



Algoritmi geometrici

Prof. Doru Popescu Anastasiu,
C.N. ”Radu Greceanu”, Slatina, Jud. Olt

1. Noțiuni generale

1.1 Drepte

Ecuția unei drepte în plan

Forma generală a ecuației unei drepte d în sistemul de coordonate xOy este următoarea:

$$d: a \cdot x + b \cdot y + c = 0 \quad (1).$$

Orice punct $M(x,y)$ de pe dreapta d verifică ecuația (1).

Ecuția unei drepte care trece prin două puncte date $A(x_A, y_A)$ și $B(x_B, y_B)$ este următoarea:

$\frac{x - x_A}{y - y_A} = \frac{x_B - x_A}{y_B - y_A}$ (2), sau echivalent cu aceasta: $(x - x_A)(y_B - y_A) = (x_B - x_A)(y - y_A)$, pentru a evita situațiile când numitorul se anulează. O altă formă a acestei ecuații este

$$\begin{vmatrix} x & y & 1 \\ x_A & y_A & 1 \\ x_B & y_B & 1 \end{vmatrix} = 0 \quad (3).$$

După efectuarea calculelor în (2) sau (3) se obține ecuația (1), în care

$$\begin{aligned} a &= y_A - y_B \\ b &= x_B - x_A \\ c &= -x_B \cdot y_A + y_B \cdot x_A \end{aligned} \quad (4)$$

Ecuțiile parametrice ale unei drepte

Considerăm două puncte distincte $M_1(x_1, y_1)$ și $M_2(x_2, y_2)$. Un punct $M(x, y)$ se găsește pe dreapta M_1M_2 , dacă

$$\begin{aligned} x &= x_1 + (x_2 - x_1)t \\ y &= y_1 + (y_2 - y_1)t, \quad t \text{ număr real.} \end{aligned} \quad (5)$$

Ecuțiile (5) se numesc ecuațiile parametrice ale dreptei M_1M_2 .

Dacă în (5) luăm t număr real din intervalul $[0,1]$, atunci obținem ecuațiile parametrice ale segmentului $[M1M2]$.

Panta unei drepte

Panta unei drepte reprezintă tangenta unghiului făcut de dreaptă cu axa Ox .

Panta unei drepte d dată prin ecuația (1) este $m = -\frac{a}{b}$, iar pentru dreapta dată de ecuația (2)

sau (3) este $m = -\frac{x_B - x_A}{y_B - y_A}$.

Proprietăți

- 1) Două drepte sunt paralele dacă și numai dacă pantele lor sunt egale.
- 2) Două drepte sunt perpendiculare dacă și numai dacă produsul pantelor este -1 .

Ecuația unei drepte care trece printr-un punct și are o pantă dată

Fie $M1(x_1, y_1)$ și un număr real m . Ecuația dreptei care trece prin punctul $M1$ și are panta m este

$$y - y_1 = m \cdot (x - x_1)$$

Poziția unui punct față de o dreaptă

Considerăm o dreaptă d dată prin ecuația $a \cdot x + b \cdot y + c = 0$ și un punct $M(x_M, y_M)$.

Atașăm dreptei d funcția $f : R \times R \rightarrow R$, $f(x, y) = a \cdot x + b \cdot y + c$.

Dreapta d împarte planul în două semiplane, $S+$ și $S-$ definite astfel:

$S+$ este mulțimea punctelor $M(x_M, y_M)$ care îndeplinesc condiția $f(x_M, y_M) > 0$.

$S-$ este mulțimea punctelor $M(x_M, y_M)$ care îndeplinesc condiția $f(x_M, y_M) < 0$.

Punctele $M(x_M, y_M)$ de pe dreapta d îndeplinesc condiția $f(x_M, y_M) = 0$.

Intersecția a două drepte

Două drepte $d1$ și $d2$, de ecuații

$$d1: a_1 \cdot x + b_1 \cdot y + c_1 = 0$$

$$d2: a_2 \cdot x + b_2 \cdot y + c_2 = 0$$

Găsirea punctului de intersecție a dreptelor $d1$ și $d2$ presupune rezolvarea sistemului:

$$\begin{cases} a_1 \cdot x + b_1 \cdot y + c_1 = 0 \\ a_2 \cdot x + b_2 \cdot y + c_2 = 0 \end{cases}$$

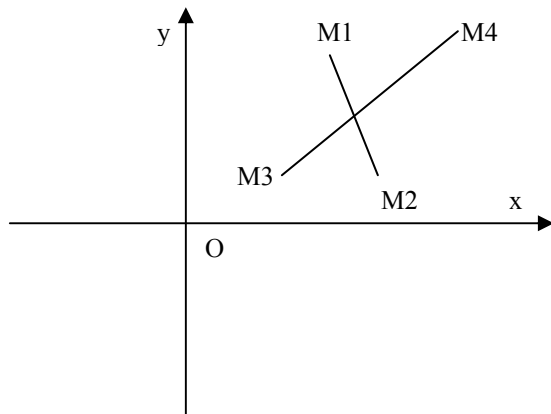
Acest sistem are soluție unică, dacă pantele celor două drepte sunt diferite, adică dacă $a_1 \cdot b_2 \neq a_2 \cdot b_1$. Dacă această condiție este îndeplinită atunci punctul de intersecție al dreptelor $d1$ și $d2$ are coordonatele:

$$x = \frac{b_1 \cdot c_2 - b_2 \cdot c_1}{a_1 \cdot b_2 - a_2 \cdot b_1}$$

$$y = \frac{c_1 \cdot a_2 - c_2 \cdot a_1}{a_1 \cdot b_2 - a_2 \cdot b_1}$$

Intersecția a două segmente

Considerăm segmentele $[M1M2]$, $[M3M4]$, date prin coordonatele capetelor. $M1(x1,y1)$, $M2(x2,y2)$, $M3(x3,y3)$, $M4(x4,y4)$.



Pentru ca cele două segmente să se intersecteze trebuie ca:

- $M1$ și $M2$ să se găsească în semiplane (închise) diferite delimitate de dreapta $M3M4$
- $M3$ și $M4$ să se găsească în semiplane (închise) diferite delimitate de dreapta $M1M2$

Dacă ecuația dreptei $M1M2$ este $a1 \cdot x + b1 \cdot y + c1 = 0$, iar cea a dreptei $M3M4$ este $a2 \cdot x + b2 \cdot y + c2 = 0$, atunci cele două condiții anterioare sunt echivalente cu

$$(a2 \cdot x1 + b2 \cdot y1 + c2) \cdot (a2 \cdot x2 + b2 \cdot y2 + c2) \leq 0 \text{ și}$$

$$(a1 \cdot x3 + b1 \cdot y3 + c1) \cdot (a1 \cdot x4 + b1 \cdot y4 + c1) \leq 0$$

1.2 Puncte. Distanțe

Distanța dintre două puncte

Considerăm două puncte $M1(x1,y1)$, $M2(x2,y2)$.

