Algoritmizálás

Horváth Gyula
Szegedi Tudományegyetem
Természettudományi és Informatikai Kar
horvath@inf.u-szeged.hu

9. Geometriai feladatok

Számos olyan gyakorlati számítási probléma van, amely megoldható olyan geometriai modellben, amelyben csak egyszerű objektumok, pontok, egyenesek, szakaszok, szögek, szögtartományok, poligonok szerepelnek. Ilyen alapfeladatokat vizsgálunk.

10. Alapfogalmak

Pont: $(x,y) \in \mathbb{R} \times \mathbb{R}$
Megjegyzés: Csak olyan feladatkat tekintünk, ahol a pontok koordinátái mindig egész számok, és a megoldásához nem kell lebegőpontos aritmetikát használni.
Szakasz
Legyen p_1, p_2 pont. A p_1 és p_2 pont által meghatározott szakasz:
$\overline{p_1 p_2} = \{p = (x,y) : x = \alpha p_1.x + (1-\alpha)p_2.x,\ y = \alpha p_1.y + (1-\alpha)\, p_2.y),\ \alpha \in \mathbb{R},\ 0 \leq \alpha \leq 1\}$
Ha p_1 és p_2 sorrendje számít, akkor irányított szakaszról beszélünk, jele: $\overrightarrow{p_1 p_2}$.
Egyenes
Egyenes megadása:
1. $y = mx + b$ egyenlettel: azon (x,y) pontok halmaza, amelyekre teljesül az egyenlet.
2. $ax + by + c = 0$ egyenlettel.
3. Egyenes megadása két pontjával: $e(p_1, p_2)$

Pontok ábrázolására a

```
1 Type
2   Pont=record x,y:longint;az: longint end;
```

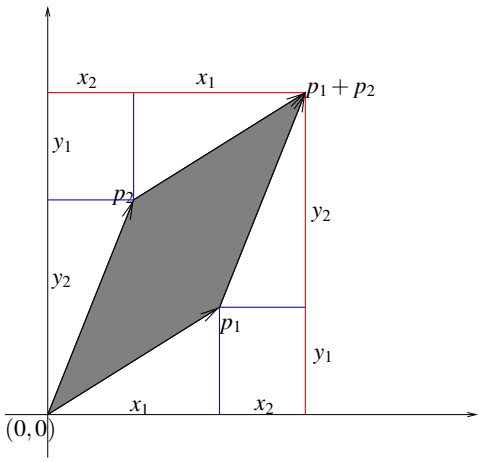
típust használjuk, ahol p.az a pont azonosítója, vagy sorszáma a feladatleírás szerint.

10.1. Forgásirány

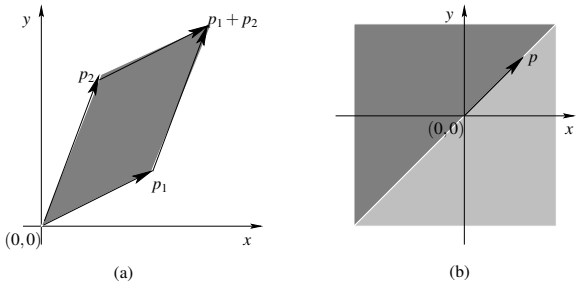
Gyakran előfordul, hogy a síkon adott p_1 és p_2 pontra el kell dönteni, hogy a p_1 ponthoz képest a p_2 pont milyen forgásirányba esik. Tekintsük a 3. ábrán látható p_1 és p_2 vektorokat. A $p_1 \times p_2$ **keresztszorzat** a $(0,0), p_1, p_2$ és $p_1 + p_2 = (p_1.x + p_2.x, p_1.y + p_2.y)$ pontok által alkotott paralelogramma előjeles területeként értelmezhető.

$$\begin{aligned} p_1 \times p_2 &= \det \begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \end{pmatrix} \\ &= x_1 y_2 - x_2 y_1 \\ &= -p_2 \times p_1\ . \end{aligned}$$

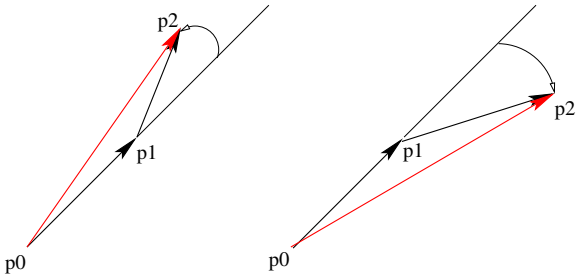
$p_1 \times p_2 > 0 \Leftrightarrow p_2$ az órajárással ellentétes irányban van p_1-hez képest.
$p_1 \times p_2 = 0 \Leftrightarrow$ a $(0,0)$, p_1 és p_2 pontok egy egyenesre esnek (kollineárisak).
$p_1 \times p_2 < 0 \Leftrightarrow p_2$ az órajárás irányban van p_1-hez képest.
Még általánosabban, adott két csatlakozó irányított szakasz, $\overrightarrow{p_0 p_1}$ és $\overrightarrow{p_1 p_2}$. Milyen irányba fordul $\overrightarrow{p_1 p_2}$ a $\overrightarrow{p_0 p_1}$-hez viszonyítva?
A válasz $(p_1 - p_0) \times (p_2 - p_0) = (p_1.x - p_0.x)(p_2.y - p_0.y) - (p_2.x - p_0.x)(p_1.y - p_0.y)$ keresztszorzat előjele alapján megadható.
$(p_1 - p_0) \times (p_2 - p_0) > 0 : \overrightarrow{p_1 p_2}$ balra fordul,



1. ábra. A paralelogramma előjeles területe:
$T = (x_1+x_2)(y_1+y_2) - x_1y_1 - x_2y_2 - x_2y_1 - x_2y_1 =$
$x_1y_1 + x_1y_2 + x_2y_1 + x_2y_2 - x_1y_1 - x_2y_2 - x_2y_1 - x_2y_1 = x_1y_2 - x_2y_1$



2. ábra. (a) A p_1 és p_2 vektorok keresztszorzata a paralelogramma előjeles területe. (b) A világos tartomány azokat a pontokat tartalmazza, amelyek a p-től órajárással egyező irányba esnek, a sötétebb pedig azokat, amelyek órajárással ellentétes irányba esnek p-től.



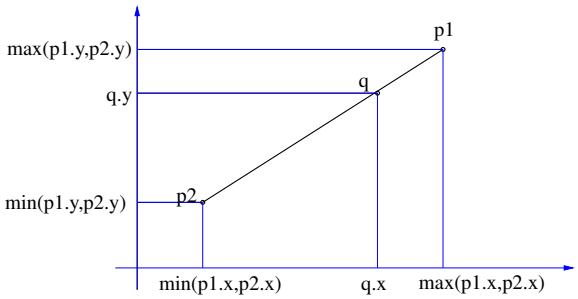
3. ábra. Csatlakozó szakaszok forgásiránya.

$(p_1 - p_0) \times (p_2 - p_0) = 0$: $\overrightarrow{p_0 p_1}$ és $\overrightarrow{p_1 p_2}$ kollineárisak,
$(p_1 - p_0) \times (p_2 - p_0) < 0$: $\overrightarrow{p_1 p_2}$ jobbra fordul.

```
Function ForgasIrany(P0,P1,P2:Pont):Integer;
 {Kimenet: +1 ha P1–P2 balra fordul,
            0 ha P0,P1 és P2 kollineárisak,
           –1 ha P1–P2 jobbra fordul.}
Var
  KeresztSzorz:Longint;
Begin{ForgasIrany}
  KeresztSzorz:=Int64(P1.x–P0.x)Λ(P2.y–P0.y)–Int64(P2.x–P0.x)Λ(P1.y–P0.y);
  If KeresztSzorz < 0 Then
    ForgasIrany:=–1
  Else If KeresztSzorz>0 Then
    ForgasIrany:=1
  Else
    ForgasIrany:=0;
End{ForgasIrany};
```

10.2. Szakaszon

Annak eldöntése, hogy a P_1 és P_2 pontok által megadott szakaszon van-e a Q pont.



