HDU 2063 二分图匹配\_HK

#include<bits/stdc++.h>

**using** **namespace** std;

//#include<ext/rope>

//using namespace \_\_gnu\_cxx

//#include<ext/pb\_ds/priority\_queue.hpp>

//using namespace \_\_gnu\_pbds;

#define lowbit(x) (x&-x)

#define pb(x) push\_back(x)

#define all(x) (x).begin(),(x).end()

#define clr(a,b) memset(a,b,sizeof(a))

#define caze(T) for(scanf("%d",&T);T;T--)

#define debug cout<<"???"<<Endl

#define inf (1<<30)

#define Endl ('\n')

#define ll long long

#define pii pair<int,int>

#define ull unsigned long long

#define IOS ios::sync\_with\_stdio(0),cin.tie(0),cout.tie(0)

**const** **int** maxp=5e5+7;

**const** **int** maxn=4e3+7;

**struct** EDGE{**int** v,nxt;}edge[1000010];

**int** tot;**int** head[maxn];

**void** AE(**int** u,**int** v){edge[tot]={v,head[u]},head[u]=tot++;}

**int** mx[maxn],my[maxn];

**int** dx[maxn],dy[maxn];

**bool** vis[maxn];

**int** nx,ny;

**int** dis;

**bool** bfs()

{

queue<**int**>q;

dis=inf;

clr(dx,-1);

clr(dy,-1);

**for**(**int** i=1;i<=nx;++i)

**if**(mx[i]==-1)

q.push(i),dx[i]=0;

**while**(!q.empty())

{

**int** u=q.front();q.pop();

**if**(dx[u]>dis) **break**;

**for**(**int** i=head[u],v;~i;i=edge[i].nxt)

{

v=edge[i].v;

**if**(dy[v]==-1)

{

dy[v]=dx[u]+1;

**if**(my[v]==-1)

dis=dy[v];

**else**

{

dx[my[v]]=dy[v]+1;

q.push(my[v]);

}

}

}

}

**return** dis!=inf;

}

**bool** dfs(**int** u)

{

**for**(**int** i=head[u],v;~i;i=edge[i].nxt)

{

v=edge[i].v;

**if**(vis[v]||dy[v]!=dx[u]+1) **continue**;

vis[v]=1;

**if**(my[v]!=-1&&dy[v]==dis) **continue**;

**if**(my[v]==-1||dfs(my[v]))

{

my[v]=u;

mx[u]=v;

**return** 1;

}

}

**return** 0;

}

**int** HK()

{

**int** ret=0;

clr(mx,-1);

clr(my,-1);

**while**(bfs())

{

clr(vis,0);

**for**(**int** i=1;i<=nx;++i)

ret+=(mx[i]==-1&&dfs(i));

}

**return** ret;

}

**int** main()

{

**int** k;

**while**(~scanf("%d",&k)&&k)

{

scanf("%d%d",&nx,&ny);

tot=0;clr(head,-1);

**for**(**int** i=0,u,v;i<k;++i)

{

scanf("%d%d",&u,&v);

AE(u,v);

}

printf("%d**\n**",HK());

}

}

二分图匹配\_西方算法(匈牙利)

**const** **int** maxn=xxx;

vector<**int**>v[maxn];

**bool** used[maxn];

**int** lef[maxn];

**void** ini()

{

**for**(**int** i=1;i<=H;++i)

v[i].clear();

clr(lef,-1);

}

**bool** dfs(**int** x)

{

**for**(**auto** c:v[x])

{

**if**(used[c]==0)

{

used[c]=1;

**if**(lef[c]==-1||dfs(lef[c]))

{

lef[c]=x;

**return** 1;

}

}

}

**return** 0;

}

**int** solve(**int** n)

{

**int** ret=0;

**for**(**int** i=1;i<=n;++i)

{

clr(used,-1);

ret+=dfs(i);

}

**return** ret;

}

奔小康赚大钱,KM裸

#include<bits/stdc++.h>

**using** **namespace** std;

#define clr(a,b) memset(a,b,sizeof(a))

#define ll long long

#define ull unsigned long long

#define lowbit(x) (x&-x)

#define pb(x) push\_back(x)

#define IOS ios::sync\_with\_stdio(0),cin.tie(0),cout.tie(0)

#define inf (1<<30)

#define Endl ('\n')

**const** **int** N=333;

**int** n,nx,ny;

**int** link[N],lx[N],ly[N],slack[N];

**int** visx[N],visy[N],w[N][N];

**bool** dfs(**int** x)

{

visx[x]=1;

**for**(**int** y=0;y<ny;++y)

{

**if**(visy[y])

**continue**;

**int** tp=lx[x]+ly[y]-w[x][y];

**if**(tp==0)

{

visy[y]=1;

**if**(link[y]==-1||dfs(link[y]))

{

link[y]=x;

**return** 1;

}

}

**else** **if**(slack[y]>tp)

slack[y]=tp;

}

**return** 0;

}

**int** KM()

{

clr(link,-1);

clr(ly,0);

**for**(**int** i=0;i<nx;++i)

{

lx[i]=-inf;

**for**(**int** j=0;j<ny;++j)

**if**(w[i][j]>lx[i])

lx[i]=w[i][j];

}

**for**(**int** x=0;x<nx;++x)

{

**for**(**int** i=0;i<ny;++i)

slack[i]=inf;

**while**(1)

{

clr(visx,0);

clr(visy,0);

**if**(dfs(x))

**break**;

**int** d=inf;

**for**(**int** i=0;i<ny;++i)

**if**(!visy[i]&&d>slack[i])

d=slack[i];

**for**(**int** i=0;i<nx;++i)

**if**(visx[i])

lx[i]-=d;

**for**(**int** i=0;i<ny;++i)

**if**(visy[i])

ly[i]+=d;

**else**

slack[i]-=d;

}

}

**int** ret=0;

**for**(**int** i=0;i<ny;++i)

**if**(link[i]!=-1)

ret+=w[link[i]][i];

**return** ret;

}

**void** solve(**int** n)

{

**for**(**int** i=0;i<n;++i)

**for**(**int** j=0;j<n;++j)

scanf("%d",&w[i][j]);

nx=ny=n;

printf("%d**\n**",KM());

}

**int** main()

{

**int** n;

**while**(~scanf("%d",&n))

solve(n);

}

计算几何

圆和多边形面积交:HDU-5462 给出个圆,给个多边形,求交集面积.输入是线,这题要判断线的方向.

