

Algoritmi avansați

Laborator 2 (săpt. 3 și 4)

Problema 1. (0.4 * punctajul din aplicație)

Punct în poligon convex

Descriere

Se consideră un poligon convex cu n vârfuri date în ordine trigonometrică $(P_1 P_2 \dots P_n)$ și m puncte în plan (R_1, R_2, \dots, R_m) . Pentru fiecare dintre cele m puncte să se stabilească dacă se află în **interiorul**, în **exteriorul** sau **pe una dintre laturile** poligonului.

Date de intrare

Se citește de la tastatură n , reprezentând numărul de vârfuri ale poligonului. Următoarele n linii vor conține câte două numere întregi x_i, y_i , coordonatele punctului P_i .

Pe următoarea linie se află m reprezentând numărul de puncte pentru care trebuie să aflăm poziția față de poligon. Următoarele m linii vor conține câte două numere întregi x_i, y_i , coordonatele punctului R_i .

Date de ieșire

Pentru fiecare punct R_i se va afișa, pe câte un rând nou, un mesaj corespunzător poziției sale față de poligon:

- **INSIDE** (dacă punctul R_i se află în poligon)
- **OUTSIDE** (dacă punctul R_i se află în afara poligonului)
- **BOUNDARY** (dacă punctul R_i se află pe una dintre laturile poligonului)

Restricții și precizări

- $3 \leq n, m \leq 10^5$.
- $-10^9 \leq x_i, y_i \leq 10^9$

Exemplu

Input

```
4
0 0
5 0
5 5
0 5
3
2 2
7 7
5 2
```

Output

```
INSIDE
OUTSIDE
BOUNDARY
```

Explicație

Reprezentarea grafică a situației de mai sus este următoarea:

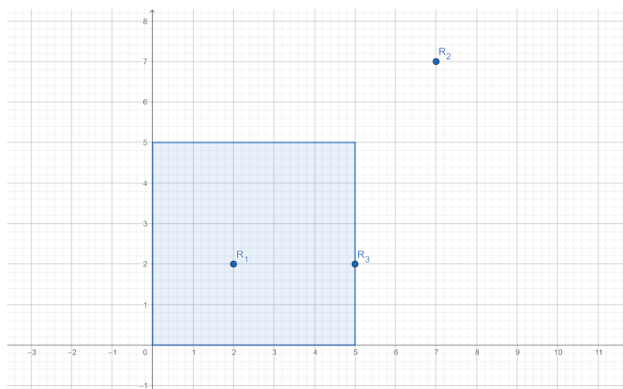


Figura 1: Reprezentarea grafică a poligonului și a punctelor date ca exemplu

Indicații de rezolvare

O metodă simplă de a verifica dacă un punct se află în interiorul unui poligon convex este descrisă [aici](#) și se bazează pe efectuarea **testului de orientare** între fiecare latură a poligonului convex și punctul ales. O astfel de verificare necesită $\mathcal{O}(n)$ timp, deci per total soluția are complexitatea-timp $\mathcal{O}(mn)$.

Pentru a trece toate testele, trebuie să implementați o soluție care să ruleze în timp $\mathcal{O}(m \log n)$. Un astfel de algoritm, care utilizează o căutare binară, este descris [pe acest site](#), respectiv la pagina 2 din [acest PDF](#).

Problema 2. (0.3 * punctajul din aplicație)

Poziția unui punct față de un poligon.

Descriere

Implementați un algoritm de complexitate de timp liniară care să determine poziția relativă a unui punct Q față de un poligon *arbitrar* P_1, \dots, P_n .

Date de intrare

Programul va citi de la tastatură un număr natural n și apoi n perechi de numere întregi separate prin spațiu $x_i y_i$, pe linii distincte, reprezentând coordonatele vârfului $P_i(x_i, y_i)$ al poligonului.

După aceea urmează numărul natural m și apoi m perechi de numere întregi separate prin spațiu $x_j y_j$, reprezentând coordonatele punctului $Q_j(x_j, y_j)$.

Date de ieșire

Pentru fiecare dintre cele m puncte, programul va afișa pe ecran:

- **INSIDE**: dacă punctul Q_j se află în interiorul poligonului;
- **OUTSIDE**: dacă punctul Q_j se află în exteriorul poligonului;
- **BOUNDARY**: dacă punctul Q_j se află pe laturile poligonului.

Restricții și precizări

- $3 \leq n \leq 1000$
- $1 \leq m \leq 1000$
- $-10^9 \leq x, y \leq 10^9$

Exemplu

Input

```
12
0 6
0 0
6 0
6 6
2 6
2 2
4 2
4 5
5 5
5 1
1 1
1 6
3
3 4
7 3
3 2
```

Output

```
INSIDE
OUTSIDE
BOUNDARY
```

Explicație

Reprezentarea grafică a situației de mai sus este:

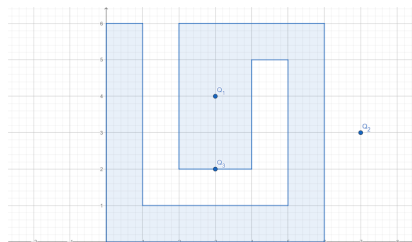


Figura 2: Reprezentare grafică a poligonului și a punctelor care trebuie verificate.

