Algoritmi avansaţi

Laborator 5 (săpt. 9 şi 10)

– Soluţiile problemelor pot fi testate în aplicaţia de evaluare automată, la link-ul https://cms.fmi.unibuc.ro/.

– Problemele sunt diferite ca nivel de dificultate şi vor avea ponderi diferite. Pentru fiecare problemă, punctajul final se obţine înmulţind punctajul obţinut în aplicaţie cu o pondere indicată explicit. Punctajul va fi notat doar în urma prezentării / discutării soluţiei realizate.

– Respectaţi regulile din Regulamentul de etică şi profesionalism al FMI!

Problema 1. (0.2 * punctajul din aplicaţie)

Testul de orientare.

Descriere

Se dau trei puncte în plan, P, Q, R, de coordonate $P = (x_P, y_P)$, $Q = (x_Q, y_Q)$ și $R = (x_R, y_R)$. Să se stabilească poziția punctului R față de dreapta PQ, folosind testul de orientare descris în curs.

Date de intrare

Se va citi de la tastatură t, reprezentând numărul de teste. Următoarele t linii vor descrie fiecare câte un test. Fiecare linie conține șase numere întregi: x_P, y_P, x_Q, y_Q, x_R și y_R, reprezentând coordonatele punctelor P, Q, R.

Date de ieșire

Pentru fiecare test se va afișa, pe câte un rând separat, un mesaj corespunzator pozitiei punctului R:

- `LEFT` (dacă punctul R se află *la stânga* dreptei PQ)
- `RIGHT` (dacă punctul R se află *la dreapta* dreptei PQ)
- `TOUCH` (dacă punctul R se află *pe* dreapta PQ)

Restricții și precizări

- $1 \leq t \leq 10^5$
- $-10^9 \leq x_p, y_p, x_q, y_q, x_r, y_r \leq 10^9$

De asemenea, trebuie să aveți în vedere că în mediul de lucru de pe CMS **nu** aveți posibilitatea să importați biblioteci externe (de exemplu, nu puteți importa `numpy` ca să folosiți `numpy.linalg.det`).

Exemplu

Input

```
3
1 1 5 3 2 3
1 1 5 3 4 1
1 1 5 3 3 2
```

Output

```
LEFT
RIGHT
TOUCH
```

Explicație

Datele de mai sus corespund următoarei situații:

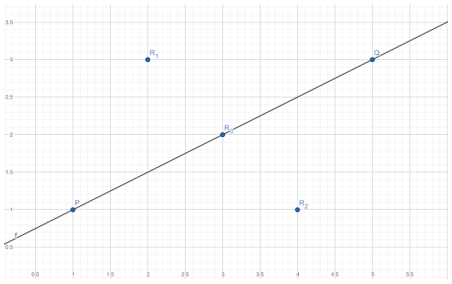

Figura 1: Reprezentarea grafică a celor trei puncte din exemplu

Problema 2. (0.3 * punctajul din aplicaţie)

Roby.

Descriere

Roby este un aspirator-robotel care are sarcina de a face curat într-o cameră. Robotelul pleacă dintr-un punct de start P_1 și apoi urmează un traseu care este o linie poligonală $P_1P_2 \dots P_nP_1$, la final robotelul întorcându-se și oprindu-se în P_1. Fiecare punct P_i este descris prin coordonatele sale (x_i, y_i). În fiecare punct P_i robotelul trebuie să vireze la stânga sau la dreapta sau să continue să meargă pe aceeași dreaptă.

La final, pe lângă curățarea camerei, Roby trebuie să indice numărul total de **viraje la stânga**, numărul total de **viraje la dreapta** și numărul de situații în care **a rămas pe aceeași dreaptă**. Ajutați-l pe Roby să își finalizeze cu bine sarcina, indicând cele trei numere.

Date de intrare

Datele de intrare se vor citi de la tastatură. Datele conțin pe prima linie un număr natural n. Pe urmatoarele n linii se află perechi de numere întregi, reprezentând coordonatele punctelor $P_1, P_2, \dots, P_n$, în această ordine. Pentru fiecare i, pentru punctul P_i sunt indicate pe aceeași linie coordonatele x_i și y_i, separate printr-un spațiu.

