 ) return -Pi;
     double fi=acos(cosfi);
     if(xmult(o,s,e)>0)return fi;
     else return -fi;
}
//p 在 l 上的投影与 l 关系
double relation(point p,line l) {
     line tl(l.a,p);
     return
dmult(tl.b-l.a,l.b-l.a)/dist2(l.a,l.b);
}
//p 在 l 上的垂足
point perpendicular(point p,line l) {
     double r=relation(p,l);
     return l.a+((l.b-l.a)*r);
}
//求点 p 到线段 l 的最短距离,并返回线段上距该点最近的点 np
double dist_p_to_seg(point p,line l,point &np)
{
     double r=relation(p,l);
     if(r<0) {
     np=l.a;
          return dist(p,l.a);
     }
     if(r>1) {
          np=l.b;
          return dist(p,l.b);
     }
     np=perpendicular(p,l);
```

```
     return dist(p,np);
}
//求点 p 到直线 l 的最短距离
double dist_p_to_line(point p,line l) {
     return
abs(xmult(p-l.a,l.b-l.a))/dist(l.a,l.b);
}
//线段之间最短距离
inline double dist_seg_to_seg(line p,line q) {
     return
min(min(dist_p_to_seg(p.a,q),dist_p_to_seg(p.b
,q)),min(dist_p_to_seg(q.a,p),dist_p_to_seg(q.
b,p)));
}
//求矢量线段夹角的余弦
double cosine(line u,line v) {
     point pu=u.b-u.a,pv=v.b-v.a;
     return
dmult(pu,pv)/sqrt(dmult(pu,pu)*dmult(pv,pv));
}
//求矢量的夹角的余弦
double cosine(point a,point b){
     return
dmult(a,b)/sqrt(dmult(a,a)*dmult(b,b));
}
//求线段夹角
double lsangle(line u,line v) {
     point o(0,0),a=u.b-u.a,b=v.b-v.a;
     return angle(o,a,b);
}
//求直线斜率
double slope(line2 l) {
     if(abs(l.a) < 1e-20)
          return 0;
     if(abs(l.b) < 1e-20)
          return inf;
     return -(l.a/l.b);
}
//直线倾斜角[0,Pi]
double alpha(line2 l) {
     if(abs(l.a)< eps)
          return 0;
     if(abs(l.b)< eps)
          return Pi/2;
     double k=slope(l);
     if(k>0)
          return atan(k);
     else
          return Pi+atan(k);
}
//点关于直线的对称点
point symmetry(line2 l,point p){
    point tp;

tp.x=((l.b*l.b-l.a*l.a)*p.x-2*l.a*l.b*p.y-2*l.
a*l.c)/(l.a*l.a+l.b*l.b);

tp.y=((l.a*l.a-l.b*l.b)*p.y-2*l.a*l.b*p.x-2*l.
b*l.c)/(l.a*l.a+l.b*l.b);
    return tp;
}
//判多边形是否逆时针
bool is_unclock(polygon pg) {
     int n=pg.n;
     pg.p[n]=pg.p[0];
     double area=0;
     for(int i=0; i<n; i++)

area+=xmult(pg.p[i].x,pg.p[i].y,pg.p[i+1].x,pg
.p[i+1].y);
     return area>-eps;
}
//改变多边形时针顺序
void to_unclock(polygon &pg) {
     for(int i=0,j=pg.n-1; i<j; i++,j--)
          swap(pg.p[i],pg.p[j]);
}

//判定凸多边形,顶点按顺时针或逆时针给出,允许相邻边共线
int is_convex(point p[],int n) {
     int i,s[3]= {1,1,1};
     for (i=0; i<n&&s[1]|s[2]; i++)

s[(sign(xmult(p[(i+1)%n],p[(i+2)%n],p[i]))+3)%
3]=0;
     return s[1]|s[2];
}
//判定凸多边形,顶点按顺时针或逆时针给出,不允许相邻边共线
int is_convex_v2(point p[],int n) {
     int i,s[3]= {1,1,1};
     for (i=0; i<n&&s[0]&&s[1]|s[2]; i++)

s[(sign(xmult(p[(i+1)%n],p[(i+2)%n],p[i]))+3)%
3]=0;
     return s[0]&&s[1]|s[2];
}
//判点在凸多边形内或多边形边上,顶点按顺时针或逆时针给出
int inside_convex(point q,point p[],int n) {
     int i,s[3]= {1,1,1};
     for (i=0; i<n&&s[1]|s[2]; i++)

s[(sign(xmult(p[(i+1)%n],q,p[i]))+3)%3]=0;
     return s[1]|s[2];
}
//判点在凸多边形内,顶点按顺时针或逆时针给出,在多边形边上返回 0
int inside_convex2(point q,point p[],int n) {
     int i,s[3]= {1,1,1};
     for (i=0; i<n&&s[0]&&s[1]|s[2]; i++)

s[(sign(xmult(p[(i+1)%n],q,p[i]))+3)%3]=0;
     return s[0]&&s[1]|s[2];
}
//判点在线段上
inline int p_on_seg(point a,point p1,point p2)
{

if(fabs(xmult(a,p1,p2))<=eps&&(a.x-p1.x)*(a.x-
p2.x)<eps&&(a.y-p1.y)*(a.y-p2.y)<eps)
          return 1;
     return 0;
}
//判点在线段端点左方
inline int p_on_segvex(point s,point p) {
     return fabs(p.y-s.y)<eps&&(p.x<=s.x+eps);
}
//判线段相交 <=:不规范相交
inline int seg_inter(line s,line p) {
     double
minx1=min(s.a.x,s.b.x),maxx1=max(s.a.x,s.b.x);
     double
minx2=min(p.a.x,p.b.x),maxx2=max(p.a.x,p.b.x);
     double
miny1=min(s.a.y,s.b.y),maxy1=max(s.a.y,s.b.y);
     double
miny2=min(p.a.y,p.b.y),maxy2=max(p.a.y,p.b.y);
     if((minx1>maxx2+eps)||(minx2>maxx1+eps)||

(miny1>maxy2+eps)||(miny2>maxy1+eps))
          return 0;
     else
          return
sign(xmult(s.a,s.b,p.a)*xmult(s.a,s.b,p.b))<=0
&&

sign(xmult(p.a,p.b,s.a)*xmult(p.a,p.b,s.b))<=0
;
}
//判点在多边形内部
inline int p_in_polygon(point a,point p[],int n)
{
     int count = 0;
     line s,ps;
     ps.a = a,ps.b = a;
     ps.b.x = inf;
     for(int i = 0; i < n; i++) {
          s.a = p[i];
          if(i + 1 < n)s.b = p[i+1];
          else s.b = p[0];
          if (s.a.y > s.b.y)swap(s.a,s.b);
          if (p_on_seg(a,s.a,s.b))return 2;
          if ((fabs(s.a.y-s.b.y)>eps)) {
               if (p_on_segvex(s.b,a)) count++;
               else if (seg_inter(ps,s))count++;
```

```
            }
        }
        if (count%2)return 1;
        return 0;
}
//多边形内部最长线段
point stk[MAXN];
double seg_max_len(line u,polygon &pg){
        double ans=0.0,tmp=inf;
        pg.p[pg.n]=pg.p[0];
        int n=pg.n,top=0;
        for(int i=0;i<n;i++){
            line v(pg.p[i],pg.p[i+1]);
            int
s1=sign(xmult(u.a,u.b,v.a)),s2=sign(xmult(u.a,
u.b,v.b));
            if(s1*s2<=0&&(s1!=0||s2!=0)){

stk[top++]=line_intersection(u,v);
            }
        }
        stk[top++]=u.a;
        stk[top++]=u.b;
        sort(stk,stk+top);
        top=unique(stk,stk+top)-stk;
        point mp,lp=stk[0];
        for(int i=1;i<top;i++){
            mp=(lp+stk[i])*0.5;
            if(!p_in_polygon(mp,pg))lp=stk[i];
            ans=max(ans,dist2(lp,stk[i]));
        }
        return sqrt(ans);
}
double maxlenth(polygon &pg){
        double ans=0.0;
        for(int i=0;i<pg.n-1;i++){
            for(int j=i+1;j<pg.n;j++){

ans=max(ans,seg_max_len(line(pg.p[i],pg.p[j]),
pg));

ans=max(ans,seg_max_len(line(pg.p[j],pg.p[i]),
pg));
            }
        }
        return ans;
}
//凸包对踵点长度
double opposite_lenth(polygon &pg){
        double ans=inf;
        int a,b,c;
        pg.p[pg.n]=pg.p[0];
        for(a=0,b=1,c=2;a<pg.n;a++,b++){

while(getarea(pg.p[a],pg.p[b],pg.p[c])<getarea
(pg.p[a],pg.p[b],pg.p[(c+1)%pg.n]))c=(c+1)%pg.
n;

ans=min(ans,dist_p_to_line(pg.p[c],line(pg.p[a
],pg.p[b])));
        }
        return ans;
}
//判p1,p2是否在l1,l2两侧
inline int opposite_side(point p1,point p2,point
l1,point l2) {
        return
xmult(l1,l2,p1)*xmult(l1,l2,p2)<-eps;
}
//判线段在任意多边形内,顶点按顺时针或逆时针给出,与边界
相交返回1
int seg_in_polygon(point l1,point l2,point
p[],int n) {
        point t[MAXN],tt;
        int i,j,k=0;
        if
(!p_in_polygon(l1,p,n)||!p_in_polygon(l2,p,n))
//不在内部
            return 0;
        for (i=0; i<n; i++)
            if
(opposite_side(l1,l2,p[i],p[(i+1)%n])&&opposit
e_side(p[i],p[(i+1)%n],l1,l2))
                return 0;
            else if (p_on_seg(l1,p[i],p[(i+1)%n]))
                t[k++]=l1;
            else if (p_on_seg(l2,p[i],p[(i+1)%n]))
                t[k++]=l2;
            else if (p_on_seg(p[i],l1,l2))
                t[k++]=p[i];
        for (i=0; i<k; i++)
            for (j=i+1; j<k; j++) {
                tt.x=(t[i].x+t[j].x)/2;
                tt.y=(t[i].y+t[j].y)/2;
                if (!p_in_polygon(tt,p,n))
                    return 0;
            }
        return 1;
}
//求直线交点,必须存在交点,或者预判断【解析几何方法】
point line_intersection(line u,line v) {
        double a1=u.b.y-u.a.y,b1=u.a.x-u.b.x;
        double c1=u.b.y*(-b1)-u.b.x*a1;
        double a2=v.b.y-v.a.y,b2=v.a.x-v.b.x;
        double c2=v.b.y*(-b2)-v.b.x*a2;
        double D=xmult(a1,b1,a2,b2);
        return
point(xmult(b1,c1,b2,c2)/D,xmult(c1,a1,c2,a2)/
D);
}
//求线段交点,必须存在交点,或者预判断【平面几何方法】
point line_intersection2(line u,line v) {
        point ret=u.a;
        double
t=xmult(u.a-v.a,v.b-v.a)/xmult(u.b-u.a,v.b-v.a
);
        t=fabs(t);
        ret.x+=(u.b.x-u.a.x)*t;
        ret.y+=(u.b.y-u.a.y)*t;
        return ret;
}
//三角形重心
point barycenter(point a,point b,point c){
        return (a+b+c)/3.0;
}
//多边形重心
point barycenter(point p[],int n) {
        point ret,t;
        double t1=0,t2;
        int i;
        ret.x=ret.y=0;
        for (i=1; i<n-1; i++)
            if
(fabs(t2=xmult(p[i+1],p[0],p[i]))>eps) {
                t=barycenter(p[0],p[i],p[i+1]);
                ret.x+=t.x*t2;
                ret.y+=t.y*t2;
                t1+=t2;
            }
        if (fabs(t1)>eps)
            ret.x/=t1,ret.y/=t1;
        return ret;
}
//求多边形面积
inline double getarea(point pg[],int n) {
        double area=0;
        pg[n]=pg[0];
        for(int i=0; i<n; i++)

area+=xmult(pg[i].x,pg[i].y,pg[i+1].x,pg[i+1].
y);
        return fabs(area)/2.0;
}
//解方程ax^2+bx+c=0
int equaltion(double a,double b,double c,double
&x1,double &x2) {
        double der=b*b-4*a*c;
        switch(sign(der)) {
        case -1:
            return 0;
        case 0:
            x1=-b/(2*a);
            return 1;
        case 1:
            der=sqrt(der);
            x1=(-b-der)/(2*a);
            x2=(-b+der)/(2*a);
            return 2;
        }
}
```

```
//线段与圆交点
int line_circle_intersection(line u,circle
c,point &p1,point &p2) {
     double dis=lenth(u.b-u.a);
     point d=(u.b-u.a)/dis;
     point E=c.o-u.a;
     double a=dmult(E,d);
     double a2=a*a;
     double e2=dmult(E,E);
     double r2=c.r*c.r;
     if((r2-e2+a2)<0){
          return 0;
     } else {
          double f=sqrt(r2 - e2 + a2);
          double t=a-f;
          int cnt=0;
          if(t>-eps&&t-dis<eps){//去掉后面变成射线
               p1=u.a+(d*t);
               cnt++;
          }
          t=a+f;
          if(t>-eps&&t-dis<eps) {
               p2=u.a+(d*t);
               cnt++;
          }
          return cnt;
     }
}
//给出在任意多边形内部的一个点
point a_point_in_polygon(polygon pg) {
     point v,a,b,r;
     int i,index;
     v=pg.p[0];
     index=0;
     for(i=1; i<pg.n; i++) {
          if(pg.p[i].y<v.y) {
               v=pg.p[i];
               index=i;
          }
     }
     a=pg.p[(index-1+pg.n)%pg.n];
     b=pg.p[(index+1)%pg.n];
     point q;
     polygon tri;
     tri.n=3;
     tri.p[0]=a;
     tri.p[1]=v;
     tri.p[2]=b;
     double md=inf;
     int in1=index;
     bool bin=false;
     for(i=0; i<pg.n; i++) {
          if(i == index)continue;
          if(i == (index-1+pg.n)%pg.n)continue;
          if(i == (index+1)%pg.n)continue;

if(!inside_convex2(pg.p[i],tri.p,3))continue;

          bin=true;
          if(dist(v,pg.p[i])<md) {
               q=pg.p[i];
               md=dist(v,q);
          }
     }
     if(!bin) {
          r.x=(a.x+b.x)/2;
          r.y=(a.y+b.y)/2;
          return r;
     }
     r.x=(v.x+q.x)/2;
     r.y=(v.y+q.y)/2;
     return r;
}
//求在多边形外面的点到凸包的切点
void p_cut_polygon(point p,polygon pg,point
&rp,point &lp) {
     line ep,en;
     bool blp,bln;
     rp=pg.p[0];
     lp=pg.p[0];
     for(int i=1; i<pg.n; i++) {
          ep.a=pg.p[(i+pg.n-1)%pg.n];
          ep.b=pg.p[i];
          en.a=pg.p[i];
          en.b=pg.p[(i+1)%pg.n];
          blp=xmult(ep.b-ep.a,p-ep.a)>=0;
          bln=xmult(en.b-en.a,p-en.a)>=0;
          if(!blp&&bln) {
               if(xmult(pg.p[i]-p,rp-p)>0)
                    rp=pg.p[i];
          }
          if(blp&&!bln) {
               if(xmult(lp-p,pg.p[i]-p)>0)
                    lp=pg.p[i];
          }
     }
     return ;
}
//判断点p在圆c内
bool p_in_circle(point p,circle c) {
     return c.r*c.r>dist2(p,c.o);
}
//求矩形第4个点
point rect4th(point a,point b,point c) {
     point d;
     if(abs(dmult(a-c,b-c))<eps) {
          d=a+b-c;
     }
     if(abs(dmult(a-b,c-b))<eps) {
          d=a+c-b;
     }
     if(abs(dmult(c-a,b-a))<eps) {
          d=c+b-a;
     }
     return d;
}
//判两圆关系
int CircleRelation(circle c1,circle c2){
     double d=lenth(c1.o-c2.o);
     if( fabs(d-c1.r-c2.r) < eps )
          return 2;//外切
     if( fabs(d-fabs(c1.r-c2.r)) < eps )
          return 4;//内切
     if( d > c1.r+c2.r )
          return 1;//相离
     if( d < fabs(c1.r-c2.r) )
          return 5;//内含
     if( fabs(c1.r-c2.r) < d && d < c1.r+c2.r)
          return 3;//相交
     return 0; //error!
}
//判圆与矩形关系,矩形水平
bool Circle_In_Rec(circle c,rectangle r) {
     if( r.a.x < c.o.x && c.o.x < r.b.x && r.c.y
< c.o.y && c.o.y < r.b.y ) {
          line line1(r.a, r.b);
          line line2(r.b, r.c);
          line line3(r.c, r.d);
          line line4(r.d, r.a);

if(c.r<dist_p_to_line(c.o,line1)&&c.r<dist_p_t
o_line(c.o,line2)&&c.r<dist_p_to_line(c.o,line
3)&&c.r<dist_p_to_line(c.o,line4))
               return true;
     }
     return false;
}
//射线关于平面的反射
void reflect(line2 u,line2 v,line2 &l){
     double n,m;
     double tpb,tpa;
     tpb=u.b*v.b+u.a*v.a;
     tpa=v.a*u.b-u.a*v.b;
     m=(tpb*u.b+tpa*u.a)/(u.b*u.b+u.a*u.a);
     n=(tpa*u.b-tpb*u.a)/(u.b*u.b+u.a*u.a);
     if(fabs(u.a*v.b-v.a*u.b)<1e-20) {
          l.a=v.a;
          l.b=v.b;
          l.c=v.c;
          return;
     }
     double xx,yy; //(xx,yy)是入射线与镜面的交点。
     xx=(u.b*v.c-v.b*u.c)/(u.a*v.b-v.a*u.b);
     yy=(v.a*u.c-u.a*v.c)/(u.a*v.b-v.a*u.b);
     l.a=n;
     l.b=-m;
     l.c=m*yy-xx*n;
}
//两圆交点（预判断不相交情况）
void c2point(circle c1,circle c2,point
&rp1,point &rp2) {
```

```
    double a,b,r;
    a=c2.o.x-c1.o.x;
    b=c2.o.y-c1.o.y;
    r=(a*a+b*b+c1.r*c1.r-c2.r*c2.r)/2;
    if(a==0&&b!=0) {
        rp1.y=rp2.y=r/b;
        rp1.x=sqrt(c1.r*c1.r-rp1.y*rp1.y);
        rp2.x=-rp1.x;
    } else if(a!=0&&b==0) {
        rp1.x=rp2.x=r/a;
        rp1.y=sqrt(c1.r*c1.r-rp1.x*rp2.x);
        rp2.y=-rp1.y;
    } else if(a!=0&&b!=0) {
        double delta;

delta=b*b*r*r-(a*a+b*b)*(r*r-c1.r*c1.r*a*a);
        rp1.y=(b*r+sqrt(delta))/(a*a+b*b);
        rp2.y=(b*r-sqrt(delta))/(a*a+b*b);
        rp1.x=(r-b*rp1.y)/a;
        rp2.x=(r-b*rp2.y)/a;
    }
    rp1=rp1+c1.o;
    rp2=rp2+c1.o;
}
//圆外一点引圆的切线
void cutpoint(circle c,point sp,point &rp1,point
&rp2) {
    circle c2;
    c2.o=(c.o+sp)/2.0;
    c2.r=lenth(c2.o-sp);
    c2point(c,c2,rp1,rp2);
}
//圆 c1 上，与 c2 的外切点
void c2cuto(circle c1, circle c2, point &p1,
point &p2) {
    double d = dist(c1.o, c2.o), dr = c1.r -
c2.r;
    double b = acos(dr / d);
    double a = angle(c2.o-c1.o);
    double a1 = a - b, a2 = a + b;
    p1=point(cos(a1) * c1.r, sin(a1) * c1.r) +
c1.o;
    p2=point(cos(a2) * c1.r, sin(a2) * c1.r) +
c1.o;
}
//圆 c1 上，与 c2 的内切点
void c2cuti(circle c1,circle c2,point &p1,point
&p2){
    point dr=c2.o-c1.o;
    dr=dr/lenth(dr);
    point a=c1.o-(dr*c1.r),b=c1.o+(dr*c1.r);
    point c=c2.o-(dr*c2.r),d=c2.o+(dr*c2.r);
    circle
E((a+c)/2.0,lenth(c-a)/2.0),F((b+d)/2.0,lenth(
d-b)/2.0);
    point q1,q2;
    c2point(E,F,q1,q2);
    point
L=line_intersection2(line(c1.o,c2.o),line(q1,q
2));
    circle c3((c1.o+L)/2.0,lenth(L-c1.o)/2.0);
    c2point(c1,c3,p1,p2);
}
```

/**************************************/

三维计算几何库

/**************************************/

```
#include<cstdlib>
#include<cmath>
#include<cstdio>
#include<algorithm>
#define max(a,b) (((a)>(b))?(a):(b))
#define min(a,b) (((a)>(b))?(b):(a))
#define sign(x) ((x)>eps?1:((x)<-eps?(-1):(0)))
using namespace std;
const int MAXN=1000;
const double eps=1e-8,inf=1e50;
struct point3{
    double x,y,z;
    point3(){}
    point3(double _x,double _y,double _z){
        x=_x;y=_y;z=_z;
    }
    point3 operator-(const point3 &ne){
        return point3(x-ne.x,y-ne.y,z-ne.z);
    }
    point3 operator+(const point3 &ne){
        return point3(x+ne.x,y+ne.y,z+ne.z);
    }
    point3 operator*(const double t){
        return point3(x*t,y*t,z*t);
    }
};
struct line3{
    point3 a,b;
    line3(){}
    line3(point3 _a,point3 _b){
        a=_a;
        b=_b;
    }
};
struct plane3{
    point3 a,b,c;
    plane3(){}
    plane3(point3 _a,point3 _b,point3 _c){
        a=_a;
        b=_b;
        c=_c;
    }
};
point3 xmult(point3 a,point3 b){
    return
point3(a.y*b.z-a.z*b.y,a.z*b.x-a.x*b.z,a.x*b.y
-a.y*b.x);
}
double dmult(point3 a,point3 b){
    return a.x*b.x+a.y*b.y+a.z*b.z;
}
double lenth(point3 v){
    return sqrt(v.x*v.x+v.y*v.y+v.z*v.z);
}
double dist(point3 a,point3 b){
    return lenth(a-b);
}
double dist2(point3 a,point3 b){
    return dmult(a-b,a-b);
}
//平面法向量
point3 pvec(plane3 s){
    return xmult(s.b-s.a,s.c-s.a);
}
//判定点是否在线段上，包括端点和共线
bool point_on_seg(point3 p,line3 s){
    return
sign(lenth(xmult(p-s.a,s.b-s.a)))==0&&(p.x-s.a
.x)*(p.x-s.b.x)<eps&&(p.y-s.a.y)*(p.y-s.b.y)<e
ps&&(p.z-s.a.z)*(p.z-s.b.y)<eps;
}
//判断点在平面上
bool point_on_plane(point3 p,plane3 s){
    return sign(dmult(p-s.a,pvec(s)))==0;
}
//判定点是否在空间三角形上，包括边界，三点共线无意义
bool point_in_triangle( point3 p, plane3 s ) {
    return
sign(lenth(xmult(s.a-s.b,s.a-s.c))-lenth(xmult
(p-s.a,p-s.b))-lenth(xmult(p-s.b,p-s.c))-lenth
(xmult(p-s.c,p-s.a)))!=0;
}
//判定点是否在空间三角形上，不包括边界，三点共线无意义
int point_in_triangle2( point3 p, plane3 s ) {
    return
point_in_triangle(p,s)&&lenth(xmult(p-s.a,p-s.
b))>eps&&lenth(xmult(p-s.b,p-s.c))>eps&&lenth(
xmult(p-s.c,p-s.a))>eps;
}
//判定两点在线段同侧，点在线段上返回 0，不共面无意义
bool same_side( point3 p1, point3 p2, line3 l )
{
    return
dmult(xmult(l.a-l.b,p1-l.b),xmult(l.a-l.b,p2-l
.b))>eps;
}
//判定两点在线段异侧，点在平面上返回 0
bool opposite_side( point3 p1, point3 p2, line3
l ) {
    return
dmult(xmult(l.a-l.b,p1-l.b),xmult(l.a-l.b,p2-l
.b))<-eps;
}
```

