分治法求最近点对

#include<cstdio>

#include<algorithm>

#include<cmath>

#include<iostream>

using *namespace* std;

*struct* Point{

*double* x, y;

*bool* operator < (const Point &*rhs*) const{

        return (x < *rhs*.x) || (x == *rhs*.x && y < *rhs*.y);

    }

}point[2000001],pp[2000001];

const *double* INF = 3e20 + 10;

*int* n;

*double* dist(*int* *x*, *int* *y*){

    return (point[*x*].x - point[*y*].x) \* (point[*x*].x - point[*y*].x) + (point[*x*].y - point[*y*].y) \* (point[*x*].y - point[*y*].y);

}

*double* min\_(*double* *a*, *double* *b*){

    if (*a* < *b*) return *a*;

    else return *b*;

}

*double* cdq(*int* *l*, *int* *r*){

    if (*r* - *l* + 1 == 1) return INF;

    else if (*r* - *l* + 1 == 2) return sqrt(dist(*l*, *r*));

*int* mid = (*l* + *r*) / 2;

*int* midx = point[mid].x;

*double* mind = min\_(cdq(*l*, mid), cdq(mid + 1, *r*));

    for (*int* i = *l*, j = mid + 1, k = *l*; i <= mid || j <= *r*; k++)

        if (i <= mid && (j > *r* || point[i].y < point[j].y)) pp[k] = point[i++];

        else pp[k] = point[j++];

*int* num = 0;

    for (*int* i = *l*; i <= *r*; i++){

        point[i] = pp[i];

        if (abs(midx - point[i].x) < mind){

            for (*int* j = num; j && point[i].y - pp[j].y < mind; j--)

                mind = min\_(mind, sqrt((point[i].x - pp[j].x) \* (point[i].x - pp[j].x) + (point[i].y - pp[j].y) \* (point[i].y - pp[j].y)));

            pp[++num] = point[i];

        }

    }

    return mind;

}

*int* main(){

    scanf("%d", &n);

    for (*int* i = 1; i <= n; i++) scanf("%lf%lf", &point[i].x, &point[i].y);

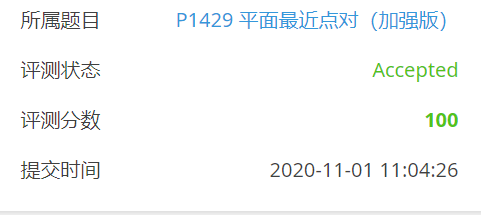
    sort(point + 1, point + n + 1);

    printf("%.4lf", cdq(1, n));

    return 0;

}

洛谷OJ



分治求凸包

#include<cstdio>

#include<iostream>

#include<algorithm>

using *namespace* std;

*bool* ans[10000];

*class* Point{

*public:*

*int* x, y;

        Point(*int* *a*, *int* *b*): x(*a*), y(*b*){}

*bool* operator < (const Point &*rhs*) const{

        return ((x < *rhs*.x) || ((x == *rhs*.x) && (y < *rhs*.y)));

    }

}point[10000] = Point(0, 0);

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

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

}

*int* det(Point *x*, Point *y*, Point *z*){

    Point xl1 = z - y;

    Point xl2 = x - y;

    return xl1.x \* xl2.y - xl2.x \* xl1.y;

}

*void* dealup(*int* *l*, *int* *r*){

    if (r - l + 1 <= 2){

        ans[l] = ans[r] = true;

        return;

    }

*int* max = 0, x = 0;

    for (*int* i = l + 1; i < r; i++){

*int* d = det(point[i], point[l], point[r]);

        if (d > max || (d == max && point[i].x < point[x].x))

            max = d, x = i;

    }

    if (!x) return;

    ans[x] = true;

    dealup(l, x);

    dealup(x, r);

}

*void* dealdown(*int* *l*, *int* *r*){

    if (r - l + 1 <= 2){

        ans[l] = ans[r] = true;

        return;

    }

*int* max = 0, x = 0;

    for (*int* i = l + 1; i < r; i++){

*int* d = -det(point[i], point[l], point[r]);

        if (d > max || (d == max && point[i].x < point[x].x))

            max = d, x = i;

    }

    if (!x) return;

    ans[x] = true;

    dealdown(l, x);

    dealdown(x, r);

}

*int* main(){

*int* n;

    printf("请输入点的个数：");

    cin >> n;

    printf("点的坐标：");

    for (*int* i = 1; i <= n; i++) cin >> point[i].x >> point[i].y;

    sort(point + 1, point + n + 1);

    dealup(1, n);

    dealdown(1, n);

    cout << "凸包的顶点为：" << endl;

    for (*int* i = 1; i <= n; i++)

        if (ans[i]) cout << point[i].x << ' ' << point[i].y << endl;

    return 0;

}

