S. 153.

Horcsin Bálint

Budapest, Németh László Gimnázium, 12. o. t.

horcsinbalint@gmail.com

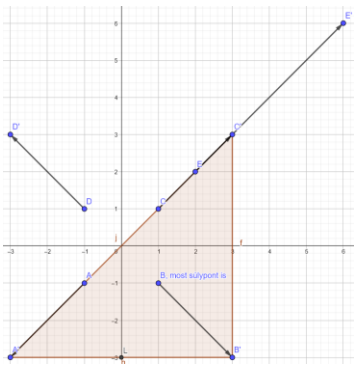
Programnyelv: C++17

Futtatási környezet: Ubuntu 20.04.2 LTS (WSL 2)

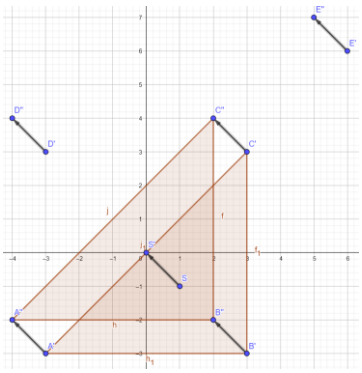
Fejlesztői környezet: VS Code

Megoldás elvi menete

Ismert, hogy egy háromszög belső pontja a súlypontja. Továbbá tudjuk, hogy a súlypont a 3 másik pont átlaga. Mivel a feladatban minden megadott pont egész, így, ha a mindegyiknek megszorozzuk 3-mal a koordinátáit, akkor a 3 pont átlaga is egész lesz. ($\frac{x_1*3+x_2*3+x_3*3}{3} = x_1 + x_2 + x_3, \frac{y_1*3+y_2*3+y_3*3}{3} = y_1 + y_2 + y_3$)

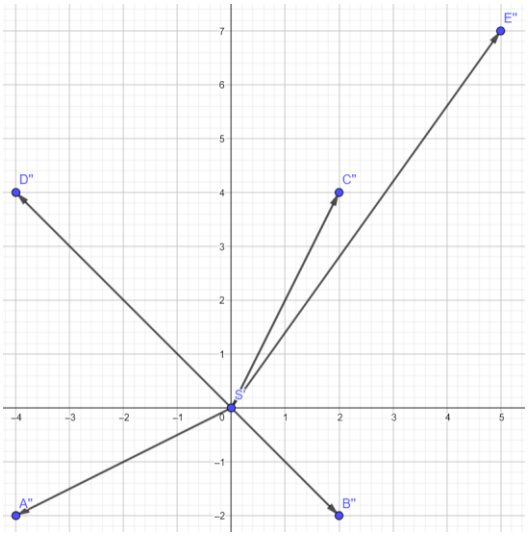


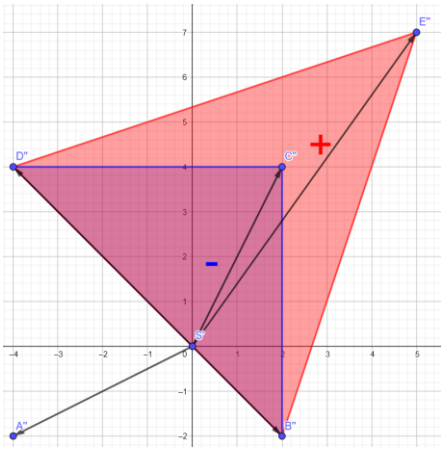
Ezt követően az összes pontot toljuk el úgy, hogy a derékszögű koordinátarendszer origójába essen a súlypont.



Mivel a súlypont benne volt a kezdeti háromszögben, így utána mindig benne lesz a konvex burokban.