#include<bits/stdc++.h>

#include<ext/rope>

**using** **namespace** std;

#define clr(a,b) memset(a,b,sizeof(a))

#define ll long long

#define ull unsigned long long

#define lowbit(x) (x&-x)

#define pb(x) push\_back(x)

#define IOS ios::sync\_with\_stdio(0),cin.tie(0),cout.tie(0)

#define inf (1<<30)

#define caze(T) for(scanf("%d",&T);T;T--)

#define Endl ('\n')

**const** **double** pi=acos(-1.0);

**const** **double** eps=1e-8;

**int** dcmp(**double** x){**return** fabs(x)<=eps?0:(x<0?-1:1);}

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

**struct** point

{

**double** x,y,id;

point(){}

point(**double** x,**double** y,**int** id=-1):x(x),y(y),id(id) {}

point **operator**-(**const** point w)**const** {**return** point(x-w.x,y-w.y);}

point **operator**+(**const** point w)**const** {**return** point(x+w.x,y+w.y);}

**double** **operator**\*(**const** point& w)**const** {**return** x\*w.x+y\*w.y;}

point **operator**\*(**double** a) {**return** point(x\*a,y\*a);}

**double** **operator**^(**const** point& w)**const** {**return** x\*w.y-y\*w.x;}

point **operator**/(**double** a) {**return** point(x/a,y/a);}

**friend** ostream &**operator**<<(ostream& out,**const** point& w) {out<<'('<<w.x<<','<<w.y<<')';**return** out;}

**void** input(){scanf("%lf%lf",&x,&y);}

**double** len2(){**return** x\*x+y\*y;}

**double** len(){**return** sqrt(x\*x+y\*y);}

point change\_len(**double** r)

{

**double** l=len();

**if**(dcmp(l)==0) **return** \***this**;

r/=l;

**return** point(x\*r,y\*r);

}

};

**inline** **double** cross(**const** point& A,**const** point& B){**return** A.x\*B.y-B.x\*A.y;}

**inline** **double** dot(**const** point& q,**const** point& w){**return** q.x\*w.x+q.y\*w.y;}

**inline** **double** Xmul(**const** point& A,**const** point& B,**const** point& C){**return** cross(C-A,B-A);}

**inline** **double** dis(**const** point& q,**const** point& w){**return** sqrt(dot(q-w,q-w));}

**inline** **double** rad(**const** point& A,**const** point& B){**return** fabs(atan2(fabs(cross(A,B)),dot(A,B)));}

**int** Andrew(**int** n,point \*st,point \*ed)

{

sort(st,st+n,[](**const** point& A,**const** point& B)->**bool**{**return** A.x==B.x?A.y<B.y:A.x<B.x;});

**int** tot=0;

**for** (**int** i = 0; i < n; ++i)

{

**while**(tot>1&&cross(ed[tot-1]-ed[tot-2],st[i]-ed[tot-2])<=0) tot--;

ed[tot++]=st[i];

}

**int** tp=tot;

**for** (**int** i = n - 2; ~i; --i)

{

**while**(tot>tp&&cross(ed[tot-1]-ed[tot-2],st[i]-ed[tot-2])<=0) tot--;

ed[tot++]=st[i];

}

tot-=(n>1);

**return** tot;

}

**double** Area(**int** n,point \*p)

{

**double** S=0;

**for** (**int** i = 1; i < n - 1; ++i)

S+=fabs(Xmul(p[0],p[i],p[i+1]));

**return** S/2;

}

**struct** Line

{

point u,v;

**double** k;

Line(){}

Line(point u,point v):u(u),v(v){k=atan2(v.y-u.y,v.x-u.x);}

Line(point u,**double** k):u(u),k(k){v=u+(dcmp(k-pi/2)?point(1,tan(k)):point(0,1));}

**void** input(){u.input();v.input();get\_angle();}

**void** get\_angle(){k=atan2(v.y-u.y,v.x-u.x);}

**double** len(){**return** dis(u,v);}

**double** pdis(point w) {**return** fabs(cross(w-u,v-u)/len());}

point **operator**&(**const** Line& b)**const**

{

point ret=u;

**double** t=(cross(u-b.u,b.u-b.v))/cross(u-v,b.u-b.v);

ret.x+=(v.x-u.x)\*t;

ret.y+=(v.y-u.y)\*t;

**return** ret;

}

point project(**const** point& w)**const**{**return** u+(((v-u)\*((v-u)\*(w-u)))/(v-u).len2());}

**friend** ostream &**operator**<<(ostream &out,**const** Line& w){out<<w.u<<"->"<<w.v;**return** out;}

};

Line Q[100010];

**void** Hpi(**int** n,Line \*line,point \*res,**int** &resn)

{

**for** (**int** i = 0; i < n; ++i) line[i].get\_angle();

**int** tot=n;

sort(line,line+n,[](**const** Line& A,**const** Line& B)->**bool**{**return** fabs(A.k-B.k)>eps?A.k<B.k:cross(A.u-B.u,B.v-B.u)<0;});

tot=1;

**for** (**int** i = 1; i < n; ++i)

**if**(fabs(line[i].k-line[i-1].k)>eps)

line[tot++]=line[i];

**int** head=0,tail=1;

Q[0]=line[0];

Q[1]=line[1];

resn=0;

**for** (**int** i = 2; i < tot; ++i)

{

**if**(fabs(cross(Q[tail].v-Q[tail].u,Q[tail-1].v-Q[tail-1].u))<eps||fabs(cross(Q[head].v-Q[head].u,Q[head+1].v-Q[head+1].u))<eps)

