

Geometriai algoritmusok

Horváth Gyula
horvath@inf.elte.hu

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.

1. Alapfogalmak

Pont: $(x, y) \in \mathbb{R} \times \mathbb{R}$

Megjegyzés: Csak olyan feladatokat 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 szakaszcól 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 typedef struct {
2     int x, y;
3     int azon;
4 } Pont;
```

típust használjuk, ahol p.azon a pont azonosítója, vagy sorszáma a feladatléírás szerint.

1.1. 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 $p_1 \times p_2$ **keresztsszorzat** 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 területként 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 órajárással azonos irányban van p_1 -hez képest.

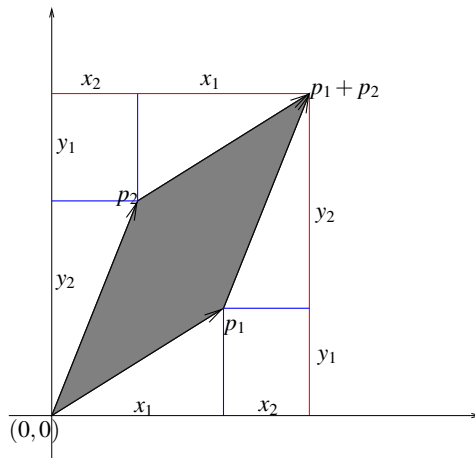
Még általánosabban, adott két csatlakozó irányított szakasz, $\overrightarrow{p_0 p_1}$ és $\overrightarrow{p_1 p_2}$. Milyen irányba fordul $\overrightarrow{p_1 p_2}$ a $\overrightarrow{p_0 p_1}$ -hez viszonyítva?

A válasz $(p_1 - p_0) \times (p_2 - p_0) = (p_1.x - p_0.x)(p_2.y - p_0.y) - (p_2.x - p_0.x)(p_1.y - p_0.y)$ keresztsszorzat előjele alapján megadható.

$(p_1 - p_0) \times (p_2 - p_0) > 0$: $\overrightarrow{p_1 p_2}$ balra fordul,

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

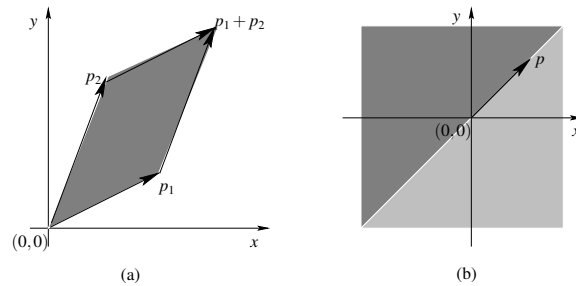
$(p_1 - p_0) \times (p_2 - p_0) < 0$: $\overrightarrow{p_1 p_2}$ jobbra fordul.



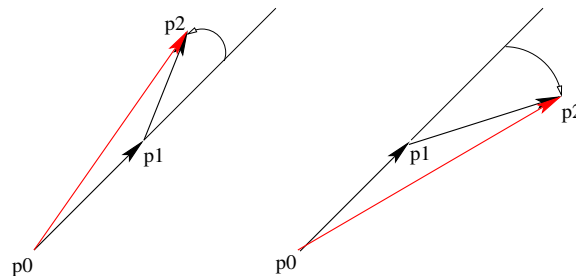
1. ábra. A paralelogramma előjeles területe:

$$T = (x_1 + x_2)(y_1 + y_2) - x_1y_1 - x_2y_2 - x_2y_1 - x_1y_2 =$$

$$x_1y_1 + x_1y_2 + x_2y_1 + x_2y_2 - x_1y_1 - x_2y_2 - x_2y_1 - x_1y_2 = x_1y_2 - x_2y_1$$



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.

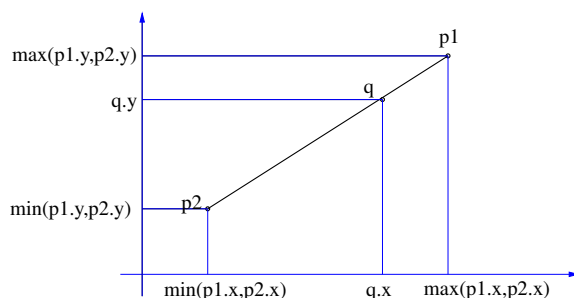
```