Indicații de rezolvare

Varianta 1 (*O soluție incompletă, care permite obținerea unui punctaj parțial*)

Puteți folosi **problema 1 de la L2**, care rezolvă cerința în cazul poligoanelor convexe. Combinând cu soluția **problemei 3 de la L1**, se ajunge la o soluție în cazul poligoanelor stelate.

Varianta 2 (*O soluție completă, bazată pe o abordare diferită*)

Soluția completă se bazează pe regula "par-impar" ("odd-even rule"), principiu folosit pentru a delimita **interiorul unui poligon** sau al unei **linii poligonale cu autointersecții**. Numele de "par-impar" derivă din următorul mecanism (descrie pe scurt):

- Se alege un punct M "departe" de poligon (de exemplu coordonatele lui M să fie mai mari / mai mici decât coordonatele corespunzătoare ale tuturor vârfurilor poligonului).
- Se determină numărul de laturi intersectate de **segmentul deschis** (MQ) **în interior**. Dacă acest număr este par, punctul Q este situat în exteriorul poligonului, iar dacă este impar, punctul este situat în interior.
- O implementare completă trebuie să trateze corect cazul în care punctul Q este situat pe una din laturile poligonului. De asemenea, dacă segmentul (MQ) trece printr-un vârf al poligonului, trebuie ales un alt punct "departe" de poligon. Se demonstrează că numărul total de intersecții se poate modifica, **dar paritatea rămâne neschimbată**.
- În exemplul din figura 3, pentru punctele Q_1 și Q_2 , numărul de intersecții dintre segmentele (MQ_1), respectiv (MQ_2) este par (4, respectiv 0), punctele fiind situate în exteriorul poligonului. Pentru punctul Q_3 , numărul de intersecții este impar (5), punctul fiind situat în interiorul poligonului.

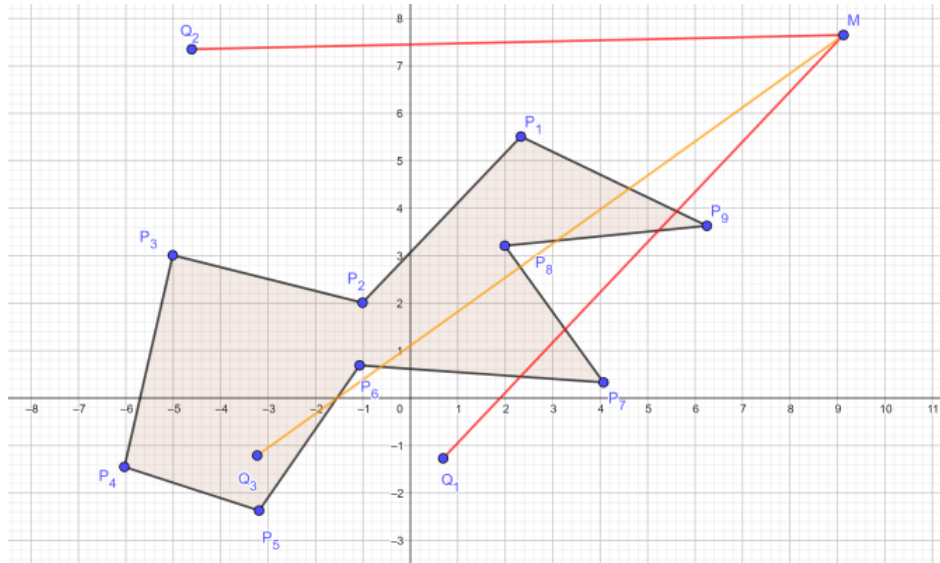


Figura 3: Exemplu pentru regula par-impar.

- Două segmente deschise (AB) și (CD) se intersectează în interior dacă și numai dacă A și B sunt de o parte și de alta a dreptei CD și C și D sunt de o parte și de alta a dreptei AB . Aceste proprietăți se verifică aplicând testul de orientare.
- În figura 4, segmentele deschise (AB) și (CD) se intersectează, fiind verificată proprietatea de mai sus. Observați că segmentele (AB) și (CE) **nu** se intersectează. Astfel, C și E sunt de o parte și de alta a dreptei AB , dar A și B nu sunt de o parte și de alta a dreptei CE . De asemenea, segmentele deschise (AB) și (CF) nu se intersectează (A este situat pe dreapta CF , deci A și B nu pot fi de o parte și de alta a dreptei CF).