4. ábra. A q pont a $\overline{p_1 p_2}$ szakaszra esik, ha p_1, p_2 és q kollineárisak és $min(p_1.x, p_2.x) \leq q.x \leq max(p_1.x, p_2.x)$ és $min(p_1.y, p_2.y) \leq q.y \leq max(p_1.y, p_2.y)$.

```
Function Szakaszon(P1,P2,Q:Pont):Boolean;
 {Kimenet: akkor és csak akkor Igaz, ha a Q pont a P1–P2 szakaszon van.}
Begin{Szakaszon}
  Szakaszon:= ((P1.x–Q.x)Λ(P2.y–Q.y)–(P2.x–Q.x)Λ(P1.y–Q.y)=0) And
              (Abs(P1.x–Q.x)<=Abs(P2.x–P1.x))                 And
              (Abs(P2.x–Q.x)<=Abs(P2.x–P1.x))                 And
              (Abs(P1.y–Q.y)<=Abs(P2.y–P1.y))                 And
              (Abs(P2.y–Q.y)<=Abs(P2.y–P1.y))
End{Szakaszon};
```

10.3. Közel

Annak eldöntése, hogy a kollineáris P_0, P_1, P_2 pontok esetén P_1 közelebb van-e P_0-hoz, mint P_2:

```
Function Kozel(P0, P1, P2:Pont):Boolean;
{ Feltétel: (P0,P1,P2) kollineáris
  Kimenet: P1 közelebb van P0–hoz, mint P2 }
Begin
  Kozel:= (Abs(P1.x–P0.x)<Abs(P2.x–P0.x)) or
```

```
6               (Abs(P1.y-P0.y)<Abs(P2.y-P0.y)) ;
7  End{Kozel};
```

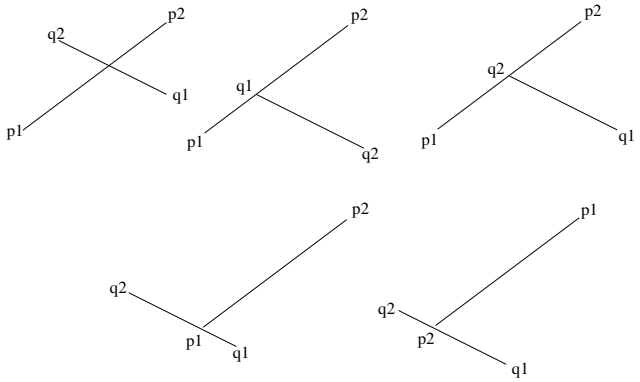
10.4. Közte

Annak eldöntése, hogy a kollineáris P_1, P_2, Q pontok esetén a Q pont a $\overline{P_1P_2}$ szakaszon van-e

```
1  Function Kozte(P1,P2,Q:Pont):Boolean;
2  begin
3   Kozte:=(Abs(P1.x-Q.x)<Abs(P2.x-P1.x)) And
4           (Abs(P2.x-Q.x)<Abs(P2.x-P1.x)) And
5           (Abs(P1.y-Q.y)<Abs(P2.y-P1.y)) And
6           (Abs(P2.y-Q.y)<Abs(P2.y-P1.y))
7  end;
```

10.5. Metsző szakaszpárok

Eldöntendő, hogy metszi-e egymást adott $\overline{p_1p_2}$ és $\overline{q_1q_2}$ szakasz?



5. ábra. Szakaszpárok metszésének öt lehetséges esete.

```
1  Function SzakaszParMetsz(P1,P2, Q1,Q2 : Pont):Boolean;
2    Var fpq1,fpq2,fqp1,fqp2:integer;
3  Begin
4    fpq1:=ForgasIrany(P1,P2, Q1);    fpq2:=ForgasIrany(P1,P2, Q2);
5    fqp1:=ForgasIrany(Q1,Q2, P1);    fqp2:=ForgasIrany(Q1,Q2, P2);
6    SzakaszParMetsz:=(fpq1Λfpq2<0) and (fqp1Λfqp2<0) or
7    Szakaszon(P1,P2, Q1) or Szakaszon(P1,P2, Q2) or
8    Szakaszon(Q1,Q2, P1) or Szakaszon(Q1,Q2, P2);
9  End;
```

11. Feladat: Bekerítés

Adott a síkon a $P = \{p_1, \ldots, p_n\}$ ponthalmaz és egy $q (q \notin P)$ pont. Határozzunk meg három olyan $a, b, c \in P$ pontot, hogy a q pont az $\triangle(a,b,c)$ háromszögbe, vagy oldalára esik, de a P ponthalmaz egyetlen más pontja sem esik a $\triangle(a,b,c)$ háromszögbe, vagy oldalára!

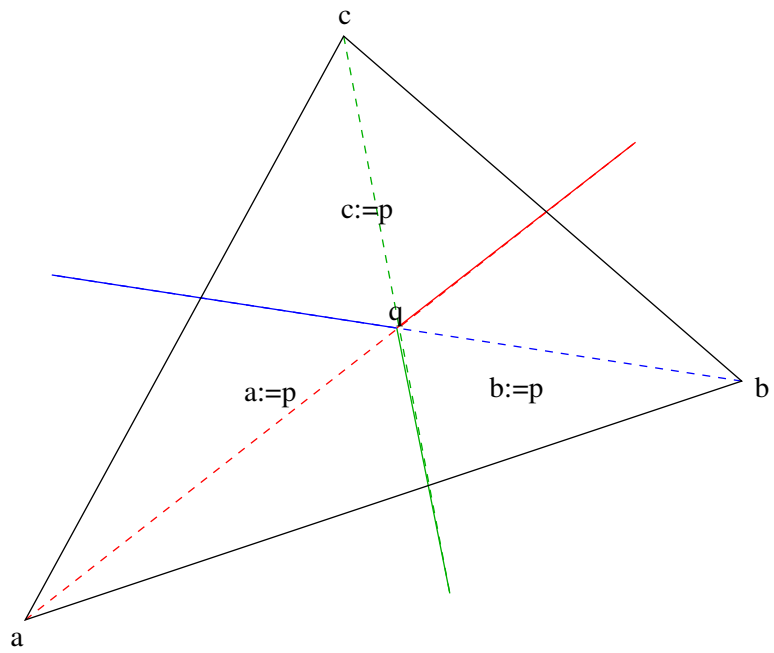
Megoldás.

1. Állítás. Ha van olyan (tetszőleges) $a, b, c \in P$ pont, hogy q az $\triangle(a,b,c)$ háromszögbe, vagy oldalára esik, akkor ez finomítható úgy, hogy a feltétel teljesüljön.

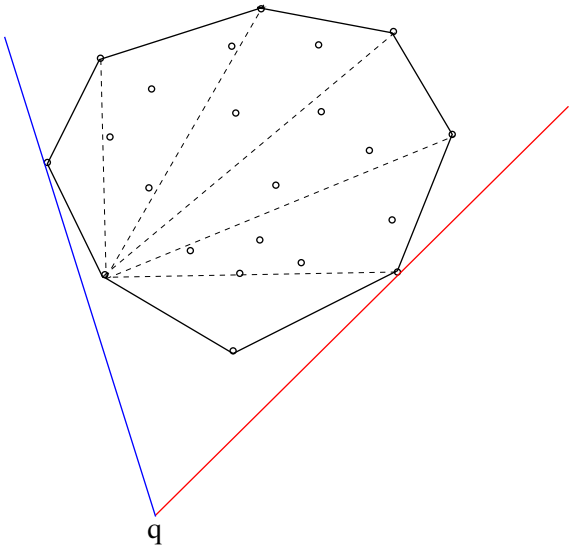
2. Állítás. Akkor és csak akkor van megoldás, ha a q pont a P ponthalmaz konvex burkán belül, vagy oldalán van. A konvex burok előállítása nélkül, gyorsan (lineáris időben) kereshető három olyan $a, b, c \in P$ pont, hogy $q \in \triangle(a,b,c)$. Legyen $a := p_1$, majd keressünk olyan $b \in P$ pontot, hogy a, b és q nem esik egy egyenesre. Ha nincs ilyen b pont, akkor nincs megoldás.