1  int ForgasIrany(Pont P0,Pont P1,Pont P2){
2  /\
3  Kimenet: +1 ha P1->P2 balra fordul,
4             0 ha P0,P1 és P2 kollineárisak,
5             -1 ha P1->P2 jobbra fordul.
6  /\
7  long long Kereszt=(long long)(P1.x-P0.x)^(P2.y-P0.y)
8                    -(long long)(P2.x-P0.x)^(P1.y-P0.y);
9  if (Kereszt<0)
10     return -1;
11 else if (Kereszt>0)
12     return 1;
13 else
14     return 0;
15 }

```

1.2. 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 $\min(p_1.y, p_2.y) \leq q.y \leq \max(p_1.y, p_2.y)$.

```

1  bool Szakaszon(Pont P1, Pont P2, Pont Q){
2  //Kimenet: akkor és csak akkor Igaz, ha a Q pont a P1-P2 szakaszon van
3  return ((P1.x-Q.x)^(P2.y-Q.y)-(P2.x-Q.x)^(P1.y-Q.y))==0) &&
4         Kozte(P1,P2,Q);
5  }

```

1.3. 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 :

```

1  bool Kozel(Pont P0,Pont P1,Pont P2){
2  //Feltétel: (P0,P1,P2) kollineáris
3  //Kimenet: P1 közelebb van P0-hoz, mint P2
4  return (abs(P1.x-P0.x)<abs(P2.x-P0.x)) ||
5         (abs(P1.y-P0.y)<abs(P2.y-P0.y)) ;
6  }

```

1.4. Közte

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

```

1  bool Kozte(Pont P1,Pont P2,Pont Q){
2  //Feltétel: (P1,P2,Q) kollineáris

```

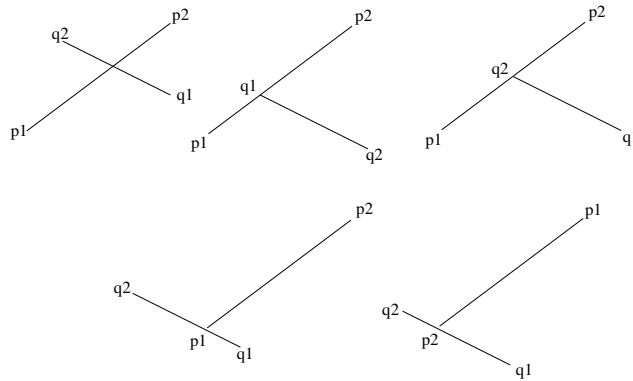
```

3 //Kimenet: P1 közelebb van P0-hoz, mint P2
4   return (abs(P1.x-Q.x)<=abs(P2.x-P1.x)) &&
5          (abs(P2.x-Q.x)<=abs(P2.x-P1.x)) &&
6          (abs(P1.y-Q.y)<=abs(P2.y-P1.y)) &&
7          (abs(P2.y-Q.y)<=abs(P2.y-P1.y)) ;
8 }

```

1.5. 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 bool SzakaszParMetsz(Pont P1,Pont P2, Pont Q1,Pont Q2 ){
2   int fpq1 ,fpq2 ,fqp1 ,fqp2 ;
3   fpq1=ForgasIrany(P1,P2, Q1);   fpq2=ForgasIrany(P1,P2, Q2);
4   fqp1=ForgasIrany(Q1,Q2, P1);   fqp2=ForgasIrany(Q1,Q2, P2);
5   return fpq1&fqp2<0 && fqp1&fpq2<0 ||
6          Szakazon(P1,P2, Q1) || Szakazon(P1,P2, Q2) ||
7          Szakazon(Q1,Q2, P1) || Szakazon(Q1,Q2, P2);
8 }

```

2. Feladat: Bekerítés

Adott a síkon a $P = \{p_1, \dots, p_n\}$ 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 előállításához, 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.

