Actinium's ACM Template

Last build at November 1, 2012

目录

1 注意事项

CodeBlocks修改终端:
只要Settings−− >Environment−− >Terminal to launch console programs
用“gnome-terminal -t $TITLE -x”替换掉“xterm -T $TITLE -e”
测试系统速度:

```
#include <iostream>
#include <cstdio>
#include <ctime>
using namespace std;
int main()
{
    int i=0,j=0,k=0;
    time_t start = clock(), end;
    for (int i=0;i<316000000;i++){
    }
    end = clock();
    printf("done, total time:%ld ms\n", (end - start)/1000);
    return 0;
}
```

对拍程序:

```
while true;do
./gen>in
./bruteforce<in>ans
./prog<in>out
if diff out ans;then
echo AC
else
echo WA
read p
fi
sleep 1
done
```

2 数据结构

2.1 倍增法求LCA

```
# include <iostream>
# include <cstring>
# include <cstdio>
# include <vector>
# define N 55010
using namespace std;
struct node
{
    int u,w;
    //node (){}
    node (int uu=0,int ww=0):u(uu),w(ww){}
};
vector <node> tree[N];
int p[N][50],d[N],dist[N];
void dfs (int x,int pre)
{
    int y,len=tree[x].size(),now;
    for (int i=0;i<len;i++)
    {
        y=tree[x][i].u;
        if (y==pre)
            continue;
        now=p[y][0]=x;
        d[y]=d[x]+1;
        dist[y]=dist[x]+tree[x][i].w;
        for (int j=0;now!=-1 && p[now][j]!=-1;j++)
        {
            p[y][j+1]=p[now][j];
            now=p[now][j];
        }
        dfs(y,x);
    }
}
int LCA(int x,int y)
{
    if (x==y)
        return x;
    int tem;
    if (d[x]<d[y])
    {
        tem=x;
        x=y;
        y=tem;
    }
    int dis=d[x]-d[y],k=0;
    while (dis)
    {
        if (dis&1)
            x=p[x][k];
        dis>>=1;
        k++;
    }
    k=0;
    while (x!=y)
    {
        if ( p[x][k]!=p[y][k] || (p[x][k]==p[y][k] && k==0) )
        {
            x=p[x][k];
            y=p[y][k];
            k++;
        }else k--;
    }
    return x;
}
int main (void)
{
    int n,u,v,c,q,m;
    char s[20];
    scanf("%d%d",&n,&m);
    for (int i=1;i<=n;i++)
        tree[i].clear();
    memset(p,-1,sizeof(p));
    for (int i=0;i<m;i++)
    {
        scanf("%d%d%d%s",&u,&v,&c,s);
        tree[u].push_back(node(v,c));
        tree[v].push_back(node(u,c));
    }
    dist[1]=0;
    d[1]=0;
    dfs(1,-1);
    scanf("%d",&q);
    while (q--)
    {
        scanf("%d%d",&u,&v);
        printf("%d\n",dist[u]+dist[v]-2*dist[LCA(u,v)]);
    }
    return 0;
}
```

2.2 扫描线求面积并

hdu 4419
扫描线+容斥原理
注意lazy标记不用向下传递，因为一条线段被插入之后还会被删除。还有线段树val值的维护

```
# include <cstring>
# include <cstdio>
# include <cstdlib>
# include <iostream>
# include <algorithm>
using namespace std;
# define N 20500
typedef long long LOL;
struct NODE
{
    int x,y1,y2,id,c;
}a[N];
int val[8][N<<2],len[N<<2],cov[8][N<<2];
int x[N],y[N];
LOL sum[10],ans[10];
void build(int idx,int l,int r)
{
    if (l==r)
    {
        len[idx]=y[l+1]-y[l];
        return ;
    }
    int mid=(l+r)>>1;
    build(idx<<1,l,mid);
    build(idx<<1|1,mid+1,r);
    len[idx]=len[idx<<1]+len[idx<<1|1];
}
inline void up(int now,int idx,int ll)
{
    if (cov[now][idx])
        val[now][idx]=len[idx];
    else
    {
        if (ll)
            val[now][idx]=val[now][idx<<1]+val[now][idx<<1|1];
        else val[now][idx]=0;
    }
}
inline int bin(int a[],int l,int r,int x)
{
    int mid;
    while (l<r)
    {
        mid=(l+r+1)>>1;
        if (a[mid]>x)
            r=mid-1;
        else l=mid;
    }
    return l;
}
void update(int now,int idx,int l,int r,int x,int y,int c)
{
    if (x<=l && y>=r)
    {
        cov[now][idx]+=c;
        up(now,idx,r-l);
        return ;
    }
    int mid=(l+r)>>1;
    if (y<=mid)
        update(now,idx<<1,l,mid,x,y,c);
    else if (x>mid)
        update(now,idx<<1|1,mid+1,r,x,y,c);
    else
    {
        update(now,idx<<1,l,mid,x,mid,c);
        update(now,idx<<1|1,mid+1,r,mid+1,y,c);
    }
    up(now,idx,r-l);
}
int cmp(NODE a,NODE b)
{
    return a.x<b.x;
}
int main (void)
{
    int t,n,ys=0;
    char op[20];
    scanf("%d",&t);
    while (t--)
    {
        scanf("%d",&n);
        int x1,x2,y1,y2,cnt=0;
        for (int i=1;i<=n;i++)
        {
            scanf("%s%d%d%d%d",op,&x1,&y1,&x2,&y2);

            ++cnt;
            if (op[0]=='R')
                a[cnt].c=1;
            else
                if (op[0]=='G')
                    a[cnt].c=2;
                else a[cnt].c=4;

            a[cnt].x=x1;
            a[cnt].y1=y1,a[cnt].y2=y2;
            a[cnt].id=1;
            x[cnt]=x1,y[cnt]=y1;

            ++cnt;
            a[cnt].c=a[cnt-1].c;
            a[cnt].x=x2;
            a[cnt].y1=y1,a[cnt].y2=y2;
            a[cnt].id=-1;
            x[cnt]=x2,y[cnt]=y2;
        }

        sort(x+1,x+1+cnt);
        sort(y+1,y+1+cnt);
        sort(a+1,a+1+cnt,cmp);

        int n1=1,n2=1;
        for (int i=2;i<=cnt;i++)
            if (x[i]!=x[i-1])
                x[++n1]=x[i];
        for (int i=2;i<=cnt;i++)
            if (y[i]!=y[i-1])
                y[++n2]=y[i];

        memset(val,0,sizeof(val));
        memset(cov,0,sizeof(cov));
        memset(sum,0,sizeof(sum));
        build(1,1,n2-1);


        int cur=1;
        for (int i=1;i<=n1;i++)
        {
            for (int p=1;p<8;p++)
                sum[p]+=(LOL)val[p][1]*(LOL)(x[i]-x[i-1]);
            while (cur<=cnt && a[cur].x==x[i])
            {
                y1=bin(y,1,n2,a[cur].y1);
                y2=bin(y,1,n2,a[cur].y2)-1;
                for (int p=1;p<8;p++)
                {
                    if (!(a[cur].c&p)) continue;
                    update(p,1,1,n2-1,y1,y2,a[cur].id);
                }
                ++cur;
            }
        }
        LOL tmp3=sum[1]+sum[2]-sum[3];
        LOL tmp5=sum[1]+sum[4]-sum[5];
        LOL tmp6=sum[2]+sum[4]-sum[6];
        LOL tmp7=sum[7]-sum[1]-sum[2]-sum[4]+tmp3+tmp5+tmp6;
        ans[1]=sum[1]-tmp3-tmp5+tmp7;
        ans[2]=sum[2]-tmp3-tmp6+tmp7;
        ans[3]=sum[4]-tmp5-tmp6+tmp7;
        ans[4]=tmp3-tmp7;
        ans[5]=tmp5-tmp7;
        ans[6]=tmp6-tmp7;
        ans[7]=tmp7;
        cout<<"Case "<<++ys<<":"<<endl;
        for (int i=1;i<8;i++)
            cout<<ans[i]<<endl;
    }
    return 0;
}
```

2.3 矩形周长并

HDU 1828.cpp
矩形周长并，统计y轴上线段条数
注意cmp函数，先入再出

```
# include <iostream>
# include <cstdio>
# include <cstdlib>
# include <cstring>
# include <algorithm>
using namespace std;
# define N 10050
struct SEG
{
    int x,y1,y2,flag;
}seg[N];
struct NODE
{
    int lc,rc,len,sum,num,cover;
}tree[N<<2];
int x[N],y[N];
int cmp(SEG a,SEG b)
{
    if (a.x!=b.x)
        return a.x<b.x;
    return a.flag>b.flag;
}
int bin_search(int a[],int l,int r,int x)
{
    int mid;
    while (l<r)
    {
        mid=(l+r)>>1;
        if (x>a[mid])
            l=mid+1;
        else r=mid;
    }
    return l;
}
void build(int idx,int l,int r)
{
    tree[idx].lc=tree[idx].rc=tree[idx].sum=tree[idx].cover=tree[idx].num=0;
    if (l==r)
        tree[idx].len=y[l+1]-y[l];
    else
    {
        int mid=(l+r)>>1;
        build(idx<<1,l,mid);
        build(idx<<1|1,mid+1,r);
        tree[idx].len=tree[idx<<1].len+tree[idx<<1|1].len;
    }
}
inline void solve(int idx,int len)
{
    if (tree[idx].cover)
    {
        tree[idx].sum=tree[idx].len;
        tree[idx].num=2;
        tree[idx].lc=tree[idx].rc=1;
    }
    else
    {
        if (len>1)
        {
            tree[idx].sum=tree[idx<<1].sum+tree[idx<<1|1].sum;
            tree[idx].num=tree[idx<<1].num+tree[idx<<1|1].num;
            if (tree[idx<<1].rc && tree[idx<<1|1].lc)
                tree[idx].num-=2;
            tree[idx].lc=tree[idx<<1].lc;
            tree[idx].rc=tree[idx<<1|1].rc;
        }
        else
        {
            tree[idx].num=tree[idx].sum=0;
            tree[idx].lc=tree[idx].rc=0;
        }
    }
}
void update(int idx,int l,int r,int x,int y,int c)
{
    if (x<=l && y>=r)
    {
        tree[idx].cover+=c;
        solve(idx,r-l+1);
        return ;
    }
    int mid=(l+r)>>1;
    if (y<=mid)
        update(idx<<1,l,mid,x,y,c);
    else
        if (x>mid)
            update(idx<<1|1,mid+1,r,x,y,c);
        else
        {
            update(idx<<1,l,mid,x,mid,c);
            update(idx<<1|1,mid+1,r,mid+1,y,c);
        }
    solve(idx,r-l+1);
}
int main (void)
{
    int n;
    while (cin>>n)
    {
        int cnt=0,x1,x2,y1,y2;
        for (int i=1;i<=n;i++)
        {
            scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
            seg[++cnt].x=x1,seg[cnt].y1=y1,seg[cnt].y2=y2,seg[cnt].flag=1;
            x[cnt]=x1,y[cnt]=y1;
            seg[++cnt].x=x2,seg[cnt].y1=y1,seg[cnt].y2=y2,seg[cnt].flag=-1;
            x[cnt]=x2,y[cnt]=y2;
        }
        sort(x+1,x+1+cnt);
        sort(y+1,y+1+cnt);
        sort(seg+1,seg+1+cnt,cmp);
        int nx=1,ny=1;
        for (int i=2;i<=cnt;i++)
        {
            if (x[i]!=x[i-1])
                x[++nx]=x[i];
            if (y[i]!=y[i-1])
                y[++ny]=y[i];
        }
        build(1,1,ny-1);
        int ans=0,last=0,j=1,k;
        for (int i=1;i<=nx;i++)
        {
            if (i>1)
                ans+=(x[i]-x[i-1])*tree[1].num;// 加矩形长边
            for (k=j;k<=cnt && seg[k].x==x[i];k++)
            {
                update(1,1,ny-1,bin_search(y,1,ny,seg[k].y1),bin_search(y,1,
                   ny,seg[k].y2)-1,seg[k].flag);
                ans+=abs(tree[1].sum-last);// 加矩形宽边，每次加的是增量
                last=tree[1].sum;
            }
            j=k;
        }
        cout<<ans<<endl;
    }
    return 0;
}
```

2.4 树状数组

2.4.1 树状数组区间加减，区间查询

树状数组区间加一个值，然后询问区间的和。假设原数组为a，再维护一个数组b,b_i记录前i个元素每个全部加了b_i的值。当我们查询前x项的值时，那么$sum = \sum_{i=1}^{x} a_i + x * \sum_{i=x+1}^{n} b_i + \sum_{i=1}^{x} b_i * i$ ，均化为了求和的操作，那么可以用树状数组去维护b_i与$b_i * i$的值。

```
#include<iostream>
#include<cstring>
#include<cstdio>
using namespace std;
int lowbit(int x)
{
    return x&(-x);
}
typedef long long LOL;
const int N = 100010;
LOL a[N],d[N],c[N],sum[N];
int n;
void Add(int x,int v)
{
    if(x==0) return ;
    LOL t=v*x;
    while(x<=n)
    {
        d[x]+=v;
        c[x]+=t;
        x+=lowbit(x);
    }
}
LOL getsumd(int x)
{
    LOL u=x;
    LOL sum=0;
    while(x>0){
        sum+=d[x];
        x-=lowbit(x);
    }
    return sum;
}
LOL getsumc(int x)
{
    LOL u=x;
    LOL sum=0;
    while(x>0){
        sum+=c[x];
        x-=lowbit(x);
    }
    return sum;
}
LOL getsum(int x)
{
    LOL s1=getsumd(n)-getsumd(x);
    LOL s2=getsumc(x);
    return sum[x]+(LOL)x*s1+s2;
}
char com[2];
int main(void)
{
    int m,w;
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++)
    {
        scanf("%lld",&a[i]);
        sum[i]=sum[i-1]+a[i];
    }
    int x,y;
    for(int i=1;i<=m;i++)
    {
        scanf("%s%d%d",com,&x,&y);
        if(com[0]=='Q')
        {
            printf("%lld\n",getsum(y)-getsum(x-1));
        }
        else
        {
            scanf("%d",&w);
            Add(y,w);Add(x-1,-w);
        }
    }
    return 0;
}
```

2.4.2 树状数组求最值

HDU1754
复杂度为$O(n * \log n * \log n)$

```
#include<iostream>
#include<cstring>
#include<cstdio>
using namespace std;
const int N = 200010;
int a[N],c[N];
int lowbit(int x)
{
    return x&(-x);
}
void init(int n)
{
    for(int i=1;i<=n;i++){
        c[i]=a[i];
        for(int j=1;j<lowbit(i);j<<=1){
            c[i]=max(c[i],c[i-j]);
        }
    }
}
int query(int l,int r)
{
    int ans=a[r];
    while(l<=r){
        ans=max(ans,a[r]);
        if(r-lowbit(r)+1>=l){
            ans=max(ans,c[r]);
            r-=lowbit(r);
        }else{
            r--;
        }
    }
    return ans;
}
void modify(int idx,int v,int n)
{
    a[idx]=v;
    for(int i=idx;i<=n;i+=lowbit(i)){
        c[i]=v;
        for(int j=1;j<lowbit(i);j<<=1){
            c[i]=max(c[i],c[i-j]);
        }
    }
}
int main(void)
{
    int n,m,x,y;
    while(scanf("%d%d",&n,&m)!=EOF)
    {
        for(int i=1;i<=n;i++){
            scanf("%d",&a[i]);
        }
        init(n);
        char op[2];
        for(int i=1;i<=m;i++){
            scanf("%s%d%d",op,&x,&y);
            if(op[0]=='Q'){
                printf("%d\n",query(x,y));
            }else{
                modify(x,y,n);
            }
        }
    }
    return 0;
}
```

2.4.3 树状数组求第K大

```
int find(int k)
{
    int cnt=0,res=0;
    for(int i=18;i>=0;i--){
        res+=1<<i;
        if(res>=maxn||cnt+c[res]>=k){
            res-=(1<<i);
        }else{
            cnt+=c[res];
        }
    }
    return res+1;
}
```

2.5 可修改堆

```cpp
#include<iostream>
#include<cstdio>
#include<cstring>
using namespace std;
const int N = 1000010;
struct HEAP
{
    int x,node;
}heap[N];
int set[N];
int size=0;
int a[N];
void Swap(int x,int y)
{
    swap(heap[x],heap[y]);
    set[heap[x].node]=x;
    set[heap[y].node]=y;
}
void up(int &pos)
{
    while(pos>=2&&heap[pos].x<heap[pos>>1].x)
    {
        Swap(pos,pos>>1);
        pos=pos>>1;
    }
}
void down(int &pos)
{
    int j=pos<<1;
    while(j<=size)
    {
        if(j<size&&heap[j+1].x<heap[j].x)
        {
            j++;
        }
        if(heap[pos].x<=heap[j].x) break;
        Swap(pos,j);
        pos=j;j=pos<<1;
    }
}
void insert(int size,int node,int val)
{
    heap[size].x=val;
    heap[size].node=node;
    set[node]=size;
    up(size);
}
int delmin(int size)
{
    int tmp=heap[1].node;
    Swap(1,size);
    size--;
    int pos=1;
    down(pos);
    return tmp;
}
void modify(int pos,int val)
{
    heap[pos].x=val;
    up(pos);
    down(pos);
}
void del(int pos)
{
    Swap(pos,size);
    size--;
    up(pos);
    down(pos);
}
int main(void)
{
    return 0;
}
```

2.6 莫队算法

基本可以处理各种序列问题，复杂度$O(n * \sqrt{n})$

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<cmath>
using namespace std;
const int N =100010;
struct T
{
    int x,y;
    int id;
}tt[N];
int sz;
bool cmp(T a,T b)
{
    if(a.x/sz==b.x/sz){
        return a.y<b.y;
    }
    return a.x/sz<b.x/sz;
}
int a[N];
int ans[N];
int cnt[N];
int num[N];
int maxx;
void add(int x)
{
    cnt[a[x]]++;
    num[cnt[a[x]]-1]--;
    num[cnt[a[x]]]++;
    if(maxx<cnt[a[x]]){
        maxx=cnt[a[x]];
    }
}
void del(int x)
{
    cnt[a[x]]--;
    num[cnt[a[x]]+1]--;
    num[cnt[a[x]]]++;
    if(maxx==cnt[a[x]]+1){
        if(num[cnt[a[x]]+1]==0){
            maxx=cnt[a[x]];
        }
    }
}
int main(void)
{
    int n,m;
    scanf("%d%d",&n,&m);
    sz=(int)sqrt((double)n);
    for(int i=1;i<=n;i++){
        scanf("%d",&a[i]);
    }
    for(int i=1;i<=m;i++){
        scanf("%d%d",&tt[i].x,&tt[i].y);
        tt[i].x++;tt[i].y++;
        tt[i].id=i;
    }
    sort(tt+1,tt+1+m,cmp);
    int cl=1,cr=1;
    num[0]=n;
    maxx=0;
    add(1);
    for(int i=1;i<=m;++i){
        while(cr < tt[i].y) add(++ cr);
        while(cl < tt[i].x) del(cl ++);
        while(cl > tt[i].x) add(-- cl);
        while(cr > tt[i].y) del(cr --);
        ans[tt[i].id] = maxx;
    }
    for(int i=1;i<=m;++i) printf("%d\n", ans[i]);
    return 0;
}
```

2.7 划分树

```
#include<iostream>
#include<cstring>
#include<cstdio>
#include<algorithm>
using namespace std;
const int N = 100010;
struct Seg
{
    int left,right;
    int mid(){
        return (left+right)>>1;
    }
}tt[N<<2];
struct Q
{
    int op,x,y,k,now;
}q[N<<1];
int val[25][N],toleft[25][N],sorted[N];
void build(int left,int right,int d,int idx)
{
    tt[idx].left=left;
    tt[idx].right=right;
    if(left==right) return ;
    int mid=tt[idx].mid();
    int lsame=mid-left+1;
    for(int i=left;i<=right;i++)
    {
        if(val[d][i]<sorted[mid])
        {
            lsame--;
        }
    }
    int lpos=left,rpos=mid+1,same=0;
    for(int i=left;i<=right;i++)
    {
        if(i==left)
        {
            toleft[d][i]=0;
        }
        else
        {
            toleft[d][i]=toleft[d][i-1];
        }
        if(val[d][i]<sorted[mid]){
            toleft[d][i]++;
            val[d+1][lpos++]=val[d][i];
        }
        else if(val[d][i]>sorted[mid])
        {
            val[d+1][rpos++]=val[d][i];
        }
        else
        {
            if(same<lsame)
            {
                same++;
                toleft[d][i]++;
                val[d+1][lpos++]=val[d][i];
            }
            else
            {
                    val[d+1][rpos++]=val[d][i];
                }
            }
        }
        build(left,mid,d+1,idx<<1);
        build(mid+1,right,d+1,idx<<1|1);
}
int query(int left,int right,int k,int d,int idx)//left到right的第k小的数
{
    if(left==right)
    {
        return val[d][left];
    }
    int s;
    int ss;
    if(left==tt[idx].left)
    {
        s=toleft[d][right];
        ss=0;
    }
    else
    {
        s=toleft[d][right]-toleft[d][left-1];
        ss=toleft[d][left-1];
    }
    if(s>=k)
    {
        int newl=tt[idx].left+ss;
        int newr=tt[idx].left+ss+s-1;
        return query(newl,newr,k,d+1,idx<<1);
    }
    else
    {
        int mid=tt[idx].mid();
        int tmp=left-tt[idx].left-ss;
        int newl=mid+1+tmp;
        int newr=mid+1+tmp+right-left-s;
        return query(newl,newr,k-s,d+1,idx<<1|1);
    }
}
int find(int d,int id,int x,int idx)//前id个数小于等于x的有多少个
{
    if(id<tt[idx].left) return 0;
    if(tt[idx].left==tt[idx].right) return sorted[id]<=x;
    int s=toleft[d][id];
    int mid=tt[idx].mid();
    if(sorted[mid]<=x)
    {
        return find(d+1,mid+id-tt[idx].left+1-s,x,idx<<1|1)+s;
    }
    else
    {
        return find(d+1,tt[idx].left+s-1,x,idx<<1);
    }
}
char com[20];
int main(void)
{
    int n,ys=0;
    while(scanf("%d",&n)!=EOF)
    {
        int len=0;
        for(int i=1;i<=n;i++)
        {
            scanf("%s",com);
            if(strcmp(com,"Insert")==0)
            {
                scanf("%d",&q[i].x);
                q[i].op=1;
                ++len;
                val[0][len]=q[i].x;
                sorted[len]=val[0][len];
            }
            else if(strcmp(com,"Query_1")==0)
            {
                scanf("%d%d%d",&q[i].x,&q[i].y,&q[i].k);
                q[i].op=2;
            }
            else if(strcmp(com,"Query_2")==0)
            {
                scanf("%d",&q[i].k);
                q[i].op=3;
                q[i].x=1;q[i].y=len;
            }
            else
            {
                scanf("%d",&q[i].k);
                q[i].op=4;
                q[i].x=1;q[i].y=len;
            }
        }
        sort(sorted+1,sorted+1+len);
        build(1,len,0,1);
        printf("Case %d:\n",++ys);
        long long sum1=0,sum2=0,sum3=0;
        for(int i=1;i<=n;i++)
        {
            if(q[i].op==2)
            {
                sum1+=(long long)query(q[i].x,q[i].y,q[i].k,0,1);
            }
            else if(q[i].op==4)
            {
                sum3+=(long long)query(q[i].x,q[i].y,q[i].k,0,1);
            }
            else if(q[i].op==3)
            {
                sum2+=(long long)find(0,q[i].y,q[i].k,1);
            }
        }
        printf("%lld\n%lld\n%lld\n",sum1,sum2,sum3);
    }
    return 0;
}
```

2.8 二维RMQ

```
#include<iostream>
#include<cstring>
#include<cstdio>
using namespace std;
const int M = 301;
int val[M][M];
int Max[9][9][M][M];
int idx[M];
void initRMQ(int n,int m){
    for(int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            Max[0][0][i][j]=val[i][j];
        }
    }
    for(int i=0;i<=idx[n];i++){
        int limit1=n+1-(1<<i);
        for(int j=0;j<=idx[m];j++){
            if(!i&&!j) continue;
            int limit2=m+1-(1<<j);
            for(int ii=1;ii<=limit1;ii++){
                for(int jj=1;jj<=limit2;jj++){
                    if(i) Max[i][j][ii][jj]=min(Max[i-1][j][ii+(1<<i>>1)][jj
                       ],Max[i-1][j][ii][jj]);
                    else Max[i][j][ii][jj]=min(Max[i][j-1][ii][jj],Max[i][j
                       -1][ii][jj+(1<<j>>1)]);
                }
            }
        }
    }
}
int query(int a,int b,int c,int d)
{
    int n=idx[c-a+1],m=idx[d-b+1];
    c-=(1<<n)-1;
    d-=(1<<m)-1;
    return min(min(Max[n][m][a][b],Max[n][m][a][d]),min(Max[n][m][c][b],Max[
       n][m][c][d]));
}
int main(void)
{
    idx[0]=-1;
    for(int i=1;i<=300;i++){
        idx[i]=(i&(i-1))?idx[i-1]:idx[i-1]+1;
    }
    int T,n;
    scanf("%d",&T);
    while(T--){
        scanf("%d",&n);
        for(int i=1;i<=n;i++){
            for(int j=1;j<=n;j++){
                scanf("%d",&val[i][j]);
            }
        }
        initRMQ(n,n);
        int a,b,c,d,m;
        scanf("%d",&m);
        while(m--){
            scanf("%d%d%d%d",&a,&b,&c,&d);
            printf("%d\n",query(a,b,c,d));
        }
    }
    return 0;
}
```

2.9 主席树

```
#include<iostream>
#include<cstring>
#include<cstdio>
#include<algorithm>
using namespace std;
const int N = 200001,M=200001,NN=3000000;
struct Seg
{
    int l,r,lson,rson,s;
}a[NN];
int root[N];
int b[N],v[N],now[N];
int m;
int bin(int x)
{
    int left=1,right=m;
    while(left<=right)
    {
        int mid=(left+right)>>1;
        if(v[mid]==x)
        {
            return mid;
        }
        if(v[mid]<x) left=mid+1;
        else right=mid-1;
    }
    return 0;
}
int tot=0;
int build(int l,int r)
{
    int k=++tot;
    a[k].l=l;a[k].r=r;a[k].s=0;
    if(l==r)
    {
        return k;
    }
    int mid=(l+r)>>1;
    a[k].lson=build(l,mid);
    a[k].rson=build(mid+1,r);
    return k;
}
int change(int p,int x,int w)
{
    int k=++tot;
    a[k].l=a[p].l;a[k].r=a[p].r;
    a[k].lson=a[p].lson;a[k].rson=a[p].rson;
    a[k].s=a[p].s+w;
    if(a[k].l==a[k].r) return k;
    int mid=(a[k].l+a[k].r)>>1;
    if(mid<x)
    {
        a[k].rson=change(a[p].rson,x,w);
    }
    else
    {
        a[k].lson=change(a[p].lson,x,w);
    }
    return k;
}
int query(int p,int q,int k)
```

```
{
    if(a[p].l==a[p].r) return a[p].l;
    int now=a[a[q].lson].s-a[a[p].lson].s;
    if(now>=k)
    {
        return query(a[p].lson,a[q].lson,k);
    }
    else
    {
        return query(a[p].rson,a[q].rson,k-now);
    }
}
int main(void)
{
    int n,q,x,y,k;
    scanf("%d%d",&n,&q);
    for(int i=1;i<=n;i++)
    {
        scanf("%d",&b[i]);
        v[i]=b[i];
    }
    sort(v+1,v+1+n);
    m=1;
    for(int i=2;i<=n;i++)
    {
        if(v[i]!=v[i-1])
        {
            v[++m]=v[i];
        }
    }
    for(int i=1;i<=n;i++)
    {
        now[i]=bin(b[i]);
    }
    root[0]=build(1,m);
    for(int i=1;i<=n;i++)
    {
        root[i]=change(root[i-1],now[i],1);
    }
    for(int i=1;i<=q;i++)
    {
        scanf("%d%d%d",&x,&y,&k);
        printf("%d\n",v[query(root[x-1],root[y],k)]);
    }
    return 0;
}
```

2.10 树链剖分

