例题 csu 1812

**Description**

Bobo 有一个三角形和一个矩形，他想求他们交的面积。

具体地，三角形和矩形由 8 个整数 x1,y1,x2,y2,x3,y3,x4,y4 描述。 表示三角形的顶点坐标是 (x1,y1),(x1,y2),(x2,y1)， 矩形的顶点坐标是 (x3,y3),(x3,y4),(x4,y4),(x4,y3).

**Input**

输入包含不超过 30000 组数据。

每组数据的第一行包含 4 个整数 x1,y1,x2,y2 (x1≠x2,y1≠y2).

第二行包含 4 个整数 x3,y3,x4,y4 (x3<x4,y3<y4).

(0≤xi,yi≤104)

**Output**

对于每组数据，输出一个实数表示交的面积。绝对误差或相对误差小于 10-6 即认为正确。

**Sample Input**

1 1 3 3

0 0 2 2

0 3 3 1

0 0 2 2

4462 1420 2060 2969

4159 257 8787 2970

**Sample Output**

1.00000000

0.75000000

439744.13967527

#include<cstdio>

#include<cmath>

#include<cstring>

#include<iostream>

#include<algorithm>

#include<cstdlib>

#include<queue>

#include<map>

#include<stack>

#include<set>

using namespace std;

const int maxn=555;

const int maxisn=10;

const double eps=1e-8;

const double pi=acos(-1.0);

int dcmp(double x){

if(x>eps) return 1;

return x<-eps ? -1 : 0;

}

inline double Sqr(double x){

return x\*x;

}

struct Point{

double x,y;

Point(){x=y=0;}

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

friend Point operator + (const Point &a,const Point &b) {

return Point(a.x+b.x,a.y+b.y);

}

friend Point operator - (const Point &a,const Point &b) {

return Point(a.x-b.x,a.y-b.y);

}

friend bool operator == (const Point &a,const Point &b) {

return dcmp(a.x-b.x)==0&&dcmp(a.y-b.y)==0;

}

friend Point operator \* (const Point &a,const double &b) {

return Point(a.x\*b,a.y\*b);

}

friend Point operator \* (const double &a,const Point &b) {

return Point(a\*b.x,a\*b.y);

}

friend Point operator / (const Point &a,const double &b) {

return Point(a.x/b,a.y/b);

}

friend bool operator < (const Point &a, const Point &b) {

return a.x < b.x || (a.x == b.x && a.y < b.y);

}

inline double dot(const Point &b)const{

return x\*b.x+y\*b.y;

}

inline double cross(const Point &b,const Point &c)const{

return (b.x-x)\*(c.y-y)-(c.x-x)\*(b.y-y);

}

};

Point LineCross(const Point &a,const Point &b,const Point &c,const Point &d){

double u=a.cross(b,c),v=b.cross(a,d);

return Point((c.x\*v+d.x\*u)/(u+v),(c.y\*v+d.y\*u)/(u+v));

}

double PolygonArea(Point p[],int n){

if(n<3) return 0.0;

double s=p[0].y\*(p[n-1].x-p[1].x);

p[n]=p[0];

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

s+=p[i].y\*(p[i-1].x-p[i+1].x);

}

return fabs(s\*0.5);

}

double CPIA(Point a[],Point b[],int na,int nb){

Point p[maxisn],temp[maxisn];

int i,j,tn,sflag,eflag;

a[na]=a[0],b[nb]=b[0];

memcpy(p,b,sizeof(Point)\*(nb+1));

for(i=0;i<na&&nb>2;++i){

sflag=dcmp(a[i].cross(a[i+1],p[0]));

for(j=tn=0;j<nb;++j,sflag=eflag){

if(sflag>=0) temp[tn++]=p[j];

eflag=dcmp(a[i].cross(a[i+1],p[j+1]));

if((sflag^eflag)==-2)

temp[tn++]=LineCross(a[i],a[i+1],p[j],p[j+1]);

}

memcpy(p,temp,sizeof(Point)\*tn);

nb=tn,p[nb]=p[0];

}

if(nb<3) return 0.0;

return PolygonArea(p,nb);

}

double SPIA(Point a[],Point b[],int na,int nb){

int i,j;

Point t1[4],t2[4];

double res=0.0,if\_clock\_t1,if\_clock\_t2;

a[na]=t1[0]=a[0];

b[nb]=t2[0]=b[0];

for(i=2;i<na;i++){

t1[1]=a[i-1],t1[2]=a[i];

if\_clock\_t1=dcmp(t1[0].cross(t1[1],t1[2]));

if(if\_clock\_t1<0) swap(t1[1],t1[2]);

for(j=2;j<nb;j++){

t2[1]=b[j-1],t2[2]=b[j];

if\_clock\_t2=dcmp(t2[0].cross(t2[1],t2[2]));

if(if\_clock\_t2<0) swap(t2[1],t2[2]);

res+=CPIA(t1,t2,3,3)\*if\_clock\_t1\*if\_clock\_t2;

}

}

return res;//Ãæ»ý½»

//return PolygonArea(a,na)+PolygonArea(b,nb)-res;//Ãæ»ý²¢

}

Point a[222],b[222];

Point aa[222],bb[222];

int main(){

double x1,y1,x2,y2;

double x3,y3,x4,y4;

while(scanf("%lf %lf %lf %lf",&x1,&y1,&x2,&y2)!=EOF){

scanf("%lf %lf %lf %lf",&x3,&y3,&x4,&y4);

a[0]=Point(x1,y1); // ÄæÊ±Õë£¿£¿

a[1]=Point(x2,y1);

a[2]=Point(x1,y2);

b[0]=Point(x3,y3);

b[1]=Point(x4,y3);

b[2]=Point(x4,y4);

b[3]=Point(x3,y4);

printf("%.8f\n",fabs(SPIA(a,b,3,4)));

//printf("%.8f\n",ConvexPolygonArea(out,m));

}

return 0;

}