凸包

int n;

struct ben

{

double x,y;

}p[10005],s[10005];

double check(ben a1,ben a2,ben b1,ben b2)//检查叉积是否大于0，如果是a就逆时针转到b

{

return (a2.x-a1.x)\*(b2.y-b1.y)-(b2.x-b1.x)\*(a2.y-a1.y);

}

double d(ben p1,ben p2)//两点间距离。。。

{

return sqrt((p2.y-p1.y)\*(p2.y-p1.y)+(p2.x-p1.x)\*(p2.x-p1.x));

}

bool cmp(ben p1,ben p2)//排序函数，这个函数别写错了，要不然功亏一篑

{

double tmp=check(p[1],p1,p[1],p2);

if(tmp>0)

return 1;

if(tmp==0&&d(p[0],p1)<d(p[0],p2))

return 1;

return 0;

}

int main()

{

scanf("%d",&n);

double mid;

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

{

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

if(i!=1&&p[i].y<p[1].y)//这是是去重

{

mid=p[1].y;p[1].y=p[i].y;p[i].y=mid;

mid=p[1].x;p[1].x=p[i].x;p[i].x=mid;

}

}

sort(p+2,p+1+n,cmp);//系统快排

s[1]=p[1];//最低点一定在凸包里

int cnt=1;

for(int i=2;i<=n;i++)

{

while(cnt>1&&check(s[cnt-1],s[cnt],s[cnt],p[i])<=0) //判断前面的会不会被踢走，如果被踢走那么出栈

cnt--;

cnt++;

s[cnt]=p[i];

}

s[cnt+1]=p[1];//最后一个点回到凸包起点

double ans=0;

for(int i=1;i<=cnt;i++)

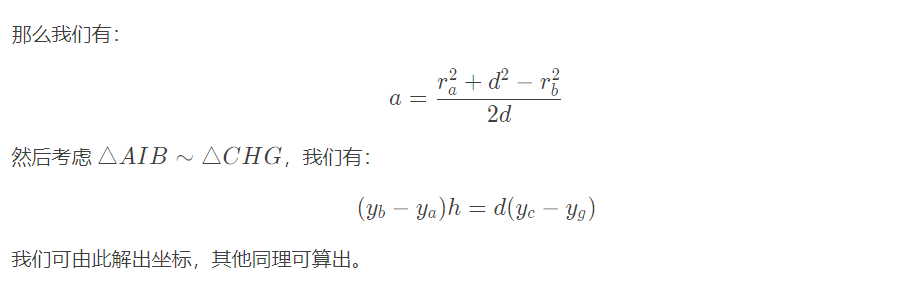
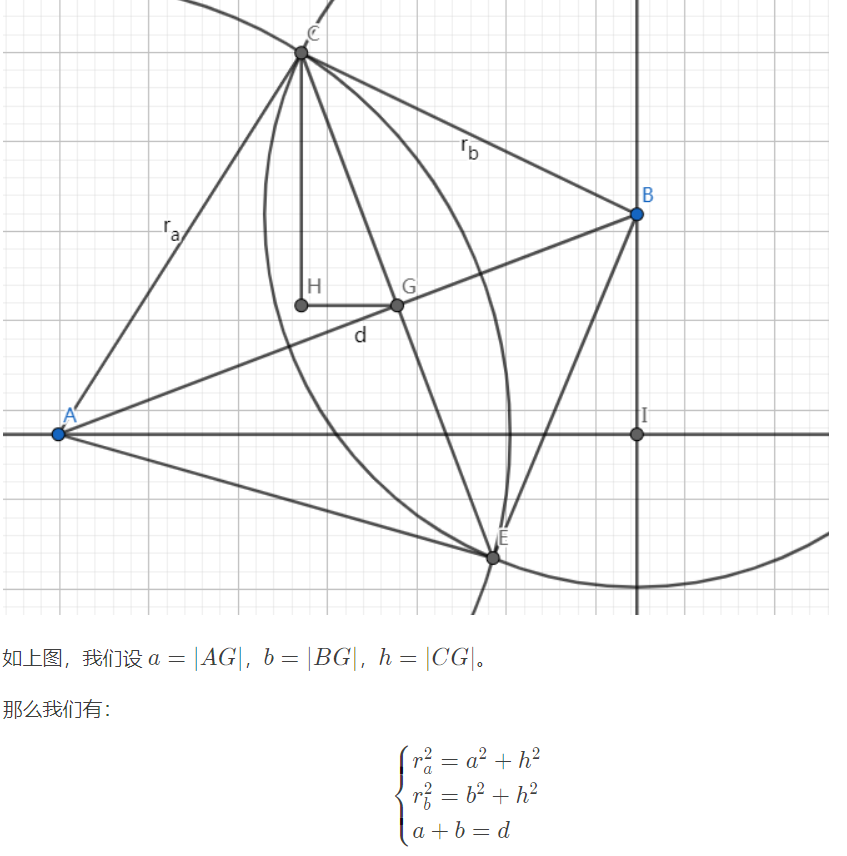
ans+=d(s[i],s[i+1]);//然后s里存好了凸包序列，只需要把两两距离累加就行

