Code Library

Himemiya Nanao @ Perfect Freeze

July 29, 2013

Contents

1 data structure

1.1 atlantis

```
#include<cstdio>
#include<algorithm>
#include<map>

#define MAXX 111
#define inf 333
#define MAX inf*5

int mid[MAX],cnt[MAX];
double len[MAX];

int n,i,cas;
double x1,x2,y1,y2;
double ans;
std::map<double,int>map;
std::map<double,int>::iterator it;
double rmap[inf];

void make(int id,int l,int r)
{
   mid[id]=(l+r)>>1;
   if(l!=r)
   {
      make(id<<1,l,mid[id]);
      make(id<<1|1,mid[id]+1,r);
   }
}

void update(int id,int ll,int rr,int l,int r,int val)
{
   if(ll==l && rr==r)
   {
      cnt[id]+=val;
      if(cnt[id])
         len[id]=rmap[r]-rmap[l-1];
      else
         if(l!=r)
            len[id]=len[id<<1]+len[id<<1|1];
         else
            len[id]=0;
      return;
   }
   if(mid[id]>=r)
      update(id<<1,ll,mid[id],l,r,val);
   else
      if(mid[id]<l)
         update(id<<1|1,mid[id]+1,rr,l,r,val);
      else
      {
         update(id<<1,ll,mid[id],l,mid[id],val);
         update(id<<1|1,mid[id]+1,rr,mid[id]+1,r,val);
      }
   if(!cnt[id])
      len[id]=len[id<<1]+len[id<<1|1];
}

struct node
{
   double l,r,h;
   char f;
   inline bool operator<(const node &a)const
   {
      return h<a.h;
   }
   inline void print()
   {
      printf("%lf %lf %lf %d\n",l,r,h,f);
   }
}ln[inf];

int main()
{
   make(1,1,inf);
   while(scanf("%d",&n),n)
   {
      n<<=1;
      map.clear();
      for(i=0;i<n;++i)
      {
         scanf("%lf%lf%lf%lf",&x1,&y1,&x2,&y2);
         if(x1>x2)
            std::swap(x1,x2);
         if(y1>y2)
            std::swap(y1,y2);
         ln[i].l=x1;
         ln[i].r=x2;
         ln[i].h=y1;
         ln[i].f=1;
         ln[++i].l=x1;
         ln[i].r=x2;
         ln[i].h=y2;
         ln[i].f=-1;
         map[x1]=1;
         map[x2]=1;
      }
      i=1;
      for(it=map.begin();it!=map.end();++it,++i)
      {
         it->second=i;
         rmap[i]=it->first;
      }
      std::sort(ln,ln+n);
      ans=0;
      update(1,1,inf,map[ln[0].l]+1,map[ln[0].r],ln[0].f);
      for(i=1;i<n;++i)
      {
         ans+=len[1]*(ln[i].h-ln[i-1].h);
         update(1,1,inf,map[ln[i].l]+1,map[ln[i].r],ln[i].f);
      }
      printf("Test case #%d\nTotal explored area: %.2lf\n\n",++cas
            ,ans);
   }
   return 0;
}
```

1.2 Binary Indexed tree

```
int tree[MAXX];

inline int lowbit(const int &a)
{
   return a&-a;
}

inline void update(int pos,const int &val)
{
   while(pos<MAXX)
   {
      tree[pos]+=val;
      pos+=lowbit(pos);
   }
}

inline int read(int pos)
{
   int re(0);
   while(pos>0)
   {
      re+=tree[pos];
      pos-=lowbit(pos);
   }
   return re;
}

int find_Kth(int k)
{
   int now=0;
   for (char i=20;i>=0;--i)
   {
      now|=(1<<i);
      if (now>MAXX || tree[now]>=k)
         now^=(1<<i);
      else k-=tree[now];
   }
   return now+1;
}
```

1.3 COT

```
#include<cstdio>
#include<algorithm>

#define MAXX 100111
#define MAX (MAXX*23)
#define N 18

int sz[MAX],lson[MAX],rson[MAX],cnt;
int head[MAXX];
int pre[MAXX][N];
int map[MAXX],m;

int edge[MAXX],nxt[MAXX<<1],to[MAXX<<1];
int n,i,j,k,q,l,r,mid;
int num[MAXX],dg[MAXX];

int make(int l,int r)
{
   if(l==r)
      return ++cnt;
   int id(++cnt),mid((l+r)>>1);
   lson[id]=make(l,mid);
   rson[id]=make(mid+1,r);
   return id;
}

inline int update(int id,int pos)
{
   int re(++cnt);
   l=1;
   r=m;
   int nid(re);
   sz[nid]=sz[id]+1;
   while(l<r)
   {
      mid=(l+r)>>1;
      if(pos<=mid)
      {
         lson[nid]=++cnt;
         rson[nid]=rson[id];
         nid=lson[nid];
         id=lson[id];
         r=mid;
      }
      else
      {
         lson[nid]=lson[id];
         rson[nid]=++cnt;
         nid=rson[nid];
         id=rson[id];
         l=mid+1;
      }
      sz[nid]=sz[id]+1;
   }
   return re;
}

void rr(int now,int fa)
{
   dg[now]=dg[fa]+1;
   head[now]=update(head[fa],num[now]);
   for(int i(edge[now]);i;i=nxt[i])
      if(to[i]!=fa)
      {
         j=1;
         for(pre[to[i]][0]=now;j<N;++j)
            pre[to[i]][j]=pre[pre[to[i]][j-1]][j-1];
         rr(to[i],now);
      }
}

inline int query(int a,int b,int n,int k)
{
   static int tmp,t;
   l=1;
   r=m;
   a=head[a];
   b=head[b];
   t=num[n];
   n=head[n];
   while(l<r)
   {
      mid=(l+r)>>1;
      tmp=sz[lson[a]]+sz[lson[b]]-2*sz[lson[n]]+(l<=t && t<=mid);
      if(tmp>=k)
      {
         a=lson[a];
         b=lson[b];
         n=lson[n];
         r=mid;
      }
      else
      {
         k-=tmp;
         a=rson[a];
         b=rson[b];
         n=rson[n];
         l=mid+1;
      }
   }
   return l;
}

inline int lca(int a,int b)
{
   static int i,j;
   j=0;
   if(dg[a]<dg[b])
      std::swap(a,b);
   for(i=dg[a]-dg[b];i;i>>=1,++j)
      if(i&1)
         a=pre[a][j];
   if(a==b)
      return a;
   for(i=N-1;i>=0;--i)
      if(pre[a][i]!=pre[b][i])
      {
         a=pre[a][i];
         b=pre[b][i];
      }
   return pre[a][0];
}

int main()
{
   scanf("%d %d",&n,&q);
   for(i=1;i<=n;++i)
   {
      scanf("%d",num+i);
      map[i]=num[i];
   }
   std::sort(map+1,map+n+1);
   m=std::unique(map+1,map+n+1)-map-1;
   for(i=1;i<=n;++i)
      num[i]=std::lower_bound(map+1,map+m+1,num[i])-map;
   for(i=1;i<n;++i)
   {
      scanf("%d %d",&j,&k);
      nxt[++cnt]=edge[j];
      edge[j]=cnt;
      to[cnt]=k;

      nxt[++cnt]=edge[k];
      edge[k]=cnt;
      to[cnt]=j;
   }
   cnt=0;
   head[0]=make(1,m);
   rr(1,0);
   while(q--)
   {
      scanf("%d %d %d",&i,&j,&k);
      printf("%d\n",map[query(i,j,lca(i,j),k)]);
   }
   return 0;
}
```

1.4 GSS7

```
#include<cstdio>
#include<algorithm>
#include<queue>

#define MAXX 100111
#define MAX (MAXX<<1)

struct node
{
   bool set,rev;
   node *pre,*nxt[2],*fa;
   int lmax,max,rmax,sum,val,sz;
   node();
   node(int a);
}*tree[MAXX],*nil,*a,*b;

```

```
node::node()
{
   rev=set=false;
   fa=pre=nil;
   nxt[0]=nxt[1]=nil;
   sz=lmax=max=rmax=sum=val=0;
}

node::node(int a)
{
   set=rev=false;
   sum=val=a;
   sz=1;
   lmax=max=rmax=std::max(0,a);
   fa=pre=nxt[0]=nxt[1]=nil;
}

inline void add(node &x,const node &l,const node &r)
{
   x.max=std::max(l.rmax+r.lmax,std::max(l.max,r.max));
   x.lmax=std::max(l.lmax,l.sum+r.lmax);
   x.rmax=std::max(r.rmax,r.sum+l.rmax);
   x.sum=l.sum+r.sum;
}

inline void up(node *id)
{
   id->sz=id->nxt[0]->sz+id->nxt[1]->sz+1;
   id->sum=id->val+id->nxt[0]->sum+id->nxt[1]->sum;
   id->lmax=std::max(id->nxt[0]->lmax,id->nxt[0]->sum+id->val+id->nxt[1]->lmax);
   id->rmax=std::max(id->nxt[1]->rmax,id->nxt[1]->sum+id->val+id->nxt[0]->rmax);
   id->max=std::max(id->nxt[0]->rmax+id->val+id->nxt[1]->lmax,std::max(id->nxt[0]->max,id->nxt[1]->max));
}

inline void set(node *id,int val)
{
   if(id==nil)
      return;
   id->set=true;
   id->val=val;
   id->sum=val*id->sz;
   id->max=id->lmax=id->rmax=std::max(0,id->sum);
}

inline void down(node *id)
{
   if(id==nil)
      return;
   if(id->rev)
   {
      id->rev=false;
      for(int i(0);i<2;++i)
         if(id->nxt[i]!=nil)
         {
            id->nxt[i]->rev^=true;
            std::swap(id->nxt[i]->nxt[0],id->nxt[i]->nxt[1]);
            std::swap(id->nxt[i]->lmax,id->nxt[i]->rmax);
         }
   }
   if(id->set)
   {
      for(int i(0);i<2;++i)
         if(id->nxt[i]!=nil)
            set(id->nxt[i],id->val);
      id->set=false;
   }
}

inline void rot(node *id,int tp)
{
   node *k(id->pre);
   k->nxt[tp^1]=id->nxt[tp];
   if(id->nxt[tp]!=nil)
      id->nxt[tp]->pre=k;
   if(k->pre!=nil)
      k->pre->nxt[k==k->pre->nxt[1]]=id;
   id->pre=k->pre;
   id->nxt[tp]=k;
   k->pre=id;
   up(k);
   up(id);
}

node *fresh(node* id)
{
   node *re(id);
   if(id->pre!=nil)
      re=fresh(id->pre);
   down(id);
   return re;
}

inline void splay(node *id)
{
   node *rt(fresh(id));
   if(id!=rt)
      for(std::swap(rt->fa,id->fa);id->pre!=nil;rot(id,id==id->pre->nxt[0]));
}

inline void access(node *id)
{
   for(node *to(nil);id!=nil;id=id->fa)
   {
      splay(id);
      id->nxt[1]->pre=nil;
      if(id->nxt[1]!=nil)
         id->nxt[1]->fa=id;
      id->nxt[1]=to;
      if(to!=nil)
         to->pre=id;
      to->fa=nil;
      up(to=id);
   }
}

inline void lca(node *&to,node *&id)
{
   access(to);
   splay(id);
   for(to=nil;id->fa!=nil;splay(id=id->fa))
   {
      id->nxt[1]->pre=nil;
      if(id->nxt[1]!=nil)
         id->nxt[1]->fa=id;
      id->nxt[1]=to;
      if(to!=nil)
         to->pre=id;
      to->fa=nil;
      up(to=id);
   }
}

int n,i,j,k;
int nxt[MAX],to[MAX],edge[MAXX],cnt;
std::queue<int>q;

inline void add(int a,int b)
{
   nxt[++cnt]=edge[a];
   edge[a]=cnt;
   to[cnt]=b;
}

void rr(int now,int fa)
{
   for(int i(edge[now]);i;i=nxt[i])
      if(to[i]!=fa)
      {
         tree[to[i]]->fa=tree[now];
         rr(to[i],now);
      }
}

/*
void print(node *id)
{
   if(id!=nil)
   {
      print(id->nxt[0]);
      printf("%2d %2d %2d %2d %2d %2d %c %2d\n",id->val,id->sum,id->sz,id->lmax,id->max,id->rmax,id->rev?’r’:’n’,id->pre->val);
      print(id->nxt[1]);
   }
}
*/

int main()
{
   nil=new node();
   scanf("%d",&n);
   for(i=1;i<=n;++i)
   {
      scanf("%d",&j);
      tree[i]=new node(j);
   }
   for(i=1;i<n;++i)
   {
      scanf("%d %d",&j,&k);
      add(j,k);
      add(k,j);
   }
   tree[0]=nil;
   rr(1,0);
   scanf("%d",&n);
   while(n--)
   {
      scanf("%d %d %d",&k,&i,&j);
      a=tree[i];
      b=tree[j];
      access(a);
      splay(a);
      a->rev^=true;
      std::swap(a->nxt[0],a->nxt[1]);
      std::swap(a->lmax,a->rmax);
      access(b);
      splay(b);
      /*
      print(b);
      puts("");
      printf("%d %d %d %d\n",b->sum,b->nxt[0]->sum,b->val,b->nxt[1]->sum);
      */
      if(k==1)
         printf("%d\n",b->max);
      else
      {
         scanf("%d",&k);
         set(b,k);
      }
   }
   return 0;
}
```

1.5 Leftist tree

```
#include<cstdio>
#include<algorithm>

#define MAXX 100111

int val[MAXX],l[MAXX],r[MAXX],d[MAXX];

int set[MAXX];

int merge(int a,int b)
{
   if(!a)
      return b;
   if(!b)
```

```
      return a;
   if(val[a]<val[b]) // max-heap
      std::swap(a,b);
   r[a]=merge(r[a],b);
   if(d[l[a]]<d[r[a]])
      std::swap(l[a],r[a]);
   d[a]=d[r[a]]+1;
   set[l[a]]=set[r[a]]=a; // set a as father of its sons
   return a;
}

inline int find(int &a)
{
   while(set[a]) //brute-force to get the index of root
      a=set[a];
   return a;
}

inline void reset(int i)
{
   l[i]=r[i]=d[i]=set[i]=0;
}

int n,i,j,k;

int main()
{
   while(scanf("%d",&n)!=EOF)
   {
      for(i=1;i<=n;++i)
      {
         scanf("%d",val+i);
         reset(i);
      }
      scanf("%d",&n);
      while(n--)
      {
         scanf("%d %d",&i,&j);
         if(find(i)==find(j))
            puts("-1");
         else
         {
            k=merge(l[i],r[i]);
            val[i]>>=1;
            reset(i);
            set[i=merge(i,k)]=0;

            k=merge(l[j],r[j]);
            val[j]>>=1;
            reset(j);
            set[j=merge(j,k)]=0;

            set[k=merge(i,j)]=0;
            printf("%d\n",val[k]);
         }
      }
   }
   return 0;
}
```

1.6 Network

```
//HLD_(:3JZ)_
#include<cstdio>
#include<algorithm>
#include<cstdlib>

#define MAXX 80111
#define MAXE (MAXX<<1)
#define N 18

int edge[MAXX],nxt[MAXE],to[MAXE],cnt;
int fa[MAXX][N],dg[MAXX];

inline int lca(int a,int b)
{
   static int i,j;
   j=0;
   if(dg[a]<dg[b])
      std::swap(a,b);
   for(i=dg[a]-dg[b];i;i>>=1,++j)
      if(i&1)
         a=fa[a][j];
   if(a==b)
      return a;
   for(i=N-1;i>=0;--i)
      if(fa[a][i]!=fa[b][i])
      {
         a=fa[a][i];
         b=fa[b][i];
      }
   return fa[a][0];
}

inline void add(int a,int b)
{
   nxt[++cnt]=edge[a];
   edge[a]=cnt;
   to[cnt]=b;
}

int sz[MAXX],pre[MAXX],next[MAXX];

void rr(int now)
{
   sz[now]=1;
   int max,id;
   max=0;
   for(int i(edge[now]);i;i=nxt[i])
      if(to[i]!=fa[now][0])
      {
         fa[to[i]][0]=now;
         dg[to[i]]=dg[now]+1;
         rr(to[i]);
         sz[now]+=sz[to[i]];
         if(sz[to[i]]>max)
         {
            max=sz[to[i]];
            id=to[i];
         }
      }
   if(max)
   {
      next[now]=id;
      pre[id]=now;
   }
}

#define MAXT (MAXX*N*5)

namespace Treap
{
   int cnt;
   int son[MAXT][2],key[MAXT],val[MAXT],sz[MAXT];

   inline void init()
   {
      key[0]=RAND_MAX;
      val[0]=0xc0c0c0c0;
      cnt=0;
   }

   inline void up(int id)
   {
      sz[id]=sz[son[id][0]]+sz[son[id][1]]+1;
   }
   inline void rot(int &id,int tp)
   {
      static int k;
      k=son[id][tp];
      son[id][tp]=son[k][tp^1];
      son[k][tp^1]=id;
      up(id);
      up(k);
      id=k;
   }
   void insert(int &id,int v)
   {
      if(id)
      {
         int k(v>=val[id]);
         insert(son[id][k],v);
         if(key[son[id][k]]<key[id])
            rot(id,k);
         else
            up(id);
         return;
      }
      id=++cnt;
      key[id]=rand()-1;
      val[id]=v;
      sz[id]=1;
      son[id][0]=son[id][1]=0;
   }
   void del(int &id,int v)
   {
      if(!id)
         return;
      if(val[id]==v)
      {
         int k(key[son[id][1]]<key[son[id][0]]);
         if(!son[id][k])
         {
            id=0;
            return;
         }
         rot(id,k);
         del(son[id][k^1],v);
      }
      else
         del(son[id][v>val[id]],v);
      up(id);
   }
   int rank(int id,int v)
   {
      if(!id)
         return 0;
      if(val[id]<=v)
         return sz[son[id][0]]+1+rank(son[id][1],v);
      return rank(son[id][0],v);
   }
   void print(int id)
   {
      if(!id)
         return;
      print(son[id][0]);
      printf("%d ",val[id]);
      print(son[id][1]);
   }
}

int head[MAXX],root[MAXX],len[MAXX],pos[MAXX];

#define MAX (MAXX*6)
#define mid (l+r>>1)
#define lc lson[id],l,mid
#define rc rson[id],mid+1,r

int lson[MAX],rson[MAX];
int treap[MAX];

void make(int &id,int l,int r,int *the)
{
   id=++cnt;
   static int k;
   for(k=l;k<=r;++k)
      Treap::insert(treap[id],the[k]);
   if(l!=r)
   {
      make(lc,the);
      make(rc,the);
   }
}

```

```
int query(int id,int l,int r,int a,int b,int q)
{
   if(a<=l && r<=b)
      return Treap::rank(treap[id],q);
   int re(0);
   if(a<=mid)
      re=query(lc,a,b,q);
   if(b>mid)
      re+=query(rc,a,b,q);
   return re;
}

inline int query(int a,int b,int v)
{
   static int re;
   for(re=0;root[a]!=root[b];a=fa[root[a]][0])
      re+=query(head[root[a]],1,len[root[a]],1,pos[a],v);
   re+=query(head[root[a]],1,len[root[a]],pos[b],pos[a],v);
   return re;
}

inline void update(int id,int l,int r,int pos,int val,int n)
{
   while(l<=r)
   {
      Treap::del(treap[id],val);
      Treap::insert(treap[id],n);
      if(l==r)
         return;
      if(pos<=mid)
      {
         id=lson[id];
         r=mid;
      }
      else
      {
         id=rson[id];
         l=mid+1;
      }
   }
}

int n,q,i,j,k;
int val[MAXX];

int main()
{
   srand(1e9+7);
   scanf("%d %d",&n,&q);
   for(i=1;i<=n;++i)
      scanf("%d",val+i);
   for(k=1;k<n;++k)
   {
      scanf("%d %d",&i,&j);
      add(i,j);
      add(j,i);
   }
   rr(rand()%n+1);
   for(j=1;j<N;++j)
      for(i=1;i<=n;++i)
         fa[i][j]=fa[fa[i][j-1]][j-1];

   Treap::init();
   cnt=0;
   for(i=1;i<=n;++i)
      if(!pre[i])
      {
         static int tmp[MAXX];
         for(k=1,j=i;j;j=next[j],++k)
         {
            pos[j]=k;
            root[j]=i;
            tmp[k]=val[j];
         }
         --k;
         len[i]=k;
         make(head[i],1,k,tmp);
      }
   while(q--)
   {
      scanf("%d",&k);
      if(k)
      {
         static int a,b,c,d,l,r,ans,m;
         scanf("%d %d",&a,&b);
         c=lca(a,b);
         if(dg[a]+dg[b]-2*dg[c]+1<k)
         {
            puts("invalid request!");
            continue;
         }
         k=dg[a]+dg[b]-2*dg[c]+1-k+1;
         if(dg[a]<dg[b])
            std::swap(a,b);
         l=-1e9;
         r=1e9;
         if(b!=c)
         {
            d=a;
            for(i=0,j=dg[a]-dg[c]-1;j;j>>=1,++i)
               if(j&1)
                  d=fa[d][i];
            while(l<=r)
            {
               m=l+r>>1;
               if(query(a,d,m)+query(b,c,m)>=k)
               {
                  ans=m;
                  r=m-1;
               }
               else
                  l=m+1;
            }
         }
         else
         {
            while(l<=r)
            {
               m=l+r>>1;
               if(query(a,c,m)>=k)
               {
                  ans=m;
                  r=m-1;
               }
               else
                  l=m+1;
            }
         }
         printf("%d\n",ans);
      }
      else
      {
         scanf("%d %d",&i,&j);
         update(head[root[i]],1,len[root[i]],pos[i],val[i],j);
         val[i]=j;
      }
   }
   return 0;
}
```

1.7 OTOCI

```
//down
//debugup/down/select
#include<cstdio>
#include<algorithm>

#define MAXX 30111

int nxt[MAXX][2],fa[MAXX],pre[MAXX],val[MAXX],sum[MAXX];
bool rev[MAXX];

inline void up(int id)
{
   static int i;
   sum[id]=val[id];
   for(i=0;i<2;++i)
      if(nxt[id][i])
         sum[id]+=sum[nxt[id][i]];
}

inline void rot(int id,int tp)
{
   static int k;
   k=pre[id];
   nxt[k][tp^1]=nxt[id][tp];
   if(nxt[id][tp])
      pre[nxt[id][tp]]=k;
   if(pre[k])
      nxt[pre[k]][k==nxt[pre[k]][1]]=id;
   pre[id]=pre[k];
   nxt[id][tp]=k;
   pre[k]=id;
   up(k);
   up(id);
}

inline void down(int id) //down
{
   static int i;
   if(rev[id])
   {
      rev[id]=false;
      std::swap(nxt[id][0],nxt[id][1]);
      for(i=0;i<2;++i)
         if(nxt[id][i])
            rev[nxt[id][i]]^=true;
   }
}

int freshen(int id)
{
   int re(id);
   if(pre[id])
      re=freshen(pre[id]);
   down(id);
   return re;
}

inline void splay(int id)//down
{
   static int rt;
   if(id!=(rt=freshen(id)))
      for(std::swap(fa[id],fa[rt]);pre[id];rot(id,id==nxt[pre[id]][0]));
   /* another faster methond:
   if(id!=rt)
   {
      std::swap(fa[id],fa[rt]);
      do
      {
         rt=pre[id];
         if(pre[rt])
         {
            k=(nxt[pre[rt]][0]==rt);
            if(nxt[rt][k]==id)
               rot(id,k^1);
            else
               rot(rt,k);
            rot(id,k);
         }
         else
            rot(id,id==nxt[rt][0]);
      }
      while(pre[id]);
   }
   */
}

inline void access(int id)
{
   static int to;
   for(to=0;id;id=fa[id])
   {
      splay(id);
      if(nxt[id][1])
      {
         pre[nxt[id][1]]=0;
         fa[nxt[id][1]]=id;
      }
      nxt[id][1]=to;
      if(to)
      {
         pre[to]=id;
         fa[to]=0;
      }
      up(to=id);
   }
}

inline int getrt(int id)
{
   access(id);
   splay(id);
   while(nxt[id][0])
   {
      id=nxt[id][0];
      down(id);
   }
   return id;
}

inline void makert(int id)
{
   access(id);
   splay(id);
   if(nxt[id][0])
      rev[id]^=true;
}

int n,i,j,k,q;
char buf[11];

int main()
{
   scanf("%d",&n);
   for(i=1;i<=n;++i)
      scanf("%d",val+i);
   scanf("%d",&q);
   while(q--)
   {
      scanf("%s %d %d",buf,&i,&j);
      switch(buf[0])
      {
         case ’b’:
            if(getrt(i)==getrt(j))
               puts("no");
            else
            {
               puts("yes");
               makert(i);
               fa[i]=j;
            }
            break;
         case ’p’:
            access(i);
            splay(i);
            val[i]=j;
            up(i);
            break;
         case ’e’:
            if(getrt(i)!=getrt(j))
               puts("impossible");
            else
            {
               makert(i);
               access(j);
               splay(j);
               printf("%d\n",sum[j]);
            }
            break;
      }
   }
   return 0;
}
```

1.8 picture

```
#include<cstdio>
#include<algorithm>
#include<map>

#define MAXX 5555
#define MAX MAXX<<3
#define inf 10011

int n,i;
int mid[MAX],cnt[MAX],len[MAX],seg[MAX];
bool rt[MAX],lf[MAX];

std::map<int,int>map;
std::map<int,int>::iterator it;
int rmap[inf];
long long sum;
int x1,x2,y1,y2,last;

void make(int id,int l,int r)
{
   mid[id]=(l+r)>>1;
   if(l!=r)
   {
      make(id<<1,l,mid[id]);
      make(id<<1|1,mid[id]+1,r);
   }
}

void update(int id,int ll,int rr,int l,int r,int val)
{
   if(l==ll && rr==r)
   {
      cnt[id]+=val;
      if(cnt[id])
      {
         rt[id]=lf[id]=true;
         len[id]=rmap[r]-rmap[l-1];
         seg[id]=1;
      }
      else
         if(l!=r)
         {
            len[id]=len[id<<1]+len[id<<1|1];
            seg[id]=seg[id<<1]+seg[id<<1|1];
            if(rt[id<<1] && lf[id<<1|1])
               --seg[id];
            rt[id]=rt[id<<1|1];
            lf[id]=lf[id<<1];
         }
         else
         {
            len[id]=0;
            rt[id]=lf[id]=false;
            seg[id]=0;
         }
      return;
   }
   if(mid[id]>=r)
      update(id<<1,ll,mid[id],l,r,val);
   else
      if(mid[id]<l)
         update(id<<1|1,mid[id]+1,rr,l,r,val);
      else
      {
         update(id<<1,ll,mid[id],l,mid[id],val);
         update(id<<1|1,mid[id]+1,rr,mid[id]+1,r,val);
      }
   if(!cnt[id])
   {
      len[id]=len[id<<1]+len[id<<1|1];
      seg[id]=seg[id<<1]+seg[id<<1|1];
      if(rt[id<<1] && lf[id<<1|1])
         --seg[id];
      rt[id]=rt[id<<1|1];
      lf[id]=lf[id<<1];
   }
}

struct node
{
   int l,r,h;
   char val;
   inline bool operator<(const node &a)const
   {
      return h==a.h?val<a.val:h<a.h; // trick watch out. val<a.val
            ? val>a.val?
   }
   inline void print()
   {
      printf("%d %d %d %d\n",l,r,h,val);
   }
}ln[inf];

int main()
{
   make(1,1,inf);
   scanf("%d",&n);
   n<<=1;
   map.clear();
   for(i=0;i<n;++i)
   {
      scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
      ln[i].l=x1;
      ln[i].r=x2;
      ln[i].h=y1;
      ln[i].val=1;
      ln[++i].l=x1;
      ln[i].r=x2;
      ln[i].h=y2;
      ln[i].val=-1;
      map[x1]=1;
      map[x2]=1;
   }
   i=1;
   for(it=map.begin();it!=map.end();++it,++i)
   {
      it->second=i;
      rmap[i]=it->first;
   }
   i=0;
   std::sort(ln,ln+n);
   update(1,1,inf,map[ln[0].l]+1,map[ln[0].r],ln[0].val);
   sum+=len[1];
   last=len[1];
   for(i=1;i<n;++i)
   {
      sum+=2*seg[1]*(ln[i].h-ln[i-1].h);
      update(1,1,inf,map[ln[i].l]+1,map[ln[i].r],ln[i].val);
      sum+=abs(len[1]-last);
      last=len[1];
   }
   printf("%lld\n",sum);
   return 0;
}
```

1.9 Size Blanced Tree

```
template<class Tp>class sbt
{
   public:
      inline void init()
      {
         rt=cnt=l[0]=r[0]=sz[0]=0;
      }
      inline void ins(const Tp &a)
      {
         ins(rt,a);
      }
      inline void del(const Tp &a)
      {
         del(rt,a);
      }
      inline bool find(const Tp &a)
      {
         return find(rt,a);
      }
      inline Tp pred(const Tp &a)
      {
         return pred(rt,a);
      }
      inline Tp succ(const Tp &a)
      {
         return succ(rt,a);
      }
      inline bool empty()
      {
         return !sz[rt];
      }
      inline Tp min()
      {
         return min(rt);
      }
      inline Tp max()
      {
         return max(rt);
      }
      inline void delsmall(const Tp &a)
      {
         dels(rt,a);
      }
      inline int rank(const Tp &a)
      {
         return rank(rt,a);
      }
      inline Tp sel(const int &a)
      {
         return sel(rt,a);
      }
      inline Tp delsel(int a)
      {
         return delsel(rt,a);
      }
   private:
      int cnt,rt,l[MAXX],r[MAXX],sz[MAXX];
      Tp val[MAXX];
      inline void rro(int &pos)
      {
         int k(l[pos]);
         l[pos]=r[k];
         r[k]=pos;
         sz[k]=sz[pos];
         sz[pos]=sz[l[pos]]+sz[r[pos]]+1;
         pos=k;
      }
      inline void lro(int &pos)
      {
         int k(r[pos]);
         r[pos]=l[k];
         l[k]=pos;
         sz[k]=sz[pos];
         sz[pos]=sz[l[pos]]+sz[r[pos]]+1;
         pos=k;
      }
      inline void mt(int &pos,bool flag)
      {
         if(!pos)
            return;
         if(flag)
            if(sz[r[r[pos]]]>sz[l[pos]])
               lro(pos);
            else
               if(sz[l[r[pos]]]>sz[l[pos]])
               {
                  rro(r[pos]);
                  lro(pos);
               }
               else
                  return;
         else
            if(sz[l[l[pos]]]>sz[r[pos]])
               rro(pos);
            else
               if(sz[r[l[pos]]]>sz[r[pos]])
               {
                  lro(l[pos]);
                  rro(pos);
               }
               else
                  return;
         mt(l[pos],false);
         mt(r[pos],true);
         mt(pos,false);
         mt(pos,true);
      }
      void ins(int &pos,const Tp &a)
      {
         if(pos)
         {
            ++sz[pos];
            if(a<val[pos])
               ins(l[pos],a);
            else
               ins(r[pos],a);
            mt(pos,a>=val[pos]);
            return;
         }
         pos=++cnt;
         l[pos]=r[pos]=0;
         val[pos]=a;
         sz[pos]=1;
      }
      Tp del(int &pos,const Tp &a)
      {
         --sz[pos];
```

```
         if(val[pos]==a || (a<val[pos] && !l[pos]) || (a>val[pos]
             && !r[pos]))
         {
            Tp ret(val[pos]);
            if(!l[pos] || !r[pos])
               pos=l[pos]+r[pos];
            else
               val[pos]=del(l[pos],val[pos]+1);
            return ret;
         }
         else
            if(a<val[pos])
               return del(l[pos],a);
            else
               return del(r[pos],a);
      }
      bool find(int &pos,const Tp &a)
      {
         if(!pos)
            return false;
         if(a<val[pos])
            return find(l[pos],a);
         else
            return (val[pos]==a || find(r[pos],a));
      }
      Tp pred(int &pos,const Tp &a)
      {
         if(!pos)
            return a;
         if(a>val[pos])
         {
            Tp ret(pred(r[pos],a));
            if(ret==a)
               return val[pos];
            else
               return ret;
         }
         return pred(l[pos],a);
      }
      Tp succ(int &pos,const Tp &a)
      {
         if(!pos)
            return a;
         if(a<val[pos])
         {
            Tp ret(succ(l[pos],a));
            if(ret==a)
               return val[pos];
            else
               return ret;
         }
         return succ(r[pos],a);
      }
      Tp min(int &pos)
      {
         if(l[pos])
            return min(l[pos]);
         else
            return val[pos];
      }
      Tp max(int &pos)
      {
         if(r[pos])
            return max(r[pos]);
         else
            return val[pos];
      }
      void dels(int &pos,const Tp &v)
      {
         if(!pos)
            return;
         if(val[pos]<v)
         {
            pos=r[pos];
            dels(pos,v);
            return;
         }
         dels(l[pos],v);
         sz[pos]=1+sz[l[pos]]+sz[r[pos]];
      }
      int rank(const int &pos,const Tp &v)
      {
         if(val[pos]==v)
            return sz[l[pos]]+1;
         if(v<val[pos])
            return rank(l[pos],v);
         return rank(r[pos],v)+sz[l[pos]]+1;
      }
      Tp sel(const int &pos,const int &v)
      {
         if(sz[l[pos]]+1==v)
            return val[pos];
         if(v>sz[l[pos]])
            return sel(r[pos],v-sz[l[pos]]-1);
         return sel(l[pos],v);
      }
      Tp delsel(int &pos,int k)
      {
         --sz[pos];
         if(sz[l[pos]]+1==k)
         {
            Tp re(val[pos]);
            if(!l[pos] || !r[pos])
               pos=l[pos]+r[pos];
            else
               val[pos]=del(l[pos],val[pos]+1);
            return re;
         }
         if(k>sz[l[pos]])
            return delsel(r[pos],k-1-sz[l[pos]]);
         return delsel(l[pos],k);
      }
};
```