Ezután minden $p \in P$ pontra $(p \neq a, p \neq b)$ határozzul meg, hogy a (q,a) és (q,b) egyenesek által meghatározott síknegyedek melyikébe esik p.

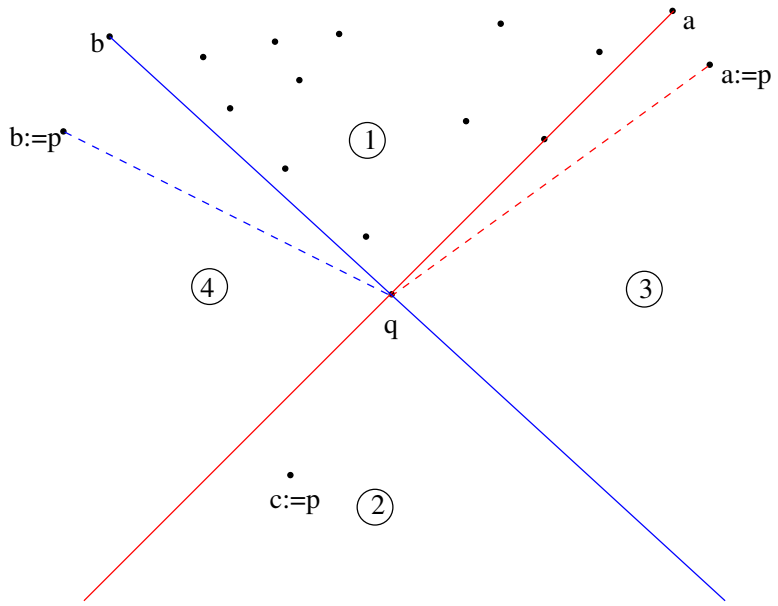
Ha nem a 2. esettel ért véget a keresés, akkor nincs megoldás, mert minden pont $\overrightarrow{qa}$-tól balra és $\overrightarrow{qb}$-től jobbra van.



6. ábra. A háromszög finomítása. Minden $p \in P$ pontra, amely az $\triangle(a,b,c)$ háromszögbe, vagy oldalára esik, $q \in \triangle(a,b,p)$, vagy $q \in \triangle(b,c,p)$ vagy $q \in \triangle(c,a,p)$ teljesül.



7. ábra.



8. ábra. Az eddig vizsgált pontok a $\triangleleft(b,q,a)$ tartományba esnek. Újabb p pont esetén:
1. eset. $ForgasIrany(q,a,p) \geq 0$ és $ForgasIrany(q,b,p) \leq 0$: nem módosul semmi.
2. eset. Egyébként, ha $ForgasIrany(q,a,p) \leq 0$ és $ForgasIrany(q,b,p) \geq 0$: $c := p$ és vége.
3. eset. Egyébként, ha $ForgasIrany(q,a,p) < 0$: $a := p$.
4. eset. Egyébként, (ha $ForgasIrany(q,b,p) > 0$): $b := p$.

```
1  program Kerit3;{ }
2  uses Geom;
3  procedure Kerito(const P: Ponthalmaz; N:longint; q:Pont; var A,B,C:longint);
4  {  Bemenet: P ponthalmaz, q pont
5   Kimenet:
6   (i,0,0) ha P minden pontja az e(q,P[i]) egyenesre esik
7   (a,b,0) ha q a P konvex burkán kívül van, és ekkor
8           az e(q,P[a]) és e(q,P[b]) a két érintő egyenes
9   (a,b,c) ha q a P konvex burkán belül van, és ekkor
10         (P[a],P[b],P[c]) olyan háromszög, hogy q a háromszögben vagy oldalán van,
11         és P egyetlen pontja sem esik a háromszögbe, sem oldalára.
12 }
13 var
14   firqa,firqb,firqc:integer;
15   i,j:longint;
16   aP,bP,cP:Pont;
17 begin
18   aP:=P[1]; i:=2;
19   while (i<=N)and(ForgasIrany(q,aP,P[i])=0) do inc(i);
20   if i>N then begin {pontjai egy egyenesre esnek}
21      A:=1;B:=0;C:=0;
22      exit;
23   end;

24   bP:=P[i]; cP.az:=0;
25   if ForgasIrany(q,aP,bP)<0 then begin{q–aP–tól bP balra essen}
26     aP:=bP; bP:=P[0];
27   end;
28   for j:=1 to N do begin
29     firqa:=ForgasIrany(q,aP,P[j]);
```

```
30      firqb:=ForgasIrany(q,bP,P[j]);
31      if (firqa<0)and(firqb>=0) or (firqa<=0)and(firqb>0) then begin//2.eset
32        cP:=P[j]; break;
33      end else if (firqa<0)and(firqb<0) then begin //3.eset
34        aP:=P[j];
35      end else if (firqa>0)and(firqb>0)then begin //4.eset
36        bP:=P[j];
37      end //else 1. eset, nincs mit tenni, sem aP, sem bP nem vátozik
38    end;
39  {ha cP.az=0, akkor q kivül van, és ekkor e(q,aP) és e(q,bP) érintő egyenes }
40    if (cP.az=0) then begin
41      A:=aP.az; B:=bP.az; C:=0;
42      exit;
43    end;

44  {Az (aP,bP,cP) háromszög tartalmazza a q pontot, finomítás: }
45    for i:=1 to N do
46      if (P[i].az<>aP.az)and(P[i].az<>bP.az)and(P[i].az<>cP.az) and
47          (ForgasIrany(aP,bP,P[i])>=0) and
48          (ForgasIrany(bP,cP,P[i])>=0) and
49          (ForgasIrany(cP,aP,P[i])>=0) then begin
50              firqa:=ForgasIrany(q,aP,P[i]);
51              firqb:=ForgasIrany(q,bP,P[i]);
52              firqc:=ForgasIrany(q,cP,P[i]);
53              if (ForgasIrany(q,aP,bP)=0)and (ForgasIrany(P[i],aP,bP)=0) then begin
54                if  firqc>0 then aP:=P[i] else bP:=P[i]
55              end else if (ForgasIrany(q,bP,cP)=0)and (ForgasIrany(P[i],bP,cP)=0) then begin
56                if  firqa>0 then bP:=P[i] else cP:=P[i]
57              end else if (ForgasIrany(q,cP,aP)=0)and (ForgasIrany(P[i],cP,aP)=0) then begin
58                if  firqb>0 then cP:=P[i] else aP:=P[i]
59              end else
60                if (firqb<=0)and(firqc>=0) then
61                  aP:=P[i]
62                else if (firqa>=0)and(firqc<=0) then
63                  bP:=P[i]
64                else
65                  cP:=P[i];
66      end;
67    A:=aP.az; B:=bP.az; C:=cP.az;
68  end;{Kerinto}
```

12. Feladat: Pont helyzetének megállapítása.

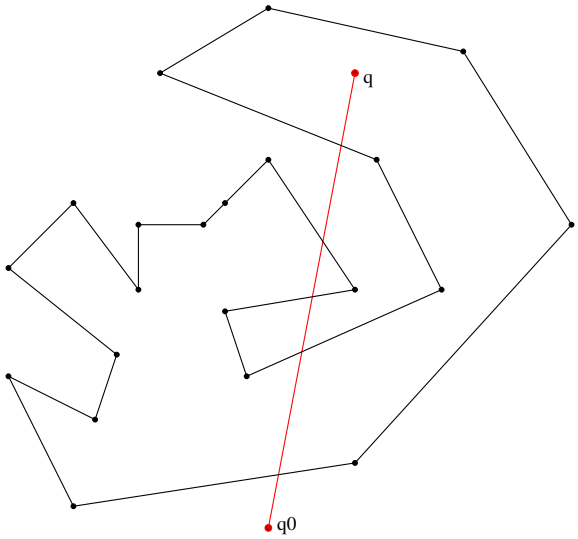
Adott a síkon egy zárt, nem-metsző poligon a csúcsainak $p_1,\ldots,p_n$ órajárással ellentétes felsorolásával és adott egy q pont. Eldöntendő, hogy a q pont a poligon belsejéhez tartozik-e? A poligon valamely oldalára eső pont nem belső pontja a poligonnak.