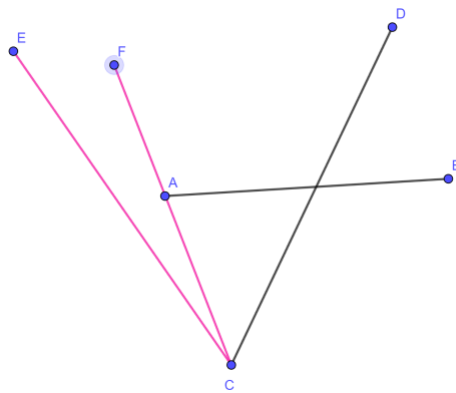


Figura 4: Intersecția unor segmente.

Problema 3. (0.3 * punctajul din aplicație)

Monotonia unui poligon

Descriere

Implementați un algoritm de complexitate de timp liniară care să verifice dacă un poligon $P_1 P_2 \dots P_n$ este **monoton** în raport cu axa Ox , respectiv Oy , folosind metoda dreptei de baleiere, descrisă în **cursul 2**.

Date de intrare

Programul va citi de la tastatură un număr natural n , reprezentând numărul de vârfuri ale poligonului, și apoi n perechi de numere întregi separate prin spațiu $x_i y_i$, pe linii distincte, reprezentând coordonatele vârfului $P_i(x_i, y_i)$ al poligonului.

Date de ieșire

Programul va afișa exact **două** rânduri, pe fiecare aflându-se unul dintre șirurile de caractere YES sau NO.

Primul rând va indica dacă poligonul dat este x -monoton, iar al doilea rând indică dacă este y -monoton.

Restricții și precizări

- $3 \leq n \leq 1\,000\,000$
- $-10^9 \leq x, y \leq 10^9$

Exemple

Exemplul 1

Input

```
8
-3 -1
-1 -4
9 -2
7 1
4 2
2 4
1 8
-2 6
```

Output

```
YES
YES
```

Explicație

Poligonul dat este atât x -monoton, cât și y -monoton.

Explicație pentru x -monotonie: vârful P_1 , situat cel mai la stânga (cu cel mai mic x) este unit cu vârful P_3 , situat cel mai la dreapta (cu cel mai mare x) prin două lanțuri: $P_1P_2P_3$, respectiv $P_1P_8P_7P_6P_5P_4P_3$. În ambele cazuri parcurgerea se efectuează de la stânga la dreapta (coordonata x crește). Se poate observa că intersecția dintre o dreaptă verticală oarecare și poligon este mulțimea vidă sau un punct sau un segment (de fapt, este o mulțime conexă, formată "dintr-o singură bucată").

Explicație pentru y -monotonie: vârful P_7 , situat cel mai sus (cu cel mai mare y) este unit cu vârful P_2 situat cel mai jos (cu cel mai mic y) prin două lanțuri: $P_7P_8P_1P_2$, respectiv $P_7P_6P_5P_4P_3P_2$. În ambele cazuri parcurgerea se efectuează de sus în jos (coordonata y descrește). Se poate observa că intersecția dintre o dreaptă orizontală oarecare și poligon este mulțimea vidă sau un punct sau un segment.

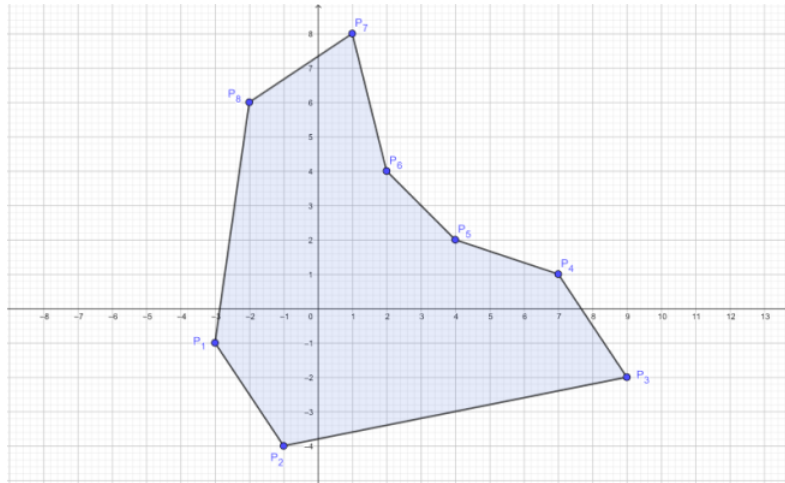


Figura 5: Poligonul este x -monoton și y -monoton.

Exemplul 2

Input

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

Output

```
NO
YES
```

Explicație

Poligonul dat nu este x -monoton, dar este y -monoton.