```
//#pragma comment(linker, "/STACK:1024000000,1024000000")
#include<iostream>
#include<cstring>
#include<cstdio>
#include<vector>
using namespace std;
const int N = 500010;
int sz[N],dep[N],fa[N],son[N],val[N],top[N];
int g[N],d[N];
vector<int> edge[N];
int loc=0;
void dfs(int x,int dp,int pa)
{
    fa[x]=pa;
    sz[x]=1;
    dep[x]=dp;
    son[x]=0;
    for(int i=0;i<edge[x].size();i++)
    {
        int u=edge[x][i];
        if(u!=pa)
        {
            dfs(u,dp+1,x);
            sz[x]+=sz[u];
            if(sz[son[x]]<sz[u])
            {
                son[x]=u;
            }
        }
    }
}
void find_link(int x,int tp)
{
    val[x]=++loc;
    d[loc]=g[x];
    top[x]=tp;
    if(son[x]!=0)
    {
        find_link(son[x],tp);
    }
    for(int i=0;i<edge[x].size();i++)
    {
        int u=edge[x][i];
        if(u!=son[x]&&fa[u]==x)
        {
            find_link(u,u);
        }
    }
}
int add[N<<2],sum[N<<2];
void up(int idx)
{
    sum[idx]=sum[idx<<1]+sum[idx<<1|1];
}
void down(int idx,int l)
{
    if(add[idx]!=0)
    {
        add[idx<<1]+=add[idx];
        add[idx<<1|1]+=add[idx];
        sum[idx<<1]+=add[idx]*(l-(l>>1));
        sum[idx<<1|1]+=add[idx]*(l>>1);
        add[idx]=0;
    }
}
void build(int left,int right,int idx)
{
    add[idx]=0;
    if(left==right)
    {
        sum[idx]=d[left];
        return ;
    }
    int mid=(left+right)>>1;
    build(left,mid,idx<<1);
    build(mid+1,right,idx<<1|1);
    up(idx);
}
void update(int left,int right,int L,int R,int v,int idx)
{
    if(left>=L&&right<=R)
    {
        add[idx]+=v;
        sum[idx]+=v*(right-left+1);
        return ;
    }
    down(idx,right-left+1);
    int mid=(left+right)>>1;
    if(mid<L)
    {
        update(mid+1,right,L,R,v,idx<<1|1);
    }
    else if(mid>=R)
    {
        update(left,mid,L,R,v,idx<<1);
    }
    else
    {
        update(left,mid,L,mid,v,idx<<1);
        update(mid+1,right,mid+1,R,v,idx<<1|1);
    }
    up(idx);
}
int query(int left,int right,int id,int idx)
{
    if(left==right)
    {
        return sum[idx];
    }
    down(idx,right-left+1);
    int mid=(left+right)>>1;
    if(mid<id)
    {
        return query(mid+1,right,id,idx<<1|1);
    }
    else
    {
        return query(left,mid,id,idx<<1);
    }
}
void modify(int va,int vb,int v)
{
    int f1=top[va],f2=top[vb];
    while(f1!=f2)
    {
        if(dep[f1]<dep[f2])
        {
            swap(f1,f2);
            swap(va,vb);
        }
        update(1,loc,val[f1],val[va],v,1);
        va=fa[f1];f1=top[va];
    }
    if(dep[va]<dep[vb]) swap(va,vb);
    update(1,loc,val[vb],val[va],v,1);
}
int main(void)
{
    int n,m,q,x,y,w;
    char op[2];
    while(scanf("%d%d%d",&n,&m,&q)!=EOF)
    {
        for(int i=1;i<=n;i++)
        {
            scanf("%d",&g[i]);
            edge[i].clear();
        }
        for(int i=1;i<=m;i++)
        {
            scanf("%d%d",&x,&y);
            edge[x].push_back(y);
            edge[y].push_back(x);
        }
        loc=0;
        dfs(1,0,-1);
        find_link(1,1);
        build(1,n,1);
        for(int i=1;i<=q;i++)
        {
            scanf("%s",op);
            if(op[0]=='I')
            {
                scanf("%d%d%d",&x,&y,&w);
                modify(x,y,w);
            }
            else if(op[0]=='Q')
            {
                scanf("%d",&x);
                int ans=query(1,n,val[x],1);
                printf("%d\n",ans);

            }
            else
            {
                scanf("%d%d%d",&x,&y,&w);
                modify(x,y,-w);
            }
        }
    }
    return 0;
}
```

2.11 KD树

```
#include<iostream>
#include<cstring>
#include<cstdio>
#include<algorithm>
#include<queue>
#include<vector>
using namespace std;
const int N = 100010;
const int oo = 20010;
int Div[N];
struct P
{
    int a[5];
    int num;
    int dis;
}tt[N],p[N],tmp,tx;
int m,now,k;
int minn[10],maxx[10];
struct Pcmp
{
    bool operator()(P a,P b)
    {
        return a.dis<b.dis;
    }
};
priority_queue<P,vector<P>,Pcmp> que;
bool cmp(P a,P b)
{
    return a.a[now]<b.a[now];
}
int dist(P a,P b)
{
    int sum=0;
    for(int i=0;i<k;i++)
    {
        sum+=(a.a[i]-b.a[i])*(a.a[i]-b.a[i]);
    }
    return sum;
}
void build(int l,int r,P p[])
{
    if(l>r) return ;
    int mid=(l+r)>>1;
    for(int i=0;i<k;i++)
    {
        minn[i]=oo;
        maxx[i]=-oo;
    }
    for(int i=l;i<=r;i++)
    {
        for(int j=0;j<k;j++)
        {
            minn[j]=min(minn[j],p[i].a[j]);
            maxx[j]=max(maxx[j],p[i].a[j]);
        }
    }
    now=0;
    for(int i=1;i<k;i++)
    {
        if(maxx[i]-minn[i]>maxx[now]-minn[now])
        {
            now=i;
        }
    }
    Div[mid]=now;
    nth_element(p+l,p+mid,p+r+1,cmp);
    build(l,mid-1,p);
    build(mid+1,r,p);
}
void find(int l,int r,P a,P p[])
{
    if(l>r) return ;
    int mid=(l+r)>>1;
    int dis=dist(a,p[mid]);
    if(que.size()<m)
    {
        tx=p[mid];tx.dis=dis;
        que.push(tx);
    }
    else if(que.top().dis>dis)
    {
        tx=p[mid];tx.dis=dis;
        que.pop();
        que.push(tx);
    }
    int d=a.a[Div[mid]]-p[mid].a[Div[mid]];
    int l1=l,l2=mid+1,r1=mid-1,r2=r;
    if(d>0) swap(l1,l2),swap(r1,r2);
    find(l1,r1,a,p);
    if(que.size()<m||d*d<que.top().dis) find(l2,r2,a,p);
}
int main(void)
{
    int n,q;
    //freopen("1009.in","r",stdin);
    //freopen("1.out","w",stdout);
    while(scanf("%d%d",&n,&k)!=EOF)
    {
        for(int i=1;i<=n;i++)
        {
            for(int j=0;j<k;j++)
            {
                scanf("%d",&tt[i].a[j]);
                p[i].a[j]=tt[i].a[j];
            }
        }
        build(1,n,p);
        scanf("%d",&q);
        while(q--)
        {
            for(int i=0;i<k;i++)
            {
                scanf("%d",&tmp.a[i]);
            }
            scanf("%d",&m);
            while(!que.empty()) que.pop();
            vector<P> v;
            find(1,n,tmp,p);
            while(!que.empty()) v.push_back(que.top()),que.pop();
            printf("the closest %d points are:\n",m);
            for(int i=v.size()-1;i>=0;i--)
            {
                for(int j=0;j<k;j++)
                {
```

```c
                    printf("%d%c",v[i].a[j],j==k-1?'\n':' ');
                }
            }
        }
    }
    return 0;
}
```

2.12 treap

```
#include<iostream>
#include<cstring>
#include<cstdio>
#include<cstdlib>
using namespace std;
const int N = 1000010;
struct Node
{
    int val,fix,left,right;
    int size,weight;
}tn[N];
int loc,root;
void init()
{
    loc=0,root=0,srand(1992);
}
int newnode(int v)
{
    ++loc;
    tn[loc].left=0;tn[loc].right=0;
    tn[loc].size=tn[loc].weight=1;
    tn[loc].val=v;
    tn[loc].fix=rand();
    return loc;
}
void up(int x)
{
    tn[x].size=tn[tn[x].left].size+tn[tn[x].right].size+tn[x].weight;
}
void lr(int &x)
{
    int y=tn[x].right;
    tn[x].right=tn[y].left;
    tn[y].left=x;
    up(x);
    up(y);
    x=y;
}
void rr(int &x)
{
    int y=tn[x].left;
    tn[x].left=tn[y].right;
    tn[y].right=x;
    up(x);
    up(y);
    x=y;
}
void insert(int &x,int v)
{
    if(x==0){
        x=newnode(v);
        return ;
    }
    tn[x].size++;
    if(tn[x].val==v){
        tn[x].weight++;
    }else if(tn[x].val<v){
        insert(tn[x].right,v);
        if(tn[tn[x].right].fix<tn[x].fix){
            lr(x);
        }
    }else{
        insert(tn[x].left,v);
        if(tn[tn[x].left].fix<tn[x].fix){
            rr(x);
        }
    }
}
bool contain(int x,int v)
{
    if(!x){
        return false;
    }
    if(tn[x].val==v){
        return true;
    }else if(tn[x].val<v){
        return contain(tn[x].right,v);
    }else{
        return contain(tn[x].left,v);
    }
}
void erase(int &x)
{
    if(tn[x].left+tn[x].right==0){
        x=0;
    }else if(tn[x].left*tn[x].right==0){
        x=tn[x].left+tn[x].right;
    }else if(tn[tn[x].left].fix<tn[tn[x].right].fix){
        rr(x);
        erase(tn[x].right);
    }else{
        lr(x);
        erase(tn[x].left);
    }
}
void remove(int &x,int v)
{
    if(!contain(x,v)){
        return ;
    }
    tn[x].size--;
    if(tn[x].val>v){
        remove(tn[x].left,v);
    }else if(tn[x].val<v){
        remove(tn[x].right,v);
    }else{
        tn[x].weight--;
        if(tn[x].weight==0){
            erase(x);
        }
    }
}
int find_kth(int x,int v)
{
    if(tn[x].size<v){
        return -1;
    }
    if(tn[tn[x].left].size>=v){
        return find_kth(tn[x].left,v);
    }else if(tn[x].weight+tn[tn[x].left].size>=v){
        return v;
    }else{
        return find_kth(tn[x].right,v-tn[x].weight-tn[tn[x].left].size);
    }
```

```
}
void Debug(int x)
{
    if(tn[x].left!=0){
        Debug(tn[x].left);
    }
    printf("%的为dvalue%，为dfix%，左儿子为d%d右儿子为,%d\n",x,tn[x].val,tn[x].fix,tn[x].left,tn[x].right);
    if(tn[x].right!=0){
        Debug(tn[x].right);
    }
}
int main(void)
{
    return 0;
}
```

2.13 DancingLinks

2.13.1 精确覆盖

```
#include<iostream>
#include<cstring>
#include<cstdio>
#define N 1010
#define M N*N
int R[M],L[M],U[M],D[M];
int C[M],S[N],O[N],row[M];
int size;
int ak;
void remove(int c)
{//删除一列，并删除同列覆盖的每行
    int i,j;
    L[R[c]]=L[c];
    R[L[c]]=R[c];
    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];
            S[C[j]]--;
        }
    }
}

void resume(int&c)
{//恢复一列及此列覆盖的行
    int i,j;
    for(i=U[c];i!=c;i=U[i]){
        for(j=L[i];j!=i;j=L[j]){
            U[D[j]]=D[U[j]]=j;
            S[C[j]]++;
        }
    }R[L[c]]=L[R[c]]=c;
}
int Dfs(int k)
{
    int min,c,i,j;
    if(R[0]==0){//得到结果
        ak=k;return 1;
    }

    for(min=1000000,c=0,i=R[0];i!=0;i=R[i]){
        if(S[i]<min) min=S[i],c=i;//选取列元素数最少的
    }
    remove(c);
    for(i=D[c];i!=c;i=D[i]){

        for(j=R[i];j!=i;j=R[j])
            remove(C[j]);//删除

        O[k]=row[i];//记录结果
        if(Dfs(k+1))
            return 1;

        for(j=L[i];j!=i;j=L[j])
            resume(C[j]);//恢复
    }
    resume(c);
    return 0;
}
```

```
int main(void)
{
     int n,m,num,x;
     while(scanf("%d%d",&n,&m)!=EOF)
     {
            memset(S, 0, sizeof(S));
            for(int i=1;i<=m;i++)
            {
                R[i-1]=L[i+1]=U[i]=D[i]=i;
            }
            R[m]=0; L[0]=m;
            size=m+1;
            for(int i=1;i<=n;i++)
            {
                int rowh=-1;
                scanf("%d",&num);
                for(int j=1;j<=num;j++)
                {
                    scanf("%d",&x);
                    C[size]=x;
                    D[U[x]]=size;
                    U[size]=U[x];
                    D[size]=x;
                    U[x]=size;
                    S[x]++;
                    row[size]=i;
                    if(rowh==-1)
                    {
                        L[size]=R[size]=size;
                        rowh=size;
                    }
                    else
                    {
                        R[size]=rowh;
                        L[size]=L[rowh];
                        R[L[rowh]]=size;
                        L[rowh]=size;
                    }
                    size++;
                }
            }
            int ans=Dfs(0);
            if(ans==0)
            {
                printf("NO\n");
            }
            else
            {
                printf("%d",ak);
                for(int i=0;i<ak;i++)
                {
                    printf(" %d",O[i]);
                }
                printf("\n");
            }
        }
    return 0;
}
```

2.13.2 重复覆盖

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

```
#include<climits>
using namespace std;
#define N 100
#define M N*N
int R[M],L[M],U[M],D[M];
int C[M],S[N],O[N],row[M];
int Col[M];
int size=0;
void remove(int &c) {
    for(int i = D[c]; i != c ; i = D[i]) {
        L[R[i]] = L[i];
        R[L[i]] = R[i];
    }
}
void resume(int &c) {
    for(int i = U[c]; i != c ; i = U[i]) {
        L[R[i]] = i;
        R[L[i]] = i;
    }
}
int h() {
    bool hash[51];
    memset(hash,false,sizeof(hash));
    int ret = 0;
    for(int c = R[0]; c != 0 ; c = R[c]) {
        if(!hash[c]) {
            ret ++;
            hash[c] = true;
            for(int i = D[c] ; i != c ; i = D[i]) {
                for(int j = R[i] ; j != i ; j = R[j]) {
                    hash[Col[j]] = true;
                }
            }
        }
    }
    return ret;
}
int ans=INT_MAX;
void dfs(int deep) {
    if(deep + h() >= ans) {
        return ;
    }
    if(R[0] == 0) {
        ans=min(ans,deep);
        return ;
    }
    int idx , i , j , minnum = 99999;
    for(i = R[0] ; i != 0 ; i = R[i]) {
        if(S[i] < minnum) {
            minnum = S[i];
            idx = i;
        }
    }
    for(i = D[idx]; i != idx; i = D[i]) {
        remove(i);
        for(j = R[i]; j != i ; j = R[j]) {
            remove(j);
        }
        dfs(deep+1);
        for(j = L[i]; j != i ; j = L[j]) {
            resume(j);
        }
        resume(i);
    }
    return ;
}
int main(void)
{
    int n,m,num,x;
    scanf("%d%d",&n,&m);
    memset(S,0,sizeof(S));
    for(int i=1;i<=n;i++){
        R[i-1]=L[i+1]=U[i]=D[i]=i;
    }
    R[n]=0;L[0]=n;
    size=n+1;
    for(int i=1;i<=m;i++){
        int rowh=-1;
        scanf("%d",&num);
        for(int j=1;j<=num;j++){
            scanf("%d",&x);
            D[U[x]]=size;
            U[size]=U[x];
            D[size]=x;
            U[x]=size;
            S[x]++;
            row[size]=i;
            Col[size]=x;
            if(rowh==-1)
            {
                L[size]=R[size]=size;
                rowh=size;
            }
            else
            {
                R[size]=rowh;
                L[size]=L[rowh];
                R[L[rowh]]=size;
                L[rowh]=size;
            }
            size++;
        }
    }
    dfs(0);
    printf("%d\n",ans);
    return 0;
}
```

3 字符串

3.1 最小表示法

返回的是最小表示法得到的起始值的位置（以0开始）

```
int minirepresent(char *s)
{
    int len=strlen(s);
    int k=0,i=0,j=1;
    while(j<len&&k<len)
    {
        if(s[(i+k)%len]==s[(j+k)%len])
        {
            k++;
        }
        else
        {
            if(s[(i+k)%len]<s[(j+k)%len])
            {
                j+=k+1;
            }
            else
            {
                i=max(j,i+k+1);
                j=i+1;
            }
            k=0;
        }
    }
    return i;
}
```

3.2 manacher

可以求出以某位为中心的回文串的最长回文串的长度（若是偶数，则中心在'#'上），最后的结果为p[id]-1;

```
void manatcher(char *s)
{
    int len=strlen(s);
    p[0]=1;p[1]=1;p[2]=2;
    int id=2;
    for(int i=3;i<len;i++)
    {
        int u=2*id-i;
        if(p[u]+i<p[id]+id)
        {
            p[i]=p[u];
        }
        else
        {
            int j=p[id]+id-i;
            while(i+j<len&&s[i+j]==s[i-j])
            {
                j++;
            }
            p[i]=j;
            id=i;
        }
    }
}
void init(char *s,int len)
{
    str[0]='$';
    str[1]='#';
    for(int i=0;i<len;i++)
    {
        str[i*2+2]=s[i];
        str[i*2+3]='#';
    }
    str[len*2+2]='\0';
}
```

3.3 扩展KMP

next[i]表示以i为起点和自己串匹配的最长公共前缀

```
void get_next(char *s)
{
    int len=strlen(s),id;
    next[0]=len;
    next[1]=0;
    int k=0;
    while(k+1<len&&s[k]==s[k+1])
    {
        k++;
        next[1]++;
    }
    id=1;
    for(int i=2;i<len;i++)
    {
        int u=i-id;
        if(next[u]+i>=next[id]+id)
        {
            int j=next[id]+id-i;
            if(j<0) j=0;
            while(j+i<len&&s[j]==s[j+i]) j++;
            next[i]=j;
            id=i;
        }
        else
        {
            next[i]=next[u];
        }
    }
}
```

t是模式串，s是原串,extend[i]表示以i为起点与模式串最长公共前缀的长度

```
void solve(char *t,char *s)//
{
    get_next(t);
    int m=strlen(t);
    int n=strlen(s);
    int k=0;
    while(k<min(n,m)&&t[k]==s[k])
    {
        k++;
    }
    extend[0]=k;
    int id=0;
    for(int i=1;i<n;i++)
    {
        int u=i-id;
        if(i+next[u]<extend[id]+id)
        {
            extend[i]=next[u];
        }
        else
        {
            int j=extend[id]+id-i;
            if(j<0) j=0;
            while(j+i<n&&t[j]==s[j+i]) j++;
            extend[i]=j;
            id=i;
        }
    }
}
```

3.4 KMP

```
void get_next(char *s)
{
    int len=strlen(s);
    next[0]=-1;
    for(int i=1;i<len;i++)
    {
        int u=next[i-1];
        while(u>=0&&s[u+1]!=s[i])
        {
            u=next[u];
        }
        if(s[u+1]==s[i])
        {
            u++;
        }
        next[i]=u;
    }
}
int solve(char *s,char *t)
{
    get_next(t);
    int sum=0;
    int u=-1;
    int n=strlen(s),m=strlen(t);
    for(int i=0;i<n;i++){
        while(u>=0&&s[i]!=t[u+1]){
            u=next[u];
        }
        if(s[i]==t[u+1]){
            u++;
        }
        if(u==m-1){
            sum++;
            u=next[u];
        }
    }
    return sum;
}
```

3.5 AC自动机

```
#include<iostream>
#include<cstring>
#include<cstdio>
#include<algorithm>
const int MAX_NODE = 50*10000+10;
const int CHILD_NUM = 26;
struct Aho_Corasick
{
    int trie[MAX_NODE][CHILD_NUM];
    int word[MAX_NODE];
    int fail[MAX_NODE];
    int vis[MAX_NODE];
    int Q[MAX_NODE];
    int sw[128];
    int sz;
    void Initialize(){
        fail[0]=0;
        for(int i=0;i<CHILD_NUM;i++){
            sw[i+(int)'a']=i;
        }
    }
    void Reset(){
        memset(trie[0],0,sizeof(trie[0]));
        word[0]=0;
        memset(vis,0,sizeof(vis));
        memset(word,0,sizeof(word));
        sz=0;
    }
    void Insert(char *s)
    {
        int r=0,len=strlen(s);
        for(int i=0;i<len;i++){
            int idx=sw[s[i]];
            if(!trie[r][idx]){
                trie[r][idx]=++sz;
                for(int j=0;j<CHILD_NUM;j++){
                    trie[sz][j]=0;
                }
                word[sz]=0;fail[sz]=0;vis[sz]=0;
            }
            r=trie[r][idx];
        }
        word[r]++;
    }
    void Build(){
        int front=0,rear=0;
        for(int i=0;i<CHILD_NUM;i++){
            int u=trie[0][i];
            if(u){
                fail[u]=0;
                Q[rear++]=u;
            }
        }
        while(front<rear){
            int u=Q[front++];
            for(int i=0;i<CHILD_NUM;i++){
                int v=trie[u][i];
                if(v){
                    fail[v]=trie[fail[u]][i];
                    Q[rear++]=v;
                }else{
                    trie[u][i]=trie[fail[u]][i];
                }
            }
        }
    }
    int Work(char *s)
    {
        int n=strlen(s);
        int sum=0;
        int r=0;
        for(int i=0;i<n;i++){
            int idx=sw[s[i]];
            r=trie[r][idx];
            int u=r;
            while(u!=0&&!vis[u]){
                sum+=word[u];
                vis[u]=1;
                u=fail[u];
            }
        }
        return sum;
    }
}AC;
char str[60],s[1000010];
int main(void)
{
    int T,n;
    AC.Initialize();
    scanf("%d",&T);
    while(T--){
        AC.Reset();
        scanf("%d",&n);
        for(int i=1;i<=n;i++){
            scanf("%s",str);
            AC.Insert(str);
        }
        AC.Build();
        scanf("%s",s);
        printf("%d\n",AC.Work(s));
    }
    return 0;
}
```

3.6 后缀数组

```
#include<iostream>
#include<cstring>
#include<cstdio>
#include<cmath>
#define lowbit(x) (x&(-x))
using namespace std;
const int N = 10000;
const int CH = 256;//字母表大小
char str[N];
int p[N],s[N],c[N],pn[N],cn[N],height[N],cnt[N];
int val[N];
void init()
{
    int n=strlen(str);
    for(int i=0;i<n;i++){
        s[i]=(int)str[i];
    }
    s[n]=0;
}
void Debug(int a[],int n)
{
    printf("/********************************/\n");
    for(int i=0;i<n;i++)
    {
        printf("%d ",a[i]);
    }
    printf("\n");
    printf("/********************************/\n");
}
void build_sa(int s[],int n,int alphabet)//s[]为字符串转化的数组, n为字符串长度
    加1,alp为字母表大小
{
    memset(cnt,0,sizeof(cnt));
    for(int i=0;i<n;i++)
        ++cnt[s[i]];
    for(int i=1;i<alphabet;i++)
        cnt[i]+=cnt[i-1];
    for(int i=n-1;i>=0;i--)
        p[--cnt[s[i]]]=i;
    c[p[0]]=0;
    int classes=1;
    for(int i=1;i<n;i++)
    {
        if(s[p[i]]!=s[p[i-1]]) ++classes;
        c[p[i]]=classes-1;
    }
    //第一次排序结束
    for(int h=0;(1<<h)<n;h++)
    {
        int m=1<<h;
        for (int i=0; i<n; ++i) {
            pn[i] = p[i] - (1<<h);
            if (pn[i] < 0)  pn[i] += n;
        }
        memset(cnt,0,sizeof(cnt));
        for(int i=0;i<n;i++)
            ++cnt[c[pn[i]]];
        for(int i=1;i<classes;i++)
            cnt[i]+=cnt[i-1];
        for(int i=n-1;i>=0;i--)
            p[--cnt[c[pn[i]]]]=pn[i];
        cn[p[0]]=0;
        classes=1;
        for(int i=1;i<n;i++)
        {
            if(c[p[i]]!=c[p[i-1]]||c[p[i]+m]!=c[p[i-1]+m])
                ++classes;
            cn[p[i]]=classes-1;
        }
        memcpy(c,cn,n*sizeof(int));
        if(classes==n) break;//有所优化
    }
}
void get_lcp(int n,int s[])//大小为n,height从1开始到n为真正有用的值,height[i]表
    示sa[i]与sa[i-1]的最长公共前缀
{
    memset(height,0,sizeof(height));
    for(int i=0;i<n;i++){
        if(c[i]==0) continue;
        int st=max(height[c[i-1]]-1,0);
        int j=i+st,k=p[c[i]-1]+st;
        while(j<n&&k<n&&s[j]==s[k])
        {
            st++;j++;k++;
        }
        height[c[i]]=st;
    }
}
void rmq_init(int n,int height[])
{
    for(int i=1;i<=n;i++)
    {
        val[i]=height[i];
    }
    for(int i=1;i<=n;i++)
    {
        for(int j=i;j<=n;j+=lowbit(j))
        {
            val[j]=min(val[j],height[i]);
        }
    }

}
int get_rmq(int l,int r)
{
    if(l>r) swap(l,r);
    l++;
    int ans=height[r];
    while(r>=l)
    {
        if(r-lowbit(r)+1>=l)
        {
            ans=min(ans,val[r]);
            r-=lowbit(r);
        }
        else
        {
            ans=min(ans,height[r]);
            r--;
        }
    }
    return ans;
}
int main(void)
```

```
{
    int T;
    scanf("%d",&T);
    while(T--)
    {
        scanf("%s",str);
        init();
        int n=strlen(str);
        build_sa(s,n+1,128);
        get_lcp(n,s);
        long long sum=0;
        for(int i=0;i<=n;i++)
        {
            sum+=n-p[i]-height[i];
        }
        cout<<sum<<endl;
        //~ Debug(p,n+1);
        //~ Debug(c,n+1);
    }
    return 0;
}
```

4 图论

4.1 SPFA判负环

如果没有负环的话跑的超慢，解决办法是初始贪心初始化，每个点由周围的点更新一次

```
void spfa(int x)
{
    if(flag) return ;
    instack[x]=1;
    for(int cur=first[x];cur!=-1;cur=edge[cur].next)
    {
        int u=edge[cur].v;
        if (dis[u]>dis[x]+edge[cur].val)
        {
            dis[u]=dis[x]+edge[cur].val;
            if(!instack[u]) spfa(u);
            else{
                flag=1;
                return ;
            }
        }
    }
    instack[x]=0;
}
```

4.2 tarjan

4.2.1 求SCC

```
#include<stdio.h>
#include<string.h>
#define N 10005
#define M 50005
int first[N],next[M],end[M],dfn[N],ins[N],low[N],stack[N],color[N],cnt[N],
    degree[N],top,c,indx;

void tarjan(int u)

{
     int i,v;

     stack[++top]=u;
     ins[u]=1;
     dfn[u]=low[u]=++indx;

     for (i=first[u];i;i=next[i])
     {
         v=end[i];

         if (dfn[v]==0)
         {
            tarjan(v);
            if (low[u]>low[v])
               low[u]=low[v];
         }

         else if (ins[v]==1&&low[u]>dfn[v])
              low[u]=dfn[v];
     }

     if (low[u]==dfn[u])
     {
        c++;

        do
        {
            v=stack[top--];
            color[v]=c;
            cnt[c]++;
            ins[v]=0;
        }
        while (u!=v);
     }
}

void solve()
{
     int i;

     for (i=1;i<=n;i++)
        if (dfn[i]==0)
            tarjan(i);
}
```

4.2.2 求割点、点双联通分支

```
#define N 2000 // 注意缩点后新建的图的顶点数最大可能达到 2*n-1
#define M 600000
using namespace std;
```

```

struct Edge
{
    int u,v;

    Edge(){}
    Edge(int a,int b):u(a),v(b){}
};

int first[N],next[M],end[M];
int cnt_edge;
int dfn[N],low[N],color[N];
int cut[N]; // cut[i]>0表示i是割点，删掉i点后原图将分成cut[i]+1个联通块
Edge stack[M];
int idx,c,b,top;
vector<int>dpt[N]; // 保存每个点双联通分支中的点。注意割点会包含在多个点双联通分支中
int qfir[N],qnex[M],qend[M]; // 保存缩点后的图
int cnt_Qedge;

void addEdge(int u,int v)
{
    end[cnt_edge]=v;
    next[cnt_edge]=first[u];
    first[u]=cnt_edge++;
}

void addQedge(int u,int v)
{
    qend[cnt_Qedge]=v;
    qnex[cnt_Qedge]=qfir[u];
    qfir[u]=cnt_Qedge++;
}

void tarjan(int u)
{
    int i,v,flag;
    Edge e;

    dfn[u]=low[u]=++idx;
    flag=0;

    for (i=first[u];i;i=next[i])
    {
        v=end[i];

        if (dfn[v]==0)
        {
            stack[++top]=Edge(u,v);

            tarjan(v);

            low[u]=min(low[u],low[v]);

            if (dfn[u]<=low[v])
            {
                cut[u]++;
                c++;

                do
                {
                    e=stack[top--];

                    if (color[e.u]!=c)
                    {
                        color[e.u]=c;
                        dpt[c].push_back(e.u);
                    }
                    if (color[e.v]!=c)
                    {
                        color[e.v]=c;
                        dpt[c].push_back(e.v);
                    }

                }while (e.u!=u||e.v!=v);
            }
        }
        else
        {
            low[u]=min(low[u],dfn[v]);
            if (dfn[u]>dfn[v])
                stack[++top]=Edge(u,v);
        }
    }
}

main()
{
    int n,m,i,j,u,v;

    while (scanf("%d%d",&n,&m)!=EOF)
    {
        memset(first,0,sizeof(first));
        memset(qfir,0,sizeof(qfir));
        memset(dfn,0,sizeof(dfn));
        memset(cut,0,sizeof(cut));
        memset(color,0,sizeof(color));
        cnt_edge=1;
        cnt_Qedge=1;
        idx=top=c=0;

        for (i=0;i<N;i++)
            dpt[i].clear();

        for (i=0;i<m;i++)
        {
            scanf("%d%d",&u,&v);
            addEdge(u,v);
            addEdge(v,u);
        }

        for (i=1;i<=n;i++)
            if (dfn[i]==0)
            {
                tarjan(i);
                cut[i]--;
            }

        b=c;

        for (i=1;i<=n;i++)
            if (cut[i])
                color[i]=++c;  // 缩点后新图中割点形成的点的编号>b，联通分支形成的点的
                   编号<=b

        for (i=1;i<=b;i++)
            for (j=0;j<dpt[i].size();j++)
```

```
                {
                    u=dpt[i][j];

                    if (cut[u])
                    {
                        addQedge(color[u],i); // 缩点建图，每个割点形成的点向周围联通分量
                            形成的点双向连边
                        addQedge(i,color[u]);
                    }
                }

        for (i=1;i<=b;i++)
        {
            for (j=0;j<dpt[i].size();j++)
                printf("%d ",dpt[i][j]);
            printf("\n");
        }
    }

    return 0;
}
```

4.2.3 求桥

```
#include<stdio.h>
#include<string.h>
#define N 5005
#define M 20005

typedef struct
{
    int u,v;
}Edge;