Distanța dintre $M1$ și $M2$ este dată de relația

$$M1M2 = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$

Puncte interior unui segment

Un punct $M(x,y)$ este interior unui segment $[M1M2]$, $M1(x1,y1)$, $M2(x2,y2)$ dacă $M1M2 = M1M + MM2$, adică dacă

$$\sqrt{(x1 - x2)^2 + (y1 - y2)^2} = \sqrt{(x1 - x)^2 + (y1 - y)^2} + \sqrt{(x - x2)^2 + (y - y2)^2}$$

Distanța de la un punct la o dreaptă

Pentru o dreaptă $d: a \cdot x + b \cdot y + c = 0$ și un punct $M0(x0,y0)$, distanța de la M la d este egală cu

$$\text{dist}(M0, d) = \frac{|a \cdot x0 + b \cdot y0 + c|}{\sqrt{a^2 + b^2}}$$

Distanța de la un punct la un segment

Considerăm un segment $[M_1M_2]$, cu $M_1(x_1, y_1)$, $M_2(x_2, y_2)$ și un punct $M(x, y)$. Prin distanța de la M la segmentul $[M_1M_2]$ înțelegem lungimea celui mai scurt segment cu un capăt în M și celălalt pe $[M_1M_2]$.

$$\text{dist}(M, [M_1M_2]) = \begin{cases} \text{dist}(M, M_1M_2), & \text{daca proiectia lui } M \text{ pe } M_1M_2 \text{ apartine lui } [M_1M_2] \\ \min\{MM_1, MM_2\}, & \text{altfel} \end{cases}$$

Centrul de greutate

Centrul de greutate al unui sistem omogen de n puncte: $M_1(x_1, y_1)$, $M_2(x_2, y_2)$, ..., $M_n(x_n, y_n)$ este un punct $G(x_G, y_G)$, cu

$$x_G = \frac{x_1 + x_2 + \dots + x_n}{n}$$

$$y_G = \frac{y_1 + y_2 + \dots + y_n}{n}$$

Aria unui triunghi

1. Dacă triunghiul este dat prin lungimile laturilor: a , b , c , atunci pentru calculul ariei se folosește formula lui Heron:

$$S = \sqrt{p(p-a)(p-b)(p-c)}, \text{ unde } p = \frac{a+b+c}{2}$$

2. Dacă triunghiul este dat prin coordonatele vârfurilor sale $A(x_A, y_A)$, $B(x_B, y_B)$, $C(x_C, y_C)$, atunci

$$S = \frac{1}{2} \text{abs} \begin{vmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{vmatrix}, \text{ adică } S = \frac{1}{2} \text{abs}(x_A \cdot y_B + x_B \cdot y_C + y_A \cdot x_C - x_C \cdot y_B - x_A \cdot y_C - x_B \cdot y_A)$$

Aria unui poligon convex

Pentru calculul ariei unui poligon convex $A_1A_2\dots A_n$, $A_i(x_i, y_i)$, $i \in \{1, 2, \dots, n\}$, se realizează o partiție a suprafeței poligonale în triunghiuri, de exemplu:

$$\text{aria}(A_1A_2\dots A_n) = \text{aria}(A_1A_2A_3) + \text{aria}(A_1A_3A_4) + \dots + \text{aria}(A_1A_{n-1}A_n)$$

O modalitate de calcul a ariei unui poligon (convex sau nu) este data de formula:

$$\text{aria}(A_1A_2\dots A_n) = \left| \sum_{i=1}^n x_i (y_{i+1} - y_{i-1}) \right| / 2 \text{ (vezi [5])}$$

Punct interior unui poligon convex

Un punct $M(x,y)$ se găsește în interiorul sau pe frontiera unui poligon convex dacă

$$\text{aria}(A_1A_2\dots A_n) = \text{aria}(MA_1A_2) + \text{aria}(MA_2A_3) + \dots + \text{aria}(MA_nA_1)$$

Rapoarte

Fie $M_1(x_1,y_1)$ și $M_2(x_2,y_2)$ distincte. $M(x,y)$ un punct pe dreapta M_1M_2 , diferit de M_2 . Numărul k definit astfel

$$k = \begin{cases} -\frac{MM_1}{MM_2}, & \text{dacă } M \in [M_1M_2] \\ \frac{MM_1}{MM_2}, & \text{dacă } M \notin [M_1M_2] \end{cases}$$

se numește raportul în care M împarte segmentul $[M_1M_2]$. Fiind cunoscut numărul real $k \neq 1$, coordonatele lui M , care împarte segmentul $[M_1M_2]$ în raportul k sunt

$$x = \frac{x_1 - kx_2}{1 - k}$$

$$y = \frac{y_1 - ky_2}{1 - k}.$$

Caz particular

M este mijlocul segmentului $[M_1M_2]$, $k = -1$. În acest caz coordonatele lui M sunt

$$x = \frac{x_1 + x_2}{2}$$

$$y = \frac{y_1 + y_2}{2}$$

Probleme propuse

1. Se dau două segmente prin coordonatele capetelor lor. Se cere să se verifice dacă segmentele se intersectează și în caz afirmativ să se determine coordonatele punctului de intersecție.
2. Se dă un poligon prin coordonatele vârfurilor sale și se cere să se verifice dacă este convex.
3. Se dă un punct M și un poligon convex P prin coordonatele vârfurilor. Se cere să se determine distanța de la punctul M la poligonul P (lungimea celui mai mic segment de la punctul M la un punct de pe poligonul P).
4. Se dă un poligon convex P prin coordonatele vârfurilor sale și un punct M dat prin coordonate. Se cere să se determine poziția punctului M față de poligonul P (interior, exterior sau pe poligon).

2. Înfășurătoarea convexă

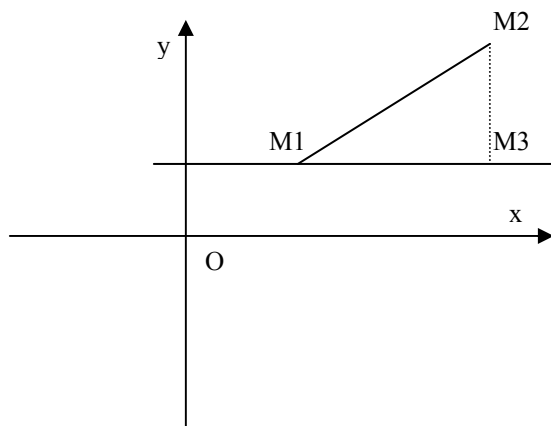
Se consideră n puncte în plan A_1, A_2, \dots, A_n , date prin coordonate: $(x_1,y_1), (x_2,y_2), \dots, (x_n,y_n)$. Vrem să determinăm un poligon convex care aibă vârfurile printre punctele date și să conțină pe laturi sau în interior celelalte puncte. Acest poligon se numește *înfășurătoarea convexă* asociată punctelor A_1, A_2, \dots, A_n .