Poligonul nu este x -monoton. Putem observa că pe lanțul $P_1P_2P_3P_4, \dots$ coordonata x a punctelor crește, apoi scade și crește din nou. Se poate observa că există drepte verticale (de exemplu dreapta de ecuație $x = 5$) pentru care intersecția cu poligonul este reuniunea a două segmente (o astfel de mulțime nu este conexă, ea are două componente conexe).

Poligonul dat este y -monoton. Vârful P_7 , situat cel mai sus (cu cel mai mare y), este unit cu vârful P_4 , situat cel mai jos (cu cel mai mic y), prin două lanțuri: $P_7P_1P_2P_3P_4$, respectiv $P_7P_6P_5P_4$. În ambele cazuri parcurgerea se efectuează de sus în jos (adică y descrește). Se poate observa că intersecția dintre o dreaptă orizontală și poligon este mulțimea vidă sau un punct sau un segment.

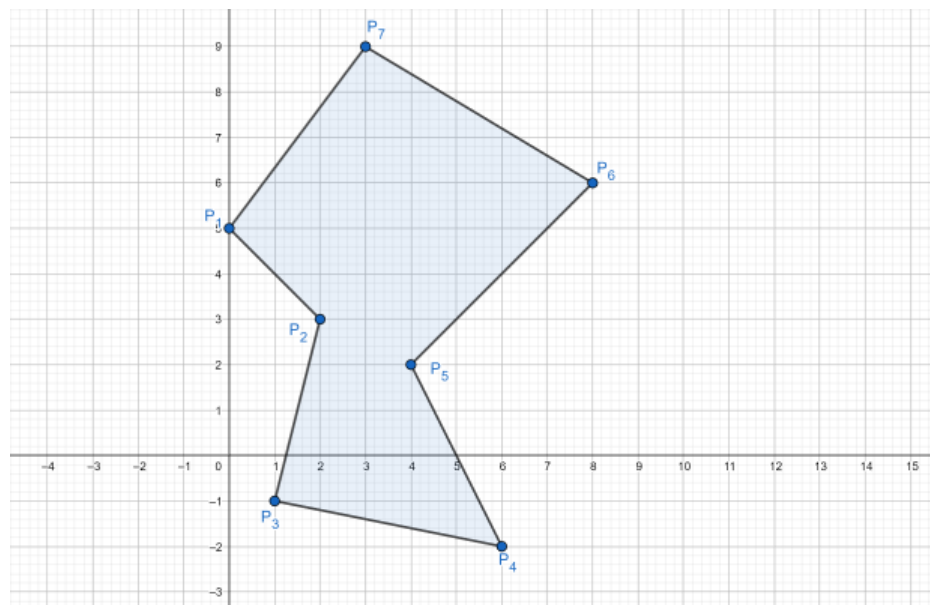


Figura 6: Poligonul nu este y -monoton, dar este y -monoton.

Exemplul 3

Input

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

Output

NO

NO

Explicație

Poligonul dat nu este nici x -monoton, nici y -monoton. Pe lanțul $P_5P_6P_7P_8$ coordonata x crește, apoi descrește, apoi crește din nou, deci poligonul nu este x -monoton. Un argument analog poate fi utilizat pentru a arăta că poligonul nu este y -monoton (găsiți un lanț care "obstrucționează" y -monotonia).

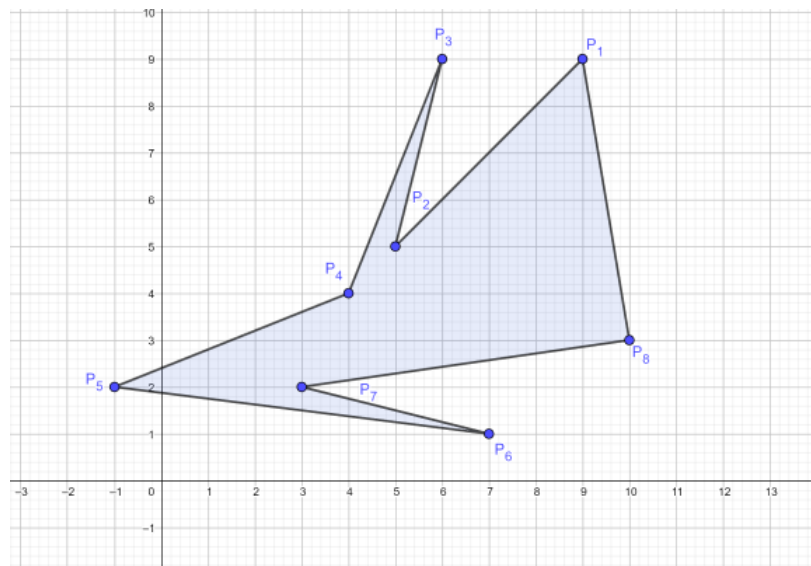


Figura 7: Poligonul nu este nici x -monoton, nici y -monoton