Megoldás

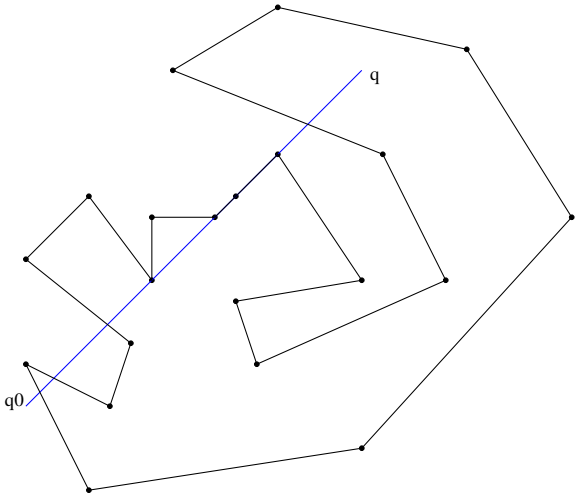
Válasszunk egy olyan q_0 pontot, amely biztosan nem tartozik a poligon belsejéhez és a poligon egyetlen csúcsa sem esik a $\overline{q_0 q}$ szakaszra, legfeljebb ha q megegyezik valamelyik csúccsal. (Van ilyen pont.)

Ha a $\overline{q_0 q}$ szakasz a poligon páratlan számú oldalát metszi, akkor q belső pont, egyébként külső. A q_0 külső pont választása. Legyen $q_0.y = \max\{p_i.y | p_i.y < q.y, i = 1,\ldots n\}$ és $q_0.x = \min\{p_i.x | i = 1,\ldots n\} - 1$.

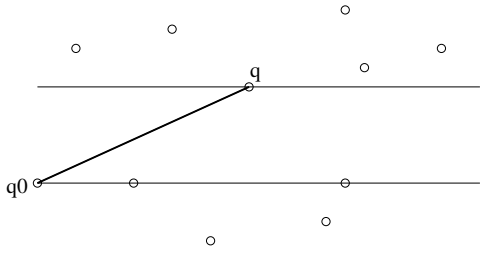
Ha a poligonnak nincs olyan csúcspontja, amelynek y-koordinátája kisebb, mint $q.y$, akkor q biztosan külső pont. A $q0$ pont ilyen választása esetén legfeljebb a q pont esik a poligon valamelyik oldalára. Ezt az esetet külön ellenőrizzük. Egyébként, a poligon bármely $\overline{p_i p_{i+1}}$ oldalának akkor és csak akkor van közös pontja a $\overline{q_0 q}$ szakasszal, ha p_i és p_{i+1} az $e(q_0,q)$ egyenes különböző oldalára esik és a q_0 és q pont az $e(p_i,p_{i+1})$ egyenes különböző oldalára esik. Ez a feltétel a FORGÁSIRANY eljárás felhasználásával egyszerűen kiszámítható:



9. ábra. A $\overline{q_0 q}$ szakasz a poligon páratlan sok oldalát metszi, tehát belső pont.



10. ábra. A poligonnak több csúcsa is a $\overline{q_0 q}$ szakaszra esik.



11. ábra. A q_0 pont választása:$q_0.y = \max\{p_i.y | p_i.y < q.y, i = 1,\ldots n\}$ és $q_0.x = \min\{p_i.x | i = 1,\ldots n\} - 1$.

```
  (ForgasIrany(P[i],P[ii],q0)*ForgasIrany(P[i],P[ii],Q)<0) And
  (ForgasIrany(q0, Q, P[i])*ForgasIrany(q0, Q, P[ii]) <0 )
```

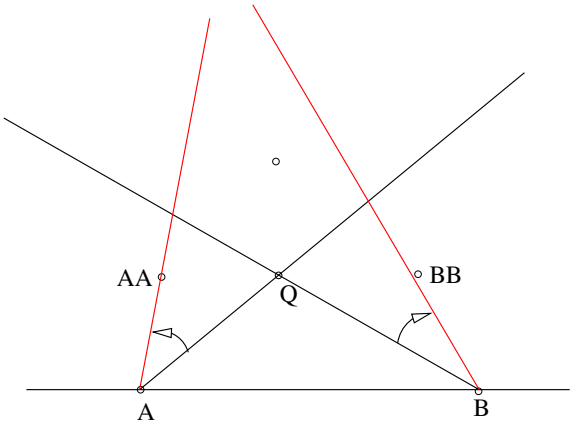
```
Function Ponthelyzete(Var P:PontHalmaz; N:Longint; Q:Pont):integer;
 {Kimenet: -1 ha Q a poligon belsejében van,
           0  ha az oldalán,
           1  ha kivül van }
Var y0,x0:Int64;
  i,ii,metsz:Longint;
  q0:POnt;
Begin
  Ponthelyzete:=0;
  y0:=q.y; x0:=P[0].x;
  For i:=1 To N Do Begin {külső pont választás}
    If (P[i].y<Q.y) And ((P[i].y>y0) Or (y0=Q.y) )  Then
      y0:=P[i].y;
   If (P[i].x<x0) Then x0:=P[i].x;
  End{for i};
  If (y0=Q.y) Then Begin {Q kölső pont}
    Ponthelyzete:=1; exit
  End;
  q0.y:=y0; q0.x:=x0-1; {q0 a külső pont}

  Ponthelyzete:=0; metsz:=0;
  For i:=1 To n Do Begin
    ii:=(i+1)mod n;
    If (Szakaszon(P[i], P[ii], Q) ) Then exit;
    If (ForgasIrany(P[i],P[ii],q0)*ForgasIrany(P[i],P[ii],Q)<0) And
       (ForgasIrany(q0, Q, P[i])*ForgasIrany(q0, Q, P[ii]) <0 ) Then
    Inc(metsz);
  End{for i};
  If Odd(metsz) Then
    Ponthelyzete:=-1
  Else
    Ponthelyzete:=1;
End{ Ponthelyzete};
```

A algoritmus futási ideje legrosszabb esetben is a poligon csúcsainak n számával arányos, tehát $O(n)$.

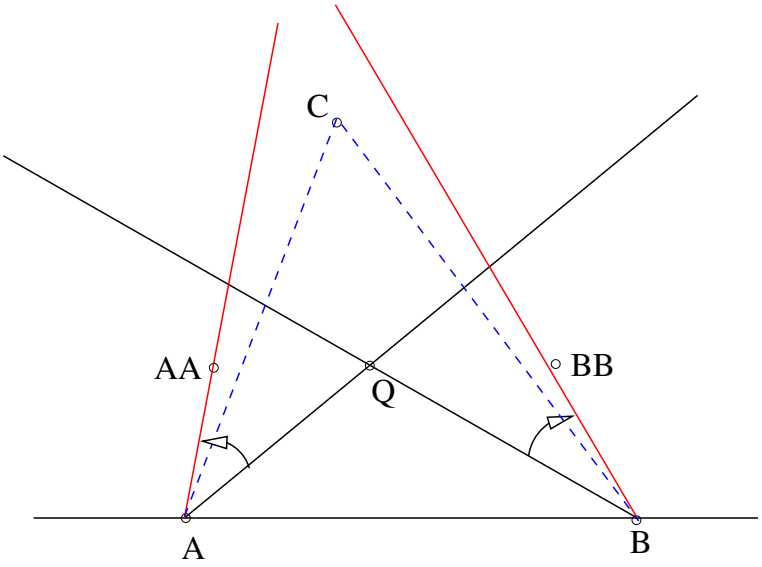
13. Feladat: IOI Válogató 2006

Adott a síkon egy P ponthalmaz és két kitüntetett pontja; A és B. Adott továbbá egy Q pont. Kiszámítandó a P ponthalmaz egy olyan C pontja, hogy a Q pont az $\triangle(A,B,C)$ háromszög belsejében van (nem eshet az oldalára sem), és a P ponthalmaz egyetlen más pontja sem esik a háromszögbe (oldalára sem eshet).
Követelmény: $O(n)$ futási idejű algoritmus.