Un algoritm cu complexitate $O(n \log n)$ care poate fi implementat ușor este scanarea Graham.

Înainte de a descrie acest algoritm vom introduce două noțiuni utile: «unghi polar» și «unghi care realizează o întoarcere».

1. Unghi polar

Fie M_1 și M_2 două puncte distincte. Pentru punctul M_1 , unghiul polar asociat lui M_2 este unghiul format de dreapta orizontală care trece prin M_1 și dreapta M_1M_2 .

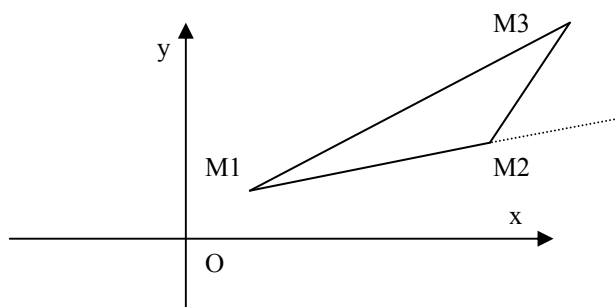


Dacă se cunosc coordonatele punctelor M_1 și M_2 se poate determina ușor distanța de la M_1 la M_2 și distanța de la M_2 la orizontala care trece prin M_1 . Folosind aceste două distanțe (M_1M_2 și M_1M_3) se poate calcula cosinusul unghiului făcut de M_1M_2 cu axa Ox și implicit unghiul făcut de M_1M_2 cu Ox (folosind funcția arccos).

2. Unghi care realizează o întoarcere

Considerăm trei puncte $M_1(x_1, y_1)$, $M_2(x_2, y_2)$, $M_3(x_3, y_3)$.

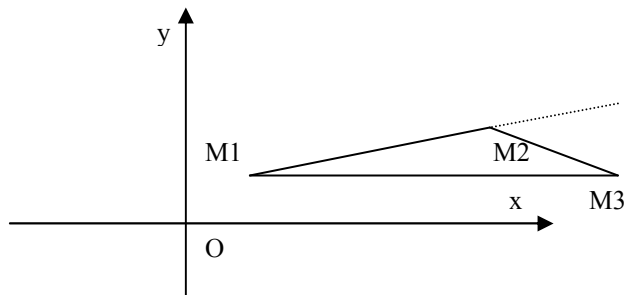
Spunem că unghiul $M_2M_1M_3$ realizează o întoarcere spre stânga (a lui M_3) față de M_2M_1 , dacă ne aflăm în situația



Această situație este caracterizată de inegalitatea

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} > 0 \Leftrightarrow x_1 \cdot y_2 + x_2 \cdot y_3 + x_3 \cdot y_1 - x_1 \cdot y_3 - x_2 \cdot y_1 - x_3 \cdot y_2 > 0$$

Spunem că unghiul $M_2M_1M_3$ realizează o întoarcere spre dreapta (a lui M_3), față de M_2M_1 , dacă ne aflăm în situația



Această situație este caracterizată de inegalitatea

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} < 0 \Leftrightarrow x_1 \cdot y_2 + x_2 \cdot y_3 + x_3 \cdot y_1 - x_1 \cdot y_3 - x_2 \cdot y_1 - x_3 \cdot y_2 < 0$$

Scanarea Graham rezolvă problema înfășurătorii convexe prin păstrarea unei stive care conține puncte. Fiecare vârf dat este inserat în stivă o singură dată, iar punctele care nu fac parte din înfășurătoare sunt eliminate pe parcurs. Astfel în final stiva va conține vârfurile înfășurătorii convexe în ordine trigonometrică.

Descriere algoritm

Pas 1. Se determină vârful A0 care are ordonata cea mai mică și în cazul în care există mai multe astfel de puncte se alege cel care are abscisa cea mai mică.

Pas 2. Se sortează punctele mulțimii $\{A_1, A_2, \dots, A_n\} \setminus \{A_0\}$ crescător după unghiurile polare în jurul vârfului A0. Dacă există mai multe vârfuri cu același unghi polar, atunci se păstrează doar cel care află la distanța maximă de A0. După această operație se obțin punctele A1, A2, ..., Am.

Pas 3. Inserăm într-o stivă (inițial vidă) punctele A0, A1, A2.

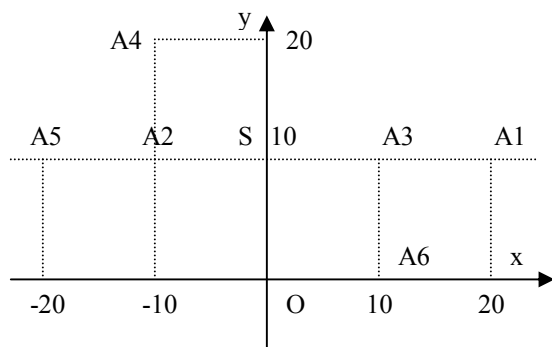
Pas 4. Vom considera pe rând nodurile $A_i, i \in \{3, 4, \dots, m\}$. Vom nota cu A_i , nodul curent, cu A_{i-1} nodul care face parte din vârful stivei și cu A_{i-2} nodul următor din stivă. Pentru fiecare nod A_i vom verifica dacă unghiul format de punctele A_{i-2}, A_{i-1}, A_i nu formează un unghi care să anuleze convexitatea poligonului. Practic acest unghi trebuie să realizeze o întoarcere spre stânga. În cazul în care nu se păstrează convexitatea, vârful A_{i-1} se elimină din stivă. Verificarea continuă (cu eliminări succesive dacă este cazul) până în momentul în care unghiul format nu indică pierderea convexității. În final stiva va conține doar vârfurile înfășurătorii convexe, în ordine trigonometrică.

3. Centrul de simetrie al unei mulțimi de puncte

Fie $A = \{A_1, A_2, \dots, A_n\}$ o mulțime de n puncte în plan.

Un punct S din plan se numește *centru de simetrie* dacă pentru orice punct A_i din mulțimea A, simetricul său față de S este tot un punct din mulțimea A.

Exemplu. Pentru punctele din figură, $S(0,10)$ este centru de simetrie.



În continuare vom rearanja punctele din A astfel încât să fie ordonate crescător după abscisă și la abscisă egală ordonate crescător după ordonată.

Proprietate

Mulțimea A are centru de simetrie dacă și numai dacă:

- n este par
- Segmentele $[A1An]$, $[A2An-1]$, $[A3An-2]$, ... au același mijloc. Acest punct este centrul de simetrie.