1.10 Sparse Table - rectangle

```
#include<iostream>
#include<cstdio>
#include<algorithm>

#define MAXX 310

int mat[MAXX][MAXX];
int table[9][9][MAXX][MAXX];
int n;
short lg[MAXX];

int main()
{
   for(int i(2);i<MAXX;++i)
      lg[i]=lg[i>>1]+1;
   int T;
   std::cin >> T;
   while (T--)
   {
      std::cin >> n;
      for (int i = 0; i < n; ++i)
         for (int j = 0; j < n; ++j)
         {
            std::cin >> mat[i][j];
            table[0][0][i][j] = mat[i][j];
         }

      //
      for(int i=0;i<=lg[n];++i) // width
      {
         for(int j=0;j<=lg[n];++j) //height
         {
            if(i==0 && j==0)
               continue;
            for(int ii=0;ii+(1<<j)<=n;++ii)
               for(int jj=0;jj+(1<<i)<=n;++jj)
                  if(i==0)
                     table[i][j][ii][jj]=std::min(table[i][j-1][ii
                            ][jj],table[i][j-1][ii+(1<<(j-1))][jj
                            ]);
                  else
                     table[i][j][ii][jj]=std::min(table[i-1][j][ii
                            ][jj],table[i-1][j][ii][jj+(1<<(i-1))
                            ]);
         }
      }
      long long N;
      std::cin >> N;
      int r1, c1, r2, c2;
      for (int i = 0; i < N; ++i)
      {
         scanf("%d%d%d%d",&r1,&c1,&r2,&c2);
         --r1;
         --c1;
         --r2;
         --c2;
         int w=lg[c2-c1+1];
         int h=lg[r2-r1+1];
         printf("%d\n",std::min(table[w][h][r1][c1],std::min(table
                [w][h][r1][c2-(1<<w)+1],std::min(table[w][h][r2
                -(1<<h)+1][c1],table[w][h][r2-(1<<h)+1][c2-(1<<w)
                +1]))));
      }
   }
   return 0;
}
```

1.11 Sparse Table - square

```
int num[MAXX][MAXX],max[MAXX][MAXX][10];
short lg[MAXX];

int main()
{
   for(i=2;i<MAXX;++i)
      lg[i]=lg[i>>1]+1;
   scanf("%hd %d",&n,&q);
   for(i=0;i<n;++i)
      for(j=0;j<n;++j)
      {
         scanf("%d",num[i]+j);
         max[i][j][0]=num[i][j];
      }
   for(k=1;k<=lg[n];++k)
   {
      l=n+1-(1<<k);
      for(i=0;i<l;++i)
         for(j=0;j<l;++j)
            max[i][j][k]=std::max(std::max(max[i][j][k-1],max[i
                  +(1<<(k-1))][j][k-1]),std::max(max[i][j+(1<<(k
                  -1))][k-1],max[i+(1<<(k-1))][j+(1<<(k-1))][k
                  -1]));
   }
   printf("Case %hd:\n",t);
   while(q--)
   {
      scanf("%hd %hd %hd",&i,&j,&l);
      --i;
      --j;
      k=lg[l];
      printf("%d\n",std::max(std::max(max[i][j][k],max[i][j+l-(1<<
             k)][k]),std::max(max[i+l-(1<<k)][j][k],max[i+l-(1<<k)
             ][j+l-(1<<k)][k])));
   }
}
```

1.12 Sparse Table

```
int num[MAXX],min[MAXX][20];
int lg[MAXX];


int main()
{
   for(i=2;i<MAXX;++i)
      lg[i]=lg[i>>1]+1;
   scanf("%d %d",&n,&q);
   for(i=1;i<=n;++i)
   {
      scanf("%d",num+i);
      min[i][0]=num[i];
   }
   for(j=1;j<=lg[n];++j)
   {
      l=n+1-(1<<j);
      j_=j-1;
      j__=(1<<j_);
      for(i=1;i<=l;++i)
         min[i][j]=std::min(min[i][j_],min[i+j__][j_]);
   }
   printf("Case %hd:\n",t);
   while(q--)
   {
      scanf("%d %d",&i,&j);
      k=lg[j-i+1];
      printf("%d\n",std::min(min[i][k],min[j-(1<<k)+1][k]));
   }
}
```

1.13 Trea

```
#include<cstdlib>
#include<ctime>
#include<cstring>

struct node
{
   node *ch[2];
   int sz,val,key;
   node(){memset(this,0,sizeof(node));}
   node(int a);
}*null;

node::node(int a):sz(1),val(a),key(rand()-1){ch[0]=ch[1]=null;}

class Treap
{
   inline void up(node *pos)
   {
      pos->sz=pos->ch[0]->sz+pos->ch[1]->sz+1;
   }
   inline void rot(node *&pos,int tp)
   {
      node *k(pos->ch[tp]);
      pos->ch[tp]=k->ch[tp^1];
      k->ch[tp^1]=pos;
      up(pos);
      up(k);
      pos=k;
   }

   void insert(node *&pos,int val)
   {
      if(pos!=null)
      {
         int t(val>=pos->val);
         insert(pos->ch[t],val);
         if(pos->ch[t]->key<pos->key)
            rot(pos,t);
         else
            up(pos);
         return;
      }
      pos=new node(val);
   }
   void rec(node *pos)
   {
      if(pos!=null)
      {
         rec(pos->ch[0]);
         rec(pos->ch[1]);
         delete pos;
      }
   }
   inline int sel(node *pos,int k)
   {
      while(pos->ch[0]->sz+1!=k)
         if(pos->ch[0]->sz>=k)
            pos=pos->ch[0];
         else
         {
            k-=pos->ch[0]->sz+1;
            pos=pos->ch[1];
         }
      return pos->val;
   }
   void del(node *&pos,int val)
   {
      if(pos!=null)
      {
         if(pos->val==val)
         {
            int t(pos->ch[1]->key<pos->ch[0]->key);
            if(pos->ch[t]==null)
            {
               delete pos;
               pos=null;
               return;
            }
            rot(pos,t);
            del(pos->ch[t^1],val);
         }
         else
            del(pos->ch[val>pos->val],val);
         up(pos);
      }
   }
   public:
   node *rt;

   Treap():rt(null){}
   inline void insert(int val)
   {
      insert(rt,val);
   }
   inline void reset()
   {
      rec(rt);
      rt=null;
   }
   inline int sel(int k)
   {
      if(k<1 || k>rt->sz)
         return 0;
      return sel(rt,rt->sz+1-k);
   }
   inline void del(int val)
   {
      del(rt,val);
   }
   inline int size()
   {
      return rt->sz;
   }
}treap[MAXX];

init:
{
   srand(time(0));
   null=new node();
   null->val=0xc0c0c0c0;
   null->sz=0;
   null->key=RAND_MAX;
   null->ch[0]=null->ch[1]=null;
   for(i=0;i<MAXX;++i)
      treap[i].rt=null;
}
```

2 dynamic programming

2.1 knapsack problem

```
multiple-choice knapsack problem:

for k
   for v=V..0
      for ik
         f[v]=max{f[v],f[v-c[i]]+w[i]}
```

2.2 LCIS

```
#include<cstdio>
#include<cstring>
#include<vector>

#define MAXX 1111

int T;
int n,m,p,i,j,k;
std::vector<int>the[2];
int dp[MAXX],path[MAXX];
int ans[MAXX];

int main()
{
   the[0].reserve(MAXX);
   the[1].reserve(MAXX);
   {
      scanf("%d",&n);
      the[0].resize(n);
      for(i=0;i<n;++i)
         scanf("%d",&the[0][i]);
      scanf("%d",&m);
      the[1].resize(m);
      for(i=0;i<m;++i)
         scanf("%d",&the[1][i]);
      memset(dp,0,sizeof dp);
      for(i=0;i<the[0].size();++i)
      {
         n=0;
         p=-1;
         for(j=0;j<the[1].size();++j)
         {
            if(the[0][i]==the[1][j] && n+1>dp[j])
            {
               dp[j]=n+1;
               path[j]=p;
            }
            if(the[1][j]<the[0][i] && n<dp[j])
            {
               n=dp[j];
               p=j;
            }
         }
      }
      n=0;
      p=-1;
      for(i=0;i<the[1].size();++i)
         if(dp[i]>n)
            n=dp[p=i];
      printf("%d\n",n);
      for(i=n-1;i>=0;--i)
      {
         ans[i]=the[1][p];
         p=path[p];
      }
      for(i=0;i<n;++i)
         printf("%d ",ans[i]);
      puts("");
   }
   return 0;
}
```

2.3 LCS

```
#include<cstdio>
#include<algorithm>
#include<vector>

#define MAXX 111
#define N 128

std::vector<char>the[2];
std::vector<int>dp(MAXX),p[N];

int i,j,k;
char buf[MAXX];
int t;

int main()
{
   the[0].reserve(MAXX);
   the[1].reserve(MAXX);
   while(gets(buf),buf[0]!='#')
   {
      the[0].resize(0);
      for(i=0;buf[i];++i)
         the[0].push_back(buf[i]);
      the[1].resize(0);
      gets(buf);
      for(i=0;buf[i];++i)
         the[1].push_back(buf[i]);
      for(i=0;i<N;++i)
         p[i].resize(0);
      for(i=0;i<the[1].size();++i)
         p[the[1][i]].push_back(i);
      dp.resize(1);
      dp[0]=-1;
      for(i=0;i<the[0].size();++i)
         for(j=p[the[0][i]].size()-1;j>=0;--j)
         {
            k=p[the[0][i]][j];
            if(k>dp.back())
               dp.push_back(k);
            else
               *std::lower_bound(dp.begin(),dp.end(),k)=k;
         }
      printf("Case #%d: you can visit at most %ld cities.\n",++t,
             dp.size()-1);
   }
   return 0;
}
```

3 geometry

3.1 3D

```
struct pv
{
  double x,y,z;
  pv() {}
  pv(double xx,double yy,double zz):x(xx),y(yy),z(zz) {}
  pv operator -(const pv& b)const
  {
    return pv(x-b.x,y-b.y,z-b.z);
  }
  pv operator *(const pv& b)const
  {
    return pv(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);
  }
  double operator &(const pv& b)const
  {
    return x*b.x+y*b.y+z*b.z;
  }
};

//
double Norm(pv p)
{
  return sqrt(p&p);
}

//'Vtheta'
pv Trans(pv pa,pv V,double theta)
{
    double s = sin(theta);
    double c = cos(theta);
    double x,y,z;
    x = V.x;
```

```
   y = V.y;
   z = V.z;
   pv pp =
      pv(
             (x*x*(1-c)+c)*pa.x+(x*y*(1-c)-z*s)*pa.y+(x*z*(1-c)+y*s)*pa.z,
             (y*x*(1-c)+z*s)*pa.x+(y*y*(1-c)+c)*pa.y+(y*z*(1-c)-x*s)*pa.z,
             (x*z*(1-c)-y*s)*pa.x+(y*z*(1-c)+x*s)*pa.y+(z*z*(1-c)+c)*pa.z
        );
   return pp;
}

//

x=r*sin()*cos();
y=r*sin()*sin();
z=r*cos();

r=sqrt(x*2+y*2+z*2);//??
r=sqrt(x^2+y^2+z^2);//??

=atan(y/x);
=acos(z/r);

r[0,]
[0,2]
[0,]

lat1[-/2,/2]
lng1[-,]

pv getpv(double lat,double lng,double r)
{
 lat += pi/2;
 lng += pi;
 return
   pv(r*sin(lat)*cos(lng),r*sin(lat)*sin(lng),r*cos(lat));
}

//

#include<cstdio>
#include<cmath>

#define MAXX 1111

char buf[MAXX];
const double r=6875.0/2,pi=acos(-1.0);
double a,b,c,x1,x2,y2,ans;

int main()
{
   double y1;
   while(gets(buf)!=NULL)
   {
      gets(buf);
      gets(buf);

      scanf("%lf^%lf'%lf\" %s\n",&a,&b,&c,buf);
      x1=a+b/60+c/3600;
      x1=x1*pi/180;
      if(buf[0]=='S')
         x1=-x1;

      scanf("%s",buf);
      scanf("%lf^%lf'%lf\" %s\n",&a,&b,&c,buf);
      y1=a+b/60+c/3600;
      y1=y1*pi/180;
      if(buf[0]=='W')
         y1=-y1;

      gets(buf);

      scanf("%lf^%lf'%lf\" %s\n",&a,&b,&c,buf);
      x2=a+b/60+c/3600;
      x2=x2*pi/180;
      if(buf[0]=='S')
         x2=-x2;

      scanf("%s",buf);
      scanf("%lf^%lf'%lf\" %s\n",&a,&b,&c,buf);
      y2=a+b/60+c/3600;
      y2=y2*pi/180;
      if(buf[0]=='W')
         y2=-y2;

      ans=acos(cos(x1)*cos(x2)*cos(y1-y2)+sin(x1)*sin(x2))*r;
      printf("The distance to the iceberg: %.2lf miles.\n",ans);
      if(ans+0.005<100)
         puts("DANGER!");

      gets(buf);
   }
   return 0;
}

inline bool ZERO(const double &a)
{
   return fabs(a)<eps;
}

//
inline bool ZERO(pv p)
{
   return (ZERO(p.x) && ZERO(p.y) && ZERO(p.z));
}

//
bool LineIntersect(Line3D L1, Line3D L2)
{
   pv s = L1.s-L1.e;
   pv e = L2.s-L2.e;
   pv p = s*e;
   if (ZERO(p))
      return false; //
   p = (L2.s-L1.e)*(L1.s-L1.e);
   return ZERO(p&L2.e); //
}

//
bool inter(pv a,pv b,pv c,pv d)
{
   pv ret = (a-b)*(c-d);
   pv t1 = (b-a)*(c-a);
   pv t2 = (b-a)*(d-a);
   pv t3 = (d-c)*(a-c);
   pv t4 = (d-c)*(b-c);
   return sgn(t1&ret)*sgn(t2&ret) < 0 && sgn(t3&ret)*sgn(t4&ret) < 0;
}

//
bool OnLine(pv p, Line3D L)
{
   return ZERO((p-L.s)*(L.e-L.s));
}

//
bool OnSeg(pv p, Line3D L)
{
   return (ZERO((L.s-p)*(L.e-p)) && EQ(Norm(p-L.s)+Norm(p-L.e),Norm(L.e-L.s)));
}

//
double Distance(pv p, Line3D L)
{
   return (Norm((p-L.s)*(L.e-L.s))/Norm(L.e-L.s));
}

//
//[0,]
double Inclination(Line3D L1, Line3D L2)
{
   pv u = L1.e - L1.s;
   pv v = L2.e - L2.s;
   return acos( (u & v) / (Norm(u)*Norm(v)) );
}
```

3.2 3DCH

```
#include<cstdio>
#include<cmath>
#include<vector>
#include<algorithm>

#define MAXX 1111
#define eps 1e-8
#define inf 1e20

struct pv
{
   double x,y,z;
   pv(){}
   pv(const double &xx,const double &yy,const double &zz):x(xx),y(yy),z(zz){}
   inline pv operator-(const pv &i)const
   {
      return pv(x-i.x,y-i.y,z-i.z);
   }
   inline pv operator*(const pv &i)const //
   {
      return pv(y*i.z-z*i.y,z*i.x-x*i.z,x*i.y-y*i.x);
   }
   inline double operator^(const pv &i)const //
   {
      return x*i.x+y*i.y+z*i.z;
   }
   inline double len()
   {
      return sqrt(x*x+y*y+z*z);
   }
};

struct pla
{
   short a,b,c;
   bool ok;
   pla(){}
   pla(const short &aa,const short &bb,const short &cc):a(aa),b(bb),c(cc),ok(true){}
   inline void set();
   inline void print()
   {
      printf("%hd %hd %hd\n",a,b,c);
   }
};

pv pnt[MAXX];
std::vector<pla>fac;
short to[MAXX][MAXX];

inline void pla::set()
{
   to[a][b]=to[b][c]=to[c][a]=fac.size();
}

inline double ptof(const pv &p,const pla &f) //?
{
   return (pnt[f.b]-pnt[f.a])*(pnt[f.c]-pnt[f.a])^(p-pnt[f.a]);
}

inline double vol(const pv &a,const pv &b,const pv &c,const pv &d) //*6
{
   return (b-a)*(c-a)^(d-a);
}

inline double ptof(const pv &p,const short &f) //pf
{
   return fabs(vol(pnt[fac[f].a],pnt[fac[f].b],pnt[fac[f].c],p)/((pnt[fac[f].b]-pnt[fac[f].a])*(pnt[fac[f].c]-pnt[fac[f].a])).len());
}

void dfs(const short&,const short&);

void deal(const short &p,const short &a,const short &b)
{
   if(fac[to[a][b]].ok)
      if(ptof(pnt[p],fac[to[a][b]])>eps)
         dfs(p,to[a][b]);
      else
      {
         pla add(b,a,p);
         add.set();
         fac.push_back(add);
      }
}

void dfs(const short &p,const short &now)
{
   fac[now].ok=false;
   deal(p,fac[now].b,fac[now].a);
   deal(p,fac[now].c,fac[now].b);
   deal(p,fac[now].a,fac[now].c);
}

inline void make()
{
   fac.resize(0);
   if(n<4)
      return;

   for(i=1;i<n;++i)
      if((pnt[0]-pnt[i]).len()>eps)
      {
         std::swap(pnt[i],pnt[1]);
         break;
      }
   if(i==n)
      return;

   for(i=2;i<n;++i)
      if(((pnt[0]-pnt[1])*(pnt[1]-pnt[i])).len()>eps)
      {
         std::swap(pnt[i],pnt[2]);
         break;
      }
   if(i==n)
      return;

   for(i=3;i<n;++i)
      if(fabs((pnt[0]-pnt[1])*(pnt[1]-pnt[2])^(pnt[2]-pnt[i]))>eps)
      {
         std::swap(pnt[3],pnt[i]);
         break;
      }
   if(i==n)
      return;

   for(i=0;i<4;++i)
   {
      pla add((i+1)%4,(i+2)%4,(i+3)%4);
      if(ptof(pnt[i],add)>0)
         std::swap(add.c,add.b);
      add.set();
      fac.push_back(add);
   }
   for(;i<n;++i)
      for(j=0;j<fac.size();++j)
         if(fac[j].ok && ptof(pnt[i],fac[j])>eps)
         {
            dfs(i,j);
            break;
         }

   short tmp(fac.size());
   fac.resize(0);
   for(i=0;i<tmp;++i)
      if(fac[i].ok)
         fac.push_back(fac[i]);
}

inline pv gc() //
{
   pv re(0,0,0),o(0,0,0);
   double all(0),v;
   for(i=0;i<fac.size();++i)
   {
      v=vol(o,pnt[fac[i].a],pnt[fac[i].b],pnt[fac[i].c]);
      re+=(pnt[fac[i].a]+pnt[fac[i].b]+pnt[fac[i].c])*0.25*v;
      all+=v;
   }
   return re*(1/all);
}

inline bool same(const short &s,const short &t) //
{
   pv &a=pnt[fac[s].a],&b=pnt[fac[s].b],&c=pnt[fac[s].c];
   return fabs(vol(a,b,c,pnt[fac[t].a]))<eps && fabs(vol(a,b,c,pnt[fac[t].b]))<eps && fabs(vol(a,b,c,pnt[fac[t].c]))<eps;
}

//
inline short facetcnt()
{
   short ans=0;
   for(short i=0;i<fac.size();++i)
   {
      for(j=0;j<i;++j)
         if(same(i,j))
            break;
      if(j==i)
         ++ans;
   }
   return ans;
}

//
inline short trianglecnt()
{
   return fac.size();
}

//*2
inline double area(const pv &a,const pv &b,const pv &c)
{
      return (b-a)*(c-a).len();
}

//
inline double area()
{
   double ret(0);
   for(i=0;i<fac.size();++i)
      ret+=area(pnt[fac[i].a],pnt[fac[i].b],pnt[fac[i].c]);
   return ret/2;
}

//
inline double volume()
{
   pv o(0,0,0);
   double ret(0);
   for(short i(0);i<fac.size();++i)
      ret+=vol(o,pnt[fac[i].a],pnt[fac[i].b],pnt[fac[i].c]);
   return fabs(ret/6);
}
```

3.3 circle ploy's intersection area

```
bool InCircle(Point a,double r)
{
 return cmp(a.x*a.x+a.y*a.y,r*r) <= 0;
 //`EPS`
}

double CalcArea(Point a,Point b,double r)
{
 Point p[4];
 int tot = 0;
 p[tot++] = a;

 Point tv = Point(a,b);
 Line tmp = Line(Point(0,0),Point(tv.y,-tv.x));
 Point near = LineToLine(Line(a,b),tmp);
 if (cmp(near.x*near.x+near.y*near.y,r*r) <= 0)
 {
  double A,B,C;
  A = near.x*near.x+near.y*near.y;
  C = r;
  B = C*C-A;
  double tvl = tv.x*tv.x+tv.y*tv.y;
  double tmp = sqrt(B/tvl); //
  p[tot] = Point(near.x+tmp*tv.x,near.y+tmp*tv.y);
  if (OnSeg(Line(a,b),p[tot]) == true) tot++;
  p[tot] = Point(near.x-tmp*tv.x,near.y-tmp*tv.y);
  if (OnSeg(Line(a,b),p[tot]) == true) tot++;
 }
 if (tot == 3)
 {
  if (cmp(Point(p[0],p[1]).Length(),Point(p[0],p[2]).Length()) > 0)
   swap(p[1],p[2]);
 }
 p[tot++] = b;

 double res = 0.0,theta,a0,a1,sgn;
 for (int i = 0;i < tot-1;i++)
 {
  if (InCircle(p[i],r) == true && InCircle(p[i+1],r) == true)
  {
   res += 0.5*xmult(p[i],p[i+1]);
  }
  else
  {
   a0 = atan2(p[i+1].y,p[i+1].x);
   a1 = atan2(p[i].y,p[i].x);
   if (a0 < a1) a0 += 2*pi;
   theta = a0-a1;
   if (cmp(theta,pi) >= 0) theta = 2*pi-theta;
   sgn = xmult(p[i],p[i+1])/2.0;
   if (cmp(sgn,0) < 0) theta = -theta;
   res += 0.5*r*r*theta;
  }
 }
 return res;
}

//

area2 = 0.0;
for (int i = 0;i < resn;i++) //
   area2 += CalcArea(p[i],p[(i+1)%resn],r);
```

3.4 circle's area

```
//
{
   for (int i = 0; i < n; i++)
   {
      scanf("%lf%lf%lf",&c[i].c.x,&c[i].c.y,&c[i].r);
      del[i] = false;
   }
   for (int i = 0; i < n; i++)
      if (del[i] == false)
      {
         if (c[i].r == 0.0)
            del[i] = true;
         for (int j = 0; j < n; j++)
            if (i != j)
               if (del[j] == false)
                  if (cmp(Point(c[i].c,c[j].c).Len()+c[i].r,c[j].r
                        ) <= 0)
                     del[i] = true;
      }
   tn = n;
   n = 0;
   for (int i = 0; i < tn; i++)
      if (del[i] == false)
         c[n++] = c[i];
}

//ans[i]i
const double pi = acos(-1.0);
const double eps = 1e-8;
struct Point
{
   double x,y;
   Point(){}
   Point(double _x,double _y)
   {
      x = _x;
      y = _y;
   }
   double Length()
   {
      return sqrt(x*x+y*y);
   }
};
struct Circle
{
   Point c;
   double r;
};
struct Event
{
   double tim;
   int typ;
   Event(){}
   Event(double _tim,int _typ)
   {
      tim = _tim;
      typ = _typ;
   }
};

int cmp(const double& a,const double& b)
{
   if (fabs(a-b) < eps) return 0;
   if (a < b) return -1;
   return 1;
}

bool Eventcmp(const Event& a,const Event& b)
{
   return cmp(a.tim,b.tim) < 0;
}

double Area(double theta,double r)
{
   return 0.5*r*r*(theta-sin(theta));
}

double xmult(Point a,Point b)
{
   return a.x*b.y-a.y*b.x;
}

int n,cur,tote;
Circle c[1000];
double ans[1001],pre[1001],AB,AC,BC,theta,fai,a0,a1;
Event e[4000];
Point lab;

int main()
{
   while (scanf("%d",&n) != EOF)
   {
      for (int i = 0;i < n;i++)
         scanf("%lf%lf%lf",&c[i].c.x,&c[i].c.y,&c[i].r);
      for (int i = 1;i <= n;i++)
         ans[i] = 0.0;
      for (int i = 0;i < n;i++)
      {
         tote = 0;
         e[tote++] = Event(-pi,1);
         e[tote++] = Event(pi,-1);
         for (int j = 0;j < n;j++)
            if (j != i)
            {
               lab = Point(c[j].c.x-c[i].c.x,c[j].c.y-c[i].c.y);
               AB = lab.Length();
               AC = c[i].r;
               BC = c[j].r;
               if (cmp(AB+AC,BC) <= 0)
               {
                  e[tote++] = Event(-pi,1);
                  e[tote++] = Event(pi,-1);
                  continue;
               }
               if (cmp(AB+BC,AC) <= 0) continue;
               if (cmp(AB,AC+BC) > 0) continue;
               theta = atan2(lab.y,lab.x);
               fai = acos((AC*AC+AB*AB-BC*BC)/(2.0*AC*AB));
               a0 = theta-fai;
               if (cmp(a0,-pi) < 0) a0 += 2*pi;
               a1 = theta+fai;
               if (cmp(a1,pi) > 0) a1 -= 2*pi;
               if (cmp(a0,a1) > 0)
               {
                  e[tote++] = Event(a0,1);
                  e[tote++] = Event(pi,-1);
                  e[tote++] = Event(-pi,1);
                  e[tote++] = Event(a1,-1);
               }
               else
               {
                  e[tote++] = Event(a0,1);
                  e[tote++] = Event(a1,-1);
               }
            }
         sort(e,e+tote,Eventcmp);
         cur = 0;
         for (int j = 0;j < tote;j++)
         {
            if (cur != 0 && cmp(e[j].tim,pre[cur]) != 0)
            {
               ans[cur] += Area(e[j].tim-pre[cur],c[i].r);
               ans[cur] += xmult(Point(c[i].c.x+c[i].r*cos(pre[cur
                     ]),c[i].c.y+c[i].r*sin(pre[cur])),
                     Point(c[i].c.x+c[i].r*cos(e[j].tim),c[i].c.y+
                           c[i].r*sin(e[j].tim)))/2.0;
            }
            cur += e[j].typ;
            pre[cur] = e[j].tim;
         }
      }
      for (int i = 1;i < n;i++)
         ans[i] -= ans[i+1];
      for (int i = 1;i <= n;i++)
         printf("[%d] = %.3f\n",i,ans[i]);
   }
   return 0;
}
```

3.5 circle

```
//
#include<cstdio>
#include<cmath>
#include<vector>
#include<algorithm>

#define MAXX 333
#define eps 1e-8

struct pv
{
   double x,y;
   pv(){}
   pv(const double &xx,const double &yy):x(xx),y(yy){}
   inline pv operator-(const pv &i)const
   {
      return pv(x-i.x,y-i.y);
   }
   inline double cross(const pv &i)const
   {
      return x*i.y-y*i.x;
   }
   inline void print()
   {
      printf("%lf %lf\n",x,y);
   }
   inline double len()
   {
      return sqrt(x*x+y*y);
   }
}pnt[MAXX];

struct node
{
   double k;
   bool flag;
   node(){}
   node(const double &kk,const bool &ff):k(kk),flag(ff){}
   inline bool operator<(const node &i)const
   {
      return k<i.k;
   }
};

std::vector<node>alpha;

short n,i,j,k,l;
short ans,sum;
double R=2;
double theta,phi,d;
const double pi(acos(-1.0));

int main()
{
   alpha.reserve(MAXX<<1);
   while(scanf("%hd",&n),n)
   {
      for(i=0;i<n;++i)
         scanf("%lf %lf",&pnt[i].x,&pnt[i].y);
      ans=0;
      for(i=0;i<n;++i)
      {
         alpha.resize(0);
         for(j=0;j<n;++j)
            if(i!=j)
            {
               if((d=(pnt[i]-pnt[j]).len())>R)
                  continue;
               if((theta=atan2(pnt[j].y-pnt[i].y,pnt[j].x-pnt[i].x
                        ))<0)
                  theta+=2*pi;
               phi=acos(d/R);
               alpha.push_back(node(theta-phi,true));
               alpha.push_back(node(theta+phi,false));
            }
         std::sort(alpha.begin(),alpha.end());
         for(j=0;j<alpha.size();++j)
         {
            if(alpha[j].flag)
               ++sum;
            else
               --sum;
            ans=std::max(ans,sum);
         }
      }
      printf("%hd\n",ans+1);
   }
   return 0;
}

//

#include<cstdio>
#include<cmath>

#define MAXX 511
#define eps 1e-8

struct pv
{
   double x,y;
   pv(){}
   pv(const double &xx,const double &yy):x(xx),y(yy){}
   inline pv operator-(const pv &i)const
   {
      return pv(x-i.x,y-i.y);
   }
   inline pv operator+(const pv &i)const
   {
      return pv(x+i.x,y+i.y);
   }
   inline double cross(const pv &i)const
   {
      return x*i.y-y*i.x;
   }
   inline double len()
   {
      return sqrt(x*x+y*y);
   }
   inline pv operator/(const double &a)const
   {
      return pv(x/a,y/a);
   }
   inline pv operator*(const double &a)const
   {
      return pv(x*a,y*a);
   }
}pnt[MAXX],o,tl,lt,aa,bb,cc,dd;

short n,i,j,k,l;
double r,u;

inline pv ins(const pv &a1,const pv &a2,const pv &b1,const pv &b2)
{
   tl=a2-a1;
   lt=b2-b1;
   u=(b1-a1).cross(lt)/(tl).cross(lt);
   return a1+tl*u;
}

inline pv get(const pv &a,const pv &b,const pv &c)
{
   aa=(a+b)/2;
   bb.x=aa.x-a.y+b.y;
   bb.y=aa.y+a.x-b.x;
   cc=(a+c)/2;
   dd.x=cc.x-a.y+c.y;
   dd.y=cc.y+a.x-c.x;
   return ins(aa,bb,cc,dd);
}

int main()
{
   while(scanf("%hd",&n),n)
   {
      for(i=0;i<n;++i)
         scanf("%lf %lf",&pnt[i].x,&pnt[i].y);
      o=pnt[0];
      r=0;
      for(i=1;i<n;++i)
         if((pnt[i]-o).len()>r+eps)
         {
            o=pnt[i];
            r=0;
            for(j=0;j<i;++j)
               if((pnt[j]-o).len()>r+eps)
               {
                  o=(pnt[i]+pnt[j])/2;
                  r=(o-pnt[j]).len();
                  for(k=0;k<j;++k)
                     if((o-pnt[k]).len()>r+eps)
                     {
                        o=get(pnt[i],pnt[j],pnt[k]);
                        r=(o-pnt[i]).len();
                     }
               }
         }
      printf("%.2lf %.2lf %.2lf\n",o.x,o.y,r);
   }
   return 0;
}

//
double dis(int x,int y)
{
   return sqrt((double)(x*x+y*y));
}

double area(int x1,int y1,int x2,int y2,double r1,double r2)
{
   double s=dis(x2-x1,y2-y1);
   if(r1+r2<s) return 0;
   else if(r2-r1>s) return PI*r1*r1;
   else if(r1-r2>s) return PI*r2*r2;
   double q1=acos((r1*r1+s*s-r2*r2)/(2*r1*s));
   double q2=acos((r2*r2+s*s-r1*r1)/(2*r2*s));
   return (r1*r1*q1+r2*r2*q2-r1*s*sin(q1));
}

//
{
   for (int i = 0; i < 3; i++)
      scanf("%lf%lf",&p[i].x,&p[i].y);
   tp = pv((p[0].x+p[1].x)/2,(p[0].y+p[1].y)/2);
   l[0] = Line(tp,pv(tp.x-(p[1].y-p[0].y),tp.y+(p[1].x-p[0].x)));
   tp = pv((p[0].x+p[2].x)/2,(p[0].y+p[2].y)/2);
   l[1] = Line(tp,pv(tp.x-(p[2].y-p[0].y),tp.y+(p[2].x-p[0].x)));
   tp = LineToLine(l[0],l[1]);
   r = pv(tp,p[0]).Length();
   printf("(%.6f,%.6f,%.6f)\n",tp.x,tp.y,r);
}

//
{
   for (int i = 0; i < 3; i++)
      scanf("%lf%lf",&p[i].x,&p[i].y);
   if (xmult(pv(p[0],p[1]),pv(p[0],p[2])) < 0)
      swap(p[1],p[2]);
   for (int i = 0; i < 3; i++)
      len[i] = pv(p[i],p[(i+1)%3]).Length();
   tr = (len[0]+len[1]+len[2])/2;
   r = sqrt((tr-len[0])*(tr-len[1])*(tr-len[2])/tr);
   for (int i = 0; i < 2; i++)
   {
      v = pv(p[i],p[i+1]);
      tv = pv(-v.y,v.x);
      tr = tv.Length();
      tv = pv(tv.x*r/tr,tv.y*r/tr);
      tp = pv(p[i].x+tv.x,p[i].y+tv.y);
      l[i].s = tp;
      tp = pv(p[i+1].x+tv.x,p[i+1].y+tv.y);
      l[i].e = tp;
   }
   tp = LineToLine(l[0],l[1]);
   printf("(%.6f,%.6f,%.6f)\n",tp.x,tp.y,r);
}
```