**return**;

**while**(head<tail&&(cross((Q[tail]&Q[tail-1])-line[i].u,line[i].v-line[i].u))>eps) tail--;

**while**(head<tail&&(cross((Q[head]&Q[head+1])-line[i].u,line[i].v-line[i].u))>eps) head++;

Q[++tail]=line[i];

}

**while**(head<tail&&(cross(((Q[tail]&Q[tail-1])-Q[head].u),Q[head].v-Q[head].u))>eps) tail--;

**while**(head<tail&&(cross(((Q[head]&Q[head-1])-Q[tail].u),Q[tail].v-Q[tail].v))>eps) head++;

**if**(tail<=head+1)

**return**;

**for** (**int** i = head; i < tail; ++i)

res[resn++]=Q[i]&Q[i+1];

**if**(head<tail-1)

res[resn++]=Q[head]&Q[tail];

}

**struct** Circle

{

point o;

**double** r;

Circle(){}

Circle(point o,**double** r):o(o),r(r){}

};

**int** relation(point w,Line l)

{

*//1:左侧 2:右侧 3:线上*

**int** c=dcmp(cross(w-l.u,l.v-l.u));

**return** c<0?1:(c==0?3:2);

}

**int** relation(point p,Circle a)

{

*//0:圆外,1:圆上,2:圆内*

**double** d=dis(p,a.o)-a.r;

**if**(dcmp(d)==0) **return** 1;

**return** (dcmp(d)<0?2:0);

}

**int** relation(Line a,Circle b)

{

*//0:相离,1:相切,2:相交*

**double** p=a.pdis(b.o);

**if** (dcmp (p-b.r) == 0) **return** 1;

**return** (dcmp (p-b.r) < 0 ? 2 : 0);

}

**int** line\_cirlce\_intersection(Line l,Circle c,point& p1,point& p2)

{

**if**(!relation(l,c))

**return** 0;

point a=l.project(c.o);

**double** d=l.pdis(c.o);

d=sqrt(c.r\*c.r-d\*d);

**if**(dcmp(d)==0)

{

p1=a,p2=a;

**return** 0;

}

p1=a+(l.v-l.u).change\_len(d);

p2=a-(l.v-l.u).change\_len(d);

**return** 2;

}

**double** circle\_traingle\_area(point a,point b,Circle c)

{

point p=c.o;**double** r=c.r;

**if**(dcmp(cross(p-a,p-b))==0)

**return** 0;

point q[6];

**int** len=0;

q[len++]=a;

Line l=Line(a,b);

**if** (line\_cirlce\_intersection (l, c, q[1], q[2]) == 2)

{

**if** (dcmp (dot (a-q[1], b-q[1])) < 0) q[len++] = q[1];

**if** (dcmp (dot (a-q[2], b-q[2])) < 0) q[len++] = q[2];

}

q[len++]=b;

**if**(len==4&&dcmp(dot (q[0]-q[1], q[2]-q[1])) > 0)

swap(q[1],q[2]);

**double** ans=0;

**for** (**int** i = 0; i < len - 1; ++i)

{

**if**(relation(q[i],c)==0||relation(q[i+1],c)==0)

{

**double** arg=rad(q[i]-p,q[i+1]-p);

ans+=r\*r\*arg/2.0;

}

**else**

ans+=fabs(cross (q[i]-p, q[i+1]-p))/2;

}

**return** ans;

}

**double** area\_polygon\_circle(Circle c,point\* p,**int** n)

{

**double** ans=0;

p[n]=p[0];

**for** (**int** i = 0; i < n; ++i)

{

**if**(dcmp(cross(p[i+1]-c.o,p[i]-c.o))>=0)

ans+=circle\_traingle\_area(p[i],p[i+1],c);

**else**

ans-=circle\_traingle\_area(p[i],p[i+1],c);

}

**return** fabs(ans);

}

point aa[105][2005];

point pa[200006];

**double** smx[105],smy[105],smqx[105],smqy[105];

Line hp[1000];

**int** main()

{

**int** T,n,cas=1,m;

caze(T)

{

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

**for** (**int** i = 0; i < n; ++i)

{

smx[i]=smy[i]=smqx[i]=smqy[i]=0;

**for** (**int** j = 0; j < m; ++j)

{

aa[i][j].input();

smx[i]+=aa[i][j].x;

smy[i]+=aa[i][j].y;

smqx[i]+=sqr(aa[i][j].x);

smqy[i]+=sqr(aa[i][j].y);

}

}

**int** tot,cnt;

printf("Case #%d:",cas++);

**for** (**int** i = 0; i < n; ++i)

{

cnt=0;

hp[cnt++]=Line(point(0,0),point(4095,0));

hp[cnt++]=Line(point(4095,0),point(4095,4095));

hp[cnt++]=Line(point(4095,4095),point(0,4095));

hp[cnt++]=Line(point(0,4095),point(0,0));

**double** A=0,B=0,C=0;

**bool** f=0;

**for** (**int** j = 0; j < n; ++j)

{

**if**(i==j) **continue**;

A=-2.0\*(smx[i]-smx[j]);

B=-2.0\*(smy[i]-smy[j]);

C=smqx[i]+smqy[i]-smqx[j]-smqy[j];

point uu,vv;

**if**(dcmp(B)!=0)

{

uu=point(0,C/-B);

**if**(dcmp(A)!=0) vv=point(C/-A,0);

**else** vv=point(1,C/-B);

}

**else**

{

**if**(dcmp(A)!=0) uu=point(C/-A,1),vv=point(C/-A,0);

**else** **if**(C>=0)

{

f=1;

**break**;

}

**else**{cout<<1/0<<Endl;}

}

**int** tp=dcmp((point(0,0)-uu)^(vv-uu));

**bool** can=1;

**if**(tp>0&&C<=0) swap(uu,vv);

**if**(tp<0&&C>=0) swap(uu,vv);

**if**(tp==0)

{

tp=dcmp((point(0,4095)-uu)^(vv-uu));

