## A-III-1 Bizónia rezervácia

Na vstupe máme $n$ priamok. Tie majú dokopy $n(n-1)/2$ priesečníkov. Hľadaná ohrada je zjavne konvexným obalom týchto $n$ bodov. Stačí teda všetkých $\Theta(n^2)$ bodov zostrojiť a následne v čase $\Theta(n^2 \log n)$ nájsť ich konvexný obal.
Trochu času vieme ušetriť tým, že si uvedomíme, že na každej priamke nám stačí nájsť „najpravejší“ a „najľavejší“ priesečník – tie medzi nimi určite nebudú vrcholmi konvexného obalu. Pre konkrétnu priamku vieme dva extrémne priesečníky nájsť v čase $\Theta(n)$, čiže dokopy strávime $\Theta(n^2)$ času tým, že si nájdeme množinu nanajvýš $2n$ zaujímavých bodov. Pre tieto potom zostrojíme konvexný obal. Keďže máme len $O(n)$ bodov, časová zložitosť zostrojenia konvexného obalu je zanedbateľná a celková časová zložitosť tohto druhého riešenia je teda $\Theta(n^2)$.

Existuje však aj výrazne efektívnejšie riešenie:

- Začneme tým, že si všetky priamky usporiadame podľa uhlu, ktorý zvierajú s osou x. (Pri implementácii toto vieme spraviť aj v celých číslach pomocou vektorového súčinu. Ekvivalentne sa na to môžeme dívať tak, že priamky usporadúvame podľa ich smernice – pri tom by ale bolo treba špeciálne ošetriť zvislú priamku, ak takú máme.)
- Následne spočítame priesečníky, ale len pre dvojice priamok, ktoré v usporiadanom poradí spolu susedia, a to vrátane priesečníku prvej a poslednej priamky.
- Na záver už len nájdeme konvexný obal pre $n$ priesečníkov, ktoré sme zostrojili v predchádzajúcom kroku.

Vyššie popísané riešenie má časovú zložitosť $\Theta(n \log n)$. Potrebujeme však dokázať, že naozaj funguje – teda že naozaj vždy všetky vrcholy hľadaného konvexného obalu sú priesečníkmi susedných priamok v našom usporiadanom poradí.

Jeden jednoduchý argument vyzerá nasledovne:
Na úvod si dokážeme, že najpravejší spomedzi všetkých priesečníkov je priesečníkom dvoch priamok, ktoré susedia v našom usporiadanom poradí. Predstavme si zvislú priamku $p$ napravo od posledného priesečníka. Priesečníky našich priamok s priamkou $p$ zodpovedajú nášmu usporiadanému poradiu – čím má priamka väčšiu smernicu, tým vyššie priamku $p$ pretne. No a keď teraz budeme posúvať priamku $p$ doľava, poradie, v ktorom ju naše priamky pretínajú, sa nebude meniť, až kým neprídeme k prvému (teda najpravejšiemu) priesečníku. A v ňom sa teda musia stretnúť niektoré dve priamky, ktoré v našom usporiadanom poradí susedili.

Práve sme si dokázali, že najpravejší spomedzi všetkých priesečníkov je priesečníkom dvoch priamok, ktoré susedia v našom usporiadanom poradí. Lenže smer „doprava“ nie je ničím špeciálny. Tento istý výsledok platí v úplne každom smere. Nech si vyberieme akýkoľvek smer, posledný priesečník v tom smere musí byť priesečníkom niektorých dvoch priamiek, ktoré v našom usporiadanom poradí susedia. (Dôkaz: Otočíme rovinu tak, aby nami zvolený smer padol na os x. Následne zopakujeme predchádzajúci argument.)

No a teraz celý dôkaz uzavrieme. Na to si už stačí len uvedomiť, že každý vrchol hľadaného konvexného obalu je nutne najvzdialenejším priesečníkom v nejakom smere.
Toto vyplýva z toho, že v každom vrchole má konvexný obal vnútorný uhol ostro menší ako 180°. Dotyčným vrcholom teda vieme preložiť priamku $q$ tak, aby celý konvexný obal ležal na jednej strane priamky $q$. Potom ale je náš vrchol zjavne najvzdialenejším priesečníkom v smere kolmom na priamku $q$.
Všetky vrcholy hľadaného konvexného obalu teda naozaj ležia medzi priesečníkmi priamok, ktoré susedia v poradí usporiadanom podľa uhlu.

### Listing programu (C++)

```
#include <algorithm>
#include <iostream>
#include <iomanip>
#include <cassert>
#include <vector>
#include <cmath>
```

```cpp
using namespace std;

const double EPSILON = 1e-7;

struct line { double a, b, c; };
struct point { double x, y; };

// operátor < na usporiadanie priamiek podľa smeru
bool operator< (const line &A, const line &B) {
    double ux = -A.b, uy = A.a, vx = -B.b, vy = B.a;
    if (uy < -EPSILON || (uy < EPSILON && ux < -EPSILON)) { ux = -ux; uy = -uy; }
    if (vy < -EPSILON || (vy < EPSILON && vx < -EPSILON)) { vx = -vx; vy = -vy; }
    return (ux*vy - vx*uy) > EPSILON;
};

// operátor < na usporiadanie bodov zľava doprava
bool operator< (const point &A, const point &B) {
    return A.x + EPSILON < B.x || ( abs(A.x-B.x) < EPSILON && A.y + EPSILON < B.y );
}

// vektorový súčin: vráti >0 / 0 / <0 podľa toho, či OAB zatáča proti smeru ručičiek / ide rovno / zatáča v smere
long long cross(const point &O, const point &A, const point &B) { return (A.x-O.x)*(B.y-O.y)-(A.y-O.y)*(B.x-O.x); }

// konvexný obal pomocou Grahamovho algoritmu
vector<point> convex_hull(vector<point> P) {
    int n = P.size(), k = 0;
    vector<point> H(2*n);
    for (int i = 0; i < n; i++) {
        while (k >= 2 && cross(H[k-2], H[k-1], P[i]) < EPSILON) k--;
        H[k++] = P[i];
    }
    for (int i = n-2, t = k+1; i >= 0; i--) {
        while (k >= t && cross(H[k-2], H[k-1], P[i]) < EPSILON) k--;
        H[k++] = P[i];
    }
    H.resize(k-1);
    return H;
}

// priesečník priamiek
point intersection(const line &A, const line &B) {
    point answer;
    answer.x = (A.b * B.c - B.b * A.c) / (B.b * A.a - A.b * B.a);
    answer.y = (A.a * B.c - B.a * A.c) / (A.b * B.a - A.a * B.b);
    return answer;
}

int main() {
    int N;
    cin >> N;
    vector<line> L(N);
    for (int n=0; n<N; ++n) cin >> L[n].a >> L[n].b >> L[n].c;
    sort(L.begin(), L.end());

    vector<point> P;
    for (int n=0; n<N; ++n) P.push_back( intersection( L[n], L[(n+1)%N] ) );
    sort(P.begin(), P.end());

    vector<point> H = convex_hull(P);
    for (const auto &pt : H) cout << pt.x << " " << pt.y << endl;
}
```

**TRIDSIATY DRUHÝ ROČNÍK OLYMPIÁDY V INFORMATIKE**

Príprava úloh: Michal Anderle, Eduard Batmendijn, Michal Forišek, Jaroslav Petrucha
Recenzia: Michal Forišek
Slovenská komisia Olympiády v informatike
Vydal: IUVENTA – Slovenský inštitút mládeže, Bratislava 2017