3.6 closest point pair

```
//1

struct Point {double x, y;} p[10], t[10];
bool cmpx(const Point& i, const Point& j) {return i.x < j.x;}
bool cmpy(const Point& i, const Point& j) {return i.y < j.y;}

double DnC(int L, int R)
{
   if (L >= R) return 1e9; //

   /* Divide */

   int M = (L + R) / 2;

   /* Conquer */

   double d = min(DnC(L,M), DnC(M+1,R));
   // if (d == 0.0) return d; //

   /* MergeYO(NlogN) */

   int N = 0; //
   for (int i=M; i>=L && p[M].x - p[i].x < d; --i) t[N++] = p[i];
   for (int i=M+1; i<=R && p[i].x - p[M].x < d; ++i) t[N++] = p[i
         ];
   sort(t, t+N, cmpy); // Quicksort O(NlogN)

   /* MergeO(N) */

   for (int i=0; i<N-1; ++i)
      for (int j=1; j<=2 && i+j<N; ++j)
         d = min(d, distance(t[i], t[i+j]));

   return d;
}

double closest_pair()
{
   sort(p, p+10, cmpx);
   return DnC(0, N-1);
}


//2

struct Point {double x, y;} p[10], t[10];
bool cmpx(const Point& i, const Point& j) {return i.x < j.x;}
bool cmpy(const Point& i, const Point& j) {return i.y < j.y;}

double DnC(int L, int R)
{
   if (L >= R) return 1e9; //

   /* Divide */

   int M = (L + R) / 2;

   // X
   double x = p[M].x;

   /* Conquer */

   // Y
   double d = min(DnC(L,M), DnC(M+1,R));
   // if (d == 0.0) return d; //

   /* MergeYO(N) */

   // Y
   int N = 0; //
   for (int i=0; i<=M; ++i)
      if (x - p[i].x < d)
         t[N++] = p[i];

   // Y
   int P = N; // P
   for (int i=M+1; i<=R; ++i)
      if (p[i].x - x < d)
         t[N++] = p[i];

   // YMerge Sort
   inplace_merge(t, t+P, t+N, cmpy);

   /* MergeO(N) */

   for (int i=0; i<N; ++i)
      for (int j=1; j<=2 && i+j<N; ++j)
         d = min(d, distance(t[i], t[i+j]));

   /* MergeYO(N) */

   // Merge Sort
   inplace_merge(p+L, p+M+1, p+R+1, cmpy);

   return d;
}

double closest_pair()
{
   sort(p, p+10, cmpx);
   return DnC(0, N-1);
}

//mzry
//
double calc_dis(Point &a ,Point &b) {
  return sqrt((a.x-b.x)*(a.x-b.x) + (a.y-b.y)*(a.y-b.y));
}
//
bool operator<(const Point &a ,const Point &b) {
  if(a.y != b.y) return a.x < b.x;
  return a.x < b.x;
}
double Gao(int l ,int r ,Point pnts[]) {
  double ret = inf;
  if(l == r) return ret;
  if(l+1 ==r) {
    ret = min(calc_dis(pnts[l],pnts[l+1]) ,ret);
    return ret;
  }
  if(l+2 ==r) {
    ret = min(calc_dis(pnts[l],pnts[l+1]) ,ret);
    ret = min(calc_dis(pnts[l],pnts[l+2]) ,ret);
    ret = min(calc_dis(pnts[l+1],pnts[l+2]) ,ret);
    return ret;
  }

  int mid = l+r>>1;
  ret = min (ret ,Gao(l ,mid,pnts));
  ret = min (ret , Gao(mid+1, r,pnts));

  for(int c = l ; c<=r; c++)
    for(int d = c+1; d <=c+7 && d<=r; d++) {
      ret = min(ret , calc_dis(pnts[c],pnts[d]));
    }
  return ret;
}

//
#include <iostream>
#include <cstdio>
#include <cstring>
#include <map>
#include <vector>
#include <cmath>
#include <algorithm>
#define Point pair<double,double>
using namespace std;

const int step[9][2] =
       {{-1,-1},{-1,0},{-1,1},{0,-1},{0,0},{0,1},{1,-1},{1,0},{1,1}};

int n,x,y,nx,ny;
map<pair<int,int>,vector<Point > > g;
vector<Point > tmp;
Point p[20000];
double tx,ty,ans,nowans;
vector<Point >::iterator it,op,ed;
pair<int,int> gird;
bool flag;

double Dis(Point p0,Point p1)
{
  return sqrt((p0.first-p1.first)*(p0.first-p1.first)+
      (p0.second-p1.second)*(p0.second-p1.second));
}

double CalcDis(Point p0,Point p1,Point p2)
{
  return Dis(p0,p1)+Dis(p0,p2)+Dis(p1,p2);
}

void build(int n,double w)
{
  g.clear();
  for (int i = 0;i < n;i++)
    g[make_pair((int)floor(p[i].first/w),(int)floor(p[i].second/w))
          ].push_back(p[i]);
}

int main()
{
  int t;
  scanf("%d",&t);
  for (int ft = 1;ft <= t;ft++)
  {
    scanf("%d",&n);
    for (int i = 0;i < n;i++)
    {
      scanf("%lf%lf",&tx,&ty);
      p[i] = make_pair(tx,ty);
    }
    random_shuffle(p,p+n);
    ans = CalcDis(p[0],p[1],p[2]);
    build(3,ans/2.0);
    for (int i = 3;i < n;i++)
    {
      x = (int)floor(2.0*p[i].first/ans);
      y = (int)floor(2.0*p[i].second/ans);
      tmp.clear();
      for (int k = 0;k < 9;k++)
      {
        nx = x+step[k][0];
        ny = y+step[k][1];
        gird = make_pair(nx,ny);
        if (g.find(gird) != g.end())
        {
          op = g[gird].begin();
          ed = g[gird].end();
          for (it = op;it != ed;it++)
            tmp.push_back(*it);
        }
      }
      flag = false;
      for (int j = 0;j < tmp.size();j++)
        for (int k = j+1;k < tmp.size();k++)
        {
          nowans = CalcDis(p[i],tmp[j],tmp[k]);
          if (nowans < ans)
          {
            ans = nowans;
            flag = true;
          }
        }
      if (flag == true)
        build(i+1,ans/2.0);
      else
        g[make_pair((int)floor(2.0*p[i].first/ans),(int)floor(2.0*p[
             i].second/ans))].push_back(p[i]);
    }
    printf("%.3f\n",ans);
  }
}
```

3.7 ellipse

```
sq(x-h)/sq(q) + sq(y-k)/sq(b) = 1

x=h+a*cos(t);
y=k+b*sin(t);

area: pi*a*b;
distance from center to focus: f=sqrt(sq(a)-sq(b));
eccentricity: e=sqrt(a-sq(b/a))=f/a;
focal parameter: sq(b)/sqrt(sq(a)-sq(b))=sq(b)/f;

double circumference(double a,double b) // accuracy: pow(0.5,53);
{
   double x=a;
   double y=b;
   if(x<y)
      std::swap(x,y);
   double digits=53,tol=sqrt(pow(0.5,digits));
   if(digits*y<tol*x)
      return 4*x;
   double s=0,m=1;
   while(x>(tol+1)*y)
   {
      double tx=x;
      double ty=y;
      x=0.5f*(tx+ty);
      y=sqrt(tx*ty);
      m*=2;
      s+=m*pow(x-y,2);
   }
   return pi*(pow(a+b,2)-s)/(x+y);
}
```

3.8 Graham's scan

```
pv pnt[MAXX];

inline bool com(const pv &a,const pv &b)
{
   if(fabs(t=(a-pnt[0]).cross(b-pnt[0]))>eps)
      return t>0;
   return (a-pnt[0]).len()<(b-pnt[0]).len();
}

inline void graham(std::vector<pv> &ch,const int n)
{
   std::nth_element(pnt,pnt,pnt+n);
   std::sort(pnt+1,pnt+n,com);
   ch.resize(0);
   ch.push_back(pnt[0]);
   ch.push_back(pnt[1]);
   static int i;
   for(i=2;i<n;++i)
      if(fabs((pnt[i]-ch[0]).cross(ch[1]-ch[0]))>eps)
      {
         ch.push_back(pnt[i++]);
         break;
      }
      else
         ch.back()=pnt[i];
   for(;i<n;++i)
   {
      while((ch.back()-ch[ch.size()-2]).cross(pnt[i]-ch[ch.size()
            -2])<eps)
         ch.pop_back();
      ch.push_back(pnt[i]);
   }
}
```

3.9 half-plane intersection

```
//abc
inline pv ins(const pv &p1,const pv &p2)
{
   u=fabs(a*p1.x+b*p1.y+c);
   v=fabs(a*p2.x+b*p2.y+c);
   return pv((p1.x*v+p2.x*u)/(u+v),(p1.y*v+p2.y*u)/(u+v));
}

inline void get(const pv& p1,const pv& p2,double & a,double & b, double & c)
{
   a=p2.y-p1.y;
   b=p1.x-p2.x;
   c=p2.x*p1.y-p2.y*p1.x;
}

inline pv ins(const pv &x,const pv &y)
{
   get(x,y,d,e,f);
   return pv((b*f-c*e)/(a*e-b*d),(a*f-c*d)/(b*d-a*e));
}

std::vector<pv>p[2];
inline bool go()
{
   k=0;
   p[k].resize(0);
   p[k].push_back(pv(-inf,inf));
   p[k].push_back(pv(-inf,-inf));
   p[k].push_back(pv(inf,-inf));
   p[k].push_back(pv(inf,inf));
   for(i=0;i<n;++i)
   {
      get(pnt[i],pnt[(i+1)%n],a,b,c);
      c+=the*sqrt(a*a+b*b);
      p[!k].resize(0);
      for(l=0;l<p[k].size();++l)
         if(a*p[k][l].x+b*p[k][l].y+c<eps)
            p[!k].push_back(p[k][l]);
         else
         {
            m=(l+p[k].size()-1)%p[k].size();
            if(a*p[k][m].x+b*p[k][m].y+c<-eps)
               p[!k].push_back(ins(p[k][m],p[k][l]));
            m=(l+1)%p[k].size();
            if(a*p[k][m].x+b*p[k][m].y+c<-eps)
               p[!k].push_back(ins(p[k][m],p[k][l]));
         }
      k=!k;
      if(p[k].empty())
         break;
   }
   //p[k]
   return p[k].empty();
}

//
//

inline pv ins(const pv &a,const pv &b)
{
   u=fabs(ln.cross(a-pnt[i]));
   v=fabs(ln.cross(b-pnt[i]))+u;
   tl=b-a;
   return pv(u*tl.x/v+a.x,u*tl.y/v+a.y);
}

int main()
{
   j=0;
   for(i=0;i<n;++i)
   {
      ln=pnt[(i+1)%n]-pnt[i];
      p[!j].resize(0);
      for(k=0;k<p[j].size();++k)
         if(ln.cross(p[j][k]-pnt[i])<=0)
            p[!j].push_back(p[j][k]);
         else
         {
            l=(k-1+p[j].size())%p[j].size();
            if(ln.cross(p[j][l]-pnt[i])<0)
               p[!j].push_back(ins(p[j][k],p[j][l]));
            l=(k+1)%p[j].size();
            if(ln.cross(p[j][l]-pnt[i])<0)
               p[!j].push_back(ins(p[j][k],p[j][l]));
         }
      j=!j;
   }
   //p[j]
}

//mrzy

bool HPIcmp(Line a, Line b)
{
   if (fabs(a.k - b.k) > eps)
      return a.k < b.k;
   return ((a.s - b.s) * (b.e-b.s)) < 0;
}

Line Q[100];

void HPI(Line line[], int n, Point res[], int &resn)
{
   int tot = n;
   std::sort(line, line + n, HPIcmp);
   tot = 1;
   for (int i = 1; i < n; i++)
      if (fabs(line[i].k - line[i - 1].k) > eps)
         line[tot++] = line[i];
   int head = 0, tail = 1;
   Q[0] = line[0];
   Q[1] = line[1];
   resn = 0;
   for (int i = 2; i < tot; i++)
   {
      if (fabs((Q[tail].e-Q[tail].s)*(Q[tail - 1].e-Q[tail - 1].s)) < eps || fabs((Q[head].e-Q[head].s)*(Q[head + 1].e-Q[head + 1].s)) < eps)
         return;
      while (head < tail && (((Q[tail]&Q[tail - 1]) - line[i].s) * (line[i].e-line[i].s)) > eps)
         --tail;
      while (head < tail && (((Q[head]&Q[head + 1]) - line[i].s) * (line[i].e-line[i].s)) > eps)
         ++head;
      Q[++tail] = line[i];
   }
   while (head < tail && (((Q[tail]&Q[tail - 1]) - Q[head].s) * (Q[head].e-Q[head].s)) > eps)
      tail--;
   while (head < tail && (((Q[head]&Q[head + 1]) - Q[tail].s) * (Q[tail].e-Q[tail].s)) > eps)
      head++;
   if (tail <= head + 1)
      return;
   for (int i = head; i < tail; i++)
      res[resn++] = Q[i] & Q[i + 1];
   if (head < tail + 1)
      res[resn++] = Q[head] & Q[tail];
}
```

3.10 kdtree

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <stack>
#include <algorithm>
using namespace std;
#define MAXN 100010
typedef long long ll;
struct Point{
   ll x,y;
   void operator =(const Point &p){
      x=p.x; y=p.y;
   }
   ll dis(const Point &a){
      return (x-a.x)*(x-a.x)+(y-a.y)*(y-a.y);
   }
}point[MAXN],pp[MAXN];

struct Node{
   int split;//{0,1} 0x1y
   Point p;//
}tree[MAXN*4];

bool cmpx(const Point &a,const Point &b)
{
   return a.x<b.x;
}

bool cmpy(const Point &a,const Point &b)
{
   return a.y<b.y;
}

void initTree(int x,int y,int split,int pos)
{
   if(y<x) return ;
   int mid=(x+y)>>1;
   random_shuffle(point+x,point+y);
   if(split==0) nth_element(point+x,point+mid,point+y+1,cmpx);
   else nth_element(point+x,point+mid,point+y+1,cmpy);
   tree[pos].split=split;
   tree[pos].p=point[mid];
   initTree(x,mid-1,(split^1),2*pos);
   initTree(mid+1,y,(split^1),2*pos+1);
}

ll ans;
void insert(int x,int y,Point &p,int pos)
{
   if(y<x) return ;
   int mid=(x+y)>>1;
   ll temp=p.dis(tree[pos].p);
   if(temp!=0) ans=min(ans,temp);
   if(tree[pos].split==0){
      if(p.x<=tree[pos].p.x){
         insert(x,mid-1,p,2*pos);
         if(ans>=(p.x-tree[pos].p.x)*(p.x-tree[pos].p.x))
            insert(mid+1,y,p,2*pos+1);
      }
      else{
         insert(mid+1,y,p,2*pos+1);
         if(ans>=(p.x-tree[pos].p.x)*(p.x-tree[pos].p.x))
            insert(x,mid-1,p,2*pos);
      }
   }
   else
   {
      if(p.y<=tree[pos].p.y){
         insert(x,mid-1,p,2*pos);
         if(ans>=(p.y-tree[pos].p.y)*(p.y-tree[pos].p.y))
            insert(mid+1,y,p,2*pos+1);
      }
      else{
         insert(mid+1,y,p,2*pos+1);
         if(ans>=(p.y-tree[pos].p.y)*(p.y-tree[pos].p.y))
            insert(x,mid-1,p,2*pos);
      }
   }
}

int main()
{
   int cases,n;
   scanf("%d",&cases);
   while(cases--)
   {
      scanf("%d",&n);
      for(int i=1;i<=n;i++){
         scanf("%I64d%I64d",&pp[i].x,&pp[i].y);
         point[i]=pp[i];
      }
      initTree(1,n,0,1);
      for(int i=1;i<=n;i++){
         ans=1LL<<62;
         insert(1,n,pp[i],1);
         printf("%I64d\n",ans);
      }
   }
   return 0;
}
```

3.11 Manhattan minimum spanning tree

```
#include <cstdio>
#include <algorithm>
#include <cstring>
#include <iostream>

using namespace std;

const int maxn = 60000;

struct node {int x, y, k[2];} b[maxn];
struct bian {int a, b, c;} g[maxn * 8];
struct point{int k[2];} d[maxn * 8];
long long s[maxn], ans;
int i, n, m, a[maxn], lim, h, mid, bh[maxn * 2], f[maxn], num, e[maxn * 2], next[maxn * 2], first[maxn], tot;
int comx(int p, int q) {return b[p].x < b[q].x;}
int comy(int p, int q) {return b[p].y < b[q].y;}
int comc(const bian &p, const bian &q) {return p.c < q.c;}
int dist(int p, int q) {return abs(b[p].x - b[q].x) + abs(b[p].y - b[q].y);}
int maxbh(int p, int q, int k) {return b[p].k[k] > b[q].k[k] ? p : q;}
int minbh(int p, int q, int k) {return b[p].k[k] < b[q].k[k] ? p : q;}
int getfa(int x) {if (f[x] != x) f[x] = getfa(f[x]); return f[x];}
long long gcd(long long p, long long q) {return (!p || !q) ? p + q : gcd(q, p % q);}
void link(int u, int v)
{
   e[++num] = v, next[num] = first[u], first[u] = num;
   e[++num] = u, next[num] = first[v], first[v] = num;
}
void add(int x, int k)
{
   int y = h + b[x].k[1]; d[y].k[0] = minbh(d[y].k[0], x, 0);
   for (y >>= 1; y; y >>= 1) d[y].k[0] = minbh(d[y << 1].k[0], d[y << 1 ^ 1].k[0], 0);
   y = h + b[x].k[0];
   d[y].k[1] = k ? maxbh(x, d[y].k[1], 1) : minbh(d[y].k[1], x, 1);
   for (y >>= 1; y; y >>= 1)
      d[y].k[1] = k ? maxbh(x, d[y << 1].k[1], 1) : minbh(d[y << 1 ^ 1].k[1], x, 1);
}
int ask(int l, int r, int k, int boss)
{
   for (mid = 0, l += h - 1, r += h + 1; (l ^ r) != 1; l >>= 1, r >>= 1)
   {
      if (!(l & 1)) mid = boss ? maxbh(mid, d[l + 1].k[k], k) : minbh(mid, d[r - 1].k[k], k);
      if (r & 1) mid = boss ? maxbh(mid, d[r - 1].k[k], k) : minbh(mid, d[r - 1].k[k], k);
   } return mid;
}
void manhattan()
{
   sort(bh + 1, bh + m + 1, comx);
   b[0].k[0] = maxn * 3, b[0].k[1] = -1;
   for (add(bh[m], 1), i = m - 1; i; add(bh[i], 1), --i)
   {
      g[++tot].a = bh[i], g[tot].b = ask(b[bh[i]].k[1], lim, 0, 0);
      g[tot].c = dist(g[tot].a, g[tot].b);
      if (g[tot].b == 0) --tot;
      g[++tot].a = bh[i], g[tot].b = ask(1, b[bh[i]].k[0], 1, 1);
      g[tot].c = dist(g[tot].a, g[tot].b);
      if (g[tot].b == 0) --tot;
   }
   b[0].k[1] = b[0].k[0];
   memset(d, 0, sizeof(d));
   sort(bh + 1, bh + m + 1, comy);
   for (add(bh[m], 0), i = m - 1; i; add(bh[i], 0), --i)
   {
      g[++tot].a = bh[i], g[tot].b = ask(1, b[bh[i]].k[1], 0, 0);
      g[tot].c = dist(g[tot].a, g[tot].b);
      if (g[tot].b == 0) --tot;
      g[++tot].a = bh[i], g[tot].b = ask(1, b[bh[i]].k[0], 1, 0);
      g[tot].c = dist(g[tot].a, g[tot].b);
      if (g[tot].b == 0) --tot;
   }
}
void kruskal()
{
   sort(g + 1, g + tot + 1, comc);
   for (i = 1; i <= tot; ++i)
   {
      int f1 = getfa(g[i].a), f2 = getfa(g[i].b);
      if (f1 != f2) link(g[i].a, g[i].b), f[f1] = f2;
   } tot = 0; memset(f, 0, sizeof(f));
}
void dfs(int x, int fa)
{
   bh[++tot] = x;
   for (int p = first[x]; p; p = next[p])
      if (e[p] != fa) dfs(e[p], x), bh[++tot] = x;
}
void del(int l, int r)
{
   if (l > r) return ;
   for (int j = l; j <= r; ++j)
      ans -= 1LL * f[a[j]] * f[a[j]], ans += 1LL * (--f[a[j]]) * f[a[j]];
}
void ins(int l, int r)
{
   if (l > r) return ;
   for (int j = l; j <= r; ++j)
      ans -= 1LL * f[a[j]] * f[a[j]], ans += 1LL * (++f[a[j]]) * f[a[j]];
}
int main()
{
   freopen("hose.in", "r", stdin);
   freopen("hose.out", "w", stdout);
   scanf("%d%d", &n, &m);
   for (i = 1; i <= n; ++i)
      scanf("%d", a+i);
   for (i = 1; i <= m; ++i)
   {
      scanf("%d%d", &b[bh[i] = f[i] = i].x, &b[i].y);
      b[i].k[0] = b[i].x + b[i].y;
      b[i].k[1] = b[i].y - b[i].x + maxn;
      lim = max(lim, max(b[i].k[0], b[i].k[1]));
   }
   for (h = 1; h <= lim; h <<= 1);
   manhattan();
   kruskal();
   dfs(1, 0);
   ins(b[bh[1]].x, b[bh[1]].y);
   for (s[1] = ans, i = 2; i <= tot; s[bh[i]] = ans, ++i)
   {
      ins(b[bh[i]].x, b[bh[i - 1]].x - 1);
      ins(b[bh[i - 1]].y + 1, b[bh[i]].y);
      del(b[bh[i - 1]].x, b[bh[i]].x - 1);
      del(b[bh[i]].y + 1, b[bh[i - 1]].y);
   }
   for (i = 1; i <= m; ++i)
   {
      long long fz = s[i] - b[i].k[1] - 1 + maxn, fm = 1LL * (b[i].k[1] + 1 - maxn) * (b[i].k[1] - maxn);
      long long gys = gcd(fz, fm);
      printf("%lld/%lld\n", fz/gys, fm/gys);
   }
   return 0;
}







#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cmath>
#include<cstring>
#define maxn 55000
#define inf 2147483647
using namespace std;
struct query
{
   int l,r,s,w;
}a[maxn];
int c[maxn];
long long col[maxn],size[maxn],ans[maxn];
int n,m,cnt,len;

long long gcd(long long x,long long y)
{
   return (!x)?y:gcd(y%x,x);
}

bool cmp(query a,query b)
{
   return (a.w==b.w)?a.r<b.r:a.w<b.w;
}

int main()
{
   //freopen("hose.in","r",stdin);
   scanf("%d%d",&n,&m);
   for (int i=1;i<=n;i++) scanf("%d",&c[i]);
   len=(int)sqrt(m);
   cnt=(len*len==m)?len:len+1;
   for (int i=1;i<=m;i++)
   {
      scanf("%d%d",&a[i].l,&a[i].r);
      if (a[i].l>a[i].r) swap(a[i].l,a[i].r);
      size[i]=a[i].r-a[i].l+1;
      a[i].w=a[i].l/len+1;
      a[i].s=i;
   }
   sort(a+1,a+m+1,cmp);
   int i=1;
   while (i<=m)
   {
      int now=a[i].w;
      memset(col,0,sizeof(col));
      for (int j=a[i].l;j<=a[i].r;j++) ans[a[i].s]+=2*(col[c[j]]++);
      i++;
      for (;a[i].w==now;i++)
      {
         ans[a[i].s]=ans[a[i-1].s];
         for (int j=a[i-1].r+1;j<=a[i].r;j++)
            ans[a[i].s]+=2*(col[c[j]]++);
         if (a[i-1].l<a[i].l)
            for (int j=a[i-1].l;j<a[i].l;j++)
               ans[a[i].s]-=2*(--col[c[j]]);
         else
            for (int j=a[i].l;j<a[i-1].l;j++)
               ans[a[i].s]+=2*(col[c[j]]++);
      }
   }
   long long all,num;
   for (int i=1;i<=m;i++)
   {
      if (size[i]==1) all=1; else all=size[i]*(size[i]-1);
      num=gcd(ans[i],all);
      printf("%lld/%lld\n",ans[i]/num,all/num);
   }
   return 0;
}
```

3.12 others

```
eps

sqrt(a), asin(a), acos(a) aa0-1e-12sqrt(a)0aa1,asin(a)acos(a)a

case0:005,0:010:005000000001()0:004999999999()printf("%.2lf", a)
aa + eps, a - eps

-0.000

double

a==b fabs(a-b)<eps
a!=b fabs(a-b)>eps
a<b a+eps<b
a<=b a<b+eps
a>b a>b+eps
a>=b a+eps>b



cos/sin/tan
acos [-1,+1][0,]
asin [-1,+1][-/2,+/2]
atan [-/2,+/2]
atan2 (y,x)(),tan(y/x),[-,+]xy

other

log (ln)
log10
ceil
floor

round

cpp:
java: add 0.5,then floor
cpp:
4
6
5050
55050

rotate mat:
[ cos(theta) -sin(theta) ]
[ sin(theta) cos(theta) ]
```

3.13 Pick's theorem

```


A:
i:
b:
A = i + b/2 - 1



A = 2i + b - 2
```

3.14 PointInPoly

```
/*
,
poly3

0 -- poly
1 -- poly
2 -- poly
*/

int inPoly(pv p,pv poly[], int n)
{
  int i, count;
  Line ray, side;

  count = 0;
  ray.s = p;
  ray.e.y = p.y;
  ray.e.x = -1; //-INF

  for (i = 0; i < n; i++)
  {
   side.s = poly[i];
   side.e = poly[(i+1)%n];

   if(OnSeg(p, side))
     return 1;

   // sidex
   if (side.s.y == side.e.y)
    continue;

      if (OnSeg(side.s, ray))
      {
         if (side.s.y > side.e.y)
            count++;
      }
      else
         if (OnSeg(side.e, ray))
         {
            if (side.e.y > side.s.y)
               count++;
         }
         else
            if (inter(ray, side))
               count++;
  }
  return ((count % 2 == 1) ? 0 : 2);
}
```

3.15 rotating caliper

```
//

inline double go()
{
   l=ans=0;
   for(i=0;i<n;++i)
   {
      tl=pnt[(i+1)%n]-pnt[i];
      while(abs(tl.cross(pnt[(l+1)%n]-pnt[i]))>=abs(tl.cross(pnt[l
           ]-pnt[i])))
         l=(l+1)%n;
      ans=std::max(ans,std::max(dist(pnt[l],pnt[i]),dist(pnt[l],
           pnt[(i+1)%n])));
   }
   return ans;
}

//
double go()
{
   sq=sp=0;
   for(i=1;i<ch[1].size();++i)
      if(ch[1][sq]<ch[1][i])
         sq=i;
   tp=sp;
   tq=sq;
   ans=(ch[0][sp]-ch[1][sq]).len();
   do
   {
      a1=ch[0][sp];
      a2=ch[0][(sp+1)%ch[0].size()];
      b1=ch[1][sq];
      b2=ch[1][(sq+1)%ch[1].size()];
      tpv=b1-(b2-a1);
      tpv.x = b1.x - (b2.x - a1.x);
      tpv.y = b1.y - (b2.y - a1.y);
      len=(tpv-a1).cross(a2-a1);
      if(fabs(len)<eps)
      {
         ans=std::min(ans,p2l(a1,b1,b2));
         ans=std::min(ans,p2l(a2,b1,b2));
         ans=std::min(ans,p2l(b1,a1,a2));
         ans=std::min(ans,p2l(b2,a1,a2));
         sp=(sp+1)%ch[0].size();
         sq=(sq+1)%ch[1].size();
      }
      else
         if(len<-eps)
         {
            ans=std::min(ans,p2l(b1,a1,a2));
            sp=(sp+1)%ch[0].size();
         }
         else
         {
            ans=std::min(ans,p2l(a1,b1,b2));
            sq=(sq+1)%ch[1].size();
         }
   }while(tp!=sp || tq!=sq);
   return ans;
}

// by mzry
inline void solve()
{
   resa = resb = 1e100;
   double dis1,dis2;
   Point xp[4];
   Line l[4];
   int a,b,c,d;
   int sa,sb,sc,sd;
   a = b = c = d = 0;
   sa = sb = sc = sd = 0;
   Point va,vb,vc,vd;
   for (a = 0; a < n; a++)
   {
      va = Point(p[a],p[(a+1)%n]);
      vc = Point(-va.x,-va.y);
      vb = Point(-va.y,va.x);
      vd = Point(-vb.x,-vb.y);
      if (sb < sa)
      {
         b = a;
         sb = sa;
      }
      while (xmult(vb,Point(p[b],p[(b+1)%n])) < 0)
      {
         b = (b+1)%n;
         sb++;
      }
      if (sc < sb)
      {
         c = b;
         sc = sb;
      }
      while (xmult(vc,Point(p[c],p[(c+1)%n])) < 0)
      {
         c = (c+1)%n;
         sc++;
      }
      if (sd < sc)
      {
         d = c;
         sd = sc;
      }
      while (xmult(vd,Point(p[d],p[(d+1)%n])) < 0)
      {
         d = (d+1)%n;
         sd++;
      }

      //`p[a],p[b],p[c],p[d]`
      sa++;
   }
}

//
 P = { p(1) , ... , p(m) } Q = { q(1) , ... , q(n) } (p(i), q(j)) P Q

(p(i), q(j))
p(i-1), p(i+1), q(j-1), q(j+1) (p(i), q(j))



1 P Q y x
2 x
3 (p(i), q(j))
4 (p(i), q(j)) p(i-1), p(i+1), q(j-1), q(j+1) (p(i), q(j))
534
6

 156 O(N) N



//
1 P y yminP Q y ymaxQ
2 yminP ymaxQ LP LQ LP LQ yminP ymaxQ
3 p(i)= yminP q(j)= ymaxQ (p(i), q(j)) p(i-1),p(i+1) (p(i), q(j)) q(j-1),q(j+1) (p(i), q(j)) CS
4
5
645 (yminP,ymaxQ)
7CS

////
1 xminP xmaxP yminP ymaxP
2 P
3
4
5/
645 90
7
```

3.16 shit

```
struct pv
{
   double x,y;
   pv():x(0),y(0){}
   pv(double xx,double yy):x(xx),y(yy){}
   inline pv operator+(const pv &i)const
   {
      return pv(x+i.x,y+i.y);
   }
   inline pv operator-(const pv &i)const
   {
      return pv(x-i.x,y-i.y);
   }
   inline bool operator ==(const pv &i)const
   {
      return fabs(x-i.x)<eps && fabs(y-i.y)<eps;
   }
   inline bool operator<(const pv &i)const
   {
      return y==i.y?x<i.x:y<i.y;
   }
   inline double cross(const pv &i)const
   {
      return x*i.y-y*i.x;
   }
   inline double dot(const pv &i)const
   {
      return x*i.x+y*i.y;
   }
   inline double len()
   {
      return sqrt(x*x+y*y);
   }
   inline pv rotate(pv p,double theta)
   {
      static pv v;
      v=*this-p;
      static double c,s;
      c=cos(theta);
      s=sin(theta);
      return pv(p.x+v.x*c-v.y*s,p.y+v.x*s+v.y*c);
   }
};

inline int dblcmp(double d)
{
   if(fabs(d)<eps)
      return 0;
   return d>eps?1:-1;
}

inline int cross(pv *a,pv *b) // 0 1 2
{
   int d1=dblcmp((a[1]-a[0]).cross(b[0]-a[0]));
   int d2=dblcmp((a[1]-a[0]).cross(b[1]-a[0]));
   int d3=dblcmp((b[1]-b[0]).cross(a[0]-b[0]));
   int d4=dblcmp((b[1]-b[0]).cross(a[1]-b[0]));
   if((d1^d2)==-2 && (d3^d4)==-2)
      return 2;
   return ((d1==0 && dblcmp((b[0]-a[0]).dot(b[0]-a[1]))<=0 )||
         (d2==0 && dblcmp((b[1]-a[0]).dot(b[1]-a[1]))<=0 )||
         (d3==0 && dblcmp((a[0]-b[0]).dot(a[0]-b[1]))<=0 )||
         (d4==0 && dblcmp((a[1]-b[0]).dot(a[1]-b[1]))<=0));
}

inline bool pntonseg(const pv &p,const pv *a)
{
   return fabs((p-a[0]).cross(p-a[1]))<eps && (p-a[0]).dot(p-a[1])<eps;
}

pv rotate(pv v,pv p,double theta,double sc=1) // rotate vector v, theta [0,2]
{
   static pv re;
   re=p;
   v=v-p;
   p.x=sc*cos(theta);
   p.y=sc*sin(theta);
   re.x+=v.x*p.x-v.y*p.y;
   re.y+=v.x*p.y+v.y*p.x;
   return re;
}

struct line
{
   pv pnt[2];
   line(double a,double b,double c) // a*x + b*y + c = 0
   {
#define maxl 1e2 //preciseness should not be too high ( compare with eps )
      if(fabs(b)>eps)
      {
         pnt[0]=pv(maxl,(c+a*maxl)/(-b));
         pnt[1]=pv(-maxl,(c-a*maxl)/(-b));
      }
      else
      {
         pnt[0]=pv(-c/a,maxl);
         pnt[1]=pv(-c/a,-maxl);
      }
#undef maxl
   }
   pv cross(const line &v)const
   {
      double a=(v.pnt[1]-v.pnt[0]).cross(pnt[0]-v.pnt[0]);
      double b=(v.pnt[1]-v.pnt[0]).cross(pnt[1]-v.pnt[0]);
      return pv((pnt[0].x*b-pnt[1].x*a)/(b-a),(pnt[0].y*b-pnt[1].y*a)/(b-a));
   }
};

inline std::pair<pv,double> getcircle(const pv &a,const pv &b,const pv &c)
{
   static pv ct;
   ct=line(2*(b.x-a.x),2*(b.y-a.y),a.len()-b.len()).cross(line(2*(c.x-b.x),2*(c.y-b.y),b.len()-c.len()));
   return std::make_pair(ct,sqrt((ct-a).len()));
}
```

3.17 sort - polar angle

```
inline bool cmp(const Point& a,const Point& b)
{
   if (a.y*b.y <= 0)
   {
      if (a.y > 0 || b.y > 0)
         return a.y < b.y;
      if (a.y == 0 && b.y == 0)
         return a.x < b.x;
   }
   return a.cross(b) > 0;
}
```

3.18 triangle

```
Area:
p=(a+b+c)/2
area=sqrt(p*(p-a)*(p-b)*(p-c));
area=a*b*sin(C)/2;
area=sq(a)*sin(B)*sin(C)/2/sin(B+C);
area=sq(a)/2/(cot(B)+cot(C));

centroid:
   center of mass
   intersection of triangle's three triangle medians

Trigonometric conditions:
tan(A/2)*tan(B/2)+tan(B/2)*tan(C/2)+tan(A/2)*tan(C/2)==1
sq(sin(A/2))+sq(sin(B/2))+sq(sin(C/2))+2*sin(A/2)*sin(B/2)*sin(C/2)==1

Circumscribed circle:
diameter=a*b*c/(2*area);
diameter=sqrt(2*area/sin(A)/sin(B)/sin(c));
diameter=a/sin(A)=b/sin(B)=c/sin(C);
```

```

Incircle:
inradius=2*area/(a+b+c);
coordinates(x,y)=a*{xa,ya}/(a+b+c)+b*{xb,yb}/(a+b+c)+c*{xc,yc}/(a+
       b+c);

Excircles:
radius[a]=2*area/(b+c-a);
radius[b]=2*area/(a+c-b);
radius[c]=2*area/(a+b-c);

Steiner circumellipse (least area circumscribed ellipse)
   area= area * 4*pi/3/sqrt(3);
   center is the triangle's centroid.