**if**(tp<0&&B\*4095+C>=0) swap(uu,vv);

**if**(tp>0&&B\*4095+C<=0) swap(uu,vv);

**if**(tp==0)

{

tp=dcmp((point(4095,0)-uu)^(vv-uu));

**if**(tp<0&&A\*4095+C>=0) swap(uu,vv);

**if**(tp>0&&A\*4095+C<=0) swap(uu,vv);

**if**(tp==0)

can=0;

}

}

**if**(can)

hp[cnt++]=Line(uu,vv);

}

**if**(!f)

Hpi(cnt,hp,pa,tot);

printf(" %d",f?0:(**int**)(Area(tot,pa)+0.5));

}

putchar('\n');

}

}

裸凸包

#include<cstdio>

#include<cmath>

#include<cstring>

#include<algorithm>

**using** **namespace** std;

#define clr(a,b) memset(a,b,sizeof(a))

#define ll long long

#define ull unsigned long long

#define lowbit(x) (x&-x)

#define pb(x) push\_back(x)

#define IOS ios::sync\_with\_stdio(0),cin.tie(0),cout.tie(0)

#define inf (1<<30)

#define caze(T) for(cin>>T;T;T--)

#define Endl ('\n')

**struct** point

{

**double** x,y;

point(){}

point(**double** x,**double** y):x(x),y(y){}

point **operator**-(**const** point w)**const** {**return** point(x-w.x,y-w.y);}

**bool** **operator**<(**const** point& w)**const** {**return** x==w.x?y<w.y:x<w.x;}

}a[2000007],p[2000007];

**inline** **double** cross(**const** point& A,**const** point& B){**return** A.x\*B.y-B.x\*A.y;}

**inline** **double** dot(**const** point& q,**const** point& w){**return** q.x\*w.x+q.y\*w.y;}

**inline** **double** Xmul(**const** point& A,**const** point& B,**const** point& C){**return** (B.x-A.x)\*(C.y-A.y)-(B.y-A.y)\*(C.x-A.x);}

**inline** **double** dis(**const** point& q,**const** point& w){**return** sqrt(dot(q-w,q-w));}

**int** n,tot;

**void** Andrew()

{

sort(a,a+n);

tot=0;

**for** (**int** i = 0; i < n; ++i)

{

**while**(tot>1&&cross(p[tot-1]-p[tot-2],a[i]-p[tot-2])<=0) tot--;

p[tot++]=a[i];

}

**int** tp=tot;

**for** (**int** i = n - 2; ~i; --i)

{

**while**(tot>tp&&cross(p[tot-1]-p[tot-2],a[i]-p[tot-2])<=0) tot--;

p[tot++]=a[i];

}

tot-=(n>1);

}

**int** main()

{

**double** R;

**while**(~scanf("%d%lf",&n,&R))

{

**for** (**int** i = 0; i < n; ++i)

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

Andrew();

**double** C=dis(p[0],p[tot-1]);

**for** (**int** i = 0; i < tot - 1; ++i)

C+=dis(p[i],p[i+1]);

printf("%d**\n**",(**int**)(C+2\*acos(-1.0)\*R+0.5));

}

}

半平面交裸:求两多边形面积并减面积交

#include<bits/stdc++.h>

#include<ext/rope>

**using** **namespace** std;

#define clr(a,b) memset(a,b,sizeof(a))

#define ll long long

#define ull unsigned long long

#define lowbit(x) (x&-x)

#define pb(x) push\_back(x)

#define IOS ios::sync\_with\_stdio(0),cin.tie(0),cout.tie(0)

#define inf (1<<30)

#define caze(T) for(scanf("%d",&T);T;T--)

#define Endl ('\n')

**const** **double** pi=acos(-1.0);

**const** **double** eps=1e-8;

**int** dcmp(**double** x){**return** fabs(x)<=1e-8?0:(x<0?-1:1);}

**struct** point

{

**double** x,y,id;

point(){}

point(**double** x,**double** y,**int** id=-1):x(x),y(y),id(id) {}

point **operator**-(**const** point w)**const** {**return** point(x-w.x,y-w.y);}

point **operator**+(**const** point w)**const** {**return** point(x+w.x,y+w.y);}

point **operator**\*(**double** a) {**return** point(x\*a,y\*a);}

point **operator**/(**double** a) {**return** point(x/a,y/a);}

**void** input(){scanf("%lf%lf",&x,&y);}

};

point aa[200007],ab[200007];

point pa[200007],pb[200007];

**inline** **double** cross(**const** point& A,**const** point& B){**return** A.x\*B.y-B.x\*A.y;}

**inline** **double** dot(**const** point& q,**const** point& w){**return** q.x\*w.x+q.y\*w.y;}

**inline** **double** Xmul(**const** point& A,**const** point& B,**const** point& C){**return** cross(C-A,B-A);}

**inline** **double** dis(**const** point& q,**const** point& w){**return** sqrt(dot(q-w,q-w));}

**inline** **double** rad(**const** point& A,**const** point& B){**return** fabs(atan2(fabs(cross(A,B)),dot(A,B)));}

**int** Andrew(**int** n,point \*st,point \*ed)

{

sort(st,st+n,[](**const** point& A,**const** point& B)->**bool**{**return** A.x==B.x?A.y<B.y:A.x<B.x;});

**int** tot=0;

**for** (**int** i = 0; i < n; ++i)

{

**while**(tot>1&&cross(ed[tot-1]-ed[tot-2],st[i]-ed[tot-2])<0) tot--;

ed[tot++]=st[i];

}

**int** tp=tot;

**for** (**int** i = n - 2; ~i; --i)

{

**while**(tot>tp&&cross(ed[tot-1]-ed[tot-2],st[i]-ed[tot-2])<0) tot--;

ed[tot++]=st[i];

}

tot-=(n>1);

**return** tot;

}

**double** Area(**int** n,point \*p)

{

**double** S=0;

**for** (**int** i = 1; i < n - 1; ++i)

S+=fabs(Xmul(p[0],p[i],p[i+1]));

**return** S/2;

}

**struct** Line

{

point u,v;