Edge bridge[M];
int first[N],next[M],end[M],dfn[N],low[N],degree[N],father[N];
int idx,cnt;

void reset(int n)
{
    int i;

    for (i=0;i<=n;i++)
        father[i]=i;
}

int find(int a)
{
    if (father[a]==a)
        return a;
    else
        return father[a]=find(father[a]);
}

void merge(int a,int b)
{
    int fa=find(a);
    int fb=find(b);

    if (fa!=fb)
        father[fa]=fb;
}

int tarjan(int u,int fu)
```

```
{
    int i,v,flag;

    dfn[u]=low[u]=++idx;
    flag=0;

    for (i=first[u];i;i=next[i])
    {
        v=end[i];

        if (flag==0&&v==fu)     // 处理返祖边的影响
        {
            flag=1;
            continue;
        }

        if (dfn[v]==0)
        {
            tarjan(v,u);

            if (low[u]>low[v])
                low[u]=low[v];

            if (dfn[u]<low[v])      //dfn[u]<low[v]说明u v为桥
            {
                bridge[cnt].u=u;
                bridge[cnt].v=v;
                cnt++;
            }
            else
                merge(u,v);
        }
        else if (low[u]>dfn[v])
            low[u]=dfn[v];
    }
}

main()
{
    int n,m,i,u,v,ans;

    scanf("%d%d",&n,&m);

    memset(first,0,sizeof(first));
    memset(dfn,0,sizeof(dfn));
    memset(degree,0,sizeof(degree));
    reset(n);
    cnt=idx=0;

    for (i=1;i<=m+m;)
    {
        scanf("%d%d",&u,&v);
        end[i]=v;
        next[i]=first[u];
        first[u]=i;
        i++;
        end[i]=u;
        next[i]=first[v];
        first[v]=i;
        i++;
    }

    tarjan(1,0);

    for (i=0;i<cnt;i++)
    // 求最少需要添加多少条边，使原图无桥：缩点后度为1（叶子）的点的个数为ans，则需添
        加(ans+1)/2条边。
    {
        u=bridge[i].u;
        v=bridge[i].v;

        degree[find(u)]++;
        degree[find(v)]++;
    }

    ans=0;

    for (i=1;i<=n;i++)
        if (degree[i]==1)
            ans++;

    printf("%d\n",(ans+1)/2);

    return 0;
}
```

4.2.4 求LCA

```
#include<stdio.h>
#include<string.h>
#define N 1000
#define M 1000000
int father[N],mark[N],res[N],ans[N];
int first[N],next[N],end[N],degree[N],qhead[N],qnext[M],qtail[M],qnum[M];
int find(int a)
{
    if (father[a]==a)
        return a;
    else
        return father[a]=find(father[a]);
}

void merge(int a,int b)
{
    father[b]=a;
}

void LCA(int u)
{
    int i,v,num;

    father[u]=u;

    for (i=first[u];i;i=next[i])
    {
        v=end[i];
        LCA(v);
        merge(u,v);
    }

    mark[u]=1;
    for (i=qhead[u];i;i=qnext[i])
    {
        v=qtail[i];
        num=qnum[i];

```

```
            if (mark[v]==1)
            {
                ans[num]=find(v);
                res[find(v)]++;
            }
        }
}
```

4.3 网络流

4.3.1 EK

```
# include <cstring>
# include <cstdio>
# include <cstdlib>
# include <iostream>
# include <queue>
# define N 20500
# define M 4000500
# define oo 200000000
using namespace std;
struct EDGE
{
    int val,v,next;
}edge[M];
struct DOT
{
    int x,y,c;
}dot[110];
int vis[110],map[110][110];
int first[N],prev[N],curedge[N];
int m,w,n,d,cnt_edge;
void addedge(int u,int v,int c)
{
    edge[cnt_edge].next=first[u];
    edge[cnt_edge].v=v;
    edge[cnt_edge].val=c;
    first[u]=cnt_edge++;

    edge[cnt_edge].next=first[v];
    edge[cnt_edge].v=u;
    edge[cnt_edge].val=0;
    first[v]=cnt_edge++;
}
int sqr(int x)
{
    return x*x;
}
int EK_MaxFlow(int st,int en)
{
    int maxflow=0;
    queue<int>q;
    while (1)
    {
        while (!q.empty())
            q.pop();
        memset(prev,-1,sizeof(prev));
        q.push(st);
        curedge[st]=oo;
        while (!q.empty())
        {
            int x=q.front();
            q.pop();
            for (int cur=first[x];cur!=-1;cur=edge[cur].next)
            {
                int v=edge[cur].v;
                if (prev[v]==-1 && edge[cur].val)
                {
                    prev[v]=x;
                    curedge[v]=cur;
                    q.push(v);
                }
            }
            if (prev[en]!=-1)
                break;
        }
        if (prev[en]==-1)
            return maxflow;
        int minflow=oo;
        for (int v=en;v!=st;v=prev[v]s)
            minflow=min(minflow,edge[curedge[v]].val);
        for (int v=en;v!=st;v=prev[v])
        {
            edge[curedge[v]].val-=minflow;
            edge[curedge[v]^1].val+=minflow;
        }
        maxflow+=minflow;
    }
    return maxflow;
}
int bfs(void)
{
    queue<int>q;
    int x;
    q.push(0);
    while (!q.empty())
    {
        x=q.front();
        q.pop();
        for (int i=1;i<=n+1;i++)
            if (map[x][i] && !vis[i])
            {
                vis[i]=1;
                q.push(i);
            }
    }
    return vis[n+1];
}
int main (void)
{
    cin>>n>>m>>d>>w;
    for (int i=1;i<=n;i++)
        scanf("%d%d%d",&dot[i].x,&dot[i].y,&dot[i].c);
    if (d>=w)
    {
        printf("1\n");
        return 0;
    }
    for (int i=1;i<=n;i++)
    {
        if (dot[i].c==0)
            continue;
        if (dot[i].y<=d)
            map[0][i]=1;
        if (w-dot[i].y<=d)
            map[i][n+1]=1;
        for (int j=i;j<=n;j++)
            if (dot[j].c)
                if (sqr(dot[i].x-dot[j].x)+sqr(dot[i].y-dot[j].y)<=d*d)
                    map[i][j]=map[j][i]=1;
    }
    if (!bfs())
    {
        printf("IMPOSSIBLE\n");
        return 0;
    }
    cnt_edge=0;
    memset(first,-1,sizeof(first));
    int st=0,en=1,tot=1,ans=0,cnt=0;
    for (;cnt<m;cnt+=EK_MaxFlow(st,en))
    {
        ans++;
        for (int i=1;i<=n;i++)
        {
            if (dot[i].c==0)
                continue;
            addedge(i+tot,i+n+tot,dot[i].c);
            if (ans>1)
            {
                for (int j=1;j<=n;j++)
                    if (map[j][i])
                        addedge(tot-n+j,tot+i,dot[i].c);
                if (map[i][n+1])
                    addedge(tot-n+i,en,dot[i].c);
            }
            if (map[0][i])
                addedge(st,i+tot,dot[i].c);
        }
        tot+=n+n;
    }
    cout<<ans<<endl;
    return 0;
}
```

4.3.2 Sap

```
# include <cstring>
# include <cstdlib>
# include <cstdio>
# include <iostream>
using namespace std;
# define M 485000
# define N 20500
# define oo 200000000
struct EDGE
{
    int v,val,next;
}edge[M];
int first[N],dis[N],gap[N],pre[N],cur[N];
int cnt_edge,NPO;
inline void addedge(int u,int v,int c1,int c2)
{
    edge[cnt_edge].next=first[u];
    edge[cnt_edge].v=v;
    edge[cnt_edge].val=c1;
    first[u]=cnt_edge++;

    edge[cnt_edge].next=first[v];
    edge[cnt_edge].v=u;
    edge[cnt_edge].val=c2;
    first[v]=cnt_edge++;
}
int Sap(int st,int en)
{
    memset(dis,0,sizeof(int)*(NPO+1));
    memset(gap,0,sizeof(int)*(NPO+1));
    for (int i=0;i<NPO;i++)
        cur[i]=first[i];
    int u=pre[st]=st,maxflow=0,aug=oo;
    gap[0]=NPO;
    while (dis[st]<NPO)
    {
loop:   for (int &i=cur[u];i!=-1;i=edge[i].next)
        {
            int v=edge[i].v;
            if (edge[i].val && dis[u]==dis[v]+1)
            {
                aug=min(aug,edge[i].val);
                pre[v]=u;
                u=v;
                if (v==en)
                {
                    maxflow+=aug;
                    for (u=pre[u];v!=st;u=pre[u])
                    {
                        edge[cur[u]].val-=aug;
                        edge[cur[u]^1].val+=aug;
                        v=u;
                    }
                    aug=oo;
                }
                goto loop;
            }
        }
        int mindis=NPO;
        for (int i=first[u];i!=-1;i=edge[i].next)
        {
            int v=edge[i].v;
            if (edge[i].val && mindis>dis[v])
            {
                cur[u]=i;
                mindis=dis[v];
            }
        }
        if (--gap[dis[u]]==0)
            break;
        gap[dis[u]=mindis+1]++;
        u=pre[u];
    }
    return maxflow;
}
int main (void)
{
    int n,m,x,y,z;
    cin>>n>>m;
    int st=0,en=n+1;
    NPO=en+1;
    memset(first,-1,sizeof(first));
    cnt_edge=0;
    for (int i=1;i<=n;i++)
    {
        scanf("%d%d",&x,&y);
        addedge(st,i,x,0);
        addedge(i,en,y,0);
    }
    for (int i=1;i<=m;i++)
    {
        scanf("%d%d%d",&x,&y,&z);
        addedge(x,y,z,z);
    }
    cout<<Sap(st,en)<<endl;
    return 0;
```

```
}
```

4.3.3 sap邻接表

```
//memset(first,-1,sizeof(first));
//cnt_edge=0
struct EDGE
{
    int v,val;
}edge[M];
int next[M];
int first[N],d[N],gap[N];
int cnt_edge,NPO,ok;
void addedge(LOL u,LOL v,LOL c)
{
    next[cnt_edge]=first[u];
    edge[cnt_edge].v=v;
    edge[cnt_edge].val=c;
    first[u]=cnt_edge++;

    next[cnt_edge]=first[v];
    edge[cnt_edge].v=u;
    edge[cnt_edge].val=0;
    first[v]=cnt_edge++;
}
LOL MIN(LOL x,LOL y)
{
    return x>y?y:x;
}
LOL relable(LOL x)
{
    LOL mm=bignum;
    for (int i=first[x];i!=-1;i=next[i])
        if (edge[i].val)
            mm=MIN(mm,d[edge[i].v]+1);
    return mm==bignum?NPO:mm;
}
LOL solve(int cur ,int t,LOL min)
{
    if (cur==t && min!=bignum)
        return min;
    for (int i=first[cur];i!=-1;i=next[i])
    {
        int u=edge[i].v;
        if (edge[i].val && d[cur]==d[u]+1)
        {
            LOL p=MIN(solve(u,t,MIN(min,edge[i].val)),min);
            if (p)
            {
                edge[i].val-=p;
                edge[i^1].val+=p;
                return p;
            }
        }
    }
    LOL x=relable(cur);
    gap[x]++;
    if (--gap[d[cur]]==0)
        ok=1;
    d[cur]=x;
    return 0;
}
int maxflow(int s,int t)
{
    int ans=0;
    ok=0;
    memset(gap,0,sizeof(gap));
    memset(d,0,sizeof(d));
    while (d[s]<NPO && !ok)
        ans+=solve(s,t,bignum);
    return ans;
}
```

4.3.4 费用流

```
struct EDGE
{
    int val,v,cost;
}edge[M];
int vis[N],pv[N],pe[N],q[N],dis[N],first[N];
int next[M];
int NPO,cnt_edge;
int MIN(int x,int y)
{
    return x>y?y:x;
}
void addedge(int u,int v,int c,int cost)
{
    next[cnt_edge]=first[u];
    edge[cnt_edge].val=c;
    edge[cnt_edge].v=v;
    edge[cnt_edge].cost=cost;
    first[u]=cnt_edge++;

    next[cnt_edge]=first[v];
    edge[cnt_edge].val=0;
    edge[cnt_edge].v=u;
    edge[cnt_edge].cost=-cost;
    first[v]=cnt_edge++;
}
int mincost (int s,int t)
{
    int x,u,cur;
    int head,tail;
    int flow,cost,min;
    for (flow=0,cost=0;;)
    {
        memset(pv,-1,sizeof(int)*(NPO+10));
        memset(vis,0,sizeof(int)*(NPO+10));
        for (int i=0;i<=NPO;i++)
            dis[i]=bignum;
        dis[s]=0;
        q[1]=s;
        vis[s]=1;
        for (head=0,tail=1;head++<tail;)
        {
            x=q[head%N];
            vis[x]=0;
            for (cur=first[x];cur!=-1;cur=next[cur])
            {
                u=edge[cur].v;
                if (edge[cur].val && dis[x]+edge[cur].cost<dis[u])
                {
                    dis[u]=dis[x]+edge[cur].cost;
                    if (!vis[u])
                    {
                        vis[u]=1;
                        q[(++tail)%N]=u;
                    }
                    pv[u]=x;
                    pe[u]=cur;
                }
            }
        }
        if (pv[t]==-1)
            break;
        for (cur=t,min=bignum;cur!=s;cur=pv[cur])
                min=MIN(min,edge[pe[cur]].val);
        for (cur=t;cur!=s;cur=pv[cur])
        {
            edge[pe[cur]].val-=min;
            edge[pe[cur]^1].val+=min;
        }
        flow+=min;
        cost+=dis[t]*min;
    }
    return cost;
}
```

4.3.5 无源无汇上下界网络流

zoj 2314
主要思想：每一个点流进来的流=流出去的流
对于每一个点i，令
Mi= sum(i点所有流进来的下界流)– sum(i点所有流出去的下界流)
如果Mi大于0，代表此点必须还要流出去Mi的自由流，那么我们从源点连一条Mi的边到该点。
如果Mi小于0，代表此点必须还要流进来Mi的自由流，那么我们从该点连一条Mi的边到汇点。
如果求S到T的最大流，看是否满流(S的相邻边都流满)

```
# include <cstring>
# include <cstdlib>
# include <cstdio>
# include <iostream>
using namespace std;
# define N 400
# define M 500000
# define oo 200000000
struct EDGE
{
    int v,val,next;
}edge[M];
int low[M];
int w[N],first[N],dis[N],gap[N],preV[N],curE[N];
int cnt_edge,NPO;
inline void addedge(int u,int v,int c)
{
    edge[cnt_edge].next=first[u];
    edge[cnt_edge].v=v;
    edge[cnt_edge].val=c;
    first[u]=cnt_edge++;

    edge[cnt_edge].next=first[v];
    edge[cnt_edge].v=u;
    edge[cnt_edge].val=0;
    first[v]=cnt_edge++;
}
int Sap(int st,int en)
{
    memset(dis,0,sizeof(int)*(NPO+1));
    memset(gap,0,sizeof(int)*(NPO+1));
    for (int i=0;i<NPO;i++)
        curE[i]=first[i];
    int u=preV[st]=st,maxflow=0,minflow=oo;
    gap[0]=NPO;
    while (dis[st]<NPO)
    {
        loop:
        for (int &i=curE[u];i!=-1;i=edge[i].next)
        {
            int v=edge[i].v;
            if (edge[i].val && dis[u]==dis[v]+1)
            {
                minflow=min(minflow,edge[i].val);
                preV[v]=u;
                u=v;
                if (v==en)
                {
                    maxflow+=minflow;
                    for (u=preV[u];v!=st;u=preV[u])
                    {
                        edge[curE[u]].val-=minflow;
                        edge[curE[u]^1].val+=minflow;
                        v=u;
                    }
                    minflow=oo;
                }
                goto loop;
            }
        }
        int mindis=NPO;
        for (int i=first[u];i!=-1;i=edge[i].next)
        {
            int v=edge[i].v;
            if (edge[i].val && mindis>dis[v])
            {
                curE[u]=i;
                mindis=dis[v];
            }
        }
        if (--gap[dis[u]]==0)
            break;
        gap[dis[u]=mindis+1]++;
        u=preV[u];
    }
    return maxflow;
}
int main (void)
{
    int t,n,m;
    cin>>t;
    while (t--)
    {
        cin>>n>>m;
        int u,v,l,h;
        cnt_edge=0;
        memset(first,-1,sizeof(first));
        memset(w,0,sizeof(w));
        int st=0,en=n+1,sum=0;
        NPO=en+1;
        for (int i=1;i<=m;i++)
        {
            scanf("%d%d%d%d",&u,&v,&l,&h);
            addedge(u,v,h-l);
            low[i]=l;
            w[v]+=l;
            w[u]-=l;
        }
        for (int i=1;i<=n;i++)
            if (w[i]<0)
            {
                sum-=w[i];
                addedge(i,en,-w[i]);
            }
            else
                addedge(st,i,w[i]);
        if (Sap(st,en)<sum)
            printf("NO\n");
        else
        {
            printf("YES\n");
            for (int i=1;i<=m;i++)
                printf("%d\n",edge[i+i-1].val+low[i]);
        }
    }
    return 0;
}
```

4.3.6 有源有汇上下界网络流

poj_2396
有源有汇上下界网络流
由汇向源连一条（0，oo）的边，下界为0，上界为oo
然后转化成无源无汇上下界网络流问题

```
# include <cstring>
# include <cstdlib>
# include <cstdio>
# include <iostream>
# include <algorithm>
# include <queue>
# include <vector>
# include <map>
# include <cmath>
# include <time.h>
# include <set>
using namespace std;
# define N 400
# define M 500000
# define oo 2000000
struct EDGE
{
    int v,val,next;
}edge[M];
int w[N],first[N],dis[N],gap[N],preV[N],curE[N];
int low[N][N],high[N][N],num[N][N];
int row[N],col[N];
int cnt_edge,NPO,flag;
inline void addedge(int u,int v,int c)
{
    edge[cnt_edge].next=first[u];
    edge[cnt_edge].v=v;
    edge[cnt_edge].val=c;
    first[u]=cnt_edge++;

    edge[cnt_edge].next=first[v];
    edge[cnt_edge].v=u;
    edge[cnt_edge].val=0;
    first[v]=cnt_edge++;
}
```

```
int Sap(int st,int en)
{
    memset(dis,0,sizeof(int)*(NPO+1));
    memset(gap,0,sizeof(int)*(NPO+1));
    for (int i=0;i<NPO;i++)
        curE[i]=first[i];
    int u=preV[st]=st,maxflow=0,minflow=oo;
    gap[0]=NPO;
    while (dis[st]<NPO)
    {
        loop:
        for (int &i=curE[u];i!=-1;i=edge[i].next)
        {
            int v=edge[i].v;
            if (edge[i].val && dis[u]==dis[v]+1)
            {
                minflow=min(minflow,edge[i].val);
                preV[v]=u;
                u=v;
                if (v==en)
                {
                    maxflow+=minflow;
                    for (u=preV[u];v!=st;u=preV[u])
                    {
                        edge[curE[u]].val-=minflow;
                        edge[curE[u]^1].val+=minflow;
                        v=u;
                    }
                    minflow=oo;
                }
                goto loop;
            }
        }
        int mindis=NPO;
        for (int i=first[u];i!=-1;i=edge[i].next)
        {
            int v=edge[i].v;
            if (edge[i].val && mindis>dis[v])
            {
                curE[u]=i;
                mindis=dis[v];
            }
        }
        if (--gap[dis[u]]==0)
            break;
        gap[dis[u]=mindis+1]++;
        u=preV[u];
    }
    return maxflow;
}
void update(int x,int y,char ch,int z)
{
    if (ch=='=')
    {
        if (!(low[x][y]<=z && high[x][y]>=z))
            flag=0;
        low[x][y]=high[x][y]=z;
    }
    else
    if (ch=='<')
    {
        high[x][y]=min(high[x][y],z-1);
        if (low[x][y]>high[x][y])
```

```
            flag=0;
    }
    else
    if (ch=='>')
    {
        low[x][y]=max(low[x][y],z+1);
        if (low[x][y]>high[x][y])
            flag=0;
    }
}
int main (void)
{
    int t,n,m,ys=0;
    //~ freopen("a.in","r",stdin);
    //~ freopen("a.out","w",stdout);
    cin>>t;
    while (t--)
    {
        int sum1=0,sum2=0;
        cin>>n>>m;
        for (int i=1;i<=n;i++)
        {
            cin>>row[i];
            sum1+=row[i];
        }
        for (int i=1;i<=m;i++)
        {
            cin>>col[i];
            sum2+=col[i];
        }
        for (int i=1;i<=n;i++)
            for (int j=1;j<=m;j++)
                low[i][j]=0,high[i][j]=oo;
        int q,x,y,z;
        flag=1;
        char s[10];
        cin>>q;
        while (q--)
        {
            cin>>x>>y>>s>>z;
            if (x==0 && y==0)
            {
                for (int i=1;i<=n;i++)
                    for (int j=1;j<=m;j++)
                        update(i,j,s[0],z);
            }
            else
            if (x==0)
            {
                for (int i=1;i<=n;i++)
                    update(i,y,s[0],z);
            }
            else
            if (y==0)
            {
                for (int i=1;i<=m;i++)
                    update(x,i,s[0],z);
            }
            else
                update(x,y,s[0],z);
        }
        if (ys)
            cout<<endl;
        else ys=1;
        if (sum1!=sum2 || !flag)
            cout<<"IMPOSSIBLE"<<endl;
        else
        {
            int st=0,en=n+m+1,cnt=0;
            int ss=en+1,ee=en+2;
            NPO=en+3;
            cnt_edge=0;
            memset(first,-1,sizeof(first));
            memset(w,0,sizeof(w));
            for (int i=1;i<=n;i++)
                for (int j=1;j<=m;j++)
                {
                    addedge(i,n+j,high[i][j]-low[i][j]);
                    w[n+j]+=low[i][j];
                    w[i]-=low[i][j];
                    num[i][j]=++cnt;
                }
            for (int i=1;i<=n;i++)
            {
                addedge(st,i,0);
                w[i]+=row[i];
                w[st]-=row[i];
            }
            for (int i=1;i<=m;i++)
            {
                addedge(i+n,en,0);
                w[en]+=col[i];
                w[i+n]-=col[i];
            }
            //add a oo edge from en to st
            addedge(en,st,oo);

            int sum=0;
            for (int i=0;i<=n+m+1;i++)
            {
                if (w[i]>0)
                {
                    addedge(ss,i,w[i]);
                    sum+=w[i];
                }
                else
                    addedge(i,ee,-w[i]);
            }
            if (Sap(ss,ee)!=sum)
                cout<<"IMPOSSIBLE"<<endl;
            else
            {
                for (int i=1;i<=n;i++)
                    for (int j=1;j<=m;j++)
                        printf("%d%c",low[i][j]+edge[num[i][j]+num[i][j]-1].
                            val,j==m?'\n':' ');
            }
        }
    }
    return 0;
}
```

4.4 2-sat

2-sat模版
poj_3683
如果i,j矛盾，则addedge(i,j+n),addedge(j,i+n)
i,j+n矛盾 addedge(i,j),addedge(j+n,i+n)
i+n,j矛盾 addedge(i+n,j+n),addedge(j,i)
i+n,j+n矛盾 addedge(i+n,j),addedge(j+n,i)
pd()调用2-sat判断是否合法
调用initialize()初始化
print_solution()输出方案， 如果ans[i]==1则选择左点，否则选择右点

```
# include <cstring>
# include <iostream>
# include <cstdio>
# include <cstdlib>
# define N 2500
# define M 2000000
using namespace std;
struct EDGE
{
    int u,v,next;
}edge[M],edge2[M];
struct node
{
    int s,e,len;
}a[N];
int first[N],DFN[N],divi[N],LOW[N],instack[N],stack[N];
int cf[N],choice[N],ind[N],q[N],ans[N],first2[N];
int cnt,cnt_edge,cnt_edge2,top,nowt;
inline void addedge(int u,int v)
{
    edge[cnt_edge].u=u;
    edge[cnt_edge].v=v;
    edge[cnt_edge].next=first[u];
    first[u]=cnt_edge++;
}
inline void addedge2(int u,int v)
{
    edge2[cnt_edge2].v=v;
    edge2[cnt_edge2].next=first2[u];
    first2[u]=cnt_edge2++;
}
void tarjan(int u)
{
    int v;
    DFN[u]=LOW[u]=++nowt;
    stack[++top]=u;
    instack[u]=1;
    for (int cur=first[u];cur!=-1;cur=edge[cur].next)
    {
        v=edge[cur].v;
        if (!DFN[v])
        {
            tarjan(v);
            LOW[u]=min(LOW[v],LOW[u]);
        }
        else
            if (instack[v])
                LOW[u]=min(LOW[u],DFN[v]);
    }
    if (DFN[u]==LOW[u])
    {
        ++cnt;
        do
        {
            v=stack[top--];
            instack[v]=0;
            divi[v]=cnt;
        }
        while (u!=v);
    }
}
void initialize(void)
{
    memset(first,-1,sizeof(first));
    memset(first2,-1,sizeof(first2));
    memset(instack,0,sizeof(instack));
    memset(divi,0,sizeof(divi));
    memset(DFN,0,sizeof(DFN));
    memset(LOW,0,sizeof(LOW));
    memset(choice,0,sizeof(choice));
    memset(ind,0,sizeof(ind));
    memset(ans,0,sizeof(ans));
    nowt=cnt=top=0;
    cnt_edge=cnt_edge2=0;
}
bool pd(int n)
{
    for (int i=1;i<=2*n;i++)
        if (!DFN[i])
            tarjan(i);
    for (int i=1;i<=n;i++)
        if (divi[i]==divi[i+n])
            return false;
    return true;
}
void print_solution(int n)
{
    for (int i=1;i<=n;i++)
    {
        cf[divi[i]]=divi[i+n];
        cf[divi[i+n]]=divi[i];
    }
    for (int i=0;i<cnt_edge;i++)
        if (divi[edge[i].u]!=divi[edge[i].v])// 反向连边
        {
            addedge2(divi[edge[i].v],divi[edge[i].u]);
            ind[divi[edge[i].u]]++;
        }
    int head=0,tail=0,x;
    for (int i=1;i<=cnt;i++)//topsort
        if (!ind[i])
            q[++tail]=i;
    while (head++<tail)
    {
        x=q[head];
        if (!choice[x])
        {
            choice[x]=1;
            choice[cf[x]]=-1;
        }
        for (int cur=first2[x];cur!=-1;cur=edge2[cur].next)
            if (--ind[edge2[cur].v]==0)
                q[++tail]=edge2[cur].v;
    }
    for (int i=1;i<=n;i++)
```

```
        if (choice[divi[i]]==1)
            ans[i]=1;
}
bool intersect(int x1,int y1,int x2,int y2)
{
    if ((x2<y1 && y2>x1) || (x1<y2 && y1>x2))
        return true;
    return false;
}
int main (void)
{
    int n,x,y;
    scanf("%d",&n);
    initialize();
    for (int i=1;i<=n;i++)
    {
        scanf("%d:%d",&x,&y);
        a[i].s=x*60+y;
        scanf("%d:%d",&x,&y);
        a[i].e=x*60+y;
        scanf("%d",&a[i].len);
        if (a[i].s+a[i].len>a[i].e)
        {
            printf("NO\n");
            return 0;
        }
    }
    for (int i=1;i<=n;i++)
        for (int j=i+1;j<=n;j++)
        {
            if (intersect(a[i].s,a[i].s+a[i].len,a[j].s,a[j].s+a[j].len))
            {
                addedge(i,j+n);
                addedge(j,i+n);
            }
            if (intersect(a[i].s,a[i].s+a[i].len,a[j].e-a[j].len,a[j].e))
            {
                addedge(i,j);
                addedge(j+n,i+n);
            }
            if (intersect(a[i].e-a[i].len,a[i].e,a[j].s,a[j].s+a[j].len))
            {
                addedge(i+n,j+n);
                addedge(j,i);
            }
            if (intersect(a[i].e-a[i].len,a[i].e,a[j].e-a[j].len,a[j].e))
            {
                addedge(i+n,j);
                addedge(j+n,i);
            }
        }
    if (!pd(n))
        printf("NO\n");
    else
    {
        printf("YES\n");
        print_solution(n);
        for (int i=1;i<=n;i++)
        {
            if (ans[i])
            {
                x=a[i].s;
                y=a[i].s+a[i].len;
            }
            else
            {
                x=a[i].e-a[i].len;
                y=a[i].e;
            }
            printf("%02d:%02d %02d:%02d\n",x/60,x%60,y/60,y%60);
        }
    }
    return 0;
}
```

4.5 A*求最短路