Steiner inellipse ( maximum area inellipse )
   area= area * pi/3/sqrt(3);
   center is the triangle's centroid.

Fermat Point:
120

120

ABC'BCA'CAB'
CC'BB'AA'
```

4 geometry/tmp

4.1 circle

```
struct circle
{
   point p;
   double r;
   circle(){}
   circle(point _p,double _r):
   p(_p),r(_r){};
   circle(double x,double y,double _r):
   p(point(x,y)),r(_r){};
   circle(point a,point b,point c)//
   {
    p=line(a.add(b).div(2),a.add(b).div(2).add(b.sub(a).rotleft()
           )).crosspoint(line(c.add(b).div(2),c.add(b).div(2).add(
           b.sub(c).rotleft())));
    r=p.distance(a);
   }
   circle(point a,point b,point c,bool t)//
   {
    line u,v;
    double m=atan2(b.y-a.y,b.x-a.x),n=atan2(c.y-a.y,c.x-a.x);
    u.a=a;
   u.b=u.a.add(point(cos((n+m)/2),sin((n+m)/2)));
    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=v.a.add(point(cos((n+m)/2),sin((n+m)/2)));
    p=u.crosspoint(v);
    r=line(a,b).dispointtoseg(p);
   }
   void input()
   {
      p.input();
      scanf("%lf",&r);
   }
   void output()
   {
      printf("%.2lf %.2lf %.2lf\n",p.x,p.y,r);
   }
   bool operator==(circle v)
   {
    return ((p==v.p)&&dblcmp(r-v.r)==0);
   }
   bool operator<(circle v)const
   {
    return ((p<v.p)||(p==v.p)&&dblcmp(r-v.r)<0);
   }
   double area()
   {
    return pi*sqr(r);
   }
   double circumference()
   {
    return 2*pi*r;
   }
   //0
   //1
   //2
   int relation(point b)
   {
      double dst=b.distance(p);
      if (dblcmp(dst-r)<0)return 2;
      if (dblcmp(dst-r)==0)return 1;
      return 0;
   }
   int relationseg(line v)
   {
      double dst=v.dispointtoseg(p);
      if (dblcmp(dst-r)<0)return 2;
      if (dblcmp(dst-r)==0)return 1;
      return 0;
   }
   int relationline(line v)
   {
      double dst=v.dispointtoline(p);
      if (dblcmp(dst-r)<0)return 2;
      if (dblcmp(dst-r)==0)return 1;
      return 0;
   }
   //a b r
   int getcircle(point a,point b,double r,circle&c1,circle&c2)
   {
      circle x(a,r),y(b,r);
      int t=x.pointcrosscircle(y,c1.p,c2.p);
   if (!t)return 0;
      c1.r=c2.r=r;
      return t;
   }
   //u q r1
   int getcircle(line u,point q,double r1,circle &c1,circle &c2)
   {
    double dis=u.dispointtoline(q);
    if (dblcmp(dis-r1*2)>0)return 0;
    if (dblcmp(dis)==0)
    {
      c1.p=q.add(u.b.sub(u.a).rotleft().trunc(r1));
      c2.p=q.add(u.b.sub(u.a).rotright().trunc(r1));
      c1.r=c2.r=r1;
      return 2;
    }
    line u1=line(u.a.add(u.b.sub(u.a).rotleft().trunc(r1)),u.b.
           add(u.b.sub(u.a).rotleft().trunc(r1)));
    line u2=line(u.a.add(u.b.sub(u.a).rotright().trunc(r1)),u.b.
           add(u.b.sub(u.a).rotright().trunc(r1)));
   circle cc=circle(q,r1);
    point p1,p2;
    if (!cc.pointcrossline(u1,p1,p2))cc.pointcrossline(u2,p1,p2);
    c1=circle(p1,r1);
    if (p1==p2)
   {
    c2=c1;return 1;
    }
    c2=circle(p2,r1);
    return 2;
  }
  //u,v r1
   int getcircle(line u,line v,double r1,circle &c1,circle &c2,
          circle &c3,circle &c4)
   {
    if (u.parallel(v))return 0;
    line u1=line(u.a.add(u.b.sub(u.a).rotleft().trunc(r1)),u.b.
           add(u.b.sub(u.a).rotleft().trunc(r1)));
    line u2=line(u.a.add(u.b.sub(u.a).rotright().trunc(r1)),u.b.
           add(u.b.sub(u.a).rotright().trunc(r1)));
    line v1=line(v.a.add(v.b.sub(v.a).rotleft().trunc(r1)),v.b.
           add(v.b.sub(v.a).rotleft().trunc(r1)));
    line v2=line(v.a.add(v.b.sub(v.a).rotright().trunc(r1)),v.b.
           add(v.b.sub(v.a).rotright().trunc(r1)));
    c1.r=c2.r=c3.r=c4.r=r1;
    c1.p=u1.crosspoint(v1);
    c2.p=u1.crosspoint(v2);
    c3.p=u2.crosspoint(v1);
    c4.p=u2.crosspoint(v2);
    return 4;
  }
  //cx,cy r1
  int getcircle(circle cx,circle cy,double r1,circle&c1,circle&c2)
   {
      circle x(cx.p,r1+cx.r),y(cy.p,r1+cy.r);
      int t=x.pointcrosscircle(y,c1.p,c2.p);
   if (!t)return 0;
      c1.r=c2.r=r1;
      return t;
   }
   int pointcrossline(line v,point &p1,point &p2)//relationseg
   {
      if (!(*this).relationline(v))return 0;
      point a=v.lineprog(p);
      double d=v.dispointtoline(p);
      d=sqrt(r*r-d*d);
      if (dblcmp(d)==0)
      {
         p1=a;
         p2=a;
         return 1;
      }
      p1=a.sub(v.b.sub(v.a).trunc(d));
      p2=a.add(v.b.sub(v.a).trunc(d));
      return 2;
   }
   //5
  //4
  //3
  //2
  //1
   int relationcircle(circle v)
   {
    double d=p.distance(v.p);
    if (dblcmp(d-r-v.r)>0)return 5;
    if (dblcmp(d-r-v.r)==0)return 4;
    double l=fabs(r-v.r);
    if (dblcmp(d-r-v.r)<0&&dblcmp(d-l)>0)return 3;
    if (dblcmp(d-l)==0)return 2;
    if (dblcmp(d-l)<0)return 1;
   }
   int pointcrosscircle(circle v,point &p1,point &p2)
   {
    int rel=relationcircle(v);
    if (rel==1||rel==5)return 0;
    double d=p.distance(v.p);
    double l=(d+(sqr(r)-sqr(v.r))/d)/2;
    double h=sqrt(sqr(r)-sqr(l));
    p1=p.add(v.p.sub(p).trunc(l).add(v.p.sub(p).rotleft().trunc(h
           )));
    p2=p.add(v.p.sub(p).trunc(l).add(v.p.sub(p).rotright().trunc(
           h)));
    if (rel==2||rel==4)
    {
      return 1;
    }
   return 2;
   }
   // ()
   int tangentline(point q,line &u,line &v)
   {
    int x=relation(q);
    if (x==2)return 0;
    if (x==1)
    {
      u=line(q,q.add(q.sub(p).rotleft()));
      v=u;
      return 1;
   }
   double d=p.distance(q);
   double l=sqr(r)/d;
   double h=sqrt(sqr(r)-sqr(l));
   u=line(q,p.add(q.sub(p).trunc(l).add(q.sub(p).rotleft().trunc(h))));
   v=line(q,p.add(q.sub(p).trunc(l).add(q.sub(p).rotright().trunc(h))));
   return 2;
 }
  double areacircle(circle v)
  {
   int rel=relationcircle(v);
   if (rel>=4)return 0.0;
    if (rel<=2)return min(area(),v.area());
    double d=p.distance(v.p);
    double hf=(r+v.r+d)/2.0;
    double ss=2*sqrt(hf*(hf-r)*(hf-v.r)*(hf-d));
    double a1=acos((r*r+d*d-v.r*v.r)/(2.0*r*d));
    a1=a1*r*r;
    double a2=acos((v.r*v.r+d*d-r*r)/(2.0*v.r*d));
    a2=a2*v.r*v.r;
    return a1+a2-ss;
  }
  double areatriangle(point a,point b)
  {
    if (dblcmp(p.sub(a).det(p.sub(b))==0))return 0.0;
    point q[5];
    int len=0;
    q[len++]=a;
    line l(a,b);
    point p1,p2;
    if (pointcrossline(l,q[1],q[2])==2)
    {
      if (dblcmp(a.sub(q[1]).dot(b.sub(q[1])))<0)q[len++]=q[1];
      if (dblcmp(a.sub(q[2]).dot(b.sub(q[2])))<0)q[len++]=q[2];
    }
    q[len++]=b;
    if (len==4&&(dblcmp(q[0].sub(q[1]).dot(q[2].sub(q[1])))>0))swap(q[1],q[2]);
    double res=0;
    int i;
    for (i=0;i<len-1;i++)
    {
      if (relation(q[i])==0||relation(q[i+1])==0)
      {
        double arg=p.rad(q[i],q[i+1]);
        res+=r*r*arg/2.0;
      }
      else
      {
        res+=fabs(q[i].sub(p).det(q[i+1].sub(p))/2.0);
      }
    }
    return res;
  }
};
```

4.2 circles

```
const int maxn=500;
struct circles
{
 circle c[maxn];
 double ans[maxn];//ans[i]i
 double pre[maxn];
 int n;
 circles(){}
 void add(circle cc)
 {
  c[n++]=cc;
 }
 bool inner(circle x,circle y)
 {
  if (x.relationcircle(y)!=1)return 0;
  return dblcmp(x.r-y.r)<=0?1:0;
 }
 void init_or()//
 {
  int i,j,k=0;
  bool mark[maxn]={0};
  for (i=0;i<n;i++)
  {
   for (j=0;j<n;j++)if (i!=j&&!mark[j])
   {
    if ((c[i]==c[j])||inner(c[i],c[j]))break;
   }
   if (j<n)mark[i]=1;
  }
  for (i=0;i<n;i++)if (!mark[i])c[k++]=c[i];
  n=k;
 }
 void init_and()//
 {
  int i,j,k=0;
  bool mark[maxn]={0};
  for (i=0;i<n;i++)
  {
   for (j=0;j<n;j++)if (i!=j&&!mark[j])
   {
    if ((c[i]==c[j])||inner(c[j],c[i]))break;
   }
   if (j<n)mark[i]=1;
  }
  for (i=0;i<n;i++)if (!mark[i])c[k++]=c[i];
  n=k;
 }
 double areaarc(double th,double r)
  {
   return 0.5*sqr(r)*(th-sin(th));
  }
 void getarea()
 {
  int i,j,k;
  memset(ans,0,sizeof(ans));
  vector<pair<double,int> >v;
  for (i=0;i<n;i++)
  {
   v.clear();
   v.push_back(make_pair(-pi,1));
   v.push_back(make_pair(pi,-1));
   for (j=0;j<n;j++)if (i!=j)
   {
    point q=c[j].p.sub(c[i].p);
    double ab=q.len(),ac=c[i].r,bc=c[j].r;
    if (dblcmp(ab+ac-bc)<=0)
     {
       v.push_back(make_pair(-pi,1));
     v.push_back(make_pair(pi,-1));
        continue;
      }
     if (dblcmp(ab+bc-ac)<=0)continue;
     if (dblcmp(ab-ac-bc)>0) continue;
    double th=atan2(q.y,q.x),fai=acos((ac*ac+ab*ab-bc*bc)/(2.0*ac*ab));
    double a0=th-fai;
    if (dblcmp(a0+pi)<0)a0+=2*pi;
    double a1=th+fai;
    if (dblcmp(a1-pi)>0)a1-=2*pi;
    if (dblcmp(a0-a1)>0)
    {
     v.push_back(make_pair(a0,1));
     v.push_back(make_pair(pi,-1));
     v.push_back(make_pair(-pi,1));
     v.push_back(make_pair(a1,-1));
    }
    else
    {
     v.push_back(make_pair(a0,1));
     v.push_back(make_pair(a1,-1));
    }
   }
   sort(v.begin(),v.end());
   int cur=0;
   for (j=0;j<v.size();j++)
   {
    if (cur&&dblcmp(v[j].first-pre[cur]))
    {
     ans[cur]+=areaarc(v[j].first-pre[cur],c[i].r);
     ans[cur]+=0.5*point(c[i].p.x+c[i].r*cos(pre[cur]),c[i].p.y+c[i].r*sin(pre[cur])).det(point(c[i].p.x+c[i].r*cos(v[j].first),c[i].p.y+c[i].r*sin(v[j].first)));
    }
    cur+=v[j].second;
    pre[cur]=v[j].first;
   }
  }
  for (i=1;i<=n;i++)
  {
   ans[i]-=ans[i+1];
  }
 }
};
```

4.3 halfplane

```
struct halfplane:public line
{
 double angle;
 halfplane(){}
 // a->b()
 halfplane(point _a,point _b)
 {
  a=_a;
  b=_b;
 }
 halfplane(line v)
 {
  a=v.a;
  b=v.b;
 }
 void calcangle()
 {
  angle=atan2(b.y-a.y,b.x-a.x);
 }
 bool operator<(const halfplane &b)const
 {
  return angle<b.angle;
 }
};
struct halfplanes
{
 int n;
 halfplane hp[maxp];
 point p[maxp];
 int que[maxp];
 int st,ed;
 void push(halfplane tmp)
 {
  hp[n++]=tmp;
 }
 void unique()
 {
  int m=1,i;
  for (i=1;i<n;i++)
  {
   if (dblcmp(hp[i].angle-hp[i-1].angle))hp[m++]=hp[i];
   else if (dblcmp(hp[m-1].b.sub(hp[m-1].a).det(hp[i].a.sub(hp[m-1].a))>0))hp[m-1]=hp[i];
  }
  n=m;
 }
 bool halfplaneinsert()
 {
  int i;
  for (i=0;i<n;i++)hp[i].calcangle();
  sort(hp,hp+n);
  unique();
```

```
   que[st=0]=0;
   que[ed=1]=1;
   p[1]=hp[0].crosspoint(hp[1]);
   for (i=2;i<n;i++)
   {
     while (st<ed&&dblcmp((hp[i].b.sub(hp[i].a).det(p[ed].sub(hp[i].a))))<0)ed--;
     while (st<ed&&dblcmp((hp[i].b.sub(hp[i].a).det(p[st+1].sub(hp[i].a))))<0)st++;
     que[++ed]=i;
     if (hp[i].parallel(hp[que[ed-1]]))return false;
     p[ed]=hp[i].crosspoint(hp[que[ed-1]]);
   }
   while (st<ed&&dblcmp(hp[que[st]].b.sub(hp[que[st]].a).det(p[ed].sub(hp[que[st]].a)))<0)ed--;
   while (st<ed&&dblcmp(hp[que[ed]].b.sub(hp[que[ed]].a).det(p[st+1].sub(hp[que[ed]].a)))<0)st++;
   if (st+1>=ed)return false;
   return true;
  }
  void getconvex(polygon &con)
  {
   p[st]=hp[que[st]].crosspoint(hp[que[ed]]);
   con.n=ed-st+1;
   int j=st,i=0;
   for (;j<=ed;i++,j++)
   {
     con.p[i]=p[j];
   }
  }
};
```

4.4 line

```
struct line
{
   point a,b;
   line(){}
   line(point _a,point _b)
   {
      a=_a;
      b=_b;
   }
   bool operator==(line v)
   {
     return (a==v.a)&&(b==v.b);
   }
   //angle
   line(point p,double angle)
   {
     a=p;
     if (dblcmp(angle-pi/2)==0)
     {
      b=a.add(point(0,1));
     }
     else
     {
      b=a.add(point(1,tan(angle)));
     }
   }
   //ax+by+c=0
   line(double _a,double _b,double _c)
   {
     if (dblcmp(_a)==0)
     {
      a=point(0,-_c/_b);
      b=point(1,-_c/_b);
     }
     else if (dblcmp(_b)==0)
     {
      a=point(-_c/_a,0);
      b=point(-_c/_a,1);
     }
     else
     {
      a=point(0,-_c/_b);
      b=point(1,(-_c-_a)/_b);
     }
   }
   void input()
   {
      a.input();
      b.input();
   }
   void adjust()
   {
     if (b<a)swap(a,b);
   }
   double length()
   {
      return a.distance(b);
   }
   double angle()// 0<=angle<180
  {
   double k=atan2(b.y-a.y,b.x-a.x);
   if (dblcmp(k)<0)k+=pi;
   if (dblcmp(k-pi)==0)k-=pi;
   return k;
  }
   //
   //1
   //2
   //3
   int relation(point p)
   {
      int c=dblcmp(p.sub(a).det(b.sub(a)));
      if (c<0)return 1;
      if (c>0)return 2;
      return 3;
   }
   bool pointonseg(point p)
   {
      return dblcmp(p.sub(a).det(b.sub(a)))==0&&dblcmp(p.sub(a).dot(p.sub(b)))<=0;
   }
   bool parallel(line v)
   {
      return dblcmp(b.sub(a).det(v.b.sub(v.a)))==0;
   }
   //2
   //1
   //0
   int segcrossseg(line v)
   {
      int d1=dblcmp(b.sub(a).det(v.a.sub(a)));
      int d2=dblcmp(b.sub(a).det(v.b.sub(a)));
      int d3=dblcmp(v.b.sub(v.a).det(a.sub(v.a)));
      int d4=dblcmp(v.b.sub(v.a).det(b.sub(v.a)));
      if ((d1^d2)==-2&&(d3^d4)==-2)return 2;
      return (d1==0&&dblcmp(v.a.sub(a).dot(v.a.sub(b)))<=0||
            d2==0&&dblcmp(v.b.sub(a).dot(v.b.sub(b)))<=0||
            d3==0&&dblcmp(a.sub(v.a).dot(a.sub(v.b)))<=0||
            d4==0&&dblcmp(b.sub(v.a).dot(b.sub(v.b)))<=0);
   }
   int linecrossseg(line v)//*this seg v line
   {
      int d1=dblcmp(b.sub(a).det(v.a.sub(a)));
      int d2=dblcmp(b.sub(a).det(v.b.sub(a)));
      if ((d1^d2)==-2)return 2;
      return (d1==0||d2==0);
   }
   //0
  //1
  //2
   int linecrossline(line v)
   {
      if ((*this).parallel(v))
      {
         return v.relation(a)==3;
      }
      return 2;
   }
   point crosspoint(line v)
   {
      double a1=v.b.sub(v.a).det(a.sub(v.a));
      double a2=v.b.sub(v.a).det(b.sub(v.a));
      return point((a.x*a2-b.x*a1)/(a2-a1),(a.y*a2-b.y*a1)/(a2-a1));
   }
   double dispointtoline(point p)
   {
      return fabs(p.sub(a).det(b.sub(a)))/length();
   }
   double dispointtoseg(point p)
   {
      if (dblcmp(p.sub(b).dot(a.sub(b)))<0||dblcmp(p.sub(a).dot(b.sub(a)))<0)
      {
         return min(p.distance(a),p.distance(b));
      }
      return dispointtoline(p);
   }
   point lineprog(point p)
   {
      return a.add(b.sub(a).mul(b.sub(a).dot(p.sub(a))/b.sub(a).len2()));
   }
   point symmetrypoint(point p)
   {
    point q=lineprog(p);
    return point(2*q.x-p.x,2*q.y-p.y);
   }
};
```

4.5 line3d

```
struct line3
{
  point3 a,b;
  line3(){}
  line3(point3 _a,point3 _b)
   {
      a=_a;
      b=_b;
   }
   bool operator==(line3 v)
   {
     return (a==v.a)&&(b==v.b);
   }
   void input()
   {
     a.input();
     b.input();
   }
   double length()
   {
     return a.distance(b);
   }
   bool pointonseg(point3 p)
   {
     return dblcmp(p.sub(a).det(p.sub(b)).len())==0&&dblcmp(a.sub(p).dot(b.sub(p)))<=0;
   }
   double dispointtoline(point3 p)
   {
     return b.sub(a).det(p.sub(a)).len()/a.distance(b);
   }
   double dispointtoseg(point3 p)
   {
      if (dblcmp(p.sub(b).dot(a.sub(b)))<0||dblcmp(p.sub(a).dot(b.sub(a)))<0)
      {
         return min(p.distance(a),p.distance(b));
      }
      return dispointtoline(p);
   }
   point3 lineprog(point3 p)
   {
     return a.add(b.sub(a).trunc(b.sub(a).dot(p.sub(a))/b.distance
            (a)));
   }
   point3 rotate(point3 p,double ang)//parg
 {
   if (dblcmp((p.sub(a).det(p.sub(b)).len()))==0)return p;
   point3 f1=b.sub(a).det(p.sub(a));
   point3 f2=b.sub(a).det(f1);
   double len=fabs(a.sub(p).det(b.sub(p)).len()/a.distance(b));
   f1=f1.trunc(len);f2=f2.trunc(len);
   point3 h=p.add(f2);
   point3 pp=h.add(f1);
   return h.add((p.sub(h)).mul(cos(ang*1.0))).add((pp.sub(h)).mul(
          sin(ang*1.0)));
 }
};
```

4.6 plane

```
struct plane
{
   point3 a,b,c,o;
   plane(){}
   plane(point3 _a,point3 _b,point3 _c)
   {
      a=_a;
      b=_b;
      c=_c;
      o=pvec();
   }
   plane(double _a,double _b,double _c,double _d)
   {
     //ax+by+cz+d=0
     o=point3(_a,_b,_c);
   if (dblcmp(_a)!=0)
   {
     a=point3((-_d-_c-_b)/_a,1,1);
   }
   else if (dblcmp(_b)!=0)
   {
     a=point3(1,(-_d-_c-_a)/_b,1);
   }
   else if (dblcmp(_c)!=0)
   {
     a=point3(1,1,(-_d-_a-_b)/_c);
   }
   }
   void input()
   {
      a.input();
      b.input();
      c.input();
      o=pvec();
   }
   point3 pvec()
   {
      return b.sub(a).det(c.sub(a));
   }
 bool pointonplane(point3 p)//
   {
     return dblcmp(p.sub(a).dot(o))==0;
   }
   //0
 //1
 //2
   int pointontriangle(point3 p)//abc
   {
     if (!pointonplane(p))return 0;
     double s=a.sub(b).det(c.sub(b)).len();
     double s1=p.sub(a).det(p.sub(b)).len();
     double s2=p.sub(a).det(p.sub(c)).len();
     double s3=p.sub(b).det(p.sub(c)).len();
     if (dblcmp(s-s1-s2-s3))return 0;
     if (dblcmp(s1)&&dblcmp(s2)&&dblcmp(s3))return 2;
     return 1;
   }
   //
   //0
   //1
   //2
   bool relationplane(plane f)
   {
      if (dblcmp(o.det(f.o).len()))return 0;
      if (pointonplane(f.a))return 2;
      return 1;
   }
   double angleplane(plane f)//
   {
     return acos(o.dot(f.o)/(o.len()*f.o.len()));
   }
   double dispoint(point3 p)//
 {
   return fabs(p.sub(a).dot(o)/o.len());
 }
   point3 pttoplane(point3 p)//
 {
   line3 u=line3(p,p.add(o));
   crossline(u,p);
   return p;
 }
   int crossline(line3 u,point3 &p)//
   {
     double x=o.dot(u.b.sub(a));
     double y=o.dot(u.a.sub(a));
     double d=x-y;
     if (dblcmp(fabs(d))==0)return 0;
     p=u.a.mul(x).sub(u.b.mul(y)).div(d);
     return 1;
   }
   int crossplane(plane f,line3 &u)//
   {
     point3 oo=o.det(f.o);
     point3 v=o.det(oo);
     double d=fabs(f.o.dot(v));
     if (dblcmp(d)==0)return 0;
     point3 q=a.add(v.mul(f.o.dot(f.a.sub(a))/d));
     u=line3(q,q.add(oo));
     return 1;
   }
};
```

4.7 point

```
using namespace std;

#define mp make_pair
#define pb push_back

const double eps=1e-8;
const double pi=acos(-1.0);
const double inf=1e20;
const int maxp=8;

int dblcmp(double d)
{
   if (fabs(d)<eps)return 0;
   return d>eps?1:-1;
}

inline double sqr(double x)
{
   return x*x;
}

struct point
{
   double x,y;
   point(){}
   point(double _x,double _y):
   x(_x),y(_y){};
   void input()
   {
      scanf("%lf%lf",&x,&y);
   }
   void output()
   {
      printf("%.2f %.2f\n",x,y);
   }
   bool operator==(point a)const
   {
      return dblcmp(a.x-x)==0&&dblcmp(a.y-y)==0;
   }
   bool operator<(point a)const
   {
      return dblcmp(a.x-x)==0?dblcmp(y-a.y)<0:x<a.x;
   }
   double len()
   {
      return hypot(x,y);
   }
   double len2()
   {
      return x*x+y*y;
   }
   double distance(point p)
   {
      return hypot(x-p.x,y-p.y);
   }
   point add(point p)
   {
      return point(x+p.x,y+p.y);
   }
   point sub(point p)
   {
      return point(x-p.x,y-p.y);
   }
   point mul(double b)
   {
      return point(x*b,y*b);
   }
   point div(double b)
   {
      return point(x/b,y/b);
   }
   double dot(point p)
   {
      return x*p.x+y*p.y;
   }
   double det(point p)
   {
      return x*p.y-y*p.x;
   }
   double rad(point a,point b)
   {
    point p=*this;
    return fabs(atan2(fabs(a.sub(p).det(b.sub(p))),a.sub(p).dot(b
           .sub(p))));
 }
   point trunc(double r)
 {
   double l=len();
   if (!dblcmp(l))return *this;
   r/=l;
   return point(x*r,y*r);
 }
   point rotleft()
   {
      return point(-y,x);
   }
   point rotright()
   {
      return point(y,-x);
   }
   point rotate(point p,double angle)//pangle
   {
      point v=this->sub(p);
      double c=cos(angle),s=sin(angle);
      return point(p.x+v.x*c-v.y*s,p.y+v.x*s+v.y*c);
   }
};
```

4.8 point3d

```
struct point3
{
 double x,y,z;
 point3(){}
 point3(double _x,double _y,double _z):
 x(_x),y(_y),z(_z){};
 void input()
 {
   scanf("%lf%lf%lf",&x,&y,&z);
 }
 void output()
 {
   printf("%.2lf %.2lf %.2lf\n",x,y,z);
 }
 bool operator==(point3 a)
   {
      return dblcmp(a.x-x)==0&&dblcmp(a.y-y)==0&&dblcmp(a.z-z)==0;
   }
   bool operator<(point3 a)const
   {
      return dblcmp(a.x-x)==0?dblcmp(y-a.y)==0?dblcmp(z-a.z)<0:y<a
             .y:x<a.x;
   }
 double len()
   {
      return sqrt(len2());
   }
   double len2()
   {
      return x*x+y*y+z*z;
   }
   double distance(point3 p)
   {
      return sqrt((p.x-x)*(p.x-x)+(p.y-y)*(p.y-y)+(p.z-z)*(p.z-z))
             ;
   }
   point3 add(point3 p)
   {
      return point3(x+p.x,y+p.y,z+p.z);
   }
   point3 sub(point3 p)
   {
      return point3(x-p.x,y-p.y,z-p.z);
   }
 point3 mul(double d)
 {
   return point3(x*d,y*d,z*d);
 }
 point3 div(double d)
 {
   return point3(x/d,y/d,z/d);
 }
 double dot(point3 p)
   {
      return x*p.x+y*p.y+z*p.z;
   }
   point3 det(point3 p)
   {
      return point3(y*p.z-p.y*z,p.x*z-x*p.z,x*p.y-p.x*y);
   }
   double rad(point3 a,point3 b)
   {
    point3 p=(*this);
    return acos(a.sub(p).dot(b.sub(p))/(a.distance(p)*b.distance(
           p)));
   }
   point3 trunc(double r)
   {
      r/=len();
      return point3(x*r,y*r,z*r);
   }
   point3 rotate(point3 o,double r) // building?
   {
   }
};
```

4.9 polygon