Date de ieșire

Se vor afișa pe o singură linie, separate prin spațiu, numarul total de viraje la stânga, numărul total de viraje la dreapta și numărul de situații în care a rămas pe aceeași dreaptă (în această ordine).

Restricții și precizări

- $3 \leq n \leq 10^5$.
- $-10\,000 \leq x_i, y_i \leq 10\,000$, $\forall i = \overline{1, n}$.
- Cazul de coliniaritate include situațiile următoare:
  1. robotelul continuă deplasarea în același sens;
  2. robotelul schimbă sensul deplasării rămânâd pe aceeași dreaptă;
  3. cel puțin două dintre punctele pentru care se realizează testarea coincid.

Exemplu

Input

```
7
1 1
2 2
2 0
3 0
4 0
5 0
6 0
```

Output

```
2 1 3
```

Explicație

Traseul parcurs de Roby are în total **6** viraje: **2** la stânga (în punctele P_3 și P_7), **1** la dreapta (în P_2) și are **3** puncte în care continuă drept înainte (în P_4, P_5 si P_6). În P_1 nu este realizat niciun viraj, deoarece robotelul se oprește.

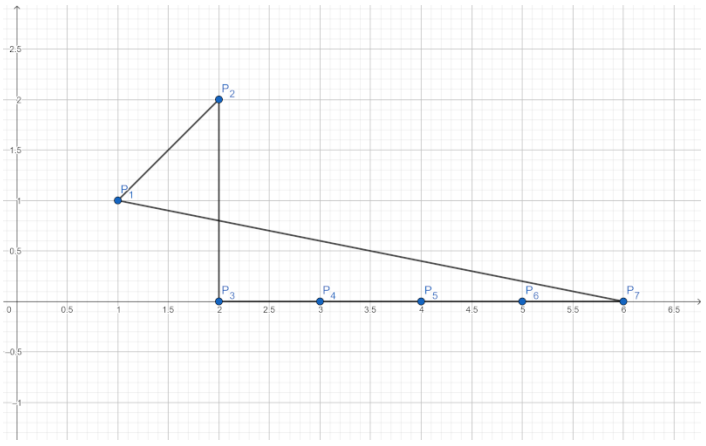

Figura 2: Reprezentarea grafică a traseului lui Roby

Problema 3. (0.5 * punctajul din aplicaţie)

Acoperirea convexă a unui poligon stelat

Descriere

Un poligon $P_1P_2 \dots P_nP_1$ se numește ***stelat*** dacă există un punct M în interiorul său astfel încât, oricum s-ar alege un punct X pe laturile poligonului sau un vârf al acestuia, segmentul $[MX]$ este conținut în întregime în interiorul poligonului.

Fiind dat un poligon stelat, trebuie să implementați un algoritm cu complexitate liniară de timp care să găsească acoperirea convexă a unui poligon stelat.

Date de intrare

Se va citi de la tastatură un număr n, reprezentând numărul de vârfuri al poligonului și apoi n linii care conțin perechi de numere întregi x_iy_i, separate prin spațiu, reprezentând coordonatele vârfului P_i, **parcurse în sens trigonometric**.

Date de ieșire

Programul va afișa un număr k, reprezentând numărul de vârfuri al acoperirii convexe a mulțimii $P_1, P_2, \dots, P_n$ și apoi k perechi de numere întregi, pe linii distincte, reprezentând coordonatele acestor vârfuri, **parcurse tot în sens trigonometric** (dar puteți porni de la orice vârf).

Restricții și precizări

- $1 \leq n \leq 100\,000$
- $-10^9 \leq x_i, y_i \leq 10^9$

Exemple

Exemplul 1

Input

```
3
-1 3
-3 -2
4 -3
```

Output

```
3
-1 3
-3 -2
4 -3
```

Explicație

Exemplul corespunde următorului poligon stelat, un triunghi oarecare:

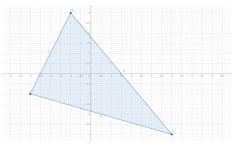

Figura 3: Triunghi oarecare, a cărui acoperire convexă este chiar el însuși

Puteți începe să descrieți acoperirea convexă de la orice vârf al ei, cât timp parcurgerea este în sens trigonometric. $(-3, 2), (4, -3), (-1, 3)$ și $(4, -3), (-1, 3), (-3, 2)$ erau de asemenea soluții acceptabile.