```
# include <cstring>
# include <queue>
# include <cstdlib>
# include <cstdio>
# include <iostream>
# define oo 200000000
# define N 1050
# define M 105000
using namespace std;
struct EDGE
{
    int v,w,next;
}E[M],E1[M];
int first[N],first1[N],dist[N],vis[N],deg[N];
struct Po
{
    int v,w;
    bool operator <(const Po &a)const
    {
        return w+dist[v]>a.w+dist[a.v];
    }
}Point;

inline void addedge(EDGE E[],int first[],int i,int u,int v,int c)
{
    E[i].next=first[u];
    E[i].v=v;
    E[i].w=c;
    first[u]=i;
}
void Dij(EDGE E[],int first[],int s,int n)
{
    priority_queue <Po> Q;
    Po tem;
    int u,now;
    memset(vis,0,sizeof(vis));
    for (int i=1;i<=n;i++)
        dist[i]=oo;
    dist[s]=0;
    tem.v=s;
    tem.w=0;
    Q.push(tem);
    while (!Q.empty())
    {
        now=Q.top().v;
        Q.pop();
        if (vis[now])
            continue;
        vis[now]=1;
        for (int cur=first[now];cur!=-1;cur=E[cur].next)
        {
            u=E[cur].v;
            if (dist[u]>dist[now]+E[cur].w)
            {
                dist[u]=dist[now]+E[cur].w;
                tem.v=u;
                tem.w=dist[u];
                Q.push(tem);
            }
        }
    }
}
int A_star(EDGE E[],int first[],int s,int t,int k)
{
    priority_queue <Po> Q;
    Po tem,now;
    memset(deg,0,sizeof(deg));
    tem.v=s;
    tem.w=0;
    Q.push(tem);
    while (!Q.empty())
    {
        tem=Q.top();
        Q.pop();
        deg[tem.v]++;
        if (deg[tem.v]==k)
            return tem.w+dist[tem.v];
        for (int cur=first[tem.v];cur!=-1;cur=E[cur].next)
        {
            now.v=E[cur].v;
            now.w=tem.w+E[cur].w;
            if (deg[now.v]<k)
                Q.push(now);
        }
    }
    return -1;
}
int main (void)
{
    int n,m,s,t,k;
    int x,y,z;
    cin>>n>>m;
    memset(first,-1,sizeof(first));
    memset(first1,-1,sizeof(first1));
    for (int i=0;i<m;i++)
    {
        scanf("%d%d%d",&x,&y,&z);
        addedge(E,first,i,x,y,z);
        addedge(E1,first1,i,y,x,z);
    }
    cin>>s>>t>>k;
    if (s==t)
        k++;
    Dij(E1,first1,t,n);
    cout<<A_star(E,first,s,t,k)<<endl;
    return 0;
}
```

4.6 带花树

求一般图最大匹配数的带花树算法

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#define N 1000
#define M 800000
using namespace std;

int first[N],next[M],end[M];
int cnt_edge;
int n;

char mark[N],blossom[N];
int que[N],pre[N],base[N],match[N]; //match数组记录匹配点

void addEdge(int u,int v) // 无向图求最大匹配需要双向加边
{
    end[cnt_edge]=v;
    next[cnt_edge]=first[u];
    first[u]=cnt_edge++;
}

void argument(int u)
{
    int k,v;

    while (~u)
    {
        v=pre[u];
        k=match[v];
        match[u]=v;
        match[v]=u;
        u=k;
    }

}

void changeblossom(int b,int u)
{
    int v;

    while (base[u]!=b)
    {
        v=match[u];
        blossom[base[v]]=blossom[base[u]]=1;
        u=pre[v];

        if (base[u]!=b)
            pre[u]=v;

    }
}

int findbase(int u,int v)
{
    char inp[N];

    memset(inp,0,sizeof(inp));

    while (1)
    {
        inp[u]=1;
        if (match[u]==-1)
            break;

        u=base[pre[match[u]]];
    }

    while (!inp[v])
        v=base[pre[match[v]]];

    return v;

}

int contract(int u,int v)
{
    int b;

    memset(blossom,0,sizeof(blossom));
    b=findbase(base[u],base[v]);
    changeblossom(b,u);
    changeblossom(b,v);

    if (base[u]!=b)
        pre[u]=v;
    if (base[v]!=b)
        pre[v]=u;

    return b;

}

int bfs(int p)
{
    int i,j,head,tail,u,v,b;

    memset(pre,-1,sizeof(pre));
    memset(mark,0,sizeof(mark));

    for (i=0;i<n;i++)
        base[i]=i;

    que[0]=p;
    mark[p]=1;
    head=0,tail=1;

    while (head<tail)
    {
        u=que[head++];

        for (i=first[u];i;i=next[i])
        {
            v=end[i];

            if (base[u]!=base[v]&&v!=match[u])
            {
                if (v==p||(match[v]!=-1&&pre[match[v]]!=-1))
                {
                    b=contract(u,v);

                    for (j=0;j<n;j++)
                        if (blossom[base[j]])
                        {
                            base[j]=b;

                            if (mark[j]==0)
                            {
                                mark[j]=1;
                                que[tail++]=j;
                            }
                        }
                }
                else if (pre[v]==-1)
                {
                    pre[v]=u;

                    if (match[v]==-1)
                    {
                        argument(v);
                        return 1;
                    }
                    else
                    {
                        que[tail++]=match[v];
                        mark[match[v]]=1;
                    }
                }
            }
        }
    }

    return 0;
}


int max_match() // 返回最大匹配数
{
    int i,res=0;

    memset(match,-1,sizeof(match));

    for (i=0;i<n;i++)
        if (match[i]==-1)
            res+=bfs(i);

    return res;
}

//ural1099: n个点（编号范围0～n-1）的无向图，求最大匹配数，并输出最大匹配方案。单组数据，读入边以EOF结束。
int main()
{
    int i,u,v,ans;

    memset(first,0,sizeof(first));
    cnt_edge=1;

    scanf("%d",&n);
    while (scanf("%d%d",&u,&v)!=EOF)
    {
        u--,v--;
        addEdge(u,v);
        addEdge(v,u);
    }

    ans=max_match();

    printf("%d\n",ans*2);

    for (i=0;i<n;i++)
        if (match[i]!=-1&&i<match[i])
            printf("%d %d\n",i+1,match[i]+1);

    return 0;
}
```

4.7 KM算法

```
#include<cstdio>
#include<cmath>
#include<cstring>
#define N 105
#define oo 0x7fffffff
using namespace std;
struct Point
{
        int x;
        int y;
};

char str[N][N],mx[N],my[N];
int r,c,n,nx,ny,map[N][N],lx[N],ly[N],slack[N],res[N];

int find(int u)
{
    int i,v,t;

    mx[u]=1;

    for (i=1;i<=n;i++)
        if (my[i]==0)
        {
            v=i;
            t=lx[u]+ly[v]-map[u][v];

            if (t==0)
            {
               my[v]=1;
               if (res[v]==0||find(res[v]))
               {
                    res[v]=u;
                    return 1;
               }
            }
            else
            {
                if (slack[v]>t)
                   slack[v]=t;
            }
        }

    return 0;
}

void KM()
{


     int i,j,d;

     memset(res,0,sizeof(res));
     memset(lx,0,sizeof(lx));
     memset(ly,0,sizeof(ly));

     for (i=1;i<=n;i++)
         for (j=1;j<=n;j++)
            if (lx[i]<map[i][j])
               lx[i]=map[i][j];

     for (i=1;i<=n;i++)
     {
         for (j=1;j<=n;j++)
             slack[j]=oo;

         while (1)
         {
               memset(mx,0,sizeof(mx));
               memset(my,0,sizeof(my));

               if (find(i))
                  break;
               else
               {
                   d=oo;
                   for (j=1;j<=n;j++)
                       if (my[j]==0&&d>slack[j])
                          d=slack[j];

                   for (j=1;j<=n;j++)
                       if (mx[j])
                          lx[j]-=d;

                   for (j=1;j<=n;j++)
                       if (my[j])
                          ly[j]+=d;
                       else
                          slack[j]-=d;
               }
         }
     }
}

main()
{
      int i,j,r,c,ans;
      Point px[N],py[N];
      while (scanf("%d%d",&r,&c),r||c)
      {
            for (i=0;i<r;i++)
                scanf("%s",str[i]);

            n=nx=ny=0;
            for (i=0;i<r;i++)
                for (j=0;j<c;j++)
                    if (str[i][j]=='m')
                    {
                       px[++nx].x=i+1;
                       px[nx].y=j+1;
                    }
                    else if (str[i][j]=='H')
                    {
                       py[++ny].x=i+1;
                       py[ny].y=j+1;
                    }
            n=nx;

            for (i=1;i<=n;i++)
                for (j=1;j<=n;j++)
                    map[i][j]=-(abs(px[i].x-py[j].x)+abs(px[i].y-py[j].y));

            KM();

            for (ans=0,i=1;i<=n;i++)
                ans+=map[res[i]][i];

            printf("%d\n",-ans);
        }
        return 0;
}
```

4.8 欧拉回路通路

当图中存在欧拉回路（通路）时，使用USACO算法能够找到一个可行方案

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<stack>
#define N 20000
#define M 110000
using namespace std;

int first[N],next[M],end[M],vis[M];
int cnt_edge;
stack<int>ans;

void addEdge(int u,int v)
{
    end[cnt_edge]=v;
    next[cnt_edge]=first[u];
    first[u]=cnt_edge++;
}

// 无向图需要双向加边，欧拉通路需要选好起点，最后逆序（弹栈）输出
void USACO(int u)
{
    int i,v;

    for (i=first[u];i;i=next[i])
    {
        v=end[i];

        if (vis[i]==0)
        {
            vis[i]=1;
            USACO(v);
        }
    }

    ans.push(u);
}

//poj2230
int main()
{
    int n,m,u,v,i,j;

    memset(first,0,sizeof(first));
    memset(vis,0,sizeof(vis));
    cnt_edge=1;

    scanf("%d%d",&n,&m);
    for (i=0;i<m;i++)
    {
        scanf("%d%d",&u,&v);
        addEdge(u,v);
        addEdge(v,u);
    }

    USACO(1);

    while (!ans.empty())
    {
        printf("%d\n",ans.top());
```

```
            ans.pop();
    }

    return 0;
}
```

4.9 汉密尔顿回路

哈密顿回路 Dirac's Theorem:对于一张顶点个数为n的无向图，若$n >= 3$且每个顶点的度不小于$(n+1)/2$，那么一定存在哈密顿回路。 以下为满足此性质的图中求哈密顿回路的算法，时间复杂度为O(n^2)。

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#define N 200
using namespace std;

struct node
{
    int id;
    node *next;

    node (int u=0,node *x=NULL)
    {
        id=u;
        next=x;
    }
};

node *hs,*he,*cur; //hs为链的头指针，he为尾指针
char map[N][N],mark[N]; //map[u][v]为1则顶点u、v关联
int n,m,size;

void init()
{
    memset(map,0,sizeof(map));
    memset(mark,0,sizeof(mark));
    hs=he=cur=NULL;
}

void reverse(node *v)
{
    if (v==he)
    {
        cur=he;
        return;
    }

    reverse(v->next);
    cur->next=v;
    cur=v;
}

void change(node *v1,node *v2)
{
    v1->next=he;
    reverse(v2);
}

void solve()
{
    int i;

    hs=new node(1,NULL);
    mark[1]=1;

    for (i=2;i<=n;i++)
        if (map[i][1])
        {
            he=new node(i,NULL);
            hs->next=he;
            mark[i]=1;
            break;
        }

    size=2;
    while (1)
    {
        for (i=1;i<=n;i++)
        {
            if (mark[i])
                continue;

            if (map[i][hs->id]&&mark[i]==0)
            {
                mark[i]=1;
                size++;
                hs=new node(i,hs);
            }

            if (map[i][he->id]&&mark[i]==0)
            {
                mark[i]=1;
                size++;
                node *tmp=he;
                he=new node(i,NULL);
                tmp->next=he;
            }
        }

        if (map[hs->id][he->id]==0)
        {
            node *tmp=hs->next;

            while (tmp->next!=he&&tmp!=he)
            {
                if (map[hs->id][tmp->next->id]==1&&map[he->id][tmp->id]==1)
                {
                    change(tmp,tmp->next);
                    he=cur;
                    he->next=NULL;
                    cur=NULL;
                    break;
                }
                tmp=tmp->next;
            }
        }

        if (size==n)
            break;

        for (i=1;i<=n;i++)
        {
            if (mark[i])
                continue;

            node *tmp=hs;

            while (tmp!=NULL)
            {
                if (map[i][tmp->id])
                {
                    he->next=hs;
                    hs=tmp->next;
                    he=new node(i,NULL);
                    tmp->next=he;
                    mark[i]=1;
                    size++;
                    break;
                }
                tmp=tmp->next;
            }

            if (map[hs->id][he->id]==0)
                break;
        }
    }
}

void print()
{
    node *tmp=hs;

    printf("%d",tmp->id);
    tmp=tmp->next;

    while (tmp!=NULL)
    {
        printf(" %d",tmp->id);
        tmp=tmp->next;
    }

    printf("\n");
}

//hdu4337: 求满足Dirac's Theorem中条件的图 的哈密顿回路的方案
int main()
{
    int i,u,v;

    while (scanf("%d%d",&n,&m)!=EOF)
    {
        init();

        for (i=0;i<m;i++)
        {
            scanf("%d%d",&u,&v);
            map[u][v]=map[v][u]=1;
        }

        solve();
        print();
    }

    return 0;
}
```

4.10 最小树形图

4.10.1 邻接表

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<cmath>
#define oo 1000000000
#define N 1005
#define M 2000100
using namespace std;

struct Point
{
    int x,y,z;
};

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

Edge edge[M];
Point pt[N];
int pre[N],in[N],color[N],mark[N];
int cnt_edge;

void addEdge(int u,int v,int w)
{
    edge[cnt_edge].u=u;
    edge[cnt_edge].v=v;
    edge[cnt_edge].w=w;
    cnt_edge++;
}


int dis(Point a,Point b)
{
    return abs(a.x-b.x)+abs(a.y-b.y)+abs(a.z-b.z);
}

int ZhuLiu(int root,int n,int m)
{
    int i,j,u,v,w,cnt;
    int res=0;

    while (1)
    {
        for (i=1;i<=n;i++)
            in[i]=oo;
        for (i=0;i<m;i++)
        {
            u=edge[i].u;
            v=edge[i].v;
            w=edge[i].w;

            if (in[v]>w&&u!=v)
            {
                in[v]=w;
                pre[v]=u;
            }
        }

        for (i=1;i<=n;i++)
            if (i!=root&&in[i]==oo)
                return -1;

        memset(mark,0,sizeof(mark));
        memset(color,0,sizeof(color));
        mark[root]=1;
        in[root]=0;
        cnt=0;

        for (i=1;i<=n;i++)
        {
            res+=in[i];
            v=i;
            while (mark[v]!=i&&color[v]==0&&v!=root)
            {
                mark[v]=i;
                v=pre[v];
            }

            if (v!=root&&color[v]==0)
            {
                cnt++;
                for (u=pre[v];u!=v;u=pre[u])
                    color[u]=cnt;
                color[v]=cnt;
            }
        }

        if (cnt==0)
            break;
        for (i=1;i<=n;i++)
            if (color[i]==0)
                color[i]=++cnt;
        for (i=0;i<m;i++)
        {
            v=edge[i].v;
            edge[i].u=color[edge[i].u];
            edge[i].v=color[edge[i].v];
            if (edge[i].u!=edge[i].v)
                edge[i].w-=in[v];
        }

        n=cnt;
        root=color[root];
    }

    return res;
}

int main()
{
    int n,m,i,u,v,w,mx,my,mz;

    while (scanf("%d%d%d%d",&n,&mx,&my,&mz),n||mx||my||mz)
    {
        cnt_edge=0;
        for (i=1;i<=n;i++)
            scanf("%d%d%d",&pt[i].x,&pt[i].y,&pt[i].z);

        for (i=1;i<=n;i++)
        {
            u=i;
```

```
                scanf("%d",&m);
                while (m--)
                {
                    scanf("%d",&v);
                    if (u==v)
                        continue;
                    w=dis(pt[u],pt[v])*my;
                    if (pt[u].z<pt[v].z)
                        w+=mz;

                    addEdge(u,v,w);
                }
                addEdge(n+1,i,pt[i].z*mx);
            }


            printf("%d\n",ZhuLiu(n+1,n+1,cnt_edge));
        }

    return 0;
}
```

4.10.2 邻接矩阵

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<cmath>
#include<algorithm>
#define oo 1e15
#define eps 1e-6
#define N 105
#define M 10010
using namespace std;

struct Point
{
    double x,y;
};
Point pt[N];

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

double map[N][N];
int pre[N];
char mark[N],flag[N];
int n;
void dfs(int u)
{
    int i;

    mark[u]=1;
    for (i=1;i<=n;i++)
        if (mark[i]==0&&map[u][i]<oo)
            dfs(i);
}

int connect(int root)
{
    int i;

    memset(mark,0,sizeof(mark));
    dfs(root);

    for (i=1;i<=n;i++)
        if (mark[i]==0)
            return 0;
    return 1;
}

double ZhuLiu(int root)
{
    int i,j,k;
    double res=0;

    memset(flag,0,sizeof(flag));

    while (1)
    {
        for (i=1;i<=n;i++)
        {
            if (flag[i]||i==root)
                continue;

            pre[i]=i;
            map[i][i]=oo;
            for (j=1;j<=n;j++)
                if (!flag[j]&&map[j][i]<map[pre[i]][i])
                    pre[i]=j;
        }

        for (i=1;i<=n;i++)
        {
            if (flag[i]||i==root)
                continue;

            memset(mark,0,sizeof(mark));
            mark[root]=1;
            j=i;

            do
            {
                mark[j]=1;
                j=pre[j];
            }while (!mark[j]);

            if (j==root)
                continue;

            i=j;
            res+=map[pre[i]][i];
            for (j=pre[i];j!=i;j=pre[j])
            {
                flag[j]=1;
                res+=map[pre[j]][j];
            }

            for (j=1;j<=n;j++)
                if (!flag[j]&&map[j][i]<oo)
                    map[j][i]-=map[pre[i]][i];

            for (j=pre[i];j!=i;j=pre[j])
                for (k=1;k<=n;k++)
                    if (!flag[k])
                    {
                        map[i][k]=min(map[i][k],map[j][k]);
                        if (map[k][j]<oo)
                            map[k][i]=min(map[k][i],map[k][j]-map[pre[j]][j
                                ]);
                    }
                break;
            }

        if (i>n)
        {
            for (j=1;j<=n;j++)
                if (!flag[j]&&j!=root)
                    res+=map[pre[j]][j];
            break;
        }
    }

    return res;
}

int main()
{
    int m,i,j,u,v;

    while (scanf("%d%d",&n,&m)!=EOF)
    {
        for (i=0;i<=n;i++)
            for (j=0;j<=n;j++)
                map[i][j]=oo;

        for (i=1;i<=n;i++)
            scanf("%lf%lf",&pt[i].x,&pt[i].y);
        for (i=0;i<m;i++)
        {
            scanf("%d%d",&u,&v);
            map[u][v]=min(map[u][v],dis(pt[u],pt[v]));
        }

        if (connect(1))
            printf("%.2f\n",ZhuLiu(1));
        else
            printf("poor snoopy\n");
    }

    return 0;
}
```

4.11 稳定婚姻

解题思路：此题为一道典型的稳定婚姻问题，如果男A与女C匹配，男B与女D 匹配，但是A更喜欢D，D更喜欢A，那么这个婚姻就不是稳定的。稳定婚姻问题的算法步骤大致如下：(男士优先)

1.男士先选择自己最爱的人去求婚，如果有多个男士的最爱相同，那么女士就选择更爱的那位男士，那么其他的男士在这次求婚中失败。

2.上次求婚失败的男士再选择自己次爱的女士进行求婚，如果这位女士没有匹配男士，那么这两个人就进行匹配，如果这位女士有匹配的男士，但是如果这位女士更喜欢这位正在求婚的男士，那么这位女士就可以抛弃原有的男士而与这位她更爱的男士匹配。而原有的男士在这次匹配中失败。

3.上次求婚失败的男士(包括被女士抛弃的男士)再次选择自己次爱的女士求婚，知道所有的男士与女士全部匹配则结束。 此算法被证明必定存在解，刚才所写的步骤是基于男士优先的。如果是女士向男士求婚，那么就是基于女士优先的，算法步骤和上面的基本相同，只要将男女调换就可以了。

```
#include<iostream>
#include<cstring>
#include<cstdio>
#include<map>
#include<string>
using namespace std;
const int N = 1010;
map<string,int> gmp,bmp;
string s;
int b[N][N],g[N][N],match[N],vis[N],v[N][N];
string boy[N],girl[N];
int n;
int num;
void find(int x)
{
    int t=0;
    for(int i=1;i<=n;i++){
        if(!v[i][x]&&(t==0||g[x][t]>g[x][i])){
            t=i;
        }
    }
    v[t][x]=1;
    if(!match[t]){
        match[t]=x;vis[x]=1;num++;
    }else{
        if(b[t][x]<b[t][match[t]]){
            vis[match[t]]=0;
            match[t]=x;vis[x]=1;
        }
    }
}
void solve()
{
    num=0;
    memset(vis,0,sizeof(vis));
    memset(v,0,sizeof(v));
    memset(match,0,sizeof(match));
    while(num<n){
        for(int i=1;i<=n;i++){
            if(!vis[i]) find(i);
        }
    }
}
int main(void)
{
    while(scanf("%d",&n)!=EOF)
    {
        bmp.clear(),gmp.clear();
        memset(g,0,sizeof(g));
        memset(b,0,sizeof(b));
        int k=0;
        for(int i=1;i<=n;i++){
            cin>>s;
            boy[i]=s;
            bmp[s]=i;
            for(int j=1;j<=n;j++){
                cin>>s;
                if(gmp[s]==0){
                    gmp[s]=++k;
                    girl[k]=s;
                }
                b[i][gmp[s]]=j;
            }
        }
        for(int i=1;i<=n;i++){
            cin>>s;
            int t=gmp[s];
            for(int j=1;j<=n;j++){
                cin>>s;
                g[t][bmp[s]]=j;
            }
        }
        solve();
        for(int i=1;i<=n;i++){
            cout<<boy[i]<<" "<<girl[match[i]]<<endl;
        }
    }
    return 0;
}
```

4.12 最优比率生成树

poj_2728
dis[i][j]是i到j的距离
cost[i][j]修建¡i,j¿这条边的花费。
要求总花费比总距离最小（单位距离花费最小）

```
# include <cstdio>
# include <iostream>
# include <cstring>
# include <cmath>
# define N 1050
# define bignum 200000000
using namespace std;
double x[N],y[N],z[N],val[N],dis[N][N],cost[N][N],map[N][N];
int vis[N];
inline double prim(double x,int n)
{
    for (int i=1;i<n;i++)
        for (int j=i+1;j<=n;j++)
            map[i][j]=map[j][i]=cost[i][j]-x*dis[i][j];
    memset(vis,0,sizeof(int)*(n+10));
    for (int i=2;i<=n;i++)
        val[i]=map[1][i];
    vis[1]=1;
    double ans=0,Min;
    int now;
    for (int p=1;p<n;p++)
    {
        Min = bignum;
        for (int i=2;i<=n;i++)
            if (!vis[i] && val[i]<Min)
            {
                now=i;
                Min=val[i];
            }
        vis[now]=1;
        ans+=Min;
        for (int i=2;i<=n;i++)
            if (!vis[i])
                val[i]=min(val[i],map[now][i]);
    }
    return ans;
}
int main (void)
{
    int n;
    while (scanf("%d",&n)!=EOF && n)
    {
        double Min=bignum;
        for (int i=1;i<=n;i++)
        {
            scanf("%lf%lf%lf",&x[i],&y[i],&z[i]);
            for (int j=1;j<i;j++)
            {
                cost[i][j]=cost[j][i]=fabs(z[i]-z[j]);
                dis[i][j]=dis[j][i]=sqrt((x[i]-x[j])*(x[i]-x[j])+(y[i]-y[j])
                   *(y[i]-y[j]));
                if ( fabs(dis[i][j])>1e-2 && cost[i][j]/dis[i][j]<Min)
                    Min=cost[i][j]/dis[i][j];
            }
        }
        double left=Min,right,mid,sum1=0,sum2=0;
        for (int i=2;i<=n;i++)
        {
            sum1+=cost[i][1];
            sum2+=dis[i][1];
        }
        right=sum1/sum2;
        while (right-left>1e-4)
        {
            mid=(right+left)/2;
            if (prim(mid,n)>0)
                left=mid;
            else right=mid;
        }
        printf("%.3f\n",left);
    }
    return 0;
}
```

5 DP

5.1 插头DP

```
# include <cstring>
# include <cstdlib>
# include <cstdio>
# include <iostream>
# define HashMod 1999997
# define LEN 700000
# define N 15
using namespace std;
int HHash[HashMod];
int state[2][LEN],tot[2];
long long sum[2][LEN];
int a[N][N];
int n,m,nn,mm;
inline void Hash_in(int k,int s,long long data)
{
    int hashpos=s%HashMod;
    while (HHash[hashpos])
    {
        if (state[k][HHash[hashpos]]==s)
        {
            sum[k][HHash[hashpos]]+=data;
            return;
        }
        hashpos++;
        if (hashpos==HashMod) hashpos=0;
    }
    HHash[hashpos]=++tot[k];
    state[k][tot[k]]=s;
    sum[k][tot[k]]=data;
}
long long solve (void)
{
    int k=0,jz[N];
    int s,temps,p,q,bracket,w;
    long long data,ans=0;
    for (int i=0;i<N;i++)
        jz[i]=i<<1;
    tot[0]=1;
    state[0][1]=0;
    sum[0][1]=1;
    for (int i=1;i<=n;i++)
    {
        for (int j=1;j<=m;j++)
        {
            k^=1;
            tot[k]=0;
            memset(HHash,0,sizeof(HHash));
            memset(state[k],0,sizeof(state[k]));
            memset(sum[k],0,sizeof(sum[k]));
            for (int now=1;now<=tot[k^1];now++)
            {
                s=state[k^1][now];
                data=sum[k^1][now];
                p=(s>>jz[j-1])%4;
                q=(s>>jz[j])%4;
                if (!a[i][j])
                {
                    if (p==0 && q==0)
                        Hash_in(k,s,data);
                    continue;
                }
                if (p==2 && q==1)
                {
                    temps=s-2*(1<<jz[j-1])-(1<<jz[j]);
                    Hash_in(k,temps,data);
                    continue;
                }
                if (p==1 && q==2)
                {
                    if (i==nn && j==mm)
                        ans+=data;
                    continue;
                }
                if (p==1 && q==1)
                {
                    bracket=1;
                    temps=s-(1<<jz[j-1])-(1<<jz[j]);
                    for (int x=j+1;x<=m;x++)
                    {
                        w=(s>>jz[x])%4;
                        if (w==1)
                            bracket++;
                        if (w==2)
                            bracket--;
                        if (bracket==0)
                        {
                            temps=temps-(1<<jz[x]);
                            break;
                        }
                    }
                    Hash_in(k,temps,data);
                    continue;
                }
                if (p==2 && q==2)
                {
                    bracket=1;
                    temps=s-2*(1<<jz[j-1])-2*(1<<jz[j]);
                    for (int x=j-2;x>=0;x--)
                    {
                        w=(s>>jz[x])%4;
                        if (w==1)
                            bracket--;
                        if (w==2)
                            bracket++;
                        if (bracket==0)
                        {
                            temps=temps+(1<<jz[x]);
                            break;
                        }
                    }
                    Hash_in(k,temps,data);
                    continue;
                }
                if (p==0 && q==0)
                {
                    if (a[i][j+1] && a[i+1][j])
                    {
                        temps=s+(1<<jz[j-1])+2*(1<<jz[j]);
                        Hash_in(k,temps,data);
                    }
                    continue;
                }
                if (p==0 && q>0)
                {
                    if (a[i][j+1])
                        Hash_in(k,s,data);
                    if (a[i+1][j])
                    {
                        temps=s-q*(1<<jz[j])+q*(1<<jz[j-1]);
                        Hash_in(k,temps,data);
                    }
                }
                if (p>0 && q==0)
                {
                    if (a[i+1][j])
                        Hash_in(k,s,data);
                    if (a[i][j+1])
                    {
                        temps=s-p*(1<<jz[j-1])+p*(1<<jz[j]);
                        Hash_in(k,temps,data);
                    }
                    continue;
                }
            }
        }
        for (int now=1;now<=tot[k];now++)
            state[k][now]<<=2;
    }
    return ans;
}
int main (void)
{
    char ss[50];
    memset(a,0,sizeof(a));
    cin>>n>>m;
    for (int i=1;i<=n;i++)
    {
        scanf("%s",ss+1);
        for (int j=1;j<=m;j++)
        {
            a[i][j]=ss[j]=='.';
            if (a[i][j])
            {
                nn=i;
                mm=j;
            }
        }
    }
    cout<<solve()<<endl;
    return 0;
}
```