4. Poziția unui punct față de un poligon

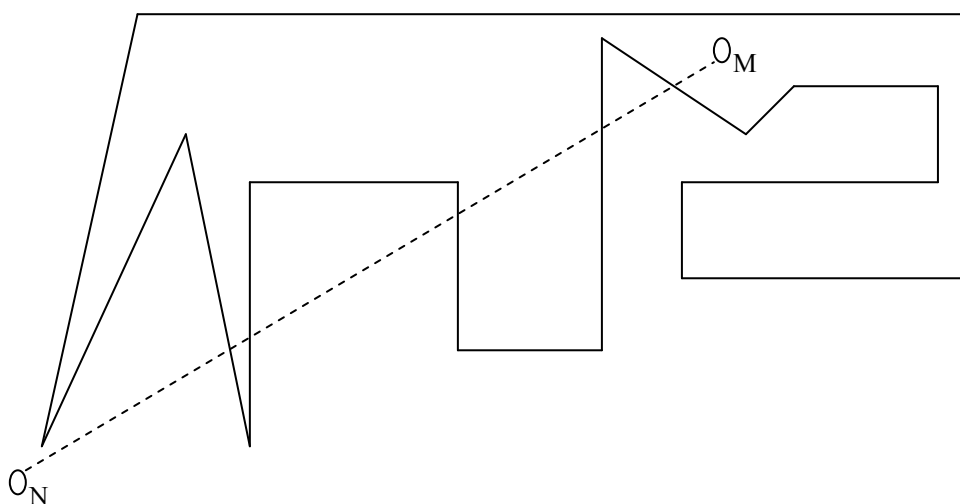
Prin poligon înțelegem o linie frântă care nu se autointersectează.

Considerăm în continuare un poligon $P=A1A2\dots An$ și un punct M .

$A1(x1,y1)$, ..., $An(xn,yn)$ și $M(x0,y0)$.

Dorim să determinăm poziția lui M față de poligonul P . Adică să verificăm dacă M este interior lui P , exterior lui P sau pe poligonul P .

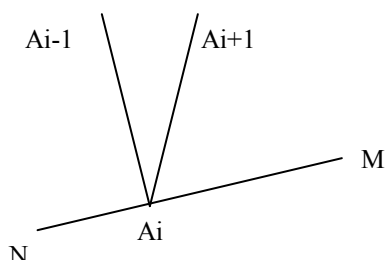
Pentru a verifica dacă M este interior lui P , folosim o observație imediată, și anume faptul că, dacă intersecția unui segment cu un capăt în M și un capăt într-un punct N exterior lui P , cu laturile poligonului este formată dintr-un număr impar de puncte, atunci M este interior lui P .



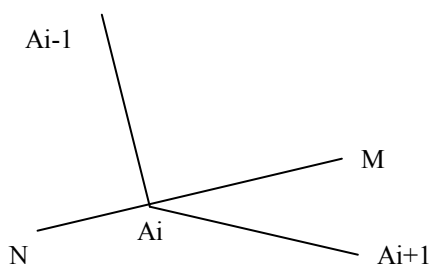
Există însă câteva situații când punctele de intersecție dintre segmentul (MN) și laturile lui P nu se numără. Aceste situații le prezentăm în continuare.

- Dacă dreapta MN coincide cu dreapta suport a unei laturi din P, atunci intersecția dintre (MN) cu latura respectivă conține o infinitate de puncte, care evident nu se numără.

- Dacă segmentul (MN) intersectează o latură ca în cazul următor



aceasta nu se numără. Însă intersecția din cazul următor se numără



Un algoritm care determină poziția unui punct față de un poligon poate fi următorul:

Pas 1. Determinăm un punct exterior lui P astfel: Dintre vârfurile poligonului P determinăm unul A_i , cu proprietatea că are abscisa minimă și în cazul în care există mai multe astfel de puncte se ia cel cu ordonata minimă. Considerăm punctul exterior $N(x_{i-1}, y_i)$.

Pas 2. Pentru orice segment $(A_i A_{i+1})$, dacă $(A_i A_{i+1})$ intersectat cu (MN) ne dă un punct, atunci mărim numărul de puncte de intersecție cu 1.

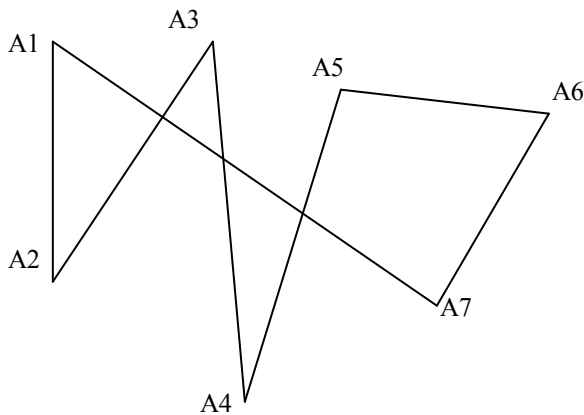
Pas 3. Pentru orice vârf A_i , al poligonului, dacă A_{i-1} și A_{i+1} sunt de o parte și de alta a dreptei MN, atunci numărul de puncte de intersecție se mărește cu 1.

Pas 4. Dacă numărul de puncte de intersecție calculat la pașii anteriori este impar, atunci M este interior lui P. Altfel, verificăm dacă M se găsește pe poligon (verificând dacă M se găsește pe fiecare latură a lui P). Dacă nu se găsește pe poligonul P, atunci M este exterior lui P.

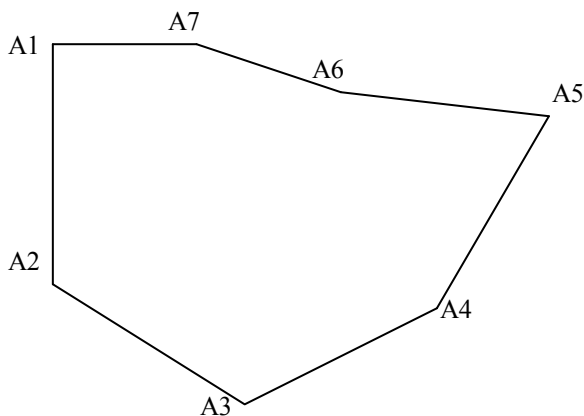
5. Determinarea unui poligon care trece printr-o mulțime de puncte

Considerăm în continuare o mulțime de puncte în plan $A = \{A_1, A_2, \dots, A_n\}$.
Vrem să renumerotăm punctele din mulțime astfel încât $A_1 A_2 \dots A_n$ să fie poligon.

Exemplu



Dacă considerăm punctele în ordinea de mai sus, $A_1 A_2 \dots A_7$ nu este poligon, însă după renumerotarea de mai jos, $A_1 A_2 \dots A_7$ este poligon.