```
//判定两点在平面同侧，点在平面上返回 0
bool same_side( point3 p1, point3 p2, plane3 s )
{
     return
dmult(pvec(s),p1-s.a)*dmult(pvec(s),p2-s.a)>ep
s;
}
//判定两点在平面异侧，点在平面上返回 0
bool opposite_side( point3 p1, point3 p2, plane3
s ) {
     return
dmult(pvec(s),p1-s.a)*dmult(pvec(s),p2-s.a)<-e
ps;
}
//判断直线平行
bool parallel(line3 u,line3 v){
     return
sign(lenth(xmult(u.b-u.a,v.b-v.a)))==0;
}
//判定两线段相交，不包括端点和部分重合
bool seg_seg_inter( line3 u, line3 v ) {
     return
point_on_plane(u.a,plane3(u.b,v.a,v.b))&&oppos
ite_side(u.a,u.b,v)&&opposite_side(v.a,v.b,u);
}
//判定线段与空间三角形相交，包括交于边界和（部分）包含
int seg_triangle_inter( line3 l, plane3 s ) {

return !same_side(l.a,l.b,s)&&!same_side(s.a,s
.b,plane3(l.a,l.b,s.c))&&!same_side(s.b,s.c,pl
ane3(l.a,l.b,s.a))&&!same_side(s.c,s.a,plane3(
l.a,l.b,s.b));
}
//判定线段与空间三角形相交，不包括交于边界和（部分）包含
int seg_triangle_inter2( line3 l, plane3 s ){
     return opposite_side( l.a, l.b, s ) &&
opposite_side( s.a, s.b, plane3(l.a, l.b, s.c) )
&& opposite_side( s.b, s.c, plane3(l.a, l.b,
s.a) ) && opposite_side( s.c, s.a,plane3(l.a,
l.b, s.b) );
}
//面面平行
bool parallel(plane3 s1,plane3 s2){
     return
sign(lenth(xmult(pvec(s1),pvec(s2))))==0;
}
//判断直线垂直
bool vertical(line3 u,line3 v){
     return sign(dmult(u.b-u.a,v.b-v.a))==0;
}
//面面垂直
bool vertical(plane3 s1,plane3 s2){
     return sign(dmult(pvec(s1),pvec(s2)))==0;
}
//判断两直线的位置关系
int line_to_line(line3 u,line3 v){
     plane3 s1(u.a,u.b,v.a),s2(u.a,u.b,v.b);
     if(sign(lenth(xmult(pvec(s1),pvec(s2)))))
          return -1;//异面
     else if(parallel(u,v))
          return 0;//平行
     else
          return 1;//相交
}
//直线与平面关系
int line_to_plane(line3 u,plane3 s){
     if(sign(dmult(pvec(s),u.b-u.a))==0){
          if(point_on_plane(u.a,s))
               return -1;//直线在平面上
          else
               return 0;//直线平行于平面
     }
     else
          return 1;//线面相交
}
//线面求交
point3 line_plane_intersection(line3 u,plane3
s){
     point3 ret=pvec(s),der=u.b-u.a;
     double
t=dmult(ret,s.a-u.a)/dmult(ret,u.b-u.a);
     return u.a+der*t;
}
//线线求交
point3 line_interseciton(line3 u,line3 v){
     point3
ret=u.a,v1=xmult(u.a-v.a,v.b-v.a),v2=xmult(u.b
-u.a,v.b-v.a);
     double
t=lenth(v1)/lenth(v2)*(dmult(v1,v2)>0?-1:1);
     return ret+((u.b-u.a)*t);
}
//面面求交
line3 plane_intersection(plane3 u,plane3 v){
     line3 ret;


ret.a=(line_to_plane(line3(v.a,v.b),u)==0)?lin
e_plane_intersection(line3(v.b,v.c),u):line_pl
ane_intersection(line3(v.a,v.b),u);


ret.b=(line_to_plane(line3(v.c,v.a),u)==0)?lin
e_plane_intersection(line3(v.b,v.c),u):line_pl
ane_intersection(line3(v.a,v.c),u);
     return ret;
}
//点线距离
double dist_point_to_line(point3 p,line3 u){
     return
lenth(xmult(p-u.a,u.b-u.a))/dist(u.a,u.b);
}
//点面距离
double dist_point_to_plane(point3 p,plane3 s){
     point3 pv=pvec(s);
     return fabs(dmult(pv,p-s.a))/lenth(pv);
}
//线线距离
double dist_line_to_line(line3 u,line3 v ) {
     point3 p=xmult(u.a-u.b,v.a-v.b);
     return fabs(dmult(u.a-v.a,p))/lenth(p);
}
//点线垂足
point3 vertical_foot(point3 p,line3 u){
     double
t=dmult(p-u.a,u.b-u.a)/dist2(u.a,u.b);
     point3 ret=u.a;
     return ret+((u.b-u.a)*t);
}
//已知四面体六边求体积
double volume(double a,double b,double c,double
d,double e,double f){
     double
a2=a*a,b2=b*b,c2=c*c,d2=d*d,e2=e*e,f2=f*f;
     double tr1=acos((c2+b2-f2)/(2.0*b*c));
     double tr2=acos((a2+c2-e2)/(2.0*a*c));
     double tr3=acos((a2+b2-d2)/(2.0*a*b));
     double tr4=(tr1+tr2+tr3)/2.0;
     double
temp=sqrt(sin(tr4)*sin(tr4-tr1)*sin(tr4-tr2)*s
in(tr4-tr3));
     return a*b*c*temp/3.0;
}
//四面体体积
double volume(point3 a,point3 b,point3 c,point3
d){
     //abc 面方向与 d 一致时为正
     return
fabs(dmult(xmult(b-a,c-a),d-a))/6.0;
}
/**************************************/
                三角形
/**************************************/
//外心
point circumcenter(point a,point b,point c){
     line u,v;
     u.a=(a+b)*0.5;
     u.b.x=u.a.x-a.y+b.y;
     u.b.y=u.a.y+a.x-b.x;
     v.a=(a+c)*0.5;
     v.b.x=v.a.x-a.y+c.y;
     v.b.y=v.a.y+a.x-c.x;
     return line_intersection(u,v);
}
//内心
point incenter(point a,point b,point c){
     line u,v;
     double m,n;
     u.a=a;
     m=atan2(b.y-a.y,b.x-a.x);
```

```
      n=atan2(c.y-a.y,c.x-a.x);
      u.b.x=u.a.x+cos((m+n)*0.5);
      u.b.y=u.a.y+sin((m+n)*0.5);
      v.a=b;
      m=atan2(a.y-b.y,a.x-b.x);
      n=atan2(c.y-b.y,c.x-b.x);
      v.b.x=v.a.x+cos((m+n)*0.5);
      v.b.y=v.a.y+sin((m+n)*0.5);
      return line_intersection(u,v);
}
//垂心
point perpencenter(point a,point b,point c){
      line u,v;
      u.a=c;
      u.b.x=u.a.x-a.y+b.y;
      u.b.y=u.a.y+a.x-b.x;
      v.a=b;
      v.b.x=v.a.x-a.y+c.y;
      v.b.y=v.a.y+a.x-c.x;
      return line_intersection(u,v);
}
//重心
//到三角形三顶点距离的平方和最小的点
//三角形内到三边距离之积最大的点
point barycenter(point a,point b,point c){
      return (a+b+c)/3.0;
}
//费马点
//到三角形三顶点距离之和最小的点
point fermentpoint(point a,point b,point c){
      point u,v;
      double
step=fabs(a.x)+fabs(a.y)+fabs(b.x)+fabs(b.y)+f
abs(c.x)+fabs(c.y);
      int i,j,k;
      u=barycenter(a,b,c);
      while (step>1e-10)
            for (k=0;k<10;step/=2,k++)
                  for (i=-1;i<=1;i++)
                        for (j=-1;j<=1;j++){
                              v.x=u.x+step*i;
                              v.y=u.y+step*j;
                              if
(dist(u,a)+dist(u,b)+dist(u,c)>dist(v,a)+dist(
v,b)+dist(v,c))
                                    u=v;
                        }
      return u;
}
/***************************************/
```

凸包

```
/***************************************/
bool comp_angle(const point a,const point b) {
      int tmp=sign(xmult(o,a,b));
      if(tmp>0||(tmp==0&&dist2(o,a)<dist2(o,b)))
            return true;
      else
            return false;
}
//极角序方法
polygon convex_hull(point p[],int n) {
      polygon pg;
      int i,u=0,top;
      for(i=1; i<n; i++)

if(sign(p[i].y-p[u].y)<0||(sign(p[i].y-p[u].y)
==0&&p[i].x<p[u].x))
                  u=i;
      swap(p[0],p[u]);
      o=p[0];
      sort(p+1,p+n,comp_angle);
      for(i=top=0; i<n; i++) {

while(top>1&&xmult(pg.p[top-2],pg.p[top-1],p[i
])<eps)
                  top--;
            pg.p[top++]=p[i];
      }
      pg.n=top;
      return pg;
}
bool comp_cod(const point a,const point b) {
      return
a.x<b.x||((sign(a.x-b.x)==0)&&a.y<b.y);
}
//坐标序方法
polygon convex_hull2(point p[],int n) {
      polygon pg;
      int i,j,top;
      sort(p,p+n,comp_cod);
      for(i=top=0; i<n; i++) {

while(top>1&&xmult(pg.p[top-2],pg.p[top-1],p[i
])<eps)
                  top--;
            pg.p[top++]=p[i];
      }
      j=top;
      for(i=n-2; i>=0; i--) {

while(top>j&&xmult(pg.p[top-2],pg.p[top-1],p[i
])<eps)
                  top--;
            pg.p[top++]=p[i];
      }
      pg.n=top-1;
      return pg;
}
/***************************************/
```

两凸包的最短距离

```
/***************************************/
inline double rotate_caliper(polygon &pp,polygon
&qq) {
      int i,pn=pp.n,qn=qq.n,p,q;
      double tmp,ans=inf;
      for(p=0,i=1; i<pn; i++)

if(pp.p[i].y<pp.p[p].y-eps||(sign(pp.p[i].y-pp
.p[p].y)==0&&pp.p[i].x<pp.p[p].x)) p=i;
      for(q=0,i=1; i<qn; i++)

if(qq.p[i].y>qq.p[q].y+eps||(sign(qq.p[i].y-qq
.p[q].y)==0&&qq.p[i].x>qq.p[q].x)) q=i;
      pp.p[pn]=pp.p[0];
      qq.p[qq.n]=qq.p[0];
      for(i=0; i<pn; i++) {

while((tmp=xmult(qq.p[q+1],pp.p[p],pp.p[p+1])-
xmult(qq.p[q],pp.p[p],pp.p[p+1]))<-eps)
                  q=(q+1)%qn;
            if(tmp>eps)

ans=min(ans,dist_p_to_seg(qq.p[q],line(pp.p[p]
,pp.p[p+1])));
            else

ans=min(ans,dist_seg_to_seg(line(pp.p[p],pp.p[
p+1]),line(qq.p[q],qq.p[q+1])));
            p=(p+1)%pn;
      }
      return ans;
}
int main() {
      int pn,qn;
      while(scanf("%d%d",&pn,&qn)!=EOF&&pn&&qn)
{
            polygon pp,qq;
            pp.n=pn;
            qq.n=qn;
            for(int i=0; i<pn; i++)

scanf("%lf%lf",&pp.p[i].x,&pp.p[i].y);
            for(int i=0; i<qn; i++)

scanf("%lf%lf",&qq.p[i].x,&qq.p[i].y);
            if(!is_unclock(pp))
                  to_unclock(pp);
            if(!is_unclock(qq))
                  to_unclock(qq);

printf("%.5lf\n",min(rotate_caliper(pp,qq),rot
ate_caliper(qq,pp)));
      }
      return 0;
}
```

/**************************************/

凸包的直径

/**************************************/

```
polygon pg;
point p[N];
int n;
inline int rotate_caliper() {
    int ans=0;
    int i,j,n=pg.n;
    pg.p[n]=pg.p[0];
    for(i=0,j=1; i<n; i++) {

while(xmult(pg.p[i],pg.p[i+1],pg.p[j])<xmult(p
g.p[i],pg.p[i+1],pg.p[j+1]))
            j=(j+1)%n;

ans=max(ans,powerdist(pg.p[i],pg.p[j]));
    }
    return ans;
}
int main() {
    while(scanf("%d",&n)!=EOF) {
        for(int i=0; i<n; i++)
            scanf("%d%d",&p[i].x,&p[i].y);
        pg=convex_hull2(p,n);
        printf("%d\n",rotate_caliper());
    }
    return 0;
}
```

/**************************************/

凸包的最大内切圆

/**************************************/

```
void cut(line cur,polygon &pg) { //cur逆时针方
向为可行域
    polygon tp;
    int n=pg.n;
    pg.p[n]=pg.p[0];
    tp.n=0;
    for(int i=0; i<n; i++) {
        point p=pg.p[i],q=pg.p[i+1];
        double
pp=xmult(cur.a,cur.b,p),qq=xmult(cur.a,cur.b,q
);
        if(pp>-eps)
            tp.p[tp.n++]=p;
        if(pp*qq<-eps)

tp.p[tp.n++]=line_intersection(cur,line(p,q));
    }
    pg=tp;
    return;
}
line pushr(point a,point b,double r) {
    double tx=b.x-a.x,ty=b.y-a.y;
    double dx=-ty,dy=tx,D=sqrt(dx*dx+dy*dy);
    double lx=dx*r/D,ly=dy*r/D;
    return
line(point(a.x+lx,a.y+ly),point(b.x+lx,b.y+ly)
);
}
bool max_cir(polygon pg,double r) {
    polygon cg=pg;
    int n=pg.n;
    pg.p[n]=pg.p[0];
    for(int i=0; i<n; i++) {
        cut(pushr(pg.p[i],pg.p[i+1],r),cg);
        if(cg.n==0)
            return false;
    }
    return true;
}
int main() {
    int n;
    while(scanf("%d",&n)!=EOF&&n) {
        polygon pg,cg;
        pg.n=n;
        for(int i=0; i<n; i++)

scanf("%lf%lf",&pg.p[i].x,&pg.p[i].y);
        double ll=0,rr=1e6,mid,ans=0.0;
        while(rr-ll>eps) {
            mid=(rr+ll)/2.0;
            if(max_cir(pg,mid)) {
                ans=mid;
                ll=mid;
            } else
                rr=mid;
        }
        printf("%.6lf\n",ans);
    }
    return 0;
}
```

/**************************************/

三维凸包

/**************************************/

```
#include<cstdio>
#include<cstring>
#include<cmath>
#include<algorithm>
using namespace std;
#define PR 1e-8
#define N 510
struct TPoint
{
    double x,y,z;
    TPoint(){}
    TPoint(double _x,double _y,double
_z):x(_x),y(_y),z(_z){}
    TPoint operator-(const TPoint p) {return
TPoint(x-p.x,y-p.y,z-p.z);}
    TPoint operator*(const TPoint p) {return
TPoint(y*p.z-z*p.y,z*p.x-x*p.z,x*p.y-y*p.x);}/
/叉积
    double operator^(const TPoint p) {return
x*p.x+y*p.y+z*p.z;}//点积
};
struct fac//
{
    int a,b,c;//凸包一个面上的三个点的编号
    bool ok;//该面是否是最终凸包中的面
};
struct T3dhull
{
    int n;//初始点数
    TPoint ply[N];//初始点
    int trianglecnt;//凸包上三角形数
    fac tri[N];//凸包三角形
    int vis[N][N];//点 i 到点 j 是属于哪个面
    double dist(TPoint a){return
sqrt(a.x*a.x+a.y*a.y+a.z*a.z);}//两点长度
    double area(TPoint a,TPoint b,TPoint
c){return dist((b-a)*(c-a));}//三角形面积*2
    double volume(TPoint a,TPoint b,TPoint
c,TPoint d){return (b-a)*(c-a)^(d-a);}//四面体有
向体积*6
    double ptoplane(TPoint &p,fac &f)//正：点在面
同向
    {
        TPoint
m=ply[f.b]-ply[f.a],n=ply[f.c]-ply[f.a],t=p-pl
y[f.a];
        return (m*n)^t;
    }
    void deal(int p,int a,int b)
    {
        int f=vis[a][b];
        fac add;
        if(tri[f].ok)
        {
            if((ptoplane(ply[p],tri[f]))>PR)
dfs(p,f);
            else
            {
          add.a=b,add.b=a,add.c=p,add.ok=1;
vis[p][b]=vis[a][p]=vis[b][a]=trianglecnt;
                tri[trianglecnt++]=add;
            }
        }
    }
    void dfs(int p,int cnt)//维护凸包，如果点 p 在凸
包外更新凸包
    {
        tri[cnt].ok=0;
        deal(p,tri[cnt].b,tri[cnt].a);
        deal(p,tri[cnt].c,tri[cnt].b);
```

```
        deal(p,tri[cnt].a,tri[cnt].c);
    }
    bool  same(int  s,int  e)//判断两个面是否为同一面
    {
        TPoint
a=ply[tri[s].a],b=ply[tri[s].b],c=ply[tri[s].c
];
        return
fabs(volume(a,b,c,ply[tri[e].a]))<PR

&&fabs(volume(a,b,c,ply[tri[e].b]))<PR

&&fabs(volume(a,b,c,ply[tri[e].c]))<PR;
    }
    void  construct()//构建凸包
    {
        int i,j;
        trianglecnt=0;
        if(n<4) return ;
        bool tmp=true;
        for(i=1;i<n;i++)//前两点不共点
        {
            if((dist(ply[0]-ply[i]))>PR)
            {
                swap(ply[1],ply[i]); tmp=false;
break;
            }
        }
        if(tmp) return;
        tmp=true;
        for(i=2;i<n;i++)//前三点不共线
        {

if((dist((ply[0]-ply[1])*(ply[1]-ply[i])))>PR)
            {
                swap(ply[2],ply[i]); tmp=false;
break;
            }
        }
        if(tmp) return ;
        tmp=true;
        for(i=3;i<n;i++)//前四点不共面
        {

if(fabs((ply[0]-ply[1])*(ply[1]-ply[2])^(ply[0
]-ply[i]))>PR)
            {
                swap(ply[3],ply[i]); tmp=false;
break;
            }
        }
        if(tmp) return ;
        fac add;
        for(i=0;i<4;i++)//构建初始四面体
        {

add.a=(i+1)%4,add.b=(i+2)%4,add.c=(i+3)%4,add.
ok=1;
            if((ptoplane(ply[i],add))>0)
swap(add.b,add.c);

vis[add.a][add.b]=vis[add.b][add.c]=vis[add.c]
[add.a]=trianglecnt;
            tri[trianglecnt++]=add;
        }
        for(i=4;i<n;i++)//构建更新凸包
        {
            for(j=0;j<trianglecnt;j++)
            {
if(tri[j].ok&&(ptoplane(ply[i],tri[j]))>PR)
                {
                    dfs(i,j); break;
                }
            }
        }
        int cnt=trianglecnt;
        trianglecnt=0;
        for(i=0;i<cnt;i++)
        {
            if(tri[i].ok)
                tri[trianglecnt++]=tri[i];
        }
    }
    double  area()//表面积
    {
        double ret=0;
        for(int i=0;i<trianglecnt;i++)

ret+=area(ply[tri[i].a],ply[tri[i].b],ply[tri[
i].c]);
        return ret/2.0;
    }
    double  volume()//体积
    {
        TPoint p(0,0,0);
        double ret=0;
        for(int i=0;i<trianglecnt;i++)

ret+=volume(p,ply[tri[i].a],ply[tri[i].b],ply[
tri[i].c]);
        return fabs(ret/6);
    }
    int facetri() {return trianglecnt;}//表面三角
形数
    int facepolygon()//表面多边形数
    {
        int ans=0,i,j,k;
        for(i=0;i<trianglecnt;i++)
        {
            for(j=0,k=1;j<i;j++)
            {
                if(same(i,j)) {k=0;break;}
            }
            ans+=k;
        }
        return ans;
    }

}hull;
int main()
{
    while(~scanf("%d",&hull.n))
    {
        int i;
        for(i=0;i<hull.n;i++)

scanf("%lf%lf%lf",&hull.ply[i].x,&hull.ply[i].
y,&hull.ply[i].z);
        hull.construct();
        printf("%.3lf\n",hull.area());
    }
    return 0;
}
```

/**************************************/

半平面交

/**************************************/

```
#include<cstdlib>
#include<cmath>
struct line{
     point a,b;
     double ang;
     line(){}
     line(point _a,point
_b){a=_a;b=_b;ang=atan2(b.y-a.y,b.x-a.x);}
}L[MAXN];
bool comp(line u,line v){
     return
u.ang<v.ang||(sign(u.ang-v.ang)==0&&xmult(u.a,
u.b,v.a)>eps);
}
bool onleft(line& s,point& p) {
   return xmult(s.b-s.a, p-s.a)>eps;
}
polygon HalfPlane_Intersection(int n) {
     sort(L, L+n,comp);  // 按极角排序
     int first, last;
     vector<point> p(n);        // p[i]为q[i]和
q[i+1]的交点
     vector<line> q(n);         // 双端队列
     polygon ans;          // 结果
     ans.n=0;
     q[first=last=0] = L[0];  // 双端队列初始化为
只有一个半平面L[0]
     for(int i = 1; i < n; i++) {
          while(first < last && !onleft(L[i],
p[last-1])) last--;
          while(first < last && !onleft(L[i],
p[first])) first++;
          q[++last] = L[i];
```

```
            if(fabs(xmult(q[last].b-q[last].a,
q[last-1].b-q[last-1].a))  < eps)  {  // 两向量平
行且同向，取内侧的一个
                last--;
                if(onleft(q[last], L[i].a))
q[last] = L[i];
            }
            if(first < last) p[last-1] =
line_intersection(q[last-1], q[last]);
        }
        while(first < last && !onleft(q[first],
p[last-1]))  last--;  // 删除无用平面
        if(last - first <= 1) return ans; // 空
集
        p[last] =line_intersection(q[last],
q[first]);  // 计算首尾两个半平面的交点
        for(int i = first; i <= last; i++)
ans.p[ans.n++]=p[i];
        return ans;
}
int main(){
        int n,num;
        while(scanf("%d",&n)!=EOF){
            num=0;
            double sx,sy,ex,ey;
            for(int i=0;i<n;i++){

scanf("%lf%lf%lf%lf",&sx,&sy,&ex,&ey);

L[num++]=line(point(sx,sy),point(ex,ey));
            }
            double big=1e4;

L[num++]=line(point(0,0),point(big,0));

L[num++]=line(point(big,0),point(big,big));

L[num++]=line(point(big,big),point(0,big));

L[num++]=line(point(0,big),point(0,0));

printf("%.1lf\n",getarea(HalfPlane_Intersectio
n(num)));
        }
        return 0;
}
```

/***/

平面最远点对

/***/

```
inline double rotate_caliper() {
        double ans=0;
        int i,j,n=pg.n;
        pg.p[n]=pg.p[0];
        for(i=0,j=1; i<n; i++) {

while(xmult(pg.p[i],pg.p[i+1],pg.p[j])<xmult(p
g.p[i],pg.p[i+1],pg.p[j+1]))
                j=(j+1)%n;
            ans=max(ans,dist2(pg.p[i],pg.p[j]));
        }
        return sqrt(ans);
}
```

/***/

网格

/***/

```
#define abs(x) ((x)>0?(x):-(x))
struct point{int x,y;};
int gcd(int a,int b){return b?gcd(b,a%b):a;}
//多边形上的网格点个数
int grid_onedge(int n,point* p){
    int i,ret=0;
    for (i=0;i<n;i++)

ret+=gcd(abs(p[i].x-p[(i+1)%n].x),abs(p[i].y-p
[(i+1)%n].y));
    return ret;
}
//多边形内的网格点个数
int grid_inside(int n,point* p){
    int i,ret=0;
    for (i=0;i<n;i++)
        ret+=p[(i+1)%n].y*(p[i].x-p[(i+2)%n].x);
    return (abs(ret)-grid_onedge(n,p))/2+1;
}
```

/***/

两圆交面积

/***/

```
double area_cir_to_cir(circle a,circle b) {
        double d=dist(a.o,b.o),r1=a.r,r2=b.r,r;
        if (r1+r2<=d) {
            return 0.0;
        } else if (fabs(r1-r2)>=d) {
            r=min(r1,r2);
            return Pi*r*r;
        } else {
            double a1=(r1*r1+d*d-r2*r2)/(2*r1*d);
            double a2=(r2*r2+d*d-r1*r1)/(2*r2*d);
            a1=2*acos(a1);
            a2=2*acos(a2);
            return
(r1*r1*(a1-sin(a1))+r2*r2*(a2-sin(a2)))*0.5;
        }
}
```

/***/

最小圆覆盖

/***/

```
struct triangle {
        point t[3];
};
circle c;
point p[N];
double triangleArea(triangle t) {
        point p1, p2;
        p1 = t.t[1] - t.t[0];
        p2 = t.t[2] - t.t[0];
        return fabs(p1.x * p2.y - p1.y * p2.x)*0.5;
}
circle circumcircleOfTriangle(triangle t) {
        circle tmp;
        double a, b, c, c1, c2;
        double xA, yA, xB, yB, xC, yC;
        a = dist(t.t[0], t.t[1]);
        b = dist(t.t[1], t.t[2]);
        c = dist(t.t[2], t.t[0]);
        tmp.r = a * b * c / triangleArea(t) / 4;
        tmp.o=circumcenter(t.t[0],t.t[1],t.t[2]);
        return tmp;
}
circle MinCircle2(int tce, triangle ce) {
        circle tmp;
        if(tce == 0) tmp.r = -2;
        else if(tce == 1) {
            tmp.o = ce.t[0];
            tmp.r = 0;
        } else if(tce == 2) {
            tmp.r = dist(ce.t[0], ce.t[1]) / 2;
            tmp.o.x = (ce.t[0].x + ce.t[1].x) / 2;
            tmp.o.y = (ce.t[0].y + ce.t[1].y) / 2;
        } else if(tce == 3) tmp =
circumcircleOfTriangle(ce);
        return tmp;
}
void MinCircle(int t, int tce, triangle ce) {
        int i, j;
        point tmp;
        c = MinCircle2(tce, ce);
        if(tce == 3) return;
        for(i = 1; i <= t; i++) {
            if(dist(p[i], c.o) > c.r) {
                ce.t[tce] = a[i];
                MinCircle(i - 1, tce + 1, ce);
                tmp = p[i];
                for(j = i; j >= 2; j--) {
                    a[j] = p[j - 1];
                }
                p[1] = tmp;
            }
        }
}
circle work(int n) {
        triangle ce;
        int i;
        MinCircle(n,0,ce);
```

```
    return c;
}
/*************************************/
```

单位圆覆盖最多点

```
/*************************************/
point find_centre(point p1, point p2) {
    point p3, mid, centre;
    double b, c, ang;
    p3.x = p2.x - p1.x;
    p3.y = p2.y - p1.y;
    mid.x = (p1.x + p2.x) / 2;
    mid.y = (p1.y + p2.y) / 2;
    b = dist2(p1, mid);
    c = sqrt(1 - b);
    if(fabs(p3.y) < eps) { //垂线的斜角 90 度
        centre.x = mid.x;
        centre.y = mid.y + c;
    } else {
        ang = atan(-p3.x / p3.y);
        centre.x = mid.x + c * cos(ang);
        centre.y = mid.y + c * sin(ang);
    }
    return centre;
}
int main() {
    int n, ans, tmpans, i, j, k;
    point p[305], centre;
    double tmp;
    while(scanf("%d", &n) && n) {
        for(i = 0; i < n; i++)
            scanf("%lf%lf", &p[i].x,
&p[i].y);
        ans = 1;
        for(i = 0; i < n; i++)
            for(j = i + 1; j < n; j++) {
                if(dist2(p[i], p[j]) > 4)
continue;
                tmpans = 0;
                centre = find_centre(p[i],
p[j]);
                for(k = 0; k < n; k++) {
                    tmp = dist2(centre,
p[k]);
                    if(tmp <= 1.000001)
tmpans++;
                }
                if(ans < tmpans) ans =
tmpans;
            }
        printf("%d\n", ans);
    }
    return 0;
}
```