5.2 数位DP

CF55D

```
#include<iostream>
#include<cstring>
#include<cstdio>
using namespace std;
#ifdef WINDOWS
#define LOL __int64
#else
#define LOL long long
#endif
LOL dp[50][2600][50];
int c[2600],g[2600][30];
int v[50];
int b[30],a[30];
int gcd(int a,int b)
{
    if(b==0) return a;
    return gcd(b,a%b);
}
int lcm(int a,int b)
{
    if(a>b) swap(a,b);
    if(a==0) return b;
    return a*b/gcd(a,b);
}
LOL dfs(int l,int pre,int lcm,bool z)
{
    if(l==0) return pre%lcm==0;
    if(!z&&dp[l][pre][c[lcm]]!=-1) return dp[l][pre][c[lcm]];
    LOL ans=0;
    int u=z?a[l]:9;
    for(int i=0;i<=u;i++)
    {
        int npre=(pre*10+i)%2520,nlcm=g[lcm][i];
        ans+=dfs(l-1,npre,nlcm,i==u&&z);
    }
    if(!z)
    {
        dp[l][pre][c[lcm]]=ans;
    }
    return ans;
}
LOL solve(LOL x)
{
    int n=0;
    while(x!=0)
    {
        a[++n]=x%10;
        x/=10;
    }
    return dfs(n,0,1,1);
}
int main(void)
{
    memset(dp,0,sizeof(dp));
    b[0]=1;
    for(int i=1;i<=20;i++)
    {
        b[i]=(b[i-1]*10)%2520;
    }
    for(int i=1,r=-1;i<=2520;i++)
    {
        if(2520%i==0)
        {
            r++;
            v[r]=i;
        }
        c[i]=r;
    }
    for(int j=0;j<10;++j){
         for(int i=1;i<=2520;++i)
             g[i][j]=j?i*j/gcd(i,j):i;
         }
    memset(dp,-1,sizeof(dp));
    int T;
    LOL a,b;
    cin>>T;
    while(T--)
    {
        cin>>a>>b;
        cout<<solve(b)-solve(a-1)<<endl;
    }
    return 0;
}
```

5.3 最大平均子段和

```
# include <cstdio>
# include <cstring>
# include <cmath>
# define N 1060000
using namespace std ;
__int64 sum [ N ] ,q[ N ];
int GetInt()
{
    char ch=getchar();
    while(ch<'0'||ch>'9')
      ch=getchar();
    int num=0;
    while(ch>='0'&&ch<='9')
    {
        num=num*10+ch-'0';
        ch=getchar();
    }
    return num;
}
double MAX (double x , double y)
{
    return x>y?x:y;
}
int main ( void )
{
    __int64 x1,y1,x2,y2;
    int head , tail ;
    int n , k ,x;
    while (scanf("%d%d",&n,&k)!=EOF)
    {
        for (int i=1;i<=n;i++)
        {
            x=GetInt();
            sum[i]=sum[i-1]+x;
        }
        tail = 0 ;
        head = 1 ;
        double ans = 0 ;
        for (int i=k;i<=n;i++)
        {
            int now = i-k;
            while (head<tail)
            {
                x1=now-q[tail];
                y1=sum[now]-sum[q[tail]];
                x2=q[tail]-q[tail-1];
                y2=sum[q[tail]]-sum[q[tail-1]];
                if (x1*y2-x2*y1>=0)
                    tail--;
                else break;
            }
            q[++tail]=now;
            while (head<tail)
            {
                x1=i-q[head+1];
                y1=sum[i]-sum[q[head+1]];
                x2=q[head+1]-q[head];
                y2=sum[q[head+1]]-sum[q[head]];
                if (x1*y2-x2*y1<=0)
                    head++;
                else break ;
            }
            ans = MAX (ans,double(sum[i]-sum[q[head]])/double (i-q[head]));
        }
        printf("%.2lf\n",ans);
    }
    return 0 ;
}
```

5.4 斯坦纳树

```
# include <cstring>
# include <cstdio>
# include <cstdlib>
# include <iostream>
using namespace std;
# define oo 200000000
# define N 55
int dp[2000][N],val[2000];
int f[N][N],num[N],l[N],n,k;
inline int lowbit(int x)
{
    return x&(-x);
}
int judge(int x)
{
    int cnt1=0,cnt2=0;
    for (int i=0;i<n;i++)
    {
        if (num[i]==-1) continue;
        if (x&(1<<num[i]))
        {
            if (num[i]<k) cnt1++;
            else cnt2++;
        }
    }
    return (cnt1==cnt2);
}
int main (void)
{
    int t,m,u,v,c;
    scanf("%d",&t);
    while (t--)
    {
        scanf("%d%d%d",&n,&m,&k);
        for (int i=0;i<n;i++)
            for (int j=0;j<n;j++)
                if (i!=j)
                    f[i][j]=oo;
                else f[i][j]=0;
        for (int i=1;i<=m;i++)
        {
            scanf("%d%d%d",&u,&v,&c);
            u--,v--;
            f[u][v]=min(f[u][v],c);
            f[v][u]=min(f[v][u],c);
        }
        for (int p=0;p<n;p++)
            for (int i=0;i<n;i++)
                for (int j=0;j<n;j++)
                    f[i][j]=min(f[i][j],f[i][p]+f[p][j]);
        memset(num,-1,sizeof(num));
        memset(dp,-1,sizeof(dp));
        int tot=0;
        for (int i=0;i<k;i++)
            num[i]=tot++;
        for (int i=n-k;i<n;i++)
            num[i]=tot++;
        for (int i=0;i<n;i++)
        {
            int s=0;
            if (num[i]!=-1)
```

```
                s=(1<<num[i]);
            dp[s][i]=0;
        }
        int all=(1<<(k<<1));
        for (int i=1;i<all;i++)
            val[i]=oo;
        for (int i=1;i<all;i++)
        {
            for (int j=0;j<n;j++)
            {
                for (int s=(i-1)&i;s;s=(s-1)&i)
                {
                    if (dp[s][j]==-1 || dp[i^s][j]==-1) continue;
                    if (dp[i][j]==-1 || dp[i][j]>dp[s][j]+dp[i^s][j])
                        dp[i][j]=dp[s][j]+dp[i^s][j];
                }
                if (dp[i][j]==-1) continue;
                val[i]=min(val[i],dp[i][j]);
                for (int q=0;q<n;q++)
                    if (f[j][q]<oo)
                        if (dp[i][q]==-1 || dp[i][q]>dp[i][j]+f[j][q])
                            dp[i][q]=dp[i][j]+f[j][q];
            }
        }
        for (int i=0;i<all;i++)
            if (judge(i))
                for (int j=0;j<i;j++)
                    if (judge(j) && ((i&j)==j))
                        val[i]=min(val[i],val[j]+val[i^j]);
        if (val[all-1]<oo)
            printf("%d\n",val[all-1]);
        else printf("No solution\n");
    }
    return 0;
}
```

5.5 四边形不等式

状态转移方程形如$m(i,j) = m(i,k-1) + m(k,j) + w(i,j)$,假设$a \le b < c \le d$,如果满足$w(a,c) + w(b,d) <= w(b,c) + w(a,d)$,那么称w满足四边形不等式。若w满足四边形不等式，则m也满足四边形不等式,其决策s(i,j)满足：$s(i,j-1) \le s(i,j) \le s(i+1,j)$

hdu3516 Consider a two-dimensional space with a set of points $(x_i,\ y_i)$ that satisfy $x_i < x_j$ and $y_i > y_j$for all $i < j$. We want to have them all connected by a directed tree whose edges go toward either right (x positive) or upward (y positive). Find a tree connecting all given points with the shortest total length of edges.

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#define N 1010
#define oo 0x3fffffff
using namespace std;

int dp[N][N],s[N][N],x[N],y[N];

int main()
{
    int n,i,j,k;

    while (scanf("%d",&n)!=EOF)
    {
        for (i=0;i<n;i++)
            scanf("%d%d",&x[i],&y[i]);

        if (n==1)
        {
            printf("0\n");
            continue;
        }

        for (j=1;j<n;j++)
        {
            dp[j-1][j]=x[j]-x[j-1]+y[j-1]-y[j];
            s[j-1][j]=j-1;

            for (i=j-2;i>=0;i--)
            {
                dp[i][j]=oo;

                for (k=s[i][j-1];k<=s[i+1][j];k++)
                    if (dp[i][j]>dp[i][k]+dp[k+1][j]+x[k+1]-x[i]+y[k]-y[j])
                    {
                        dp[i][j]=dp[i][k]+dp[k+1][j]+x[k+1]-x[i]+y[k]-y[j];
                        s[i][j]=k;
                    }
            }
        }

        printf("%d\n",dp[0][n-1]);
    }

    return 0;
}
```

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#define N 301
#define M 32
#define oo 0x3fffffff
using namespace std;
```

```

int dp[M][N],s[M][N],w[N][N],p[N];

int main()
{
    int n,m,i,j,k;

    scanf("%d%d",&n,&m);

    for (i=1;i<=n;i++)
        scanf("%d",&p[i]);

    for (i=1;i<=n;i++)
    {
        w[i][i]=0;
        for (j=i+1;j<=n;j++)
            w[i][j]=w[i][j-1]+p[j]-p[(i+j)/2];
    }

    memset(dp,0,sizeof(dp));

    for (i=0;i<=n;i++)
    {
        dp[min(i,m+1)][i]=0;
        s[min(i,m+1)][i]=i;
    }

    for (i=1;i<=n;i++)
    {
        dp[0][i]=oo;
        for (j=min(i-1,m);j>0;j--)
        {
            dp[j][i]=oo;
            for (k=s[j][i-1];k<=s[j+1][i];k++)
                if (dp[j][i]>dp[j-1][k-1]+w[k][i])
                {
                    dp[j][i]=dp[j-1][k-1]+w[k][i];
                    s[j][i]=k;
                }
        }
    }

    printf("%d\n",dp[m][n]);

    return 0;
}
```

6 数学

6.1 数学结论

当$b \geq \phi(c)$时:

$$a^b = a^{b\%\phi(c)+\phi(c)} \pmod c$$

6.2 数论基础

```
#include <cstdio>
#include <cmath>
#include <cstring>
#include <iostream>
#define M 9901
using namespace std;
typedef __int64 typec;
///teoy's number theory template
///functions
/**************************************
gcd
**************************************/
typec gcd(typec a, typec b)
{
    if(b==0) return a;
    return gcd(b,a%b);
}
/**************************************
Extend_GCD
**************************************/
typec extendGCD(typec a, typec b, typec& x, typec& y)
{
    if(!b) return x = 1, y = 0, a;
    typec res = extendGCD(b, a % b, x, y), tmp = x;
    x = y, y = tmp - (a / b) * y;
    return res;
}
/**************************************求对的逆元
ap
**************************************/
typec inverse(typec a, typec p)
{
    typec x, y;
    y = extendGCD(a, p, x, y);
    return x < 0 ? x += p : x;
}
/**************************************
abss for abs();
kgcd for quick gcd;
**************************************/

typec abss(typec a)
{
    if(a<0) return -a;
    return a;
}
typec kgcd(typec a,typec b)
{
    if(a==0) return b;
    if(b==0) return a;
    if(!(a&1)&&!(b&1)) return kgcd(a>>1,b>>1)<<1;
    else if(!(b&1)) return kgcd(a,b>>1);
    else if(!(a&1)) return kgcd(a>>1,b);
    else return kgcd(abss(a-b),min(a,b));
}

/**************************************
for x^k
**************************************/
typec power(typec x,typec k)
{
    typec ans=1;
    while(k)
    {
        if(k&1) ans*=x;
        x*=x;
        k>>=1;
    }
    return ans;
}
/***************************************
for (x^k)%mod
***************************************/
typec powermod(typec x,typec k,typec mod)
{
    typec ans=1;
    while(k)
    {
        if(k&1)
        {
            ans=(ans*x)%mod;
        }
        x=(x*x)%mod;
        k>>=1;
    }
    return ans;
}
/***************************************
prime table
O(n)
prime[0] for the primes number in the PRIMERANGE
prime[i] for i th prime number
***************************************/
const int PRIMERANGE = 100000;
int prime[PRIMERANGE + 1];
int getPrime()
{
    memset (prime, 0, sizeof (int) * (PRIMERANGE + 1));
    for (int i = 2; i <= PRIMERANGE; i++)
    {
        if (!prime[i]) prime[++prime[0]] = i;
        for (int j = 1; j <= prime[0] && prime[j] <= PRIMERANGE / i; j++)
        {
            prime[prime[j]*i] = 1;
            if (i % prime[j] == 0) break;
        }
    }
    return prime[0];
}
/***************************************
IsPrime table
IsPrime[i]==true if(i is not a prime)
Pmaxn is the range of numbers;
***************************************/
const int Pmaxn=20000000;
bool IsPrime[Pmaxn];
void Isprime()
{
    memset(IsPrime,false,sizeof(IsPrime));
    for(int i=2;i<=Pmaxn;i++)
    {
        if(!IsPrime[i])
        {
            for(int j=i;j<=Pmaxn;j++)
            {
                IsPrime[i*j]=true;
            }
        }
    }
}
/*************************************
euler function
*************************************/
typec euler(typec x)
{
    typec res=x;
    for(int i=2;i*i<(typec)(x*1.0)+1;i++)
    {
        if(x%i==0)
        {
            res=res/i*(i-1);
            while(x%i==0) x/=i;
        }
    }
    if(x>1) res=res/x*(x-1);
    return res;
}
///you should init the prime table before
int factor[100][3], facCnt;
int getFactors(int x)
{
    facCnt = 0;
    int tmp = x;
    for(int i = 1; prime[i] <= tmp / prime[i]; i++)
    {
        factor[facCnt][1] = 1, factor[facCnt][2] = 0;
        if(tmp % prime[i] == 0)
            factor[facCnt][0] = prime[i];
        while(tmp % prime[i] == 0)
            factor[facCnt][2]++, factor[facCnt][1] *= prime[i], tmp /= prime
               [i];
        if(factor[facCnt][1] > 1) facCnt++;
    }
    if(tmp != 1)
        factor[facCnt][0] = tmp, factor[facCnt][1] = tmp, factor[facCnt
           ++][2] = 1;
    return facCnt;
}
```

6.3 筛法求素数

1. Pri数组中的素数是递增的,当i能整除Pri[j]，那么i*Pri[j+1]这个合数肯定被Pri[j]乘以某个数筛掉。 fir[i]代表i的最小素因子，可以几乎线性的分解i. 2. 线性求1 n之间所有数的欧拉函数:

```
    if ((i/fir[i])%fir[i]==0)
        ouler[i]=ouler[i/fir[i]]*fir[i];
    else ouler[i]=ouler[i/fir[i]]*(fir[i]-1);
```

3. 线性求1 n之间所有数约数个数 ei表示n的第i个质因数的个数.

```
    if (i%Pri[j]==0)
    {
        divnum[i*Pri[j]]=divsum[i]/(e[i]+1)*(e[i]+2); //最小素因子次数加1
        e[i*Pri[j]]=e[i]+1;
    }
    else
    {
        divnum[i*Pri[j]]=divnum[i]*2;  //满足积性函数条件
        e[i*Pri[j]]=1;
    }
```

```
# define RANGE 100000
int init(void)
{
    int cnt=0,x;
    for (int i=2;i<=RANGE;i++)
    {
        if (!Pri[i])
        {
            fir[i]=Pri[++cnt]=i;
            pos[i]=cnt;
        }
        for (int j=1;j<=cnt;j++)
        {
            x=i*Pri[j];
            if (x>RANGE) break;
            Pri[x]=1;
            fir[x]=Pri[j];
            if (i%Pri[j]==0) break;
        }
    }
    return cnt;
}
```

6.4 同余方程

```
# include <cstring>
# include <cstdio>
# include <cstdlib>
# include <iostream>
using namespace std;
typedef long long LOL;
LOL extend_gcd(LOL a,LOL b,LOL &x,LOL &y)
{
    if (!b)
    {
        x=1,y=0;
        return a;
    }
    LOL tem,r;
    r=extend_gcd(b,a%b,x,y);
    tem=x;
    x=y;
    y=tem-a/b*y;
    return r;
}
int main (void)
{
    int n;
    while (scanf("%d",&n)!=EOF)
    {
        LOL r=0,a=1,r1,a1,x,y,tem,t;
        int flag=0;
        for (int i=1;i<=n;i++)
        {
            scanf("%lld%lld",&a1,&r1);
            if (flag) continue;
            tem=extend_gcd(a,a1,x,y);
            if ((r1-r)%tem)
                flag=1;
            else
            {
                t=a1/tem;

                x=((r1-r)/tem*x%t+t)%t;
                r+=x*a;
                a*=(a1/tem);
                r=(r%a+a)%a;
            }
        }
        if (!flag)
            printf("%lld\n",r);
        else printf("-1\n");
    }
    return 0;
}
```

6.5 高次同余方程

poj 3243 hdu 2815 poj 2417
这个是求扩展离散对数问题。$X^Y \bmod Z = K$,给出X,Z,K,求Y。
当Z时素数的时候直接用baby-step 算法即可。
方程$a^X = b \pmod c$，可以进行一系列的转化。假设

$$d = \gcd(a, c)$$

由

$$a^{x-1} * a = b \pmod c$$

知道a^{x-1}要存在必须满足$\gcd(a,c) \mid b$,如果满足这个条件，那么我们可以在方程两边同时除以d,方程是不变的。因为

$$a^x = b + k * c$$

再除以公约数d,得到方程

$$a^{x-1} * \frac{a}{d} = \frac{b}{d} + k * \frac{c}{d}$$

假设我们除了k次，那么方程转化为

$$a^{x-k} * \frac{a^k}{d^k} = \frac{b}{d^k} + k * \frac{c}{d^k}$$

令$d = \frac{a^k}{d^k}$,$b' = \frac{b}{d^k}$,$c' = \frac{c}{d^k}$,$x' = x - k$,方程转化为:

$$a^{x'} * d = b' \pmod{c'}$$

得到:

$$a^{x'} = b' * d^{-1} \pmod{c'}$$

现在直接用baby-step解方程

$$a^{x'} = b' * d^{-1} \pmod{c'}$$

即可。注意到$x = x' + k$，如果存在x小于k的解，那么x'小于0，但是baby-step是不会求负的次数的，所以需要先枚举一下是否存在小于k的解，由于输入的数据不会超过10^9的，假设k不超过50进行枚举即可了。

```
#include <iostream>
#include <stdio.h>
#include <cmath>
using namespace std;
typedef long long LOL;
const int maxn = 65535;
struct Hashh
{
    int a,b,next;
}Hash[maxn<<1];

int flag[maxn+100];//注意要赋初值0
int top,idx;//注意要赋初值maxn

void ins(int a,int b)
{
    int k=b&maxn;
    if(flag[k]!=idx)//第b&maxn个槽为空
    {
        flag[k]=idx;
        Hash[k].a=a;
        Hash[k].b=b;
        Hash[k].next=-1;
        return ;
    }
    //第b&maxn个槽不为空
    while(Hash[k].next!=-1)// 到链表的最后一个
    {
        if(Hash[k].b==b) return;//若b已经存在，返回
        k=Hash[k].next;
    }
    Hash[k].next=++top;
    Hash[top].next=-1;
    Hash[top].a=a;
    Hash[top].b=b;
}

int find(int b)
{
    int k=b&maxn;
    if(flag[k]!=idx) return -1;//为空
    while(k!=-1)
    {
        if(Hash[k].b==b)
           return Hash[k].a;
        k=Hash[k].next;
    }
    return -1;
}

int gcd(int a,int b)
{
    if(b==0)return a;
    return gcd(b,a%b);
}

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

int exmod(LOL a,int b,int c)
{
    LOL ret=1%c;
    a%=c;
    while(b)
    {
        if(b&1) ret=ret*a%c;
        a=a*a%c;
        b>>=1;
    }
    return ret;
}

int invmod(int a,int b,int n)
{
    int x,y,e;
    exgcd(a,n,x,y);
    e=(LOL)x*b%n;
    return e<0?e+n:e;
}

int babystep(int a,int b,int c)
{
    top=maxn;idx++;
    LOL buf=1%c,K;
    LOL D=buf;
    int tmp,w,d=0;
    for(int i=0;i<=100;i++)
    {

        if(buf==b) return i;
        buf=buf*a%c;
    }
    while((tmp=gcd(a,c))!=1)
    {
        if(b%tmp) return -1;
        ++d;
        c/=tmp;
        b/=tmp;
        D=D*a/tmp%c;
    }
    int m=(int)ceil(sqrt((double)(c-1)));
    buf=1%c;
    for(int i=0;i<=m;i++)
    {
        ins(i,buf);
        buf=buf*a%c;
    }
    K=exmod((LOL)a,m,c);
    for(int i=0;i<=m;i++)
    {
        tmp=invmod((int)D,b,c);
        w=find(tmp);
        if(tmp>=0 && (w!=-1))
           return i*m+w+d;
        D=(D*K%c+c)%c;
    }
    return -1;
}


int main()
{
    int a,b,c,tmp;
    // a^X = b % c
    while(scanf("%d%d%d",&c,&a,&b)!=EOF && (a||b||c))
    {
        //~ if(b>=c)
        //~ {
            //~ printf("no solution\n");
            //~ continue;
        //~ }
        tmp=babystep(a,b,c);
        if(tmp<0) printf("no solution\n");
        else printf("%d\n",tmp);
    }
    return 0;
}
```

6.6 高斯消元

6.6.1 高斯消元解实数方程

```
int Gauss(int n, int m, double a[][N*N])
{
    int i, j, r, c, pvt;
    double maxp;
    for (r=0, c=0; r<n && c<m; ++r, ++c)
    {
        for (maxp=0, i=r; i < n; ++i)
            if (fabs(a[i][c])>fabs(maxp)) maxp = a[pvt=i][c];
        if (sgn(maxp)==0)
        {
            r--;
            continue;
        }
        if (pvt != r)
            for (j = r; j <= m; ++j) swap(a[r][j], a[pvt][j]);
        for (j = c+1; j <= m; ++j)
        {
            a[r][j] /= maxp;
            for (i = r+1; i < n; ++i)
                a[i][j] -= a[i][c]*a[r][j];
        }
    }
    for (i = r; i < n; ++i)
        if (sgn(a[i][m])) return -1;
    if (r < m) return m-r;
    for (i = m-1; i >= 0; --i)
        for (j = i+1; j < m; ++j)
            a[i][m] -= a[j][m]*a[i][j];
    return 0;
}
```

6.6.2 高斯消元解异或方程组

高斯消元解异或方程+枚举变元

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<vector>
#define PB push_back
#define MP make_pair
using namespace std;
const int M = 16;
const int maxn=M*M;
typedef pair<int,int> PII;
char s[M][M];
int num[4];
int dx[4][8];
int dy[4][8];
int N;
int a[maxn][maxn],b[maxn][maxn],c[maxn],d[maxn];
int n,m;
int get(int x,int y)
{
    return x*m+y;
}
void build(int op)
{
    memset(a,0,sizeof(a));
    for(int i=0;i<n;i++){
        for(int j=0;j<m;j++){
                int u=get(i,j);
                if(s[i][j]=='1'){
                    a[u][N]=1;
                }else{
                    a[u][N]=0;
                }
                for(int k=0;k<num[op];k++){
                    int tx=i+dx[op][k];
                    int ty=j+dy[op][k];
                    if(tx<0||tx>=n||ty<0||ty>=m) continue;
                    int v=get(tx,ty);
                    a[v][u]=1;
                }
            }
        }
}
void debug(int a[maxn][maxn])
{
    for(int i=0;i<N;i++){
        for(int j=0;j<=N;j++){
            printf("%d ",a[i][j]);
        }
        printf("\n");
    }
}
int gauss()
{
    int col,row;
    vector<int>vec;
    vector<PII>use;
    int ans=n*m+1;
    for(col=0,row=0;col<N&&row<N;col++){
        int mark=row;
        for(int i=row+1;i<N;i++){
            if(a[i][col]){
                mark=i;
            }
        }
        for(int i=0;i<N+1;i++){
            swap(a[row][i],a[mark][i]);
        }
        if(!a[row][col]){
            vec.PB(col);
            continue;
        }
        use.PB(MP(row,col));
        for(int i=row+1;i<N;i++){
            if(!a[i][col]) continue;
            for(int j=0;j<N+1;j++){
                a[i][j]^=a[row][j];
            }
        }
        row++;
    }
    //debug(a);
    for(int i=row;i<N;i++){
        if(a[i][N]){
            return n*m+1;
        }
    }
    int sz=vec.size();
    for(int i=0;i<(1<<sz);i++){
        for(int j=0;j<N;j++){
            for(int k=0;k<N+1;k++){
                b[j][k]=a[j][k];
            }
        }
        memset(d,0,sizeof(d));
        int cnt=0;
        for(int j=0;j<sz;j++){
            int u;
            if(i&(1<<j)){
                u=1;
            }else u=0;
            d[vec[j]]=u;cnt+=d[vec[j]];
        }
        for(int j=row-1;j>=0;j--){
            int y=b[j][N];
            for(int k=use[j].second+1;k<N;k++){
                y^=(b[j][k]&&d[k]);
            }
            d[use[j].second]=y;
            cnt+=d[use[j].second];
        }
        ans=min(ans,cnt);
    }
    return ans;
}
```

6.6.3 模意义下求行列式的值

```
#include<iostream>
#include<cstring>
#include<cstdio>
using namespace std;
const int N = 210;
typedef long long LOL;
LOL a[N][N];
LOL solve(int n,LOL p)
{
    LOL ans=1;
    for(int i=1;i<=n;i++){
        for(int j=i+1;j<=n;j++){
            while(a[j][i]!=0){
                LOL t=a[i][i]/a[j][i];
                for(int k=i;k<=n;k++){
                    a[i][k]=a[i][k]-a[j][k]*t;
                    a[i][k]%=p;
                }
                for(int k=i;k<=n;k++){
                    swap(a[i][k],a[j][k]);
                }
                ans=-ans;
            }
        }
        if(a[i][i]==0) return 0;
        else ans=((ans*a[i][i])%p+p)%p;
    }
    return ans;
}
int main(void)
{
    int n;
    LOL p;
    while(scanf("%d%lld",&n,&p)!=EOF){
        for(int i=1;i<=n;i++){
            for(int j=1;j<=n;j++){
                scanf("%lld",&a[i][j]);
                a[i][j]%=p;
            }
        }
        LOL ans=solve(n,p);
        ans=(ans%p+p)%p;
        cout<<ans<<endl;
    }
    return 0;
}
```

6.7 POLYA

6.7.1 概念

定义 1 置换：设X是一个有限集，取X为包含前n个正整数的集合*1,2,...,n*，X的每个置换i_1,i_2,...i_n可视为X到其自身定义的一个一对一的函数

$$f : X \to X$$

其中$f(1) = i_1$,$f(2) = i_2$,...,$f(n) = i_n$

定义 2 置换群：如果S_n中的置换的非空子集G满足如下三条性质，则定义它为X的一个置换群：
1) （合成运算的封闭性）对G中所有的置换f与g，$f \circ g$也属于G。
2) （单位元）S_n中的恒等置换τ属于G。
3) （逆元的封闭性）对G中的每一个置换f，它的逆f^{-1}也属于G。

定义 3 着色等价：令G是作用在集合X上的一个置换群，通常取X为前n个正整数的集合。令C是X的一个着色集合。设c_1,c_2是C中的两种着色，如果G中存在一个置换f，使得

$$f * c_1 = c_2$$

则称c_1（在G的作用下）等价于c_2。

定义 4 稳定核：使着色c保持不变的所有置换的集合$G(c)$称为c的稳定核。

定理 1 对于每一种着色c，c的稳定核$G(c)$是一个置换群，而且对G中任意置换f与g，$g * c = f * c$当且仅当$f^{-1} \circ g$属于$G(c)$。

证明：
1）因为f和g都使c保持不变，则$(g \circ f)(c) = c$，满足合成运算的封闭性。
2) 显然单位元τ使所有着色不变。
3) 如果f使得c不变，那么f^{-1}也使得c不变，于是$G(c)$具有对逆元的封闭性。 所以$G(c)$是一个置换群。
　假设$f * c = g * c$,可得

$$(f^{-1} \circ g) * c = f^{-1} * (g * c) = f^{-1} * (f * c) = (f^{-1} \circ f) * c = \tau * c = c$$

6.7.2 题目

POJ1286 题目大意：用红蓝绿三种颜色去染n元环，问方案数。
解法：旋转同构$\sum_{i=1}^{n} 3^{gcd(n,i)}$翻转同构，如果$n$为奇数，$3^{n/2+1}$,如果$n$为偶数，$(n/2) * 3^{n/2} + (n/2) * 3^{n/2+1}$
POJ2409 题目大意：用c种颜色染n元环，问方案数。
解法：旋转同构$\sum_{i=1}^{n} c^{gcd(n,i)}$翻转同构，如果$n$为奇数，$c^{n/2+1}$,如果$n$为偶数，$(n/2) * c^{n/2} + (n/2) * c^{n/2+1}$
POJ2154 题目大意：用n种颜色染n元环，$1 \le n \le 1000000000$，并且有3500组数据。
解法：解法和上一个相同，但是因为n过大，枚举n的复杂度会超时，我们转过来想对于x有多少i$(1 \le i \le n)$，使得$gcd(n,i) = x$，这个的个数为$\phi(n/x)$（$\phi(x)$表示x的欧拉函数，即小于等于x的数有多少个)。这样枚举n的约数，然后就能算出结果，复杂度为$O(\sqrt{n})$

```
for(int i=1;i*i<=n;i++){
    if(n%i==0&&i*i!=n){
        ans+=gao(n,i-1)*euler(n/i);
        ans+=gao(n,n/i-1)*euler(i);
        ans%=p;
        }
    if(i*i==n){
        ans+=gao(n,i-1)*euler(i);
        ans%=p;
    }
}
```

优化：通过DFS直接枚举出n的所有约数和约数的欧拉函数。