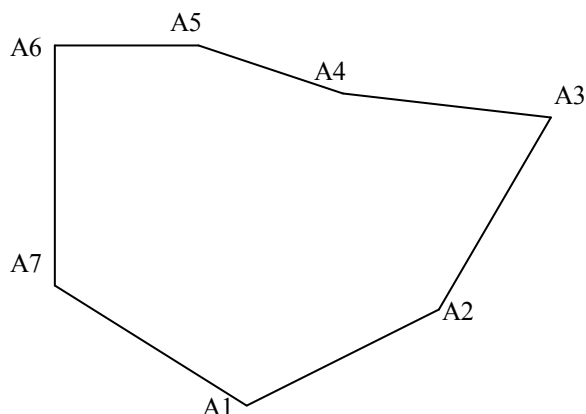


O modalitate de permutare a punctelor mulțimii A , pentru a obține un poligon este inspirată de scanarea Graham. Mai precis:

Pas 1. Se determină vârful A_i care are ordonata cea mai mică și în cazul în care există mai multe astfel de puncte se alege cel care are abscisa cea mai mică. Interschimbăm A_i cu A_1 .

Pas 2. Se sortează punctele mulțimii $\{A_2, \dots, A_n\}$ crescător după unghiurile polare în jurul vârfului A_1 . Dacă există mai multe vârfuri cu același unghi polar, atunci acestea se ordonează descrescător după distanța de la A_1 . După această operație se obțin vârfurile poligonului dorit, în ordinea A_1, A_2, \dots, A_n .

Pentru exemplul anterior se obține ordinea:

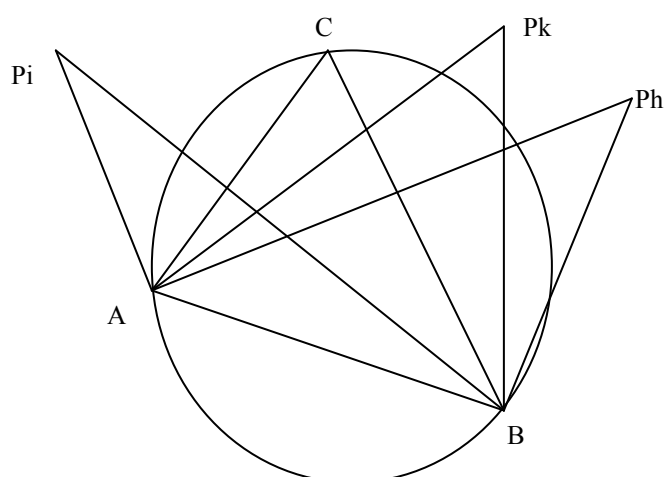


6. Poziționarea unui cerc într-o mulțime de puncte

Considerăm n puncte în plan astfel încât oricare trei să fie necoliniare, date prin coordonatele lor carteziane $P_1(x_1, y_1)$; $P_2(x_2, y_2)$; ...; $P_n(x_n, y_n)$, numere întregi. Vrem să determinăm un cerc (centrul și raza sa) care să treacă prin cel puțin trei din cele n puncte și să nu conțină nici un punct în interior.

Putem găsi acest cerc astfel:

1. Determinăm două puncte A și B printre cele n puncte date cu proprietatea că toate celelalte $n-2$ puncte se află într-un semiplan delimitat de dreapta AB .
2. Se determină punctul C printre punctele diferite de A și B astfel încât unghiul $\angle ACB$ să fie maxim.
3. A, B, C sunt punctele căutate, pentru care se determină cercul circumscris cu centrul în punctul de coordonate (a, b) și rază r .



Pe aceeași idee putem rezolva problema găsirii unui cerc care să treacă prin trei puncte și care să le conțină pe toate celelalte în interior. Mai precis în algoritmul anterior se consideră la pasul 2 în loc de unghiul cel mai mare, unghiul cel mai mic.

O altă problemă legată de găsirea unui cerc cu o anumită proprietate relativă la o mulțime de puncte este următoarea:

Se dau $2n+3$ puncte în plan astfel încât oricare trei să fie necoliniare și oricare 4 puncte nu se află pe același cerc, date prin coordonatele lor carteziane $P1(x_1,y_1)$; $P2(x_2,y_2)$; ...; $Pn(x_n,y_n)$, numere întregi. Se cere să se determine un cerc (centrul și raza sa) care să treacă prin trei din cele $2n+3$ puncte, să conțină în interior n puncte, iar în exterior n puncte.

Ideea de rezolvare a acestei probleme este dată de următorii pași:

1. Printre cele $2n+3$ puncte se caută două puncte B și C cu proprietatea că toate celelalte puncte (notate cu $A_1, A_2, \dots, A_{2n+1}$) sunt în același semiplan.
2. Ordonăm punctele $A_1, A_2, \dots, A_{2n+1}$ astfel încât $m(\angle BA_1C) < m(\angle BA_2C) < \dots < m(\angle BA_{2n+1}C)$.
3. Cercul care trece prin punctele A_{n+1}, B, C va lăsa punctele A_1, \dots, A_n în afară și pe A_{n+2}, \dots, A_{2n+1} în interiorul său.
4. Pentru cercul de la pasul anterior se determină coordonatele centrului și raza sa.

Probleme propuse în concursuri

1.

Agentul **007** are de distrus o tabără de teroriști. Tabăra de teroriști este formată din mai multe obiective (depozite de muniție, pavilioane pentru teroriști, etc.), considerate punctiforme în plan. Agentul **007** primește de la serviciul de informații o hartă cu n obiective din tabăra teroriștilor, date prin coordonatele carteziane. Pe lângă hartă, agentul **007** mai primește și o armă specială (construită pentru această misiune). Arma primită are două țevi și permite tragerea simultană pe aceeași direcție (rectilinie), dar în sens invers a două rachete cu aceeași viteză. După ce se trage cu arma, odată cu atingerea unei ținte explodează și cealaltă rachetă (chiar dacă aceasta din urmă nu și-a atins ținta).

Cerință

Agentul **007** vrea să distrugă tabăra cât mai repede și cu cât mai puține rachete, pentru acest lucru el studiază posibilitatea să se așeze **într-un punct** din tabără (diferit de obiective) care să permită trageri eficiente, adică la fiecare tragere să distrugă câte două obiective simultan. Determinați dacă este posibil să se găsească un astfel de punct.

Date de intrare

În fișierul **a007.in** pe prima linie se află numărul de teste k , după care urmează date pentru fiecare test. Pentru fiecare test pe o linie se află n , iar pe următoarele n linii sunt coordonatele obiectivelor din tabăra teroriștilor (separate printr-un spațiu în ordinea abscisă ordonată).

Date de ieșire

În fișierul **a007.out** se vor scrie k linii, pe fiecare linie se va scrie **1**, dacă există soluție și **0** dacă nu există soluție. În cazul în care există soluție se va scrie în continuare pe aceeași linie separate, printr-un spațiu coordonatele punctului cerut (numere reale trunchiate la 4 zecimale, în ordinea abscisă ordonată).