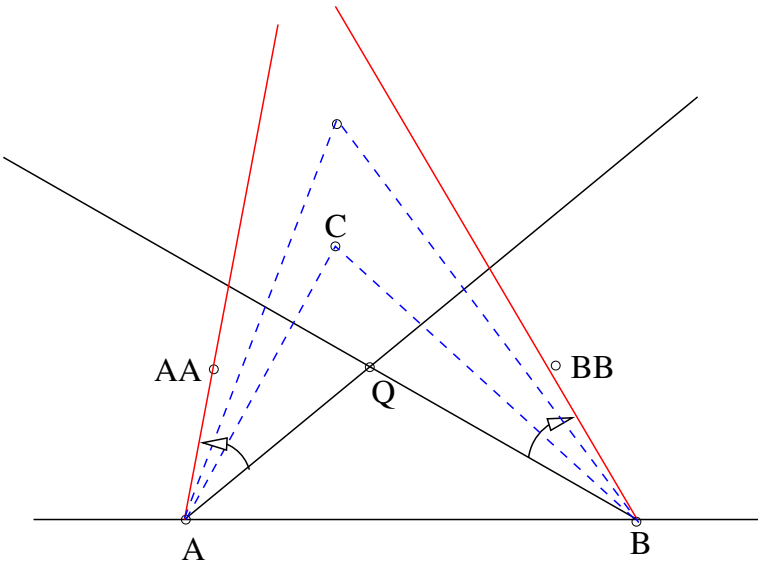
```
Program Harom;
const
  MaxN=100000;
Type
  Pont=record x,y:longint;az: longint end;
var
   n: longint;
   KiF:  text;
   P: array[1..MaxN] of Pont;
   Aaz,Baz,i:longint;
```



12. ábra.



13. ábra.



14. ábra.

```
11      Q,A,B,C,AA,BB:Pont;
12  Function ForgasIrany(P0,P1,P2:Pont):Integer;
13  Var
14    KeresztSzorz:Int64;
15  Begin{ForgasIrany}
16    KeresztSzorz:=Int64(P1.x-P0.x)*(P2.y-P0.y)-Int64(P2.x-P0.x)*(P1.y-P0.y);
17    If KeresztSzorz < 0 Then
18      ForgasIrany:=-1
19    Else If KeresztSzorz>0 Then
20      ForgasIrany:=1
21    Else
22      ForgasIrany:=0;
23  End{ForgasIrany};

24  Procedure Beolvas;
25    var
26        BeF:Text;
27        i,x,y:longint;
28    begin
29      assign(BeF,'harom.be'); reset(beF);
30      readln(Bef,N, x,y, Aaz,Baz);
31      Q.x:=x; Q.y:=y;
32      for i:=1 to N do begin
33        readln(BeF, x,y);
34        P[i].x:=x; P[i].y:=y;
35        P[i].az:=i;
36      end;
37      close(BeF);
38      if ForgasIrany(P[Aaz],P[Baz],Q)>0 then begin
39        A:=P[Aaz]; B:=P[Baz];
40      end else begin
41        B:=P[Aaz]; A:=P[Baz];
42      end;
43  end(*Beolvas*);

44  var
45    FirAB,FirBQ,FirQA,FirAAA,FirBBB,FirAC,FirBC:integer;
46  begin{prog}
47    Beolvas;
48    assign(KiF,'harom.ki'); rewrite(KiF);
49    AA.az:=0; BB.az:=0; C.az:=0;
50    for i:=1 to n do begin (* AA és BB meghatározása *)
51        if (P[i].az=Aaz)or(P[i].az=Baz) then continue;
52        FirAB:= ForgasIrany(A,B,P[i]);
53        FirBQ:= ForgasIrany(B,Q,P[i]);
54        FirQA:= ForgasIrany(Q,A,P[i]);
55        if (FirAB>=0)and(FirBQ>=0)and(FirQA>=0) then begin
56          writeln(Kif,'0');
57          close(Kif);        halt;
58        end;
59        if (FirAB>0)and(FirQA<0)and(FirBQ>0) then begin
60           if (AA.az=0)or(ForgasIrany(A,AA,P[i])<0) then
61             AA:=P[i];
62        end;
63        if (FirAB>0)and(FirQA>0)and(FirBQ<0) then
64           if (BB.az=0)or(ForgasIrany(B,BB,P[i])>0) then
65             BB:=P[i];
66    end;
```

```
67    for i:=1 to n do begin (Λ egy C meghatározása Λ)
68        if (P[i].az=Aaz)or(P[i].az=Baz)or(P[i].az=AA.az)or(P[i].az=BB.az) then continue;
69        if AA.az>0 then FirAAA:= ForgasIrany(A,AA,P[i]);
70        if BB.az>0 then FirBBB:= ForgasIrany(B,BB,P[i]);
71       FirQA:= ForgasIrany(Q,A,P[i]);
72       FirBQ:= ForgasIrany(B,Q,P[i]);
73        if ((AA.az=0)or(FirAAA<0))and((BB.az=0)or(FirBBB>0))and
74            (FirQA<0)and(FirBQ<0) then begin
75          C:=P[i];
76          break;
77       end;
78   end;
79   if C.az=0 then begin
80     writeln(KiF, 0);     halt;
81   end;
82   for i:=C.az+1 to n do begin(Λ C finomítása Λ)
83        if (P[i].az=Aaz)or(P[i].az=Baz)or(P[i].az=AA.az)or(P[i].az=BB.az) then continue;
84       FirAB:= ForgasIrany(A,B,P[i]);        FirAC:= ForgasIrany(A,C,P[i]);
85       FirBC:= ForgasIrany(B,C,P[i]);
86        if (FirAB>0)and(FirAC<0)and(FirBC>0)then    C:=P[i];
87   end;
88   writeln(KiF, C.az);
89   close(KiF);
90 end.
```

14. Feladat: Pontok összekötése zárt, nem-metsző poligonná.

Adott a síkon n darab pont, amelyek nem esnek egy egyenesre. A pontok (x,y) koordinátáikkal adottak, amelyek egész számok. A pontokat a $1,\ldots,n$ számokkal azonosítjuk. Kössünk össze pontpárokat egyenes szakaszokkal úgy, hogy olyan zárt poligont kapjunk, amelyben nincs metsző szakaszpár. Egy ilyen zárt, nem-metsző poligon megadható a pontok azonosítóinak egy felsorolásával: a felsorolásban egymást követő pontokat kötjük össze egyenes szakaszokkal, továbbá, az utolsót az elsővel is összekötjük.

Bemenet

A `poligon.be` szöveges állomány első sora a pontok $n\,(3 < n < 1000)$ számát tartalmazza. A további n sor mindegyike két egész számot tartalmaz, egy pont x és y koordinátáit, $(-20000 \leq x,y \leq 20000)$. A pontok nem esnek egy egyenesre.

Kimenet

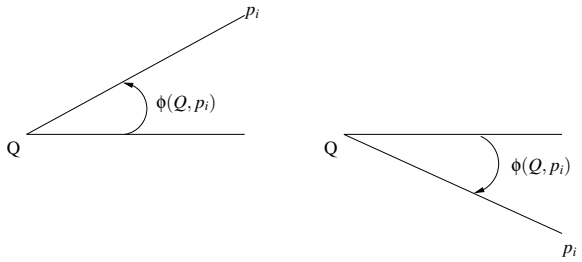
A `poligon.ki` szöveges állományba a bemenetre kiszámított zárt, nem-metsző poligont leíró sorozatot kell kiírni.