```
struct polygon
{
   int n;
   point p[maxp];
   line l[maxp];
   void input()
   {
    n=4;
    p[0].input();
    p[2].input();
    double dis=p[0].distance(p[2]);
    p[1]=p[2].rotate(p[0],pi/4);
    p[1]=p[0].add((p[1].sub(p[0])).trunc(dis/sqrt(2.0)));
    p[3]=p[2].rotate(p[0],2*pi-pi/4);
    p[3]=p[0].add((p[3].sub(p[0])).trunc(dis/sqrt(2.0)));
   }
   void add(point q)
   {
    p[n++]=q;
   }
   void getline()
   {
      for (int i=0;i<n;i++)
      {
         l[i]=line(p[i],p[(i+1)%n]);
      }
   }
 struct cmp
{
   point p;
   cmp(const point &p0){p=p0;}
   bool operator()(const point &aa,const point &bb)
   {
      point a=aa,b=bb;
      int d=dblcmp(a.sub(p).det(b.sub(p)));
      if (d==0)
      {
         return dblcmp(a.distance(p)-b.distance(p))<0;
      }
      return d>0;
   }
};
 void norm()
 {
    point mi=p[0];
    for (int i=1;i<n;i++)mi=min(mi,p[i]);
    sort(p,p+n,cmp(mi));
 }
 void getconvex(polygon &convex)
 {
    int i,j,k;
    sort(p,p+n);
    convex.n=n;
    for (i=0;i<min(n,2);i++)
    {
       convex.p[i]=p[i];
    }
    if (n<=2)return;
    int &top=convex.n;
    top=1;
    for (i=2;i<n;i++)
    {
       while (top&&convex.p[top].sub(p[i]).det(convex.p[top-1].
              sub(p[i]))<=0)
          top--;
       convex.p[++top]=p[i];
    }
    int temp=top;
    convex.p[++top]=p[n-2];
    for (i=n-3;i>=0;i--)
    {
       while (top!=temp&&convex.p[top].sub(p[i]).det(convex.p[
              top-1].sub(p[i]))<=0)
          top--;
       convex.p[++top]=p[i];
    }
 }
 bool isconvex()
 {
   bool s[3];
   memset(s,0,sizeof(s));
   int i,j,k;
   for (i=0;i<n;i++)
   {
    j=(i+1)%n;
    k=(j+1)%n;
    s[dblcmp(p[j].sub(p[i]).det(p[k].sub(p[i])))+1]=1;
    if (s[0]&&s[2])return 0;
   }
   return 1;
 }
 //3
//2
//1
//0
 int relationpoint(point q)
 {
   int i,j;
   for (i=0;i<n;i++)
   {
    if (p[i]==q)return 3;
   }
   getline();
   for (i=0;i<n;i++)
   {
    if (l[i].pointonseg(q))return 2;
   }
   int cnt=0;
   for (i=0;i<n;i++)
 {
   j=(i+1)%n;
   int k=dblcmp(q.sub(p[j]).det(p[i].sub(p[j])));
   int u=dblcmp(p[i].y-q.y);
   int v=dblcmp(p[j].y-q.y);
   if (k>0&&u<0&&v>=0)cnt++;
   if (k<0&&v<0&&u>=0)cnt--;
 }
 return cnt!=0;
 }
 //1
 //2
//0
 int relationline(line u)
 {
   int i,j,k=0;
   getline();
   for (i=0;i<n;i++)
   {
    if (l[i].segcrossseg(u)==2)return 1;
    if (l[i].segcrossseg(u)==1)k=1;
   }
   if (!k)return 0;
   vector<point>vp;
   for (i=0;i<n;i++)
   {
    if (l[i].segcrossseg(u))
     {
      if (l[i].parallel(u))
      {
       vp.pb(u.a);
       vp.pb(u.b);
       vp.pb(l[i].a);
       vp.pb(l[i].b);
       continue;
      }
    vp.pb(l[i].crosspoint(u));
   }
  }
  sort(vp.begin(),vp.end());
  int sz=vp.size();
  for (i=0;i<sz-1;i++)
  {
   point mid=vp[i].add(vp[i+1]).div(2);
   if (relationpoint(mid)==1)return 1;
  }
  return 2;
 }
 //u
 //
 void convexcut(line u,polygon &po)
 {
   int i,j,k;
   int &top=po.n;
   top=0;
   for (i=0;i<n;i++)
   {
     int d1=dblcmp(p[i].sub(u.a).det(u.b.sub(u.a)));
     int d2=dblcmp(p[(i+1)%n].sub(u.a).det(u.b.sub(u.a)));
     if (d1>=0)po.p[top++]=p[i];
     if (d1*d2<0)po.p[top++]=u.crosspoint(line(p[i],p[(i+1)%n
          ]));
   }
 }
 double getcircumference()
 {
   double sum=0;
   int i;
   for (i=0;i<n;i++)
   {
     sum+=p[i].distance(p[(i+1)%n]);
   }
   return sum;
 }
 double getarea()
 {
   double sum=0;
   int i;
   for (i=0;i<n;i++)
   {
     sum+=p[i].det(p[(i+1)%n]);
   }
   return fabs(sum)/2;
 }
 bool getdir()//1 0
 {
   double sum=0;
   int i;
   for (i=0;i<n;i++)
   {
     sum+=p[i].det(p[(i+1)%n]);
   }
   if (dblcmp(sum)>0)return 1;
   return 0;
 }
 point getbarycentre() // centroid
 {
   point ret(0,0);
   double area=0;
   int i;
   for (i=1;i<n-1;i++)
   {
     double tmp=p[i].sub(p[0]).det(p[i+1].sub(p[0]));
     if (dblcmp(tmp)==0)continue;
     area+=tmp;
     ret.x+=(p[0].x+p[i].x+p[i+1].x)/3*tmp;
     ret.y+=(p[0].y+p[i].y+p[i+1].y)/3*tmp;
   }
   if (dblcmp(area))ret=ret.div(area);
   return ret;
 }
 double areaintersection(polygon po) // refer: HPI
 {
 }
 double areaunion(polygon po)
 {
  return getarea()+po.getarea()-areaintersection(po);
 }
 double areacircle(circle c)
 {
 int i,j,k,l,m;
 double ans=0;
 for (i=0;i<n;i++)
 {
  int j=(i+1)%n;
  if (dblcmp(p[j].sub(c.p).det(p[i].sub(c.p)))>=0)
  {
   ans+=c.areatriangle(p[i],p[j]);
  }
  else
  {
   ans-=c.areatriangle(p[i],p[j]);
  }
 }
 return fabs(ans);
 }
 //
//0
//1
//2
 int relationcircle(circle c)
 {
  getline();
  int i,x=2;
  if (relationpoint(c.p)!=1)return 0;
  for (i=0;i<n;i++)
  {
   if (c.relationseg(l[i])==2)return 0;
   if (c.relationseg(l[i])==1)x=1;
  }
  return x;
 }
 void find(int st,point tri[],circle &c)
 {
  if (!st)
  {
    c=circle(point(0,0),-2);
   }
   if (st==1)
   {
    c=circle(tri[0],0);
   }
   if (st==2)
   {
    c=circle(tri[0].add(tri[1]).div(2),tri[0].distance(tri[1])
         /2.0);
   }
   if (st==3)
   {
    c=circle(tri[0],tri[1],tri[2]);
   }
  }
  void solve(int cur,int st,point tri[],circle &c)
  {
   find(st,tri,c);
   if (st==3)return;
   int i;
   for (i=0;i<cur;i++)
   {
    if (dblcmp(p[i].distance(c.p)-c.r)>0)
    {
     tri[st]=p[i];
     solve(i,st+1,tri,c);
    }
   }
  }
  circle mincircle()//
 {
  random_shuffle(p,p+n);
  point tri[4];
  circle c;
  solve(n,0,tri,c);
  return c;
 }
 int circlecover(double r)//
 {
  int ans=0,i,j;
  vector<pair<double,int> >v;
  for (i=0;i<n;i++)
  {
   v.clear();
   for (j=0;j<n;j++)if (i!=j)
   {
    point q=p[i].sub(p[j]);
    double d=q.len();
    if (dblcmp(d-2*r)<=0)
    {
     double arg=atan2(q.y,q.x);
     if (dblcmp(arg)<0)arg+=2*pi;
     double t=acos(d/(2*r));
     v.push_back(make_pair(arg-t+2*pi,-1));
     v.push_back(make_pair(arg+t+2*pi,1));
    }
   }
   sort(v.begin(),v.end());
   int cur=0;
   for (j=0;j<v.size();j++)
   {
    if (v[j].second==-1)++cur;
    else --cur;
    ans=max(ans,cur);
   }
  }
  return ans+1;
 }
 int pointinpolygon(point q)//
 {
  if (getdir())reverse(p,p+n);
  if (dblcmp(q.sub(p[0]).det(p[n-1].sub(p[0])))==0)
  {
   if (line(p[n-1],p[0]).pointonseg(q))return n-1;
   return -1;
  }
  int low=1,high=n-2,mid;
  while (low<=high)
  {
   mid=(low+high)>>1;
   if (dblcmp(q.sub(p[0]).det(p[mid].sub(p[0])))>=0&&dblcmp(q.
        sub(p[0]).det(p[mid+1].sub(p[0])))<0)
   {
    polygon c;
    c.p[0]=p[mid];
    c.p[1]=p[mid+1];
    c.p[2]=p[0];
    c.n=3;
    if (c.relationpoint(q))return mid;
    return -1;
   }
   if (dblcmp(q.sub(p[0]).det(p[mid].sub(p[0])))>0)
   {
    low=mid+1;
   }
   else
   {
    high=mid-1;
   }
  }
  return -1;
 }
};
```

4.10 polygons

```
struct polygons
{
 vector<polygon>p;
 polygons()
 {
  p.clear();
 }
 void clear()
 {
  p.clear();
 }
 void push(polygon q)
 {
  if (dblcmp(q.getarea()))p.pb(q);
 }
 vector<pair<double,int> >e;
 void ins(point s,point t,point X,int i)
 {
  double r=fabs(t.x-s.x)>eps?(X.x-s.x)/(t.x-s.x):(X.y-s.y)/(t.y-s.y);
  r=min(r,1.0);r=max(r,0.0);
  e.pb(mp(r,i));
 }
 double polyareaunion()
 {
  double ans=0.0;
  int c0,c1,c2,i,j,k,w;
  for (i=0;i<p.size();i++)
  {
   if (p[i].getdir()==0)reverse(p[i].p,p[i].p+p[i].n);
  }
  for (i=0;i<p.size();i++)
  {
   for (k=0;k<p[i].n;k++)
   {
    point &s=p[i].p[k],&t=p[i].p[(k+1)%p[i].n];
    if (!dblcmp(s.det(t)))continue;
    e.clear();
    e.pb(mp(0.0,1));
    e.pb(mp(1.0,-1));
    for (j=0;j<p.size();j++)if (i!=j)
    {
     for (w=0;w<p[j].n;w++)
     {
      point a=p[j].p[w],b=p[j].p[(w+1)%p[j].n],c=p[j].p[(w-1+p[j].n)%p[j].n];
      c0=dblcmp(t.sub(s).det(c.sub(s)));
      c1=dblcmp(t.sub(s).det(a.sub(s)));
      c2=dblcmp(t.sub(s).det(b.sub(s)));
      if (c1*c2<0)ins(s,t,line(s,t).crosspoint(line(a,b)),-c2);
      else if (!c1&&c0*c2<0)ins(s,t,a,-c2);
      else if (!c1&&!c2)
      {
       int c3=dblcmp(t.sub(s).det(p[j].p[(w+2)%p[j].n].sub(s)));
       int dp=dblcmp(t.sub(s).dot(b.sub(a)));
       if (dp&&c0)ins(s,t,a,dp>0?c0*((j>i)^(c0<0)):-(c0<0));
       if (dp&&c3)ins(s,t,b,dp>0?-c3*((j>i)^(c3<0)):c3<0);
      }
     }
    }
    sort(e.begin(),e.end());
    int ct=0;
    double tot=0.0,last;
    for (j=0;j<e.size();j++)
    {
     if (ct==p.size())tot+=e[j].first-last;
     ct+=e[j].second;
     last=e[j].first;
    }
    ans+=s.det(t)*tot;
   }
  }
  return fabs(ans)*0.5;
 }
};
```

5 graph

5.1 2-sat

```
#define maxn 2008
struct Twosat
{
   int n;
   std::vector<int>G[maxn*2];
   bool mark[maxn*2];
   int s[maxn*2],c;

   bool dfs(int x)
   {
      if(mark[x^1])return false;
      if(mark[x])return true;
      mark[x]=true;
      s[c++]=x;
      for(int i=0;i<G[x].size();++i)
         if(!dfs(G[x][i]))return false;
      return true;
   }

   void init(int n)
   {
      this->n=n;
      for(int i=0;i<n*2;++i)
         G[i].clear();
      memset(mark,0,sizeof(mark));
   }
   void add_clause(int x,int xval,int y,int yval)//
   {
      x=x*2+xval;
      y=y*2+yval;
      G[x^1].push_back(y);
      G[y^1].push_back(x);
   }

   bool solve()
   {
      for(int i=0;i<n*2;i+=2)
         if(!mark[i]&&!mark[i+1])
         {
            c=0;
            if(!dfs(i))
            {
               while(c>0)
                  mark[s[--c]]=false;
               if(!dfs(i+1))
                  return false;
            }
         }
      return true;
   }
};
```

5.2 Articulation

```
void dfs(int now,int fa) // now1
{
   int p(0);
   dfn[now]=low[now]=cnt++;
   for(std::list<int>::const_iterator it(edge[now].begin());it!=edge[now].end();++it)
      if(dfn[*it]==-1)
      {
         dfs(*it,now);
         ++p;
         low[now]=std::min(low[now],low[*it]);
         if((now==1 && p>1) || (now!=1 && low[*it]>=dfn[now])) //
            ans.insert(now);
      }
      else
         if(*it!=fa)
            low[now]=std::min(low[now],dfn[*it]);
}
```

5.3 Augmenting Path Algorithm for Maximum Cardinality Bipartite Matching

```
#include<cstdio>
#include<cstring>

#define MAXX 111

bool Map[MAXX][MAXX],visit[MAXX];
int link[MAXX],n,m;
bool dfs(int t)
{
   for (int i=0; i<m; i++)
      if (!visit[i] && Map[t][i]){
         visit[i] = true;
         if (link[i]==-1 || dfs(link[i])){
            link[i] = t;
            return true;
         }
      }
   return false;
}
int main()
{
   int k,a,b,c;
   while (scanf("%d",&n),n){
      memset(Map,false,sizeof(Map));
      scanf("%d%d",&m,&k);
      while (k--){
         scanf("%d%d%d",&a,&b,&c);
         if (b && c)
            Map[b][c] = true;
      }
      memset(link,-1,sizeof(link));
      int ans = 0;
      for (int i=0; i<n; i++){
         memset(visit,false,sizeof(visit));
         if (dfs(i))
            ans++;
      }
      printf("%d\n",ans);
   }
}
```

5.4 Biconnected Component - Edge

```
// hdu 4612
#include<cstdio>
#include<algorithm>
#include<set>
#include<cstring>
#include<stack>
#include<queue>

#define MAXX 200111
#define MAXE (1000111*2)
#pragma comment(linker, "/STACK:16777216")

int edge[MAXX],to[MAXE],nxt[MAXE],cnt;
#define v to[i]
inline void add(int a,int b)
{
   nxt[++cnt]=edge[a];
   edge[a]=cnt;
   to[cnt]=b;
}

int dfn[MAXX],low[MAXX],col[MAXX],belong[MAXX];
int idx,bcnt;
std::stack<int>st;

void tarjan(int now,int last)
{
   col[now]=1;
   st.push(now);
   dfn[now]=low[now]=++idx;
   bool flag(false);
   for(int i(edge[now]);i;i=nxt[i])
   {
      if(v==last && !flag)
      {
         flag=true;
         continue;
      }
      if(!col[v])
      {
         tarjan(v,now);
         low[now]=std::min(low[now],low[v]);
         /*
         if(low[v]>dfn[now])
         then this is a bridge
         */
      }
      else
         if(col[v]==1)
            low[now]=std::min(low[now],dfn[v]);
   }
   col[now]=2;
   if(dfn[now]==low[now])
   {
      ++bcnt;
      static int x;
      do
      {
         x=st.top();
         st.pop();
         belong[x]=bcnt;
      }while(x!=now);
   }
}

std::set<int>set[MAXX];

int dist[MAXX];
std::queue<int>q;
int n,m,i,j,k;

inline int go(int s)
{
   static std::set<int>::const_iterator it;
   memset(dist,0x3f,sizeof dist);
   dist[s]=0;
   q.push(s);
   while(!q.empty())
   {
      s=q.front();
      q.pop();
      for(it=set[s].begin();it!=set[s].end();++it)
         if(dist[*it]>dist[s]+1)
         {
            dist[*it]=dist[s]+1;
            q.push(*it);
         }
   }
   return std::max_element(dist+1,dist+1+bcnt)-dist;
}

int main()
{
   while(scanf("%d %d",&n,&m),(n||m))
   {
      cnt=0;
      memset(edge,0,sizeof edge);
      while(m--)
      {
         scanf("%d %d",&i,&j);
         add(i,j);
         add(j,i);
      }

      memset(dfn,0,sizeof dfn);
      memset(belong,0,sizeof belong);
      memset(low,0,sizeof low);
      memset(col,0,sizeof col);
      bcnt=idx=0;
      while(!st.empty())
         st.pop();

      tarjan(1,-1);
      for(i=1;i<=bcnt;++i)
         set[i].clear();
      for(i=1;i<=n;++i)
         for(j=edge[i];j;j=nxt[j])
            set[belong[i]].insert(belong[to[j]]);
      for(i=1;i<=bcnt;++i)
         set[i].erase(i);
      /*
      printf("%d\n",dist[go(go(1))]);
      for(i=1;i<=bcnt;++i)
         printf("%d\n",dist[i]);
      puts("");
      */
      printf("%d\n",bcnt-1-dist[go(go(1))]);
   }
   return 0;
}
```

5.5 Biconnected Component

```
#include<cstdio>
#include<cstring>
#include<stack>
#include<queue>
#include<algorithm>

const int MAXN=100000*2;
const int MAXM=200000;

//0-based

struct edges
{
   int to,next;
   bool cut,visit;
} edge[MAXM<<1];

int head[MAXN],low[MAXN],dpt[MAXN],L;
bool visit[MAXN],cut[MAXN];
int idx;
std::stack<int> st;
int bcc[MAXM];

void init(int n)
{
   L=0;
   memset(head,-1,4*n);
   memset(visit,0,n);
}

void add_edge(int u,int v)
{
   edge[L].cut=edge[L].visit=false;
   edge[L].to=v;
   edge[L].next=head[u];
   head[u]=L++;
}

void dfs(int u,int fu,int deg)
{
   cut[u]=false;
   visit[u]=true;
   low[u]=dpt[u]=deg;
   int tot=0;
   for (int i=head[u]; i!=-1; i=edge[i].next)
   {
      int v=edge[i].to;
      if (edge[i].visit)
         continue;
      st.push(i/2);
      edge[i].visit=edge[i^1].visit=true;
      if (visit[v])
      {
         low[u]=dpt[v]>low[u]?low[u]:dpt[v];
         continue;
      }
      dfs(v,u,deg+1);
      edge[i].cut=edge[i^1].cut=(low[v]>dpt[u] || edge[i].cut);
      if (u!=fu) cut[u]=low[v]>=dpt[u]?1:cut[u];
      if (low[v]>=dpt[u] || u==fu)
      {
         while (st.top()!=i/2)
         {
            int x=st.top()*2,y=st.top()*2+1;
            bcc[st.top()]=idx;
            st.pop();
         }
         bcc[i/2]=idx++;
         st.pop();
      }
      low[u]=low[v]>low[u]?low[u]:low[v];
      tot++;
   }
   if (u==fu && tot>1)
      cut[u]=true;
}

int main()
{
   int n,m;
   while (scanf("%d%d",&n,&m)!=EOF)
   {
      init(n);
      for (int i=0; i<m; i++)
      {
         int u,v;
         scanf("%d%d",&u,&v);
         add_edge(u,v);
         add_edge(v,u);
      }
      idx=0;
      for (int i=0; i<n; i++)
         if (!visit[i])
            dfs(i,i,0);
   }
   return 0;
}
```

5.6 Blossom algorithm

```
#include<cstdio>
#include<vector>
#include<cstring>
#include<algorithm>

#define MAXX 233

bool map[MAXX][MAXX];
std::vector<int>p[MAXX];
int m[MAXX];
int vis[MAXX];
int q[MAXX],*qf,*qb;

int n;

inline void label(int x,int y,int b)
{
   static int i,z;
   for(i=b+1;i<p[x].size();++i)
      if(vis[z=p[x][i]]==1)
      {
         p[z]=p[y];
         p[z].insert(p[z].end(),p[x].rbegin(),p[x].rend()-i);
         vis[z]=0;
         *qb++=z;
      }
}

inline bool bfs(int now)
{
   static int i,x,y,z,b;
   for(i=0;i<n;++i)
      p[i].resize(0);
   p[now].push_back(now);
   memset(vis,-1,sizeof vis);
   vis[now]=0;
   qf=qb=q;
   *qb++=now;

   while(qf<qb)
      for(x=*qf++,y=0;y<n;++y)
         if(map[x][y] && m[y]!=y && vis[y]!=1)
         {
            if(vis[y]==-1)
               if(m[y]==-1)
               {
                  for(i=0;i+1<p[x].size();i+=2)
                  {
                     m[p[x][i]]=p[x][i+1];
                     m[p[x][i+1]]=p[x][i];
                  }
                  m[x]=y;
                  m[y]=x;
                  return true;
               }
               else
               {
                  p[z=m[y]]=p[x];
                  p[z].push_back(y);
                  p[z].push_back(z);
                  vis[y]=1;
                  vis[z]=0;
                  *qb++=z;
               }
            else
            {
               for(b=0;b<p[x].size() && b<p[y].size() && p[x][b]==
                     p[y][b];++b);
               --b;
               label(x,y,b);
               label(y,x,b);
            }
         }
   return false;
}

int i,j,k;
int ans;

int main()
{
   scanf("%d",&n);
   for(i=0;i<n;++i)
      p[i].reserve(n);
   while(scanf("%d %d",&i,&j)!=EOF)
   {
      --i;
      --j;
      map[i][j]=map[j][i]=true;
   }
   memset(m,-1,sizeof m);
   for(i=0;i<n;++i)
      if(m[i]==-1)
      {
         if(bfs(i))
            ++ans;
         else
            m[i]=i;
      }
   printf("%d\n",ans<<1);
   for(i=0;i<n;++i)
      if(i<m[i])
         printf("%d %d\n",i+1,m[i]+1);
   return 0;
}
```

5.7 Bridge

```
void dfs(const short &now,const short &fa)
{
   dfn[now]=low[now]=cnt++;
   for(int i(0);i<edge[now].size();++i)
      if(dfn[edge[now][i]]==-1)
      {
         dfs(edge[now][i],now);
         low[now]=std::min(low[now],low[edge[now][i]]);
         if(low[edge[now][i]]>dfn[now]) //,
         {
            if(edge[now][i]<now)
            {
               j=edge[now][i];
               k=now;
            }
            else
            {
               j=now;
               k=edge[now][i];
            }
            ans.push_back(node(j,k));
         }
      }
      else
         if(edge[now][i]!=fa)
            low[now]=std::min(low[now],low[edge[now][i]]);
}
```

5.8 chu-liu algorithm

```
#include<cstdio>
#include<cstring>
#include<algorithm>

const int inf = 0x5fffffff;

int n,m,u,v,cost,dis[1001][1001],L;
int pre[1001],id[1001],visit[1001],in[1001];

void init(int n)
{
   L = 0;
   for (int i = 0; i < n; i++)
      for (int j = 0; j < n; j++)
         dis[i][j] = inf;
}

struct Edge
{
   int u,v,cost;
};

Edge e[1001*1001];


int zhuliu(int root,int n,int m,Edge e[])
{
   int res = 0,u,v;
   while (true)
   {
      for (int i = 0; i < n; i++)
         in[i] = inf;
      for (int i = 0; i < m; i++)
         if (e[i].u != e[i].v && e[i].cost < in[e[i].v])
         {
            pre[e[i].v] = e[i].u;
            in[e[i].v] = e[i].cost;
         }
      for (int i = 0; i < n; i++)
         if (i != root)
            if (in[i] == inf)
               return -1;
      int tn = 0;
      memset(id,-1,sizeof(id));
      memset(visit,-1,sizeof(visit));
      in[root] = 0;
      for (int i = 0; i < n; i++)
      {
         res += in[i];
         v = i;
         while (visit[v] != i && id[v] == -1 && v != root)
         {
            visit[v] = i;
            v = pre[v];
         }
         if(v != root && id[v] == -1)
         {
            for(int u = pre[v] ; u != v ; u = pre[u])
               id[u] = tn;
            id[v] = tn++;
         }
      }
      if(tn == 0) break;
      for (int i = 0; i < n; i++)
         if (id[i] == -1)
            id[i] = tn++;
      for (int i = 0; i < m;)
      {
         int v = e[i].v;
         e[i].u = id[e[i].u];
         e[i].v = id[e[i].v];
         if (e[i].u != e[i].v)
            e[i++].cost -= in[v];
         else
            std::swap(e[i],e[--m]);
      }
      n = tn;
      root = id[root];
   }
   return res;
}

int main()
{
   while (scanf("%d%d",&n,&m) != EOF)
   {
      init(n);
      for (int i = 0; i < m; i++)
      {
         scanf("%d%d%d",&u,&v,&cost);
         if (u == v) continue;
         dis[u][v] = std::min(dis[u][v],cost);
      }
      L = 0;
      for (int i = 0; i < n; i++)
         for (int j = 0; j < n; j++)
            if (dis[i][j] != inf)
            {
               e[L].u = i;
               e[L].v = j;
               e[L++].cost = dis[i][j];
            }
      printf("%d\n",zhuliu(0,n,L,e));
   }
   return 0;
}
```

5.9 Covering problems

```



 ,., .

 SGus, ssssG

()

 GDDuvu,vu,vG=G = (V;E)UVu,vU < u; v >EUGGUGUGGGUVu; vU< u; v > EUGGUGUGGG


+= V
=
=

minimum cover:
vertex cover vertex bipartite graph = maximum cardinality bipartite matching

 X
 Y

 Graph Traversal

 Graph Traversal

vertex cover edge

edge cover vertex
 Maximum Matching Minimum Edge Cover

edge cover edge

path cover vertex
general graph: NP-H
tree: DP
DAG: ,ans=-

path cover edge
minimize the count of euler path ( greedy is ok? )

cycle cover vertex
general: NP-H
weighted: do like path cover vertex, with KM algorithm

cycle cover edge
NP-H
```

5.10 Difference constraints

```
for a - b <= c
   add(b,a,c);



//?()

0
```

5.11 Dinitz's algorithm

```
#include<cstdio>
#include<algorithm>
#include<cstring>

#define MAXX 111
#define MAXM (MAXX*MAXX*4)
#define inf 0x3f3f3f3f

int n;
int w[MAXX],h[MAXX],q[MAXX];
int edge[MAXX],to[MAXM],cap[MAXM],nxt[MAXM],cnt;
int source,sink;

inline void add(int a,int b,int c)
{
   nxt[cnt]=edge[a];
   edge[a]=cnt;
   to[cnt]=b;
   cap[cnt]=c;
   ++cnt;
}

inline bool bfs()
{
   static int *qf,*qb;
   static int i;
   memset(h,-1,sizeof h);
   qf=qb=q;
   h[*qb++=source]=0;
   for(;qf!=qb;++qf)
      for(i=edge[*qf];i!=-1;i=nxt[i])
         if(cap[i] && h[to[i]]==-1)
            h[*qb++=to[i]]=h[*qf]+1;
   return h[sink]!=-1;
}

int dfs(int now,int maxcap)
{
   if(now==sink)
      return maxcap;
   int flow(maxcap),d;
   for(int &i(w[now]);i!=-1;i=nxt[i])
      if(cap[i] && h[to[i]]==h[now]+1)// && (flow=dfs(to[i],std::min(maxcap,cap[i]))))
      {
         d=dfs(to[i],std::min(flow,cap[i]));
         cap[i]-=d;
         cap[i^1]+=d;
         flow-=d;
         if(!flow)
            return maxcap;
      }
   return maxcap-flow;
}

int nc,np,m,i,j,k;
int ans;

int main()
{
   while(scanf("%d %d %d %d",&n,&np,&nc,&m)!=EOF)
   {
      cnt=0;
      memset(edge,-1,sizeof edge);
      while(m--)
      {
         while(getchar()!='(');
         scanf("%d",&i);
         while(getchar()!=',');
         scanf("%d",&j);
         while(getchar()!=')');
         scanf("%d",&k);
         if(i!=j)
         {
            ++i;
            ++j;
            add(i,j,k);
            add(j,i,0);
         }
      }
      source=++n;
      while(np--)
      {
         while(getchar()!='(');
         scanf("%d",&i);
         while(getchar()!=')');
         scanf("%d",&j);
         ++i;
         add(source,i,j);
         add(i,source,0);
      }
      sink=++n;
      while(nc--)
      {
         while(getchar()!='(');
         scanf("%d",&i);
         while(getchar()!=')');
         scanf("%d",&j);
         ++i;
         add(i,sink,j);
         add(sink,i,0);
      }
      ans=0;
      while(bfs())
      {
         memcpy(w,edge,sizeof edge);
         ans+=dfs(source,inf);
         /*
         while((k=dfs(source,inf)))
            ans+=k;
            */
      }
      printf("%d\n",ans);
   }
   return 0;
}
```

5.12 Feasible flow problem

```
#include<cstdio>
#include<cstring>
#include<algorithm>

#define MAXX (255)
#define inf 0x3f3f3f3f

int cap[MAXX][MAXX];
int h[MAXX];
int last[MAXX];
int source,sink;

int mat[MAXX][MAXX][2];
bool bg,flag;

int n;

inline bool bfs()
{
   static int q[MAXX],*qf,*qb,i;
   memset(h,-1,sizeof h);
   qf=qb=q;
   for(h[*qb++=source]=0;qf!=qb;++qf)
      for(i=1;i<=n;++i)
         if(cap[*qf][i] && h[i]==-1)
         {
            h[*qb++=i]=h[*qf]+1;
            if(i==sink)
               return true;
         }
   return false;
}

int dfs(int now,int maxcap)
{
   if(now==sink)
      return maxcap;
   for(int i(last[now]),f;i<=n;++i)
      if(cap[now][i] && h[i]==h[now]+1 && (f=dfs(i,std::min(maxcap,cap[now][i]))))
      {
         cap[now][i]-=f;
         cap[i][now]+=f;
         return f;
      }
   return 0;
}

int T;
int m,i,j,k,c;
int s,t,a,b;
int sr[MAXX],sc[MAXX];
char buf[11];

inline void gao(int x,int y)
{
   switch(buf[0])
   {
      case '>':
         mat[x][y][0]=std::max(mat[x][y][0],k+1);
         if(mat[x][y][0]>mat[x][y][1])
            flag=true;
         break;
      case '=':
         if(k<mat[x][y][0] || k>mat[x][y][1])
            flag=true;
         mat[x][y][0]=mat[x][y][1]=k;
         break;
      case '<':
         mat[x][y][1]=std::min(mat[x][y][1],k-1);
         if(mat[x][y][0]>mat[x][y][1])
            flag=true;
         break;
   }
}

int main()
{
   bg=true;
   scanf("%d",&T);
   while(T--)
   {
      if(!bg)
         puts("");
      memset(mat,0,sizeof mat);
      scanf("%d %d",&n,&m);
      for(i=1;i<=n;++i)
         scanf("%d",sr+i);
      for(i=1;i<=m;++i)
         scanf("%d",sc+i);
      s=n+m+1;
      t=s+1;
      source=t+1;
      sink=source+1;
      for(i=1;i<=n;++i)
         for(j=1;j<=m;++j)
         {
            mat[i][j+n][0]=0;
            mat[i][j+n][1]=inf;
         }
      bg=flag=false;
      scanf("%d",&c);
      while(c--)
      {
         scanf("%d %d %s %d",&i,&j,buf,&k);
         if(i)
            if(j)
               gao(i,j+n);
            else
               for(j=1;j<=m;++j)
                  gao(i,j+n);
         else
            if(j)
               for(i=1;i<=n;++i)
                  gao(i,j+n);
            else
               for(i=1;i<=n;++i)
                  for(j=1;j<=m;++j)
                     gao(i,j+n);
      }
      if(flag)
      {
         puts("IMPOSSIBLE");
         continue;
      }
      memset(cap,0,sizeof cap);
      for(i=1;i<=n;++i)
         mat[s][i][0]=mat[s][i][1]=sr[i];
      for(i=1;i<=m;++i)
         mat[i+n][t][0]=mat[i+n][t][1]=sc[i];

      a=0;
      for(i=1;i<=t;++i)
      {
         b=0;
         for(j=1;j<=t;++j)
         {
            b+=mat[j][i][0]-mat[i][j][0];
            cap[i][j]=mat[i][j][1]-mat[i][j][0];
         }
         if(b>0)
            a+=(cap[source][i]=b);
         else
            cap[i][sink]=-b;
      }
      cap[t][s]=inf;
      c=n;
      n=sink;
      for(i=1;i<=n;++i)
         last[i]=1;
      for(b=0;bfs();b+=dfs(source,inf));