Restricții

$$0 \leq n \leq 10000$$

$$1 \leq k \leq 3$$

coordonatele punctelor sunt întregi din intervalul $[-10000, 10000]$

Observație

Un obiectiv este distrus dacă racheta explodează exact în punctul corespunzător lui.

Exemplu

a007.in	a007.out
2	1 5.0000 5.0000
4	0
10 0	
10 10	
0 10	
0 0	
6	
0 0	
10 0	
2 10	
12 0	
5 0	
7 0	

(ONI, 2004)

2.

În regatul *XLand* orașele erau înconjurate de ziduri în formă de poligoane convexe. Împăratul a dispus construirea unui drum de legătură directă între capitală și un alt oraș dat. Fiecare extremitate a drumului poate fi orice punct situat pe zidul orașului, respectiv capitalei. Lungimea drumului este distanța dintre extremitățile sale.

Cerință

Determinați cel mai scurt drum dintre capitală și orașul dat.

Date de intrare

Fișier de intrare: **DRUM.IN**

Linia 1: K_1

- număr natural nenul, reprezentând numărul de colțuri ale zidurilor capitalei;

Linia 2: $x_1 y_1 x_2 y_2 \dots x_{K_1} y_{K_1}$

- K_1 perechi de numere întregi, separate prin câte un spațiu, reprezentând coordonatele vârfurilor zidurilor capitalei;

Linia 3: K_2

- număr natural nenul, reprezentând numărul de colțuri ale zidurilor orașului dat;

Linia 4: $x_1 y_1 x_2 y_2 \dots x_{K_2} y_{K_2}$

- K_2 perechi de numere întregi, separate prin câte un spațiu, reprezentând coordonatele vârfurilor zidurilor acestui oraș.

Date de ieșire

Fișier de ieșire: **DRUM.OUT**

Linia 1: $x_1 y_1 x_2 y_2$

- patru numere reale trunchiate la 4 zecimale, separate prin câte un spațiu, reprezentând extremitățile drumului de legătură respectiv.

Restricții

- $2 \leq K_1, K_2 \leq 20$
- Coordonatele vârfurilor zidurilor ce înconjoară orașul, respectiv capitala sunt numere întregi aparținând intervalului $[-100, 100]$ și sunt date fie în ordinea deplasării acelor de ceasornic, fie în sens invers deplasării acelor de ceasornic.
- Capitala și orașul nu au nici un punct comun (nu au puncte interioare comune și nu au puncte comune pe zidurile lor).

Exemplu

DRUM.IN	DRUM.OUT
4	5.0000 3.5000 8.0000 3.5000
3 4 3 2 5 2 5 4	
4	
8 3 8 6 11 6 11 3	

Timp maxim de executare/test: 1 secundă

(ONI, 2001, Bacău)

3.

Într-o zonă agricolă care conține mai multe proprietăți (de formă poligonală – convexe sau concave) s-a desfășurat un concurs de parașutism. La acest concurs au participat N parașutiști codificați prin $1, 2, \dots, N$. Concursul este câștigat de acei parașutiști care aterizează pe una dintre proprietățile cu aria cea mai mare.

Cerință

Scrieți un program care determină parașutiștii care câștigă concursul.

Date de intrare

Fișierul de intrare **concurs.in** conține:

concurs.in	<i>Semnificație</i>
N	N – numărul de parașutiști
a₁ b₁	a_i b_i – abscisa și ordonata punctului în care aterizează parașutistul i
a₂ b₂	
...	
a_N b_N	
m	m – numărul de proprietăți
c₁ x₁₁ y₁₁ x₁₂ y₁₂ ... x_{1c₁} y_{1c₁}	c_i reprezintă numărul de colțuri ale proprietății i ,
c₂ x₂₁ y₂₁ x₂₂ y₂₂ ... x_{2c₂} y_{2c₂}	iar x_{i,j} y_{i,j} reprezintă abscisa, respectiv ordonata unui colț al proprietății i . Colțurile unei proprietăți
...	sunt date într-o ordine astfel încât să se poată parcurge continuu frontiera ei.
c_m x_{m1} y_{m1} x_{m2} y_{m2} ... x_{mc₁} y_{mc₁}	

Date de ieșire

Fișierul de ieșire **concurs.out** conține pe prima linie codurile asociate parașutiștilor care au câștigat concursul în ordine crescătoare (separate prin câte un spațiu). Dacă nu există câștigători se va scrie cifra **0**.

Restricții

- **N** număr natural, $1 \leq N \leq 50$;
- $0 \leq a_i, b_i \leq 100$, numere reale cu maxim 4 cifre zecimale;
- **m** număr natural, $1 \leq m \leq 20$;
- **c_i** numere naturale, $1 \leq c_i \leq 20$;
- **x_{i,j}, y_{i,j}** sunt numere naturale ≤ 100

Observații

- Frontiera unei proprietăți nu face parte din proprietate;
- Proprietățile sunt disjuncte (nu există două proprietăți cu suprafețe de teren comune).
- Două numere reale x și y le considerăm egale dacă $|x - y| < 10^{-4}$.

Exemplu

concurs.in	concurs.out
4	1 4
3 5	
10 4	
7 1	
3.002 6	
2	
4 9 3 11 3 11 5 9 5	
5 5 3 4 5 5 7 3 7 2 5	

Timp maxim de execuție: 8 secunde/test

(Baraj, Bucuresti, 2002)

4.

Se da un poligon convex cu n vârfuri ($50 \geq n \geq 4$) prin coordonatele vârfurilor (numere întregi din intervalul $[-2000, 2000]$) date în sensul acelor de ceasornic.

- a) Sa se verifice daca poligonul are centru de simetrie.
- b) Sa se împarta poligonul (daca are centru de simetrie) în paralelograme.

Datele de intrare se citesc din fi[ierul text **POLIGON.IN** cu structura:

```
n
x1 y1
x2 y2
...
xn yn
```

Datele de iesire se vor introduce în fi[ierul text **POLIGON.OUT** cu structura:

```
mesaj
px11 py11 px21 py21 px31 py31 px41 py41
px12 py12 px22 py22 px32 py32 px42 py42
...
px1k py1k px2k py2k px3k py3k px4k py4k
```

unde **mesaj** poate fi “**DA**” sau “**NU**”, în functie de existenta sau nu a centrului de simetrie. Daca mesajul este “**NU**”, fisierul va contine doar prima linie, altfel va mai contine coordonatele vârfulor (în sensul acelor de ceasornic) paralelogramelor, pe câte un rând fiecare.

