

Algoritmi avansați

Laborator 5 (săpt. 9 și 10)

1. (0,5p) Implementați / utilizați testul de orientare.

Input. Trei puncte $P = (x_P, y_P), Q = (x_Q, y_Q), R = (x_R, y_R)$ (în această ordine) din \mathbb{R}^2 .

Output. Programul afișează natura virajului PQR (viraj la stânga, viraj la dreapta, puncte coliniare).

2. (1p) Roby

Descriere. Roby este un aspirator-robotel care are sarcina de a face curat într-o camera. 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 oprindu-se în P_1 . Fiecare punct P_i este indicat 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 dreapta. 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 dreapta. Ajutați-l pe Roby să își finalizeze cu bine sarcina, indicând cele trei numere.

Date de intrare. Fisierul de intrare `robby.in` conține pe prima linie un număr natural n . Pe următoarele 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 prin spațiu.

Date de ieșire. Fisierul de ieșire `robby.out` va conține pe o singură linie, separate prin spațiu, numărul total de viraje la stânga, numărul total de viraje la dreapta și numărul de situații în care rămas pe aceeași dreapta (în această ordine).

Restricții și precizări.

- $3 \leq n \leq 1\,000$.
- $-10\,000 < x_i, y_i < 10\,000, \forall i = \overline{1, n}$.
- Cazul de coliniaritate include situațiile următoare: (i) robotelul continuă deplasarea în același sens, (ii) robotelul schimbă sensul deplasării rămânând pe aceeași dreapta, (iii) cel puțin două dintre punctele pentru care se realizează testarea coincid.

Exemplu.

Input

robby.in

7

1 1

2 2

2 0

3 0

4 0

5 0

6 0

Output

robby.out

2 1 3

Explicatie Traseul parcurs de Roby are in total 6 viraje: 2 la stanga (in punctele P_3 si P_7), 1 la dreapta (in P_2) si are 3 puncte in care continua drept inainte (in P_4 , P_5 si P_6). In P_1 nu este realizat niciun viraj, deoarece robotul se opreste.

3. (0,5p) *Algoritm cu complexitate-timp liniară pentru frontiera acoperirii convexe a unui poligon stelat dat.*

Input. Numărul de vârfuri n , vârfurile poligonului: $P_1 = (x_{P_1}, y_{P_1}), P_2 = (x_{P_2}, y_{P_2}), \dots, P_n = (x_{P_n}, y_{P_n})$ (în această ordine) din \mathbb{R}^2 .

Output. Programul afișează vârfurile acoperirii convexe a mulțimii $\{P_1, \dots, P_n\}$.

Precizare. Pentru testare, $P_1 P_2 \dots P_n$ reprezintă un poligon parcurs în sens trigonometric (aceste ipoteze nu mai trebuie verificate). Un poligon este stelat dacă există un punct M în interiorul său astfel ca, oricum se alege un punct X pe laturile poligonului sau vârf al acestuia, segmentul $[MX]$ este conținut în întregime în interiorul poligonului. Algoritmul va avea complexitatea-timp liniară.

4. (1p) *Algoritm eficient pentru stabilirea poziției unui punct față de un poligon convex.*

Input. Numărul de vârfuri n , vârfurile poligonului convex $P_1 = (x_{P_1}, y_{P_1}), P_2 = (x_{P_2}, y_{P_2}), \dots, P_n = (x_{P_n}, y_{P_n})$ (în această ordine), un punct Q din \mathbb{R}^2 .

Output. Programul afișează poziția relativă a punctului Q față de poligon (în interior, în exterior, pe laturi).

Precizare. Pentru testare, $P_1 P_2 \dots P_n$ reprezintă un poligon convex parcurs în sens trigonometric (acest lucru nu mai trebuie verificat). Algoritmul va fi cât mai eficient.

5. (Suplimentar) *Implementați **algoritmul** care construiește, în context euclidian, un traseu optim pentru TSP folosind acoperirea convexă.*