```
/*************************************/
```

最小球覆盖

```
/*************************************/
#include<stdio.h>
#include<math.h>
#include<memory>
#include<stdlib.h>
using namespace std;
const double eps = 1e-10;
struct point_type { double x, y, z; };
int npoint, nouter ;
point_type point [1000], outer[4], res;
double radius, tmp ;
inline double dist(point_type p1 , point_type p2)
{
double dx=p1.x-p2.x, dy=p1.y-p2.y,dz=p1.z-p2.z ;
return ( dx*dx + dy*dy + dz*dz ) ;
}
inline double dot( point_type p1 , point_type p2 )
{
return p1.x*p2.x + p1.y*p2.y + p1.z*p2.z;
}
void ball()
{
   point_type q[3];
   double m[3][3],sol[3],L[3],det; int i,j;
   res.x=res.y=res.z=-1000;
   radius=0;
   switch ( nouter )
      {
       case 1 : res=outer[0]; break;
       case 2 :
res.x=(outer[0].x+outer[1].x)/2;
                res.y=(outer[0].y+outer[1].y)/2;
                res.z=(outer[0].z+outer[1].z)/2;
                radius=dist(res,outer[0]);
                break;
       case 3 :
                for ( i=0; i<2; ++i ) {

q[i].x=outer[i+1].x-outer[0].x;

q[i].y=outer[i+1].y-outer[0].y;

q[i].z=outer[i+1].z-outer[0].z;
                }
                for ( i=0; i<2; ++i )
                    for ( j=0; j<2; ++j )
                      m[i][j]=dot(q[i],q[j])*2 ;
                for ( i=0; i<2; ++i )
sol[i]=dot(q[i],q[i]);
                if
(fabs(det=m[0][0]*m[1][1]-m[0][1]*m[1][0]) <
eps ) return ;


L[0]=(sol[0]*m[1][1]-sol[1]*m[0][1])/det;

L[1]=(sol[1]*m[0][0]-sol[0]*m[1][0])/det;

res.x=outer[0].x+q[0].x*L[0]+q[1].x*L[1];

res.y=outer[0].y+q[0].y*L[0]+q[1].y*L[1];

res.z=outer[0].z+q[0].z*L[0]+q[1].z*L[1];
                radius=dist(res,outer[0]);
                break;
       case 4 :
                for ( i=0; i<3; ++i ){

q[i].x=outer[i+1].x-outer[0].x;

q[i].y=outer[i+1].y-outer[0].y;

q[i].z=outer[i+1].z-outer[0].z;
                  sol[i]=dot(q[i],q[i]);
                  }
                for ( i=0; i<3; ++i)
                   for ( j=0; j<3; ++j)
m[i][j]=dot(q[i],q[j])*2;

det=m[0][0]*m[1][1]*m[2][2]+m[0][1]*m[1][2]*m[
2][0]+m[0][2]*m[2][1]*m[1][0]-m[0][2]*m[1][1]*
m[2][0]-m[0][1]*m[1][0]*m[2][2]-m[0][0]*m[1][2
]*m[2][1];

                if ( fabs( det )<eps ) return;

                for ( j=0; j<3; ++j ){
                   for ( i=0; i<3; ++i )
m[i][j]=sol[i];
                    L[j]=( m[0][0]*m[1][1]*m[2][2]
+ m[0][1]*m[1][2]*m[2][0]
                         + m[0][2]*m[2][1]*m[1][0]
- m[0][2]*m[1][1]*m[2][0]
                         - m[0][1]*m[1][0]*m[2][2]
- m[0][0]*m[1][2]*m[2][1]
                         ) / det;
                   for( i=0; i<3; ++i )
m[i][j]=dot(q[i],q[j])*2;
                }
                res=outer[0];
                for ( i=0; i<3; ++i ) {
                   res.x+=q[i].x*L[i];
                   res.y+=q[i].y*L[i];
                   res.z+=q[i].z*L[i];
                   }
              radius=dist(res,outer[0]);
       }
}
void minball(int n)
{
  ball();
  if ( nouter <4 )
       for ( int i=0; i<n; ++i )
         if( dist(res,point[i])-radius>eps)
           {
              outer[nouter]=point[i];
```

```
            ++nouter;
            minball(i);
            --nouter;
            if(i>0)
            {
              point_type Tt = point[i] ;
              memmove(&point[1], &point[0] ,
sizeof ( point_type )*i );
              point[0]=Tt;
            }
          }
}
int main()
{
 int i;
 while(scanf("%d",&npoint)!=EOF,npoint)
   {
    for(i=0;i<npoint;i++)

scanf("%lf%lf%lf",&point[i].x,&point[i].y,&poi
nt[i].z);
    nouter=0;
    minball(npoint);
    printf("%.8lf\n",sqrt(radius)+eps);
   }
 return 0;
}
```

/**************************************/

圆和多边形的交

/**************************************/

```
//顶点标号从1开始
struct state_type {
     double angle;
     double CoverArea;
};
point p[M];
point center;
int n;
double R;
inline double area_triangle(point &a,point
&b,point &c) {
     return (b.x - a.x) * (c.y - a.y) - (c.x
- a.x) * (b.y - a.y);
}
void init() {
     int i;
     double temp,sum;
     for (i=2; i<n; i++) {
          temp = area_triangle(p[1],p[i],p[i +
1]);
          sum += temp;
     }
     if (sum < 0) reverse(p + 1,p + n + 1);
     p[n + 1] = p[1];
}
inline bool inCircle(point &s) {
     return dist(center,s) <= R;
}
bool SameSide(point a,point b) {
     if (dist(a,center) > dist(b,center))
swap(a,b);
     return dmult(a,b,center) < eps;
}
double ShadomOnCircle(point a,point b) {
     double flag = area_triangle(center,a,b),res
= 0;
     if (fabs(flag) < eps) return 0;

     bool ina = inCircle(a),inb = inCircle(b);
     if (ina && inb) {
          res = fabs(area_triangle(center,a,b))
/ 2;
     } else if (!ina && !inb) {
          if (SameSide(a,b)) {
               double theta =
acos(dmult(center,a,b) / dist(center,a) /
dist(center,b));
               res = R * R * theta / 2;
          } else {
               double height =
fabs(area_triangle(center,a,b)) / dist(a,b);
               double theta =
acos(dmult(center,a,b) / dist(center,a) /
dist(center,b));
               if (height >= R) {
                    res = R * R * theta / 2;
               } else {
                    double _theta = 2 *
acos(height / R);
                    res = R * R * (theta - _theta)
/ 2 + R * R * sin(_theta) / 2;
               }
          }
     } else {
          if (!ina && inb) swap(a,b);
          double height =
fabs(area_triangle(center,a,b)) / dist(a,b);
          double temp = dmult(a,center,b);
          double theta = acos(dmult(center,a,b)
/ dist(center,a) /
dist(center,b)),theta1,theta2;
          if (fabs(temp) < eps) {
               double _theta = acos(height / R);
               res += R * height / 2 *
sin(_theta);
               res += R * R / 2 * (theta - _theta);
          } else {
               theta1 = asin(height / R);
               theta2 = asin(height /
dist(a,center));
               if (temp > 0) {
                    res += dist(center,a) * R /
2 * sin(PI - theta1 - theta2);
                    res += R * R / 2 * (theta
+ theta1 + theta2 - PI);
               } else {
                    res += dist(center,a) * R /
2 * sin(theta2 - theta1);
                    res += R * R / 2 * (theta
- theta2 + theta1);
               }
          }
     }
     if (flag < 0) return -res;
     else return res;
}
double Cover() {
     int i;
     double res = 0;
     for (i=1; i<=n; i++)
          res += ShadomOnCircle(p[i],p[i + 1]);
     return res;
}
int main() {
     double ans;
     while (read_data()) {
          init();
          ans = Cover();
          printf("%.2lf\n",ans);
     }
     return 0;
}
```

/**************************************/

直线关于圆的反射

/**************************************/

```
#include <iostream>
#include <cmath>
using namespace std;
#define INF 999999999
const double eps = 1e-6;
int up;
typedef struct TPoint
{
   double x;
   double y;
}TPoint;
typedef struct TCircle
{
   TPoint center;
   double r;
}TCircle;
typedef struct TLine
{
   //直线标准式中的系数
   double a, b, c;
}TLine;
void SloveLine(TLine &line, TPoint start, TPoint
dir)
{
   //根据直线上一点和直线的方向求直线的方程
```

```
    if(dir.x == 0){
        line.a = 1;
        line.b = 0;
        line.c = start.x;
    }
    else {
        double k = dir.y / dir.x;
        line.a = k;
        line.b = -1;
        line.c = start.y - k * start.x;
    }
}
TLine lineFromSegment(TPoint p1, TPoint p2)
{
    //线段所在直线,返回直线方程的三个系统
    TLine tmp;
    tmp.a = p2.y - p1.y;
    tmp.b = p1.x - p2.x;
    tmp.c = p2.x * p1.y - p1.x * p2.y;
    return tmp;
}
TPoint symmetricalPointofLine(TPoint p, TLine L)
{
    //p 点关于直线 L 的对称点
    TPoint p2;
    double d;
    d = L.a * L.a + L.b * L.b;
    p2.x = (L.b * L.b * p.x - L.a * L.a * p.x -
           2 * L.a * L.b * p.y - 2 * L.a * L.c) /
d;
    p2.y = (L.a * L.a * p.y - L.b * L.b * p.y -
           2 * L.a * L.b * p.x - 2 * L.b * L.c) /
d;
    return p2;
}
double distanc(TPoint p1, TPoint p2)
{
    //计算平面上两个点之间的距离
   return sqrt((p1.x - p2.x) * (p1.x - p2.x) + (p1.y
- p2.y) * (p1.y - p2.y));
}
bool samedir(TPoint dir, TPoint start, TPoint
point)
{
    //判断方向
    TPoint tmp;
    tmp.x = point.x - start.x;
    tmp.y = point.y - start.y;
    if(tmp.x != 0 && dir.x != 0){
        if(tmp.x / dir.x > 0) return true;
        else return false;
    }
    else if(tmp.y != 0 && dir.y != 0){
        if(tmp.y / dir.y > 0) return true;
        else return false;
    }
    return true;
}
bool Intersected(TPoint &point, TLine line, const
TCircle circle[],
                 TPoint start, TPoint dir, int
which)
{
    //如果圆与直线有(有效交点)交点就存放在变量 point 中
    double a = line.a, b = line.b, c = line.c;
    double x0 = circle[which].center.x, y0 =
circle[which].center.y;
    double r =  circle[which].r;
    //有交点，求交点
    double x2front = b * b + a * a;
    double x1front = -2 * x0 * b * b + 2 * a * b
* y0 + 2 * a * c;
    double front = x0 * x0 * b * b + y0 * y0 * b
* b
        + c * c + 2 * c * y0 * b - b * b * r * r;
    double d = x1front * x1front - 4 * x2front *
front;
    TPoint p1, p2;
    bool k1, k2;
    if(fabs(d) < eps){
        //x2front 不可能等于零
       point.x = -x1front / x2front / 2;
       point.y = (-c - a * point.x) / b;
       //判断方向
       if(samedir(dir, start, point)) return true;
       else return false;
    }
    else if(d < 0) return false;
    else {
        p1.x = (-x1front + sqrt(d)) / 2 / x2front;
        p1.y = (-c - a * p1.x) / b;
        p2.x = (-x1front - sqrt(d)) / 2 / x2front;
        p2.y = (-c - a * p2.x) / b;
        k1 = samedir(dir, start, p1);
        k2 = samedir(dir, start, p2);
        if(k1 == false && k2 == false) return false;
        if(k1 == true && k2 == true){
            double dis1 = distanc(p1, start);
            double dis2 = distanc(p2, start);
            if(dis1 < dis2) point = p1;
            else point = p2;
            return true;
        }
        else if(k1 == true) point = p1;
        else point = p2;
        return true;
    }
}
void Reflect(int &num, TCircle circle[], TPoint
start, TPoint dir, int n)
{
    //反复反射
    int i;
    TLine line;
    TPoint interpoint, newstart;
    int u;
    SloveLine(line, start, dir);
    int tag = 0;
    double mindis = INF;
    for(i = 1;i <= n;i++){
        if(i != up && Intersected(interpoint, line,
circle, start, dir, i)){
            double dis = distanc(start,
interpoint);
            if(dis < mindis){
                tag = 1;
                u = i;
                mindis = dis;
                newstart = interpoint;
            }
        }
    }
    if(tag == 0){
        cout << "inf" << endl;
        return ;
    }
    else {
        if(num == 10){
            cout << "..." << endl;
           return ;
        }
        cout << u << " ";
        num++;
        //新的方向
        TLine line1;
        TPoint p;
        line1 = lineFromSegment(newstart,
circle[u].center);
        if(fabs(line1.a * start.x + line1.b *
start.y +line1.c) <= eps){
            dir.x = -dir.x;
            dir.y = -dir.y;
        }
        else {
            p = symmetricalPointofLine(start,
line1);//start 的对称点
            dir.x = p.x - newstart.x;
            dir.y = p.y - newstart.y;
        }

        start = newstart;
        up = u;
        Reflect(num, circle, start, dir, n);
    }
}
int main()
{
    //freopen("fzu_1035.in", "r", stdin);
    //freopen("fzu_1035.out", "w", stdout);
    int n, i, j, num, test = 1;
    TCircle circle[30];
    TPoint start, dir;
    while(cin >> n && n){
```

```
        for(i = 1;i <= n;i++){
            cin >> circle[i].center.x >>
circle[i].center.y >> circle[i].r;
        }
        cin >> start.x >> start.y >> dir.x >> dir.y;
        cout << "Scene " << test++ << endl;
        num = 0;
        up = -1;
        Reflect(num, circle, start, dir, n);
        cout << endl;
    }
    return 0;
}
```

/**************************************/

扇形的重心

/**************************************/

```
//Xc = 2*R*sinA/3/A
//A为圆心角的一半
#include <stdio.h>
#include <math.h>
int main()
{
    double r, angle;
    while(scanf("%lf%lf", &r, &angle) != EOF){
        angle /= 2;
        printf("%.6lf\n", 2 * r * sin(angle) / 3 /
angle);
    }
    return 0;
}
```

/**************************************/

三角形内部点数

/**************************************/

s[i][j]代表直线 ij 的右方的点数（不包括 ij）
idx[i][j]代表从 i 到 x 轴正方向至 i 到 j 的向量的扇形区域的点数（不包括 ij）

```
struct pt
{
    long long x,y;
    int id;
}p[size];
struct angle
{
    double a;
    int id;
    angle(){}
    angle(double ang,int idd)
    {
        a=ang,id=idd;
    }
    friend bool operator < (angle a,angle b)
    {
        return a.a<b.a;
    }
}q[size];
inline double calang (pt a,pt b)//a->b,并且保
证平行于 x 轴正方向的角度为最小
{
    double res=atan2((b.y-a.y)*1.,b.x-a.x);
    res+=res<0?2*pi:0;
    return res;
}
int s[size][size],idx[size][size];
void init ()
{
    int i,j,k,cnt,qn;
    for (i=0;i<n;i++)
    {
        qn=0;
        for (j=0;j<n;j++)
            if (i!=j)

q[qn++]=angle(calang(p[i],p[j]),p[j].id);
        sort(q,q+qn);
        for (j=0;j<qn;j++)
        {
            q[j+qn]=q[j];
            q[j+qn].a+=2*pi;
        }
        cnt=0;
        for (j=0,k=0;j<qn;j++)
        {
            while (q[k].a-q[j].a<pi)
                cnt++,k++;
            s[i][q[j].id]=n-cnt-1;
            idx[i][q[j].id]=j;
            cnt--;
        }
    }
}
inline long long X (pt a,pt b,pt c)//叉积 a->b
X a->c
{
    long long x1=b.x-a.x;
    long long y1=b.y-a.y;
    long long x2=c.x-a.x;
    long long y2=c.y-a.y;
    return x1*y2-x2*y1;
}
inline int getnum (int a,int b,int c)//保证 a->c
在 a->b 的逆时针方向
{
    if (X(p[a],p[b],p[c])<0) swap(b,c);
    int res=idx[a][c]-idx[a][b]-1;//减 1 表示减
去 b 点
    return res>=0?res:res+n-1;//考虑 a->c 和
a->b 在 x 轴两侧
}
inline int find (int a,int b,int c)
{
    int ret=0;
    if (X(p[a],p[b],p[c])<0) swap(b,c);

ret+=getnum(a,b,c)+getnum(b,c,a)+getnum(c,a
,b);
    ret+=s[a][b]+s[b][c]+s[c][a];
    ret-=2*(n-3);
    return ret;
}
```

/**************************************/

Pick 公式求面积

/**************************************/

```
//A = b / 2 + i -1  其中 b 与 i 分别表示在边界上及内
部的格子点之个数
int triangleArea(point p1, point p2, point p3)
{
    int k = p1.x * p2.y + p2.x * p3.y + p3.x
* p1.y
            - p2.x * p1.y - p3.x * p2.y - p1.x
* p3.y;
    if(k < 0) return -k;
    else return k;
}
line2 lineFromSegment(point p1, point p2) {
    line2 tmp;
    tmp.a = p2.y - p1.y;
    tmp.b = p1.x - p2.x;
    tmp.c = p2.x * p1.y - p1.x * p2.y;
    return tmp;
}
int Count(point p1, point p2) {
    int i, sum = 0, y;
    line2 l1 = lineFromSegment(p1, p2);
    if(l1.b == 0) return abs(p2.y - p1.y) + 1;
    if(p1.x > p2.x) swap(p1.x, p2.x);
    for(i = p1.x; i <= p2.x; i++) {
        y = -l1.c - l1.a * i;
        if(y % l1.b == 0) sum++;
    }
    return sum;
}
int main() {
    point p1, p2, p3;
    while(scanf("%d%d%d%d%d%d", &p1.x, &p1.y,
&p2.x, &p2.y, &p3.x, &p3.y) != EOF) {
        if(p1.x == 0 && p1.y == 0 && p2.x ==
0 && p2.y == 0 && p3.x == 0 && p3.y == 0) break;
        int A = triangleArea(p1, p2, p3);
        int b = 0;
        int i;
```

```
          b = Count(p1, p2) + Count(p1, p3) +
Count(p3, p2) - 3;
          i = (A - b) / 2 + 1;
          printf("%d\n", i);
     }
     return 0;
}
```

/**************************************/

共线最多的点的个数

/**************************************/

```
struct Line {
     int a, b;
     double k;
     bool end;
} line[N*N];
int n, ncount;
bool operator <(const Line &a, const Line &b)
{
     if (abs(a.k - b.k) > eps && !a.end && !b.end)
          return a.k < b.k;
     if (a.end != b.end)
          return a.end < b.end;
     if (a.a != b.a)
          return a.a < b.a;
     return a.b < b.b;
}

double getk(point &a, point &b) {
     return (b.y - a.y) * 1.0 / (b.x - a.x);
}
int main() {
     while (~scanf("%d", &n)) {
          ncount = 0;
          for (int i = 0; i < n; i++)
               scanf("%d%d", &po[i].x,
&po[i].y);
          for (int i = 0; i < n - 1; i++)
               for (int j = i + 1; j < n; j++)
{
                    line[ncount].a = i;
                    line[ncount].b = j;
                    if (po[i].x == po[j].x)
                         line[ncount].end =
true;
                    else {
                         line[ncount].end =
false;
                         line[ncount].k =
getk(po[i], po[j]);
                    }
                    ncount++;
               }
          sort(line, line + ncount);
          int start = 0;
          int ans = 0;
          for (int i = 1; i < ncount; i++) {
               if (!((line[i].end &&
line[start].end) || ((!line[i].end
&& !line[start].end) && (line[i].k -
line[start].k) < eps)) || line[i].a !=
line[start].a) {
                    start = i;
                    continue;
               }
               if (i - start > ans) {
                    ans = i - start;
               }
          }
          printf("%d\n", ans + 2);
     }
     return 0;
}
```

/**************************************/

N 个点最多组成正方形个数

/**************************************/

```
int n;
struct HASH {
     int cnt;
     int next;
} hash[50000];
int hashl;
int Hash(int n) {
     int i = n % PRIME;
     while(hash[i].next != -1) {
          if(hash[hash[i].next].cnt == n) return
1;
          else if(hash[hash[i].next].cnt > n)
break;
          i = hash[i].next;
     }
     hash[hashl].cnt = n;
     hash[hashl].next = hash[i].next;
     hash[i].next = hashl;
     hashl++;
     return 0;
}
int Hash2(int n) {
     int i = n % PRIME;
     while(hash[i].next != -1) {
          if(hash[hash[i].next].cnt == n) return
1;
          else if(hash[hash[i].next].cnt > n)
return 0;
          i = hash[i].next;
     }
     return 0;
}
int check(double ax, double ay, int &x, int &y)
{
     int a0 = (int)ax;
     int b0 = (int)ay;
     int tag1 = 0, tag2 = 0;
     if(fabs(a0 - ax) < eps) {
          tag1 = 1;
          x = a0;
     } else if(fabs(a0 + 1 - ax) < eps) {
          tag1 = 1;
          x = a0 + 1;
     }
     if(fabs(b0 - ay) < eps) {
          tag2 = 1;
          y = b0;
     } else if(fabs(b0 + 1 - ay) < eps) {
          y = b0 + 1;
          tag2 = 1;
     }
     if(tag1 == 1 && tag2 == 1) return 1;
     else return 0;
}
int squares(point p1, point p2, point &p3, point
&p4) {
     double a = (double)p2.x - p1.x;
     double b = (double)p2.y - p1.y;
     double midx = ((double)p1.x + p2.x) / 2;
     double midy = ((double)p1.y + p2.y) / 2;
     double tmp = a * a + b * b;
     double x1 = sqrt(b * b) / 2;
     double y1;
     if(fabs(b) < eps) y1 = sqrt(a * a + b *
b) / 2;
     else y1 = -a * x1 / b;
     x1 += midx;
     y1 += midy;
     if(check(x1, y1, p3.x, p3.y) == 0) return
0;
     x1 = 2 * midx - x1;
     y1 = 2 * midy - y1;
     if(check(x1, y1, p4.x, p4.y) == 0) return
0;
     return 1;
}
int main() {
     int i, j, cnt;
     while(scanf("%d", &n) != EOF && n) {
          for(i = 0; i < PRIME; i++) hash[i].next
= -1;
          hashl = PRIME;
          int x1, y1, x2, y2;
          for (i = 0; i < n; i++) {
               scanf("%d%d", &p[i].x, &p[i].y);
               Hash((p[i].x + 100000) * 100000
+ p[i].y + 100000);
          }
          cnt = 0;
          for (i = 0; i < n; i++) {
               for (j = i + 1; j < n; j++) {
                    point a, b;
                    if(squares(p[i], p[j], a, b)
== 0) continue;
                    if(Hash2((a.x + 100000) *
100000 + a.y + 100000) == 0) continue;
```

```
                    if(Hash2((b.x + 100000) *
100000 + b.y + 100000) == 0) continue;
                    cnt++;
               }
          }
          printf("%d\n", cnt / 2);
     }
     return 0;
}
```

/**************************************/

N 个点最多确定互不平行的直线

/**************************************/

```
double FindSlewRate(point p1, point p2) {
     point p;
     p.x = p2.x - p1.x;
     p.y = p2.y - p1.y;
     return atan2(p.y , p.x);
}
int main() {
     int n, rt;
     point p[205];
     double rate[40005];
     while(scanf("%d", &n) != EOF) {
          for(int i = 0; i < n; i++)
               scanf("%lf%lf",
&p[i].x ,&p[i].y);
          rt = 0;
          for(int i = 0; i < n; i++)
               for(int j = i + 1; j < n; j++)
                    rate[rt++] =
FindSlewRate(p[i], p[j]);
          sort(rate,rate+rt);
          int ans = 1;
          for(int i = 1; i < rt; i++)
               if(rate[i] > rate[i - 1]) ans++;
          printf("%d\n", ans);
     }
     return 0;
}
```

/**************************************/

求多边形的核

/**************************************/