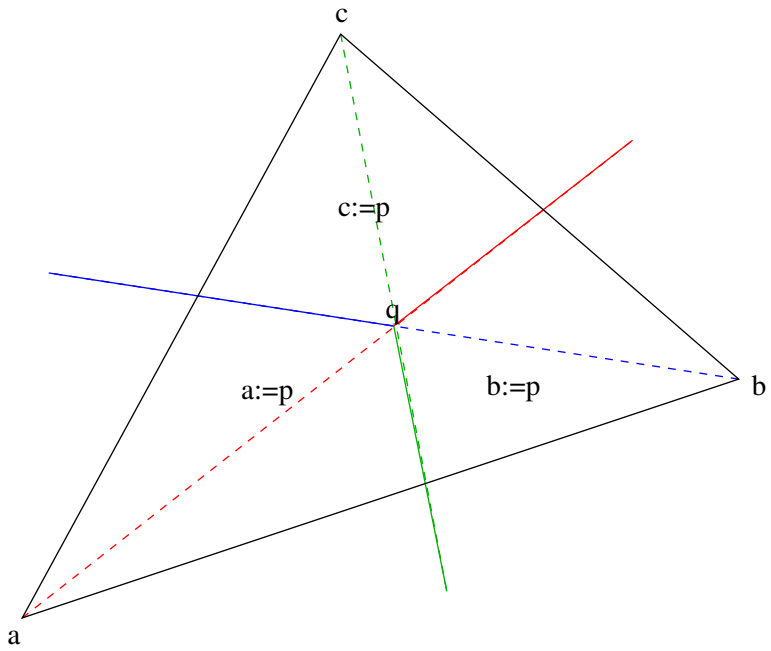
Ezután minden $p \in P$ pontra ($p \neq a, p \neq b$) határozzuk meg, hogy a (q, a) és (q, b) egyenesek által meghatározott síknyegyedek melyikébe esik p .

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.

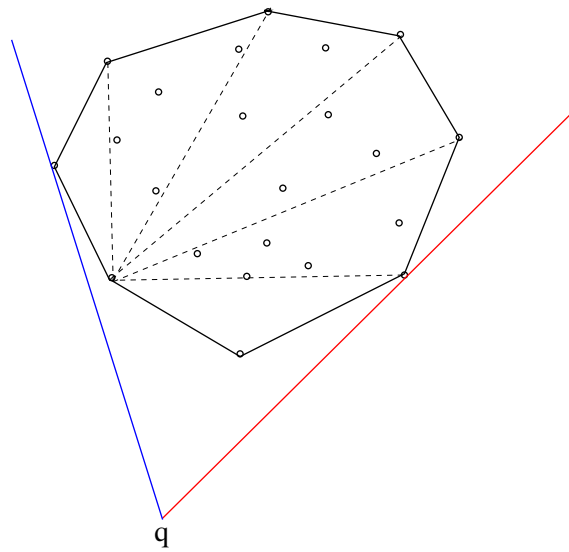
```

1 void Kerito3(Pont P[], int n, Pont q, int A, int B, int C){
2   /Δ
3   Bemenet: P ponthalmaz, q pont
4   Kimenet:
5   (i,0,0) ha P minden pontja az e(q,P[i]) egyenesre esik
6   (a,b,0) ha q a P konvex burkán kívül van, és ekkor
7           az e(q,P[a]) és e(q,P[b]) a két érintő egyenes

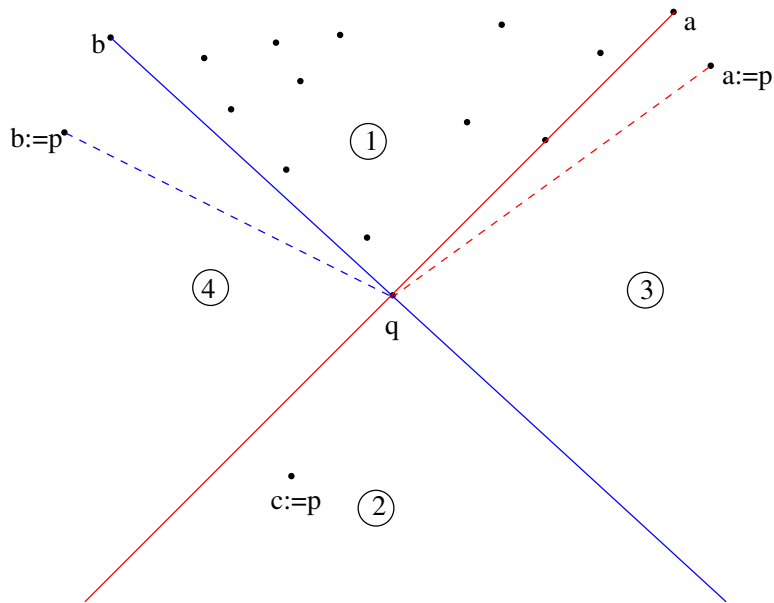
```