```
int ans=0;
void dfs(int now,int euler,int val)
{
    if(now>=facCnt){
        ans=ans+(gao(n,n/val-1)*euler)%mod;
        ans%=mod;
        return ;
    }
    int pre=1,sum=1;
    for(int i=0;i<=fac[now][1];i++){
        dfs(now+1,euler*pre%mod,val*sum);
        if(i==0){
            pre*=(fac[now][0]-1);
        }else{
            pre*=fac[now][0];
        }
        sum*=fac[now][0];
    }
}
```

立方体染色 立方体面：$1*1^6+6*1^2*4^1+3*1^2*2^2+8*3^2+6*2^3$
立方体楞：$1*1^{12}+6*4^3+3*2^6+8*3^4+6*2^5*1^2$
立方体点：$1*1^8+6*4^2+9*2^4+8*1^2*3^2$
对立方体面来说：如果用三种颜色染色，

$$I=(3^6+6*3^3+3*3^4+8*3^2+6*3^3)/24=57$$

如果限定每种颜色用两次，那么

$$\begin{aligned}I=(r+b+g)^6+6*(r+g+b)^2*(r^4+b^4+g^4)+3*(r+b+g)^2*(r^2+b^2+g^2)^2\\+8*(r^3+b^3+g^3)^2+6*(r^2+b^2+g^2)^3\end{aligned} \tag{1}$$

然后方案数就是$r^2*b^2*g^2$的系数
UVA11255 题目大意：用三种颜色，每种a,b,c个去染$n=a+b+c$的环，问方案数。
解法：同上，用dfs求系数。

```
int a[3],n;
LOL com[45][45];
LOL gao(int g,int t)
{
    if(a[0]%t!=0||a[1]%t!=0||a[2]%t!=0) return 0;
    LOL ans=0;
    int ta=a[0]/t,tb=a[1]/t,tc=a[2]/t;
    ans=com[ta+tb+tc][ta]*com[tb+tc][tb];
    return ans;
}
int main(void)
{
    int T;
    for(int i=0;i<=40;i++){
        com[i][0]=1;
    }
    for(int i=1;i<=40;i++){
        for(int j=1;j<=i;j++){
            com[i][j]=com[i-1][j]+com[i-1][j-1];
        }
    }
    scanf("%d",&T);
    while(T--){
        n=0;
        for(int i=0;i<3;i++){
            scanf("%d",&a[i]);
            n+=a[i];
        }
        LOL ans=0;
        for(int i=1;i<=n;i++){
            int g=gcd(i,n);
            ans+=gao(g,n/g);
        }
        if(n%2==0){
            ans+=n/2*gao(n/2,2);
            for(int i=0;i<3;i++){
                for(int j=0;j<3;j++){
                    a[i]--;a[j]--;
                    ans+=n/2*gao((n-2)/2,2);
                    a[i]++;a[j]++;
                }
            }

        }
        else{
            for(int i=0;i<3;i++){
                a[i]--;
                ans+=n*gao(n/2,2);
                a[i]++;
            }
        }
        cout<<ans/(2*n)<<endl;
    }
    return 0;
}
```

7 计算几何

7.1 几何基础

7.1.1 求两向量的叉积

```
int cross(int x1,int y1,int x2,int y2) //int型
{
    return x1*y2-x2*y1;
}

double cross(double x1,double y1,double x2,double y2)  //double型
{
    return x1*y2-x2*y1;
}

double cross(Point a,Point b)  // 向量叉积
{
    return a.x*b.y-b.x*a.y;
}
```

7.1.2 求平面两点欧氏距离

```
double distance(Point a,Point b)
{
    return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
}
```

7.1.3 判断点是否在线段上

```
int on_segment(Segment seg,Point k)
{
    Point a=seg.s,Point b=seg.e;

    if (cross(k.x-a.x,k.y-a.y,b.x-a.x,b.y-a.y)==0&&
      k.x>=min(a.x,b.x)&&k.x<=max(a.x,b.x)&&
        k.y>=min(a.y,b.y)&&k.y<=max(a.y,b.y))

       return 1;

    else
       return 0;
}
```

7.1.4 判断两线段是否相交（快速排斥+跨立实验）

```
nt segment_intersect(Segment seg1,Segment seg2) //endpoint exlusive
{
    Point s1=seg1.s,e1=seg1.e,s2=seg2.s,e2=seg2.e;

    if (max(s1.x,e1.x)>min(s2.x,e2.x)&&
        max(s2.x,e2.x)>min(s1.x,e1.x)&&
        max(s1.y,e1.y)>min(s2.y,e2.y)&&
        max(s2.y,e2.y)>min(s1.y,e1.y)&&
        cross(e1.x-s1.x,e1.y-s1.y,s2.x-s1.x,s2.y-s1.y)*cross(e1.x-s1.x,e1.y-s1.y,e2.x-s1.x,e2.y-s1.y)<0&&
        cross(e2.x-s2.x,e2.y-s2.y,s1.x-s2.x,s1.y-s2.y)*cross(e2.x-s2.x,e2.y-s2.y,e1.x-s2.x,e1.y-s2.y)<0)

        return 1;

    else
        return 0;
}

```

```
int segment_intersect(Segment seg1,Segment seg2) //endpoint inclusive
{
    Point s1=seg1.s,e1=seg1.e,s2=seg2.s,e2=seg2.e;

    if (max(s1.x,e1.x)>=min(s2.x,e2.x)&&
        max(s2.x,e2.x)>=min(s1.x,e1.x)&&
        max(s1.y,e1.y)>=min(s2.y,e2.y)&&
        max(s2.y,e2.y)>=min(s1.y,e1.y)&&
        cross(e1.x-s1.x,e1.y-s1.y,s2.x-s1.x,s2.y-s1.y)*cross(e1.x-s1.x,e1.y-
           s1.y,e2.x-s1.x,e2.y-s1.y)<=0&&
        cross(e2.x-s2.x,e2.y-s2.y,s1.x-s2.x,s1.y-s2.y)*cross(e2.x-s2.x,e2.y-
           s2.y,e1.x-s2.x,e1.y-s2.y)<=0)

        return 1;

    else
        return 0;
}
```

7.1.5 判断double型变量的符号

```
#define eps 1e-8
int dlcmp(double x)
{
    return x<-eps?-1:x>eps;
}
```

7.1.6 求点到线段的最短距离(此做法过于naive,可用点积+叉积来求)

```
#include<cmath>
#include<algorithm>
#define eps 1e-6
struct Point
{
    double x,y;
};

struct Segment
{
    Point s,e;
};

struct Line  //ax+by+c=0;
{
    double a,b,c;
    Line(double d1=1,double d2=-1,double d3=0) {a=d1;b=d2;c=d3;}
};

int dlcmp(double x)
{
    return x<-eps?-1:x>eps;
}
double dis(Point a,Point b)
{
    return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
}

Line make_line(Point a,Point b)
{
    Line l;
    int sign=1;

    l.a=b.y-a.y;

    if (l.a<0)
    {
        sign=-1;
        l.a=l.a*sign;
    }

    l.b=sign*(a.x-b.x);
    l.c=sign*(a.y*b.x-a.x*b.y);
    return l;
}

double dis_point_segment(Point a,Segment seg)
{
    double A=dis(a,seg.s);
    double B=dis(a,seg.e);
    double C=dis(seg.s,seg.e);

    if (!dlcmp(A+B-C))
        return 0;

    if (!dlcmp(A+C-B)||!dlcmp(B+C-A))
        return min(A,B);

    if (dlcmp(A*A+C*C-B*B)<=0||dlcmp(B*B+C*C-A*A)<=0)
        return min(A,B);

    Line l=make_line(seg.s,seg.e);
    double t=fabs(a.x*l.a+a.y*l.b+l.c);
    t/=sqrt(l.a*l.a+l.b*l.b);

    return t;

}
```

7.1.7 求两线段间最短距离

```
double dis_segments(Segment seg1,Segment seg2)
{
    double m1=dis_point_segment(seg1.s,seg2);

    double m2=dis_point_segment(seg1.e,seg2);
    double m3=dis_point_segment(seg2.s,seg1);
    double m4=dis_point_segment(seg2.e,seg1);

    return min(min(m1,m2),min(m3,m4));
}
```

7.1.8 判断两直线是否相交（共线及平行，若相交则求出交点）

```
#include<math.h>
#define eps 1e-8
struct Point
{
        double x;
        double y;
};

struct Line
{
        Point s;
        Point e;
        Point v;
};

```

```
Line l1,l2;

double x,y交点坐标;//

int line_intersect(Line l1,Line l2)
{
    Point vec;
    double r;

    vec.x=l1.s.x-l2.s.x;
    vec.y=l1.s.y-l2.s.y;

    if (fabs(l1.v.x*l2.v.y-l2.v.x*l1.v.y)<eps)
    {
       if (fabs(l1.v.x*vec.y-vec.x*l1.v.y)<eps)
          return 2;    //共线
       else
          return 0;   //平行
    }
    else
    {
        r=((l1.s.x-l2.s.x)*l2.v.y-(l1.s.y-l2.s.y)*l2.v.x)/(l1.v.x*l2.v.y-l1.
           v.y*l2.v.x);
        x=-r*l1.v.x+l1.s.x;
        y=-r*l1.v.y+l1.s.y;

        return 1;      //相交
    }
}
```

7.1.9 求两直线交点

```
Point line_intersect(Point s1,Point e1,Point s2,Point e2)
{
    Point v1,v2,res;
    double r;

    v1=s1-e1;
    v2=s2-e2;

    r=((s1.x-s2.x)*v2.y-(s1.y-s2.y)*v2.x)/(v1.x*v2.y-v1.y*v2.x);
    res.x=-r*v1.x+s1.x;
    res.y=-r*v1.y+s1.y;

    return res;
}
```

7.1.10 求两线段交点

```
Point line_intersect(Point s1,Point e1,Point s2,Point e2)
{
    Point res;

    double cs=fabs(cross(s2,e1,s1));
    double ce=fabs(cross(e2,e1,s1));

    res.x=(ce*s2.x+cs*e2.x)/(cs+ce);
    res.y=(ce*s2.y+cs*e2.y)/(cs+ce);

    return res;
}
```

7.2 多边形

7.2.1 判断线段是否与矩形相交（包括线段在矩形内部）

```
struct  Rectangle
{
    Point lt;//lefttop
    Point rb;//rightbottom
};

int segement_rectangle_intersect(Segment l,Rectangle r)
{
    Segment d1,d2;//retangle's diagonal

    d1.s=r.lt;
    d1.e=r.rb;
    d2.s.x=d1.e.x;
    d2.s.y=d1.s.y;
    d2.e.x=d1.s.x;
    d2.e.y=d1.e.y;

    if (l.s.x>=r.lt.x&&l.s.x<=r.rb.x&&
        l.s.y<=r.lt.y&&l.s.y>=r.rb.y||
        l.e.x>=r.lt.x&&l.e.x<=r.rb.x&&
        l.e.y<=r.lt.y&&l.e.y>=r.rb.y)

        return 1;

    if (segment_intersect(l,d1)|| segment_intersect(l,d2))  //
       segment_intersect(endpoint inclusive)
        return 1;

    return 0;
}
```

7.2.2 判断点是否在多边形内部（包含在边上）,射线法

```
int point_on_segment(Point o,Point a,Point b)//点在线段上
{
    if (dlcmp(cross(a-o,b-o))==0&&
        o.x>=min(a.x,b.x)&&o.x<=max(a.x,b.x)&&
        o.y>=min(a.y,b.y)&&o.y<=max(a.y,b.y))

        return 1;
    else
        return 0;
}

int point_in_polygon(Point o,Point pln[],int n)//判断点在多边形内部
{
    int i,j,cnt=0;
    Point far(oo,o.y);
    int d1,d2,d3,d4;

    for (i=0;i<n;i++)
    {
        j=(i+1)%n;
        if (point_on_segment(o,pln[i],pln[j]))
            return 1;

        d1=dlcmp(cross(far-o,pln[i]-o));
        d2=dlcmp(cross(far-o,pln[j]-o));
        d3=dlcmp(cross(pln[j]-pln[i],o-pln[i]));
        d4=dlcmp(cross(pln[j]-pln[i],far-pln[i]));

        if (d1*d2<0&&d3*d4<0)
            cnt++;
        else if ((d1*d2==0&&d3*d4<0)&&
                 dlcmp(o.y-max(pln[i].y,pln[j].y))==0)
            cnt++;
    }

    if (cnt&1)
        return 1;
    else
        return 0;
}
```

7.2.3 求简单多边形重心

```
Point get_center(Point pt[],int n)
{
    double sum,area;
    Point res(0,0),o(0,0);
    int i;

    sum=0;
    for (i=0;i<n;i++)
    {
        area=cross(pt[i]-o,pt[(i+1)%n]-o);
        res=res+(pt[i]+pt[(i+1)%n])/3*area;
        sum+=area;
    }

    res=res/sum;
    return res;
}
```

7.2.4 graham_scan求凸包

测试报告：hdu1348 1392

```
#include<cmath>
#include<algorithm>
#define eps 1e-6
#define N 50005
using namespace std;

struct Point
{
    double x,y;
};

Point pt[N],pln[N];

int dlcmp(double x)
{
    return x<-eps?-1:x>eps;
}

double cross(Point a,Point b,Point s)
{
    double x1=a.x-s.x,y1=a.y-s.y;
    double x2=b.x-s.x,y2=b.y-s.y;

    return x1*y2-x2*y1;
}

double dis(Point a,Point b)
```

```
{
    return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
}

int cmp(Point a,Point b)
{
    if (dlcmp(cross(a,b,pt[0]))==1||
        dlcmp(cross(a,b,pt[0]))==0&&
        dis(pt[0],a)<dis(pt[0],b))

        return 1;
    else
        return 0;
}

int graham_scan(int n)
{
    int i,top,t;

    if (n<=1)
        return n;

    for (t=0,i=1;i<n;i++)
        if (dlcmp(pt[i].y-pt[t].y)==-1||
            dlcmp(pt[i].y-pt[t].y)==0&&
            dlcmp(pt[i].x-pt[t].x)==-1)

            t=i;

    swap(pt[0],pt[t]);

    sort(pt+1,pt+n,cmp);

    top=2;
    for (i=0;i<2;i++)
        pln[i]=pt[i];

    for (i=2;i<n;i++)
    {
        while (top>1&&dlcmp(cross(pln[top-1],pt[i],pln[top-2]))<=0)
              top--;
        pln[top++]=pt[i];
    }

    return top;
}
```

7.2.5 旋转卡壳求凸包直径

测试报告：poj2187

```
double  rotating_calipers(Point pln[],int n)
{
    int p,q;
    double res;

    pln[n]=pln[0];
    res=0;

    for (p=0,q=1;p<n;p++)
    {
        while (cross(pln[p+1],pln[q],pln[p])<
               cross(pln[p+1],pln[q+1],pln[p]))

        q=(q+1)%n;

        res=max(res,max(dis(pln[p],pln[q]),dis(pln[p+1],pln[q+1])));
    }

    return res;
}
```

7.2.6 凸包间最短距离（旋转卡壳）

测试报告：poj3608

```
const double eps=1e-6;
const double oo=1e100;
const double PI=3.141592657589793;

struct Point
{
    double x,y;
};


struct Segment
{
    Point s,e;
};

struct Line  //ax+by+c=0;
{
    double a,b,c;

    Line(double d1=1,double d2=-1,double d3=0) {a=d1;b=d2;c=d3;}
};

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

double cross1(double x1,double y1,double x2,double y2)
{
    return x1*y2-x2*y1;
}

double cross2(Point a,Point b)
{
    return a.x*b.y-b.x*a.y;
}

double cross3(Point a,Point b,Point s)
{

    double x1=a.x-s.x,y1=a.y-s.y;
    double x2=b.x-s.x,y2=b.y-s.y;

    return x1*y2-x2*y1;
}

int dlcmp(double x)
{
    return x<-eps?-1:x>eps;
}
```

```

double polygon_area(Point pln[],int n)
{
    double sum=0;
    int i;

    for (i=0;i<n;i++)
        sum+=cross2(pln[i],pln[(i+1)%n]);

    return sum/2;
}

void reverse_clockwise(Point pln[],int n)
{
    int i;

    for (i=0;i<=(n-1)/2;i++)
        swap(pln[i],pln[n-i-1]);
}

double calculate_degree(Point a,Point b,double cur)
{
    double ang,res;
    Point p;

    p.x=b.x-a.x;
    p.y=b.y-a.y;

    if (!dlcmp(p.x))
    {
        if (p.y>0)
            ang=PI/2;
        else
            ang=3*PI/2;
    }
    else
    {
        ang=atan(p.y/p.x);
        if (p.x<0)
            ang+=PI;
    }

    while (ang<0)
        ang+=2*PI;

    if (ang>=PI)
        cur+=PI;

    if (ang>cur)
        res=ang-cur;
    else
        res=PI-(cur-ang);

    while (res>=PI)
        res-=PI;

    if (!dlcmp(res-PI))
        res=0;

    return res;
}


```

```

Line make_line(Point a,Point b)
{
    Line l;
    int sign=1;

    l.a=b.y-a.y;

    if (l.a<0)
    {
        sign=-1;
        l.a=l.a*sign;
    }

    l.b=sign*(a.x-b.x);
    l.c=sign*(a.y*b.x-a.x*b.y);

    return l;
}

double dis_point_segment(Point a,Segment seg)
{
    double A=dis(a,seg.s);
    double B=dis(a,seg.e);
    double C=dis(seg.s,seg.e);

    if (!dlcmp(A+B-C))
        return 0;

    if (!dlcmp(A+C-B)||!dlcmp(B+C-A))
        return min(A,B);

    if (dlcmp(A*A+C*C-B*B)<=0||dlcmp(B*B+C*C-A*A)<=0)
        return min(A,B);

    Line l=make_line(seg.s,seg.e);
    double t=fabs(a.x*l.a+a.y*l.b+l.c);
    t/=sqrt(l.a*l.a+l.b*l.b);

    return t;
}

double dis_segments(Segment seg1,Segment seg2)
{
    double m1=dis_point_segment(seg1.s,seg2);
    double m2=dis_point_segment(seg1.e,seg2);
    double m3=dis_point_segment(seg2.s,seg1);
    double m4=dis_point_segment(seg2.e,seg1);

    return min(min(m1,m2),min(m3,m4));
}


double dis_polygons(Point pln1[],int n1,Point pln2[],int n2)
{
    int i,j,k,p1=0,p2=0;
    double res=oo,arg=0,cur=0;
    double ang1,ang2,tmp;
    Segment seg1,seg2;

    if (polygon_area(pln1,n1)<0)
        reverse_clockwise(pln1,n1);
    if (polygon_area(pln2,n2)<0)
```

```
        reverse_clockwise(pln2,n2);

    for (i=1;i<n1;i++)
        if (pln1[i].y<pln1[p1].y)
            p1=i;

    for (i=1;i<n2;i++)
        if (pln2[i].y>pln2[p2].y)
            p2=i;

    while (arg<=360)
    {
           while (cur>=PI)
            cur-=PI;

           if (!dlcmp(cur-PI))
            cur=0;

           ang1=calculate_degree(pln1[p1],pln1[(p1+1)%n1],cur);
           ang2=calculate_degree(pln2[p2],pln2[(p2+1)%n2],cur);

           if (!dlcmp(ang1-ang2))
           {
               cur+=ang1;
               arg+=ang1;

               seg1.s=pln1[p1],seg1.e=pln1[(p1+1)%n1];
               seg2.s=pln2[p2],seg2.e=pln2[(p2+1)%n2];

               tmp=dis_segments(seg1,seg2);
               res=min(res,tmp);

               p1=(p1+1)%n1;
               p2=(p2+1)%n2;
           }
           else if (dlcmp(ang1-ang2)>0)
           {
               cur+=ang2;
               arg+=ang2;

               seg2.s=pln2[p2],seg2.e=pln2[(p2+1)%n2];
               tmp=dis_point_segment(pln1[p1],seg2);
               res=min(tmp,res);

               p2=(p2+1)%n2;
           }
           else
           {
               cur+=ang1;
               arg+=ang1;
               seg1.s=pln1[p1],seg1.e=pln1[(p1+1)%n1];
               tmp=dis_point_segment(pln2[p2],seg1);

               res=min(tmp,res);
               p1=(p1+1)%n1;
           }
    }

    return res;
}

main()
{
    int n1,n2,i;
    double ans;
    Point pln1[N],pln2[N];

    while (scanf("%d%d",&n1,&n2),n1||n2)
    {
        for (i=0;i<n1;i++)
            scanf("%lf%lf",&pln1[i].x,&pln1[i].y);
        for (i=0;i<n2;i++)
            scanf("%lf%lf",&pln2[i].x,&pln2[i].y);

        ans=dis_polygons(pln1,n1,pln2,n2);

        printf("%.5lf\n",ans);
    }

    return 0;
}
```

7.2.7 判断两凸多边形是否相交（graham_scan求凸包+枚举边、点）

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#define N 110
using namespace std;

struct Point
{
    int x,y;
};

struct Polygon
{
    Point p[N];
    int n;
};

Point pt[N];
int stack[N];

int cross(Point a,Point b,Point s)
{
    return (a.x-s.x)*(b.y-s.y)-(b.x-s.x)*(a.y-s.y);
}

int dist(Point a,Point b)
{
    return (a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y);
}

int cmp(Point a,Point b)
{
    if (cross(a,b,pt[0])>0||cross(a,b,pt[0])==0&&dist(a,pt[0])<dist(b,pt[0])
       )
        return 1;
    else
        return 0;
}

int on_segment(Point s,Point e,Point o)
{
    if(cross(s,e,o)==0&&o.x>=min(s.x,e.x)&&
       o.x<=max(s.x,e.x)&&o.y>=min(s.y,e.y)&&o.y<=max(s.y,e.y))

        return 1;
    else
        return 0;
}

int graham_scan(int n)
{
    int i,top,t;

    if (n<=1)
    {
        stack[0]=0;
        return n;
    }

    for (t=0,i=1;i<n;i++)
        if (pt[i].y<pt[t].y||pt[i].y==pt[t].y&&pt[i].x<pt[t].x)
            t=i;

    swap(pt[0],pt[t]);

    sort(pt+1,pt+n,cmp);

    top=2;
    for (i=0;i<2;i++)
        stack[i]=i;

    for (i=2;i<n;i++)
    {
        while (top>1&&cross(pt[stack[top-1]],pt[i],pt[stack[top-2]])<=0)
            top--;
        stack[top++]=i;
    }

    return top;
}

int segment_intersect(Point s1,Point e1,Point s2,Point e2)
{
    if (max(s1.x,e1.x)>=min(s2.x,e2.x)&&
       max(s2.x,e2.x)>=min(s1.x,e1.x)&&
       max(s1.y,e1.y)>=min(s2.y,e2.y)&&
       max(s2.y,e2.y)>=min(s1.y,e2.y)&&
       (double)cross(s2,e1,s1)*(double)cross(e2,e1,s1)<=0&&
       (double)cross(s1,e2,s2)*(double)cross(e1,e2,s2)<=0)

       return 1;
    else
       return 0;
}

int point_inside(Point o,Polygon pln)
{
    int i,a1=0,a2=0,n=pln.n;

    pln.p[n]=pln.p[0];

    for (i=0;i<n;i++)
        a1+=abs(cross(pln.p[i],pln.p[i+1],o));
    for (i=1;i<n;i++)
```

```
        a2+=abs(cross(pln.p[i],pln.p[i+1],pln.p[0]));

    if (a1==a2)
        return 1;

    else
        return 0;
}

int convex_polygon_intersect(Polygon pln1,Polygon pln2)
{
    int i,j;

    pln1.p[pln1.n]=pln1.p[0];
    pln2.p[pln2.n]=pln2.p[0];

    for (i=0;i<pln1.n;i++)
        for (j=0;j<pln2.n;j++)
            if (segment_intersect(pln1.p[i],pln1.p[i+1],pln2.p[j],pln2.p[j
                +1]))
                return 1;

    if (point_inside(pln1.p[0],pln2)||point_inside(pln2.p[0],pln1))
        return 1;

    return 0;
}

int main()
{
    int n,m,i,vertexnum,ans;
    Polygon pln1,pln2;

    while (cin>>n>>m,n||m)
    {
        for (i=0;i<n;i++)
            cin>>pt[i].x>>pt[i].y;
        vertexnum=graham_scan(n);
        pln1.n=vertexnum;

        for (i=0;i<vertexnum;i++)
            pln1.p[i]=pt[stack[i]];

        for (i=0;i<m;i++)
            cin>>pt[i].x>>pt[i].y;

        vertexnum=graham_scan(m);
        pln2.n=vertexnum;

        for (i=0;i<vertexnum;i++)
            pln2.p[i]=pt[stack[i]];

        if (pln1.n==1&&pln2.n==1)
            ans=1;
        else if (pln1.n==1&&pln2.n==2)
        {
            if (on_segment(pln2.p[0],pln2.p[1],pln1.p[0]))
                ans=0;
            else
                ans=1;
        }
        else if (pln1.n==2&&pln2.n==1)
        {
            if (on_segment(pln1.p[0],pln1.p[1],pln2.p[0]))
                ans=0;
            else
                ans=1;
        }
        else if (pln1.n==2&&pln2.n==2)
        {
            if (segment_intersect(pln1.p[0],pln1.p[1],pln2.p[0],pln2.p[1]))
                ans=0;
            else
                ans=1;
        }
        else if (pln1.n==1)
        {
            if (point_inside(pln1.p[0],pln2))
                ans=0;
            else
                ans=1;
        }
        else if (pln2.n==1)
        {
            if (point_inside(pln2.p[0],pln1))
                ans=0;
            else
                ans=1;
        }
        else
        {
            if (convex_polygon_intersect(pln1,pln2)==0)
                ans=1;
            else
                ans=0;
        }

        if (ans==0)
            cout<<"NO"<<endl;
        else
           cout<<"YES"<<endl;
    }

    return 0;
}
```

7.3 半平面交

7.3.1 求多边形内核

测试报告：poj3130 3335 3384
复杂度$O(n^2)$

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<cmath>
#define eps 1e-8
#define N 1010
using namespace std;

struct Point
{
    double x,y;

    Point(){};
    Point(double a,double b):x(a),y(b){}

    Point operator - (const Point a) const {return Point(x-a.x,y-a.y);}

};
Point kernel[N],pt[N];

int dlcmp(double x)
{
    if (x<-eps)
        return -1;
    else
        return x>eps?1:0;
}

double cross(Point v1,Point v2) //向量叉积
{
    return v1.x*v2.y-v2.x*v1.y;
}

Point line_intersect(Point s1,Point e1,Point s2,Point e2)  //求两点所在直线的交点
{
    Point v1,v2,res;
    double r;

    v1=s1-e1;
    v2=s2-e2;

    r=((s1.x-s2.x)*v2.y-(s1.y-s2.y)*v2.x)/(v1.x*v2.y-v1.y*v2.x);
    res.x=-r*v1.x+s1.x;
    res.y=-r*v1.y+s1.y;

    return res;
}

void rev_points(Point p[],int n)
{
    int i;

    for (i=0;i<n/2;i++)
        swap(p[i],p[n-i-1]);
}

double polygon_area(Point pln[],int n) //求多边形有向面积，顺时针为负，逆时针为正
```

```
{
    int i;
    double res=0;

    for (i=0;i<n;i++)
        res+=cross(pln[i],pln[(i+1)%n]);

    return res/2.0;
}

void cut_polygon(Point org[],int on,Point des[],int &dn,Point s,Point e) //多
    边形切割
{
    int i;
    int d1,d2;

    dn=0;

    for (i=0;i<on;i++)
    {
        d1=dlcmp(cross(e-s,org[i]-s));
        d2=dlcmp(cross(e-s,org[(i+1)%on]-s));

        if (d1>=0)
            des[dn++]=org[i];
        if (d1*d2<0)
            des[dn++]=line_intersect(s,e,org[i],org[(i+1)%on]);
    }
}

void polygon_kernel(Point org[],int on,Point kernel[],int &kn) //求多边形内核
{
    int i,j,dn;
    Point des[N];

    if (dlcmp(polygon_area(org,on))<0)
        rev_points(org,on);

    for (i=0;i<on;i++)
        kernel[i]=org[i];
    kn=on;

    for (i=0;i<on;i++)
    {
        cut_polygon(kernel,kn,des,dn,org[i],org[(i+1)%on]);

        for (j=0;j<dn;j++)
            kernel[j]=des[j];
        kn=dn;
    }
}

//poj3335
int main()
{
    int t,i,pn,kn;

    scanf("%d",&t);
    while (t--)
    {
        scanf("%d",&pn);

        for (i=0;i<pn;i++)
```

```
            scanf("%lf%lf",&pt[i].x,&pt[i].y);

        polygon_kernel(pt,pn,kernel,kn);

        if (kn==0)
            printf("NO\n");
        else
            printf("YES\n");
    }

    return 0;
}
```

7.3.2 zzyO(n*logn)做法

测试报告：poj2451

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<vector>
#include<string>
#include<queue>
#include<cmath>
#define N 60000
#define eps 1e-8
using namespace std;

int dlcmp(double x)
{
    return x<-eps?-1:x>eps;
}

struct Point
{
    double x,y;

    Point(){}
    Point(double a,double b):x(a),y(b){}

    Point operator + (const Point a) const {return Point(x+a.x,y+a.y);}
    Point operator - (const Point a) const {return Point(x-a.x,y-a.y);}
};

double cross(Point v1,Point v2)
{
    return v1.x*v2.y-v2.x*v1.y;
}

Point line_intersect(Point s1,Point e1,Point s2,Point e2)
{
    Point v1,v2,res;
    double r;

    v1=s1-e1;
    v2=s2-e2;
    r=((s1.x-s2.x)*v2.y-(s1.y-s2.y)*v2.x)/(v1.x*v2.y-v1.y*v2.x);

    res.x=-r*v1.x+s1.x;
    res.y=-r*v1.y+s1.y;

    return res;
}
```

```

struct Vector
{
    Point s,e;
    double ang;

    Vector(){}
    Vector(Point a,Point b):s(a),e(b){ang=atan2(e.y-s.y,e.x-s.x);}

    bool operator == (const Vector &v) const
    {
        return dlcmp(ang-v.ang)==0;
    }

    bool operator < (const Vector &v) const
    {
        if (dlcmp(ang-v.ang)==0)
            return dlcmp(cross(v.e-v.s,e-v.s))>=0;
        else
            return dlcmp(ang-v.ang)<0;
    }
};

struct HalfPlane
{
    Vector plane[N];
    int n;

    HalfPlane():n(0){}

    void add(Vector v)
    {
        plane[n++]=v;
    }

    void add(Point s,Point e)
    {
        Vector v(s,e);
        plane[n++]=v;
    }

    int check(Vector &v1,Vector &v2,Vector &v0)
    {
        Point o=line_intersect(v1.s,v1.e,v2.s,v2.e);

        return dlcmp(cross(o-v0.s,v0.e-v0.s))>0;
    }

    //首先需要保证存在交集
    void work(Point pln[],int &pn)
    {
        int i,cnt,head,tail;
        Vector deque[N];

        sort(plane,plane+n);
        for (cnt=i=1;i<n;i++)
            if (!(plane[i]==plane[i-1]))
                plane[cnt++]=plane[i];
        n=cnt;

        head=1,tail=2;
        deque[head]=plane[0];
        deque[tail]=plane[1];

        for (i=2;i<n;i++)
        {
            while (head<tail&&check(deque[tail-1],deque[tail],plane[i]))
                tail--;
            while (head<tail&&check(deque[head+1],deque[head],plane[i]))
                head++;
            deque[++tail]=plane[i];
        }

        while (head<tail&&check(deque[tail-1],deque[tail],deque[head]))
            tail--;
        while (head<tail&&check(deque[head+1],deque[head],deque[tail]))
            head++;
        deque[--head]=deque[tail];

        for (pn=0,i=head+1;i<=tail;i++)
            pln[pn++]=line_intersect(deque[i-1].s,deque[i-1].e,deque[i].s,
                deque[i].e);
    }
};

Point pln[N];
HalfPlane w;

double get_area(Point pln[],int n)
{
    int i;
    double res=0;

    for (i=1;i<n-1;i++)
        res+=cross(pln[i]-pln[0],pln[i+1]-pln[0]);

    return fabs(res/2);
}

//poj2451
int main()
{
    int n,pn,i;
    Point s,e;
    double ans;

    s=Point(0,0);e=Point(10000,0);w.add(s,e);
    s=Point(10000,0);e=Point(10000,10000);w.add(s,e);
    s=Point(10000,10000);e=Point(0,10000);w.add(s,e);
    s=Point(0,10000);e=Point(0,0);w.add(s,e);

    scanf("%d",&n);
    for (i=0;i<n;i++)
    {
        scanf("%lf%lf%lf%lf",&s.x,&s.y,&e.x,&e.y);
        w.add(s,e);
    }

    w.work(pln,pn);
    ans=get_area(pln,pn);

    printf("%.1f\n",ans);

    return 0;
}
```