```
void cut(line cur,polygon &pg) { //cur逆时针方
向为可行域
     polygon tp;
     int n=pg.n;
     pg.p[n]=pg.p[0];
     tp.n=0;
     for(int i=0; i<n; i++) {
          point p=pg.p[i],q=pg.p[i+1];
          double
pp=xmult(cur.a,cur.b,p),qq=xmult(cur.a,cur.b,q
);
          if(pp>-eps)
               tp.p[tp.n++]=p;
          if(pp*qq<-eps)

tp.p[tp.n++]=line_intersection(cur,line(p,q));
     }
     pg=tp;
}
void polygon_core(polygon pg,polygon &cg) {
     if(!is_unclock(pg))
          to_unclock(pg);
     cg=pg;
     int n=pg.n;
     pg.p[n]=pg.p[0];
     for(int i=0; i<n; i++) {
          cut(line(pg.p[i],pg.p[i+1]),cg);
          if(cg.n==0)
               return;
     }
}
```

/**********/

数学

/**********/

/**************************************/

数论

/**************************************/

1. $(a^n,b^n)=(a,b)^n$
2. $(an,bn)=n(a,b)$
3. $ad\equiv bd(\bmod md)$
   $\Rightarrow a\equiv b(\bmod m)$
4. $x\equiv a(\bmod m_1)$
   $x\equiv a(\bmod m_2)$
   的全部解为$x\equiv a(\bmod\{m_1,m_2\})$
5. $\{n_1,n_2.....n_k\}=\{m_1,m_2.....m_k\}$
   且$\forall i\to n_i\mid m_i$
   则同余式组$x\equiv b_i(\bmod n_i)$与
   $x\equiv b_i(\bmod m_i)$
   有相同解$x\equiv x_0(\bmod\{n_1,n_2.....n_k\})$
6. $a\mid b,a\mid c\Rightarrow a\mid(b,c)$
7. $a\equiv b(\bmod m)\Rightarrow(a,b)=(b,m)$
8. p是质数$\Rightarrow\varphi(p^l)=p^{l-1}(p-1)$
9. $a=\prod_{i=1}^{k}p_i^{\alpha_i}\Rightarrow\varphi(a)=\prod_{i=1}^{k}p_i^{\alpha_i-1}(p_i-1)$
10. p是质数，则$\sum_{i=0}^{n}\varphi(p^i)=p^n$
11. $n>2\Rightarrow\varphi(n)\equiv 0(\bmod 2)$
    $(a,b)=1\Rightarrow\varphi(a,b)=\varphi(a)\varphi(b)$
12. $n>1$时，不大于n且与n互质的所有正整数之和为$\frac{1}{2}\varphi(n)\,n$
13. p是质数，则$(a+b)^p\equiv a^p+b^p(\bmod p)$
14. $(a,b)=1$且存在最小的h使得$a^h\equiv 1(\bmod b)$
    $\Rightarrow h\mid\varphi(b)$
15. n是质数$\Leftrightarrow\varphi(n)\mid(n-1),(n+1)\mid\sigma(n)$
    $\sigma(n)=\sum_{d\mid n}d$

16. $$\lfloor x\rfloor+\lfloor y\rfloor\le\lfloor x+y\rfloor$$
$$\{x\}+\{y\}\ge\{x+y\}$$

17. $$\lfloor a\rfloor+\left\lfloor a+\frac{1}{n}\right\rfloor+.....+\left\lfloor a+\frac{n-1}{n}\right\rfloor=\lfloor na\rfloor$$

18. $$\lfloor 2x\rfloor+\lfloor 2y\rfloor\ge\lfloor x\rfloor+\lfloor y\rfloor+\lfloor x+y\rfloor$$

19. $$\sum_{k=1}^{n}\left\lfloor\frac{k}{2}\right\rfloor=\left\lfloor\frac{n^2}{4}\right\rfloor$$

20. $$\sum_{k=1}^{n}\left\lfloor\frac{k}{3}\right\rfloor=\left\lfloor\frac{n(n-1)}{6}\right\rfloor$$

21. $$\sum_{t|n}t=n^{\frac{d(n)}{2}}$$
$d(n)$为n的因子个数

22. $$n=\sum_{d|n}\varphi(d)$$

23. $(p,q)=1$且p、q为奇正整数
$$\sum_{0<l<\frac{q}{2}}\left\lfloor\frac{p}{q}l\right\rfloor+\sum_{0<k<\frac{p}{2}}\left\lfloor\frac{q}{p}k\right\rfloor$$

24. 不定方程：$ax+by=c$
其中$c>=lcm(a,b)$，那么
如果方程有解，则
方程必定有非负整数解

/**********定理与定义**********/

费马定理

如果 a,p 互质，则有 $a^{p-1}\equiv 1\pmod p$

威尔森定理：

p是质数，则$(p-1)!\equiv -1\pmod p$

欧拉定理

1. 如果 a,p 互质，则有 $a^{\phi(p)}\equiv 1\pmod p$，如果不互质，那么不存在 n 使得 $a^n\equiv 1\pmod p$

2. $x^n\equiv x^{n\%\varphi(m)+\varphi(m)}\pmod m\ (x\ge\varphi(m))$

原根

设 p 是正整数，a 是整数，若 a 模 p 的阶等于φ(p)，则称 a 为模 p 的一个原根。

如果 p 为质数，且 a是 p 的原根，则 a^i mod p 两两不同(0<i<p,1<a<p)，p 的原根个数等于 φ（p-1）

求分数的最小公倍数

1、 约分。
2、 求分子的最小公倍数和分母的最大公约数。
3、 最小公倍数除以最大公约数即为所求。

鸽巢原理的运用

1、 给定 m 个整数 $a_1, a_2, a_3, \cdots a_m$，存在整数 k 和 l，$0\le k<l\le m$，使得 $a_{k+1}+a_{k+2}+\cdots+a_l$ 能被 m 整除。
2、 令 m 与 n 为互素的正整数，并令 a 和 b 为两整数，且 $0\le a\le m-1$ 以及 $0\le b\le n-1$。于是存在一个正整数 x，使得 x 除以 m 的余数为 a，且除以 n 的余数为 b。
3、 每个由 n^2+1 个实数构成的序列 $a_1, a_2, \cdots, a_{n^2+1}$ 或者含有长度为 n+1 的递增子序列，或者含有长度为 n+1 的递减子序列。

Burning side定理

设 G 是 X 的一个置换群，C 是 X 的一个着色集并且使得对于 G 中的任意 f 与 C 中的任意 c，f*c 属于 C，则 C 中不等价的着色数 N（G,C）为

$$N(G,C)=\frac{1}{|G|}\sum_{f\in G}|C(f)|$$

换言之，C 中不等价的着色数等于使着色通过 G 中的置换保持不变的着色的平均数。

Polya 定理

若可用 m 种颜色，对于一个置换 a_j，设它的循环数为 $c(a_j)$，则在此置换下不变的着色方案数为 $m^{c(a_j)}$

$$L=\frac{1}{|G|}(m^{c(g_1)}+m^{c(g_2)}+\cdots+m^{c(g_s)})$$

旋转：

若转过 k 个珠子，则循环节数（c(ak)）为 gcd(k,n)。

（循环节数为 r 的置换数为 $\phi(n/r)$）

翻转：

1、若总数为奇，则共有循环节数为(n+1)/2 的置换 n 个
2、若总数为偶，则有循环节数为 n/2 的置换 n/2 个，循环节数为 n/2+1 的置换 n/2 个

乘法逆元

当我们要求（a/b） mod p 的值，且 a 很大，无法直接求得 a/b 的值时，我们就要用到乘法逆元。

我们可以通过求 b 关于 p 的乘法逆元 k，将 a 乘上 k 再模 p，即(a*k) mod p。其结果与(a/b) mod p 等价。

格雷码：

相邻两位数二进制位只相差 1 位

十进制数	自然二进制数	格雷码
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001

15	1111	1000

二进制转格雷码：从最右边一位起，依次将每一位与左边一位异或（XOR），作为对应格雷码该位的值，最左边一位不变（相当于左边是0）；
格雷码转二进制：从左边第二位起，将每位与左边一位解码后的值异或，作为该位解码后的值（最左边一位依然不变）。

积性函数

在数论中的积性函数：对于正整数 n 的一个函数 f(n)，当中 f(1)=1 且当 a,b 互质，f(ab)=f(a)f(b)，在数论上就称它为积性函数。若某函数 f(n) 符合 f(1)=1，且就算 a,b 不互质，f(ab)=f(a)f(b)，则称它为完全积性函数。

例如：φ(n) －欧拉函数，计算与 n 互质的正整数之数目
μ(n) －莫比乌斯函数，关于非平方数的质因子数目
gcd(n,k) －最大公因子，当k 固定的情况
d(n) －n 的正因子数目
σ(n) －n 的所有正因子之和

偏序集：

链：
链上任意两个元素都可以比较

反链：
反链上任意两个元素都不可以比较

Dilworth 定理：
令 $(X,\leq)$ 是一个有限偏序集，并令 r 是链的最大的大小，则 X 可以被划分成 r 个但不能再少的反链
同理，令 m 是反链的最大的大小，则 X 可以被划分成 m 个但不能再少的链

生成排列

1、 生成全排列
由一个排列 p[1]p[2]...p[n]生成下一个排列的算法：
a) i=max{j|p[j-1]<p[j]},j=2,3,...,n
b) j=max{k|p[i-1]<p[k]},k=1,2,...,n
c) p[i-1]与 p[j]互换
d) p[i],p[i+1],...,p[n]逆转，即变成 p[n],p[n-1],...,p[i],其它的不变

2、 生成字典序 r-组合
字典序 r-组合即从{1，2…n}中选出 r 个元素，使得只要 i<j，则 ai<aj。
已知一个字典序 r-组合，它的下一个字典序 r-组合求法：
A） 确定最大的整数 k 使得 ak+1<=n 且 ak+1 不是 a1,a2..ar 中的元素。
B） 下一序列为 a1…a[k-1],ak+1,ak+2..ak+r-k+1;

/**************************************/

公式与序列

/**************************************/

N 条直线最多将平面分割成的区域数：Ln=1+Sn//Sn 为首项为1，公差为 1 的等差数列前 n 项和

汉诺塔移动步数=2^n-1：

递归式：
公式一：T(1)=1，T(n)=T(n-1)+n，则 T(n)为 n(n+1)/2
公式二：T(1)=1，T(n)=T(n/2)+1，则 T(n)约为 lgN
公式三：T(1)=0，T(n)=T(n/2)+n，则 T(n)约为 2n
公式四：T(1)=0，T(n)=2T(n/2)+n，则 T(n)约为 nlgN
公式五：T(1)=1，T(n)=2T(n/2)+1，则 T(n)约为 2n

欧拉常数 γ = 0.57721566490153286060651209

调和级数 S=1+1/2+1/3+……Sn=ln(n)+ γ

斐波那契数列：(1/√5)*{[(1+√5)/2]^n - [(1-√5)/2]^n}

卡特兰数：h(n)=h(n-1)*(4*n-2)/(n+1)=C(2n,n)/(n+1) (n=1,2,3,...)
应用
{
1) 将 n + 2 边形沿弦切割成 n 个三角形的不同切割数
2) n + 1 个数相乘，给每两个元素加上括号的不同方法数
3) n 个节点的不同形状的二叉树数
4) 从 n * n 方格的左上角移动到右下角不穿过对角线路径数
}

斯特林公式：

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

贝叶斯公式

$$P(A|B)=\frac{P(B|A)*P(A)}{P(B)}$$

贝尔数： B（n+1）=Sum（0,n）*C（n，k）*B（k）;
应用
{
1）n 元集合的划分种数。
}

法雷级数：分母小于 n 的真分数的个数在加上 0,1
f[0]=0;f[1]=3;
f[n]=f[n-1]+phi[n]*2;

第二类斯特林数： S(n,n) = 1， S(n,0) = 0；
S(n,k) = S(n-1,k-1) + kS(n-1,k)
应用
{
n 元集合划分为 k 个子集（非空）
n 个有标号的小球放入 k 个无标号的盒子中
}
第二类斯特林数的奇偶性：

$$\left\{ {n \atop k} \right\} \equiv \binom{z}{w} \pmod 2,\ z = n - \left\lceil \frac{k+1}{2} \right\rceil,\ w = \left\lfloor \frac{k-1}{2} \right\rfloor$$

杨式图表：
杨式图表是一个矩阵，它满足条件：
如果格子[i, j]没有元素，则[i+1, j]也一定没有元素
如果格子[i, j]有元素 a[i, j]，则[i+1, j]要么没有元素，要么 a[i+1, j] > a[i, j]
Y[n]代表 n 个数所组成的杨式图表的个数
公式：
Y[1]=1;Y[2]=2;
Y[n]=Y[n-1]+(n-1)*Y[n-2];(n>2)

整数划分：
将整数 n 分成 k 份，且每份不能为空，任意两种分法不能相同
1) 不考虑顺序
```
for(int p=1; p<=n ;p++)
   for(int i=p; i<=n ;i++)
      for(int j=k; j>=1 ;j--)
         dp[i][j] += dp[i-p][j-1];
cout<< dp[n][k] <<endl;
```
2) 考虑顺序
dp[i][j] += dp[i-k][j-1]; (k=1..i)
3) 若分解出来的每个数均有一个上限 m
dp[i][j] += dp[i-k][j-1]; (k=1..m)

错位排列：
d[1]=0,d[2]=1;
d[n]=(n-1)*(d[n-1]+d[n-2]);

n 的约数的个数

若 n 满足 $n=\prod_{i=1}^{m} a_i^{n_i}$，则 n 的约数的个数为 $\prod_{i=1}^{m}(n_i+1)$

约数的个数：∏(1+pi)
约数的和：Σ(ai^(pi+1)-1)/(ai-1)
求和公式：

$$\sum k^2 = \frac{n(n+1)(2n+1)}{6}$$

$$\sum k^3 = \frac{n^2(n+1)^2}{4}$$

$$\sum (2k-1)^2 = \frac{n(4n^2-1)}{3}$$

$$\sum (2k-1)^3 = n^2(2n^2-1)$$

$$\sum k^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}$$

$$\sum k^5 = \frac{n^2(n+1)^2(2n^2+2n-1)}{12}$$

$$\sum k(k+1) = \frac{n(n+1)(n+2)}{3}$$

$$\sum k(k+1)(k+2) = \frac{n(n+1)(n+2)(n+3)}{4}$$

$$\sum k(k+1)(k+2)(k+3) = \frac{n(n+1)(n+2)(n+3)(n+4)}{5}$$

/**************************************/

排列与组合

/**************************************/

1、 **错位排列**

$$n!\left(1-\frac{1}{1!}+\frac{1}{2!}-\frac{1}{3!}+\cdots\pm\frac{1}{n!}\right)$$

f(n) = (n-1)*(f(n-1)+f(n-2))

2、 **循环排列**

a) n 个不同元素集合的循环 r 排列 $\frac{P(n,r)}{r}$

b) S 是重复数为 $n_1,n_2....n_k$ 的多重集，$n_1=1$，令 $n=\sum_{i=2}^{k}n_i$，则 S 的循环排列数 $\frac{n!}{n_2!n_3!....n_k!}$

3、 **多重集的排列（元素重数有限）**

S 是一个含 k 个元素的多重集，每个元素重数为 n1，n2...nk，则 S 的排列数为 $\frac{n!}{n_1!n_2!.....n_k!}$

4、 **集合划分**

A） n 个元素的集合划分为 k 个有标号子集且子集 i 含有 ni 个元素：

$$\frac{n!}{n_1!n_2!.....n_k!}$$

B） n 个元素的集合划分为 k 个无标号子集且第 i 个子集含有 ni 个元素：

$$\frac{n!}{k!n_1!n_2!.....n_k!}$$

5、 **多重集的组合（元素重数无限）**

S 是一个含 k 个元素的多重集，则 S 的 r 组合数为

$$C(r+k-1,r)=C(r+k-1,k-1)$$

PS: 当元素个数有限时，可以当做无限来做，再利用容斥原理减去不符合要求的即可

6、 $$C(2n,n)=\sum_{k=0}^{n}C(n,k)^2$$

$$\sum_{0\le k\le m}C(n,k)=C(n+1,m+1)$$

7、 $$C(n,k)=C(0,k-1)+C(1,k-1)+..C(n-1,k-1)$$

8、 $$\sum_{a\le k\le n-b}C(k,a)*C(n-k,b)=C(n+1,a+$$

9、

序号	n 个球	r 个盒子	允许空盒	方案数
1	不同	不同	允许	r^n
2	不同	不同	不允许	r!S(n,r)
3	不同	相同	允许	Sum(S(n,k)),k=1..r
4	不同	相同	不允许	S（n,r）
5	相同	不同	允许	C（n+r-1,n）
6	相同	不同	不允许	C（n-1,r-1）
7	相同	相同	允许	Sum(Pk(n)),k=1..r
8	相同	相同	不允许	Pr（n）

Pr(n)代表将 n 非空划分成 r 份的种数

10、 **多项式系数** $C(\frac{n}{n_1,n_2...n_t})=\frac{n!}{n_1!n_2!...n_t!}$

（x1+x2+…xt）^n 展开式系数，n1+n2+..nt=n

/**************************************/

数学常用库

```
/**************************************/
//筛素数
bool  notp[mr];
int  pr[N],pn;
void  getpri()  {
      pn=0;
      memset(notp,0,sizeof(notp));
      for(int  i=2;  i<mr;  i++)  {
            if(!notp[i])  {
                  pr[pn++]=i;
            }
            for(int j=0; j<pn && i*pr[j]<mr; j++)
{
                  int  k=i*pr[j];
                  notp[k]=1;
                  if(i%pr[j]==0)  {
                        break;
                  }
            }
      }
}
//筛欧拉函数
int  phi[mr];
void  getphi()  {
      pn=0;
      phi[1]=1;
      memset(notp,0,sizeof(notp));
      for(int  i=2;  i<mr;  i++)  {
            if(!notp[i])  {
                  pr[pn++]=i;
                  phi[i]=i-1;
            }
            for(int j=0; j<pn && i*pr[j]<mr; j++)
{
                  int  k=i*pr[j];
                  notp[k]=1;
                  if(i%pr[j]==0)  {
                        phi[k]=phi[i]*pr[j];
                        break;
                  } else  {
                        phi[k]=phi[i]*(pr[j]-1);
                  }
            }
      }
}
//筛约数个数、最小素数次数
int  divnum[mr];
char  e[mr];
void  getdivnum()  {
      pn=0;
      memset(notp,0,sizeof(notp));
      for(int  i=2;  i<mr;  i++)  {
            if(!notp[i])  {
                  pr[pn++]=i;
                  e[i]=1;
                  divnum[i]=2;
            }
            for(int j=0; j<pn && i*pr[j]<mr; j++)
{
                  int  k=i*pr[j];
                  notp[k]=1;
                  if(i%pr[j]==0)  {

divnum[k]=divnum[i]/(e[i]+1)*(e[i]+2);
                        e[k]=e[i]+1;
                        break;
                  } else  {

divnum[k]=divnum[i]*divnum[pr[j]];
                        e[k]=1;
                  }
            }
      }
}
//筛莫比乌斯函数
int  miu[mr];
void  getmiu()  {
      pn=0;
      miu[1]=1;
      memset(notp,0,sizeof(notp));
      for(int  i=2;  i<mr;  i++)  {
            if(!notp[i])  {
                  pr[pn++]=i;
                  miu[i]=-1;
            }
```

```
            for(int j=0; j<pn && i*pr[j]<mr; j++)
{
                int k=i*pr[j];
                miu[k]=(i%pr[j])?-miu[i]:0;
                notp[k]=1;
                if(i%pr[j]==0) {
                    break;
                }
            }
        }
}
//欧几里得
inline int gcd(int a,int b) {
    int tp;
    while(b) {
        tp=a;
        a=b;
        b=tp%b;
    }
    return a;
}
//扩展欧几里得
int Egcd (int a, int b, int &x, int &y){
    if (b==0){
        x=1,y=0;
        return a;
    }
    int d, tp;
    d = Egcd (b, a%b, x, y);
    tp = x;
    x = y;
    y = tp - a/b*y;
    return d;
}
//乘法逆元，【确保有逆元】
int inv(int a) {
    if(a == 1) return 1;
    return mul((MOD - MOD/ a),inv(MOD % a));
}
//最小公倍数
inline int lcm(int a,int b) {
    return a/gcd(a,b)*b;
}
//数位统计,[1,con]区间内数字出现次数
void cac(ll con,ll cnt[],ll t) {
    //如果从0开始，则在一开始便需要加上特判，
if(con==0)cnt[0]++;
    if(con<=0)return;
    ll x,y,n=con/10;
    ll i,j;
    x=con/10,y=con%10;
    for(i=0; i<=y; i++) cnt[i]=cnt[i]+t;
    for(; x!=0; x/=10) cnt[x%10]+=(y+1)*t;
    for(i=0; i<10; i++) cnt[i]+=n*t;
    cnt[0]-=t;
    cac(n-1,cnt,t*10);
}
//防高精度乘法
ll mulmod(ll a, ll b, ll mod){
    a = a%mod, b = b%mod;
    ll re = 0;
    while(b){
        if (b&1) re = (re + a)%mod;
        a = (a << 1)%mod;
        b >>= 1;
    }
    return re;
}
//快速幂取模
ll fastmod(ll a,int n,int m){
    ll re=1;
    for(;n;n>>=1,a=mulmod(a,a,m)){
        if(n&1)re=mulmod(re,a,m);
    }
    return re;
}
//米勒拉宾伪素数测试
bool miller_rabin(int n,int s){
    if(n==2)return true;
    else if(n%2==0)return false;
    for(int i = 1; i <= s; ++i){
        int a = rand()%(n-2)+2;
        if(fastmod(a, n-1, n)!=1)return false;
    }
    return true;
}
```

/**************************************/

整数拆分

/**************************************/

```
void Prepare() {
    LL i , j , k , s , o ;
    P[0] = 1 ;
    for ( i = 1 ; i <= MAXN ; i ++ ) {
        j = 1 , k = 1 , s = 0 , o = -1 ;
        for ( ; j > 0 ; k ++ , o *= -1 )
{
            j = i - (3*k*k+k)/2 ;
            if ( j >= 0 ) s -= o * P[j] ;
            j = i - (3*k*k-k)/2 ;
            if ( j >= 0 ) s -= o * P[j] ;
            s = ((s % MOD)+MOD)%MOD ;
        }
        P[i] = s ;
    }
}
```

/**************************************/

连分数

/**************************************/

1、

$$连分数[a_1,a_2.....a_k]=\frac{p_k}{q_k}$$

$$p_1=a_1,q_1=1$$

$$p_2=a_1*a_2+1,q_2=a_2$$

$$\Rightarrow$$

$$p_k=a_k*p_{k-1}+p_{k-2}$$

$$q_k=a_k*q_{k-1}+q_{k-2}$$

2、

$$对于连分数[a_1,a_2.....a_k],$$

$$k\geq 2\Rightarrow p_k*q_{k-1}-p_{k-1}*q_k=(-1)^k$$

$$k\geq 3\Rightarrow p_k*q_{k-2}-p_{k-2}*q_k=(-1)^{k-1}a_k$$

3、

$$\frac{p_1}{q_1}<\frac{p_3}{q_3}<\frac{p_5}{q_5}<.....[a_1,a_2.....]$$

$$<.....\frac{p_6}{q_6}<\frac{p_4}{q_4}<\frac{p_2}{q_2}$$

/**************************************/

伯努利数

/**************************************/

1、

伯努利数：

$$B_n=\sum_{j=0}^{n}\frac{(-1)^k}{k+1}\sum_{j=0}^{k}(-1)^{k-j}C(k,j)j^n$$

伯努利多项式：

$$B_n(x)=\sum_{k=0}^{n}C(n,k)B_k x^{n-k}$$

$n\geq 1$时

$$B_n(x+1)-B_n(x)=nx^{n-1}$$

$$\sum_{k=1}^{m}k^n=\frac{1}{n}(B_{n+1}(m+1)-B_{n+1})$$

2、

$k\geq 1$时

$$B_{2k+1}=0$$

$$B_0=1,B_1=-\frac{1}{2},B_2=\frac{1}{6},B_4=-\frac{1}{30}$$

$$B_6=\frac{1}{42},B_8=-\frac{1}{30},B_{10}=\frac{5}{66}$$

$$B_{12}=-\frac{691}{2730},B_{14}=\frac{7}{6},B_{16}=-\frac{3617}{510}$$

$$B_{18}=\frac{43867}{798},B_{20}=-\frac{174611}{330}$$

/**************************************/

伯努利公式求幂方和

/**************************************/

```
struct data{
	int a,b;
	data(){}
	data(int _a,int _b){a=_a,b=_b;}
	void updata(){
		int d=gcd(abs(a),abs(b));
		a/=d,b/=d;
	}
}B[30];
int C[30][30];
void Init(){
	B[0]=data(1,1);
	B[1]=data(-1,2);
	B[2]=data(1,6);
	B[4]=data(-1,30);
	B[6]=data(1,42);
	B[8]=data(-1,30);
	B[10]=data(5,66);
	B[12]=data(-691,2730);
	B[14]=data(7,6);
	B[16]=data(-3617,510);
	B[18]=data(43867,798);
	B[20]=data(-174611,330);
	for(int i=3;i<=21;i+=2)B[i]=data(0,1);
	C[0][0]=1;
	for(int i=1;i<=21;i++){
		C[i][0]=C[i][i]=1;
		for(int j=1;j<i;j++){
			C[i][j]=C[i-1][j]+C[i-1][j-1];
		}
	}
}
data ans[30];
data mult(int cc,data x){
	int d=gcd(cc,x.b);
	cc/=d,x.b/=d;
	return data(cc*x.a,x.b);
}
data sub(data x,data y){
	data z(x.a*y.b-x.b*y.a,x.b*y.b);
	z.updata();
	return z;
}
data add(data x,int t){
	data z(x.a+x.b*t,x.b);
	z.updata();
	return z;
}
int main(){
	int n;
	Init();
	while(scanf("%d",&n)!=EOF){
		for(int i=0;i<=n;i++){
			ans[i]=mult(C[n+1][i],B[i]);
		}
		ans[n+1]=data(0,1);
		int m=1;
		for(int i=0;i<=n;i++){
			m=lcm(m,ans[i].b);
		}
		ans[1]=add(ans[1],(n+1));
		printf("%d",m*(n+1));
		for(int i=0;i<=n+1;i++){
			ans[i].a=m/ans[i].b*ans[i].a;
			printf(" %d",ans[i].a);
		}
		printf("\n");
	}
	return 0;
}
```

/**************************************/

差分序列

/**************************************/

1、

令$h_0,h_1...h_n...$为一数列

定义$\Delta h_0,\Delta h_1...\Delta h_n$为新的序列并满足

$\Delta h_n=h_{n+1}-h_n$为原数列的一阶差分序列

同理，满足$\Delta^p h_n=\Delta^{p-1}h_{n+1}-\Delta^{p-1}h_n$为序列的p阶差分序列

2、

$$h_n=a_p n^p+a_{p-1}n^{p-1}+...a_0$$

则对所有$n\geq 0,\Delta^{p+1}h_n=0$

3、

差分序列的第0条对角线等于

$c_0,c_1...c_p,0,0,0...$的序列的一般项满足

$$h_n=c_0\binom{n}{0}+c_1\binom{n}{1}+..c_p\binom{n}{p}$$

4、

序列$h_0,h_1...h_n...$有一差分表，
第0条对角线
等于$c_0,c_1,c_2.....c_p,0,0...$

则$\sum_{k=0}^{n} h_k =$

$$c_0\binom{n+1}{1}+c_1\binom{n+1}{2}+..c_p\binom{n+1}{p+1}$$

/**************************************/

差分序列求幂方和

/**************************************/

```
import java.util.*;
import java.io.*;
import java.math.BigInteger;
public class Main
{
    static BigInteger c[]=new BigInteger[105];
    static BigInteger h[][]=new
BigInteger[105][105];
    static BigInteger cal[]=new
BigInteger[105];
    static void Init(BigInteger n,int p){
        cal[0]=BigInteger.ONE;
        for(int i=1;i<=p;i++){

cal[i]=cal[i-1].multiply(n).divide(BigInteg
er.valueOf(i));
            n=n.subtract(BigInteger.ONE);
        }
    }
    public static void main(String[] args)
    {
        Scanner cin = new Scanner(System.in);
        int T=cin.nextInt();
        while(T>0){
            T--;
            BigInteger
n=cin.nextBigInteger(),ans=BigInteger.ZERO;
            int m=cin.nextInt();
            Init(n.add(BigInteger.ONE),m+1);
            for(int i=0;i<=m;i++){

h[0][i]=BigInteger.valueOf(i).pow(m);
            }
            c[0]=h[0][0];
            for(int i=1;i<=m;i++){
                for(int j=0;j<=m-i;j++){

h[i][j]=h[i-1][j+1].subtract(h[i-1][j]);
                }
                c[i]=h[i][0];
            }
            for(int i=1;i<=m+1;i++){

ans=ans.add(c[i-1].multiply(cal[i]));
            }
            System.out.println(ans);
        }
    }

}
```

/**************************************/

定积分（龙贝格）

/**************************************/

```
double Function(double x) {
    return pow(x,3.0/2.0);
}
double Romberg(double a,double b) {
    double h,T[MAXN][MAXN],fz,t;
    int k,l,m,n,i,j;
    h=b-a;
    T[0][0]=h*(Function(b)+Function(a))/2;
    for(l=1; l<100; l++) {
        n=1<<(l-1);
        h=h/2;
        fz=0;
        for(k=0; k<n; k++) {
            t=a+(2*k+1)*h;
            fz=fz+Function(t);
        }
        T[0][l]=(T[0][l-1])/2+h*fz;
        for(m=1; m<=l; m++) {