// printf("%d %d\n",a,b);
      if(a!=b)
         puts("IMPOSSIBLE");
      else
      {
         n=c;
         for(i=1;i<=n;++i)
         {
            for(j=1;j<=m;++j)
               printf("%d ",cap[j+n][i]+mat[i][j+n][0]);
            puts("");
         }
      }
   }
   return 0;
}
```

5.13 Flow network

```
Maximum weighted closure of a graph:

closure

inf



sum{}-{}





Eulerian circuit:




:
1 //
1
abs(/2)
abs(/2)

trick

pathcircuit




Feasible flow problem:
refer Feasible flow problem.cpp

0inf

<a->b cap{u,d}><ss->b cap(u)><a->st cap(u)><a->b cap(d-u)>

Maximum flow: //

Minimum flow: //
t->s
->
tips:





Minimum cost feasible flow problem:
TODO





Minimum weighted vertex cover edge for bipartite graph:
for all vertex in X:
edge < s->x cap(weight(x)) >
for all vertex in Y:
edge < y->t cap(weight(y)) >
for original edges
edge < x->y cap(inf) >

ans={maximum flow}={minimum cut}
( ( && ) || ( && ) )




Maximum weighted vertex independent set for bipartite graph:
ans=Sum{}-value{Minimum weighted vertex cover edge}





:



/inf

ans=sum{}-{}



:



cap[i^1]cap[i]



```

```
void rr(int now)
{
   done[now]=true;
   ++cnt;
   for(int i(edge[now]);i!=-1;i=nxt[i])
      if(cap[i] && !done[v])
         rr(v);
}

void dfs(int now)
{
   done[now]=true;
   ++cnt;
   for(int i(edge[now]);i!=-1;i=nxt[i])
      if(cap[i^1] && !done[v])
         dfs(v);
}

memset(done,0,sizeof done);
cnt=0;
rr(source);
dfs(sink);
puts(cnt==n?"UNIQUE":"AMBIGUOUS");



Tips:
;
inf;
;
inf;
```

5.14 Hamiltonian circuit

```
//if every point connect with not less than [(N+1)/2] points
#include<cstdio>
#include<algorithm>
#include<cstring>

#define MAXX 177
#define MAX (MAXX*MAXX)

int edge[MAXX],nxt[MAX],to[MAX],cnt;

inline void add(int a,int b)
{
   nxt[++cnt]=edge[a];
   edge[a]=cnt;
   to[cnt]=b;
}

bool done[MAXX];
int n,m,i,j,k;

inline int find(int a)
{
   static int i;
   for(i=edge[a];i;i=nxt[i])
      if(!done[to[i]])
      {
         edge[a]=nxt[i];
         return to[i];
      }
   return 0;
}

int a,b;
int next[MAXX],pre[MAXX];
bool mat[MAXX][MAXX];

int main()
{
   while(scanf("%d %d",&n,&m)!=EOF)
   {
      for(i=1;i<=n;++i)
         next[i]=done[i]=edge[i]=0;
      memset(mat,0,sizeof mat);
      cnt=0;
      while(m--)
      {
         scanf("%d %d",&i,&j);
         add(i,j);
         add(j,i);
         mat[i][j]=mat[j][i]=true;
      }
      a=1;
      b=to[edge[a]];
      cnt=2;
      done[a]=done[b]=true;
      next[a]=b;
      while(cnt<n)
      {
         while(i=find(a))
         {
            next[i]=a;
            done[a=i]=true;
            ++cnt;
         }
         while(i=find(b))
         {
            next[b]=i;
            done[b=i]=true;
            ++cnt;
         }
         if(!mat[a][b])
            for(i=next[a];next[i]!=b;i=next[i])
               if(mat[a][next[i]] && mat[i][b])
               {
                  for(j=next[i];j!=b;j=next[j])
                     pre[next[j]]=j;
                  for(j=b;j!=next[i];j=pre[j])
                     next[j]=pre[j];
                  std::swap(next[i],b);
                  break;
               }
         next[b]=a;
         for(i=a;i!=b;i=next[i])
            if(find(i))
            {
               a=next[b=i];
               break;
            }
      }
      while(a!=b)
      {
         printf("%d ",a);
         a=next[a];
      }
      printf("%d\n",b);
   }
   return 0;
}
```

5.15 Hopcroft-Karp algorithm

```
#include<cstdio>
#include<cstring>

#define MAXX 50111
#define MAX 150111

int nx,p;
int i,j,k;
int x,y;
int ans;
bool flag;

int edge[MAXX],nxt[MAX],to[MAX],cnt;

int cx[MAXX],cy[MAXX];
int px[MAXX],py[MAXX];

int q[MAXX],*qf,*qb;

bool ag(int i)
{
   int j,k;
   for(k=edge[i];k;k=nxt[k])
      if(py[j=to[k]]==px[i]+1)
      {
         py[j]=0;
         if(cy[j]==-1 || ag(cy[j]))
         {
            cx[i]=j;
            cy[j]=i;
            return true;
         }
      }
   return false;
}

int main()
{
   scanf("%d %*d %d",&nx,&p);
   while(p--)
   {
      scanf("%d %d",&i,&j);
      nxt[++cnt]=edge[i];
      edge[i]=cnt;
      to[cnt]=j;
   }
   memset(cx,-1,sizeof cx);
   memset(cy,-1,sizeof cy);
   while(true)
   {
      memset(px,0,sizeof(px));
      memset(py,0,sizeof(py));
      qf=qb=q;
      flag=false;

      for(i=1;i<=nx;++i)
         if(cx[i]==-1)
            *qb++=i;
      while(qf!=qb)
         for(k=edge[i=*qf++];k;k=nxt[k])
            if(!py[j=to[k]])
            {
               py[j]=px[i]+1;
               if(cy[j]==-1)
                  flag=true;
               else
               {
                  px[cy[j]]=py[j]+1;
                  *qb++=cy[j];
               }
            }
      if(!flag)
         break;
      for(i=1;i<=nx;++i)
         if(cx[i]==-1 && ag(i))
            ++ans;
   }
   printf("%d\n",ans);
   return 0;
}
```

5.16 Improved Shortest Augmenting Path Algorithm

```
#include<cstdio>
#include<cstring>
```

```
#include<algorithm>

#define MAXX 5111
#define MAXM (30111*4)
#define inf 0x3f3f3f3f3f3f3f3fll

int edge[MAXX],to[MAXM],nxt[MAXM],cnt;
#define v to[i]
long long cap[MAXM];

int n;
int h[MAXX],gap[MAXX],pre[MAXX],w[MAXX];

inline void add(int a,int b,long long c)
{
   nxt[++cnt]=edge[a];
   edge[a]=cnt;
   to[cnt]=b;
   cap[cnt]=c;
}

int source,sink;

inline long long go()
{
   static int now,N,i;
   static long long min,mf;
   memset(gap,0,sizeof gap);
   memset(h,0,sizeof h);
   memcpy(w,edge,sizeof w);
   gap[0]=N=sink; // caution
   mf=0;

   pre[now=source]=-1;
   while(h[source]<N)
   {
      if(now==sink)
      {
         min=inf;
         for(i=pre[now];i!=-1;i=pre[to[i^1]])
            min=std::min(min,cap[i]);
         for(i=pre[now];i!=-1;i=pre[to[i^1]])
         {
            cap[i]-=min;
            cap[i^1]+=min;
         }
         now=source;
         mf+=min;
      }
      for(i=w[now];i!=-1;i=nxt[i])
         if(cap[i] && h[v]+1==h[now])
         {
            w[now]=pre[v]=i;
            now=v;
            break;
         }
      if(i!=-1)
         continue;
      if(!--gap[h[now]])
         return mf;
      min=N;
      for(i=w[now]=edge[now];i!=-1;i=nxt[i])
         if(cap[i])
            min=std::min(min,(long long)h[v]);
      ++gap[h[now]=min+1];
      if(now!=source)
         now=to[pre[now]^1];
   }
   return mf;
}

int m,i,j,k;
long long ans;

int main()
{
   scanf("%d %d",&n,&m);
   source=1;
   sink=n;
   cnt=-1;
   memset(edge,-1,sizeof edge);
   while(m--)
   {
      scanf("%d %d %lld",&i,&j,&ans);
      add(i,j,ans);
      add(j,i,ans);
   }
   printf("%lld\n",go());
   return 0;
}
```

5.17 k Shortest Path

```
#include<cstdio>
#include<cstring>
#include<queue>
#include<vector>

int K;

class states
{
   public:
      int cost,id;
};

int dist[1000];

class cmp
{
   public:
      bool operator ()(const states &i,const states &j)
      {
         return i.cost>j.cost;
      }
};

class cmp2
{
   public:
      bool operator ()(const states &i,const states &j)
      {
         return i.cost+dist[i.id]>j.cost+dist[j.id];
      }
};

struct edges
{
   int to,next,cost;
} edger[100000],edge[100000];

int headr[1000],head[1000],Lr,L;

void dijkstra(int s)
{
   states u;
   u.id=s;
   u.cost=0;
   dist[s]=0;
   std::priority_queue<states,std::vector<states>,cmp> q;
   q.push(u);
   while (!q.empty())
   {
      u=q.top();
      q.pop();
      if (u.cost!=dist[u.id])
         continue;
      for (int i=headr[u.id]; i!=-1; i=edger[i].next)
      {
         states v=u;
         v.id=edger[i].to;
         if (dist[v.id]>dist[u.id]+edger[i].cost)
         {
            v.cost=dist[v.id]=dist[u.id]+edger[i].cost;
            q.push(v);
         }
      }
   }
}

int num[1000];

inline void init(int n)
{
   Lr=L=0;
   memset(head,-1,4*n);
   memset(headr,-1,4*n);
   memset(dist,63,4*n);
   memset(num,0,4*n);
}

void add_edge(int u,int v,int x)
{
   edge[L].to=v;
   edge[L].cost=x;
   edge[L].next=head[u];
   head[u]=L++;
   edger[Lr].to=u;
   edger[Lr].cost=x;
   edger[Lr].next=headr[v];
   headr[v]=Lr++;
}

inline int a_star(int s,int t)
{
   if (dist[s]==0x3f3f3f3f)
      return -1;
   std::priority_queue<states,std::vector<states>,cmp2> q;
   states tmp;
   tmp.id=s;
   tmp.cost=0;
   q.push(tmp);
   while (!q.empty())
   {
      states u=q.top();
      q.pop();
      num[u.id]++;
      if (num[t]==K)
         return u.cost;
      for (int i=head[u.id]; i!=-1; i=edge[i].next)
      {
         int v=edge[i].to;
         tmp.id=v;
         tmp.cost=u.cost+edge[i].cost;
         q.push(tmp);
      }
   }
   return -1;
}

int main()
{
   int n,m;
   scanf("%d%d",&n,&m);
   init(n);
   for (int i=0; i<m; i++)
   {
      int u,v,x;
      scanf("%d%d%d",&u,&v,&x);
      add_edge(u-1,v-1,x);
   }
   int s,t;
   scanf("%d%d%d",&s,&t,&K);
   if (s==t)
      ++K;
   dijkstra(t-1);
   printf("%d\n",a_star(s-1,t-1));
}
```

5.18 Kariv-Hakimi Algorithm

```
#include<cstdio>
#include<algorithm>
#include<vector>
#include<cstring>
#include<set>

#define MAXX 211
#define inf 0x3f3f3f3f

int e[MAXX][MAXX],dist[MAXX][MAXX];
double dp[MAXX],ta;
int ans,d;
int n,m,a,b;
int i,j,k;
typedef std::pair<int,int> pii;
std::vector<pii>vt[2];
bool done[MAXX];
typedef std::pair<double,int> pdi;
std::multiset<pdi>q;
int pre[MAXX];

int main()
{
   vt[0].reserve(MAXX);
   vt[1].reserve(MAXX);
   scanf("%d %d",&n,&m);
   memset(e,0x3f,sizeof(e));
   while(m--)
   {
      scanf("%d %d %d",&i,&j,&k);
      e[i][j]=e[j][i]=std::min(e[i][j],k);
   }
   for(i=1;i<=n;++i)
      e[i][i]=0;
   memcpy(dist,e,sizeof(dist));
   for(k=1;k<=n;++k)
      for(i=1;i<=n;++i)
         for(j=1;j<=n;++j)
            dist[i][j]=std::min(dist[i][j],dist[i][k]+dist[k][j]);
   ans=inf;
   for(i=1;i<=n;++i)
      for(j=i;j<=n;++j)
         if(e[i][j]!=inf)
         {
            vt[0].resize(0);
            vt[1].resize(0);
            static int i;
            for(i=1;i<=n;++i)
               vt[0].push_back(pii(dist[::i][i],dist[j][i]));
            std::sort(vt[0].begin(),vt[0].end());
            for(i=0;i<vt[0].size();++i)
            {
               while(!vt[1].empty() && vt[1].back().second<=vt[0][i].second)
                  vt[1].pop_back();
               vt[1].push_back(vt[0][i]);
            }
            d=inf;
            if(vt[1].size()==1)
               if(vt[1][0].first<vt[1][0].second)
               {
                  ta=0;
                  d=(vt[1][0].first<<1);
               }
               else
               {
                  ta=e[::i][j];
                  d=(vt[1][0].second<<1);
               }
            else
               for(i=1;i<vt[1].size();++i)
                  if(d>e[::i][j]+vt[1][i-1].first+vt[1][i].second)
                  {
                     ta=(e[::i][j]+vt[1][i].second-vt[1][i-1].first)/(double)2.0f;
                     d=e[::i][j]+vt[1][i-1].first+vt[1][i].second;
                  }
            if(d<ans)
            {
               ans=d;
               a=::i;
               b=j;
               dp[::i]=ta;
               dp[j]=e[::i][j]-ta;
            }
         }
   printf("%d\n",ans);
   for(i=1;i<=n;++i)
      if(i!=a && i!=b)
         dp[i]=1e20;
   q.insert(pdi(dp[a],a));
   if(a!=b)
      q.insert(pdi(dp[b],b));
   if(a!=b)
      pre[b]=a;
   while(!q.empty())
   {
      k=q.begin()->second;
      q.erase(q.begin());
      if(done[k])
         continue;
      done[k]=true;
      for(i=1;i<=n;++i)
         if(e[k][i]!=inf && dp[k]+e[k][i]<dp[i])
         {
            dp[i]=dp[k]+e[k][i];
            q.insert(pdi(dp[i],i));
            pre[i]=k;
         }
   }
   vt[0].resize(0);
   for(i=1;i<=n;++i)
      if(pre[i])
         if(i<pre[i])
            printf("%d %d\n",i,pre[i]);
         else
            printf("%d %d\n",pre[i],i);
   return 0;
}
```

5.19 Kuhn-Munkres algorithm

```
bool match(int u)//
{
   vx[u]=true;
   for(int i=1;i<=n;++i)
      if(lx[u]+ly[i]==g[u][i]&&!vy[i])
      {
         vy[i]=true;
         if(!d[i]||match(d[i]))
         {
            d[i]=u;
            return true;
         }
      }
   return false;
}
inline void update()//
{
   int i,j;
   int a=1<<30;
   for(i=1;i<=n;++i)if(vx[i])
      for(j=1;j<=n;++j)if(!vy[j])
         a=min(a,lx[i]+ly[j]-g[i][j]);
   for(i=1;i<=n;++i)
   {
      if(vx[i])lx[i]-=a;
      if(vy[i])ly[i]+=a;
   }
}
void km()
{
   int i,j;
   for(i=1;i<=n;++i)
   {
      lx[i]=ly[i]=d[i]=0;
      for(j=1;j<=n;++j)
         lx[i]=max(lx[i],g[i][j]);
   }
   for(i=1;i<=n;++i)
   {
      while(true)
      {
         memset(vx,0,sizeof(vx));
         memset(vy,0,sizeof(vy));
         if(match(i))
            break;
         update();
      }
   }
   int ans=0;
   for(i=1;i<=n;++i)
      if(d[i]!=0)
         ans+=g[d[i]][i];
   printf("%d\n",ans);
}
int main()
{
   while(scanf("%d\n",&n)!=EOF)
   {
      for(int i=1;i<=n;++i)gets(s[i]);
      memset(g,0,sizeof(g));
      for(int i=1;i<=n;++i)
         for(int j=1;j<=n;++j)
            if(i!=j) g[i][j]=cal(s[i],s[j]);
      km();
   }
   return 0;
}


//bupt

//km n^3
int dfs(int u)//
{
   int v;
   sx[u]=1;
   for ( v=1; v<=n; v++)
      if (!sy[v] && lx[u]+ly[v]==map[u][v])
      {
         sy[v]=1;
         if (match[v]==-1 || dfs(match[v]))
         {
            match[v]=u;
            return 1;
         }
      }
   return 0;
}

int bestmatch(void)//km
{
   int i,j,u;
   for (i=1; i<=n; i++)//
   {
      lx[i]=-1;
      ly[i]=0;
      for (j=1; j<=n; j++)
         if (lx[i]<map[i][j])
            lx[i]=map[i][j];
   }
   memset(match, -1, sizeof(match));
   for (u=1; u<=n; u++)
   {
      while (true)
      {
         memset(sx,0,sizeof(sx));
         memset(sy,0,sizeof(sy));
         if (dfs(u))
```

```
            break;
         int dx=Inf;//~~
         for (i=1; i<=n; i++)
         {
            if (sx[i])
               for (j=1; j<=n; j++)
                  if(!sy[j] && dx>lx[i]+ly[j]-map[i][j])
                     dx=lx[i]+ly[j]-map[i][j];
         }
         for (i=1; i<=n; i++)
         {
            if (sx[i])
               lx[i]-=dx;
            if (sy[i])
               ly[i]+=dx;
         }
      }
   }
   int sum=0;
   for (i=1; i<=n; i++)
      sum+=map[match[i]][i];
   return sum;
}
```

5.20 LCA - DA

```
int edge[MAXX],nxt[MAXX<<1],to[MAXX<<1],cnt;
int pre[MAXX][N],dg[MAXX];

inline void add(int j,int k)
{
   nxt[++cnt]=edge[j];
   edge[j]=cnt;
   to[cnt]=k;
}

void rr(int now,int fa)
{
   dg[now]=dg[fa]+1;
   for(int i(edge[now]);i;i=nxt[i])
      if(to[i]!=fa)
      {
         static int j;
         j=1;
         for(pre[to[i]][0]=now;j<N;++j)
            pre[to[i]][j]=pre[pre[to[i]][j-1]][j-1];
         rr(to[i],now);
      }
}

inline int lca(int a,int b)
{
   static int i,j;
   j=0;
   if(dg[a]<dg[b])
      std::swap(a,b);
   for(i=dg[a]-dg[b];i;i>>=1,++j)
      if(i&1)
         a=pre[a][j];
   if(a==b)
      return a;
   for(i=N-1;i>=0;--i)
      if(pre[a][i]!=pre[b][i])
      {
         a=pre[a][i];
         b=pre[b][i];
      }
   return pre[a][0];

// looks like above is a wrong version

   static int i,log;
   for(log=0;(1<<(log+1))<=dg[a];++log);
   for(i=log;i>=0;--i)
      if(dg[a]-(1<<i)>=dg[b])
         a=pre[a][i];
   if(a==b)
      return a;
   for(i=log;i>=0;--i)
      if(pre[a][i]!=-1 && pre[a][i]!=pre[b][i])
         a=pre[a][i],b=pre[b][i];
   return pre[a][0];
}
```

5.21 LCA - tarjan - minmax

```
#include<cstdio>
#include<list>
#include<algorithm>
#include<cstring>

#define MAXX 100111
#define inf 0x5fffffff

short T,t;
int set[MAXX],min[MAXX],max[MAXX],ans[2][MAXX];
bool done[MAXX];
std::list<std::pair<int,int> >edge[MAXX];
std::list<std::pair<int,int> >q[MAXX];
int n,i,j,k,l,m;

struct node
{
   int a,b,id;
   node() {}
   node(const int &aa,const int &bb,const int &idd): a(aa),b(bb),id(idd){}
};

std::list<node>to[MAXX];

int find(const int &a)
{
   if(set[a]==a)
      return a;
   int b(set[a]);
   set[a]=find(set[a]);
   max[a]=std::max(max[a],max[b]);
   min[a]=std::min(min[a],min[b]);
   return set[a];
}

void tarjan(const int &now)
{
   done[now]=true;
   for(std::list<std::pair<int,int> >::const_iterator it(q[now].begin());it!=q[now].end();++it)
      if(done[it->first])
         if(it->second>0)
            to[find(it->first)].push_back(node(now,it->first,it->second));
         else
            to[find(it->first)].push_back(node(it->first,now,-it->second));
   for(std::list<std::pair<int,int> >::const_iterator it(edge[now].begin());it!=edge[now].end();++it)
      if(!done[it->first])
      {
         tarjan(it->first);
         set[it->first]=now;
         min[it->first]=it->second;
         max[it->first]=it->second;
      }
   for(std::list<node>::const_iterator it(to[now].begin());it!=to[now].end();++it)
   {
      find(it->a);
      find(it->b);
      ans[0][it->id]=std::min(min[it->b],min[it->a]);
      ans[1][it->id]=std::max(max[it->a],max[it->b]);
   }
}

int main()
{
   scanf("%hd",&T);
   for(t=1;t<=T;++t)
   {
      scanf("%d",&n);
      for(i=1;i<=n;++i)
      {
         edge[i].clear();
         q[i].clear();
         to[i].clear();
         done[i]=false;
         set[i]=i;
         min[i]=inf;
         max[i]=0;
      }
      for(i=1;i<n;++i)
      {
         scanf("%d%d%d",&j,&k,&l);
         edge[j].push_back(std::make_pair(k,l));
         edge[k].push_back(std::make_pair(j,l));
      }
      scanf("%d",&m);
      for(i=0;i<m;++i)
      {
         scanf("%d %d",&j,&k);
         q[j].push_back(std::make_pair(k,i));
         q[k].push_back(std::make_pair(j,-i));
      }
      tarjan(1);
      printf("Case %hd:\n",t);
      for(i=0;i<m;++i)
         printf("%d %d\n",ans[0][i],ans[1][i]);
   }
   return 0;
}
```

5.22 Minimum Ratio Spanning Tree

```
#include<cstdio>
#include<cstring>
#include<cmath>

#define MAXX 1111

struct
{
   int x,y;
   double z;
} node[MAXX];

struct
{
   double l,c;
} map[MAXX][MAXX];

int n,l,f[MAXX],pre[MAXX];
double dis[MAXX];

double mst(double x)
{
   int i,j,tmp;
   double min,s=0,t=0;
   memset(f,0,sizeof(f));
   f[1]=1;
   for (i=2; i<=n; i++)
   {
      dis[i]=map[1][i].c-map[1][i].l*x;
      pre[i]=1;
   }
   for (i=1; i<n; i++)
   {
      min=1e10;
      for (j=1; j<=n; j++)
         if (!f[j] && min>dis[j])
         {
            min=dis[j];
            tmp=j;
         }
      f[tmp]=1;
      t+=map[pre[tmp]][tmp].l;
      s+=map[pre[tmp]][tmp].c;
      for (j=1; j<=n; j++)
         if (!f[j] && map[tmp][j].c-map[tmp][j].l*x<dis[j])
         {
            dis[j]=map[tmp][j].c-map[tmp][j].l*x;
            pre[j]=tmp;
         }
   }
   return s/t;
}

int main()
{
   int i,j;
   double a,b;
   while (scanf("%d",&n),n);
   {
      for (i=1; i<=n; i++)
         scanf("%d%d%lf",&node[i].x,&node[i].y,&node[i].z);
      for (i=1; i<=n; i++)
         for (j=i+1; j<=n; j++)
         {
            map[j][i].l=map[i][j].l=sqrt(1.0*(node[i].x-node[j].x)
                  *(node[i].x-node[j].x)+(node[i].y-node[j].y)*(
                  node[i].y-node[j].y));
            map[j][i].c=map[i][j].c=fabs(node[i].z-node[j].z);
         }
      a=0,b=mst(a);
      while (fabs(b-a)>1e-8)
      {
         a=b;
         b=mst(a);
      }
      printf("%.3lf\n",b);
   }
   return 0;

}
```

5.23 Minimum-cost flow problem

```
// like Edmonds-Karp Algorithm
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<queue>

#define MAXX 5011
#define MAXE (MAXX*10*2)
#define inf 0x3f3f3f3f

int edge[MAXX],nxt[MAXE],to[MAXE],cap[MAXE],cst[MAXE],cnt;
#define v to[i]
inline void adde(int a,int b,int c,int d)
{
   nxt[++cnt]=edge[a];
   edge[a]=cnt;
   to[cnt]=b;
   cap[cnt]=c;
   cst[cnt]=d;
}
inline void add(int a,int b,int c,int d)
{ adde(a,b,c,d);adde(b,a,0,-d);}

int dist[MAXX],pre[MAXX];
int source,sink;
std::queue<int>q;
bool in[MAXX];

inline bool go()
{
   static int now,i;
   memset(dist,0x3f,sizeof dist);
   dist[source]=0;
   pre[source]=-1;
   q.push(source);
   in[source]=true;
   while(!q.empty())
   {
      in[now=q.front()]=false;
      q.pop();
      for(i=edge[now];i!=-1;i=nxt[i])
         if(cap[i] && dist[v]>dist[now]+cst[i])
         {
            dist[v]=dist[now]+cst[i];
            pre[v]=i;
            if(!in[v])
            {
               q.push(v);
               in[v]=true;
            }
         }
   }
   return dist[sink]!=inf;
}

inline int mcmf(int &flow)
{
   static int ans,i;
   flow=ans=0;
   while(go())
   {
      static int min;
      min=inf;
      for(i=pre[sink];i!=-1;i=pre[to[i^1]])
         min=std::min(min,cap[i]);
      flow+=min;
      ans+=min*dist[sink];
      for(i=pre[sink];i!=-1;i=pre[to[i^1]])
      {
         cap[i]-=min;
         cap[i^1]+=min;
      }
   }
   return ans;
}

// TQ's version
struct mcmf
{
   struct Edge
   {
      int from,to,cap,flow,cost;
   };
   int n,m,s,t;
   std::vector<Edge>edges;
   std::vector<int>G[maxn];
   int inq[maxn],d[maxn],p[maxn],a[maxn];

   void init(int n)
   {
      this->n=n;
      for(int i=0;i<n;++i)
         G[i].clear();
      edges.clear();
   }

   void addedge(int from,int to,int cap,int cost)
   {
      Edge x={from,to,cap,0,cost};
      edges.push_back(x);
      Edge y={to,from,0,0,-cost};
      edges.push_back(y);
      m=edges.size();
      G[from].push_back(m-2);
      G[to].push_back(m-1);
   }
   int mincost(int s,int t)
   {
      int flow=0,cost=0;
      while(BellmanFord(s,t,flow,cost));
      if(flow!=(n-1)/2)return -1;
      return cost;
   }
private:
   bool BellmanFord(int s,int t,int& flow,int& cost)
   {
      for(int i=0;i<=n;++i)
         d[i]=INF;
      memset(inq,0,sizeof(inq));
      d[s]=0; inq[s]=1; p[s]=0; a[s]=INF;
      std::queue<int>Q;
      Q.push(s);
      while(!Q.empty())
      {
         int u=Q.front();
         Q.pop();
         inq[u]=0;
         for(int i=0;i<G[u].size();++i)
         {
            Edge& e=edges[G[u][i]];
            if(e.cap>e.flow && d[e.to]>d[u]+e.cost)
            {
               d[e.to]=d[u]+e.cost;
               p[e.to]=G[u][i];
               a[e.to]=min(a[u],e.cap-e.flow);
               if(!inq[e.to])
               {
                  Q.push(e.to);
                  inq[e.to]=1;
               }
            }
         }
      }
      if(d[t]==INF)
         return false;
      flow+=a[t];
      cost+=d[t]*a[t];
      int u=t;
      while(u!=s)
      {
         edges[p[u]].flow+=a[t];
         edges[p[u]^1].flow-=a[t];
         u=edges[p[u]].from;
      }
      return true;
   }

}G;
```

5.24 Stable Marriage

```
//

while(!g.empty()) //
{
   if(dfn[edge[g.front()].front()]==-1)
      dfn[edge[g.front()].front()]=g.front(); //
   else
   {
      for(it=edge[edge[g.front()].front()].begin();it!=edge[edge[g
            .front()].front()].end();++it)
         if(*it==dfn[edge[g.front()].front()] || *it==g.front())
               //
            break;
      if(*it==g.front()) //
      {
         g.push_back(dfn[edge[g.front()].front()]);
         dfn[edge[g.front()].front()]=g.front();
      }
      else
         g.push_back(g.front()); //
   }
   edge[g.front()].pop_front(); //
   g.pop_front();
}
```

5.25 Stoer-Wagner Algorithm

```
#include <iostream>
using namespace std;
const int maxn=510;
int map[maxn][maxn];
int n;
void contract(int x,int y)//
{
  int i,j;
   for (i=0; i<n; i++)
   if (i!=x) map[x][i]+=map[y][i],map[i][x]+=map[i][y];
   for (i=y+1; i<n; i++) for (j=0; j<n; j++)
   {
      map[i-1][j]=map[i][j];
      map[j][i-1]=map[j][i];
   }
   n--;
}
int w[maxn],c[maxn];
int sx,tx;
int mincut()
//
{
   int i,j,k,t;
   memset(c,0,sizeof(c));
   c[0]=1;
   for (i=0; i<n; i++) w[i]=map[0][i];
   for (i=1; i+1<n; i++)
   {
      t=k=-1;
      for (j=0; j<n; j++) if (c[j]==0&&w[j]>k)
      k=w[t=j];
      c[sx=t]=1;
      for (j=0; j<n; j++) w[j]+=map[t][j];
   }
   for (i=0; i<n; i++) if (c[i]==0) return w[tx=i];
}
int main()
{
   int i,j,k,m;
   while (scanf("%d%d",&n,&m)!=EOF)
   {
      memset(map,0,sizeof(map));
      while (m--)
      {
         scanf("%d%d%d",&i,&j,&k);
         map[i][j]+=k;
         map[j][i]+=k;
      }
      int mint=999999999;
      while (n>1)
      {
         k=mincut();
         if (k<mint) mint=k;
         contract(sx,tx);
      }
   printf("%d\n",mint);
   }
return 0;
}
```

5.26 Strongly Connected Component

```
//
void dfs(const short &now)
{
   dfn[now]=low[now]=cnt++;
   st.push(now);
   for(std::list<short>::const_iterator it(edge[now].begin());it!=
         edge[now].end();++it)
      if(dfn[*it]==-1)
      {
         dfs(*it);
         low[now]=std::min(low[now],low[*it]);
      }
      else
         if(sc[*it]==-1)
            low[now]=std::min(low[now],dfn[*it]);
   if(dfn[now]==low[now])
   {
      while(sc[now]==-1)
      {
         sc[st.top()]=p;
         st.pop();
      }
      ++p;
   }
}
```

5.27 ZKW's Minimum-cost flow

```
#include<cstdio>
#include<algorithm>
#include<cstring>
#include<vector>
#include<deque>