**double** k;

Line(){}

Line(point u,point v):u(u),v(v){k=atan2(v.y-u.y,v.x-u.x);}

Line(point u,**double** k):u(u),k(k){v=u+(dcmp(k-pi/2)?point(1,tan(k)):point(0,1));}

**void** input(){u.input();v.input();}

**double** len(){**return** dis(u,v);}

point **operator**&(**const** Line& b)**const**

{

point ret=u;

**double** t=(cross(u-b.u,b.u-b.v))/cross(u-v,b.u-b.v);

ret.x+=(v.x-u.x)\*t;

ret.y+=(v.y-u.y)\*t;

**return** ret;

}

};

Line ln[100010];

Line hp[100010];

**void** Hpi(**int** n,Line \*line,point \*res,**int** &resn)

{

**int** tot=1;

sort(line,line+n,[](**const** Line& A,**const** Line& B)->**bool**{**return** fabs(A.k-B.k)>eps?A.k<B.k:cross(A.u-B.u,B.v-B.u)<0;});

**for** (**int** i = 1; i < n; ++i)

**if**(fabs(line[i].k-line[i-1].k)>eps)

line[tot++]=line[i];

**int** head=0,tail=1;

ln[0]=line[0];

ln[1]=line[1];

resn=0;

**for** (**int** i = 2; i < tot; ++i)

{

**if**(fabs(cross(ln[tail].v-ln[tail].u,ln[tail-1].v-ln[tail-1].u))<eps||fabs(cross(ln[head].v-ln[head].u,ln[head+1].v-ln[head+1].u))<eps)

**return**;

**while**(head<tail&&(cross((ln[tail]&ln[tail-1])-line[i].u,line[i].v-line[i].u))>eps) tail--;

**while**(head<tail&&(cross((ln[head]&ln[head+1])-line[i].u,line[i].v-line[i].u))>eps) head++;

ln[++tail]=line[i];

}

**while**(head<tail&&(cross(((ln[tail]&ln[tail-1])-ln[head].u),ln[head].v-ln[head].u))>eps) tail--;

**while**(head<tail&&(cross(((ln[head]&ln[head-1])-ln[tail].u),ln[tail].v-ln[tail].v))>eps) head++;

**if**(tail<=head+1)

**return**;

**for** (**int** i = head; i < tail; ++i)

res[resn++]=ln[i]&ln[i+1];

**if**(head<tail-1)

res[resn++]=ln[head]&ln[tail];

}

**int** main()

{

**int** T;

**int** n,m;

**while**(scanf("%d",&n)&&n)

{

**for** (**int** i = 0; i < n; ++i)

aa[i].input();

**double** Sa,Sb,Sc;

**int** tota=Andrew(n,aa,pa);

Sa=Area(tota,pa);

scanf("%d",&n);

**for** (**int** i = 0; i < n; ++i)

ab[i].input();

**int** totb=Andrew(n,ab,pb);

Sb=Area(totb,pb);

**int** cnt=0;

**for** (**int** i = 0; i < tota; ++i)

hp[cnt++]=Line(pa[i],pa[(i+1)%tota]);

**for** (**int** i = 0; i < totb; ++i)

hp[cnt++]=Line(pb[i],pb[(i+1)%totb]);

**int** totc=0;

Hpi(cnt,hp,aa,totc);

Sc=Area(totc,aa);

printf("%8.2f",Sa+Sb-Sc-Sc);

}

puts("");

}

旋转卡壳求最远点对

**const** **double** pi=acos(-1.0);

**const** **double** eps=1e-8;

**int** dcmp(**double** x){**return** fabs(x)<=eps?0:(x<0?-1:1);}

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

**struct** point

{

**double** x,y,id;

point(){}

point(**double** x,**double** y,**int** id=-1):x(x),y(y),id(id) {}

point **operator**-(**const** point w)**const** {**return** point(x-w.x,y-w.y);}

point **operator**+(**const** point w)**const** {**return** point(x+w.x,y+w.y);}

**double** **operator**\*(**const** point& w)**const** {**return** x\*w.x+y\*w.y;}

point **operator**\*(**double** a) {**return** point(x\*a,y\*a);}

**double** **operator**^(**const** point& w)**const** {**return** x\*w.y-y\*w.x;}

point **operator**/(**double** a) {**return** point(x/a,y/a);}

*//friend ostream &operator<<(ostream& out,const point& w) {out<<'('<<w.x<<','<<w.y<<')';return out;}*

**void** input(){scanf("%lf%lf",&x,&y);}

**double** len2(){**return** x\*x+y\*y;}

**double** len(){**return** sqrt(x\*x+y\*y);}

**bool** **operator**<(**const** point& w)**const**{**return** x==w.x?y<w.y:x<w.x;}

point change\_len(**double** r)

{

**double** l=len();

**if**(dcmp(l)==0) **return** \***this**;

r/=l;

**return** point(x\*r,y\*r);

}

};

**inline** **double** cross(**const** point& A,**const** point& B){**return** A.x\*B.y-B.x\*A.y;}

**inline** **double** dot(**const** point& q,**const** point& w){**return** q.x\*w.x+q.y\*w.y;}

**inline** **double** Xmul(**const** point& A,**const** point& B,**const** point& C){**return** cross(C-A,B-A);}

**inline** **double** dis(**const** point& q,**const** point& w){**return** sqrt(dot(q-w,q-w));}

**inline** **double** rad(**const** point& A,**const** point& B){**return** fabs(atan2(fabs(cross(A,B)),dot(A,B)));}

**int** Andrew(**int** n,point \*st,point \*ed)

{

sort(st,st+n);

**int** tot=0;

**for** (**int** i = 0; i < n; ++i)

{

**while**(tot>1&&cross(ed[tot-1]-ed[tot-2],st[i]-ed[tot-2])<=0) tot--;

ed[tot++]=st[i];

}

**int** tp=tot;

**for** (**int** i = n - 2; ~i; --i)

{

**while**(tot>tp&&cross(ed[tot-1]-ed[tot-2],st[i]-ed[tot-2])<=0) tot--;

ed[tot++]=st[i];