7.3.3 两个简单多边形求面积并、交

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<cmath>
#define eps 1e-8
#define N 550
using namespace std;

struct Point
{
    double x,y;

    Point (){}
    Point (double a,double b):x(a),y(b){}

    Point operator - (const Point a) const {return Point(x-a.x,y-a.y);}
};

Point zero(0,0);
int dlcmp(double x) {return x<-eps?-1:x>eps;}
double cross(Point v1,Point v2) {return v1.x*v2.y-v2.x*v1.y;}

struct Polygon
{
    Point p[N];
    int n;

    Polygon():n(0){}
    void clear(){n=0;}
    void add(Point a){p[n++]=a;}

    double area()
    {
        double res=0;
        for (int i=1;i<n-1;i++)
            res+=cross(p[i]-p[0],p[i+1]-p[0]);
        return fabs(res/2);
    }
};

Polygon A,B,rec;


Point line_intersect(Point s1,Point e1,Point s2,Point e2) //两直线交点
{
    Point v1,v2,res;
    double r;

    v1=s1-e1;
    v2=s2-e2;
    r=((s1.x-s2.x)*v2.y-(s1.y-s2.y)*v2.x)/(v1.x*v2.y-v1.y*v2.x);
    res.x=-r*v1.x+s1.x;
    res.y=-r*v1.y+s1.y;

    return res;
}

void cut(Point s,Point e)  //半平面交
{
    int i,j,d1,d2;
    Polygon ker;

    for (i=0;i<rec.n;i++)
    {
        j=(i+1)%rec.n;
        d1=dlcmp(cross(e-s,rec.p[i]-s));
        d2=dlcmp(cross(e-s,rec.p[j]-s));

        if (d1>=0)
            ker.add(rec.p[i]);
        if (d1*d2<0)
            ker.add(line_intersect(s,e,rec.p[i],rec.p[j]));
    }

    rec=ker;
}

double calc(Point p1,Point p2,Point q1,Point q2)
{
    int dp=dlcmp(cross(p1,p2)),dq=dlcmp(cross(q1,q2));
    int sgn=dp*dq;

    if (sgn==0)
        return 0;

    rec.clear();
    rec.add(zero); rec.add(p1); rec.add(p2);
    if (dp<0)
        swap(rec.p[1],rec.p[2]);
    if (dq>0)
    {
        cut(zero,q1);
        cut(q1,q2);
        cut(q2,zero);
    }
    else
    {
        cut(zero,q2);
        cut(q2,q1);
        cut(q1,zero);
    }

    return sgn*rec.area();
}

double solve()
{
    double res=A.area()+B.area();
    double sum=0;
    int i,j;

    //对两个多边形三角剖分，分别求两个三角形的面积交
    for (i=0;i<A.n;i++)
        for (j=0;j<B.n;j++)
            sum+=calc(A.p[i],A.p[(i+1)%A.n],B.p[j],B.p[(j+1)%B.n]);
    res-=fabs(sum);    //fabs(sum)为两个多边形的面积交

    return res; //面积并
}
//hdu3060(题目数据有误)
int main()
{
    int n,m,i;
    Point pt;
    double ans;

    while (scanf("%d%d",&n,&m)!=EOF)
    {
        A.clear();B.clear();
        for (i=0;i<n;i++)
        {
            scanf("%lf%lf",&pt.x,&pt.y);
            A.add(pt);
        }
        for (i=0;i<m;i++)
        {
            scanf("%lf%lf",&pt.x,&pt.y);
            B.add(pt);
        }

        ans=solve();

        printf("%.2f\n",ans+eps);
    }

    return 0;
}
```

7.4 圆

7.4.1 点类

```
struct Point
{
    double x,y;

    Point(){}
    Point(double a,double b):x(a),y(b){}
    Point operator + (const Point a) const {return Point(x+a.x,y+a.y);}
    Point operator - (const Point a) const {return Point(x-a.x,y-a.y);}
    Point operator * (const double a) const {return Point(x*a,y*a);}
    Point operator / (const double a) const {return Point(x/a,y/a);}

    bool operator < (const Point a) const
    {
        if (dlcmp(x-a.x)==0)
            return dlcmp(x-a.y)<0;
        else
            return dlcmp(x-a.x)<0;
    }
    bool operator == (const Point a) const
    {
        return !dlcmp(x-a.x)&&!dlcmp(y-a.y);
    }

    //向量长度定为d
    Point trunc(double d)
    {
        double dis(Point,Point);
        double len=dis(*this,Point(0,0));
        return Point(x*d/len,y*d/len);
    }

    //坐标逆时针旋转a度
    Point rotate(double a)
    {
        return Point(x*cos(a)-y*sin(a),y*cos(a)+x*sin(a));
    }
};

double dis(Point a,Point b)
{
    return sqrt(sqr(a.x-b.x)+sqr(a.y-b.y));
}

double cross(Point a,Point b,Point s)
{
    double x1=a.x-s.x,y1=a.y-s.y;
    double x2=b.x-s.x,y2=b.y-s.y;

    return x1*y2-x2*y1;
}

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

double dot(Point a,Point b,Point s)
{
    double x1=a.x-s.x,y1=a.y-s.y;
    double x2=b.x-s.x,y2=b.y-s.y;

    return x1*x2+y1*y2;
}

double dot(Point a,Point b)
{
    return a.x*b.x+a.y*b.y;
}
```

7.4.2　圆类

```
struct Circle
{
    Point o;
    double r;


    Circle(){}
    Circle(Point a,double l):o(a),r(l){}

    double area(){return sqr(r)*PI;}
};

//判断圆a是否含于圆b
int inner_circle(Circle a,Circle b)
{
    if (dlcmp(a.r-b.r)>0)
        return 0;
    return dlcmp(dis(a.o,b.o)+a.r-b.r)<=0;
}

//以base点为基点，极角排序，排序前base需赋初值
Point base;
int cmp(const Point a,const Point b)
{
    return atan2(a.y-base.y,a.x-base.x)<atan2(b.y-base.y,b.x-base.x);
}

//向量a,b的夹角
double vec_angle(Point a,Point b)
{
    double tmp=dot(a,b)/(dis(a,Point(0,0))*dis(b,Point(0,0)));
    if (dlcmp(tmp-1)>=0) tmp=1;
    if (dlcmp(tmp+1)<=0) tmp=-1;

    return acos(tmp);
}

//计算由a到b逆时针方向的弓形面积
double arc_area(Point a,Point b,Circle c)
{
    double theta=vec_angle(a-c.o,b-c.o);
    double sf=sqr(c.r)*theta/2.0;
    double st=sqr(c.r)*sin(theta)/2.0;

    if (dlcmp(cross(a,b,c.o))>0)
        return sf-st;
    else
        return c.area()-sf+st;
}

double arc_area(double th,double r)
```

```
{
        return 0.5*sqr(r)*(th-sin(th));
}
```

7.4.3 圆面积交、并

测试报告：sgu435 hdu3239 spojCRCU spojCRCUT

```
//求两圆交点，排除相切的情况，不考虑内含
int inter_circle_or(Circle c1,Circle c2,Point &p1,Point &p2)
{
    double len=dis(c1.o,c2.o);

    if (dlcmp(len-c1.r-c2.r)>=0)
        return 0;

    double s=(sqr(c1.r)-sqr(c2.r)+sqr(len))/len/2;
    double h=sqrt(sqr(c1.r)-sqr(s));
    Point vec=c2.o-c1.o;
    Point p0=c1.o+vec.trunc(s);

    p1=p0+vec.rotate(PI/2).trunc(h);
    p2=p0-vec.rotate(PI/2).trunc(h);
    return 1;
}

//求两圆交点，不排除相切的情况，不考虑内含
int inter_circle_and(Circle c1,Circle c2,Point &p1,Point &p2)
{
    double len=dis(c1.o,c2.o);

    if (dlcmp(len-c1.r-c2.r)>0)
        return 0;

    double s=(sqr(c1.r)-sqr(c2.r)+sqr(len))/len/2;
    double h=sqrt(sqr(c1.r)-sqr(s));
    Point vec=c2.o-c1.o;
    Point p0=c1.o+vec.trunc(s);

    p1=p0+vec.rotate(PI/2).trunc(h);
    p2=p0-vec.rotate(PI/2).trunc(h);
    return 1;
}

struct Circles
{
    int n;
    Circle c[MAXN];

    Circles():n(0){}
    void add(Circle cc) {c[n++]=cc;}
    void clear() {n=0;}

    //初始化圆的面积并，去掉能被其他圆覆盖的圆
    void init_or()
    {
        char mark[MAXN]={0};

        int i,j,cnt=0;

        for (i=0;i<n;i++)
            for (j=0;j<n;j++)
                if (i!=j&&!mark[j]&&inner_circle(c[i],c[j]))
                {
                    mark[i]=1;
                    break;
                }

        for (i=0;i<n;i++)
            if (!mark[i])
                c[cnt++]=c[i];
        n=cnt;
    }

    //初始化圆的面积并，去掉能把其他圆覆盖的圆
    void init_and()
    {
        char mark[MAXN]={0};

        int i,j,cnt=0;

        for (i=0;i<n;i++)
            for (j=0;j<n;j++)
                if (i!=j&&!mark[j]&&inner_circle(c[j],c[i]))
                {
                    mark[i]=1;
                    break;
                }

        for (i=0;i<n;i++)
            if (!mark[i])
                c[cnt++]=c[i];
        n=cnt;
    }

    //判断圆弧是否被其他圆覆盖
    int isvalid_or(Point a,Point b,int num)
    {
        Point vec,p;
        int i;

        vec=a-b;
        p=c[num].o+vec.rotate(PI/2).trunc(c[num].r);

        for (i=0;i<n;i++)
            if (i!=num&&dlcmp(dis(p,c[i].o)-c[i].r)<0)
                return 0;
        return 1;
    }

    //判断点是否被其他圆覆盖
    int isvalid_and(Point a)
    {
        int i;

        for (i=0;i<n;i++)
            if (dlcmp(dis(a,c[i].o)-c[i].r)>0)
                return 0;
        return 1;
    }

    //判断圆弧是否被其他圆覆盖
    int isvalid_and(Point a,Point b,int num)
    {
        Point vec,p;
        int i;

        vec=a-b;
        p=c[num].o+vec.rotate(PI/2).trunc(c[num].r);

        return isvalid_and(p);
    }

    //计算圆的面积并
    double area_or()
    {
        int i,j,k;
        vector<Point>s[MAXN];
        Point a,b;
        double sa=0,sp=0;

        init_or();

        for (i=0;i<n;i++)
            for (j=i+1;j<n;j++)
                if (inter_circle_or(c[i],c[j],a,b))
                {
                    s[i].push_back(a);
                    s[i].push_back(b);
                    s[j].push_back(a);
                    s[j].push_back(b);
                }


        for (i=0;i<n;i++)
        {
            if (s[i].empty())
            {
                sa+=c[i].area();
                continue;
            }

            base=c[i].o;
            sort(s[i].begin(),s[i].end(),cmp);
            s[i].resize(unique(s[i].begin(),s[i].end())-s[i].begin());
            if (s[i].front()==s[i].back())
                s[i].pop_back();

            for (j=0;j<s[i].size();j++)
            {
                k=(j+1)%s[i].size();

                if (isvalid_or(s[i][j],s[i][k],i))
                {
                    sa+=arc_area(s[i][j],s[i][k],c[i]);
                    sp+=cross(s[i][j],s[i][k],Point(0,0));
                }
            }
        }

        return sa+fabs(sp)/2.0;
    }

    //计算圆的面积交，若交集为空，返回-1.0，若交集为一点，保存交点到res中
    double area_and(Point &res)
    {
        int i,j,k;
        vector<Point>s[MAXN];
        Point a,b;
        double sa=0,sp=0;

        init_and();
        if (n==1)
            return c[0].area();

        for (i=0;i<n;i++)
            for (j=i+1;j<n;j++)
                if (inter_circle_and(c[i],c[j],a,b))
                {
                    s[i].push_back(a);
                    s[i].push_back(b);
                    s[j].push_back(a);
                    s[j].push_back(b);
                }
                else
                    return -1.0;

        for (i=0;i<n;i++)
        {
            base=c[i].o;
            sort(s[i].begin(),s[i].end(),cmp);

            s[i].resize(unique(s[i].begin(),s[i].end())-s[i].begin());
            if (s[i].front()==s[i].back())
                s[i].pop_back();

            if (s[i].size()==1)
            {
                if (isvalid_and(s[i][0]))
                    res=s[i][0];
                continue;
            }

            for (j=0;j<s[i].size();j++)
            {
                if (isvalid_and(s[i][j]))
                    res=s[i][j];

                k=(j+1)%s[i].size();

                if (isvalid_and(s[i][j],s[i][k],i))
                {
                    sa+=arc_area(s[i][j],s[i][k],c[i]);
                    sp+=cross(s[i][j],s[i][k],Point(0,0));
                }
            }
        }

        return sa+fabs(sp)/2.0;
    }


    //计算被覆盖i次的面积
    double ans[MAXN],pre[MAXN]; //ans[i]保存被覆盖i次的面积

    void get_area()
    {
        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].o-c[i].o;
                    double ab=dis(q,Point(0,0)),ac=c[i].r,bc=c[j].r;

                    if (dlcmp(ab+ac-bc)<=0)
                    {
                        v.push_back(make_pair(-PI,1));
                        v.push_back(make_pair(PI,-1));
                        continue;
                    }

                    if (dlcmp(ab+bc-ac)<=0||dlcmp(ab-ac-bc)>0)
                        continue;

                    double th=atan2(q.y,q.x);
                    double fai=acos((ac*ac+ab*ab-bc*bc)/(2.0*ac*ab));

                    double a0=th-fai;
                    if (dlcmp(a0+PI)<0)
                        a0+=2*PI;
                    double a1=th+fai;
                    if (dlcmp(a1-PI)>0)
                        a1-=2*PI;

                    if (dlcmp(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&&dlcmp(v[j].first-pre[cur]))
                {
                    ans[cur]+=arc_area(v[j].first-pre[cur],c[i].r);
                    Point pa(c[i].o.x+c[i].r*cos(pre[cur]),c[i].o.y+c[i].r*
                        sin(pre[cur]));
                    Point pb(c[i].o.x+c[i].r*cos(v[j].first),c[i].o.y+c[i].r
                        *sin(v[j].first));
                    ans[cur]+=0.5*cross(pa,pb);
                }
                cur+=v[j].second;
                pre[cur]=v[j].first;
            }
        }

        for (i=1;i<=n;i++)
            ans[i]-=ans[i+1];
    }
};
```

7.4.4 简单多边形与圆求面积交

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<cmath>
#define N 200
#define eps 1e-8
using namespace std;

const double PI=acos(-1.0);

struct Point
{
    double x,y;
};

Point pt[N];
int n;

int dlcmp(double x)
{
    return x<-eps?-1:x>eps;
}

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

double dis(Point a,Point b)
{
    return sqrt(sqr(a.x-b.x)+sqr(a.y-b.y));
}

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

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

double calc_area(Point a,Point b,Point c,double r)
{
    double A,B,C,x,y,tS;

    A=dis(b,c);
    B=dis(a,c);
    C=dis(b,a);

    if (A<r&&B<r)
        return outer(a,b,c)/2;

    else if (A<r&&B>=r)
    {
        x=(inner(a,c,b)+sqrt(sqr(r)*sqr(C)-sqr(outer(a,c,b))))/C;
        tS=outer(a,b,c)/2;

        return asin(tS*(1-x/C)*2/r/B)*sqr(r)/2+tS*x/C;
    }
    else if (A>=r&&B<r)
    {
        y=(inner(b,c,a)+sqrt(sqr(r)*sqr(C)-sqr(outer(b,c,a))))/C;
        tS=outer(a,b,c)/2;

        return asin(tS*(1-y/C)*2/r/A)*sqr(r)/2+tS*y/C;
    }
    else if (fabs(outer(a,b,c))>=r*C||inner(b,c,a)<=0||inner(a,c,b)<=0)
    {
        if (inner(a,b,c)<0)
        {
            if (outer(a,b,c)<0)
                return (-PI-asin(outer(a,b,c)/A/B))*sqr(r)/2;
            else
                return (PI-asin(outer(a,b,c)/A/B))*sqr(r)/2;
        }
        else
            return asin(outer(a,b,c)/A/B)*sqr(r)/2;
    }
    else
    {
        x=(inner(a,c,b)+sqrt(sqr(r)*sqr(C)-sqr(outer(a,c,b))))/C;
        y=(inner(b,c,a)+sqrt(sqr(r)*sqr(C)-sqr(outer(b,c,a))))/C;
        tS=outer(a,b,c)/2;

        return (asin(tS*(1-x/C)*2/r/B)+asin(tS*(1-y/C)*2/r/A))*sqr(r)/2+tS
           *((y+x)/C-1);
    }
}

//计算一般多边形与圆的交面积（将多边形划分为三角形，然后有向三角形与圆求有向面积交）
double solve(Point o,double r)
{
    int i,j;
    double res,sum;
    Point tri[3];

    res=0;
    for (i=1;i<n-1;i++)
    {
        tri[0]=pt[0];
        tri[1]=pt[i];
        tri[2]=pt[i+1];
        sum=0;

        for (j=0;j<3;j++)
            sum+=calc_area(tri[j],tri[(j+1)%3],o,r);

        //sum为三角形与圆交的有向面积
        res+=sum;
    }

    return fabs(res);
}

```

```
//poj3675
int main()
{
    double x0,y0,v,vx,vy,g,r,t,theta,ans;
    Point o;
    int i;

    o.x=o.y=0;

    while (scanf("%lf",&r)!=EOF)
    {
        scanf("%d",&n);
        for (i=0;i<n;i++)
            scanf("%lf%lf",&pt[i].x,&pt[i].y);

        ans=solve(o,r);

        printf("%.2f\n",ans);
    }

    return 0;
}
```

7.4.5 求线段与圆的交点

若求直线与圆的交点类似，无需讨论k1、k2的取值范围

```
int inter_circle_segment(Circle c,Point a,Point b,Point &p1,Point &p2)
{
    Point vec=b-a;
    double A=sqr(vec.x)+sqr(vec.y);
    double B=2*(vec.x*(a.x-c.o.x)+vec.y*(a.y-c.o.y));
    double C=sqr(a.x-c.o.x)+sqr(a.y-c.o.y)-sqr(c.r);
    double delta=sqr(B)-4*A*C;

    if (dlcmp(delta)<0)
        return 0;

    double k1=(-B-sqrt(fabs(delta)))/(2*A);
    double k2=(-B+sqrt(fabs(delta)))/(2*A);
    int res=0;


    if (dlcmp(k1)>=0&&dlcmp(k1-1)<=0)
    {
        res++;
        p1=a+vec*k1;
    }

    if (dlcmp(k2)>=0&&dlcmp(k2-1)<=0)
    {
        res++;

        if (res==1)
            p1=a+vec*k2;
        else
            p2=a+vec*k2;
    }

    return res;
}
```

7.4.6 求两圆公切线

```
//求两相离的圆的两条内共切线
void get_InCommonTangent(Circle c1,Circle c2,Point &s1,Point &e1,Point &s2,
    Point &e2)
{
    double l=dis(c1.o,c2.o);
    double d=l*c1.r/(c1.r+c2.r);
    double tmp=c1.r/d;
    tmp=fix(tmp);
    double theta=acos(tmp);
    Point vec=c2.o-c1.o;

    vec=vec.trunc(c1.r);
    s1=c1.o+vec.rotate(theta);
    s2=c1.o+vec.rotate(-theta);

    vec=c1.o-c2.o;
    vec=vec.trunc(c2.r);
    e1=c2.o+vec.rotate(theta);
    e2=c2.o+vec.rotate(-theta);
}

//求两相离的圆的两条外公切线
void get_OutCommonTangent(Circle c1,Circle c2,Point &s1,Point &e1,Point &s2,
    Point &e2)
{
    double l=dis(c1.o,c2.o);
    double d=fabs(c1.r-c2.r);
    double theta=acos(d/l);

    if (dlcmp(c1.r-c2.r)>0)
        swap(c1,c2);

    Point vec=c1.o-c2.o;
    vec=vec.trunc(c1.r);
    s1=c1.o+vec.rotate(theta);
    s2=c1.o+vec.rotate(-theta);
    vec=vec.trunc(c2.r);
    e1=c2.o+vec.rotate(theta);
    e2=c2.o+vec.rotate(-theta);
}
```

7.4.7 最小圆覆盖

测试报告：zoj1450 hysbz1336 1337 hdu3007 3932

```
#define eps 1e-8
#define MAX_P 2000
struct Point
{
    double x,y;

    Point operator-(Point &a)
    {
        Point t;

        t.x=x-a.x;
        t.y=y-a.y;

        return t;

    }
};
struct Circle
{
    double r;
    Point center;
};

struct Triangle
{
    Point t[3];
};

Point pt[MAX_P];   //点集
Circle c;          //最小圆

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

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

double triangle_area(Triangle tri)   //三角形距离
{
    Point v1=tri.t[1]-tri.t[0];
    Point v2=tri.t[2]-tri.t[0];

    return fabs(cross(v1,v2))/2;
}

Circle circumcircle_triangle(Triangle tri)  //三角形外接圆
{
    Circle res;

    double a,b,c,c1,c2;
    double xA,yA,xB,yB,xC,yC;

    a=distance(tri.t[0],tri.t[1]);
    b=distance(tri.t[1],tri.t[2]);
    c=distance(tri.t[2],tri.t[0]);

    res.r=a*b*c/triangle_area(tri)/4;

    xA=tri.t[0].x; yA=tri.t[0].y;
    xB=tri.t[1].x; yB=tri.t[1].y;
    xC=tri.t[2].x; yC=tri.t[2].y;

    c1=(xA*xA+yA*yA-xB*xB-yB*yB)/2;
    c2 = (xA*xA+yA*yA-xC*xC-yC*yC)/2;

    res.center.x=(c1*(yA-yC)-c2*(yA-yB))/ ((xA-xB)*(yA-yC)-(xA-xC)*(yA-yB));
    res.center.y = (c1*(xA-xC)-c2*(xA-xB))/ ((yA-yB)*(xA-xC)-(yA-yC)*(xA-xB)
        );

    return res;
}

Circle mincircle_triangle(int trinum,Triangle tri)
{
    Circle res;

    if (trinum==0)
        res.r=-2;
    else if (trinum==1)
    {
        res.center=tri.t[0];
        res.r=0;
    }
    else if (trinum==2)
    {
        res.center.x=(tri.t[0].x+tri.t[1].x)/2;
        res.center.y=(tri.t[0].y+tri.t[1].y)/2;
        res.r=distance(tri.t[0],tri.t[1])/2;
    }
    else if (trinum==3)
       res=circumcircle_triangle(tri);

    return res;
}

void mincircle_pointset(int m,int trinum,Triangle tri)  //求点集的最小覆盖圆
{
    int i,j;
    Point tmp;

    c=mincircle_triangle(trinum,tri);

    if (trinum==3)
        return;

    for (i=0;i<m;i++)
        if (distance(pt[i],c.center)>c.r)
        {
            tri.t[trinum]=pt[i];

            mincircle_pointset(i,trinum+1,tri);

            tmp=pt[i];

            for (j=i;j>=1;j--)
                pt[j]=pt[j-1];

            pt[0]=tmp;
        }
}

main()
{
    int n,i,f1,f2;
    Triangle tri;

    while (scanf("%d%d%d",&f1,&f2,&n)!=EOF)
    {
        for (i=0;i<n;i++)
            scanf("%lf%lf",&pt[i].x,&pt[i].y);

        mincircle_pointset(n,0,tri);
        printf("%lf %lf %lf\n",c.center.x,c.center.y,c.r);
    }
    return 0;
}
```

7.4.8 单位圆覆盖

测试报告：poj1981

```
#include<math.h>
```

```
#define eps 1e-8
#define MAX_P 505
const double r=1.0;//单位圆半径

struct Point
{
       double x,y;
};

Point pt[MAX_P];

double distance(Point a,Point b)
{
    return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
    //sqrt函数速度较慢，应尽量避免出现，此处可优化为距离的平方和的形式
}

Point find_center(Point a,Point b)
{
      Point v,mid,center;
      double d,s,ang;

      v.x=a.x-b.x;
      v.y=a.y-b.y;

      mid.x=(a.x+b.x)/2;
      mid.y=(a.y+b.y)/2;

      d=distance(a,mid);
      s=sqrt(r*r-d*d);       //优化为s=sqrt(r*r-d);

      if (fabs(v.y)<eps)
      {
         center.x=mid.x;
         center.y=mid.y+s;
      }
      else
      {
          ang=atan(-v.x/v.y);
          center.x=mid.x+s*cos(ang);
          center.y=mid.y+s*sin(ang);
      }

      return center;
}

main()
{

      int n,i,j,k,ans,cnt;
      double tmp;
      Point center;

      while (scanf("%d",&n),n)
      {
            for (i=0;i<n;i++)
                scanf("%lf%lf",&pt[i].x,&pt[i].y);

            ans=1;
            for (i=0;i<n;i++)
                for (j=i+1;j<n;j++)
                {
                    if (distance(pt[i],pt[j])>2*r)   //优化为distance(pt[i],pt[j])>2*2*r*r
                        continue;

                    cnt=0;
                    center=find_center(pt[i],pt[j]);

                    for (k=0;k<n;k++)
                        if (distance(pt[k],center)<=r+eps)
                            cnt++;

                   if (ans<cnt)
                      ans=cnt;
                }

          printf("%d\n",ans);
      }

      return 0;
}
```

7.5 模拟退火

7.5.1 求多边形费马点

测试报告：poj2420

```
#include<iostream>
#include<cstdio>
#include<cmath>
#define eps 1e-6
#define N 105
using namespace std;

struct Point
{
    double x,y;
};


double point_dis(Point a,Point b)
{
    return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));

}

double sum_dis(Point pt[],int n,Point o)
{
    double res=0;
    int i;

    for (i=0;i<n;i++)
        res+=point_dis(pt[i],o);

    return res;
}

double polygon_Fermatpoint(Point pln[],int n)
{
    Point cp,np,tmp;
    double min,step,d;
    int flag;

    cp=pln[0];       //cp保存当前更新后最优的费马点
    min=sum_dis(pln,n,cp);
    step=10000;     //选取坐标范围的最大值

    while (step>eps)
    {
        flag=1;
        while (flag)
        {
            flag=0;
            np=cp;

            tmp=cp,tmp.x+=step;
            d=sum_dis(pln,n,tmp);

            if (min>d)
                min=d, np=tmp, flag=1;

            tmp=cp,tmp.x-=step;
            d=sum_dis(pln,n,tmp);

            if (min>d)
```

```
                min=d, np=tmp,flag=1;

            tmp=cp,tmp.y+=step;
            d=sum_dis(pln,n,tmp);

            if (min>d)
                min=d, np=tmp,flag=1;

            tmp=cp,tmp.y-=step;
            d=sum_dis(pln,n,tmp);

            if (min>d)
                min=d, np=tmp,flag=1;

            cp=np;
        }

        step*=0.98;  //系数根据精度要求修改
    }

    return min;
}

main()
{
    int n,i;
    double min;
    Point pt[N];

    cin>>n;

    for (i=0;i<n;i++)
        cin>>pt[i].x>>pt[i].y;

    min=polygon_Fermatpoint(pt,n);

    printf("%.0f\n",min);

    return 0;
}
```