T[m][l-m]=(pow(4,m)*T[m-1][l-m+1]-T[m-1][l-m])
/(pow(4,m)-1);
        }
        if(fabs(T[l][0]-T[l-1][0])<=eps)
            break;
    }
    return T[l][0];
}
```

/**************************************/

定积分（自适应辛普森）

/**************************************/

```
double function (double x) {
    return x;
}
double simpson (double l , double r ) {
    return (function (l) + 4 * function ((l +
r) / 2.0 ) + function (r )) * (r - l) / 6.0;
}
double simpson (double l , double r , double
all , double eps) {
    double m = (l + r) / 2.0;
    double L = simpson (l , m) , R = simpson
(m , r);
    if (fabs (L + R - all) <= 15 * eps) return
L + R + (L + R - all) / 15;
    return simpson (l , m , L , eps / 2.0)
+ simpson (m , r , R , eps / 2.0);
}
double simpson (double l , double r , double
eps) {
    return simpson (l , r , simpson (l , r) ,
eps);
}
```

/**************************************/

线性规划（单纯型）

/**************************************/

```
#include<cstring>
#include<cstdio>
#include<algorithm>
using namespace std;
const int N=305;
const double INF=1e50;
class DanChun {
    /*
    下标从1开始
    sum(A[i,j]*X[j])<=A[i][Num_X+Num_C+1];
    X[i]>=0
    max(min) sum(C[i]X[i])
    */
public:
    double      C[N];//目标函数
    double      A[N][N];//系数矩阵，最后一列是常数
项
    double      TestNum[N];//检验数，第一个数是最
大（或最小）检验数的坐标
    double      TheTa[N];//theta
    double      CB[N];//
    int         XB[N];//
    double      X[N];//解
    int         Num_X;//变量个数
    int         Num_C;//方程（约束条件）个数
    int      MaxMin;//求的是最大值还是最小值?
0:max; 1:min
    void Init(){
        memset(CB,0,sizeof(CB));
    }
    void Read() {
        scanf("%d%d",&Num_X,&Num_C);
        scanf("%d",&MaxMin);
        int BL=Num_X+Num_C;
```

```
          for(int k=1; k<=Num_X;
k++)scanf("%lf",&C[k]);
          for(int i=1; i<=Num_C; i++) {
               char op;
               for(int j=1; j<=Num_X; j++)
scanf("%lf",&A[i][j]);
               scanf(" %c",&op);
               A[i][0]=(op=='<')?0:1;
               A[i][Num_X+i]=1;
               scanf("%lf",&A[i][BL+1]);
               XB[i]=Num_X+i;
          }
     }
     void GetTestNum() {
          int h=-1;//记录最大检验数的坐标
          for(int i=1; i<=Num_X+Num_C; i++) {
               double temp=0;
               for(int j=1; j<=Num_C; j++)
temp+=(CB[j]*A[j][i]);
               double ta=C[i]-temp;
               TestNum[i]=ta;
               if(h==-1||TestNum[h]<ta)h=i;
          }
          TestNum[0]=h;
     }
     bool Check() {
          if(MaxMin==0) {
               for(int i=1; i<=Num_X+Num_C; i++)
{
                    if(TestNum[i]>0)return true;
               }
          } else {
               for(int i=1; i<=Num_X+Num_C; i++)
{
                    if(TestNum[i]<0)return true;
               }
          }
          return false;
     }
     void Set_X() {
          TheTa[0]=INF;
          int kk=0;
          int j = TestNum[0] , i;//确定换位的坐
标
          for(int k=1; k<=Num_C; k++) {

               if(A[k][ (int)TestNum[0] ] != 0)
{
                    TheTa[k] =
A[k][(int)(Num_X+Num_C+1)] /
A[k][ (int)(TestNum[0]) ];
                    if((TheTa[0] > TheTa[k]) &&
TheTa[k] > 0) {
                         TheTa[0] = TheTa[k];
                         kk=k;
                    }
               }
          }
          i = kk;
          CB[i]=C[j];
          XB[i]=j;
          for(int x=1; x<=Num_C; x++) {
               if(x!=i) {
                    double chushu = (-A[x][j]) /
(A[i][j]);
                    for(int y=1;
y<=Num_C+Num_X+1; y++) {
                         A[x][y] +=
A[i][y]*chushu;
                    }
               }
               else if(x==i) {
                    double chushu = A[i][j];
                    for(int y=1;
y<=Num_C+Num_X+1; y++) {
                         A[x][y] /= chushu;
                    }
               }
          }
     }
     void Set_Ans(){
          for(int i=1; i<=Num_C;
i++)X[XB[i]]=A[i][Num_X+Num_C+1];
     }
     void Show_Ans(){
          Set_Ans();
          for(int i=1; i<=Num_X;
i++)printf("X[%d]=%.2lf\n",i,X[i]);
     }
     double Get_Val(){
          Set_Ans();
          double ans=0;
          for(int
i=1;i<=Num_X;i++)ans+=X[i]*C[i];
          return ans;
     }
}XXGH;
int main() {
     XXGH.Init();
     XXGH.Read();
     XXGH.GetTestNum();
     while(XXGH.Check()) {
          XXGH.Set_X();
          XXGH.GetTestNum();
     }
     XXGH.Show_Ans();
     printf("%.2lf\n",XXGH.Get_Val());
     return 0;
}
```

/**************************************/

多项式乘法（FFT）

/**************************************/

```
double cof1[MAX], cof2[MAX];
int n, k, permutation[MAX];
struct complex {
     double r, v;
     complex operator + (complex& obj) {
          complex temp;
          temp.r = r + obj.r;
          temp.v = v + obj.v;
          return temp;
     }
     complex operator - (complex& obj) {
          complex temp;
          temp.r = r - obj.r;
          temp.v= v - obj.v;
          return temp;
     }
     complex operator * ( complex& obj) {
          complex temp;
          temp.r = r*obj.r - v*obj.v;
          temp.v = r*obj.v + v*obj.r;
          return temp;
     }
} p1[MAX], p2[MAX], omiga[MAX], result1[MAX],
result2[MAX];
void caculate_permutation(int s, int interval,
int w, int next) {
     if(interval==n) {
          permutation[w] = s;
          return ;
     }
     caculate_permutation(s,interval*2, w,
next/2);
     caculate_permutation(s+interval,
interval*2, w+next, next/2);
}
void fft(complex transform[], complex p[]) {
     int i, j, l, num, m;
     complex temp1, temp2;
     for(i=0; i<n; i++)transform[i] =
p[ permutation[i] ] ;
     num = 1, m = n;
     for(i=1; i<=k; i++) {
          for(j=0; j<n; j+=num*2)
               for(l=0; l<num; l++) {
                    temp2 =
omiga[m*l]*transform[j+l+num];
                    temp1 = transform[j+l];
                    transform[j+l] = temp1 +
temp2;
                    transform[j+l+num] = temp1 -
temp2;
               }
          num*=2,m/=2;
     }
}
void polynomial_by(int n1,int n2) {
     int i;
     double angle;
     k = 0, n = 1;
     while(n<n1+n2-1)k++,n*=2;
```

```
    for(i=0; i<n1; i++)p1[i].r = cof1[i], p1[i].v = 0;
    while(i<n)p1[i].r = p1[i].v = 0, i++;
    for(i=0; i<n2; i++)p2[i].r = cof2[i], p2[i].v = 0;
    while(i<n)p2[i].r = p2[i].v = 0, i++;
    caculate_permutation(0,1,0,n/2);
    angle = Pi/n;
    for(i=0; i<n; i++)omiga[i].r = cos(angle*i), omiga[i].v = sin(angle*i);
    fft(result1,p1);
    fft(result2,p2);
    for(i=0; i<n; i++)result1[i]= result1[i]*result2[i];
    for(i=0; i<n; i++)omiga[i].v = -omiga[i].v;
    fft(result2, result1);
    for(i=0; i<n; i++)result2[i].r/=n;
    i = n -1;
    while(i&&fabs(result2[i].r)<EPS)i--;
    n = i+1;
    while(i>=0) cof1[i] = result2[i].r, i--;
}
```

/**************************************/

莫比乌斯反演(定义)

/**************************************/

定义：

有偏序集 $(G,\le)$，同时，存在函数：

$$\begin{cases} F:G\to R \\ H:G\to R \end{cases}$$

，R 为实数集

且满足：

$$H(K)=\sum_{L\le K}F(L)$$

则有：

$$F(K)=\sum_{L\le K}H(L)\mu(L,K)$$

其中，$\mu(L,K)$为莫比乌斯函数。

举例：

1、 对于集合上由包含关系导出的偏序集的莫比乌斯函数为：$(-1)^{|K|-|L|}$

2、 对于整数上由正处关系导出的偏序集的莫比乌斯函数为：

$$\mu(d,n)=\mu(n/d)=\begin{cases} 1 & \text{当} n/d=1\text{时} \\ (-1)^r & \text{当} n/d \text{是} r\text{个不同素数的乘积时} \\ 0 & \text{当} n/d \text{能被一个素数的平方整除时} \end{cases}$$

3、 对于线性有序集导出的偏序集的莫比乌斯函数为：

$$\mu(k,l)=\begin{cases} 1 & l=k \\ -1 & l=k+1 \\ 0 & \text{其他} \end{cases}$$

4、

$$g(d)=\sum_{k=1,2,3\ldots}f(d*k)$$

$$f(d)=\sum_{k=1,2,3\ldots}g(d*k)\mu(k)$$

$$\sum_{d\in P}f(d)=\sum_{n=1,2,3\ldots}g(n)\sum_{d|n\&\&d\in P}\mu(n/d)$$

P为题目对d的约束，

重定义$sum(\mu(n))$，可以在$O(n\log n)$筛出$sum(\mu(n))$

/**************************************/

莫比乌斯反演(例程)

/**************************************/

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<ctime>
using namespace std;
typedef long long ll;
const int inf=0x3f3f3f3f;
const int mr=1000005;
int smiu[mr],pr[mr],miu[mr],pn;
bool notp[mr];
void getmiu() {
    pn=0;
    miu[1]=1;
    memset(notp,0,sizeof(notp));
    for(int i=2; i<mr; i++) {
        if(!notp[i]) {
            pr[pn++]=i;
            miu[i]=-1;
        }
        for(int j=0; j<pn && i*pr[j]<mr; j++) {
            int k=i*pr[j];
            miu[k]=(i%pr[j])?-miu[i]:0;
            notp[k]=1;
            if(i%pr[j]==0) break;
        }
    }
    smiu[0]=0;
    for(int i=1;i<mr;i++)
        smiu[i]=smiu[i-1]+miu[i];
}
void getmiuPime() {//计算gcd(x,y)=Prime时专用，重定义miu以及smiu,计算时调用work(a,b)
    pn=0;
    memset(cnt_p,0,sizeof(cnt_p));
    memset(cnt_p2,0,sizeof(cnt_p2));
    smiu[0]=smiu[1]=0;
    for(int i=2; i<N; i++) {
        if(!cnt_p[i]) {
            pr[pn++]=i;
            cnt_p[i]=1;
        }
        for(int j=0; j<pn && i*pr[j]<N; j++) {
            int k=i*pr[j];
            cnt_p[k]=cnt_p[i]+1;

cnt_p2[k]=cnt_p2[i]+(i%pr[j]==0);
            if(i%pr[j]==0) {
                break;
            }
        }
        int t;
        if(cnt_p2[i]>1)t=0;
        else
if(cnt_p2[i]==1)t=((cnt_p[i]-1)&1)?-1:1;
        else
t=((cnt_p[i]-1)&1)?-cnt_p[i]:cnt_p[i];
        smiu[i]=smiu[i-1]+t;
    }
}
ll work(int a,int b){//计算<[1..a],[1..b]>中，gcd=1的个数
```

```
      //当a==b时，ans=Sum(phi[1..a])*2-1;
      int n=min(a,b),d1,d2,n1,n2,nn;
      ll ans=0;
      for(int i=1,nn;i<=n;i=nn+1){
            d1=a/i,d2=b/i;
            n1=a/d1,n2=b/d2;
            nn=min(n1,n2);
            ans+=(ll)d1*d2*(smiu[nn]-smiu[i-1]);
      }
      return ans;
}
ll work(int a,int b,int c){//计算
<[1..a],[1..b],[1..c]>中，gcd=1的个数
      int
n=min(min(a,b),c),d1,d2,d3,n1,n2,n3,nn;
      ll ans=0;
      for(int i=1;i<=n;i=nn+1){
            d1=a/i,d2=b/i,d3=c/i;
            n1=a/d1,n2=b/d2,n3=c/d3;
            nn=min(min(n1,n2),n3);

ans+=(ll)d1*d2*d3*(smiu[nn]-smiu[i-1]);
      }
      return ans;
}
ll lookfor(int a,int b,int d){//计算
<[1..a],[1..b]>中，gcd=d的个数
      if(a==0||b==0||d==0)return 0;
      return work(a/d,b/d);
}
ll lookfor(int a,int b,int c,int d){//计算
<[1..a],[1..b],[1..c]>中，gcd=d的个数
      if(a==0||b==0||c==0||d==0)return 0;
      return work(a/d,b/d,c/d);
}
ll solve(int a,int b,int c){//计算x,y,z方向各有
为a,b,c个点的长方体从一个顶点看能够看到的整点数
      a--,b--,c--;
      return
3+work(a,b,c)+work(a,c)+work(a,b)+work(b,c);
}
int main() {
      int n;
      getmiu();
      scanf("%d",&n);
      while(scanf("%d",&n)!=EOF){
            printf("%lld\n",solve(n+1,n+1,n+1));
      }
      return 0;
}
```

/**************************************/

表达式求值

/**************************************/

```
char op[8]= {'+','-','*','/','(',')','\0'};
char cp[7][7]= {
      {'>','>','<','<','<','>','>'},
      {'>','>','<','<','<','>','>'},
      {'>','>','>','>','<','>','>'},
      {'>','>','>','>','<','>','>'},
      {'<','<','<','<','<','=','@'},
      {'>','>','>','>','@','>','>'},
      {'<','<','<','<','<','@','='}
};
char st_char[N];
int st_int[N],top_char,top_int;
int cal(int a,char op,int b) {
      switch(op) {
      case '+':
            return a+b;
            break;
      case '-':
            return a-b;
            break;
      case '*':
            return a*b;
            break;
      case '/':
            return a/b;
            break;
      default:
            while(1)puts("You Will Output Limit
Exceeded\n");
      }
}
char comp(char a,char b) {
      int i,j;
      for(i=0; i<7; i++)
            if(a==op[i])
                  break;
      for(j=0; j<7; j++)
            if(b==op[j])
                  break;
      return cp[i][j];
}
int cac(char s[]) {
      top_char=top_int=0;
      st_char[top_char++]='\0';
      int i=0,v,a,b;
      char ch;
      bool flag=true;

while(s[i]!='\0'||st_char[top_char-1]!='\0') {
            if(s[i]==' '||s[i]=='\t') {
                  i++;
                  continue;
            } else if(isdigit(s[i])) {
                  v=0;
                  while(isdigit(s[i])) {
                        v=v*10+s[i]-'0';
                        i++;
                  }
                  st_int[top_int++]=v;
            } else {

switch(comp(st_char[top_char-1],s[i])) {
                  case '<':
                        st_char[top_char++]=s[i++];
                        break;
                  case '=':
                        top_char--;
                        i++;
                        break;
                  case '>':
                        ch=st_char[--top_char];
                        b=st_int[--top_int];
                        a=st_int[--top_int];
                        v=cal(a,ch,b);
                        st_int[top_int++]=v;
                        break;
                  }
            }
      }
      return st_int[0];
}
```

/**************************************/

模线性方程组

/**************************************/

```
//x=B[i](mod M[i]), 不要求M互质
ll solve(ll B[], ll M[], int num) {
      int i;
      bool flag = false;
      ll m1 = M[0], m2, b1 = B[0], b2, bb, d,
t, k, x, y;
      for (i = 1; i < num; i++) {
            m2 = M[i], b2 = B[i];
            bb = b2 - b1;
            d = Egcd (m1, m2, x, y);
            if (bb % d) {
                  flag = true;
                  break;
            }
            k = bb / d * x;
            t = m2 / d;
            if (t < 0) t = -t;
            k = (k % t + t) % t;
            b1 = b1 + m1*k;
            m1 = m1 / d * m2;
      }
      if (flag)
            return -1;
      return b1;
}
```

/**************************************/

Lucas-组合数取模

/**************************************/

```
//mod必须是质数
const int mod=10007;
```

```
int a[mod];
void init() {
    int i;
    a[0]=1;
    for(i=1; i<mod; i++)
        a[i]=(a[i-1]*i)%mod;
}
int choose(int n,int m) {
    if(m>n)return 0;
    else if(n==m)return 1;
    int nn=a[n],mm=(a[m]*a[n-m])%mod;
    int d=gcd(nn,mm);
    nn/=d;
    mm/=d;
    int x,y;
    Egcd(mm,mod,x,y);
    x=(x+mod)%mod;
    return (x*nn)%mod;
}
```

/**************************************/

快速组合数取模

/**************************************/

```
/**
mod不一定是质数
init:
   getpri()
**/
ll count(ll n,ll prime) {
    ll ret=0;
    while(n/prime) {
        ret+=n/prime;
        n/=prime;
    }
    return ret;
}
ll choose(ll n,ll m) {
    ll ans=1,cnt;
    for(int i=0; i<pn&&pr[i]<=n; i++) {

cnt=count(n,pr[i])-count(m,pr[i])-count(n-m,pr
[i]);
        ans=ans*fastMod(pr[i],cnt)%mod;
        if(ans==0)break;
    }
    return ans;
}
```

/**************************************/

整数的质因数分解

/**************************************/

```
ll factor[100],fac_top = -1;
ll pollard_rho(ll n,int c) {
    ll x,y,d,i = 1,k = 2;
    x = rand()%(n-1)+1;
    y = x;
    while(true) {
        i++;
        x = (mulmod(x,x,n) + c) % n;
        d = gcd(y-x, n);
        if(1 < d && d < n)
            return d;
        if( y == x)
            return n;
        if(i == k) {
            y = x;
            k <<= 1;
        }
    }
}
void findFactor(ll n,int k) {
    if(n==1)return;
    if(miller_rabin(n, 6)) {
        factor[++fac_top] = n;
        return;
    }
    ll p = n;
    while(p >= n)
        p = pollard_rho(p,k--);
    findFactor(p,k);
    findFactor(n/p,k);
}
ll tot[100],num[100],top;
void solve(ll n) {
    srand(time(0));
    top=0;
    if(miller_rabin(n,6)) {
        num[0]=n;
        tot[0]=1;
        top=1;
        return;
    }
    findFactor(n,107);
    for(int i=0; i<=fac_top; i++) {
        if(n%factor[i]==0) {
            num[top]=factor[i];
            tot[top]=0;
            while(n%factor[i]==0) {
                tot[top]++;
                n/=factor[i];
            }
            top++;
        }
    }
}
```

/**************************************/

递归求等比数列之和

/**************************************/

```
long long sum(long long p,long long n) //递归二
分求 (1 + p + p^2 + p^3 +...+ p^n)%mod
{
   //奇数二分式 (1 + p + p^2 +...+ p^(n/2)) * (1 +
p^(n/2+1))
   if(n==0)              //偶数二分式 (1 + p + p^2
+...+ p^(n/2-1)) * (1+p^(n/2+1)) + p^(n/2)
       return 1;
   if(n%2)   //n为奇数,
       return
(sum(p,n/2)*(1+power(p,n/2+1)))%mod;
   else     //n为偶数
       return
(sum(p,n/2-1)*(1+power(p,n/2+1))+power(p,n/2))
%mod;
}
```

/**************************************/

最优子区间

/**************************************/

```
//给定长为n的数组，求一个区间使得该区间的平均值最大，要
求区间长度至少为m
ll cow[N],dp[N],num[N],sum[N];
int main() {
    int n,m,i;
    ll ans,len;
    scanf("%d%d",&n,&m);
    sum[0]=0;
    for(i=1; i<=n; i++) {
        scanf("%lld",cow+i);
        sum[i]=sum[i-1]+cow[i];
    }
    dp[m]=sum[m];
    num[m]=m;
    ans=dp[m];
    len=m;
    for(i=m+1; i<=n; i++) {
        dp[i]=dp[i-1]+cow[i];
        num[i]=num[i-1]+1;
        if(dp[i]*m<(sum[i]-sum[i-m])*num[i])
{
            dp[i]=sum[i]-sum[i-m];
            num[i]=m;
        }
        if(ans*num[i]<dp[i]*len) {
            ans=dp[i];
            len=num[i];
        }
    }
    printf("%lld\n",ans/len);
    return 0;
}
```

/**************************************/

高精度

```
/************************************/
const  int MAXL = 10000;
struct  BigNum {
     int  num[MAXL];
     int  len;
};
int  Comp(BigNum a, BigNum b) {
     int  i;
     if(a.len != b.len) return (a.len > b.len) ?
1 : -1;
     for(i = a.len-1; i >= 0; i--)
          if(a.num[i] != b.num[i]) return
(a.num[i] > b.num[i]) ? 1 : -1;
     return  0;
}
BigNum IntToBigNum(int a) {
     BigNum c;
     c.len =0;
     int i=0;
     while(a>0) {
          c.num[i]=a%10;
          i++;
          a=a/10;
     }
     c.len=i;
     return c;
}
BigNum  Add(BigNum a, BigNum b) { //a+b
     BigNum c;
     int  i, len;
     len = (a.len > b.len) ? a.len : b.len;
     memset(c.num, 0, sizeof(c.num));
     for(i = 0; i < len; i++) {
          c.num[i] += (a.num[i]+b.num[i]);
          if(c.num[i] >= 10) {
               c.num[i+1]++;
               c.num[i] -= 10;
          }
     }
     if(c.num[len]) len++;
     c.len = len;
     return  c;
}
BigNum Sub(BigNum &a, BigNum &b) {//a-b
     BigNum  c;
     int  i, len;
     len = (a.len > b.len) ? a.len : b.len;
     memset(c.num, 0, sizeof(c.num));
     for(i = 0; i < len; i++) {
          c.num[i] += (a.num[i]-b.num[i]);
          if(c.num[i] < 0) {
               c.num[i] += 10;
               c.num[i+1]--;
          }
     }
     while(c.num[len-1] == 0 && len > 1) len--;
     if(c.num[len]) len++;
     c.len = len;
     return  c;
}
BigNum Mul1(BigNum a, int  b) { //a*b
     BigNum c;
     int  i, len;
     len = a.len;
     memset(c.num, 0, sizeof(c.num));
     if(b == 0) {
          c.len = 1;
          return  c;
     }
     for(i = 0; i < len; i++) {
          c.num[i] += (a.num[i]*b);
          if(c.num[i] >= 10) {
               c.num[i+1] = c.num[i]/10;
               c.num[i] %= 10;
          }
     }
     while(c.num[len] > 0) {
          c.num[len+1] = c.num[len]/10;
          c.num[len++] %= 10;
     }
     c.len = len;
     return  c;
}
BigNum  Mul2(BigNum a, BigNum b) { //a*b
     int  i, j, len = 0;
     BigNum  c;
     memset(c.num, 0, sizeof(c.num));
     for(i = 0; i < a.len; i++)
          for(j = 0; j < b.len; j++) {
               c.num[i+j] +=
(a.num[i]*b.num[j]);
               if(c.num[i+j] >= 10) {
                    c.num[i+j+1] +=
c.num[i+j]/10;
                    c.num[i+j] %= 10;
               }
          }
     len = a.len+b.len;
     while(c.num[len-1] == 0 && len > 1) len--;
     if(c.num[len]) len++;
     c.len = len;
     return  c;
}
//高精度除以低精度,除的结果为c, 余数为f
void Div1(BigNum &a, int &b, BigNum &c, int &f)
{
     int  i, len = a.len;
     memset(c.num, 0, sizeof(c.num));
     f = 0;
     for(i = a.len-1; i >= 0; i--) {
          f = f*10+a.num[i];
          c.num[i] = f/b;
          f %= b;
     }
     while(len > 1 && c.num[len-1] == 0) len--;
     c.len = len;
}
void  Mul10(BigNum &a) { //a*10
     int  i, len = a.len;
     for(i = len+1; i >= 1; i--)
          a.num[i] = a.num[i-1];
     a.num[i] = 0;
     len++;
     //if a == 0
     while(len > 1 && a.num[len-1] == 0)  len--;
}
//高精度除以高精度, 除的结果为c, 余数为f
void Div2(BigNum &a, BigNum &b, BigNum &c, BigNum
&f) {
     int  i, len = a.len;
     memset(c.num, 0, sizeof(c.num));
     memset(f.num, 0, sizeof(f.num));
     f.len = 1;
     for(i = len-1; i >= 0; i--) {
          Mul10(f);
          f.num[0] = a.num[i];
          while(Comp(f, b) >= 0) {
               f = Sub(f, b);
               c.num[i]++;
          }
     }
     while(len > 1 && c.num[len-1] == 0)
          len--;
     c.len = len;
}
void  print(BigNum &a) {
     int  i;
     for(i = a.len-1; i >= 0; i--)
          printf("%d", a.num[i]);
     puts("");
}
//将字符串转为大数存在BigNum结构体里面
BigNum ToNum(char *s) {
     int i, j;
     BigNum  a;
     a.len = strlen(s);
     for(i = 0, j = a.len-1; s[i] != '\0'; i++,
j--)
          a.num[i] = s[j]-'0';
     return  a;
}
void  Init(BigNum &a, char *s, int &tag) {
     memset(a.num, 0, sizeof(a.num));
     int  i = 0, j = strlen(s);
     if(s[0] == '-') {
          j--;
          i++;
          tag*= -1;
     }
     a.len = j;
     for(; s[i] != '\0'; i++, j--)
          a.num[j-1] = s[i]-'0';
}
```

/**************************************/

第 K 个与 m 互质的数

/**************************************/

```
int p[1000000],pn;
void solve(int m) {
    p[0]=1;
    pn=1;
    for(int i=2; i<m; i++)
        if(gcd(i,m)==1)
            p[pn++]=i;
    return;
}
int main() {
    int m,k;
    while(scanf("%d%d",&m,&k)!=EOF) {
        solve(m);

printf("%lld\n",(ll)p[(k-1)%pn]+(k-1)/pn*m);
    }
    return 0;
}
```