Példa bemenet és kimenet

bemenet	**kimenet**
6 2 0 1 4 0 2 3 2 2 4 2 6	3 1 4 5 6 2

Megoldás
Válasszuk ki a legkisebb x-koordinátájú pontot, ha több ilyen van, akkor ezek közül válasszuk a legkisebb y-koordinátájút. Ezt

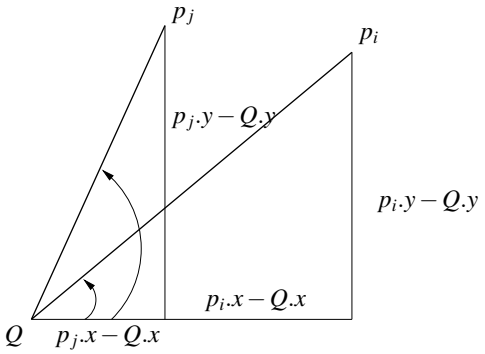
nevezzük (bal-alsó) sarokpontnak és jelöljük Q-val. Húzzunk (fél) egyenest a Q sarokpontból minden ponthoz. Rendezzük az egyeneseket a Q ponton áthaladó, x-tengellyel párhuzamos egyenessel bezárt (előjeles) szög alapján. Rendezzük a pontokat úgy, hogy a Q sarokpont legyen az első, és p_i előbb legyen mint p_j akkor és csak akkor, ha



15. ábra. A p_i ponthoz tartozó előjeles szög: $\phi(Q,p_i)$.

A $\phi(Q,p_i) < \phi(Q,p_j)$, vagy
$\phi(Q,p_i) = \phi(Q,p_j)$ és p_i közelebb van Q-hoz, azaz $p_i.x < p_j.x$, vagy
$\phi(Q,p_i) = \phi(Q,p_j)$ és $p_i.x = p_j.x$ és $p_i.y < p_j.y$.
Ha ebben a sorrendben kötjük össze a pontokat, kivéve, hogy az utolsó egyenesen lévő pontokat fordított sorrendben vesszük, akkor egy zárt, nem metsző poligont kapunk.

Hogyan dönthető el, hogy $\phi(Q,p_i) < \phi(Q,p_j)$?
Minden $i >$ 1-re $-\frac{\pi}{2} < \phi(Q,p_i) \leq \frac{\pi}{2}$ és ebben a tartományban a tangens függvény szigorúan monoton növekvő, tehát



16. ábra. A $\phi(Q,p_i)$ és $\phi(Q,p_j)$ szögek viszonya.

$\phi(Q,p_i) < \phi(Q,p_j)$ akkor és csak akkor, ha

$$\tan(\phi(Q,p_i)) = \frac{p_i.y - Q.y}{p_i.x - Q.x} < \frac{p_j.y - Q.y}{p_j.x - Q.x} = \tan(\phi(Q,p_j))$$

Mivel Q sarokpont, így $p_i.x - Q.x \geq 0$ és $p_j.x - Q.x \geq 0$, tehát

$$\phi(Q,p_i) < \phi(Q,p_j)$$

akkor és csak akkor, ha

$$(p_i.y - Q.y)(p_j.x - Q.x) < (p_j.y - Q.y)(p_i.x - Q.x)$$

Tehát a pontok rendezése: p_i megelőzi p_j-t akkor és csak akkor, ha

$$\begin{aligned}
(p_i.y - Q.y)(p_j.x - Q.x) &< (p_j.y - Q.y)(p_i.x - Q.x) \vee \\
(p_i.y - Q.y)(p_j.x - Q.x) &= (p_j.y - Q.y)(p_i.x - Q.x) \wedge p_i.x < p_j.x \vee \\
(p_i.y - Q.y)(p_j.x - Q.x) &= (p_j.y - Q.y)(p_i.x - Q.x) \wedge p_i.x = p_j.x \wedge p_i.y < p_j.y
\end{aligned}$$

Vegyük észre, hogy a bal-alsó sarokponthoz viszonyított polárszög szerinti rendezés rendezési relációja megadható a FORGASIRANY és a KOZEL (vagy KOZTE) műveletekkel:

```
  forg:=ForgasIrany(sarok,p[i],p[j]);
  (forg>0) Or (forg=0) And Kozel(sarok,p[i],p[j]);
```

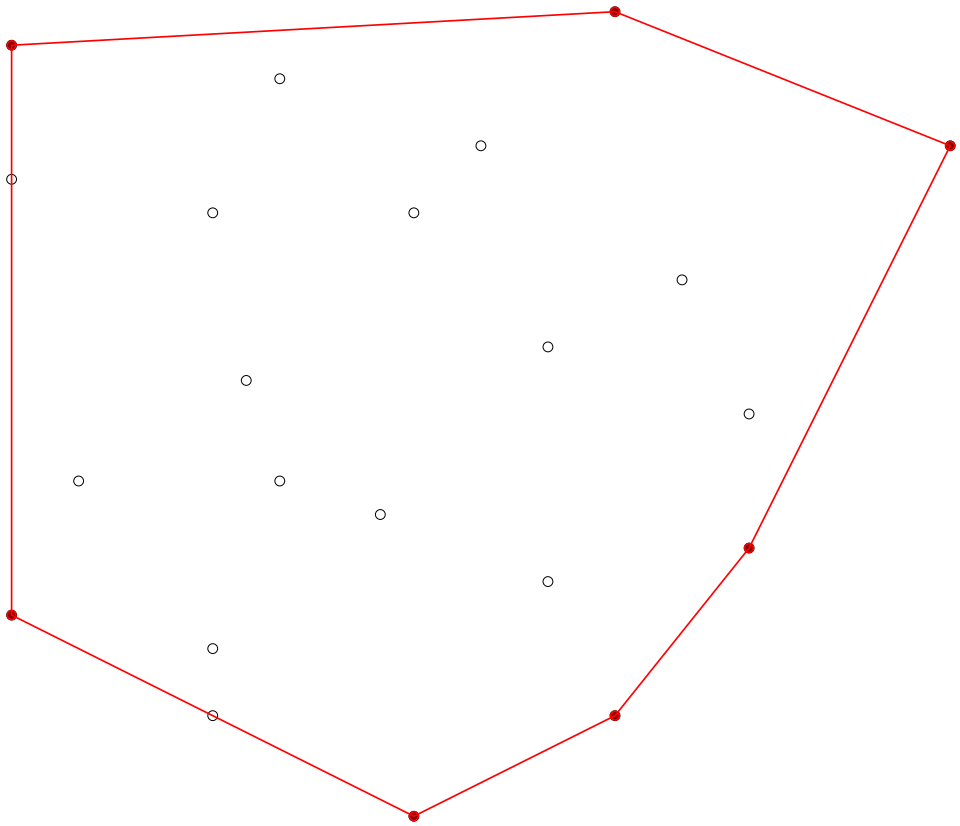
```
Program Poligon;
Const
  MaxN=10000;           {a pontok max. száma}
Type
  Pont=Record
         x,y:Longint;{a pont koordinátái}
         az:Longint; {a pont azonosítója}
       End;
  PontHalmaz=Array[1..MaxN] of Pont;
Var
  N:Longint;            {a pontok száma}
  P:PontHalmaz;         {a bemeneti ponthalmaz}
  P0:Pont;              {sarokpont}
Procedure Beolvas;      {Global: P, N}
  Var InpF:Text;        {bemeneti állomány}
    i:word;
  Begin
    Assign(InpF,'poligon.be'); Reset(InpF);
    ReadLn(InpF,N);
    For i:=1 To N Do Begin
      Read(InpF,P[i].x, P[i].y);  P[i].az:=i; End;
    Close(InpF)
  End {Beolvas};

Procedure KiIr(j:Word);
{Global: P, N}
  Var
    OutF:Text;
    i:Word;
  Begin
   Assign(OutF,'poligon.ki'); Rewrite(OutF);
   For i:=1 To j Do
      Write(OutF, P[i].az:1,'␣');
   For i:=N DownTo j+1 Do
      Write(OutF, P[i].az:1,'␣');
    WriteLn(OutF);
    Close(OutF);
  End{KiIr};

Procedure SarokPontRendez(Var P:PontHalmaz; N:Word);
 { A P ponthalmaz bal-alsó sarokpontjára vonatkozó
   polárszög szerinti rendezése }
 Var
  Q:Pont;    {a sarokpont}
  i,i0:Word;
  Function Kisebb(i,j:Word):Boolean;
  {A (Q,P[i]) szög kisebb, mint a (Q,P[j]) szög?}
  Begin
    Kisebb:=
       ((P[i].y-Q.y)*(P[j].x-Q.x) < (P[j].y-Q.y)*(P[i].x-Q.x))
    Or (
        ((P[i].y-Q.y)*(P[j].x-Q.x)=(P[j].y-Q.y)*(P[i].x-Q.x)) And
        ((P[i].x<P[j].x)Or((P[i].x=P[j].x) and (P[i].y<P[j].y)))
       )
  End;
```

```
54    Function Feloszt( Bal,Jobb : Word): Word ;
55      Var
56        E : Pont;
57        i,j,Fe : Word;
58      Begin
59        Fe := (Bal+Jobb) Div 2;
60        i := Bal-1; j := Jobb+1;
61        While True Do Begin
62          Repeat
63            Inc(i)
64          Until (Fe=i)Or Kisebb(Fe, i);
65          Repeat
66            Dec(j)
67          Until (Fe=j)Or Kisebb(j, Fe);
68          If i < j  Then Begin
69             E :=P[i]; P[i]:=P[j]; P[j] := E;
70          End Else Begin
71            Feloszt:= j; Exit
72          End;
73        End;
74      End (Λ Feloszt Λ);

75    Procedure Rendez(Bal,Jobb : Integer);
76      Var
77        f : Word;
78      Begin
79        f:= Feloszt(Bal, Jobb);
80        If Bal<f Then
81          Rendez(Bal, f);
82        If f+1 < Jobb Then
83          Rendez(f+1, Jobb)
84      End (Λ Rendez Λ);
85    Begin{SarokPontRendez}
86      i0:=1;              {a bal-alsó sarokpont meghatározása}
87      For i:=2 To N Do
88        If (P[i].x<P[i0].x)or
89           ((P[i].x=P[i0].x)And(P[i].y<P[i0].y)) Then
90          i0:=i;
91      Q.x:=P[i0].x; Q.y:=P[i0].y; Q.az:=P[i0].az;
92      P[i0]:=P[1]; P[1]:=Q;
93      Rendez(2, N)
94    End{SarokPontRendez};

95  Procedure Szamol;
96  {Global: P, N }
97    Var
98      j:Word;
99    Begin{Szamol}
100     SarokPontRendez(P,N);
101
102     j:=N-1;
103     While ((P[N].y-P[1].y)Λ(P[j].x-P[1].x)=
104            (P[j].y-P[1].y)Λ(P[N].x-P[1].x)) Do Dec(j);
105     KiIr(j);
106     {Output: S[1..j],S[N..j+1]}
107   End{Szamol};
108
109 Begin{Program}
```

```
110     Beolvas;
111     Szamol;
112   End.
```