}

tot-=(n>1);

**return** tot;

}

**double** Area(**int** n,point \*p)

{

**double** S=0;

**for** (**int** i = 1; i < n - 1; ++i)

S+=fabs(Xmul(p[0],p[i],p[i+1]));

**return** S/2;

}

**int** cal(**int** n,point \*p)

{

**int** ret=0;

**int** tp=1;

**for** (**int** i = 0; i < n; ++i)

{

**while**(((p[(i+1)%n]-p[i])^(p[tp]-p[i]))<((p[(i+1)%n]-p[i])^(p[(tp+1)%n]-p[i])))

(tp+=1)%=n;

ret=max(ret,(**int**)((p[tp]-p[i]).len2()+eps));

}

**return** ret;

}

point aa[50005];

point pa[50005];

**int** main()

{

**int** n;

scanf("%d",&n);

**for** (**int** i = 0; i < n; ++i)

aa[i].input();

**int** tot=Andrew(n,aa,pa);

printf("%d**\n**",cal(tot,pa));

}

给定n个点求最大三角形面积,旋转卡壳:

#include<iostream>

#include<cstdio>

#include<cstring>

#include<cstdlib>

#include<set>

#include<ctime>

#include<vector>

#include<queue>

#include<algorithm>

#include<map>

#include<cmath>

**using** **namespace** std;

#define clr(a,b) memset(a,b,sizeof(a))

#define ll long long

#define ull unsigned long long

#define lowbit(x) (x&-x)

#define pb(x) push\_back(x)

#define IOS ios::sync\_with\_stdio(0),cin.tie(0),cout.tie(0)

#define inf (1<<30)

#define caze(T) for(scanf("%d",&T);T;T--)

#define Endl ('\n')

**const** **double** pi=acos(-1.0);

**const** **double** eps=1e-8;

**int** dcmp(**double** x){**return** fabs(x)<=eps?0:(x<0?-1:1);}

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

**struct** point

{

**double** x,y,id;

point(){}

point(**double** x,**double** y,**int** id=-1):x(x),y(y),id(id) {}

point **operator**-(**const** point w)**const** {**return** point(x-w.x,y-w.y);}

point **operator**+(**const** point w)**const** {**return** point(x+w.x,y+w.y);}

**double** **operator**\*(**const** point& w)**const** {**return** x\*w.x+y\*w.y;}

point **operator**\*(**double** a) {**return** point(x\*a,y\*a);}

**double** **operator**^(**const** point& w)**const** {**return** x\*w.y-y\*w.x;}

point **operator**/(**double** a) {**return** point(x/a,y/a);}

**friend** ostream &**operator**<<(ostream& out,**const** point& w) {out<<'('<<w.x<<','<<w.y<<')';**return** out;}

**void** input(){scanf("%lf%lf",&x,&y);}

**double** len2(){**return** x\*x+y\*y;}

**double** len(){**return** sqrt(x\*x+y\*y);}

point change\_len(**double** r)

{

**double** l=len();

**if**(dcmp(l)==0) **return** \***this**;

r/=l;

**return** point(x\*r,y\*r);

}

**bool** **operator**<(**const** point& w)**const** {**return** x==w.x?y<w.y:x<w.x;}

};

**inline** **double** cross(**const** point& A,**const** point& B){**return** A.x\*B.y-B.x\*A.y;}

**inline** **double** dot(**const** point& q,**const** point& w){**return** q.x\*w.x+q.y\*w.y;}

**inline** **double** Xmul(**const** point& A,**const** point& B,**const** point& C){**return** cross(C-A,B-A);}

**inline** **double** dis(**const** point& q,**const** point& w){**return** sqrt(dot(q-w,q-w));}

**inline** **double** rad(**const** point& A,**const** point& B){**return** fabs(atan2(fabs(cross(A,B)),dot(A,B)));}

**int** Andrew(**int** n,point \*st,point \*ed)

{

sort(st,st+n);

*//sort(st,st+n,[](const point& A,const point& B)->bool{return A.x==B.x?A.y<B.y:A.x<B.x;});*

**int** tot=0;

**for** (**int** i = 0; i < n; ++i)

{

**while**(tot>1&&cross(ed[tot-1]-ed[tot-2],st[i]-ed[tot-2])<=0) tot--;

ed[tot++]=st[i];

}

**int** tp=tot;

**for** (**int** i = n - 2; ~i; --i)

{

**while**(tot>tp&&cross(ed[tot-1]-ed[tot-2],st[i]-ed[tot-2])<=0) tot--;

ed[tot++]=st[i];

}

tot-=(n>1);

**return** tot;

}

**double** Area(**int** n,point \*p)

{

**double** S=0;

**for** (**int** i = 1; i < n - 1; ++i)

S+=fabs(Xmul(p[0],p[i],p[i+1]));

**return** S/2;

}

**double** cal(**int** n,point \*p)

{

**double** ret=0;

**int** t1=1,t2=2;

**for** (**int** i = 0; i < n; ++i)

{

**while**(((p[t1]-p[i])^(p[t2]-p[i]))<((p[t1]-p[i])^(p[(t2+1)%n]-p[i])))

(t2+=1)%=n;

ret=max(ret,((p[t1]-p[i])^(p[t2]-p[i]))/2.0);

**while**(((p[t1]-p[i])^(p[t2]-p[i]))<((p[(t1+1)%n]-p[i])^(p[t2]-p[i])))

(t1+=1)%=n;

ret=max(ret,((p[t1]-p[i])^(p[t2]-p[i]))/2.0);

}

**return** ret;

}

point aa[50005];

point pa[50005];

**int** main()

{

**int** n;

**while**(scanf("%d",&n)&&(~n))

{

**for** (**int** i = 0; i < n; ++i)

aa[i].input();

**int** tot=Andrew(n,aa,pa);

printf("%.2f**\n**",cal(tot,pa));