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

return 0;

}

相似三角形求两圆的交点



#include<bits/stdc++.h>

using namespace std;

#define reg register

typedef long long ll;

const double eps=1e-6;

inline int sgn(reg double x){

if(fabs(x)<eps)

return 0;

else

return x<0?-1:1;

}

inline double sqr(reg double x){

return x\*x;

}

const int MAXN=2e3+5;

struct Vector{

double x,y;

inline Vector(reg double x=0,reg double y=0):x(x),y(y){

return;

}

inline Vector operator+(const Vector& a)const{

return Vector(x+a.x,y+a.y);

}

inline Vector operator-(const Vector& a)const{

return Vector(x-a.x,y-a.y);

}

inline Vector operator\*(const double a)const{

return Vector(x\*a,y\*a);

}

};

inline double dot(const Vector& a,const Vector& b){

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

}

inline double cross(const Vector& a,const Vector& b){

return a.x\*b.y-a.y\*b.x;

}

typedef Vector Point;

inline double getDis2(const Point& a,const Point& b){

return dot(a-b,a-b);

}

inline double getDis(const Point& a,const Point& b){

return sqrt(getDis2(a,b));

}

inline bool isEmpty(const Point& a){

return isnan(a.x)||isnan(a.y);

}

struct Circle{

Point o;

double r;

inline bool contain(const Point& x)const{

return sgn(sqr(r)-getDis2(x,o))>=0;

}

inline Point getRig(void)const{

return o+Vector(r,0);

}

};

inline bool isCon(const Circle& a,const Circle& b){

return sgn(sqr(a.r-b.r)-getDis2(a.o,b.o))>=0;

}

inline bool isSep(const Circle& a,const Circle& b){

return sgn(getDis2(a.o,b.o)-sqr(a.r+b.r))>0;

}

inline Point getPot(const Circle &a,const Circle &b){

if(isCon(a,b))

if(sgn(b.getRig().x-a.getRig().x)>0)

return a.getRig();

else

return b.getRig();

else if(isSep(a,b))

return Point(nan(""),nan(""));

else{

if(a.contain(b.getRig()))

return b.getRig();

else if(b.contain(a.getRig()))

return a.getRig();

else{

reg double d=getDis(a.o,b.o);

reg double val=(sqr(a.r)+sqr(d)-sqr(b.r))/(2\*d);

reg double h=sqrt(sqr(a.r)-sqr(val));

Point bas=a.o+(b.o-a.o)\*(val/d);

Vector tmp=Vector(b.o.y-a.o.y,a.o.x-b.o.x)\*(h/d);

Point p1=bas-tmp,p2=bas+tmp;

if(sgn(p2.x-p1.x)>0)

return p2;

else

return p1;

}

}

}