15. Ponthalmaz konvex burka

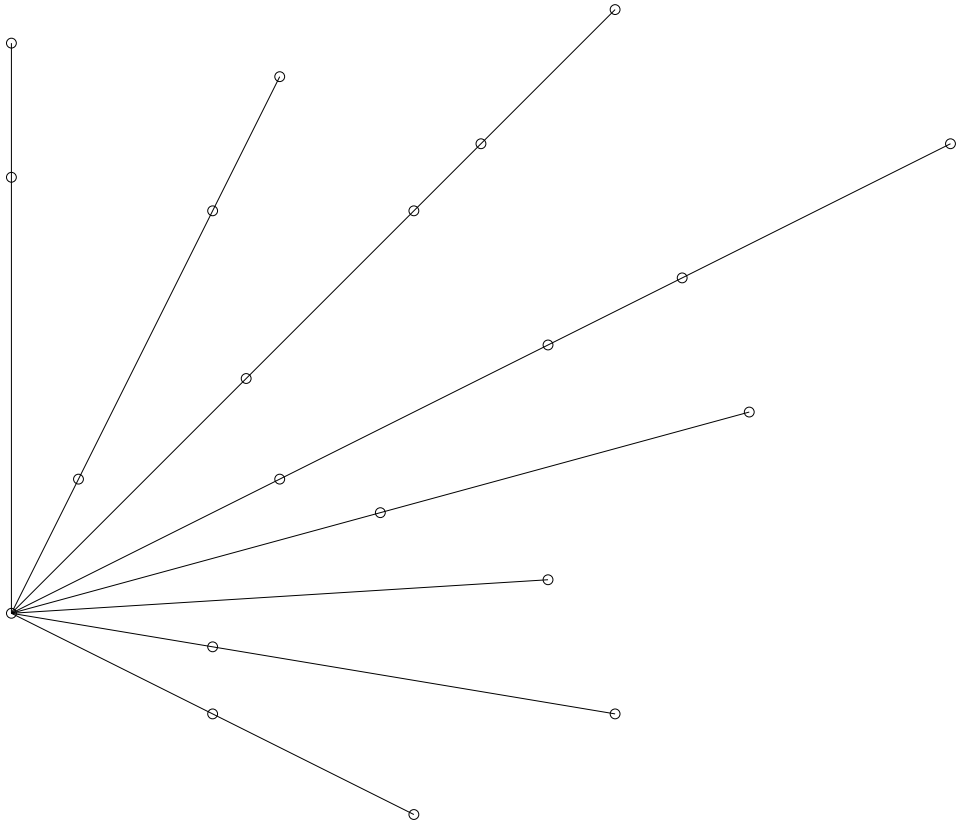
Definíció. Egy egyszerű (nem metsző) poligon konvex, ha bármely két belső pontját összekötő szakasz minden pontja a poligon belsejében, vagy határán van.
Definíció. Egy $P = \{p_1, \ldots, p_n\}$ ponthalmaz konvex burka az a legszűkebb Q konvex poligon, amely a ponthalmaz minden pontját tartalmazza.
Megoldás. Első lépésként rendezzük a ponthalmazt a bal-alsó sarokpontra vett polárszög szerint, majd minden egyenesen csak



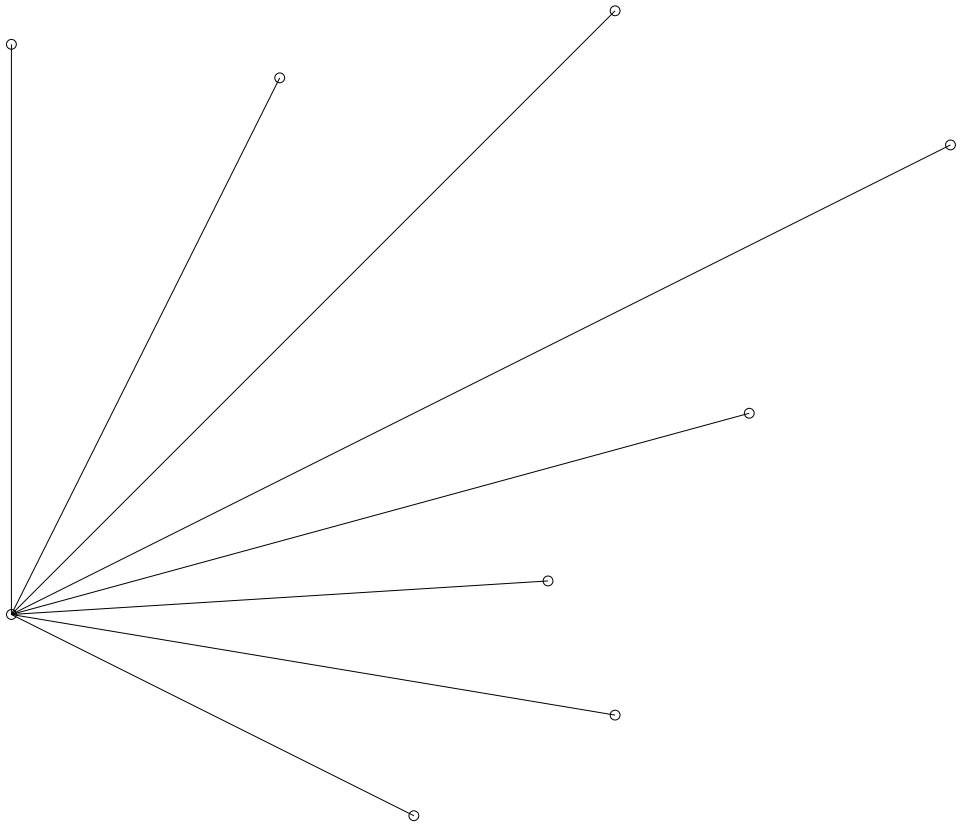
17. ábra. Ponthalmaz konvex burka.