}

}

求两凸包最短距离

**const** **double** pi=acos(-1.0);

**const** **double** eps=1e-8;

**int** dcmp(**double** x){**return** fabs(x)<=eps?0:(x<0?-1:1);}

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

**struct** point

{

**double** x,y,id;

point(){}

point(**double** x,**double** y,**int** id=-1):x(x),y(y),id(id) {}

point **operator**-(**const** point w)**const** {**return** point(x-w.x,y-w.y);}

point **operator**+(**const** point w)**const** {**return** point(x+w.x,y+w.y);}

**double** **operator**\*(**const** point& w)**const** {**return** x\*w.x+y\*w.y;}

point **operator**\*(**double** a) {**return** point(x\*a,y\*a);}

**double** **operator**^(**const** point& w)**const** {**return** x\*w.y-y\*w.x;}

point **operator**/(**double** a) {**return** point(x/a,y/a);}

**friend** ostream &**operator**<<(ostream& out,**const** point& w) {out<<'('<<w.x<<','<<w.y<<')';**return** out;}

**void** input(){scanf("%lf%lf",&x,&y);}

**double** len2(){**return** x\*x+y\*y;}

**double** len(){**return** sqrt(x\*x+y\*y);}

point change\_len(**double** r)

{

**double** l=len();

**if**(dcmp(l)==0) **return** \***this**;

r/=l;

**return** point(x\*r,y\*r);

}

**bool** **operator**<(**const** point& w)**const** {**return** x==w.x?y<w.y:x<w.x;}

};

**inline** **double** cross(**const** point& A,**const** point& B){**return** A.x\*B.y-B.x\*A.y;}

**inline** **double** dot(**const** point& q,**const** point& w){**return** q.x\*w.x+q.y\*w.y;}

**inline** **double** Xmul(**const** point& A,**const** point& B,**const** point& C){**return** cross(C-A,B-A);}

**inline** **double** dis(**const** point& q,**const** point& w){**return** sqrt(dot(q-w,q-w));}

**inline** **double** rad(**const** point& A,**const** point& B){**return** fabs(atan2(fabs(cross(A,B)),dot(A,B)));}

**int** Andrew(**int** n,point \*st,point \*ed)

{

sort(st,st+n);

*//sort(st,st+n,[](const point& A,const point& B)->bool{return A.x==B.x?A.y<B.y:A.x<B.x;});*

**int** tot=0;

**for** (**int** i = 0; i < n; ++i)

{

**while**(tot>1&&cross(ed[tot-1]-ed[tot-2],st[i]-ed[tot-2])<=0) tot--;

ed[tot++]=st[i];

}

**int** tp=tot;

**for** (**int** i = n - 2; ~i; --i)

{

**while**(tot>tp&&cross(ed[tot-1]-ed[tot-2],st[i]-ed[tot-2])<=0) tot--;

ed[tot++]=st[i];

}

tot-=(n>1);

**return** tot;

}

**double** Area(**int** n,point \*p)

{

**double** S=0;

**for** (**int** i = 1; i < n - 1; ++i)

S+=fabs(Xmul(p[0],p[i],p[i+1]));

**return** S/2;

}

point aa[10007],bb[10007];

point pa[10007],pb[10007];

**double** dist(point a,point b,point c)

{

**double** tp=fabs((b-a)^(c-a));

point t1=b-a,t2=c-a,t3=c-b;

**if**(dcmp(dot(t1,t2))<0) **return** t2.len();

**if**(dcmp(dot(t1,t3))>0) **return** t3.len();

**return** tp/dis(a,b);

}

**double** cal(point a,point b,point c,point d)

{

**double** t[4];

**int** cnt=0;

t[cnt++]=dist(a,b,c);

t[cnt++]=dist(a,b,d);

t[cnt++]=dist(c,d,a);

t[cnt++]=dist(c,d,b);

sort(t,t+cnt);

**return** t[0];

}

**double** rot(point \*p,point \*q,**int** n,**int** m)

{

**int** mq=0,mp=0;

p[n]=p[0],q[m]=q[0];

**for** (**int** i = 1; i < n; ++i) mp=p[i].y<p[mp].y?i:mp;

**for** (**int** i = 1; i < m; ++i) mq=q[i].y>q[mq].y?i:mq;

**double** ans=dis(p[mp],q[mq]),t;

**for** (**int** i = 0; i < n; ++i)

{

**while**(dcmp(t=((q[mq+1]-p[mp+1])^(p[mp]-p[mp+1]))-((q[mq]-p[mp+1])^(p[mp]-p[mp+1])))==1)

(mq+=1)%=m;

**if**(dcmp(t)<0)

ans=min(ans,dist(p[mp],p[mp+1],q[mq]));

**else**

ans=min(ans,cal(q[mq],q[mq+1],p[mp],p[mp+1]));

(mp+=1)%=n;

}

**return** ans;

}

**int** main()

{

**int** n,m;

**while**(scanf("%d%d",&n,&m)&&n)

{

**for** (**int** i = 0; i < n; ++i)

aa[i].input();

**for** (**int** i = 0; i < m; ++i)

bb[i].input();

**int** ta=Andrew(n,aa,pa);

**int** tb=Andrew(m,bb,pb);

printf("%.5f**\n**",min(rot(pa,pb,ta,tb),rot(pb,pa,tb,ta)));

}

}

Tarjan

**const** **int** maxn=50020;

**struct** EDGE{**int** v,w,nxt;}edge[1000010];

**int** tot;

**int** head[maxn];

**void** AE(**int** u,**int** v,**int** w){edge[tot]={v,w,head[u]},head[u]=tot++;}

**int** n,m;

**int** ttime,idx,col;

**int** dfn[maxn];

**int** low[maxn];

**int** stk[maxn];

**bool** vis[maxn];

**int** belong[maxn];

**void** init()

{

tot=ttime=idx=col=0;

clr(dfn,0);

clr(head,-1);

clr(vis,0);

clr(belong,0);

}

**void** tarjan(**int** u)

{

dfn[u]=low[u]=++ttime;

vis[u]=1;

stk[++idx]=u;

**int** tp=0;

**for**(**int** i=head[u];~i;i=edge[i].nxt)

{

**int** v=edge[i].v;

**if**(!dfn[v])

{

tarjan(v);