#define MAXX 111
#define MAXN 211
#define MAXE (MAXN*MAXN*3)
#define inf 0x3f3f3f3f

char buf[MAXX];

int edge[MAXN],nxt[MAXE],to[MAXE],cap[MAXE],cst[MAXE],cnt;

inline void adde(int a,int b,int c,int k)
{
   nxt[cnt]=edge[a];
   edge[a]=cnt;
   to[cnt]=b;
   cap[cnt]=c;
   cst[cnt]=k;
   ++cnt;
}

inline void add(int a,int b,int c,int k)
{
   adde(a,b,c,k);
   adde(b,a,0,-k);
}

int n,mf,cost,pi1;
int source,sink;
bool done[MAXN];

int aug(int now,int maxcap)
{
   if(now==sink)
   {
      mf+=maxcap;
      cost+=maxcap*pi1;
      return maxcap;
   }
   done[now]=true;
   int l=maxcap;
   for(int i(edge[now]);i!=-1;i=nxt[i])
      if(cap[i] && !cst[i] && !done[to[i]])
      {
         int d(aug(to[i],std::min(l,cap[i])));
         cap[i]-=d;
         cap[i^1]+=d;
         l-=d;
         if(!l)
            return maxcap;
      }
   return maxcap-l;
}

inline bool label()
{
   static int d,i,j;
   d=inf;
   for(i=1;i<=n;++i)
      if(done[i])
         for(j=edge[i];j!=-1;j=nxt[j])
            if(cap[j] && !done[to[j]] && cst[j]<d)
               d=cst[j];
   if(d==inf)
      return false;
   for(i=1;i<=n;++i)
      if(done[i])
         for(j=edge[i];j!=-1;j=nxt[j])
         {
            cst[j]-=d;
            cst[j^1]+=d;
         }
   pi1+=d;
   return true;
   /* primal-dual approach
   static int d[MAXN],i,j;
   static std::deque<int>q;
   memset(d,0x3f,sizeof d);
   d[sink]=0;
   q.push_back(sink);
   while(!q.empty())
   {
      static int dt,now;
      now=q.front();
      q.pop_front();
      for(i=edge[now];i!=-1;i=nxt[i])
         if(cap[i^1] && (dt=d[now]-cst[i])<d[to[i]])
            if((d[to[i]]=dt)<=d[q.empty()?0:q.front()])
               q.push_front(to[i]);
            else
               q.push_back(to[i]);
   }
   for(i=1;i<=n;++i)
      for(j=edge[i];j!=-1;j=nxt[j])
         cst[j]+=d[to[j]]-d[i];
   pi1+=d[source];
   return d[source]!=inf;
   */
}

int m,i,j,k;
typedef std::pair<int,int> pii;
std::vector<pii>M(MAXN),H(MAXN);

int main()
{
   while(scanf("%d %d",&n,&m),(n||m))
   {
      M.resize(0);
      H.resize(0);
      for(i=0;i<n;++i)
      {
         scanf("%s",buf);
         for(j=0;j<m;++j)
            if(buf[j]=='m')
               M.push_back(pii(i,j));
            else
               if(buf[j]=='H')
                  H.push_back(pii(i,j));
      }
      n=M.size()+H.size();
      source=++n;
      sink=++n;
      memset(edge,-1,sizeof edge);
      cnt=0;
      for(i=0;i<M.size();++i)
         for(j=0;j<H.size();++j)
            add(i+1,j+1+M.size(),1,abs(M[i].first-H[j].first)+abs(M[i].second-H[j].second));
      for(i=0;i<M.size();++i)
         add(source,i+1,1,0);
      for(i=0;i<H.size();++i)
         add(i+1+M.size(),sink,1,0);
      mf=cost=pi1=0;
      do
         do
            memset(done,0,sizeof done);
         while(aug(source,inf));
      while(label());
      /* primal-dual approach
      while(label())
         do
            memset(done,0,sizeof done);
         while(aug(source,inf));
      */
      printf("%d\n",cost);
   }
   return 0;
}
```

5.28 ZKW's SAP

```
// wrong answer at poj 1149
// wrong answer at uestc 1195
#include<cstdio>
#include<algorithm>
#include<cstring>

#define MAXX 5111
#define MAXM (30111*4)
#define inf 0x3f3f3f3f3f3f3f3fll

int edge[MAXX],to[MAXM],nxt[MAXM],cnt;
int w[MAXX];
long long cap[MAXM];

int n;
int h[MAXX],vh[MAXX];

inline void add(int a,int b,long long c)
{
   nxt[cnt]=edge[a];
   edge[a]=cnt;
   to[cnt]=b;
   cap[cnt]=c;
   ++cnt;
}

int source,sink;

long long aug(int now,long long flow)
{
   if(now==sink)
      return flow;
   long long l(flow);
   for(int &i(edge[now]);i!=-1;i=nxt[i])
      if(cap[i] && h[to[i]]+1==h[now])
      {
         long long d(aug(to[i],std::min(l,cap[i])));
         cap[i]-=d;
         cap[i^1]+=d;
         l-=d;
         if(h[source]==n || !l)
            return flow-l;
      }
   int minh(n);
   for(int i(edge[now]=w[now]);i!=-1;i=nxt[i])
      if(cap[i] && h[to[i]]+1<minh)
         minh=h[to[i]]+1;
   if(!--vh[h[now]])
      h[source]=n;
   else
      ++vh[h[now]=minh];
   return flow-l;
}

int m,i,j,k;
long long ans;

int main()
{
   scanf("%d %d",&n,&m);
   source=1;
   sink=n;
   memset(edge,-1,sizeof edge);
   while(m--)
   {
      scanf("%d %d %lld",&i,&j,&ans);
      add(i,j,ans);
      add(j,i,0);

      add(j,i,ans);
      add(i,j,0);
   }
   memcpy(w,edge,sizeof edge);
   memset(h,0,sizeof h);
   memset(vh,0,sizeof vh);
   vh[0]=n;
   ans=0;
   while(h[source]<n)
      ans+=aug(source,inf);
   printf("%lld\n",ans);
   return 0;
}
```

6 math

6.1 cantor

```
const int PermSize = 12;
int fac[PermSize] = {1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880, 3628800, 39916800};

inline int Cantor(int a[])
{
   int i, j, cnt;
   int res = 0;
   for (i = 0; i < PermSize; ++i)
   {
      cnt = 0;
      for (j = i + 1; j < PermSize; ++j)
         if (a[i] > a[j])
            ++cnt;
      res = res + cnt * fac[PermSize - i - 1];
   }
   return res;
}

bool h[13];

inline void UnCantor(int x, int res[])
{
   int i,j,l,t;
   for (i = 1;i <= 12;i++)
      h[i] = false;
   for (i = 1; i <= 12; i++)
   {
      t = x / fac[12 - i];
      x -= t * fac[12 - i];
      for (j = 1, l = 0; l <= t; j++)
         if (!h[j])
            l++;
      j--;
      h[j] = true;
      res[i - 1] = j;
   }
}
```

6.2 Continued fraction

```
// not tested yet
#include<cstdio>
#include <iostream>
#include <cmath>
#include <cstring>

#define min(a,b) (a>b?b:a)

long long d[10000], num[10000], dnm[10000];
long long i, p;
long long l1, l2;

void rr(double num)
{
   int sub = floor(num);
   d[i++] = sub;
   if (sub == num)
      return;
   if (i > 2000)
      return;
   rr(1 / (num - sub));
}

long long numerator(int n)
{
   if (num[n] != 0)
      return num[n];
   long long i = -1;
   if (n == 0)
      i = d[0];
   else
      if (n == 1)
         i = d[0] * d[1] + 1;
      else
         i = numerator(n - 1) * d[n] + numerator(n - 2);
   num[n] = i;
   return i;

   if (i > p)
   {
      l1 = n - 1;
      i = 0;
      num[n] = 0;
   }
   else
      num[n] = i;
   return i;
}

long long denominator(int n)
{
   if (dnm[n] != 0)
```

```
      return dnm[n];
   long long i = -1;
   if (n == 0)
      i = 1;
   else
      if (n == 1)
         i = d[1];
      else
         i = denominator(n - 1) * d[n] + denominator(n - 2);
   dnm[n] = i;
   return i;

   if (i > p)
   {
      l2 = n - 1;
      i = 0;
      dnm[n] = 0;
   }
   else
      dnm[n] = i;
   return i;
}

int main()
{
   int n;
   while (scanf("%d",&n)!=EOF)
   {
      if (n == 0)
         return 0;

      memset(num, 0, sizeof num);
      memset(dnm, 0, sizeof dnm);

      i = 0;
      rr(sqrt((double)n));

      numerator(25);
      denominator(25);

      int f;
      for (f = 0; f < 25; ++f)
         printf("%lld/%lld\n",num[f],dnm[f]);
   }
   return 0;
}
```

6.3 Discrete logarithms - BSGS

```
//The running time of BSGS and the space complexity is O(\sqrt{n})
//Pollard's rho algorithm for logarithms' running time is approximately O(\sqrt{p}) where p is n's largest prime factor.
#include<cstdio>
#include<cmath>
#include<cstring>

struct Hash // std::map is bad. clear()
{
   static const int mod=100003; // prime is good
   static const int MAXX=47111; // bigger than sqrt(c)
   int hd[mod],nxt[MAXX],cnt;
   long long v[MAXX],k[MAXX]; // a^k v (mod c)
   inline void init()
   {
      memset(hd,0,sizeof hd);
      cnt=0;
   }
   inline long long find(long long v)
   {
      static int now;
      for(now=hd[v%mod];now;now=nxt[now])
         if(this->v[now]==v)
            return k[now];
      return -1ll;
   }
   inline void insert(long long k,long long v)
   {
      if(find(v)!=-1ll)
         return;
      nxt[++cnt]=hd[v%mod];
      hd[v%mod]=cnt;
      this->v[cnt]=v;
      this->k[cnt]=k;
   }
}hash;

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

long long exgcd(long long a,long long b,long long &x,long long &y)
{
   if(b)
   {
      long long re(exgcd(b,a%b,x,y)),tmp(x);
      x=y;
      y=tmp-(a/b)*y;
      return re;
   }
   x=1ll;
   y=0ll;
   return a;
}

inline long long bsgs(long long a,long long b,long long c) // a^x b (mod c)
{
   static long long x,y,d,g,m,am,k;
   static int i,cnt;
   a%=c;
   b%=c;
   x=1ll%c; // if c==1....
   for(i=0;i<100;++i)
   {
      if(x==b)
         return i;
      x=(x*a)%c;
   }
   d=1ll%c;
   cnt=0;
   while((g=gcd(a,c))!=1ll)
   {
      if(b%g)
         return -1ll;
      ++cnt;
      c/=g;
      b/=g;
      d=a/g*d%c;
   }
   hash.init();
   m=sqrt((double)c); // maybe need a ceil
   am=1ll%c;
   hash.insert(0,am);
   for(i=1;i<=m;++i)
   {
      am=am*a%c;
      hash.insert(i,am);
   }
   for(i=0;i<=m;++i)
   {
      g=exgcd(d,c,x,y);
      x=(x*b/g%c+c)%c;
      k=hash.find(x);
      if(k!=-1ll)
         return i*m+k+cnt;
      d=d*am%c;
   }
   return -1ll;
}

long long k,p,n;

int main()
{
   while(scanf("%lld %lld %lld",&k,&p,&n)!=EOF)
   {
      if(n>p || (k=bsgs(k,n,p))==-1ll)
         puts("Orz,I cant find D!");
      else
         printf("%lld\n",k);
   }
   return 0;
}
```

6.4 Divisor function

```
sum of positive divisors function
(n)=(pow(p[0],a[0]+1)-1)/(p[0]-1)* (pow(p[1],a[1]+1)-1)/(p[1]-1)* ... (pow(p[n-1],a[n-1]+1)-1);
```

6.5 Extended Euclidean Algorithm

```
//ax+by=gcd(a,b)
long long ex_gcd(long long a,long long b,long long &x,long long &y)
{
   if (b)
   {
      long long ret = ex_gcd(b,a%b,x,y),tmp = x;
      x = y;
      y = tmp-(a/b)*y;
      return ret;
   }
   else
   {
      x = 1;
      y = 0;
      return a;
   }
}
```

6.6 Gaussian elimination

```
#define N

inline int ge(int a[N][N],int n) //
{
   static int i,j,k,l;
   for(j=i=0;j<n;++j) //i,j
   {
      for(k=i;k<n;++k)
         if(a[k][j])
            break;
      if(k==n)
         continue;
      for(l=0;l<=n;++l)
         std::swap(a[i][l],a[k][l]);
      for(l=0;l<=n;++l)
         if(l!=i && a[l][j])
            for(k=0;k<=n;++k)
               a[l][k]^=a[i][k];
      ++i;
   }
   for(j=i;j<n;++j)
      if(a[j][n])
```

```
         return -1; //
   return i;
}
/*
 */

void dfs(int v)
{
   if(v==n)
   {
      static int x[MAXX],ta[MAXX][MAXX];
      static int tmp;
      memcpy(x,ans,sizeof(x));
      memcpy(ta,a,sizeof(ta));
      for(i=l-1;i>=0;--i)
      {
         for(j=i+1;j<n;++j)
            ta[i][n]^=(x[j]&&ta[i][j]); //
         x[i]=ta[i][n];
      }
      for(tmp=i=0;i<n;++i)
         if(x[i])
            ++tmp;
      cnt=std::min(cnt,tmp);
      return;
   }
   ans[v]=0;
   dfs(v+1);
   ans[v]=1;
   dfs(v+1);
}

inline int ge(int a[N][N],int n)
{
   static int i,j,k,l;
   for(i=j=0;j<n;++j)
   {
      for(k=i;k<n;++k)
         if(a[k][i])
            break;
      if(k<n)
      {
         for(l=0;l<=n;++l)
            std::swap(a[i][l],a[k][l]);
         for(k=0;k<n;++k)
            if(k!=i && a[k][i])
               for(l=0;l<=n;++l)
                  a[k][l]^=a[i][l];
         ++i;
      }
      else //
      {
         l=n-1-j+i;
         for(k=0;k<n;++k)
            std::swap(a[k][l],a[k][i]);
      }
   }
   if(i==n)
   {
      for(i=cnt=0;i<n;++i)
         if(a[i][n])
            ++cnt;
      printf("%d\n",cnt);
      continue;
   }
   for(j=i;j<n;++j)
      if(a[j][n])
         break;
   if(j<n)
      puts("impossible");
   else
   {
      memset(ans,0,sizeof(ans));
      cnt=111;
      dfs(l=i);
      printf("%d\n",cnt);
   }
}

/*
 */

inline void ge(int a[N][N],int m,int n) // m*n
{
   static int i,j,k,l,b,c;
   for(i=j=0;i<m && j<n;++j)
   {
      for(k=i;k<m;++k)
         if(a[k][j])
            break;
      if(k==m)
         continue;
      for(l=0;l<=n;++l)
         std::swap(a[i][l],a[k][l]);
      for(k=0;k<m;++k)
         if(k!=i && a[k][j])
         {
            b=a[k][j];
            c=a[i][j];
            for(l=0;l<=n;++l)
               a[k][l]=((a[k][l]*c-a[i][l]*b)%7+7)%7;
         }
      ++i;
   }
   for(j=i;j<m;++j)
      if(a[j][n])
         break;
   if(j<m)
   {
      puts("Inconsistent data.");
      return;
   }
   if(i<n)
      puts("Multiple solutions.");
   else
   {
      memset(ans,0,sizeof(ans));
      for(i=n-1;i>=0;--i)
      {
         k=a[i][n];
         for(j=i+1;j<n;++j)
            k=((k-a[i][j]*ans[j])%7+7)%7;
         while(k%a[i][i])
            k+=7;
         ans[i]=(k/a[i][i])%7;
      }
      for(i=0;i<n;++i)
         printf("%d%c",ans[i],i+1==n?'\n':' ');
   }
}
```

6.7 inverse element

```
inline void getInv2(int x,int mod)
{
   inv[1]=1;
   for (int i=2; i<=x; i++)
      inv[i]=(mod-(mod/i)*inv[mod%i]%mod)%mod;
}

long long power(long long x,long long y,int mod)
{
   long long ret=1;
   for (long long a=x%mod; y; y>>=1,a=a*a%mod)
      if (y&1)
         ret=ret*a%mod;
   return ret;
}

inline int getInv(int x,int mod)//mod
{
   return power(x,mod-2);
}
```

6.8 Linear programming

```
#include<cstdio>
#include<cstring>
#include<cmath>
#include<algorithm>

#define MAXN 33
#define MAXM 33
#define eps 1e-8

double a[MAXN][MAXM],b[MAXN],c[MAXM];
double x[MAXM],d[MAXN][MAXM];
int ix[MAXN+MAXM];
double ans;
int n,m;
int i,j,k,r,s;
double D;

inline bool simplex()
{
   r=n;
   s=m++;
   for(i=0;i<n+m;++i)
      ix[i]=i;
   memset(d,0,sizeof d);
   for(i=0;i<n;++i)
   {
      for(j=0;j+1<m;++j)
         d[i][j]=-a[i][j];
      d[i][m-1]=1;
      d[i][m]=b[i];
      if(d[r][m]>d[i][m])
         r=i;
   }
   for(j=0;j+1<m;++j)
      d[n][j]=c[j];
   d[n+1][m-1]=-1;
   while(true)
   {
      if(r<n)
      {
         std::swap(ix[s],ix[r+m]);
         d[r][s]=1./d[r][s];
         for(j=0;j<=m;++j)
            if(j!=s)
               d[r][j]*=-d[r][s];
         for(i=0;i<=n+1;++i)
            if(i!=r)
            {
               for(j=0;j<=m;++j)
                  if(j!=s)
                     d[i][j]+=d[r][j]*d[i][s];
               d[i][s]*=d[r][s];
            }
      }
      r=-1;
      s=-1;
      for(j=0;j<m;++j)
         if((s<0 || ix[s]>ix[j]) && (d[n+1][j]>eps || (d[n+1][j]>-
               eps && d[n][j]>eps)))
            s=j;
      if(s<0)
         break;
      for(i=0;i<n;++i)
         if(d[i][s]<-eps && (r<0 || (D=(d[r][m]/d[r][s]-d[i][m]/d[
               i][s]))<-eps || (D<eps && ix[r+m]>ix[i+m])))
            r=i;
      if(r<0)
         return false;
   }
   if(d[n+1][m]<-eps)
      return false;
   for(i=m;i<n+m;++i)
```

```
      if(ix[i]+1<m)
         x[ix[i]]=d[i-m][m]; // answer
   ans=d[n][m]; // maxium value
   return true;
}

int main()
{
   while(scanf("%d %d",&m,&n)!=EOF)
   {
      for(i=0;i<m;++i)
         scanf("%lf",c+i); // max{ sum{c[i]*x[i]} }
      for(i=0;i<n;++i)
      {
         for(j=0;j<m;++j)
            scanf("%lf",a[i]+j); // sum{ a[i]*x[i] } <= b
         scanf("%lf",b+i);
         b[i]*=n;
      }
      simplex();
      printf("Nasa can spend %.0lf taka.\n",ceil(ans));
   }
   return 0;
}
```

6.9 Lucas' theorem(2)

```
#include<cstdio>
#include<cstring>
#include<iostream>

int mod;
long long num[100000];
int ni[100],mi[100];
int len;

void init(int p)
{
   mod=p;
   num[0]=1;
   for (int i=1; i<p; i++)
      num[i]=i*num[i-1]%p;
}

void get(int n,int ni[],int p)
{
   for (int i = 0; i < 100; i++)
      ni[i] = 0;
   int tlen = 0;
   while (n != 0)
   {
      ni[tlen++] = n%p;
      n /= p;
   }
   len = tlen;
}

long long power(long long x,long long y)
{
   long long ret=1;
   for (long long a=x%mod; y; y>>=1,a=a*a%mod)
      if (y&1)
         ret=ret*a%mod;
   return ret;
}

long long getInv(long long x)//`mod`
{
   return power(x,mod-2);
}

long long calc(int n,int m,int p)//C(n,m)%p
{
   init(p);
   long long ans=1;
   for (; n && m && ans; n/=p,m/=p)
   {
      if (n%p>=m%p)
         ans = ans*num[n%p]%p *getInv(num[m%p]%p)%p *getInv(num[n%
               p-m%p])%p;
      else
         ans=0;
   }
   return ans;
}

int main()
{
   int t;
   scanf("%d",&t);
   while (t--)
   {
      int n,m,p;
      scanf("%d%d%d",&n,&m,&p);
      printf("%lld\n",calc(n+m,m,p));
   }
   return 0;
}
```

6.10 Lucas' theorem

```
#include <cstdio>
/*
  Lucas C(n,m)%p
  */
void gcd(int n,int k,int &x,int &y)
{
   if(k)
   {
      gcd(k,n%k,x,y);
      int t=x;
      x=y;
      y=t-(n/k)*y;
      return;
   }
   x=1;
   y=0;
}

int CmodP(int n,int k,int p)
{
   if(k>n)
      return 0;
   int a,b,flag=0,x,y;
   a=b=1;
   for(int i=1;i<=k;i++)
   {
      x=n-i+1;
      y=i;
      while(x%p==0)
      {
         x/=p;
         ++flag;
      }
      while(y%p==0)
      {
         y/=p;
         --flag;
      }
      x%=p;
      y%=p;

      a*=x;
      b*=y;

      b%=p;
      a%=p;
   }
   if(flag)
      return 0;
   gcd(b,p,x,y);
   if(x<0)
      x+=p;
   a*=x;
   a%=p;
   return a;
}

//Lucas C(n,m) % p ,p
long long Lucas(long long n, long long m, long long p)
{
   long long ans=1;
   while(m && n && ans)
   {
      ans*=(CmodP(n%p,m%p,p));
      ans=ans%p;
      n=n/p;
      m=m/p;
   }
   return ans;
}
int main()
{
   long long n,k,p,ans;
   int cas=0;
   while(scanf("%I64d%I64d%I64d",&n,&k,&p)!=EOF)
   {
      if(k>n-k)
         k=n-k;
      ans=Lucas(n+1,k,p)+n-k;
      printf("Case #%d: %I64d\n",++cas,ans%p);
   }
   return 0;
}
```

6.11 Matrix

```
struct Matrix
{
   const int N(52);
   int a[N][N];
   inline Matrix operator*(const Matrix &b)const
   {
      static Matrixres;
      static int i,j,k;
      for(i=0;i<N;++i)
         for(j=0;j<N;++j)
         {
            res.a[i][j]=0;
            for(k=0;k<N;++k)
               res.a[i][j]+=a[i][k]*b.a[k][j];
         }
      return res;
   }
   inline Matrix operator^(int y)const
   {
      static Matrix res,x;
      static int i,j;
      for(i=0;i<N;++i)
      {
         for(j=0;j<N;++j)
         {
            res.a[i][j]=0;
            x.a[i][j]=a[i][j];
         }
         res.a[i][i]=1;
      }
      for(;y;y>>=1,x=x*x)
         if(y&1)
            res=res*x;
      return res;
   }
};
```

```

Fibonacci Matrix
[1 1]
[1 0]
```

6.12 Miller-Rabin Algorithm

```
inline unsigned long long multi_mod(const unsigned long long &a,unsigned long long b,const unsigned long long &n)
{
   unsigned long long exp(a%n),tmp(0);
   while(b)
   {
      if(b&1)
      {
         tmp+=exp;
         if(tmp>n)
            tmp-=n;
      }
      exp<<=1;
      if(exp>n)
         exp-=n;
      b>>=1;
   }
   return tmp;
}

inline unsigned long long exp_mod(unsigned long long a,unsigned long long b,const unsigned long long &c)
{
   unsigned long long tmp(1);
   while(b)
   {
      if(b&1)
         tmp=multi_mod(tmp,a,c);
      a=multi_mod(a,a,c);
      b>>=1;
   }
   return tmp;
}

inline bool miller_rabbin(const unsigned long long &n,short T)
{
   if(n==2)
      return true;
   if(n<2 || !(n&1))
      return false;
   unsigned long long a,u(n-1),x,y;
   short t(0),i;
   while(!(u&1))
   {
      ++t;
      u>>=1;
   }
   while(T--)
   {
      a=rand()%(n-1)+1;
      x=exp_mod(a,u,n);
      for(i=0;i<t;++i)
      {
         y=multi_mod(x,x,n);
         if(y==1 && x!=1 && x!=n-1)
            return false;
         x=y;
      }
      if(y!=1)
         return false;
   }
   return true;
}
```

6.13 Multiset

```
Permutation:
MultiSet S={1 m,4 s,4 i,2 p}
P(S)=(1+4+4+2)!/1!/4!/4!/2!

Combination:
MultiSet S={ a1, a2,... ak}
C(S,r)=(r+k-1)!/r!/(k-1)!=C(r,r+k-1)

if(r>min{count(element[i])})
   you have to resolve this problem with inclusion-exclusion principle.

MS T={3 a,4 b,5 c}
MS T*={ a, b, c}
A1={C(T*,10)|count(a)>3} // C(6,8)
A2={C(T*,10)|count(b)>4} // C(5,7)
A3={C(T*,10)|count(c)>5} // C(4,6)

C(T,10)=C(T*,10)-(|A1|+|A2|+|A3|)+(|A1 A2|+|A1 A3|+|A2 A3|)-|A1 A2 A3|
      C(10,12) C(1,3) C(0,2) 0 0
ans=6
```

6.14 Pell's equation

```
find the (x,y)pair that x^2-n*y^2=1
these is not solution if and only if n is a square number.

solution:
simply brute-force search the integer y, get (x1,y1). ( O(sqrt(n)))
or we can enumerate the continued fraction of sqrt(n), as (x/y), it will be much more faster

other solution pairs' matrix:
[ x1 n*y1 ]
[ y1 x1 ]
k-th solution is pow({matrix},k)
```

6.15 Pollard's rho algorithm

```
#include<cstdio>
#include<cstdlib>
#include<list>

short T;
unsigned long long a;
std::list<unsigned long long>fac;

inline unsigned long long multi_mod(const unsigned long long &a,unsigned long long b,const unsigned long long &n)
{
   unsigned long long exp(a%n),tmp(0);
   while(b)
   {
      if(b&1)
      {
         tmp+=exp;
         if(tmp>n)
            tmp-=n;
      }
      exp<<=1;
      if(exp>n)
         exp-=n;
      b>>=1;
   }
   return tmp;
}

inline unsigned long long exp_mod(unsigned long long a,unsigned long long b,const unsigned long long &c)
{
   unsigned long long tmp(1);
   while(b)
   {
      if(b&1)
         tmp=multi_mod(tmp,a,c);
      a=multi_mod(a,a,c);
      b>>=1;
   }
   return tmp;
}

inline bool miller_rabbin(const unsigned long long &n,short T)
{
   if(n==2)
      return true;
   if(n<2 || !(n&1))
      return false;
   unsigned long long a,u(n-1),x,y;
   short t(0),i;
   while(!(u&1))
   {
      ++t;
      u>>=1;
   }
   while(T--)
   {
      a=rand()%(n-1)+1;
      x=exp_mod(a,u,n);
      for(i=0;i<t;++i)
      {
         y=multi_mod(x,x,n);
         if(y==1 && x!=1 && x!=n-1)
            return false;
         x=y;
      }
      if(y!=1)
         return false;
   }
   return true;
}

unsigned long long gcd(const unsigned long long &a,const unsigned long long &b)
{
   return b?gcd(b,a%b):a;
}

inline unsigned long long pollar_rho(const unsigned long long n,const unsigned long long &c)
{
   unsigned long long x(rand()%(n-1)+1),y,d,i(1),k(2);
   y=x;
   while(true)
   {
      ++i;
      x=(multi_mod(x,x,n)+c)%n;
      d=gcd((x-y+n)%n,n);
      if(d>1 && d<n)
         return d;
      if(x==y)
         return n;
      if(i==k)
      {
         k<<=1;
         y=x;
      }
   }
}

void find(const unsigned long long &n,short c)
{
    if(n==1)
        return;
    if(miller_rabbin(n,6))
    {
        fac.push_back(n);
        return;
    }
    unsigned long long p(n);
    short k(c);
    while(p>=n)
        p=pollar_rho(p,c--);
    find(p,k);
    find(n/p,k);
}

int main()
{
    scanf("%hd",&T);
    while(T--)
    {
        scanf("%llu",&a);
        fac.clear();
        find(a,120);
        if(fac.size()==1)
            puts("Prime");
        else
        {
            fac.sort();
            printf("%llu\n",fac.front());
        }
    }
    return 0;
}
```

6.16 Prime

```
#include<vector>

std::vector<int>prm;
bool flag[MAXX];

int main()
{
    prm.reserve(MAXX); // pi(x)=x/ln(x);
    for(i=2;i<MAXX;++i)
    {
        if(!flag[i])
            prm.push_back(i);
        for(j=0;j<prm.size() && i*prm[j]<MAXX;++j)
        {
            flag[i*prm[j]]=true;
            if(i%pmr[j]==0)
                break;
        }
    }
    return 0;
}
```

6.17 Reduced Residue System

```
Euler's totient function:

nnnn
(1)=111
m,n(mn)=(m)(n)

inline long long phi(int n)
{
    static int i;
    static int re;
    re=n;
    for(i=0;prm[i]*prm[i]<=n;++i)
        if(n%prm[i]==0)
        {
            re-=re/prm[i];
            do
                n/=prm[i];
            while(n%prm[i]==0);
        }
    if(n!=1)
        re-=re/n;
    return re;
}

inline void Euler()
{
    static int i,j;
    phi[1]=1;
    for(i=2;i<MAXX;++i)
        if(!phi[i])
            for(j=i;j<MAXX;j+=i)
            {
                if(!phi[j])
                    phi[j]=j;
                phi[j]=phi[j]/i*(i-1);
            }
}

Multiplicative order:

the multiplicative order of a modulo n is the smallest positive integer k with
    a^k 1 (mod n).

mx,ord(x)(m) (aka. Euler's totient theorem)

:
method 1(m)(m)d pow(x,d,m)==1;
method 2
inline long long ord(long long x,long long m)
{
    static long long ans;
    static int i,j;
    ans=phi(m);
    for(i=0;i<fac.size();++i)
        for(j=0;j<fac[i].second && pow(x,ans/fac[i].first,m)==1ll;++j)
            ans/=fac[i].first;
    return ans;
}


Primitive root:

ord(x)==(m)xm
 pow(x,d) {d(m)} pow(x,d)%m==1 dd(m)xm

m= 1,2,4,pow(p,n),2*pow(p,n) {p,n} m //

m((m))


iip[j],pow(i,(m)/p[j])%m1imord(i)==(m)


Carmichael function:

(n) is defined as the smallest positive integer m such that
    pow(a,m)%n==1 { for a!=1 && gcd(a,n)==1 }
(1)x lcm{ord(x)}

if n=pow(p[0],a[0])*pow(p[1],a[1])*...*pow(p[m-1],a[m-1])
    then (n)=lcm((pow(p[0],a[0])),(pow(p[1],a[1])),...,(pow(p[m-1],a[m-1])));

if n=pow(2,a)*pow(p[0],a[0])*pow(p[1],a[1])*...*pow(p[m-1],a[m-1])
    then (n)=lcm(pow(2,c),(pow(p[0],a[0])),(pow(p[1],a[1])),...,(pow(p[m-1],a[m-1])));
    { c=0 if a<2; c=1 if a==2; c=a-2 if a>3; }


Carmichael's theorem:
if gcd(a,n)==1
    then pow(a,(n))%n==1
```

6.18 System of linear congruences

```
// minimal val that for all (m,a) , val%m == a
#include<cstdio>

#define MAXX 11

int T,t;
int m[MAXX],a[MAXX];
int n,i,j,k;
int x,y,c,d;
int lcm;

int exgcd(int a,int b,int &x,int &y)
{
    if(b)
    {
        int re(exgcd(b,a%b,x,y)),tmp(x);
        x=y;
        y=tmp-(a/b)*y;
        return re;
    }
    x=1;
    y=0;
    return a;
}

int main()
{
    scanf("%d",&T);
    for(t=1;t<=T;++t)
    {
        scanf("%d",&n);
        lcm=1;
        for(i=0;i<n;++i)
        {
            scanf("%d",m+i);
            lcm*=m[i]/exgcd(lcm,m[i],x,y);
        }
        for(i=0;i<n;++i)
            scanf("%d",a+i);
        for(i=1;i<n;++i)
        {
            c=a[i]-a[0];
            d=exgcd(m[0],m[i],x,y);
            if(c%d)
                break;
            y=m[i]/d;
            c/=d;
            x=(x*c%y+y)%y;
            a[0]+=m[0]*x;
            m[0]*=y;
        }
        printf("Case %d: %d\n",t,i<n?-1:(a[0]?a[0]:lcm));
    }
    return 0;
}
```

7 others

7.1 .vimrc

```
set number
set history=1000000
set autoindent
set smartindent
set tabstop=4
set shiftwidth=4
set expandtab
set showmatch

set nocp
filetype plugin indent on

filetype on
syntax on
```

7.2 bigint

```cpp
// header files
#include <cstdio>
#include <string>
#include <algorithm>
#include <iostream>

struct Bigint
{
   // representations and structures
   std::string a; // to store the digits
   int sign; // sign = -1 for negative numbers, sign = 1 otherwise
   // constructors
   Bigint() {} // default constructor
   Bigint( std::string b ) { (*this) = b; } // constructor for std::string
   // some helpful methods
   int size() // returns number of digits
   {
      return a.size();
   }
   Bigint inverseSign() // changes the sign
   {
      sign *= -1;
      return (*this);
   }
   Bigint normalize( int newSign ) // removes leading 0, fixes sign
   {
      for( int i = a.size() - 1; i > 0 && a[i] == '0'; i-- )
         a.erase(a.begin() + i);
      sign = ( a.size() == 1 && a[0] == '0' ) ? 1 : newSign;
      return (*this);
   }
   // assignment operator
   void operator = ( std::string b ) // assigns a std::string to Bigint
   {
      a = b[0] == '-' ? b.substr(1) : b;
      reverse( a.begin(), a.end() );
      this->normalize( b[0] == '-' ? -1 : 1 );
   }
   // conditional operators
   bool operator < ( const Bigint &b ) const // less than operator
   {
      if( sign != b.sign )
         return sign < b.sign;
      if( a.size() != b.a.size() )
         return sign == 1 ? a.size() < b.a.size() : a.size() > b.a.size();
      for( int i = a.size() - 1; i >= 0; i-- )
         if( a[i] != b.a[i] )
            return sign == 1 ? a[i] < b.a[i] : a[i] > b.a[i];
      return false;
   }
   bool operator == ( const Bigint &b ) const // operator for equality
   {
      return a == b.a && sign == b.sign;
   }