Exemple:

Pentru fisierul **POLIGON.IN** cu continutul:

```
8
20 30
30 30
40 20
40 10
30 0
20 0
10 10
10 20
```

Fisierul de iesire **POLIGON.OUT** poate contine spre exemplu solutia:

```
DA
20 0 10 10 20 10 30 0
10 10 10 20 20 20 20 10
10 20 20 30 30 30 20 20
20 10 20 20 30 10 30 0
20 20 30 30 40 20 30 10
40 20 40 10 30 0 30 10
```

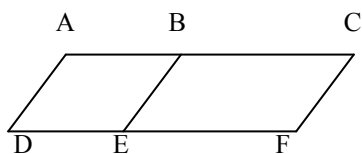
Pentru fisierul **POLIGON.IN** cu continutul:

```
4
10 20
40 30
40 10
20 10
```

Fisierul de iesire **POLIGON.OUT** va contine:

```
NU
```

Observatie: Sirul de paralelograme folosit la împartirea poligonului nu va contine paralelograme în configuratia din figura alaturata, pentru ca paralelogramele ABED si BCFE pot fi înlocuite cu ACFD.



(ONI, 2000)

5.

Generalul Donald are pe câmpul de luptă n soldați. Pentru a ști exact unde sunt dispuși soldații a primit de la un colonel o hartă pe care este precizată poziția lor. Pentru a obține informații noi din tabăra inamicului a trimis o grupă de cercetare în teren. După câteva ore această grupă se întoarce cu o informație prețioasă, inamicul are o armă care poate distruge pe o direcție dreaptă orice obstacol de pe hartă. Având această informație generalul Donald studiază harta și observă că are cam mulți soldați dispuși pe aceeași linie dreaptă.

Cerință

Se cere să se determine numărul maxim de soldați care se găsesc poziționați pe o aceeași linie dreaptă.

Date de intrare

Fișierul de intrare **sol.in** conține pe prima linie numărul n , iar pe următoarele n linii coordonatele carteziene ale fiecărui soldat pe hartă. O astfel de linie are formatul:
abscisă ordonată (separate între ele printr-un singur spațiu).

Date de ieșire

Fișierul de ieșire **sol.out** se va conține numărul maxim de soldați care se găsesc poziționați pe o aceeași linie dreaptă.

Restricții

$$1 \leq n \leq 300$$

Toate coordonatele pozițiilor soldaților sunt numere întregi din intervalul $[-1000,1000]$.

Nu există doi soldații aflați în aceeași poziție.

Exemplu

sol.in	sol.out	Explicație
5	3	Există trei soldați care se găsesc poziționați pe o linie dreaptă.
5 5		
10 2		
18 5		
5 2		
10 5		

Timp maxim de execuție: 0.1 secunde/test

(.campion, 2005-2006)

6.

O companie formată din n soldați (codificați prin numerele $1, 2, \dots, n$) are ca misiune să păzească un obiectiv militar de maximă importanță. Soldații au fiecare la dispoziție un anumit număr de cartușe pe care le poate folosi pentru apărarea obiectivului. Fiecare soldat $i, i \in \{1, 2, \dots, n\}$ se așează într-un anumit loc caracterizat prin coordonate (abscisă și ordonată) și își sapă un adăpost pentru a putea rezista inamicului. Nu se pot așeza doi soldați pe același loc.

Comandantul stabilește zonele de apărare contrice. O zonă de apărare este delimitată de acei soldați care sunt în viață și care dacă sunt uniți prin tranșee (segmente de dreaptă) formează un poligon convex ce conține în interiorul său toți ceilalți soldați aflați în viață. O zonă de apărare conține cel puțin trei soldați și se formează imediat după ce zona anterioară a cedat.

O zonă de apărare cedează când cel puțin un soldat din ea nu mai are cartușe (în acel moment toți soldații de pe zona de apărare respectivă sunt omorâți de inamic).

Se știe că soldații de pe aceeași zonă de apărare trag la un interval de o secundă, în același timp. După ce o zonă de apărare este distrusă, imediat în secunda următoare intră în acțiune zona de apărare următoare.

Cerință

Să se determine după câte secunde va cuceri inamicul obiectivul păzit.

Date de intrare

În fișierul **sold.in** se află pe prima linie n , iar pe următoarele n linii coordonatele poziției soldaților $1, 2, \dots, n$ împreună cu numărul de cartușe pe care le au separate prin câte un spațiu (ordinea pe linie este următoarea: abscisă, ordonată și număr cartușe).

Date de ieșire

În fișierul `sold.out` se va scrie numărul de secunde cerut.

Restricții

n - număr natural nenul < 3200

coordonatele poziției soldaților sunt numere întregi din intervalul $[-32000, 32000]$.

numărul de cartușe pentru fiecare soldat este natural nenul și < 1000000 .

Exemplu

<code>sold.in</code>	<code>sold.out</code>
11	30
5 0 23	
0 2 100	
4 3 20000	
5 4 10	
2 1 301	
0 0 45	
3 4 567	
2 3 342	
2 0 2123	
3 2 90	
4 1 20	

Timp maxim de execuție/test: 1 secundă

(campion, 2004)

7.

Ionică a primit de ziua lui de la tatăl său un joc format din piese de formă de triunghiulară de dimensiuni diferite și o suprafață magnetică pe care acestea pot fi așezate. Pe suprafața magnetică este desenat un triunghi dreptunghic cu lungimile catetelor a , respectiv b și un sistem de coordonate xOy cu originea în unghiul drept al triunghiului, semiaxa $[Ox]$ pe cateta de lungime a , respectiv semiaxa $[Oy]$ pe cateta de lungime b . La un moment dat Ionică așează pe tabla magnetică n piese, pentru care se cunosc coordonatele vârfulor lor. Tatăl lui Ionică vrea să verifice dacă pe tablă piesele realizează o **partiție** a triunghiului dreptunghic desenat, adică dacă sunt îndeplinite condițiile:

nu există piese suprapuse;

piesele acoperă toată porțiunea desenată (în formă de triunghi dreptunghic);

nu există porțiuni din piese în afara triunghiului desenat.

Cerință

Se cere să se verifice dacă piesele plasate pe tabla magnetică formează o **partiție** a triunghiului desenat pe tabla magnetică.

Date de intrare

Fișierul de intrare `part.in` conține pe prima linie un număr natural k , reprezentând numărul de seturi de date din fișier. Urmează k grupe de linii, câte o grupă pentru fiecare set de date. Grupa de linii corespunzătoare unui set este formată dintr-o linie cu numerele a , b , n separate între ele prin câte un spațiu și n linii cu câte șase numere întregi separate prin spații reprezentând coordonatele vârfulor (abscisă ordonată) celor n piese, câte o piesă pe o linie.

Date de ieșire

În fișierul `part.out` se vor scrie k linii, câte o linie pentru fiecare set de date. Pe linia i ($i=1, 2, \dots, k$) se va scrie 1 dacă triunghiurile din setul de date i formează o **partiție** a triunghiului desenat pe tabla magnetică sau 0 în caz contrar.