/**************************************/

行列式计算

/**************************************/

```
int mat[500][500];
int Gauss(int n) {
    int x,y;
    for (int i = 1; i < MOD; i++) {
        Egcd(i,MOD,x,y);
        ni[i] = (x+MOD) % MOD;
    }
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++)
            mat[i][j] = (mat[i][j] + MOD) %
MOD;
    }
    int col = 0, k;
    int ans = 1;
    for(k = 0; k < n && col < n; ++k, ++col)
{
        if(mat[k][col] == 0) {
            for(int i = k + 1; i < n; ++i)
{
                if(!(mat[i][col] == 0)) {
                    for(int j = col; j < n;
++j)swap(mat[k][j], mat[i][j]);
                    ans *= -1;
                    break;
                }
            }
        }
        int x = mat[k][col];
        ans *= x;
        ans %= MOD;
        ans += MOD;
        ans %= MOD;
        for(int i = k + 1; i < n; ++i) {
            int y = mat[i][col];
            if(x == 0 || y == 0) continue;
            int d = gcd(abs(x),abs(y)),lcm =
abs(x * y / d);
            int tx = lcm / x,ty = lcm / y;
            for(int j = col; j < n; ++j) {
                mat[i][j] = -tx * mat[k][j]
+ ty * mat[i][j];
                mat[i][j] %= MOD;
                mat[i][j] = (mat[i][j] +
MOD) % MOD;
            }
            ans = (ans * ni[ty]) % MOD;
            ans = (ans + MOD) % MOD;
        }
    }
    ans %= MOD;
    ans += MOD;
    ans %= MOD;
    return ans;
}
```

/**************************************/

排列 P（n，m）最后非零位

/**************************************/

```
int getnum(int n,int p) {
    int sum=0,val=p;
    while(n/val&&val>0) {
        sum=sum+(n/val);
        val=val*p;
    }
    return sum;
}
int getx(int n,int p) {
    int i,j,sum=0;
    for(i=n; i; i>>=1) //将2,4,6,8,10...变为
1,2,3,4,5...
        for(j=i; j; j/=5) { //将
5,10,15,25,30.....变为1,2,3,4,5...
            sum=sum+j/10+(j%10>=p);
        }
    return sum;
}
int num[10];
int po[4][4]=
{6,2,4,8,1,3,9,7,1,7,9,3,1,9,1,9};
int main() {
    int n,m;
    while(~scanf("%d%d",&n,&m)) {
        m=n-m;
        num[2]=getnum(n,2)-getnum(m,2);
        num[5]=getnum(n,5)-getnum(m,5);
        int ans;
        if(num[5]>num[2]) {
            printf("5\n");
            continue;
        } else if(num[5]==num[2]) {
            ans=1;
        } else {
            ans=po[0][(num[2]-num[5])%4];
        }
        num[3]=getx(n,3)-getx(m,3);
        num[7]=getx(n,7)-getx(m,7);
        num[9]=getx(n,9)-getx(m,9);
        ans=ans*po[1][num[3]%4]%10;
        ans=ans*po[2][num[7]%4]%10;
        ans=ans*po[3][num[9]%4]%10;
        printf("%d\n",ans);
    }
    return 0;
}
```

/**************************************/

取石子游戏

/**************************************/

1、 N 堆石子，每次选择一堆取走若干个，必胜态为

$$n_1 \wedge n_2 \ldots . \wedge n_N \neq 0$$

2、 N 堆石子，每次选择一堆取走最多 m 个，必胜态为

$$[n_1 \%(m+1)] \wedge [n_2 \%(m+1)] \ldots . \wedge [n_N \%(m+1)] \neq 0$$

3、 N 堆石子，每次选择最多 k 堆取走若干个，必胜态，对每一位分别相加，然后再%（k+1），不全为 0。

4、 2 堆石子，每次选择一堆取走任意个或者选择两堆取走相同个，必败态为(a,b)，其中 Θ*(b-a)下取整等于 a, Θ 为黄金分割比

$$\frac{\sqrt{5}+1}{2}$$

5、 **Anti-nim 游戏**

[定义]

(1)桌子上有 N 堆石子，游戏者轮流取石子。

(2)每次只能从一堆中取出任意数目的石子，但不能不取。

(3)取走最后一个石子者败。

[结论]

先手必胜当且仅当：

(1)所有堆的石子数都为 1 且游戏的 SG 值为 0；

(2)有些堆的石子数大于 1 且游戏的 SG 值不为 0。

[定理]（SJ 定理）

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。

6、 **Every-nim 游戏**

[定义]

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

[结论]

对于Every-SG 游戏先手必胜当且仅当单一游戏中最大 step为奇数。（对于SG 值为0 的点，我们需要知道最快几步能将游戏带入终止状态，对于SG 值不为0 的点，我们需要知道最

慢几步游戏会被带入终止状态，我们用函数来表示这个值。step）

$$step(u)=\begin{cases}0 & u\text{为终止状态}\\ \max(step(v))+1 & SG(u)>0\text{^v为u后继^}SG(v\\ \min(step(v))+1 & SG(u)=0\text{^v为u后继}\end{cases}$$

/**************************************/

Nim 游戏必胜方法数

/**************************************/

```
int a[1005];
int main() {
    int n,i,res,cnt;
    while(scanf("%d",&n)&&n) {
        res=0;
        for(i=0; i<n; i++) {
            scanf("%d",a+i);
            res^=a[i];
        }
        if(res==0) {
            printf("%d\n",0);
            continue;
        }
        for(cnt=i=0; i<n; i++) {
            if((a[i]^res)<a[i])//异或优先级最低，注意加括号
                cnt++;
        }
        printf("%d\n",cnt);
    }
    return 0;
}
```

/**************************************/

猜数游戏

/**************************************/

```
//给定可以猜的次数以及可以超出数的次数，求数范围的最大值，使得存在必胜态
int dp[35][35];
int main() {
    int i,j,k;
    memset(dp,0,sizeof(dp));
    for(i=1; i<=30; i++)
        dp[i][0]=i;
    for(i=1; i<=30; i++) {
        for(j=1; j<=30; j++) {

dp[j][i]=dp[j-1][i-1]+dp[j-1][i]+1;
        }
    }
    k=0;
    while(scanf("%d%d",&i,&j)) {
        if(i==0&&j==0)
            break;
        else

printf("Case %d: %d\n",++k,dp[i][j]);
    }
    return 0;
}
```

/**************************************/

逆序数为 m 的最小序列

/**************************************/

```
ll c[50005];
int main() {
    int i,j,mid,k,n,m;
    c[0]=c[1]=0;
    for(i=2; i<=50000; i++)
        c[i]=c[i-1]+i-1;
    while(scanf("%d%d",&n,&m)) {
        if(n==-1&&m==-1)
            break;
        if(m==0) {
            for(i=1; i<n; i++)
                printf("%d ",i);
            printf("%d\n",n);
            continue;
        } else {
            for(i=2,j=n; i<j;) {
                mid=(i+j)>>1;
                if(c[mid]>=m)
                    j=mid;
                else
                    i=mid+1;
            }
            m=m-c[j-1];
            k=n-j+1+m;
            if(j!=n)
                printf("1");
            for(i=2; i<=n-j; i++)
                printf(" %d",i);
            if(j!=n)
                printf(" %d",k);
            else
                printf("%d",k);
            for(i=n; i>n-j; i--) {
                if(i!=k)
                    printf(" %d",i);
            }
            printf("\n");
        }
    }
}
```

/**************************************/

区间最大权选取

/**************************************/

```
//每个区间有权值，选取最多不重合区间使得权值最大
struct point {
    int x,y,c;
    bool operator<(const point next)const {
        if(y!=next.y)
            return y<next.y;
        else
            return x<next.x;
    }
} po[1005];
int dp[1005];
int main() {
    int n,m,r,i,j,k,ans=0;
    scanf("%d%d%d",&n,&m,&r);//每个区间必须至少相距 r
    for(i=1; i<=m; i++) {

scanf("%d%d%d",&po[i].x,&po[i].y,&po[i].c);
    }
    sort(po+1,po+m+1);
    for(i=1; i<=m; i++) {
        dp[i]=po[i].c;
    }
    ans=dp[1];
    for(i=2; i<=m; i++) {
        for(j=1; j<i; j++) {
            if(po[i].x>=po[j].y+r)

dp[i]=max(dp[i],dp[j]+po[i].c);
        }
        ans=max(ans,dp[i]);
    }
    printf("%d\n",ans);
    return 0;
}
```

/**************************************/

第 n 个回文数

/**************************************/

```
//从 1 开始
ll dp[100]= {0,9,9,90};
ll sum[100]= {0,9,18,108};
ll po[100]= {1,10,100,1000};
int cac(ll x) {
    int i;
    for(i=1; sum[i]<x; i++);
    return i;
}
void solve(ll k) {
    if(k<10ll) {
        printf("%d\n",(int)k);
        return;
    } else if(k<=18ll) {
        printf("%d%d\n",(int)k-9,(int)k-9);
        return;
    }
    int i,len,ll,stack[100],top=0;
```

```
    ll=len=cac(k);
    k=k-sum[ll-1]-1;
    top=ll=(ll+1)/2;
    stack[0]=(k/po[ll-1])+1;
    k=k-po[ll-1]*(stack[0]-1);
    while(k) {
        stack[--ll]=k%10;
        k/=10;
    }
    while(ll>1)
        stack[--ll]=0;
    if(len&1) {
        for(i=0; i<top; i++)
            printf("%d",stack[i]);
        for(i=top-2; i>=0; i--)
            printf("%d",stack[i]);
        printf("\n");
    } else {
        for(i=0; i<top; i++)
            printf("%d",stack[i]);
        for(i=top-1; i>=0; i--)
            printf("%d",stack[i]);
        printf("\n");
    }
}
int main() {
    int i;
    for(i=4; i<20; i++) {
        if(i&1)
            dp[i]=dp[i-1]*10;
        else
            dp[i]=dp[i-1];
        sum[i]=dp[i]+sum[i-1];
        po[i]=po[i-1]*10;
    }
    ll n;
    while(scanf("%lld",&n)!=EOF&&n) {
        solve(n);
    }
    return 0;
}
/**************************************/
```

权值最大子矩形

```
/**************************************/
int n,m,left[1002],s[1002],right[1002];
int getmax() {
    int i,j,tmax=0;
    left[0]=0;
    right[m+1]=0;
    for(i=1; i<=m; i++) {
        if(s[i]==0)continue;
        j=i-1;
        while(s[j]>=s[i])j=left[j];
        left[i]=j;
    }
    for(i=m; i>=1; i--) {
        if(s[i]==0)continue;
        j=i+1;
        while(s[j]>=s[i])j=right[j];
        right[i]=j;
    }
    for(i=1; i<=m;
i++)tmax=max(tmax,s[i]*(right[i]-left[i]-1));
    return tmax;
}
int main() {
    int T;
    for(scanf("%d",&T); T; T--) {
        int i,j,k,t,tmax=-inf;
        scanf("%d%d",&n,&m);
        memset(s,0,sizeof(s));
        for(i=1; i<=n; i++) {
            for(j=1; j<=m; j++) {
                char ch;
                scanf(" %c",&ch);
                if(ch=='F')s[j]++;
                else s[j]=0;
            }
            tmax=max(tmax,getmax());
        }
        printf("%d\n",tmax);
    }
    return 0;
}
/**************************************/
```

$$\sum_{i=1}^{n} \gcd(i,n)$$

```
/**************************************/
/**
init:
    getpri()
**/
ll work(ll n) {
    ll ans=1,x=n;
    for(int i=0,cnt; i<pn&&pr[i]*pr[i]<=x; i++)
{
        if(x%pr[i]==0) {
            cnt=0;
            while(x%pr[i]==0) {
                x/=pr[i];
                cnt++;
            }

ans*=(cnt*fastmod(pr[i],cnt-1,mod)*(pr[i]-1)+f
astmod(pr[i],cnt,mod));
        }
    }
    if(x>1) {
        ll tp=n/x;
        ans*=2*x-1;
    }
    returna ans;
}

/**************************************/
```

$$\sum_{i=1}^{n} k\%i$$

```
/**************************************/
ll work(ll n,ll k) {
    ll ans=0,d,next,low,high;
    if(n>k) {
        ans=(n-k)*k;
        n=k;
    }
    d=k/n;
    while(n>1) {
        next=k/(d+1);
        if(n==next) {
            ans+=k%n;
            n--;
            d=k/n;
            continue;
        }
        low=k%n;
        high=k%(next+1);
        ans+=(low+high)*(n-next)/2;
        d++;
        n=next;
    }
    return ans;
}
/**************************************/
```

$$a^x \equiv b(\mathrm{mod}\, c)$$

```
/**************************************/
typedef struct num {
    int ii, value;
} num;
num Num[nmax];
bool cmp(num a, num b) {
    return a.value<b.value;
}
//ax=b (mod n)
int inval(int a, int b, int n) {
    LL res,x,y;
    Egcd(a, n,x,y);
    res = (LL) x;
    res = res * b;
    res = (res % n + n) % n;
    return (int) res;
}
int bfindNum(int key, int n) {
    int left, right, mid;
    left = 0, right = n;
    while (left <= right) {
        mid = (left + right) >> 1;
        if (Num[mid].value == key)
```

```
            return Num[mid].ii;
        else if (Num[mid].value > key)
            right = mid - 1;
        else
            left = mid + 1;
    }
    return -1;
}
void baby_step_giant_step(int a, int b, int c)
{
    int i, j, te, d, cd, aa, ttemp;
    LL temp, tem;
    if (b >= c) {
        puts("No solution");
        return;
    }
    for (i = 0, temp = 1 % c, tem = temp; i
< nnum; i++, temp = temp * a % c) {
        if (temp == b) {
            printf("%d\n", i);
            return;
        }
    }
    cd = 0;
    while ((d = gcd(a, c)) != 1) {
        if (b % d) {
            puts("No solution");
            return;
        }
        cd++;
        c /= d, b /= d;
        tem = tem * a / d % c;
    }
    te = (int) (sqrt(c * 1.0) + 0.5);
    for (i = 0, temp = 1 % c; i <= te; i++,
temp = temp * a % c) {
        Num[i].ii = i;
        Num[i].value = (int) temp;
    }
    sort(Num,Num+te+1,cmp);
    for (i = 0; i <= te; i++, tem = tem * aa %
c) {
        ttemp = inval(tem, b, c);
        if (ttemp >= 0) {
            j = bfindNum(ttemp, te + 1);
            if (j != -1) {
                printf("%d\n", i * te + j +
cd);
                return;
            }
        }
    }
    puts("No solution");
}
void solve(int a, int b, int c) {
    baby_step_giant_step(a, b, c);
}
```

/**************************************/

带限制的项链计数

/**************************************/

```
//m种珠子凑成长度为n的项链，不考虑对称，有k对珠子不能
相邻
const LL mod=9973;
int fac[102],num[102];
int top,n,m;
LL ans;
struct MAT {
    LL bas[13][13];
    void init() {
        memset(bas,0,sizeof(bas));
    }
} mat[50];
MAT mul(MAT a,MAT b) {
    MAT c;
    c.init();
    for(int i=1; i<=m; i++)
        for(int k=1; k<=m; k++) {
            if(a.bas[i][k]) {
                for(int j=1; j<=m; j++) {

c.bas[i][j]+=a.bas[i][k]*b.bas[k][j];
                    if(c.bas[i][j]>=mod)
                        c.bas[i][j]%=mod;
                }
            }
        }
    return c;
}
void divn() {
    int nn=n;
    top=0;
    int lim=(int)sqrt((double(nn)))+1;
    for(int i=0; pr[i]<=lim; i++) {
        if(nn%pr[i]==0) {
            fac[top]=pr[i];
            num[top]=0;
            while(nn%pr[i]==0)
                num[top]++,nn/=pr[i];
            top++;
        }
    }
    if(nn>1)
        fac[top]=nn,num[top++]=1;
}
int phi(int x) {
    int i, res=x;
    for (i=0; pr[i]<(int)sqrt((double)x)+1;
i++)
        if(x%pr[i]==0) {
            res=res/pr[i]*(pr[i]-1);
            while(x%pr[i]==0)x/=pr[i];
        }
    if(x>1)res=res/x*(x-1);
    return res;
}
void solve(int r) {
    int res=phi(n/r);
    MAT mt;
    mt.init();
    for(int i=1; i<=m; i++)
        mt.bas[i][i]=1;
    for(int i=1,tp=r; tp; i++,tp>>=1)
        if(tp&1)mt=mul(mt,mat[i]);
    for(int i=1; i<=m; i++) {
        ans+=mt.bas[i][i]*res;
        if(ans>=mod)ans%=mod;
    }
}
void dfs(int id,int sum) {
    if(id==top) {
        solve(sum);
        return;
    } else {
        dfs(id+1,sum);
        for(int ct=0; ct<num[id]; ct++)
            dfs(id+1,sum=sum*fac[id]);
    }
}
void init() {
    for(int i=2; i<50; i++)
        mat[i]=mul(mat[i-1],mat[i-1]);
}
int main() {
    getpri();
    int T;
    for(scanf("%d",&T); T; T--) {
        int k;
        scanf("%d%d%d",&n,&m,&k);
        ans=0;
        for(int i=1; i<=m; i++)
            for(int j=1; j<=m; j++)
                mat[1].bas[i][j]=1;
        for(int a,b,i=0; i<k; i++) {
            scanf("%d%d",&a,&b);

mat[1].bas[a][b]=mat[1].bas[b][a]=0;
        }
        init();
        divn();
        dfs(0,1);
        printf("%d\n",ans*getni()%mod);
    }
    return 0;
}
```

/**************************************/

整数拆分的最大的最小公倍数

/**************************************/

```
double lod[500][N];
int dp[500][N],n,mod;
```

```
inline int mul(int a,int b) {
    ll c=(ll)a*b;
    if(c>=mod)return c%mod;
    return c;
}
int main() {
    getpri();
    while(scanf("%d%d",&n,&mod)!=EOF) {
        for(int i=0; i<=n; i++) {
            dp[0][i]=1;
            lod[0][i]=0.0;
        }
        int ans=1;
        for(int i=0,p=pr[i]; i<pn&&p<=n;
p=pr[++i]) {
            for(int j=0; j<=n; j++) {
                lod[i+1][j]=lod[i][j];
                dp[i+1][j]=dp[i][j];
            }
            for(int j=p; j<=n; j++) {
                double
mxf=-1,tmp,lp=log((double)p);
                int pt=1;
                for(int pk=p,k=1; pk<=j;
pk*=p,k++) {
                    tmp=lod[i][j-pk]+k*lp;
                    if(tmp>mxf) {
                        mxf=tmp;
                        pt=pk;
                    }
                }
                if(mxf>lod[i+1][j]) {
                    lod[i+1][j]=mxf;

dp[i+1][j]=mul(dp[i][j-pt],pt);
                }
            }
            ans=dp[i+1][n];
        }
        printf("%d\n",ans);
    }
    return 0;
}
```

/**************************************/

第二类斯特林数奇偶性

```
/**************************************/
int Find(int x) {
    int ans=0;
    for(int i=2; x>=i; i<<=1)ans+=x/i;
    return ans;
}
int main() {
    int T;
    for(scanf("%d",&T); T; T--) {
        int n,k,z,w;
        scanf("%d%d",&n,&k);
        z=n-(k+2)/2;
        w=(k-1)/2;

if(Find(z)-Find(w)-Find(z-w)>0)puts("0");
        else puts("1");
    }
    return 0;
}
```

/**********/

数据结构

/**********/

/**************************************/

堆

```
/**************************************/
void up(int a[],int i) {
    int t=a[i];
    while(i>1&&t<a[i/2]) {
        a[i]=a[i/2];
        i/=2;
    }
    a[i]=t;
}
void down(int a[],int i,int n) {
    int t=a[i],u=i*2;
    while(u+1<=n) {
        u=a[u]<a[u+1]?u:u+1;
        if(t<=a[u])
            break;
        a[i]=a[u];
        i=u;
        u=i*2;
    }
    a[i]=t;
}
```

/**************************************/

AC 自动机

```
/**************************************/
#define maxchar 26
#define MAX 1000001
struct Node {
    int next[maxchar],fall,f;
    void init() {
        memset(&next,-1,sizeof(next));
        f = 0;
    }
} node[MAX];
int NT;
void preprocess() {
    node[NT=0].init();
    node[0].fall = -1;
}
void insert(char a[]) {
    int father,index,i;
    father = 0;
    for(i=0; a[i]; i++) {
        int x = a[i]-'a';
        index = node[father].next[x];
        if(index==-1) {
            ++NT;
            node[NT].init();
            node[father].next[x] = NT;
            index = NT;
        }
        father = index;
    }
    node[father].f++;
}
void KMP() {
    int i,father,index;
    queue<int>q;
    q.push(0);
    while(!q.empty()) {
        int t = q.front();
        q.pop();
        for(i=0; i<maxchar; i++) {
            index = node[t].next[i];
            if(index!=-1) {
                father = node[t].fall;

while(father!=-1&&node[father].next[i]==-1) {
                    father =
node[father].fall;
                }
                if(father!=-1) {
                    node[index].fall =
node[father].next[i];
                } else {
                    node[index].fall = 0;
                }
                q.push(index);
            }
        }
    }
}
int find(char a[]) {
    int i,father,index,ct = 0;
    father = 0;
    for(i=0; a[i];) {
        int x = a[i]-'a';
        index = node[father].next[x];
        if(index!=-1) {
            if(node[index].f) {
                ct+=node[index].f;
                node[index].f = 0;
            }
            father = index;
            i++;
        } else {
```

```
while(father!=-1&&node[father].next[x]==-1) {
                    father = node[father].fall;
                    if(node[father].f) {
                         ct+=node[father].f;
                         node[father].f = 0;
                    }
               }
               if(father==-1) {
                    father = 0;
                    i++;
               }
          }
     }
     return ct;
}
```

/**************************************/

划分树

/**************************************/

```
const int maxn = 100020;
int Left[20][maxn], sorted[maxn],
tree[20][maxn];
//left[i][j]表示第 i 层前 j 个数中有几个被分到左子树中
//sorted 表示排好序的
//tree[i][j]记录第 i 层第 j 个元素
void build_tree( int L, int R, int v ) {
     int i;
     int mid = ( L + R ) /2;
     if( L == R ) return;
     int m = sorted[mid];
     int same = mid - L + 1;// same 表示和 m = sorted[mid] 相等且分到左边的
     for( i = L; i <= R; i++ )
          if( tree[v][i] < m ) same--;
     int lpos = L;
     int rpos = mid+1;
     int ss = 0;
     for( i = L; i <= R; i++ ) {
          if( i == L ) Left[v][i] = 0;
          else Left[v][i] = Left[v][i-1];
          if( tree[v][i] < m ) {
               tree[v+1][lpos++] = tree[v][i];
               Left[v][i]++;
          } else if( tree[v][i] > m ) {
               tree[v+1][rpos++] = tree[v][i];
          } else {
               if( ss < same ) {
                    tree[v+1][lpos++] = tree[v][i];
                    Left[v][i]++;
                    ss++;
               } else tree[v+1][rpos++] = tree[v][i];
          }
     }
     build_tree( L, mid, v + 1 );
     build_tree( mid + 1, R, v + 1 );
}
int query( int L, int R, int l, int r, int k, int v ) {
     int mid = ( L + R ) /2 ;
     if( l == r ) return tree[v][l];
     int off;        // off 表示 [ L, l-1 ]有多少个分到左边
     int cnt;        // cnt 表示 [ l, r ]有多少个分到左边
     if( l == L ) {
          off = 0;
          cnt = Left[v][r];
     } else {
          off = Left[v][l-1];
          cnt = Left[v][r] - Left[v][l-1];
     }
     if( cnt >= k ) { //有多于 k 个分到左边,显然去左儿子区间找第 k 个
          int lnew = L + off;
          int rnew = lnew + cnt - 1;
          return query( L, mid, lnew, rnew, k, v + 1 );
     } else {
          off = l - L - off;        // off 表示 [ L, l-1 ]有多少个分到右边
          k = k - cnt;
          cnt = r - l + 1 - cnt;  // cnt 表示 [ l, r ]有多少个分到右边
          int lnew = mid + 1 + off;
          int rnew = lnew + cnt - 1;
          return query( mid + 1, R, lnew, rnew, k, v + 1 );
     }
}
int main() {
     int n, m, l, r, k,i;
     while(scanf("%d%d",&n,&m)!=-1) {
          for( i = 1; i <= n; i++ ) {
               scanf("%d",&tree[0][i]);
               sorted[i] = tree[0][i];
          }
          sort( sorted + 1, sorted + n + 1 );
          build_tree( 1, n, 0 );
          for( i = 0; i < m; i++ ) {
               scanf("%d%d%d",&l,&r,&k);
               printf("%d\n",query( 1, n, l, r, k, 0 ) );
          }
     }
     return 0;
}
```

/**************************************/

树状数组（第 k 大值）

/**************************************/

```
/**
init:
    对所有 v,add(v,1)
**/
int find_k(int k) {
     int ans=0,cnt=0;
     for(int i=19; i>=0; i--) {
          ans+=(1<<i);
          if(ans>N||cnt+ar[ans]>=k) {
               ans-=(1<<i);
          } else {
               cnt+=ar[ans];
          }
     }
     return ans+1;
}
```