7.5.2 最小球覆盖

测试报告: poj2069 cf106E

```
#include<iostream>
#include<cstdio>
#include<cmath>
#define oo 1e20
#define eps 1e-10
#define N 105
using namespace std;

struct Point
{
    double x,y,z;
};

double dis(Point a,Point b)
{
    return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y)+(a.z-b.z)*(a.z-b.z))
        ;
}
```

```

int max_dis(Point pt[],int n,Point o)
{
    int i,res;
    double max,tmp;

    max=0;
    res=0;

    for (i=0;i<n;i++)
    {
        tmp=dis(pt[i],o);

        if (max<tmp)
        {
            max=tmp;
            res=i;
        }
    }

    return res;
}

int main()
{
    Point pt[N],o;
    int n,i,t;
    double dx,dy,dz,step,r,tmp;

    cin>>n;

        for (i=0;i<n;i++)
            cin>>pt[i].x>>pt[i].y>>pt[i].z;

        step=10000;  //step选取最大的坐标范围
        r=oo;

        if (n==1)
        {
            o.x=pt[0].x;
            o.y=pt[0].y;
            o.z=pt[0].z;
        }
        else
        {
            o.x=o.y=o.z=0;
            while (step>eps)
            {
                t=max_dis(pt,n,o);
                tmp=dis(pt[t],o);

                if (r>tmp)
                    r=tmp;

                dx=(pt[t].x-o.x)/tmp;
                dy=(pt[t].y-o.y)/tmp;
                dz=(pt[t].z-o.z)/tmp;

                o.x+=step*dx;
                o.y+=step*dy;
                o.z+=step*dz;

                step*=0.9993;   //系数的选取根据具体精度调整
                }
            }
            printf("%.6f %.6f %.6f\n",o.x,o.y,o.z);

    return 0;
}
```

7.6 三维几何

7.6.1 三维凸包

测试报告：hdu3662 poj3528

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<cmath>
#include<algorithm>
#define N 505
#define eps 1e-8
using namespace std;

struct Point
{
    double x,y,z;
    Point(){}
    Point (double px,double py,double pz):x(px),y(py),z(pz){}
    Point operator - (const Point p)
    {
        return Point(x-p.x,y-p.y,z-p.z);
    }
    Point operator * (const Point p)
    {
        return Point(y*p.z-z*p.y,z*p.x-x*p.z,x*p.y-y*p.x);
    }
    double operator ^ (const Point p)
    {
        return x*p.x+y*p.y+z*p.z;
    }
};

struct ConvexPolygon3D
{
    struct Face
    {
        int a,b,c;
        bool flag;
    };

    int n;
    Point pt[N];
    int tri_num;
    Face face[8*N];
    int g[N][N];

    double veclen(const Point &p)
    {
        return sqrt(p.x*p.x+p.y*p.y+p.z*p.z);
    }

    Point cross(const Point &a, const Point &b, const Point &c)
    {
        return Point((b.y-a.y)*(c.z-a.z)-(b.z-a.z)*(c.y-a.y),-((b.x-a.x)*(c.
           z-a.z)-(b.z-a.z)*(c.x-a.x)),(b.x-a.x)*(c.y-a.y)-(b.y-a.y)*(c.x-a.
           x));
    }

    double tri_area(Point a,Point b,Point c)
    {
        return veclen((a-c)*(b-c))/2;
    }

    double  tetrahedron_volume(Point a,Point b,Point c,Point d)
    {
        return ((b-a)*(c-a)^(d-a))/6;
    }

    double dlcmp(Point &p,Face &f)
    {
        Point m=pt[f.b]-pt[f.a];
        Point n=pt[f.c]-pt[f.a];
        Point t=p-pt[f.a];

        return (m*n)^t;
    }

    void deal(int a,int b,int p)
    {
        int f=g[a][b];
        Face add;

        if (face[f].flag)
        {
            if (dlcmp(pt[p],face[f])>eps)
                dfs(p,f);
            else
            {
                add.a=b;
                add.b=a;
                add.c=p;
                add.flag=1;
                g[p][b]=g[a][p]=g[b][a]=tri_num;
                face[tri_num++]=add;
            }
        }
    }

    void dfs(int p,int now)
    {
        face[now].flag=0;
        deal(face[now].b,face[now].a,p);
        deal(face[now].c,face[now].b,p);
        deal(face[now].a,face[now].c,p);
    }

    bool same(int s,int t)
    {
        Point &a=pt[face[s].a];
        Point &b=pt[face[s].b];
        Point &c=pt[face[s].c];

        bool res=fabs(tetrahedron_volume(a,b,c,pt[face[t].a]))<eps&&
                 fabs(tetrahedron_volume(a,b,c,pt[face[t].b]))<eps&&
                 fabs(tetrahedron_volume(a,b,c,pt[face[t].c]))<eps;

        return res;
    }

    void solve()
    {
        int i,j,tmp;
        Face add;
        bool flag;

         tri_num=0;

        if (n<4)
            return;

        flag=true;
        for (i=1;i<n;i++)
            if (veclen((pt[0]-pt[1])*(pt[1]-pt[i]))>eps)
            {
                swap(pt[2],pt[i]);
                flag=false;
                break;
            }

        if (flag)
            return;

        flag=true;
        for (i=2;i<n;i++)
            if (fabs((pt[0]-pt[1])*(pt[1]-pt[2])^(pt[0]-pt[i]))>eps)
            {
                swap(pt[3],pt[i]);
                flag=false;
                break;
            }

        if (flag)
            return;

        flag=true;
        for (i=3;i<n;i++)
            if (veclen(pt[0]-pt[i])>eps)
            {
                swap(pt[1],pt[i]);
                flag=false;
                break;
            }

        if (flag)
            return;

        for (i=0;i<4;i++)
        {
            add.a=(i+1)%4;
            add.b=(i+2)%4;
            add.c=(i+3)%4;
            add.flag=true;

            if (dlcmp(pt[i],add)>0)
                swap(add.b,add.c);

            g[add.a][add.b]=g[add.b][add.c]=g[add.c][add.a]=tri_num;
            face[tri_num++]=add;
        }

        for (i=4;i<n;i++)
            for (j=0;j<tri_num;j++)
                if (face[j].flag&&dlcmp(pt[i],face[j])>eps)
                {
                    dfs(i,j);
                    break;
                }

        tmp=tri_num;
        for (i=tri_num=0;i<tmp;i++)
            if (face[i].flag)
                face[tri_num++]=face[i];
    }

    double area()
    {
        double res=0;

        if (n==3)
        {
            Point p=cross(pt[0],pt[1],pt[2]);
            res=veclen(p)/2;
        }
        else
        {
            for (int i=0;i<tri_num;i++)
                res+=tri_area(pt[face[i].a],pt[face[i].b],pt[face[i].c]);
        }

        return res;
    }

    double volume()
    {
        double res=0;
        Point tmp(0,0,0);

        for (int i=0;i<tri_num;i++)
            res+=tetrahedron_volume(tmp,pt[face[i].a],pt[face[i].b],pt[face[
                i].c]);

        return fabs(res);
    }

      Point get_center()  //凸包重心
      {
        Point res(0,0,0),o(0,0,0),p;
        double sum,vol;
            int i;

         sum=0;
           for (i=0;i<tri_num;i++)
          {
                vol=tetrahedron_volume(o,pt[face[i].a],pt[face[i].b],pt[face
                    [i].c]);
                sum+=vol;
                p=(pt[face[i].a]+pt[face[i].b]+pt[face[i].c])/4;
                p.x*=vol; p.y*=vol; p.z*=vol;
                res=res+p;
            }

            res=res/sum;
            return res;
      }

    int triangle_num()
    {
        return tri_num;
    }

    int polygon_num()
    {
        int i,j,res,flag;

        res=0;
        for (i=0;i<tri_num;i++)
        {
            flag=1;
            for (j=0;j<i;j++)
                if (same(i,j))
                {
                    flag=0;
                    break;
                }
            res+=flag;
        }

        return res;
    }
};

ConvexPolygon3D hull;

//点p到平面abc的距离
double dis_point_face(Point p,Point a,Point b,Point c)
{
    Point vec=(b-a)*(c-a);
    Point t=a-p;
    double tmp=(vec^t)/(vec.len()*t.len());

    return fabs(t.len()*tmp);
}

int main()
{
    int i;

    while(scanf("%d",&hull.n)!=EOF)
    {

         for(i=0;i<hull.n;i++)
           scanf("%lf%lf%lf",&hull.pt[i].x,&hull.pt[i].y,&hull.pt[i].z);
         hull.solve();

         printf("%.3f\n",hull.area());
    }

    return 0;
}
```

7.6.2 求两球体积并

测试报告：zoj3500

```
#define PI (acos(-1.0))

struct Point
{
    double x,y,z;
};

double dis(Point a,Point b)
{
    return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y)+(a.z-b.z)*(a.z-b.z))
        ;
}

double ball_volume_combination(Point o1,double r1,Point o2,double r2)
{
    double d,R,r,p,l,H,h,res;
    R=max(r1,r2),r=min(r1,r2);
    d=dis(o1,o2);
    res=PI*(R*R*R+r*r*r)*4/3;

    if (R+r>d)
    {
        if (R-r<=d)
        {
            p=(R+r+d)/2;
            l=sqrt(p*(p-R)*(p-r)*(p-d))*2/d;
            H=sqrt(R*R-l*l);
            h=sqrt(r*r-l*l);

            res-=PI*(R*R*R*2/3-R*R*H+H*H*H/3);

            if (R*R-r*r<=d*d)
                res-=PI*(r*r*r*2/3-r*r*h+h*h*h/3);
            else
                res-=PI*r*r*r*4/3-PI*(r*r*r*2/3-r*r*h+h*h*h/3);
        }
        else
            res-=PI*r*r*r*4/3;
    }

    return res;
}
```

7.7 三维仿射变换

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<vector>
#include<stack>
#include<cmath>
#define eps 1e-6
#define SZ 10
using namespace std;

const double PI=acos(-1.0);
int dlcmp(double x) {return x<-eps?-1:x>eps;}
double sqr(double x) {return x*x;}

struct Matrix
{
    double m[SZ][SZ];

    void Identity()
    {
        for (int i=0;i<SZ;i++)
            for (int j=0;j<SZ;j++)
                m[i][j]=(i==j?1.0:0.0);
    }

    Matrix (){Identity();}

    Matrix operator * (const Matrix &a)
    {
        Matrix res;

        for (int i=0;i<SZ;i++)
            for (int j=0;j<SZ;j++)
            {
                res.m[i][j]=0;
                for (int k=0;k<SZ;k++)
                    res.m[i][j]+=m[i][k]*a.m[k][j];
            }

        return res;
    }
};

struct Point
{
    double x,y,z;

    Point (){}
    Point (double a,double b,double c):x(a),y(b),z(c){}

    Point operator + (const Point &a) const {return Point(x+a.x,y+a.y,z+a.z)
        ;}
    Point operator - (const Point &a) const {return Point(x-a.x,y-a.y,z-a.z)
        ;}

    Point operator * (const Matrix &a) const
    {
        Point res;

        res.x=x*a.m[0][0]+y*a.m[1][0]+z*a.m[2][0]+a.m[3][0];
        res.y=x*a.m[0][1]+y*a.m[1][1]+z*a.m[2][1]+a.m[3][1];
        res.z=x*a.m[0][2]+y*a.m[1][2]+z*a.m[2][2]+a.m[3][2];

        return res;
    }

    void norm()
    {
        double len=sqrt(sqr(x)+sqr(y)+sqr(z));
        x/=len; y/=len; z/=len;
    }
};

stack<Matrix>sm;
stack<int>sn;

Matrix pow(Matrix a,int k)
{
    Matrix res;

    while (k)
    {
        if (k&1)
            res=res*a;
        a=a*a;
        k/=2;
    }

    return res;
}

Matrix get_trans(Point v)
{
    Matrix res;
    res.m[3][0]=v.x; res.m[3][1]=v.y; res.m[3][2]=v.z;

    return res;
}

Matrix get_scale(Point v)
{
    Matrix res;
    res.m[0][0]=v.x; res.m[1][1]=v.y; res.m[2][2]=v.z;

    return res;
}

Matrix get_rotate(Point v,double ang)
{
    Matrix res;
    double d=ang/180*PI;
    v.norm();

    res.m[0][0]=(1-cos(d))*v.x*v.x+cos(d);
    res.m[0][1]=(1-cos(d))*v.x*v.y+sin(d)*v.z;
    res.m[0][2]=(1-cos(d))*v.x*v.z-sin(d)*v.y;
    res.m[1][0]=(1-cos(d))*v.y*v.x-sin(d)*v.z;
    res.m[1][1]=(1-cos(d))*v.y*v.y+cos(d);
    res.m[1][2]=(1-cos(d))*v.y*v.z+sin(d)*v.x;
    res.m[2][0]=(1-cos(d))*v.z*v.x+sin(d)*v.y;
    res.m[2][1]=(1-cos(d))*v.z*v.y-sin(d)*v.x;
    res.m[2][2]=(1-cos(d))*v.z*v.z+cos(d);

    return res;
```

```
}

int main()
{
    int i,n,m;
    Matrix cur,tmp;
    double ang;
    Point p,v;
    char str[100];

    while (scanf("%d",&n),n)
    {
        while (!sm.empty())
            sm.pop();
        while (!sn.empty())
            sn.pop();

        sm.push(Matrix());
        sn.push(-1);

        while (!sm.empty()&&!sn.empty())
        {
            scanf("%s",str);

            if (!strcmp(str,"translate"))
            {
                scanf("%lf%lf%lf",&v.x,&v.y,&v.z);
                tmp=get_trans(v);
                sm.top()=sm.top()*tmp;
            }
            else if (!strcmp(str,"scale"))
            {
                scanf("%lf%lf%lf",&v.x,&v.y,&v.z);
                tmp=get_scale(v);
                sm.top()=sm.top()*tmp;
            }
            else if (!strcmp(str,"rotate"))
            {
                scanf("%lf%lf%lf%lf",&v.x,&v.y,&v.z,&ang);
                tmp=get_rotate(v,ang);
                sm.top()=sm.top()*tmp;
            }
            else if (!strcmp(str,"repeat"))
            {
                scanf("%d",&m);
                sn.push(m);
                sm.push(Matrix());
            }
            else if (!strcmp(str,"end"))
            {
                m=sn.top();
                sn.pop();
                tmp=sm.top();
                sm.pop();
                if (m==-1)
                    cur=tmp;
                else
                {
                    tmp=pow(tmp,m);
                    sm.top()=sm.top()*tmp;
                }

            }
        }

        for (i=0;i<n;i++)
        {
            scanf("%lf%lf%lf",&p.x,&p.y,&p.z);
            p=p*cur;
            printf("%.2f %.2f %.2f\n",p.x+eps,p.y+eps,p.z+eps);
        }
        printf("\n");
    }

    return 0;
}
```

8 其他

8.1 矩阵乘

注意初始化sz的大小

```
const int N = 60;
typedef long long LOL;
LOL mod=1000000007ll;
LOL c[N][N],a[N][N],b[N][N],g[N][N];
int sz;
void matcopy(LOL a[N][N],LOL b[N][N])
{
    for(int i=1;i<=sz;i++){
        for(int j=1;j<=sz;j++){
            a[i][j]=b[i][j];
        }
    }
}
void matmul(LOL a[N][N],LOL b[N][N])
{
    memset(c,0,sizeof(c));
    for(int i=1;i<=sz;i++){
        for(int j=1;j<=sz;j++){
            if(a[i][j]){
                for(int k=1;k<=sz;k++){
                    c[i][k]+=a[i][j]*b[j][k];
                    if(c[i][k]>mod) c[i][k]%=mod;
                }
            }
        }
    }
    matcopy(a,c);
}
void matpow(LOL a[N][N],LOL x)
{
    memset(b,0,sizeof(b));
    for(int i=1;i<=sz;i++){
        b[i][i]=1;
    }
    while(x!=0)
    {
        if(x%2) matmul(b,a);
        matmul(a,a);
        x/=2;
    }
    matcopy(a,b);
}
```

8.2 平面最近点对

测试报告：poj3714 hdu1007

```
#include<iostream>
#include<cstdio>
#include<cmath>
#include<algorithm>
#define oo 1e30
#define eps 1e-8
#define N 100005
using namespace std;
int stack[N];

struct Point
{
    double x,y;
};

Point pt[N];

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

bool cmp_x(Point a,Point b)
{
    if (fabs(a.x-b.x)<eps)
        return a.y<b.y;
    else
        return a.x<b.x;
}

bool cmp_y(int a,int b)
{
    return pt[a].y<pt[b].y;
}

double find_minimum_points(Point pt[],int l,int r)
{
    int i,j,top,mid;
    double d1,d2,res;

    if (r-l==1)
        return dis(pt[l],pt[r]);
    else if (r-l==2)
        return min(dis(pt[l],pt[l+1]),min(dis(pt[l],pt[r]),dis(pt[l+1],pt[r
           ])));
    else
    {
        mid=(l+r)/2;
        d1=find_minimum_points(pt,l,mid);
        d2=find_minimum_points(pt,mid+1,r);
        res=min(d1,d2);

        top=0;
        for (i=mid;i>=l&&fabs(pt[mid+1].x-pt[i].x)<=res;i--)
            stack[top++]=i;
        for (i=mid+1;i<=r&&fabs(pt[mid].x-pt[i].x)<=res;i++)
            stack[top++]=i;

        if (top>0)
            sort(stack,stack+top,cmp_y);

        for (i=0;i<top;i++)
            for (j=i+1;pt[stack[j]].y-pt[stack[i]].y<=res&&j<top;j++)
                res=min(res,dis(pt[stack[i]],pt[stack[j]]));

        return res;
    }
}

double minimium_distance_pointset(Point pt[],int n)
{
    int i;

    sort(pt,pt+n,cmp_x);
    for (i=1;i<n;i++)

        if (fabs(pt[i].x-pt[i-1].x)<eps&&fabs(pt[i].y-pt[i-1].y)<eps)
            return 0.0;

    return find_minimum_points(pt,0,n-1);
}

int main()
{
    int i,n;
    double ans;

    while (cin>>n,n)
    {

        for (i=0;i<n;i++)
            scanf("%lf%lf",&pt[i].x,&pt[i].y);

        ans=minimium_distance_pointset(pt,n);
        printf("%.2f\n",ans/2);
    }

    return 0;
}
```

8.3 读入外挂

```
inline int ScanInt(void) {
        int r = 0, c, d;
        while (!isdigit(c = getchar()) && c != '-');
        if (c != '-') r = c - '0'; d = c;
        while ( isdigit(c = getchar())) r = r * 10 + c - '0';
        return d=='-'?-r:r;
    }

inline Int64 ScanInt(void) {
        Int64 r = 0, c, d;
        while (!isdigit(c = getchar()) && c != '-');
        if (c != '-') r = c - '0'; d = c;
        while ( isdigit(c = getchar())) r = r * 10ll + c - '0';
        return d=='-'?-r:r;
    }
```

8.4 JAVA分数类

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

class BigFraction {

    BigFraction() {
        numerator = BigInteger.ZERO;
        Denominator = BigInteger.ONE;
    }

    BigFraction(BigInteger _numerator, BigInteger _Denominator) {
        numerator = _numerator;
        Denominator = _Denominator;
    }

    public BigFraction add(BigFraction bf) {
        BigInteger n = numerator.multiply(bf.Denominator).add(Denominator.
           multiply(bf.numerator));
        BigInteger d = Denominator.multiply(bf.Denominator);

        BigFraction ret = new BigFraction(n, d);
        ret.simplify();

        return ret;
    }

    public BigFraction subtract(BigFraction bf) {
        BigInteger n = numerator.multiply(bf.Denominator).subtract(
           Denominator.multiply(bf.numerator));
        BigInteger d = Denominator.multiply(bf.Denominator);

        BigFraction ret = new BigFraction(n, d);
        ret.simplify();

        return ret;
    }

    public BigFraction multiply(BigFraction bf) {
        BigInteger n = numerator.multiply(bf.numerator);
        BigInteger d = Denominator.multiply(bf.Denominator);

        BigFraction ret = new BigFraction(n, d);
        ret.simplify();

        return ret;
    }

    public BigFraction divide(BigFraction bf) {
        BigInteger n = numerator.multiply(bf.Denominator);
        BigInteger d = Denominator.multiply(bf.numerator);

        BigFraction ret = new BigFraction(n, d);
        ret.simplify();

        return ret;
    }

    public int compareTo(BigFraction bf) {
        BigInteger ret = numerator.multiply(bf.Denominator).subtract(
           Denominator.multiply(bf.numerator));
        return ret.compareTo(BigInteger.ZERO);
    }


    public BigFraction abs() {
        BigFraction ret = new BigFraction(numerator.abs(), Denominator.abs()
            );
        return ret;
    }

    public BigFraction negate() {
        numerator = numerator.negate();
        return this;
    }

    public boolean isInteger() {
        return Denominator.equals(BigInteger.ONE);
    }

    // for simplify numerator and Denominator
    private void simplify() {
        BigInteger g = numerator.gcd(Denominator);
        numerator = numerator.divide(g);
        Denominator = Denominator.divide(g);
        if (Denominator.compareTo(BigInteger.ZERO) < 0) {
            numerator = numerator.negate();
            Denominator = Denominator.negate();
        }
    }

    public BigInteger numerator;
    public BigInteger Denominator;
    public static BigFraction ZERO = new BigFraction(BigInteger.ZERO,
        BigInteger.ONE);
    public static BigFraction ONE = new BigFraction(BigInteger.ONE,
        BigInteger.ONE);
}
public class Main
{
    static BigFraction c[][]=new BigFraction[120][120];
    static BigFraction b[][]=new BigFraction[120][120];
    public static void main(String args[])
    {
        for(int i=0;i<=101;i++){
            for(int j=0;j<=101;j++){
                c[i][j]=BigFraction.ZERO;
                b[i][j]=BigFraction.ZERO;
            }
        }
        for(int i=0;i<=101;i++){
            c[i][0]=BigFraction.ONE;
        }
        for(int i=1;i<=101;i++){
            for(int j=1;j<=i;j++){
                c[i][j]=c[i-1][j-1].add(c[i-1][j]);
            }
        }
        b[0][1]=BigFraction.ONE;
        for(int i=1;i<=100;i++){
            for(int j=1;j<=i+1;j++){
                b[i][j]=c[i+1][j];
            }
            for(int j=0;j<i;j++){
                for(int k=1;k<=j+1;k++){
                    b[i][k]=b[i][k].subtract(c[i+1][i+1-j].multiply(b[j][k])
                        );
                }
            }
            for(int j=1;j<=i+1;j++){
                b[i][j]=b[i][j].divide(c[i+1][1]);
            }
        }
        Scanner cin=new Scanner(new BufferedInputStream(System.in));
        BigInteger n;
        int m;
        while(cin.hasNext())
        {
            n=cin.nextBigInteger();
            m=cin.nextInt();
            BigFraction ans=BigFraction.ZERO;
            BigInteger tmp=n;
            for(int i=1;i<=m+1;i++){
                BigFraction tx = BigFraction.ONE;
                tx.numerator=tmp;
                tx.Denominator=BigInteger.ONE;
                ans=ans.add(tx.multiply(b[m][i]));
                tmp=tmp.multiply(n);
            }
            System.out.println(ans.numerator);
        }
    }

}
```

8.5 魔方

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<vector>
#include<string>
#include<queue>
#include<cmath>
using namespace std;

int rubik[55];

//魔方初始化。根据题目要求决定是每个面的每个格子一个编号，还是每个面的格子标同一个编号
void init() {for (int i=1;i<=54;i++) rubik[i]=i;}

//对每个面进行顺时针旋转度90
void rotate(int x1,int x2,int x3,int x4,int x5,int x6,int x7,int x8,int x9)
{
    int a[55];

    memcpy(a,rubik,sizeof(a));
    rubik[x1]=a[x7]; rubik[x2]=a[x4]; rubik[x3]=a[x1];
    rubik[x4]=a[x8]; rubik[x5]=a[x5]; rubik[x6]=a[x2];
    rubik[x7]=a[x9]; rubik[x8]=a[x6]; rubik[x9]=a[x3];
}

//侧边顺时针选装度90
void trans(int x1,int x2,int x3,int x4,int x5,int x6,int x7,int x8,int x9,
    int x10,int x11,int x12)
{
    int a[55];

    memcpy(a,rubik,sizeof(a));
    rubik[x1]=a[x10]; rubik[x2]=a[x11]; rubik[x3]=a[x12];
    rubik[x4]=a[x1];  rubik[x5]=a[x2];  rubik[x6]=a[x3];
    rubik[x7]=a[x4];  rubik[x8]=a[x5];  rubik[x9]=a[x6];
    rubik[x10]=a[x7]; rubik[x11]=a[x8]; rubik[x12]=a[x9];
}

//从上面看去，顺时针旋转第一层
void turn_U()
{
    rotate(1,2,3,4,5,6,7,8,9);
    trans(30,29,28,21,20,19,12,11,10,39,38,37);
}
//从上面看去，顺时针旋转第二层
void turn_X()
{
    trans(33,32,31,24,23,22,15,14,13,42,41,40);
}
//从上面看去，顺时针旋转第三层
void turn_D()
{
    rotate(48,47,46,51,50,49,54,53,52);
    trans(36,35,34,27,26,25,18,17,16,45,44,43);
}
//从右面看去，顺治针旋转第一层
void turn_R()
{
    rotate(19,20,21,22,23,24,25,26,27);
    trans(9,6,3,28,31,34,46,49,52,18,15,12);
```

```
}
//从右面看去，顺治针旋转第二层
void turn_Y()
{
    trans(8,5,2,29,32,35,47,50,53,17,14,11);
}
//从右面看去，顺治针旋转第三层
void turn_L()
{
    rotate(39,38,37,42,41,40,45,44,43);
    trans(7,4,1,30,33,36,48,51,54,16,13,10);
}
//从前面看去，顺治针旋转第一层
void turn_F()
{
    rotate(10,11,12,13,14,15,16,17,18);
    trans(7,8,9,19,22,25,52,53,54,45,42,39);
}
//从前面看去，顺治针旋转第二层
void turn_Z()
{
    trans(4,5,6,20,23,26,49,50,51,44,41,38);
}
//从前面看去，顺治针旋转第三层
void turn_B()
{
    rotate(30,29,28,33,32,31,36,35,34);
    trans(1,2,3,21,24,27,46,47,48,43,40,37);
}

//hdu4397：询问操作后是否与初始状态完全相同（每个面的每个格子都完全一样）
int main()
{
    int i,j,ans,ys;
    char str[1000];

    ys=0;
    freopen("data.in","r",stdin);
    while (scanf("%s",str)!=EOF)
    {
        init();
        for (i=0;str[i];i++)
            switch (str[i])
            {
                case 'U':
                    turn_U();
                break;
                case 'u':
                    turn_U();turn_U();turn_U();
                break;
                case 'X':
                    turn_X();
                break;
                case 'x':
                    turn_X();turn_X();turn_X();
                break;
                case 'D':
                    turn_D();turn_D();turn_D();
                break;
                case 'd':
                    turn_D();
                break;
                case 'R':
                    turn_R();
                break;
                case 'r':
                    turn_R();turn_R();turn_R();
                break;
                case 'Y':
                    turn_Y();
                break;
                case 'y':
                    turn_Y();turn_Y();turn_Y();
                break;
                case 'L':
                    turn_L();turn_L();turn_L();
                break;
                case 'l':
                    turn_L();
                break;
                case 'F':
                    turn_F();
                break;
                case 'f':
                    turn_F();turn_F();turn_F();
                break;
                case 'Z':
                    turn_Z();turn_Z();turn_Z();
                break;
                case 'z':
                    turn_Z();
                break;
                case 'B':
                    turn_B();turn_B();turn_B();
                break;
                case 'b':
                    turn_B();
                break;
            }

        ans=1;
        for (i=1;i<=54;i++)
            if (rubik[i]!=i)
            {
                ans=0;
                break;
            }

        ys++;
        if (ys>1)
            printf("\n");

        if (ans)
            printf("Yes\n");
        else
            printf("No\n");
    }

    return 0;
}
```

8.6 Hashmap

GCC中的hash_map定义在<ext/hash_map>文件，namespace __gnu_cxx中。要定义一个hash_map<int, int>非常简单：

```
#include <ext/hash_map>
using namespace __gnu_cxx;
hash_map<int, int> hm;
```

在使用map时，如果我们想要改变元素顺序，或以自定义的struct/class作为key的时候，可以设定map第三个模板参数（默认是less<Key>，即operator<）。对于hash_map，我们需要设定其第三个(hash<Key>)和第四个模板参数(equal_to<Key>, operator==)。

```
typedef long long my_type;
typedef int any_type;
struct my_hash {
    size_t operator()(const my_type& key) const {
        return (key >> 32) ^ key;
    }
};
struct my_equal_to {
    bool operator()(const my_type& lhs, const my_type& rhs) const {
        return lhs == rhs;
    }
};
hash_map<my_type, any_type, my_hash, my_equal_to> my_hash_map;
```