Így az S-ből minden csúcsba mutató vektor iránya alapján rendezve vannak a pontok a konvex burkon is. Ha egy set-ben tároljuk, azt, hogy az addigi konvex burokban mely pontok vannak benne, akkor bináris kereséssel megkereshető, hogy az új hova kerülne a set-ben. Ha egy pont kiegészíti a burkot, akkor az a mellette lévő pontok esetleges törlésével és az új pont felvételével megoldható (ez jelenthet több pont törlését is akár). Hasonlóan ahhoz kell ezt elvégezni, ahogy a konvex burkot fel szokás venni, csak mindkét irányból meg kell csinálni. Közben számolható, hogy mely háromszögek kerülnek ki, illetve be.





E pont beszúrásakor így kell a háromszögeket levonni, hozzáadni.

A megoldás során figyelembe kell venni, hogy pontok bármely síknegyedben előfordulhatnak, valamint azt is, hogy a set-eknek van elejük és végük, itt viszont körbe kell záródnia az adattárolónak. Az első problémát egy elterjedt összehasonlító függvénnyel lehet megoldani, míg másodikhoz egy iterátort készítettem.

A végén a kapott kétszeres területet el kellett osztani 9-cel, az átméretezés miatt.

Futásidő

$$O(N * \log(N))$$

Megfelelőség

A megoldást random tesztesetekkel szemben kipróbáltam, az ellenőrzést az alábbi programmal végeztem (ezt nagyrészt másoltam, forrás: https://cp-algorithms.com/geometry/grahams-scan-convex-hull.html, https://cp-algorithms.com/geometry/area-of-simple-polygon.html):

```
#include<bits/stdc++.h>

using namespace std;

struct pt {
    long long x, y;
};

bool cmp(pt a, pt b) {
    return a.x < b.x || (a.x == b.x && a.y < b.y);
}

bool cw(pt a, pt b, pt c) {
    return a.x*(b.y-c.y)+b.x*(c.y-a.y)+c.x*(a.y-b.y) < 0;
}

bool ccw(pt a, pt b, pt c) {
    return a.x*(b.y-c.y)+b.x*(c.y-a.y)+c.x*(a.y-b.y) > 0;
}

void convex_hull(vector<pt>& a) {
    if (a.size() == 1)
        return;

    sort(a.begin(), a.end(), &cmp);
    pt p1 = a[0], p2 = a.back();
    vector<pt> up, down;
    up.push_back(p1);
    down.push_back(p1);
    for (int i = 1; i < (int)a.size(); i++) {
        if (i == a.size() - 1 || cw(p1, a[i], p2)) {
            while (up.size() >= 2 && !cw(up[up.size()-2], up[up.size()-1], a[i]))
                up.pop_back();
            up.push_back(a[i]);
        }
        if (i == a.size() - 1 || ccw(p1, a[i], p2)) {
            while(down.size() >= 2 && !ccw(down[down.size()-2], down[down.size()-
1], a[i]))
                down.pop_back();
            down.push_back(a[i]);
```

```
        }
    }

    a.clear();
    for (int i = 0; i < (int)up.size(); i++)
        a.push_back(up[i]);
    for (int i = down.size() - 2; i > 0; i--)
        a.push_back(down[i]);
}

long long area(const vector<pt>& fig) {
    long long res = 0;
    for (unsigned i = 0; i < fig.size(); i++) {
        pt p = i ? fig[i - 1] : fig.back();
        pt q = fig[i];
        res += (p.x - q.x) * (p.y + q.y);
    }
    return abs(res);
}

int main(){
ios::sync_with_stdio(0);
cin.tie(0); cout.tie(0);
    vector<pt> p(3);
    cin >> p[0].x >> p[0].y;
    cin >> p[1].x >> p[1].y;
    cin >> p[2].x >> p[2].y;
    cerr << area(p) << '\n';
    int n;
    cin >> n;
    for(int i=0;i<n;i++){
        pt pp;
        cin >> pp.x >> pp.y;
        p.push_back(pp);
        convex_hull(p);
        cout << area(p) << '\n';
    }
return 0;
}
```