Exemplul 2

Input

```
10
0 3
-1 1
-5 0
-2 -1
-4 -5
1 -2
5 -3
3 0
6 3
2 2
```

Output

```
5
-4 -5
5 -3
6 3
0 3
-5 0
```

Explicație

Exemplul corespunde următorului poligon stelat, o stea neregulată cu 5 colțuri:

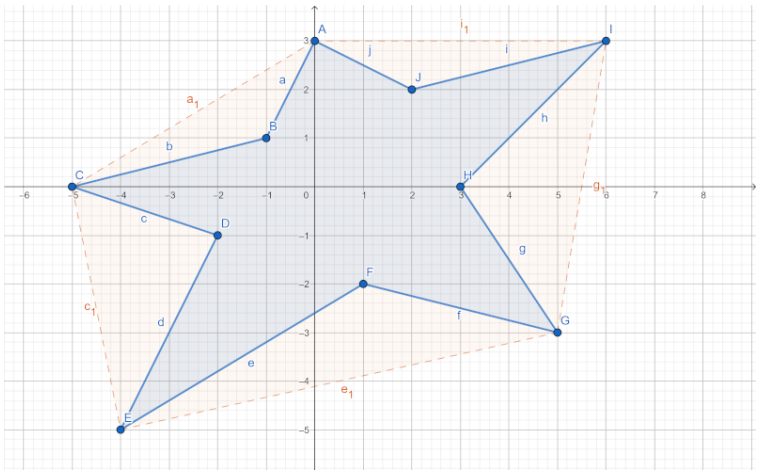

Figura 4: Stea cu cinci colțuri, a cărui acoperire convexă este formată dintr-o submulțime a vârfurilor sale

Exemplul 3

Input

```
8
0 2
-2 2
-2 0
-2 -2
0 -2
2 -2
2 0
2 2
```

Output

```
4
-2 2
-2 -2
2 -2
2 2
```

Explicație

Exemplul dat este o stea cu 4 colțuri degenerată, care e de fapt un pătrat:

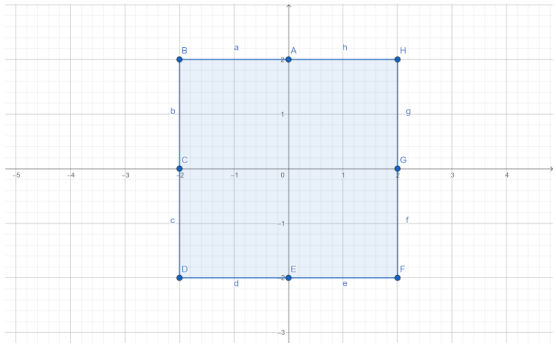

Figura 5: Pătrat

Problema 4. (Suplimentar)

Travelling Salesman Problem - Convex Hull

Descriere

Implementați algoritmul care construiește, în context euclidian, un traseu optim pentru *Travelling Salesman Problem* folosind acoperirea convexă.

Date de intrare

Se vor citi de la tastatură n, numărul de puncte din plan, și apoi n perechi de numere întregi x_i, y_i, reprezentând coordonatele punctelor.

Date de ieșire

Se vor afisa pe ecran vârfurile unui ciclu hamiltonian de cost minim, primul vârf din ciclu fiind cel cu abscisa minimă.

Restricții și precizări

- $3 \leq n \leq 1\,000$.
- $-1\,000 \leq x_i, y_i \leq 1\,000$

Exemple

Input

```
10
6 10
0 5
4 -7
3 8
3 -8
-4 -2
-10 -1
0 -9
4 -3
-7 10
```

Output

```
-10 -1
-4 -2
0 -9
3 -8
4 -7
4 -3
6 10
3 8
0 5
-7 10
-10 -1
```