   // mathematical operators
   Bigint operator + ( Bigint b ) // addition operator overloading
   {
      if( sign != b.sign )
         return (*this) - b.inverseSign();
      Bigint c;
      for(int i = 0, carry = 0; i<a.size() || i<b.size() || carry; i++ )
      {
         carry+=(i<a.size() ? a[i]-48 : 0)+(i<b.a.size() ? b.a[i]-48 : 0);
         c.a += (carry % 10 + 48);
         carry /= 10;
      }
      return c.normalize(sign);
   }

   Bigint operator - ( Bigint b ) // subtraction operator overloading
   {
      if( sign != b.sign )
         return (*this) + b.inverseSign();
      int s = sign; sign = b.sign = 1;
      if( (*this) < b )
         return ((b - (*this)).inverseSign()).normalize(-s);
      Bigint c;
      for( int i = 0, borrow = 0; i < a.size(); i++ )
      {
         borrow = a[i] - borrow - (i < b.size() ? b.a[i] : 48);
         c.a += borrow >= 0 ? borrow + 48 : borrow + 58;
         borrow = borrow >= 0 ? 0 : 1;
      }
      return c.normalize(s);
   }
   Bigint operator * ( Bigint b ) // multiplication operator overloading
   {
      Bigint c("0");
      for( int i = 0, k = a[i] - 48; i < a.size(); i++, k = a[i] - 48 )
      {
         while(k--)
            c = c + b; // ith digit is k, so, we add k times
         b.a.insert(b.a.begin(), '0'); // multiplied by 10
      }
      return c.normalize(sign * b.sign);
   }
   Bigint operator / ( Bigint b ) // division operator overloading
   {
      if( b.size() == 1 && b.a[0] == '0' )
         b.a[0] /= ( b.a[0] - 48 );
      Bigint c("0"), d;
      for( int j = 0; j < a.size(); j++ )
         d.a += "0";
      int dSign = sign * b.sign;
      b.sign = 1;
      for( int i = a.size() - 1; i >= 0; i-- )
      {
         c.a.insert( c.a.begin(), '0');
         c = c + a.substr( i, 1 );
         while( !( c < b ) )
         {
            c = c - b;
            d.a[i]++;
         }
      }
      return d.normalize(dSign);
   }
   Bigint operator % ( Bigint b ) // modulo operator overloading
   {
      if( b.size() == 1 && b.a[0] == '0' )
         b.a[0] /= ( b.a[0] - 48 );
      Bigint c("0");
      b.sign = 1;
      for( int i = a.size() - 1; i >= 0; i-- )
      {
         c.a.insert( c.a.begin(), '0');
         c = c + a.substr( i, 1 );
         while( !( c < b ) )
            c = c - b;
      }
      return c.normalize(sign);
   }

   // output method
   void print()
   {
      if( sign == -1 )
         putchar('-');
      for( int i = a.size() - 1; i >= 0; i-- )
         putchar(a[i]);
   }
};



int main()
{
   Bigint a, b, c; // declared some Bigint variables
   /////////////////////////
   // taking Bigint input //
   /////////////////////////

   std::string input; // std::string to take input
   std::cin >> input; // take the Big integer as std::string
   a = input; // assign the std::string to Bigint a

   std::cin >> input; // take the Big integer as std::string
   b = input; // assign the std::string to Bigint b

   //////////////////////////////////
   // Using mathematical operators //
   //////////////////////////////////

   c = a + b; // adding a and b
   c.print(); // printing the Bigint
   puts(""); // newline

   c = a - b; // subtracting b from a
   c.print(); // printing the Bigint
   puts(""); // newline

   c = a * b; // multiplying a and b
   c.print(); // printing the Bigint
   puts(""); // newline

   c = a / b; // dividing a by b
   c.print(); // printing the Bigint
   puts(""); // newline

   c = a % b; // a modulo b
   c.print(); // printing the Bigint
   puts(""); // newline

   /////////////////////////////////
   // Using conditional operators //
   /////////////////////////////////

   if( a == b )
      puts("equal"); // checking equality
   else
      puts("not equal");

   if( a < b )
      puts("a is smaller than b"); // checking less than operator

   return 0;
}
```

7.3 Binary Search

```
//[0,n)
inline int go(int A[],int n,int x) // return the least i that make A[i]==x;
{
   static int l,r,mid,re;
   l=0;
   r=n-1;
   re=-1;
   while(l<=r)
   {
      mid=l+r>>1;
      if(A[mid]<x)
         l=mid+1;
      else
      {
         r=mid-1;
         if(A[mid]==x)
            re=mid;
      }
   }
   return re;
}

inline int go(int A[],int n,int x) // return the largest i that make A[i]==x;
{
   static int l,r,mid,re;
   l=0;
   r=n-1;
   re=-1;
   while(l<=r)
   {
      mid=l+r>>1;
      if(A[mid]<=x)
      {
         l=mid+1;
         if(A[mid]==x)
            re=mid;
      }
      else
         r=mid-1;
   }
   return re;
}

inline int go(int A[],int n,int x) // retrun the largest i that make A[i]<x;
{
   static int l,r,mid,re;
   l=0;
   r=n-1;
   re=-1;
   while(l<=r)
   {
      mid=l+r>>1;
      if(A[mid]<x)
      {
         l=mid+1;
         re=mid;
      }
      else
         r=mid-1;
   }
   return re;
}

inline int go(int A[],int n,int x)// return the largest i that make A[i]<=x;
{
   static int l,r,mid,re;
   l=0;
   r=n-1;
   re=-1;
   while(l<=r)
   {
      mid=l+r>>1;
      if(A[mid]<=x)
      {
         l=mid+1;
         re=mid;
      }
      else
         r=mid-1;
   }
   return re;
}

inline int go(int A[],int n,int x)// return the least i that make A[i]>x;
{
   static int l,r,mid,re;
   l=0;
   r=n-1;
   re=-1;
   while(l<=r)
   {
      mid=l+r>>1;
      if(A[mid]<=x)
         l=mid+1;
      else
      {
         r=mid-1;
         re=mid;
      }
   }
   return re;
}

inline int go(int A[],int n,int x)// upper_bound();
{
   static int l,r,mid;
   l=0;
   r=n-1;
   while(l<r)
   {
      mid=l+r>>1;
      if(A[mid]<=x)
         l=mid+1;
      else
         r=mid;
   }
   return r;
}

inline int go(int A[],int n,int x)// lower_bound();
{
   static int l,r,mid,;
   l=0;
   r=n-1;
   while(l<r)
   {
      mid=l+r>>1;
      if(A[mid]<x)
         l=mid+1;
      else
         r=mid;
   }
   return r;
}
```

7.4 java

```
//Scanner

Scanner in=new Scanner(new FileReader("asdf"));
PrintWriter pw=new PrintWriter(new Filewriter("out"));
boolean in.hasNext();
String in.next();
BigDecimal in.nextBigDecimal();
BigInteger in.nextBigInteger();
BigInteger in.nextBigInteger(int radix);
double in.nextDouble();
int in.nextInt();
int in.nextInt(int radix);
String in.nextLine();
long in.nextLong();
long in.nextLong(int radix);
short in.nextShort();
short in.nextShort(int radix);
int in.radix(); //Returns this scanner's default radix.
Scanner in.useRadix(int radix);// Sets this scanner's default radix to the specified radix.
void in.close();//Closes this scanner.

//String

char str.charAt(int index);
int str.compareTo(String anotherString); // <0 if less. ==0 if equal. >0 if greater.
int str.compareToIgnoreCase(String str);
String str.concat(String str);
boolean str.contains(CharSequence s);
boolean str.endsWith(String suffix);
boolean str.startsWith(String preffix);
boolean str.startsWith(String preffix,int toffset);
int str.hashCode();
int str.indexOf(int ch);
int str.indexOf(int ch,int fromIndex);
int str.indexOf(String str);
int str.indexOf(String str,int fromIndex);
int str.lastIndexOf(int ch);
int str.lastIndexOf(int ch,int fromIndex);
//(ry
int str.length();
String str.substring(int beginIndex);
String str.substring(int beginIndex,int endIndex);
String str.toLowerCase();
String str.toUpperCase();
String str.trim();// Returns a copy of the string, with leading and trailing whitespace omitted.

//StringBuilder
StringBuilder str.insert(int offset,...);
StringBuilder str.reverse();
void str.setCharAt(int index,int ch);

//BigInteger
compareTo(); equals(); doubleValue(); longValue(); hashCode(); toString(); toString(int radix); max(); min(); mod(); modPow(BigInteger exp,BigInteger m); nextProbablePrime(); pow();
andNot(); and(); xor(); not(); or(); getLowestSetBit(); bitCount(); bitLength(); setBig(int n); shiftLeft(int n); shiftRight(int n);
add(); divide(); divideAndRemainder(); remainder(); multiply(); subtract(); gcd(); abs(); signum(); negate();

//BigDecimal
movePointLeft(); movePointRight(); precision(); stripTrailingZeros(); toBigInteger(); toPlainString();


//sort
class pii implements Comparable
{
   public int a,b;
   public int compareTo(Object i)
   {
      pii c=(pii)i;
      return a==c.a?c.b-b:c.a-a;
   }
}

class Main
{
   public static void main(String[] args)
   {
      pii[] the=new pii[2];
      the[0]=new pii();
      the[1]=new pii();
      the[0].a=1;
      the[0].b=1;
      the[1].a=1;
      the[1].b=2;
      Arrays.sort(the);
      for(int i=0;i<2;++i)
         System.out.printf("%d %d\n",the[i].a,the[i].b);
   }
}
```

7.5 others

```
god damn it windows:
#pragma comment(linker, "/STACK:16777216")
#pragma comment(linker,"/STACK:102400000,102400000")


chmod +x [filename]

while true; do
./gen > input
./sol < input > output.sol
./bf < input > output.bf

diff output.sol output.bf
if[ $? -ne 0];then break fi
done


1
2calm_down();calm_down();calm_down();
3
4
5/
6//hash//
6.1 or
6.2
```

8 search

8.1 dlx

```

011


011
```

8.2 dlx - exact cover

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<vector>

#define N 256
#define MAXN N*22
#define MAXM N*5
#define inf 0x3f3f3f3f
const int MAXX(MAXN*MAXM);

bool mat[MAXN][MAXM];

int u[MAXX],d[MAXX],l[MAXX],r[MAXX],ch[MAXX],rh[MAXX];
int sz[MAXM];
std::vector<int>ans(MAXX);
int hd,cnt;

inline int node(int up,int down,int left,int right)
{
   u[cnt]=up;
   d[cnt]=down;
   l[cnt]=left;
   r[cnt]=right;
   u[down]=d[up]=l[right]=r[left]=cnt;
   return cnt++;
}

inline void init(int n,int m)
{
   cnt=0;
   hd=node(0,0,0,0);
   static int i,j,k,r;
   for(j=1;j<=m;++j)
   {
      ch[j]=node(cnt,cnt,l[hd],hd);
      sz[j]=0;
   }
   for(i=1;i<=n;++i)
   {
      r=-1;
      for(j=1;j<=m;++j)
         if(mat[i][j])
         {
            if(r==-1)
            {
               r=node(u[ch[j]],ch[j],cnt,cnt);
               rh[r]=i;
               ch[r]=ch[j];
            }
            else
            {
               k=node(u[ch[j]],ch[j],l[r],r);
               rh[k]=i;
               ch[k]=ch[j];
            }
            ++sz[j];
         }
   }
}

inline void rm(int c)
{
   l[r[c]]=l[c];
   r[l[c]]=r[c];
   static int i,j;
   for(i=d[c];i!=c;i=d[i])
      for(j=r[i];j!=i;j=r[j])
      {
         u[d[j]]=u[j];
         d[u[j]]=d[j];
         --sz[ch[j]];
      }
}

inline void add(int c)
{
   static int i,j;
   for(i=u[c];i!=c;i=u[i])
      for(j=l[i];j!=i;j=l[j])
      {
         ++sz[ch[j]];
         u[d[j]]=d[u[j]]=j;
      }
   l[r[c]]=r[l[c]]=c;
}

bool dlx(int k)
{
   if(hd==r[hd])
   {
      ans.resize(k);
      return true;
   }
   int s=inf,c;
   int i,j;
   for(i=r[hd];i!=hd;i=r[i])
      if(sz[i]<s)
      {
         s=sz[i];
         c=i;
      }
   rm(c);
   for(i=d[c];i!=c;i=d[i])
   {
      ans[k]=rh[i];
      for(j=r[i];j!=i;j=r[j])
         rm(ch[j]);
      if(dlx(k+1))
         return true;
      for(j=l[i];j!=i;j=l[j])
         add(ch[j]);
   }
   add(c);
   return false;
}

#include <cstdio>
#include <cstring>

#define N 1024
#define M 1024*110
using namespace std;

int l[M], r[M], d[M], u[M], col[M], row[M], h[M], res[N], cntcol[N];
int dcnt = 0;
//
inline void addnode(int &x)
{
   ++x;
   r[x] = l[x] = u[x] = d[x] = x;
}
//xrowx
inline void insert_row(int rowx, int x)
{
   r[l[rowx]] = x;
   l[x] = l[rowx];
   r[x] = rowx;
   l[rowx] = x;
}
//xcolx
inline void insert_col(int colx, int x)
{
   d[u[colx]] = x;
   u[x] = u[colx];
   d[x] = colx;
   u[colx] = x;
}
//
inline void dlx_init(int cols)
{
   memset(h, -1, sizeof(h));
   memset(cntcol, 0, sizeof(cntcol));
   dcnt = -1;
   addnode(dcnt);
   for (int i = 1; i <= cols; ++i)
   {
      addnode(dcnt);
      insert_row(0, dcnt);
   }
}
//
inline void remove(int c)
{
   l[r[c]] = l[c];
   r[l[c]] = r[c];
   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];
         cntcol[col[j]]--;
      }
}
//
inline void resume(int c)
{
   for (int i = u[c]; i != c; i = u[i])
      for (int j = l[i]; j != i; j = l[j])
      {
         u[d[j]] = j;
         d[u[j]] = j;
         cntcol[col[j]]++;
      }
   l[r[c]] = c;
   r[l[c]] = c;
}
//
bool DLX(int deep)
{
   if (r[0] == 0)
   {
//Do anything you want to do here
      printf("%d", deep);
      for (int i = 0; i < deep; ++i) printf(" %d", res[i]);
      puts("");
      return true;
   }
   int min = INT_MAX, tempc;
   for (int i = r[0]; i != 0; i = r[i])
      if (cntcol[i] < min)
      {
         min = cntcol[i];
         tempc = i;
      }
   remove(tempc);
   for (int i = d[tempc]; i != tempc; i = d[i])
   {
      res[deep] = row[i];
      for (int j = r[i]; j != i; j = r[j]) remove(col[j]);
      if (DLX(deep + 1)) return true;
      for (int j = l[i]; j != i; j = l[j]) resume(col[j]);
   }
   resume(tempc);
   return false;
}
//"1"
inline void insert_node(int x, int y)
{
   cntcol[y]++;
   addnode(dcnt);
   row[dcnt] = x;
   col[dcnt] = y;
   insert_col(y, dcnt);
   if (h[x] == -1) h[x] = dcnt;
   else insert_row(h[x], dcnt);
}
int main()
{
   int n, m;
   while (~scanf("%d%d", &n, &m))
   {
      dlx_init(m);
      for (int i = 1; i <= n; ++i)
      {
         int k, x;
         scanf("%d", &k);
         while (k--)
         {
            scanf("%d", &x);
            insert_node(i, x);
         }
      }
      if (!DLX(0))
         puts("NO");
   }
   return 0;
}
```

8.3 dlx - repeat cover

```
#include<cstdio>
#include<cstring>
#include<algorithm>

#define MAXN 110
#define MAXM 1000000
#define INF 0x7FFFFFFF

using namespace std;

int G[MAXN][MAXN];
int L[MAXM], R[MAXM], U[MAXM], D[MAXM];
int size, ans, S[MAXM], H[MAXM], C[MAXM];
bool vis[MAXN * 100];
void Link(int r, int c)
{
   U[size] = c;
   D[size] = D[c];
   U[D[c]] = size;
   D[c] = size;
   if (H[r] < 0)
      H[r] = L[size] = R[size] = size;
   else
   {
      L[size] = H[r];
      R[size] = R[H[r]];
      L[R[H[r]]] = size;
      R[H[r]] = size;
   }
   S[c]++;
   C[size++] = c;
}
void Remove(int c)
{
   int i;
   for (i = D[c]; i != c; i = D[i])
   {
      L[R[i]] = L[i];
      R[L[i]] = R[i];
   }
}
void Resume(int c)
{
   int i;
   for (i = D[c]; i != c; i = D[i])
      L[R[i]] = R[L[i]] = i;
}
int A()
{
   int i, j, k, res;
   memset(vis, false, sizeof(vis));
   for (res = 0, i = R[0]; i; i = R[i])
   {
      if (!vis[i])
      {
         res++;
         for (j = D[i]; j != i; j = D[j])
         {
            for (k = R[j]; k != j; k = R[k])
               vis[C[k]] = true;
         }
      }
   }
   return res;
}
void Dance(int now)
{
   if (R[0] == 0)
      ans = min(ans, now);
   else if (now + A() < ans)
   {
      int i, j, temp, c;
      for (temp = INF,i = R[0]; i; i = R[i])
      {
         if (temp > S[i])
         {
            temp = S[i];
            c = i;
         }
      }
      for (i = D[c]; i != c; i = D[i])
      {
         Remove(i);
         for (j = R[i]; j != i; j = R[j])
            Remove(j);
         Dance(now + 1);
         for (j = L[i]; j != i; j = L[j])
            Resume(j);
         Resume(i);
      }
   }
}
void Init(int m)
{
   int i;
   for (i = 0; i <= m; i++)
   {
      R[i] = i + 1;
      L[i + 1] = i;
      U[i] = D[i] = i;
      S[i] = 0;
   }
   R[m] = 0;
   size = m + 1;
}
```

8.4 fibonacci knapsack

```
#include<stdio.h>
#include<stdlib.h>
#include<algorithm>

#define MAXX 71

struct mono
{
   long long weig,cost;
}goods[MAXX];

short n,T,t,i;
long long carry,sumw,sumc;
long long ans,las[MAXX];

int com(const void *n,const void *m)
{
   struct mono *a=(struct mono *)n,*b=(struct mono *)m;
   if(a->weig!=b->weig)
      return a->weig-b->weig;
   else
      return b->cost-a->cost;
}

bool comp(const struct mono a,const struct mono b)
{
   if(a.weig!=b.weig)
      return a.weig<b.weig;
   else
      return b.cost<a.cost;
}

void dfs(short i,long long cost_n,long long carry_n,short last)
{
   if(ans<cost_n)
      ans=cost_n;
   if(i==n || goods[i].weig>carry_n || cost_n+las[i]<=ans)
      return;
   if(last || (goods[i].weig!=goods[i-1].weig && goods[i].cost>goods[i-1].cost))
      dfs(i+1,cost_n+goods[i].cost,carry_n-goods[i].weig,1);
   dfs(i+1,cost_n,carry_n,0);
}

int main()
{
   // freopen("asdf","r",stdin);
   scanf("%hd",&T);
   for(t=1;t<=T;++t)
   {
      scanf("%hd%lld",&n,&carry);
      sumw=0;
      sumc=0;
      ans=0;
      for(i=0;i<n;++i)
      {
         scanf("%lld%lld",&goods[i].weig,&goods[i].cost);
         sumw+=goods[i].weig;
         sumc+=goods[i].cost;
      }
      if(sumw<=carry)
      {
         printf("Case %hd: %lld\n",t,sumc);
         continue;
      }
// qsort(goods,n,sizeof(struct mono),com);
      std::sort(goods,goods+n,comp);
      for(i=0;i<n;++i)
      {
// printf("%lld %lld\n",goods[i].weig,goods[i].cost);
         las[i]=sumc;
         sumc-=goods[i].cost;
      }
      dfs(0,0,carry,1);
      printf("Case %hd: %lld\n",t,ans);
   }
   return 0;
}
```

9 string

9.1 Aho-Corasick Algorithm

```
//trie graph
#include<cstring>
#include<queue>

#define MAX 1000111
#define N 26

int nxt[MAX][N],fal[MAX],cnt;
bool ed[MAX];
char buf[MAX];

inline void init(int a)
{
   memset(nxt[a],0,sizeof(nxt[0]));
   fal[a]=0;
   ed[a]=false;
}

inline void insert()
{
   static int i,p;
   for(i=p=0;buf[i];++i)
   {
      if(!nxt[p][map[buf[i]]])
         init(nxt[p][map[buf[i]]]=++cnt);
      p=nxt[p][map[buf[i]]];
   }
   ed[p]=true;
}

inline void make()
{
   static std::queue<int>q;
   int i,now,p;
   q.push(0);
   while(!q.empty())
   {
      now=q.front();
      q.pop();
      for(i=0;i<N;++i)
         if(nxt[now][i])
         {
            q.push(p=nxt[now][i]);
            if(now)
               fal[p]=nxt[fal[now]][i];
            ed[p]|=ed[fal[p]];
         }
         else
            nxt[now][i]=nxt[fal[now]][i]; // trienxt
   }
}

// normal version

#define N 128

char buf[MAXX];
int cnt[1111];

struct node
{
   node *fal,*nxt[N];
   int idx;
   node() { memset(this,0,sizeof node); }
}*rt;
std::queue<node*>Q;

void free(node *p)
{
   for(int i(0);i<N;++i)
      if(p->nxt[i])
         free(p->nxt[i]);
   delete p;
}

inline void add(char *s,int idx)
{
   static node *p;
   for(p=rt;*s;++s)
   {
      if(!p->nxt[*s])
         p->nxt[*s]=new node();
      p=p->nxt[*s];
   }
   p->idx=idx;
}

inline void make()
{
   Q.push(rt);
   static node *p,*q;
   static int i;
   while(!Q.empty())
   {
      p=Q.front();
      Q.pop();
      for(i=0;i<N;++i)
         if(p->nxt[i])
         {
            q=p->fal;
            while(q)
            {
               if(q->nxt[i])
               {
                  p->nxt[i]->fal=q->nxt[i];
                  break;
               }
               q=q->fal;
            }
            if(!q)
               p->nxt[i]->fal=rt;
            Q.push(p->nxt[i]);
         }
   }
}

inline void match(const char *s)
{
   static node *p,*q;
   for(p=rt;*s;++s)
   {
      while(p!=rt && !p->nxt[*s])
         p=p->fal;
      p=p->nxt[*s];
      if(!p)
         p=rt;
      for(q=p;q!=rt && q->idx;q=q->fal) // why q->idx ? looks like not necessary at all, I delete it in an other solution
         ++cnt[q->idx];
   }
}

//dfsfal
//BIT
```

9.2 Gusfield's Z Algorithm

```
inline void make(int *z,char *buf)
{
   int i,j,l,r;
   l=0;
   r=1;
   z[0]=strlen(buf);
   for(i=1;i<z[0];++i)
      if(r<=i || z[i-l]>=r-i)
      {
         j=std::max(i,r);
         while(j<z[0] && buf[j]==buf[j-i])
            ++j;
         z[i]=j-i;
         if(i<j)
         {
            l=i;
            r=j;
         }
      }
      else
         z[i]=z[i-l];
}

for(i=1;i<len && i+z[i]<len;++i); //i=
```

9.3 Manacher's Algorithm

```
#include<cstdio>
#include<vector>

#define MAXX 1111

std::vector<char>str;
char buf[MAXX];
int z[MAXX<<1];
int i,j,l,r;
int ii,n,c;

inline int match(const int &a,const int &b)
{
   int i(0);
   while(a-i>=0 && b+i<str.size() && str[a-i]==str[b+i])//i1
      ++i;
   return i;
}

int main()
{
   gets(buf);
   str.reserve(MAXX<<1);
   for(i=0;buf[i];++i)
   {
      str.push_back(’$’);
      str.push_back(buf[i]);
   }
   str.push_back(’$’);

   z[0]=1;
   c=l=r=0;
   for(i=1;i<str.size();++i)
   {
      ii=(l<<1)-i;
      n=r+1-i;

      if(i>r)
      {
         z[i]=match(i,i);
         l=i;
         r=i+z[i]-1;
      }
      else
         if(z[ii]==n)
         {
            z[i]=n+match(i-n,i+n);
            l=i;
            r=i+z[i]-1;
         }
         else
            z[i]=std::min(z[ii],n);
      if(z[i]>z[c])
         c=i;
   }

   for(i=c-z[c]+2,n=c+z[c];i<n;i+=2)
      putchar(str[i]);
   puts("");
   return 0;
}

//package:

inline int match(const int a,const int b,const std::vector<int> &
      str)
{
   static int i;
   i=0;
   while(a-i>=0 && b+i<str.size() && str[a-i]==str[b+i])
      ++i;
   return i;
}

inline void go(int *z,const std::vector<int> &str)
{
   static int c,l,r,i,ii,n;
   z[0]=1;
   c=l=r=0;
   for(i=1;i<str.size();++i)
   {
      ii=(l<<1)-i;
      n=r+1-i;

      if(i>r)
      {
         z[i]=match(i,i,str);
         l=i;
         r=i+z[i]-1;
      }
      else
         if(z[ii]==n)
         {
            z[i]=n+match(i-n,i+n,str);
            l=i;
            r=i+z[i]-1;
         }
         else
            z[i]=std::min(z[ii],n);
      if(z[i]>z[c])
         c=i;
   }
}

inline bool check(int *z,int a,int b) //[a,b]
{
   a=a*2-1;
   b=b*2-1;
   int m=(a+b)/2;
   return z[m]>=b-m+1;
}
```

9.4 Morris-Pratt Algorithm

```
inline void make(char *buf,int *fal)
{
   static int i,j;
   fal[0]=-1;
   for(i=1,j=-1;buf[i];++i)
   {
      while(j>=0 && buf[j+1]!=buf[i])
         j=fal[j];
      if(buf[j+1]==buf[i])
         ++j;
      fal[i]=j;
   }

}

inline int match(char *p,char *t,int* fal)
{
   static int i,j,re;
   re=0;
   for(i=0,j=-1;t[i];++i)
   {
      while(j>=0 && p[j+1]!=t[i])
         j=fal[j];
      if(p[j+1]==t[i])
         ++j;
      if(!p[j+1])
      {
         ++re;
         j=fal[j];
      }
   }
   return re;
}
```

9.5 smallest representation

```
int min(char a[],int len)
{
   int i = 0,j = 1,k = 0;
   while (i < len && j < len && k < len)
   {
      int cmp = a[(j+k)%len]-a[(i+k)%len];
      if (cmp == 0)
         k++;
      else
      {
         if (cmp > 0)
            j += k+1;
         else
            i += k+1;
         if (i == j) j++;
         k = 0;
      }
   }
   return std::min(i,j);
}
```

9.6 Suffix Array - DC3 Algorithm

```
#include<cstdio>
#include<cstring>
#include<algorithm>

#define MAXX 1111
#define F(x) ((x)/3+((x)%3==1?0:tb))
#define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)

int wa[MAXX],wb[MAXX],wv[MAXX],ws[MAXX];

inline bool c0(const int *str,const int &a,const int &b)
{
   return str[a]==str[b] && str[a+1]==str[b+1] && str[a+2]==str[b
         +2];
}

inline bool c12(const int *str,const int &k,const int &a,const int
      &b)
{
   if(k==2)
      return str[a]<str[b] || str[a]==str[b] && c12(str,1,a+1,b+1)
            ;
   else
      return str[a]<str[b] || str[a]==str[b] && wv[a+1]<wv[b+1];
}

inline void sort(int *str,int *a,int *b,const int &n,const int &m)
{
   memset(ws,0,sizeof(ws));
   int i;
   for(i=0;i<n;++i)
      ++ws[wv[i]=str[a[i]]];
   for(i=1;i<m;++i)
      ws[i]+=ws[i-1];
   for(i=n-1;i>=0;--i)
      b[--ws[wv[i]]]=a[i];
}

inline void dc3(int *str,int *sa,const int &n,const int &m)
{
   int *strn(str+n);
   int *san(sa+n),tb((n+1)/3),ta(0),tbc(0),i,j,k;
   str[n]=str[n+1]=0;
   for(i=0;i<n;++i)
      if(i%3)
         wa[tbc++]=i;
   sort(str+2,wa,wb,tbc,m);
   sort(str+1,wb,wa,tbc,m);
   sort(str,wa,wb,tbc,m);
   for(i=j=1,strn[F(wb[0])]=0;i<tbc;++i)
      strn[F(wb[i])]=c0(str,wb[i-1],wb[i])?j-1:j++;
   if(j<tbc)
      dc3(strn,san,tbc,j);
   else
      for(i=0;i<tbc;++i)
         san[strn[i]]=i;
   for(i=0;i<tbc;++i)
      if(san[i]<tb)
```

```
         wb[ta++]=san[i]*3;
   if(n%3==1)
      wb[ta++]=n-1;
   sort(str,wb,wa,ta,m);
   for(i=0;i<tbc;++i)
      wv[wb[i]=G(san[i])]=i;
   for(i=j=k=0;i<ta && j<tbc;)
      sa[k++]=c12(str,wb[j]%3,wa[i],wb[j])?wa[i++]:wb[j++];
   while(i<ta)
      sa[k++]=wa[i++];
   while(j<tbc)
      sa[k++]=wb[j++];
}

int rk[MAXX],lcpa[MAXX],sa[MAXX*3];
int str[MAXX*3]; //int

int main()
{
   scanf("%d %d",&n,&j);
   for(i=0;i<n;++i)
   {
      scanf("%d",&k);
      num[i]=k-j+100;
      j=k;
   }
   num[n]=0;

   dc3(num,sa,n+1,191); //191: str

   for(i=1;i<=n;++i) // rank
      rk[sa[i]]=i;
   for(i=k=0;i<n;++i) // lcp
      if(!rk[i])
         lcpa[0]=0;
      else
      {
         j=sa[rk[i]-1];
         if(k>0)
            --k;
         while(num[i+k]==num[j+k])
            ++k;
         lcpa[rk[i]]=k;
      }


   for(i=1;i<=n;++i)
      sptb[0][i]=i;
   for(i=1;i<=lg[n];++i) //sparse table RMQ
   {
      k=n+1-(1<<i);
      for(j=1;j<=k;++j)
      {
         a=sptb[i-1][j];
         b=sptb[i-1][j+(1<<(i-1))];
         sptb[i][j]=lcpa[a]<lcpa[b]?a:b;
      }
   }
}

inline int ask(int l,int r)
{
   a=lg[r-l+1];
   r-=(1<<a)-1;
   l=sptb[a][l];
   r=sptb[a][r];
   return lcpa[l]<lcpa[r]?l:r;
}

inline int lcp(int l,int r) // [l,r]rmq
{
   l=rk[l];
   r=rk[r];
   if(l>r)
      std::swap(l,r);
   return lcpa[ask(l+1,r)];
}
```

9.7 Suffix Array - Prefix-doubling Algorithm

```
int wx[maxn],wy[maxn],*x,*y,wss[maxn],wv[maxn];

bool cmp(int *r,int n,int a,int b,int l)
{
   return a+l<n && b+l<n && r[a]==r[b]&&r[a+l]==r[b+l];
}
void da(int str[],int sa[],int rank[],int height[],int n,int m)
{
   int *s = str;
   int *x=wx,*y=wy,*t,p;
   int i,j;
   for(i=0; i<m; i++)
      wss[i]=0;
   for(i=0; i<n; i++)
      wss[x[i]=s[i]]++;
   for(i=1; i<m; i++)
      wss[i]+=wss[i-1];
   for(i=n-1; i>=0; i--)
      sa[--wss[x[i]]]=i;
   for(j=1,p=1; p<n && j<n; j*=2,m=p)
   {
      for(i=n-j,p=0; i<n; i++)
         y[p++]=i;
      for(i=0; i<n; i++)
         if(sa[i]-j>=0)
            y[p++]=sa[i]-j;
      for(i=0; i<n; i++)
         wv[i]=x[y[i]];
      for(i=0; i<m; i++)
         wss[i]=0;
      for(i=0; i<n; i++)
         wss[wv[i]]++;
      for(i=1; i<m; i++)
         wss[i]+=wss[i-1];
      for(i=n-1; i>=0; i--)
         sa[--wss[wv[i]]]=y[i];
      for(t=x,x=y,y=t,p=1,i=1,x[sa[0]]=0; i<n; i++)
         x[sa[i]]=cmp(y,n,sa[i-1],sa[i],j)?p-1:p++;
   }
   for(int i=0; i<n; i++)
      rank[sa[i]]=i;
   for(int i=0,j=0,k=0; i<n; height[rank[i++]]=k)
      if(rank[i]>0)
         for(k?k--:0,j=sa[rank[i]-1]; i+k < n && j+k < n && str[i+
                k]==str[j+k]; ++k);
}
```