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

}

**else** **if**(vis[v])

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

}

**if**(dfn[u]==low[u])

{

col++;

**do**

{

vis[stk[idx]]=0;

belong[stk[idx--]]=col;

} **while** (vis[u]);

}

}

MCMF

dij:

**const** **int** MAXN=222;

**struct** EDGE{**int** to,cap,cost,flow,nxt;}edge[1<<22];

**int** head[MAXN];

**int** tot;

**void** AE(**int** from,**int** to,**int** cap,**int** cost)

{

edge[tot]={to,cap,cost,0,head[from]},head[from]=tot++;

edge[tot]={from,0,-cost,0,head[to]},head[to]=tot++;

}

**int** cost,flow;

**int** h[MAXN];

**int** dist[MAXN],pre[MAXN];

**void** min\_cost\_flow(**int** s,**int** t,**int** f,**int** N)

{

fill(h,h+1+N,0);

**while**(f>0)

{

priority\_queue<pii,vector<pii>,greater<pii> >q;

clr(dist,inf);

dist[s]=0,q.push(pii(0,s));

clr(pre,-1);

**while**(!q.empty())

{

pii now=q.top();

q.pop();

**if**(dist[now.second]<now.first) **continue**;

**int** u=now.second;

**for** (**int** i = head[u]; ~i; i=edge[i].nxt)

{

EDGE &e=edge[i];

**if** (e.cap>e.flow&&dist[e.to]>dist[u]+e.cost+h[u]-h[e.to])

{

dist[e.to]=dist[u]+e.cost+h[u]-h[e.to];

pre[e.to]=i;

q.push(pii(dist[e.to],e.to));

}

}

}

**if**(dist[t]==inf) **break**;

**for** (**int** i = 0; i <= N; ++i)

h[i]+=dist[i];

**int** d=f;

**for** (**int** i = pre[t]; ~i; i=pre[edge[i^1].to])

d=min(d,edge[i].cap-edge[i].flow);

f-=d;flow+=d;

cost+=d\*h[t];

**for** (**int** i = pre[t]; ~i; i=pre[edge[i^1].to])

{

edge[i].flow+=d;

edge[i^1].flow-=d;

}

}

}

**char** mp[111][111];

**int** xx[2][111],yy[2][111];

**int** w[111][111];

**int** nx,ny;

**int** main()

{

**int** nn,mm;

**while**(scanf("%d%d",&nn,&mm)&&nn)

{

nx=0,ny=0;

**for** (**int** i = 0; i < nn; ++i)

{

getchar();

**for** (**int** j = 0; j < mm; ++j)

{

mp[i][j]=getchar();

**if**(mp[i][j]=='H') xx[0][nx]=i,yy[0][nx++]=j;

**if**(mp[i][j]=='m') xx[1][ny]=i,yy[1][ny++]=j;

}

}

**for** (**int** i = 0; i < nx; ++i)

**for** (**int** j = 0; j < ny; ++j)

w[i][j]=abs(xx[0][i]-xx[1][j])+abs(yy[0][i]-yy[1][j]);

**int** s=1,t=nx+ny+2,n=t;

tot=0;

clr(head,-1);

flow=0,cost=0;

**for** (**int** i = 0; i < nx; ++i)

AE(s,i+2,1,0);

**for** (**int** i = 0; i < nx; ++i)

**for** (**int** j = 0; j < ny; ++j)

AE(i+2,j+nx+2,1,w[i][j]);

**for** (**int** i = 0; i < ny; ++i)

AE(i+nx+2,t,1,0);

min\_cost\_flow(s,t,inf,n);

printf("%d**\n**",cost);

}

}

SPFA(网上板子):

**const** **int** N=1005;

**const** **int** M=50000;

**const** **int** inf=0x3f3f3f3f;

queue<**int**> que;

**int** n,m,ans=0;

**int** first[50000],next[50000],go[50000],rest[50000],cost[50000],dis[1005],tot=1;

**bool** visit[50000],work[50000];

**int** src,des;

**void** combin(**int** u,**int** v,**int** r,**int** w)

{

next[++tot]=first[u],first[u]=tot,go[tot]=v,rest[tot]=r,cost[tot]=w;

next[++tot]=first[v],first[v]=tot,go[tot]=u,rest[tot]=0,cost[tot]=-w;

}

**void** init(**int** n,**int** m)

{

src=0,des=n+1;

**for**(**int** i=1;i<=m;i++)

{

**int** u,v,w;

scanf("%d%d%d",&u,&v,&w);

combin(u,v,1,w);

combin(v,u,1,w);

}

combin(src,1,2,0);

combin(n,des,2,0);

}

**bool** spfa()

{

memset(dis,inf,**sizeof**(dis));

memset(work,**false**,**sizeof**(work));

**int** u;

que.push(src),dis[src]=0;

**while**(!que.empty())

{

u=que.front(),que.pop();

visit[u]=**false**;

**for**(**int** e=first[u];e;e=next[e])

{

**int** v=go[e];

**if**(rest[e]&&dis[u]+cost[e]<dis[v])

{

dis[v]=dis[u]+cost[e];

**if**(!visit[v])

{

que.push(v);

visit[v]=**true**;

}

}

}

}

**return** dis[des]<inf;

}

**int** dinic(**int** u,**int** flow)

{

**if**(u==des)

{

ans+=flow\*dis[des];

**return** flow;

}

work[u]=**true**;

**int** res=0,temp,v,e;

**for**(e=first[u];e;e=next[e])

{

**if**(!work[v=go[e]]&&rest[e]&&dis[v]==dis[u]+cost[e])

{

temp=dinic(v,min(rest[e],flow-res));

**if**(temp)

{

rest[e]-=temp,rest[e^1]+=temp;

res+=temp;

**if**(res==flow) **break**;

}

}

}

**return** res;

}

**int** maxflow()

{

**while**(spfa()) dinic(src,inf);

**return** ans;

}

**int** main()

{

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

init(n,m);

cout<<maxflow()<<endl;

**return** 0;

}