/**************************************/

树状数组（区域维护）

/**************************************/

```
//一维区间维护：区间加减，区间求和
ll a[N],d[N],d2[N];
ll sum(ll *ar,int i){
     ll s=0;
     for(;i>0;s+=ar[i],i-=lowbit(i));
     return s;
}
void add(ll *ar,int i,ll v){
     for(;i<N;ar[i]+=v,i+=lowbit(i));
}
ll query(int l,int r){
     return
a[r]-a[l-1]+(r+1)*sum(d,r)-sum(d2,r)-l*sum(d,l-1)+sum(d2,l-1);
}
int main(){
     int n,m;
     while(scanf("%d%d",&n,&m)!=EOF){
          memset(d,0,sizeof(d));
          memset(d2,0,sizeof(d2));
          a[0]=0;
          for(int i=1;i<=n;i++){
               scanf("%lld",a+i);
               a[i]+=a[i-1];
          }
          char op;
          int l,r;
          ll c;
          for(int i=0;i<m;i++){
               scanf(" %c",&op);
               if(op=='C'){
                    scanf("%d%d%lld",&l,&r,&c);
                    add(d,l,c);
                    add(d,r+1,-c);
                    add(d2,l,c*l);
                    add(d2,r+1,-c*(r+1));
```

```
                }
                else{
                    scanf("%d%d",&l,&r);

printf("%lld\n",query(l,r));
                }
            }
        }
        return 0;
}
//二维区域维护：区域加减，区域求和
int d[N][N],d1[N][N],d2[N][N],d3[N][N];
int gs(int ar[][N],int x,int y){
        int s=0;
        for(;x>0;x-=lowbit(x)){
            for(int j=y;j>0;j-=lowbit(j)){
                s+=ar[x][j];
            }
        }
        return s;
}
int ga(int ar[][N],int x,int y,int v){
        for(;x<N;x+=lowbit(x)){
            for(int j=y;j<N;j+=lowbit(j)){
                ar[x][j]+=v;
            }
        }
}
int sum(int x,int y){
        return
(x+1)*(y+1)*gs(d,x,y)-(y+1)*gs(d1,x,y)-(x+1)*g
s(d2,x,y)+gs(d3,x,y);
}
int sum(int x1,int y1,int x2,int y2){
        return
sum(x2,y2)-sum(x2,y1-1)-sum(x1-1,y2)+sum(x1-1,
y1-1);
}
int add(int x1,int y1,int x2,int y2,int v){
        ga(d,x1,y1,v);
        ga(d,x2+1,y1,-v);
        ga(d,x1,y2+1,-v);
        ga(d,x2+1,y2+1,v);

        ga(d1,x1,y1,v*x1);
        ga(d1,x2+1,y1,-v*(x2+1));
        ga(d1,x1,y2+1,-v*x1);
        ga(d1,x2+1,y2+1,v*(x2+1));

        ga(d2,x1,y1,v*y1);
        ga(d2,x2+1,y1,-v*y1);
        ga(d2,x1,y2+1,-v*(y2+1));
        ga(d2,x2+1,y2+1,v*(y2+1));

        ga(d3,x1,y1,v*x1*y1);
        ga(d3,x2+1,y1,-v*(x2+1)*y1);
        ga(d3,x1,y2+1,-v*x1*(y2+1));
        ga(d3,x2+1,y2+1,v*(x2+1)*(y2+1));
}
int main(){
        char op[2];
        int row,col,x1,x2,y1,y2,v;
        while(scanf("%s",op)!=EOF){
            switch(op[0]){
                case 'X':
                    scanf("%d%d",&row,&col);
                    memset(d,0,sizeof(d));
                    memset(d1,0,sizeof(d1));
                    memset(d2,0,sizeof(d2));
                    memset(d3,0,sizeof(d3));
                    break;
                case 'L':

scanf("%d%d%d%d%d",&x1,&y1,&x2,&y2,&v);
                    add(x1,y1,x2,y2,v);
                    break;
                case 'k':

scanf("%d%d%d%d",&x1,&y1,&x2,&y2);

printf("%d\n",sum(x1,y1,x2,y2));
                    break;
            }
        }
        return 0;
}
```

/**************************************/

树状数组（约瑟夫环）

/**************************************/

```
/*
树状数组实现，每个人有标号，从第 k 个人开始，报到 to[i]的
人出列，i 为上一个出列的人的编号。
*/
int main() {
    int m,k;
    while(scanf("%d%d",&n,&k)!=EOF) {
        memset(ar,0,sizeof(ar));
        int tp,i,cur,nn=n;
        for(i=1; i<=n; i++) {
            scanf("d",&to[i]);
            add(i,1);
        }
        cur=k;
        while(--nn) {
            tp=find_k(cur);
            cur=cur+to[tp];
            if(to[tp]>0)
                cur--;
            cur=(cur%nn+nn)%nn;
            if(cur==0)
                cur=nn;
            add(tp,-1);
        }
        tp=find_k(1);
        printf("%d\n",tp);
    }
    return 0;
}
```

/**************************************/

笛卡尔树（Treap）

/**************************************/

```
//笛卡尔树储存 pair<key,value>类型，只看 key 值满足二
叉搜索树条件，只看 value 满足堆的条件
struct Treap_Node
{
    Treap_Node *left,*right; //节点的左右子树的指针
    int value,pri;  //节点的值和优先级
    Treap_Node()
    {
        left=NULL;
        right=NULL;
    }
    void Treap_Left_Rotate(Treap_Node *&a) //左旋
节点指针一定要传递引用
    {
        Treap_Node *b=a->right;
        a->right=b->left;
        b->left=a;
        a=b;
    }
    void Treap_Right_Rotate(Treap_Node *&a) //右
旋 节点指针一定要传递引用
    {
        Treap_Node *b=a->left;
        a->left=b->right;
        b->right=a;
        a=b;
    }
    void Treap_Insert(Treap_Node *&P,int
value,int pri) //节点指针一定要传递引用
    {
        if (!P) //找到位置，建立节点
        {
            P=new Treap_Node();
            P->value=value;
            P->pri=pri;//生成随机的修正值
        }
        else if (value <= P->value)
        {
            Treap_Insert(P->left,value,pri);
            if ((P->left->pri)<(P->pri))
                Treap_Right_Rotate(P);//左子节点修
正值小于当前节点修正值，右旋当前节点
        }
```

```
        else
        {
            Treap_Insert(P->right,value,pri);
            if((P->right->pri)<(P->pri))
                Treap_Left_Rotate(P);//右子节点修正值小于当前节点修正值，左旋当前节点
        }
    }
    void Treap_Delete(Treap_Node *&P,int value) //节点指针要传递引用
    {
        if (value==P->value) //找到要删除的节点 对其删除
        {
            if (!P->right||!P->left) //情况一，该节点可以直接被删除
            {
                Treap_Node *t=P;
                if (!P->right)
                    P=P->left;  //用左子节点代替它
                else
                    P=P->right;  //用右子节点代替它
                delete t; //删除该节点
            }
            else //情况二
            {
                if((P->left->pri)<(P->right->pri)) //左子节点修正值较小，右旋
                {
                    Treap_Right_Rotate(P);
                    Treap_Delete(P->right,value);
                }
                else //左子节点修正值较小，左旋
                {
                    Treap_Left_Rotate(P);
                    Treap_Delete(P->left,value);
                }
            }
        }
        else if (value < P->value)
            Treap_Delete(P->left,value);  //在左子树查找要删除的节点
        else
            Treap_Delete(P->right,value); //在右子树查找要删除的节点
    }
};
```

/**************************************/

笛卡尔树_2（Treap）

/**************************************/

```
//不带删除
struct data{
    int v,p;
    data(){}
    data(int _v,int _p){
        v=_v;
        p=_p;
    }
}po[N];
bool comp(data a,data b){
    return a.v<b.v;
}
struct Node{
    int lc,rc;
    int v,p;
    void init(){
        lc=rc=-1;
    }
    void init(int _v,int _p){
        v=_v;
        p=_p;
        lc=rc=-1;
    }
}node[N];
int stk[N],top,cnt,root;
int main(){
    int T,n,cas=0;
    for(scanf("%d",&T);T;T--){
        scanf("%d",&n);
        for(int i=0;i<n;i++){
            scanf("%d%d",&po[i].v,&po[i].p);
        }
        sort(po,po+n,comp);
        node[0].init();
        node[0].p=-1;
        stk[0]=0;
        top=cnt=1;
        for(int i=0;i<n;i++,cnt++){
            node[cnt].init(po[i].v,po[i].p);

            while(node[stk[top-1]].p>node[cnt].p)top--;
                int fa=stk[top-1];
                node[cnt].lc=node[fa].rc;
                node[fa].rc=cnt;
                node[cnt].rc=-1;
                stk[top++]=cnt;
            }
            root=node[0].rc;
        }
        return 0;
}
```

/**************************************/

二叉平衡树（AVL）

/**************************************/

```
typedef struct Node* Tree;
typedef struct Node* Node_t;
typedef int Type;
struct Node
{
    Node_t left;
    Node_t right;
    int height;
    Type data;
    Node(Type x)
    {
        data=x;
        left=NULL;
        right=NULL;
    }
};
int Height(Node_t node)
{
    if(node!=NULL)
        return node->height;
    else
        return 0;
}
Node_t RightRotate(Node_t a)
{
    Node_t b = a->left;
    a->left = b->right;
    b->right = a;
    a->height = max(Height(a->left),
Height(a->right))+1;
    b->height = max(Height(b->left),
Height(b->right))+1;
    return b;
}
Node_t LeftRotate(Node_t a)
{
    Node_t b = a->right;
    a->right = b->left;
    b->left = a;
    a->height = max(Height(a->left),
Height(a->right))+1;
    b->height = max(Height(b->left),
Height(b->right))+1;
    return b;
}
Node_t LeftRightRotate(Node_t a)
{
    a->left = LeftRotate(a->left);
    return RightRotate(a);
}
Node_t RightLeftRotate(Node_t a)
{
    a->right = RightRotate(a->right);
    return LeftRotate(a);
}
Node_t Insert(Type x, Tree t)
{
    if(t == NULL)
    {
        t = new Node(x);
    }
    else if(x < t->data)
    {
        t->left = Insert(x,t->left);
```

```
        if(Height(t->left)-Height(t->right) == 2)
        {
            if(x<(t->left->data))
            {
                t = RightRotate(t);
            }
            else
            {
                t = LeftRightRotate(t);
            }
        }
    }
    else
    {
        t->right = Insert(x,t->right);
        if(Height(t->right) - Height(t->left) ==
2)
        {
            if(x > t->right->data)
            {
                t = LeftRotate(t);
            }
            else
            {
                t = RightLeftRotate(t);
            }
        }
    }

t->height=max(Height(t->left),Height(t->right)
)+1;
    return t;
}
void Rotate(Type x,Tree &t)
{
    if(Height(t->left)-Height(t->right) == 2)
    {
        if(x<(t->left->data))
        {
            t = RightRotate(t);
        }
        else
        {
            t = LeftRightRotate(t);
        }
    }
    else if(Height(t->right) - Height(t->left) ==
2)
    {
        if(x > t->right->data)
        {
            t = LeftRotate(t);
        }
        else
        {
            t = RightLeftRotate(t);
        }
    }
}
Node_t Delete(Type x, Tree &t)
{
    if(t == NULL) return NULL;
    if(t->data == x)
    {
        if(t->right == NULL)
        {
            Node_t temp = t;
            t = t->left;
            delete temp;
        }
        else
        {
            Node_t head = t->right;
            while(head->left)
            {
                head=head->left;
            }
            t->data=head->data; //just copy data
            t->right=Delete(t->data, t->right);

t->height=max(Height(t->left),Height(t->right)
)+1;
        }
        return t;
    }
    else if(t->data<x)
    {
        Delete(x,t->right);
        if(t->right)
            Rotate(x, t->right);
    }
    else
    {
        Delete(x, t->left);
        if(t->left)
            Rotate(x, t->left);
    }
    if(t)
        Rotate(x, t);
}
Tree root;
```

/**************************************/

KD 树（空间距离前 k 近点）

/**************************************/

```
//查找空间中距离某点最近的前 k 个点
const int inf = 1000000000;
const int maxn = 100000+10;
const ll ll_inf = 1ll<<60;
const int maxD = 6;
const int maxK = 20;
int m;
struct point{
    int x[maxD];
    point(){}
    void read(){
        for (int i=0;i<m;++i)
            scanf("%d",&x[i]);
    }
};
vector<point> a;
int t[maxn][maxD];
int divX[maxn];
int n,now,K;

bool cmp(point a, point b) {
    return a.x[now] < b.x[now];
}

void buildTree(int left, int right, point a[]) {
    if (left >= right) return;
    int mid = (left + right) >> 1;
    int minx[maxD],maxx[maxD];
    for (int i=0;i<m;++i){
        minx[i]=inf;
        maxx[i]=-inf;
    }
    for (int i = left; i < right; i++)
        for (int j=0;j<m;++j){
            minx[j]=min(minx[j],a[i].x[j]);
            maxx[j]=max(maxx[j],a[i].x[j]);
        }
    now=0;
    for (int i=1;i<m;++i)
        if (maxx[i]-minx[i]>maxx[now]-minx[now])
now=i;
    divX[mid]=now;
    nth_element(a + left, a + mid, a + right, cmp);

    for (int i=0;i<m;++i) t[mid][i]=a[mid].x[i];

    if (left + 1 == right) return;
    buildTree(left, mid, a);
    buildTree(mid + 1, right, a);
}

long long closestDist[maxK];
int closestNode[maxK][maxD];

void update(ll d,int pt[]){
    for (int i=1;i<=K;++i)
        if (closestDist[i]>d){
            for (int j=K;j>i;--j){
                closestDist[j]=closestDist[j-1];
                for (int k=0;k<m;++k)

closestNode[j][k]=closestNode[j-1][k];
            }
            closestDist[i]=d;
            for (int k=0;k<m;++k)
                closestNode[i][k]=pt[k];
            return;
        }
}
```

```
void findD(int left, int right,const point& p) {
    if (left >= right) return;
    int mid = (left + right) >> 1;
    ll dx[maxD];
    ll d=0;
    for (int i=0;i<m;++i){
       dx[i]=p.x[i]-t[mid][i];
       d+=dx[i]*dx[i];
    }
    //注意能否查自己,d(距离)能否等于0
    update(d,t[mid]);

    if (left + 1 == right) return;
    ll delta = dx[divX[mid]];
    ll delta2 = delta*delta;
    int l1=left,r1=mid;
    int l2=mid+1,r2=right;
    if (delta>0){
        swap(l1,l2);
        swap(r1,r2);
    }
    findD(l1, r1, p);
    if (delta2 < closestDist[K])
        findD(l2, r2, p);
}

void findNearestNeighbour(int n, const point& p)
{
    for (int i=1;i<=K;++i)
        closestDist[i] = ll_inf;
    findD(0, n, p);
}

void print(){
    printf("the closest %d points are:\n",K);
    for (int i=1;i<=K;++i){
        for (int j=0;j<m-1;++j)
            printf("%d ",closestNode[i][j]);
        printf("%d\n",closestNode[i][m-1]);
    }
}

int main(){
    while (scanf("%d%d",&n,&m)==2){
        a.clear();
        point P;
        for (int i=0;i<n;++i){
            P.read();
            a.push_back(P);
        }
        vector<point> b(a);
        buildTree(0,n,&b[0]);

        int q;
        scanf("%d",&q);
        while (q--){
            P.read();
            scanf("%d",&K);
            findNearestNeighbour(n,P);
            print();
        }
    }
    return 0;
}
```

/**************************************/

Splay(动态数组)

/**************************************/

```
typedef struct Splay_Node * Node;
struct Splay_Node {
     Node pre,ch[2];
     int value,lazy,Min,size;//结点价值，lazy标记，子树最小值，子树大小
     bool rev;//是否旋转
     void Init(int _value) {
          pre=ch[0]=ch[1]=NULL;
          Min=value=_value;
          lazy=0;
          size=1;
          rev=false;
     }
};
struct Splay_Tree {
     Splay_Node nstack[MAXN];
     int scnt;
     Node root;
     Splay_Tree(){
          Init();
     }
     void Init() { //Splay初始化由于区间操作需要把将要操作区间旋转到一棵子树上，所以需要额外声明两节点
          scnt=0;
          root=nstack+scnt++;
          root->Init(-inf);
          root->ch[1]=nstack+scnt++;
          root->ch[1]->Init(inf);
     }
     inline int Getsize(Node &x) { //取得x子树大小，主要是解决x=NULL的情况
          return x?x->size:0;
     }
     void Pushdown(Node &x) { //将x标记下移
          if(!x)return;
          if(x->lazy) {
               int w=x->lazy;
               x->value+=w;
               if(x->ch[0]) {
                    x->ch[0]->lazy+=w;
                    x->ch[0]->Min+=w;
               }
               if(x->ch[1]) {
                    x->ch[1]->lazy+=w;
                    x->ch[1]->Min+=w;
               }
               x->lazy=0;
          }
          if(x->rev) {
               Node t=x->ch[0];
               x->ch[0]=x->ch[1];
               x->ch[1]=t;
               x->rev=false;
               if(x->ch[0])
                    x->ch[0]->rev^=1;
               if(x->ch[1])
                    x->ch[1]->rev^=1;
          }
     }
     void Updata(Node &x) { //更新x结点信息
          if(!x)return;
          x->size=1;
          x->Min=x->value;
          if(x->ch[0]) {

x->Min=min(x->Min,x->ch[0]->Min);
               x->size+=x->ch[0]->size;
          }
          if(x->ch[1]) {

x->Min=min(x->Min,x->ch[1]->Min);
               x->size+=x->ch[1]->size;
          }
     }
     void Rotate(Node &x, int d) { // 旋转操作，d=0 表示左旋，d=1 表示右旋
          Node y=x->pre;
          Pushdown(y);
          Pushdown(x);
          //pushdown(x->ch[d]);
          y->ch[!d]=x->ch[d];
          if (x->ch[d]!=NULL) x->ch[d]->pre=y;
          x->pre = y->pre;
          if (y->pre!=NULL) {
               if (y->pre->ch[0]==y)
y->pre->ch[0]=x;
               else y->pre->ch[1]=x;
          }
          x->ch[d]=y,y->pre=x;
          Updata(y);
          if (y == root)//因为是指针，所以root可能被转下去了
               root = x;
     }
     void Splay(Node &x, Node &f) { // Splay 操作，表示把结点x 转到结点f
          for (Pushdown(x); x!=f;) {
               if(x->pre==f) {
                    if(f->ch[0]==x)Rotate(x,1);
                    else Rotate(x,0);
                    break;
               } else {
                    Node y=x->pre,z=y->pre;
```

```
                    if(z->ch[0]==y)
                        if
(y->ch[0]==x)Rotate(y,1),Rotate(x,1);  // 一字形旋转
                        else
Rotate(x,0),Rotate(x,1);  // 之字形旋转
                    else
if(y->ch[1]==x)Rotate(y,0),Rotate(x,0);  // 一字形旋转
                    else Rotate(x, 1), Rotate(x, 0);  // 之字形旋转
                    if(z==f)//转了之后，x 就到了原来 z 的位置，如果 z 就是要到的地方，就可以退出了
                        break;
                }
                Updata(x);
            }
            Updata(x);
        }
        void Select(int k, Node &f) { //把第 k 个结点转到 f 位置
            int tmp;
            Node t;
            for(t=root;;) {
                Pushdown(t);
                tmp=Getsize(t->ch[0]); // 得到 t 左子树的大小
                if (k == tmp + 1) break;  // 得出 t 即为查找结点，退出循环
                if (k <= tmp) // 第 k 个结点在 t 左边，向左走
                    t = t->ch[0];
                else // 否则在右边，而且在右子树中，这个结点不再是第 k 个
                    k -=tmp+1,t=t->ch[1];
            }
            Pushdown(t);
            Splay(t, f); // 执行旋转
        }
        void Insert(int pos,int value) { //插入 value 到 pos 位置之后
            Select(pos+1,root);
            Select(pos+2,root->ch[1]);
            Node t=nstack+scnt++,x=root->ch[1];
            Pushdown(root);
            Pushdown(x);
            t->Init(value);
            t->ch[1]=x;
            x->pre=t;
            root->ch[1]=t;
            t->pre=root;
            Splay(x,root);
        }
        void Add(int a,int b,int d) {//区间[a,b] 加上 c
            Select(a,root);
            Select(b+2,root->ch[1]);
            Node x=root->ch[1]->ch[0];
            Pushdown(x);
            Updata(x);
            x->Min+=d;
            x->lazy+=d;
            Splay(x,root);
        }
        void Reverse(int a,int b) {//区间[a,b]翻转
            Select(a,root);
            Select(b+2,root->ch[1]);
            root->ch[1]->ch[0]->rev^=1;
            Node x=root->ch[1]->ch[0];
            Splay(x,root);
        }
        void Revolve(int a,int b,int t) {//区间[a,b] 循环移位 t 次
            Node p1,p2;
            Select(a,root);
            Select(b+2,root->ch[1]);
            Select(b+1-t,root->ch[1]->ch[0]);
            p1=root->ch[1]->ch[0];
            Pushdown(p1);
            p2=p1->ch[1];
            p1->ch[1]=NULL;
            Select(a+1,root->ch[1]->ch[0]);
            p1=root->ch[1]->ch[0];
            Pushdown(p1);
            p1->ch[0]=p2;
            p2->pre=p1;
            Splay(p2,root);
        }
        int Getmin(int a,int b) {//得到区间[a,b]最小值
            Select(a,root);
            Select(b+2,root->ch[1]);
            Node x=root->ch[1];
            Pushdown(x);
            x=x->ch[0];
            Pushdown(x);
            Updata(x);
            return x->Min;
        }
        void Erase(int pos) {//删除第 pos 个元素
            Select(pos,root);
            Select(pos+2,root->ch[1]);
            Pushdown(root->ch[1]);
            root->ch[1]->ch[0]=NULL;
            Node x=root->ch[1];
            Splay(x,root);
        }
        void Cut(int a,int b,int c){//剪切区间[a,b],然后贴在新生成序列的 c 位置后面
            Select(a,root);
            Select(b+2,root->ch[1]);
            Node x=root->ch[1],y;
            Pushdown(root);
            Pushdown(x);
            y=x->ch[0];
            y->pre=0;
            x->ch[0]=0;
            Select(c+1,root);
            Select(c+2,root->ch[1]);
            x=root->ch[1];
            Pushdown(root);
            Pushdown(x);
            x->ch[0]=y;
            y->pre=x;
            Splay(y,root);
        }
        void Print(){//输出序列
            bool first=true;
            Node lc,rc,now=root;
            while(now){
                Pushdown(now);
                lc=now->ch[0];
                rc=now->ch[1];
                if(lc&&!(lc->fg)){
                    now=lc;
                }
                else if(!(now->fg)){
                    now->fg=true;
                    if(checkend(now->value)){
                        continue;
                    }
                    if(!first)putchar(' ');
                    else first=false;
                    printf("%d",now->value);
                }
                else if(rc&&!(rc->fg)){
                    now=rc;
                }
                else{
                    now=now->pre;
                }
            }
            printf("\n");
        }
};
Splay_Tree S;
/**************************************/
```

Dancing links(精确覆盖)

```
/**************************************/
int
cnt,L[NUM],R[NUM],S[NUM],D[NUM],U[NUM],C[NUM],
O[NUM],H[NUM],X[NUM];
/*
    NUM:最大结点数
    U,D,L,R: 上下左右结点
    C: 列的头指针位置
    O: 储存答案
    X: 与 O 配合代表第几行（X[O[i]]）
```

```
    通过 link（r,c）加点，dfs（0）运算，行列从 1 开始算
*/
void remove(const int &c)
{
    L[R[c]]=L[c];
    R[L[c]]=R[c];
    //如果这里直接返回就是求可重叠覆盖
    for(int i=D[c];i!=c;i=D[i])
    {
        for(int j=R[i];j!=i;j=R[j])
        {
            U[D[j]]=U[j];
            D[U[j]]=D[j];
            --S[C[j]];
        }
    }
}
void resume(const int &c)
{
    L[R[c]]=c;
    R[L[c]]=c;
    //如果这里直接返回就是求可重叠覆盖
    for(int i=U[c];i!=c;i=U[i])
    {
        for(int j=L[i];j!=i;j=L[j])
        {
            ++S[C[j]];
            U[D[j]]=j;
            D[U[j]]=j;
        }
    }
}
bool dfs(const int &k)
{
    if(!R[0])
    {
        return  true;
    }
    int s(inf),c;
    for(int t=R[0];t!=0;t=R[t])
    {
        if(S[t]<s)
        {
            s=S[t];
            c=t;
        }
    }
    remove(c);
    for(int i=D[c];i!=c;i=D[i])
    {
        O[k]=i;
        for(int j=R[i];j!=i;j=R[j])
        {
            remove(C[j]);
        }
        if(dfs(k+1))
        {
            return  true;
        }
        for(int j=L[i];j!=i;j=L[j])
        {
            resume(C[j]);
        }
    }
    resume(c);
    return false;
}
void build(int r,int c)
{
    for(int i=0;i<=c;i++)
    {
        U[i]=D[i]=i;
        L[i+1]=i;
        R[i]=i+1;
        S[i]=0;
        C[i]=i;
    }
    R[cnt=c]=0;
    while(r)
        H[r--]=-1;
}
void link(int r,int c)
{
    ++S[C[++cnt]=c];
    X[cnt]=r;
    D[cnt]=D[c];
    U[D[c]]=cnt;
    U[cnt]=c;
    D[c]=cnt;
    if(H[r]<0)
        H[r]=L[cnt]=R[cnt]=cnt;
    else
    {
        R[cnt]=R[H[r]];
        L[R[H[r]]]=cnt;
        L[cnt]=H[r];
        R[H[r]]=cnt;
    }
}
```

/**************************************/

块状链表

/**************************************/

```
int bs,top;
struct Block
{
    int size,next;
    char s[3000];
    void push_back(char ch)
    {
        s[size++]=ch;
    }
    void insert(int pos,char ch)
    {
        for(int i=size++;i>pos;i--)
            s[i]=s[i-1];
        s[pos]=ch;
    }
}block[3000];
void update(int x)
{
    if(block[x].size<bs*2)
        return;
    ++top;
    int i,j,k=block[x].size;
    for(i=bs,j=0;i<k;i++,j++)
        block[top].s[j]=block[x].s[i];
    block[top].size=j;
    block[x].size=bs;
    block[top].next=block[x].next;
    block[x].next=top;
}
int main()
{
    char s[maxn];
    gets(s);
    int len=strlen(s),m;
    scanf("%d",&m);
    bs=sqrt((double)(len+m))+1;
    top=0;
    block[0].size=0;
    for(int i=0;i<len;i++)
    {
        if(block[top].size==bs)
        {
            block[top].next=top+1;
            block[++top].size=0;
        }
        block[top].push_back(s[i]);
    }
    block[top].next=-1;
    while(m--)
    {
        char op,ch;
        int pos,k;
        scanf(" %c",&op);
        if(op=='Q')
        {
            scanf("%d",&pos);
            k=0;
            while(pos>block[k].size)

pos-=block[k].size,k=block[k].next;
            printf("%c\n",block[k].s[pos-1]);
        }
        else
        {
            scanf(" %c %d",&ch,&pos);
            k=0;

while(block[k].next!=-1&&pos>block[k].size)

pos-=block[k].size,k=block[k].next;
```

```
block[k].insert(min(pos-1,block[k].size+1),ch)
;
            update(k);
        }
    }
    return 0;
}
```