Restricții

$1 \leq n \leq 150$

$1 \leq k \leq 10$

a, b sunt numere întregi din intervalul $[0, 31000]$

Coordonatele vârfulor pieselor sunt numere întregi din intervalul $[0, 31000]$.

Exemplu

part.in	part.out
2	1
20 10 4	0
0 5 0 10 10 5	
0 0 10 5 0 5	
0 0 10 0 10 5	
10 0 20 0 10 5	
20 10 2	
0 0 0 10 10 5	
0 0 20 0 20 10	

Timp maxim de execuție: 0.3 secunde/test

(Baraj, 2006)

8.

Într-un document arheologic recent descoperit se face referire la un mare tezaur. Datorită faptului că documentul poate fi interpretat în mai multe moduri se face apel la n arheologi care vor studia independent documentul.

La terminarea studiului fiecare arheolog întocmește o hartă pe care marchează o zonă poligonală închisă și convexă ce este considerată a conține tezaurul.

Pentru că fondurile alocate pentru descoperirea tezaurului sunt reduse se ia hotararea să se înceapă cercetările pe teren doar în zona de pe hartă precizată de toți arheologii.

Cunoscând n și coordonatele vârfurilor zonelor determinate de arheologi se cere să se determine suprafața (aria) zonei de pe hartă precizată de toți arheologii (intersecția celor n zone

Date de intrare:

Fisierul de intrare **TEZAUR.IN** conține:

n

$m[1]$

-numărul de vârfuri pentru zona primului arheolog

$x_{11} y_{11} \dots x_1 y_{1m[1]}$

-coordonatele varfurilor zonei primului arheolog

...

(date în sens direct sau invers acelor de ceasornic)

$m[n]$

-numarul de varfuri pentru zona ultimului arheolog

$x_{1m[n]} y_{1m[n]} \dots x_{nm[n]} y_{nm[n]}$

-coordonatele varfurilor zonei celui de-al n -lea arheolog

(date în sens direct sau invers acelor de ceasornic)

Date de ieșire:

Ieșirea se va face în fișierul **TEZAUR.OUT** ce va conține pe prima linie suprafața (aria) zonei de pe hartă precizată de toți arheologii.

Observație: Dacă nu există suprafață comună zonelor celor n arheologi se va scrie în fișierul **TEZAUR.OUT** valoarea 0.

Exemplu:

TEZAUR . IN

TEZAUR . OUT

3

300.0000

4

-20 30 40 30 40 -20 -20 -20

5

10 -30 60 -50 100 60 70 80 10 80

5

40 0 70 0 70 60 20 60 20 30

Restricții:

- $1 \leq n \leq 30$;
- $3 \leq m_i \leq 20, i \in \{1, 2, \dots, n\}$;
- Coordonatele vârfurilor zonelor sunt numere întregi din intervalul $[-10000, 10000]$;
- Numărul din fișierul de ieșire se va scrie cu 4 zecimale.

Timp maxim de execuție pe test: 1 secundă

(ONI, Constanța, 2000)

9.

Se dau N pătrate situate în plan, având laturile paralele cu axele de coordonate. Coordonatele vârfurilor sunt numere întregi. Pătratele nu se ating și nu se suprapun.

Se cere să se determine numărul de pătrate vizibile din originea O , $O=(0,0)$.

Un pătrat este *vizibil* din originea O , dacă există două puncte distincte A și B de coordonate întregi, situate pe o aceeași latură a pătratului, astfel încât interiorul triunghiului OAB nu are puncte comune cu nici unul dintre celelalte pătrate.

Date de intrare

Prima linie a fișierului de intrare `SQUARES.IN` conține un număr întreg N , $1 \leq N \leq 1000$, reprezentând numărul de pătrate.

Fiecare din următoarele N linii descrie un pătrat prin numerele întregi X, Y și L separate prin câte un spațiu $1 \leq X, Y, L \leq 10000$. X și Y sunt coordonatele colțului din stânga jos al pătratului, iar L este lungimea laturii.

Date de ieșire

Prima (și singura) linie a fișierului de ieșire `SQUARES.OUT` trebuie să conțină numărul pătratelor vizibile din origine.

Exemple

<code>SQUARES . IN</code>	<code>SQUARES . OUT</code>
3 2 6 3 1 4 1 3 4 1	3
<code>SQUARES . IN</code>	<code>SQUARES . OUT</code>
4 1 2 1 3 1 1 2 4 2 3 7 1	2

(CEOI, Croația, 1998)

10.

N soldați din țara *Caroiaj* sunt așezați la întâmplare pe teritoriul țării. O poziție în țara *Caroiaj* este dată de coordonatele (x,y) , numere întregi. Soldații pot să-și schimbe poziția astfel: într-o mutare, un soldat poate să se deplaseze cu o unitate în sus, în jos, la stânga sau la dreapta (își schimbă fie coordonata x , fie coordonata y cu 1 sau -1).

Soldații vor să se alinieze pe orizontală, unul lângă altul (astfel încât pozițiile lor finale vor fi (x,y) , $(x+1,y)$, ..., $(x+N-1,y)$, pentru un anumit x și y). Întregii x și y sunt oarecare. Ordinea finală a soldaților pe linie, este de asemenea oarecare.

Să se determine numărul minim de interschimbări de poziție a tuturor soldaților, necesare pentru aliniere.

Doi sau mai mulți soldați nu pot ocupa în același timp aceeași poziție.

Date de intrare

Prima linie a fișierului `SOLDIERS.IN` conține un număr întreg N , $1 \leq N \leq 10000$, reprezentând numărul de soldați.

Următoarele N linii conțin pozițiile inițiale ale soldaților: pentru orice i , $1 \leq i \leq N$, linia $i+1$ conține o pereche de numere întregi $x[i]$ și $y[i]$ separate printr-un singur spațiu, reprezentând coordonatele pătratului i , $-10000 \leq x[i], y[i] \leq 10000$.

Date de ieșire

Prima și singura linie a fișierului de ieșire `SOLDIERS.OUT` va conține numărul minim total de schimbări de poziție a soldaților, necesar pentru aliniere.

Exemple

SOLDIERS . IN	SOLDIERS . OUT
3	4
1 0	
2 4	
3 2	
SOLDIERS . IN	SOLDIERS . OUT
5	8
1 2	
2 2	
1 3	
3 -2	
3 3	

(CEOI, Croația, 1998)

Bibliografie

1. Grigore Albeanu, Algoritmi geometrici elementari, Gazeta de Informatică, 4-5, 1998
2. T. Cormen, C.Leiserson, R. Rivest, Introducere în algoritmi, Ed. Agora, 2000
3. Victor Mitrana, Provocarea algoritmilor, Ed. Agni, 1994
4. Mihai Scorțaru, Informatica pentru grupele de excelență, Ed. Dacia, 2004
5. E. Cerchez, M. Șerban, Programarea în limbajul C++, III, Ed. Polirom, pg. 2258.