a sarokponttól legtávolabbi pontot hagyjuk meg, a többit töröljük. Az így törölt pontok biztosan nem lehetnek csúcspontjai a konvex buroknak. Legyen $\langle q_1, \ldots, q_m \rangle$ az így kapott pontsorozat polárszög szerinti rendezésben. Mindaddig, amíg van három olyan cirkulárisan egymást követő q_i, q_{i+1} q_{i+2} pont, hogy $\overrightarrow{q_{i+1}q_{i+2}}$ nem balra fordul $\overrightarrow{q_iq_{i+1}}$-hez képest, hagyjuk el a q_{i+1} pontot. Az algoritmus futási ideje $O(n \lg n)$, ha a polárszög szerinti rendezést olyan algoritmussal valósítjuk meg amelynek futási ideje $O(n \lg n)$.

```
1   Function KonvexBurok(Var P:PontHalmaz; N:Longint):Longint;
2   {Bemenet: P[1],...,P[N],
3    Kimenet: P[1],...,P[M] a konvex burok csúcsai }
4   Var
5     M,i:Longint;
6     xy:Pont;
7   Begin{KonvexBurok}
8     SarokRendez(P,N); {A ponthalmaz bal-alsó sarokpont szerinti rendezése}
9     M:=2;
10    For i:=3 To N Do Begin
```



18. ábra. A pontok halmazának rendezése polárszög szerint.



19. ábra. Minden egyenesen csak a legtávolabbi pont marad meg.

```
11        While  (M>2)And(ForgasIrany(P[M-1], P[M], P[i])<=0)  Do
12          Dec(M);
13        Inc(M);
14        xy:=P[M]; P[M]:=P[i]; P[i]:=xy;
15      End{for};
16      KonvexBurok:=M;
17  End{KonvexBurok};
```

16. Feladat: Fekete-fehér párosítás a síkon.

Adott a síkon n darab fehér és n darab fekete pont úgy, hogy bármely három pont nem esik egy egyenesre. Párosítani kell a fehér pontokat fekete pontokkal úgy, hogy a párokat összekötő szakaszok ne metsszék egymást! A pontokat a $1,\ldots,n$ számokkal azonosítjuk.

Bemenet

A `paros.be` szöveges állomány első sora a fehér (és fekete) pontok $n\,(2 < n < 10000)$ számát tartalmazza. A további $2n$ sor mindegyike két egész számot tartalmaz, egy pont x és y koordinátáit, $(-20000 \leq x,y \leq 20000)$. Az első n sor a fehér, a második n sor a fekete pontokat tartalmazza.

Kimenet

A `paros.ki` szöveges állományba pontosan n sort kell kiírni, minden sorban egy fehér és a hozzá párosított fekete pont sorszáma álljon.

Példa bemenet és kimenet

bemenet

```
{paros}
 5
 6 17
 0 2
 14 1
 -2 23
 -7 19
 32 13
 26 14
 30 24
 21 22
 14 26
```

kimenet

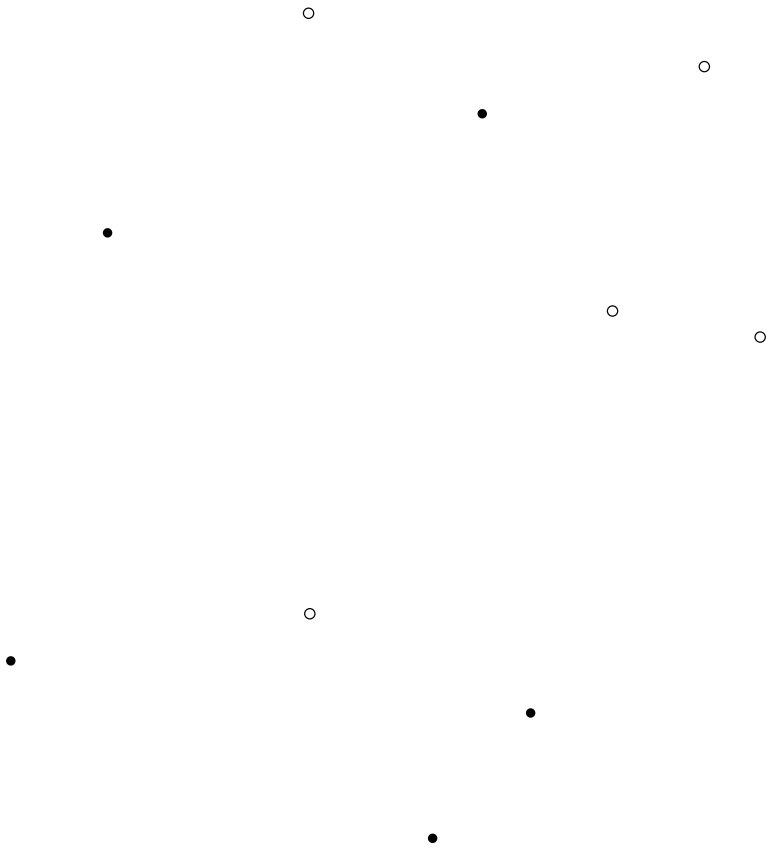
```
{paros}
 1 2
 2 4
 3 2
 4 1
 5 5
```

Megoldás

Legyen $P = \{p_1,\ldots,p_n,\ldots,p_{2n}\}$ a pontok halmaza.

16.1. lemma. *Létezik olyan p_i és p_j különböző színű pontpár, hogy az $e(p_i,p_j)$ egyenes mindkét oldalán a fehér pontok száma megegyezik a fekete pontok számával.*

Bizonyítás. Rendezzük a pontokat a bal-alsó sarokponthoz viszonyított polárszög szerint. Tegyük fel, hogy a rendezésben az első pont fekete. Jelölje $d_i, i = 1,\ldots,2n$. az első i pont közül a feketék számából kivonva a fehérek számát. Tehát $d_1 = 1$, $d_{2n} = 0$ és $d_{i+1} = d_i + 1$ ha az $i+1$-edik pont fekete, egyébként $d_{i+1} = d_i - 1$. Ha a rendezésben utolsó, azaz $2n$-edik pont fehér, akkor az 1. és $2n$-edik pontpár megoldás. Ha a $2n$-edik pont fekete, akkor $d_{2n-1} = -1$, de mivel $d_1 = 1$ és $d_{i+1} = d_i \pm 1$, így van olyan $1 < i < 2n-1$ index, hogy $d_i = 0$. Ha az i-edik pont nem fehér, akkor a keresést az $[1,i-1]$ intervallumban kell folytatni, ami véges sok lépés után végetér. ■



20. ábra. Párosítandó pontok

```
PAROSITAS(P,N)
   Procedure Parosit(bal, jobb);
   begin {Parosit}
      if jobb=bal+1 then begin
         KiIr(Bal,Jobb); exit;
      end;
      SarokPontRendez(P, bal jobb);
      d:=1; i:=bal+1
      while  true do begin
         if (P[bal].az>n) Xor (P[i].az>n) then
            d:=d-1
         else
            d:=d+1;
         if (d=0)and (P[bal].az>n) Xor (P[i].az>n) then break;
         i:=i+1;
      end {while}
      KiIr(bal, i);
      if bal+1 < i-1 then Parosit(bal+1, i-1);
      if i+1 < jobb then Parosit(i+1,jobb);
   end {Parosit}
begin {Parositas}
   Parosit(1, 2*n);
end {Parositas}
```

Az algoritmus futási ideje $O(n^2 \lg n)$.