/**************************************/

KMP

/**************************************/

```
int kmp(char* st1, char* st2)
{
    int len1,len2;
    len1=strlen(st1), len2=strlen(st2);
    int i,j=0,t=next[0]=-1;
    while (j<len2)
    {
        if (t<0 || st2[j]==st2[t])
next[++j]=++t;
        else t=next[t];
    }
    for (i=j=0; i<len1 && j<len2; )
    {
        if (j<0 || st1[i]==st2[j]) i++,j++;
        else j=next[j];
    }
    return i-j;
}
void extendkmp(char* st1,char* st2)
{
    int len1,len2;
    len1=strlen(st1), len2=strlen(st2);
    int i,j,k,len,L;
    j=0;
    while (st2[j+1]==st2[j] && j+1<len2) j++;
    next[1]=j, k=1;
    for (i=2; i<len2; i++)
    {
        len=k+next[k], L=next[i-k];
        if (len>L+i) next[i]=L;
        else
        {
            j=len-i>0 ? len-i : 0;
            while (st2[i+j]==st2[j] &&
i+j<len2) j++;
            next[i]=j, k=i;
        }
    }
    j=0;
    while (st1[j]==st2[j] && j<len1 && j<len2)
j++;
    ext[0]=j, k=0;
    for (i=1; i<len1; i++)
    {
        len=k+ext[k], L=next[i-k];
        if (len>L+i) ext[i]=L;
        else
        {
            j=len-i>0 ? len-i : 0;
            while (st1[i+j]==st2[j] &&
i+j<len1 && j<len2) j++;
            ext[i]=j, k=i;
        }
    }
}
```

/**********/

其它

/**********/

/**************************************/

C++库（不常用）

/**************************************/

1.rotate(begin,mid,end)

循环移位，将 mid 至 end 之前的所有数据循环移到前面。

2.next_permutation(begin,end)

得到原序列的下一序列，如果没有下一序列，则返回空指针

3.set<T>

lower_bound(val) :返回 set 中大于等于 val 的位置

upper_bound(val):返回 set 中大于 val 的位置

4.<cctype>

isdigit()/isupper()/islower 是否数字/大写/小写

ispunct()是否标点符号

isalpha()是否字母

isgraph()是否是可打印字符

5.nth_element(begin,mid,end)/nth_element(begin,mid,end,comp)

将序列[begin，end)从 mid 处断开，使得 mid 左边的都比 mid 小，右边都比 mid 大(或 comp 函数左边均为 true，右边均为 false)

6. sscanf(s,"%d.%d",&a,&b)

将字符串 s 当输入设备读入数据。

7. sprintf(s,"%d.%d",a,b);

将字符串 s 当输出设备输出数据。

/**************************************/

操作

/**************************************/

二维指针声明

```
int **ps;
ps=(int **)new int *[si];
for(i=0;i<si;i++)
   ps[i]=new int[si];
```

将最右侧 0 位改为 1 位： x | (x+1)

二进制补码运算公式：

```
x==y:    ~(x-y|y-x)
x!=y:    x-y|y-x
x< y:    (x-y)^((x^y)&((x-y)^x))
x<=y:    (x|~y)&((x^y)|~(y-x))
x< y:    (~x&y)|((~x|y)&(x-y))//无符号 x,y 比较
x<=y:    (~x|y)&((x^y)|~(y-x))//无符号 x,y 比较
```

不使用第三方交换 x,y:

```
x ^= y ; y ^= x ; x ^= y ;
```

双值交换： //常规编码为 x = (x==a) ? b :a ;

```
x = a^b^x ;
```

下舍入到 2 的 k 次方的倍数：

```
x & ((-1)<<k)
```

上舍入：

```
t = (1<<k)-1 ; x = (x+t)&~t ;
```

位计数,统计 1 位的数量：

```
int pop(unsigned x)
{
x = x-((x>>1)&0x55555555) ;
x = (x&0x33333333) + ((x>>2) & 0x33333333 ) ;
x = (x+(x>>4)) & 0x0f0f0f0f ;
x = x + (x>>8) ;
x = x + (x>>16) ;
return x & 0x0000003f ;
}
```

位反转：

```
unsigned rev(unsigned x)
{
x = (x & 0x55555555) << 1 | (x>>1) & 0x55555555 ;
x = (x & 0x33333333) << 2 | (x>>2) & 0x33333333 ;
x = (x & 0x0f0f0f0f) << 4 | (x>>4) & 0x0f0f0f0f ;
x = (x<<24) | ((x&0xff00)<<8) | ((x>>8) & 0xff00)
| (x>>24) ;
return x ;
}
```

找出最左 0 字节的位置：

```
int zbytel( unsigned x )
{
static cahr table[16] = { 4,3,2,2, 1,1,1,1, 0,0,0,0,
0,0,0,0 } ;
unsigned y ;
y = (x&0x7f7f7f7f) + 0x7f7f7f7f ;
y = ~(y|x|0x7f7f7f7f) ;
return table[y*0x00204081 >> 28] ;//乘法可用移位和
加完成
```

找出最右 1 字节的位置：

```
int lowbit(int x)
{
return x&(-x);
}
```

重载优先队列比较级：

```
struct comp
{
    bool operator()(int a,int b)
    {
        return a>b;
    }
};
priority_queue<int,vector<int>,comp> Q;
```

```
/*C++扩栈*/#pragma comment(linker,
"/STACK:102400000,102400000")
inline int rd(){//输入外挂
    int num=0,neg=0;char in;
    while(((in=getchar()) > '9' || in<'0') &&
in!='-') ;
    if(in=='-')neg=true;
    else num=in-'0';

while(in=getchar(),in>='0'&&in<='9')num=num*10
+in-'0';
    return (neg?-1:1)*num;
}
```

/**************************************/

关于 G++与 C++的输入输出

/**************************************/

速度：

G++用 putchar 与 puts 更快，

putchar(ch)　相当于 printf（"%c", ch）。

puts(str)　相当于 printf（"%s\n",str）。

C++用 printf()更快

浮点数：

G++读入数据时用%lf，输出时用%f。

C++读入与输出都用%lf。

关于 scanf()：

%*[]　：　跳过[]里面的东西

%[^c]　：　读入字符串直到遇到字母 c，但是不读入 c

%[a-z]:　读入字符串，直到没遇到 a-z 中的字符为止

/**************************************/

JAVA 汇总

/**************************************/

1. 输入：

格式为：Scanner cin = new Scanner (new BufferedInputStream(System.in));

例程：

```
import java.io.*;
import java.math.*;
import java.util.*;
import java.text.*;
public class Main
{
public static void main(String[] args)
{
Scanner cin = new Scanner (new
BufferedInputStream(System.in));
int a; double b; BigInteger c; String st;
a = cin.nextInt(); b = cin.nextDouble(); c =
cin.nextBigInteger(); d = cin.nextLine(); // 每
种类型都有相应的输入函数.
}
}
```

2. 输出

函数：System.out.print(); System.out.println();

System.out.printf();

System.out.print(); // cout << …;

System.out.println(); // cout << … << endl;

System.out.printf(); // 与 C 中的 printf 用法类似.

例程：

```
import java.io.*;
import java.math.*;
import java.util.*;
import java.text.*;
public class Main
{
public static void main(String[] args)
{
Scanner cin = new Scanner (new
BufferedInputStream(System.in));
int a; double b;
a = 12345; b = 1.234567;
System.out.println(a + " " + b);
System.out.printf("%d %10.5f\n", a, b); // 输入 b
为字宽为 10，右对齐，保留小数点后 5 位，四舍五入.
}
}
```

规格化的输出：

函数：

```
// 这里 0 指一位数字，#指除 0 以外的数字 (如果是 0，则不显
示),四舍五入.
DecimalFormat fd = new DecimalFormat("#.00#");
DecimalFormat gd = new DecimalFormat("0.000");
System.out.println("x =" + fd.format(x));
System.out.println("x =" + gd.format(x));
```

3. 字符串处理

java 中字符串 String 是不可以修改的，要修改只能转换为字符数组.

例程：

```
import java.io.*;
import java.math.*;
import java.util.*;
import java.text.*;
public class Main
{
public static void main(String[] args)
{
int i;
Scanner cin = new Scanner (new
BufferedInputStream(System.in));
String st = "abcdefg";
System.out.println(st.charAt(0)); // st.charAt(i)
就相当于 st[i].
char [] ch;
ch = st.toCharArray(); // 字符串转换为字符数组.
for (i = 0; i < ch.length; i++) ch[i] += 1;
System.out.println(ch); // 输入为"bcdefgh".
if (st.startsWith("a")) // 如果字符串以'0'开头.
{
st = st.substring(1); // 则从第 1 位开始 copy (开头为
第 0 位).
}
http://hi.baidu.com/lewutian
}
}
```

4. 高精度

BigInteger 和 BigDecimal 可以说是 acmer 选择 java 的首要原因。

函数：add, subtract, divide, mod, compareTo 等，其中加减乘除模都要求是 BigInteger(BigDecimal)和 BigInteger(BigDecimal)之间的运算，所以需要把 int(double)类型转换为 BigInteger(BigDecimal)，用函数 BigInteger.valueOf().

例程：

```
import java.io.*;
import java.math.*;
import java.util.*;
import java.text.*;
public class Main
{
public static void main(String[] args)
{
Scanner cin = new Scanner (new
BufferedInputStream(System.in));
int a = 123, b = 456, c = 7890;
BigInteger x, y, z, ans;
x = BigInteger.valueOf(a); y =
BigInteger.valueOf(b); z =
BigInteger.valueOf(c);
ans = x.add(y); System.out.println(ans);
ans = z.divide(y); System.out.println(ans);
ans = x.mod(z); System.out.println(ans);
if (ans.compareTo(x) == 0)
System.out.println("1");
}
}
```

5. 进制转换

java 很强大的一个功能。

函数：

```
String st = Integer.toString(num, base); // 把 num
当做 10 进制的数转成 base 进制的 st (base <= 35).
int num = Integer.parseInt(st, base); // 把 st 当
做 base 进制，转成 10 进制的 int (parseInt 有两个参数,第
一个为要转的字符串,第二个为说明是什么进制).
BigInter m = new BigInteger(st, base); // st 是字
符串，base 是 st 的进制.
```

6. 排序

函数：Arrays.sort();至于怎么排序结构体，像 C++里写个 cmp 的方法，在 java 还不太清楚，希望有人指点下～～
例程：

```
import java.io.*;
import java.math.*;
import java.util.*;
import java.text.*;
public class Main
{
public static void main(String[] args)
{
Scanner cin = new Scanner (new
BufferedInputStream(System.in));
int n = cin.nextInt();
int a[] = new int [n];
for (int i = 0; i < n; i++) a[i] = cin.nextInt();
Arrays.sort(a);
for (int i = 0; i < n; i++) System.out.print(a[i]
+ " ");
}
}
```

7.HashMap

```
HashMap<K,V> hash=new HashMap<K,V>;
Iterator<K,V> it=hash.entrySet().iterator();
while(it.hasnext())
{
Map.Entry entry=(Map.Entry)it.next();
Object val = entry.getValue();
Object key = entry.getKey();
```

/**************************************/

DP 优化

/**************************************/

```
//四边形不等式优化
//对于任意的a <= b <= c <= d 都满足 cost[a][c] +
cost[b][d] <= cost[a][d] + cost[b][c]
```

第一类：`dp[i][j] = min(dp[i][k] + dp[k + 1][j] + cost[i][j]);`

第二类：`dp[i][j] = min(dp[i][j - 1] + cost[k + 1][i]);//i 个人分成 j 组`

```
int main () {
     int n,m,i,j,k,ca=0

memset(dp,127,sizeof(dp));
     for (i=1; i<=n; i++) {
          dp[i][1]=sum[1][i];
          s[i][1]=1;
     }
     for (j=2; j<=m; j++) {
          s[n+1][j]=n;
          for (i=n; i>=j; i--) {
               for (k=s[i][j-1]; k<=s[i+1][j];
k++)
                    if
(dp[i][j]>dp[k][j-1]+sum[k+1][i]) {

dp[i][j]=dp[k][j-1]+sum[k+1][i];
                         s[i][j]=k;
                    }
          }
     }


}
int y (int j1,int j2) {
     return
(dp[j2]-dp[j1]+sqr(sum[j2])-sqr(sum[j1]));
}
int x (int j1,int j2) {
     return 2*(sum[j2]-sum[j1]);
}
int main () {
     dp[0]=0;
     q[0]=l=r=0;
     for (i=1; i<=n; i++) {
          while
(l<r&&y(q[l],q[l+1])<=sum[i]*x(q[l],q[l+1]))
               l++;

dp[i]=dp[q[l]]+m+sqr(sum[i]-sum[q[l]]);
          while
(l<r&&y(q[r],i)*x(q[r-1],q[r])<=y(q[r-1],q[r])
*x(q[r],i))
               r--;
          q[++r]=i;
     }
}
//斜率优化 DP
//dp[i]=min(dp[k]+b[k]*a[i])
//k<j 时，用 j 取代 k 需要满足
//   dp[j]+b[j]*a[i]<=dp[k]+b[k]*a[i]
//<=>(dp[j]-dp[k])/(b[j]-b[k])>=-a[i]
ll dp[N],a[N],b[N];
bool CompTwo(int j,int k,ll sum) { //k<j 时，
用决策 j 取代 k 的合理性
     return
(double)(dp[j]-dp[k])/(double)(b[j]-b[k])>=sum
;
}
bool CompThree(int i,int j,int k) { //i<j<k
时，k 是否比 j 更优
     return
(double)(dp[j]-dp[i])/(double)(b[j]-b[i])<=(do
uble)(dp[k]-dp[j])/(double)(b[k]-b[j]);
}
int ls[N],front,rear;
int main() {
     int n;
     while(scanf("%d",&n)!=EOF) {
          for(int i=1; i<=n; i++)
               scanf("%I64d",&a[i]);
          for(int i=1; i<=n; i++)
               scanf("%I64d",&b[i]);
          memset(dp,0,sizeof(dp));
          front=rear=0;
          dp[1]=0;
          ls[rear++]=1;
          for(int i=2; i<=n; i++) {

while(rear-front>1&&CompTwo(ls[front+1],ls[fro
nt],-a[i]))front++;
               int k=ls[front];
               dp[i]=dp[k]+b[k]*a[i];

while(rear-front>1&&CompThree(ls[rear-2],ls[re
ar-1],i))rear--;
               ls[rear++]=i;
          }
          printf("%I64d\n",dp[n]);
     }
     return 0;
}
```

/**************************************/

前缀等于后缀的子串个数

/**************************************/

```
int next[400005];
char s[400005];
void dfs(int j) {
     if(j==0)
          return ;
     dfs(next[j]);
     printf("%d ",j);
}
int main() {
     s[0]='\0';
     while(scanf(" %s",s)!=EOF) {
          getnext();//KMP
          int i,len=strlen(s);
          dfs(next[len]);
          printf("%d\n",len);
          s[0]='\0';
     }
     return 0;
}
```

/**************************************/

最长回文子串

/**************************************/

```
char s[maxn];
char st[maxn];
int p[maxn];
void manacher(int n) {
     int mx=0,id;
     for (int i=1; i<n; i++) {
          if (mx>i) p[i]=min(p[2*id-i],mx-i);
          else p[i]=1;
          for (; st[i+p[i]]==st[i-p[i]];
p[i]++) ;
```

```
            if (p[i]+i>mx) {
                mx=p[i]+i;
                id=i;
            }
        }
}
int main() {
    while (scanf("%s",s)!=EOF) {
        st[0]='$';
        st[1]='#';
        int len=strlen(s);
        for (int i=0; i<len; i++) {
            st[2*i+2]=s[i];
            st[2*i+3]='#';
        }
        st[2*len+2]='\0';
        int n=strlen(st);
        manacher(n);
        int ans=0;
        for (int i=1; i<n; i++)
            if (p[i]-1>ans) ans=p[i]-1;
        printf("%d\n",ans);
    }
    return 0;
}
```

/**************************************/

背包问题

/**************************************/

```
//每件物品只能使用一次
void onezeropack(int v,int c) {
    int j;
    for(j=val; j>=v; j--) {
        f[j]=max(f[j-v]+c,f[j]);
    }

}
//每件物品可以无限使用
void completepack(int v,int c) {
    int j;
    for(j=v; j<=val; j++) {
        f[j]=max(f[j-v]+c,f[j]);
    }
}
//每件物品有限次使用
void multiplepack(int v,int c,int num) {
    if(c*num>=val) {
        completepack(v,c);
        return;
    }
    int k=1;
    while(k<num) {
        onezeropack(k*v,k*c);
        num=num-k;
        k=k*2;
    }
    onezeropack(num*v,num*c);
}
```

/**************************************/

基数排序

/**************************************/

```
const int N=1000;
int maxbit(int data[],int n) { //辅助函数，求最大位数
    int d=1,p=10;
    for(int i=0; i<n; i++) {
        while(data[i]>=p) {
            p*=10;
            d++;
        }
    }
    return d;
}
void radixsort(int data[],int n) {
    int d=maxbit(data,n);
    int tmp[N],count[10];
    int radix=1,k,i,j;
    for(i=1; i<=d; i++) {
        for(j=0; j<10; j++) {
            count[j]=0;
        }
        for(j=0; j<n; j++) {
            k=(data[j]/radix)%10;
            count[k]++;
        }
        for(j=1; j<10; j++) {
            count[j]+=count[j-1];
        }
        for(j=n-1; j>=0; j--) {
            k=(data[j]/radix)%10;
            count[k]--;
            tmp[count[k]]=data[j];
        }
        for(j=0; j<n; j++) {
            data[j]=tmp[j];
        }
        radix*=10;
    }
}
```

/**************************************/

字符环的最小表示法

/**************************************/

说把一个长为 len 的字符串围成一个圈，然后从任意一个字符作为起点顺时针转，都会产生一个新的长为 len 字符串，现在要求所有的可以产生的字符串中字典序最小的那个。下面这个函数就是解决这个问题的，返回值即为从原串中这个位置起产生的串就是字典序最小的。

```
int MinimumRepresentation(char *s, int l) {
    int i = 0, j = 1, k = 0, t;
    while (i < l && j < l && k < l) {
        t = s[(i + k)%l] - s[(j + k)%l];
        if (!t) ++ k;
        else {
            if (t > 0) i = i + k + 1;
            else j = j + k + 1;
            if (i == j) ++j;
            k = 0;
        }
    }
    return min(i,j);
}
```

/**************************************/

最小循环矩阵

/**************************************/

```
//字符矩阵的最小子矩阵使得其他都由这子矩阵循环而来
char row[100005][80],col[80][100005];
int next[100005];
int main() {
    int r,c,i,j,rr,cc;
    while(scanf("%d %d ",&r,&c)!=EOF) {
        for(i=0; i<r; i++) {
            for(j=0; j<c; j++) {
                row[i][j]=getchar();
                col[j][i]=row[i][j];
            }
            getchar();
            row[i][c]='\0';
        }
        for(i=0; i<c; i++)
            col[i][r]='\0';
        rr=cc=1;
        for(i=0; i<r; i++) {
            getnext(row[i]);//KMP
            if(next[c]==0)rr=c;
            else rr=lcm(rr,c-1-next[c-1]);
            if(rr>=c) {
                rr=c;
                break;
            }
        }
        for(i=0; i<c; i++) {
            getnext(col[i]);//KMP
            if(next[r]==0)cc=r;
            else cc=lcm(cc,r-1-next[r-1]);
            if(cc>=r) {
                cc=r;
                break;
            }
        }
        printf("%d\n",rr*cc);
    }
    return 0;
}
```

/**************************************/

最短非子序列长度

```
/**************************************/
int main() {
    int n,k,i;
    while(scanf("%d%d",&n,&k)!=EOF) {
        int t,a[101000],cnt=1;
        for(i=0; i<n; i++)
            scanf("%d",&a[i]);
        bool mark[10100];
        memset(mark,false,sizeof(mark));
        for(i=0,t=k; i<n; i++) {
            if(!mark[a[i]]) {
                t--;
                if(t==0) {
                    t=k;
                    cnt++;

memset(mark,false,sizeof(mark));
                } else
                    mark[a[i]]=true;
            }
        }
        printf("%d\n",cnt);
    }
    return 0;
}
```

最长下降子序列长度与个数

```
/**************************************/
int maxnum[5005],maxlen[5005],a[5005];
int main() {
    int n;
    while(scanf("%d",&n)!=EOF) {
        int i,j,k;
        for(i=1; i<=n; i++) {
            scanf("%d",&a[i]);
            maxnum[i]=0;
            maxlen[i]=1;
        }
        for(i=1; i<=n; i++) {
            for(j=1; j<i; j++) {
                if(a[j]>a[i]) {

maxlen[i]=max(maxlen[j]+1,maxlen[i]);
                }
            }
        }
        for(i=1; i<=n; i++)
            if(maxlen[i]==1)
                maxnum[i]=1;
        for(i=1; i<=n; i++) {
            for(j=i-1; j>=1; j--) {
                if(a[j]>a[i]) {

if(maxlen[j]+1==maxlen[i]) {

maxnum[i]+=maxnum[j];
                }
            } else if(a[j]==a[i]) {
                if(maxlen[i]==1)
                    maxnum[i]=0;
                break;
            }
        }
        int mmax=0,ans=0;
        for(i=1; i<=n; i++)
            mmax=max(mmax,maxlen[i]);
        for(i=1; i<=n; i++)
            if(mmax==maxlen[i])
                ans+=maxnum[i];
        printf("%d %d\n",mmax,ans);
    }
    return 0;
}
```

最少偏序集个数

```
/**************************************/
//满足 xi<xj&&yi<yj 的点可合成一个集合，求最少集合个数
struct data {
    int x,y;
}po[20005];
bool comp(const data &a,const data &b) {
    if(a.x!=b.x)return a.x<b.x;
    else return b.y<a.y;//如果相等时也可重叠则交换b.y,a.y
}
int dp[20005],ans;//dp 递减
int main() {
    int num;
    for(scanf("%d",&num); num; num--) {
        int i,n,ll,rr,mid;
        scanf("%d",&n);
        for(i=0; i<n; i++)
            scanf("%d%d",&po[i].x,&po[i].y);
        sort(po,po+n,comp);
        dp[ans=1]=po[0].y;
        for(i=1; i<n; i++) {
            ll=1;
            rr=ans;
            while(ll<rr) {
                mid=(ll+rr)/2;
                if(dp[mid]>=po[i].y)//相等则改为>
                    ll=mid+1;
                else
                    rr=mid;
            }
            if(dp[rr]>=po[i].y)//相等则改为>
                dp[++ans]=po[i].y;
            else
                dp[rr]=po[i].y;
        }
        printf("%d\n",ans);
    }
    return 0;
}
```

N 皇后问题构造方法

```
/**************************************/
/*
一、当n mod 6 != 2 或 n mod 6 != 3时:

[2,4,6,8,...,n],[1,3,5,7,...,n-1]          (n为偶数)

[2,4,6,8,...,n-1],[1,3,5,7,...,n  ]        (n为奇数)

二、当n mod 6 == 2 或 n mod 6 == 3时

(当n为偶数,k=n/2; 当n为奇数,k=(n-1)/2)

[k,k+2,k+4,...,n],[2,4,...,k-2],[k+3,k+5,...,n-1],[1,3,5,...,k+1]          (k为偶数,n为偶数)

[k,k+2,k+4,...,n-1],[2,4,...,k-2],[k+3,k+5,...,n-2],[1,3,5,...,k+1],[n]      (k为偶数,n为奇数)

[k,k+2,k+4,...,n-1],[1,3,5,...,k-2],[k+3,...,n],[2,4,...,k+1]               (k为奇数,n为偶数)

[k,k+2,k+4,...,n-2],[1,3,5,...,k-2],[k+3,...,n-1],[2,4,...,k+1],[n  ]      (k为奇数,n为奇数)
*/

void work(int n) {
    int i,k;
    if(n%6!=2&&n%6!=3) {
        for(i=2; i<=n; i+=2)printf("%d ",i);
        for(i=1; i<n-1; i+=2)printf("%d ",i);
        printf("%d\n",i);
    } else {
        k=n>>1;
        if(k&1) {
            if(n&1) {
                for(i=k; i<n;
i+=2)printf("%d ",i);
                for(i=1; i<k;
i+=2)printf("%d ",i);
                for(i=k+3; i<n;
i+=2)printf("%d ",i);
                for(i=2; i<=k+1;
i+=2)printf("%d ",i);
                printf("%d\n",n);
            } else {
```

```
                    for(i=k; i<n;
i+=2)printf("%d ",i);
                    for(i=1; i<k;
i+=2)printf("%d ",i);
                    for(i=k+3; i<=n;
i+=2)printf("%d ",i);
                    for(i=2; i<k+1;
i+=2)printf("%d ",i);
                    printf("%d\n",i);
               }
          } else {
               if(n&1) {
                    for(i=k; i<=n;
i+=2)printf("%d ",i);
                    for(i=2; i<k;
i+=2)printf("%d ",i);
                    for(i=k+3; i<n;
i+=2)printf("%d ",i);
                    for(i=1; i<=k+1;
i+=2)printf("%d ",i);
                    printf("%d\n",n);
               } else {
                    for(i=k; i<=n;
i+=2)printf("%d ",i);
                    for(i=2; i<k;
i+=2)printf("%d ",i);
                    for(i=k+3; i<n;
i+=2)printf("%d ",i);
                    for(i=1; i<k+1;
i+=2)printf("%d ",i);
                    printf("%d\n",i);
               }
          }
     }
}
```

/**************************************/

N*M 数码有解判定

/**************************************/

```
/*
首先将矩阵存入 n*m 的一维数组，求去掉 0 以后的逆序数。
1.左右移动一次不改变逆序数奇偶性，
2.上下移动一次时：
(1) 如果列数为奇数，逆序数奇偶性不变
(2) 如果列数为偶数，逆序数奇偶性改变一次，此时要统计始态和终态 0 的行数差的绝对值，若为偶数则始态和终态逆序数奇偶性相同，否则相反
*/
int a[N];
int main() {
     int n,m;
     while(scanf("%d%d",&n,&m),n||m) {
          int x,y,t,s=0,nu=0;
          for(int i=1; i<=n; i++)
               for(int j=1; j<=m; j++) {
                    scanf("%d",&t);
                    if(t==0)x=i,y=j;
                    else a[nu++]=t;
               }
          memset(ar,0,sizeof(ar));//树状数组
          for(int i=nu-1; i>=0; i--) {
               s+=sum(a[i]-1);
               add(a[i],1);
          }
          if(m&1)
               if(s&1)puts("NO");
               else   puts("YES");
          else if(((n-x)^s)&1) puts("NO");
          else                 puts("YES");
     }
     return 0;
}
```

/**************************************/

堆排序最坏情况构造

/**************************************/

```
int a[50005],n;
void up(int i) {
     int t=a[i];
     while(i>1&&t>a[i/2]) {
          a[i]=a[i/2];
          i/=2;
     }
     a[i]=t;
}
void insert(int v) {
     a[++n]=v;
     up(n);
}
int main() {
     int num;
     while(scanf("%d",&num)!=EOF) {
          n=0;
          for(int i=2; i<=num; i++)
               insert(i);
          insert(1);
          for(int i=1; i<=num; i++)

printf("%d%c",a[i],(i==num)?'\n':' ');
     }
     return 0;
}
```