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 $\triangle(b, q, a)$ 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 $ForgasIrany(q, b, p) > 0$): $b := p$.

```

8  (a,b,c) ha q a P konvex burkán belül van, és ekkor
9      (P[a],P[b],P[c]) olyan háromszög, hogy q a háromszögben vagy oldalán van,
10     és P egyetlen pontja sem esik a háromszögbe, sem oldalára.
11  Δ/
12  int firqa , firqb , firqc ;
13  int i , j ;
14  Pont aP , bP , cP ;
15
16  aP=P[1] ; i=2 ;
17  while ( i<=n && ForgasIrany(q, aP, P[i])==0 ) i++ ;
18  if ( i>n ) { //pontjai egy egyenesre esnek
19      A=1 ; B=0 ; C=0 ;
20      exit ;
21  }
22  bP=P[i] ; cP.azon=0 ;
23  if ( ForgasIrany(q, aP, bP)<0 ) { //q-aP-tól bP balra essen
24      aP=bP ; bP=P[0] ;
25  }
26  for ( j=1 ; j<=n ; j++ ) {
27      firqa=ForgasIrany(q, aP, P[j]) ;
28      firqb=ForgasIrany(q, bP, P[j]) ;
29      if ( firqa<0 && firqb>=0 || firqa<=0 && firqb>0 ) { // 2. eset
30          cP=P[j] ; break ;
31      } else if ( firqa<0 && firqb<0 ) { // 3. eset
32          aP=P[j] ;
33      } else if ( firqa>0 && firqb>0 ) { // 4. eset
34          bP=P[j] ;
35      } // else 1. eset , nincs mit tenni , sem aP , sem bP nem változik
36  }

```

```

37 //ha cP.azon=0, akkor q kívül van, és ekkor e(q,aP) és e(q,bP) érintő egyenes
38   if (cP.azon==0){
39       A=aP.azon; B=bP.azon; C=0;
40       exit;
41   }

42 //Az (aP,bP,cP) háromszög tartalmazza a q pontot, finomítás:
43   for (i=1; i<=n; i++){
44       if (P[i].azon!=aP.azon && P[i].azon!=bP.azon && P[i].azon!=cP.azon &&
45           (ForgasIrany(aP,bP,P[i])>=0)&&(ForgasIrany(bP,cP,P[i])>=0)&&(ForgasIrany(cP,aP,P[i])>=0)
46           firqa=ForgasIrany(q,aP,P[i]);
47           firqb=ForgasIrany(q,bP,P[i]);
48           firqc=ForgasIrany(q,cP,P[i]);
49           if (ForgasIrany(q,aP,bP)==0 && ForgasIrany(P[i],aP,bP)==0){
50               if (firqc>0) aP=P[i]; else bP=P[i];
51           } else if (ForgasIrany(q,bP,cP)==0 && ForgasIrany(P[i],bP,cP)==0){
52               if (firqa>0) bP=P[i]; else cP=P[i];
53           } else if (ForgasIrany(q,cP,aP)==0&& ForgasIrany(P[i],cP,aP)==0){
54               if (firqb>0) cP=P[i]; else aP=P[i];
55           } else {
56               if (firqb<=0 && firqc>=0)
57                   aP=P[i];
58               else if (firqa>=0 && firqc<=0)
59                   bP=P[i];
60               else
61                   cP=P[i];
62           }
63       }
64   }
65   A=aP.azon; B=bP.azon; C=cP.azon;
66 }

```

3. Feladat: Pont helyzetének megállapítása.

Adott a síkon egy zárt, nem-metsző poligon a csúcsainak p_1, \dots, 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 $\overline{q_0q}$ szakaszra, legfeljebb ha q megegyezik valamelyik csúcscsal. (Van ilyen pont.)

Ha a $\overline{q_0q}$ szakasz a poligon páratlan számú oldalát metszi, akkor q belső pont, egyébként külső. A q_0 külső pont választása.

Legyen $q_0.y = \max\{p_i.y \mid p_i.y < q.y, i = 1, \dots, n\}$ és $q_0.x = \min\{p_i.x \mid i = 1, \dots, n\} - 1$.

Ha a poligonnak nincs olyan csúcspontja, amelynek y -koordinátája kisebb, mint $q.y$, akkor q biztosan külső pont. A q_0 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 $\overline{q_0q}$ 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:

```

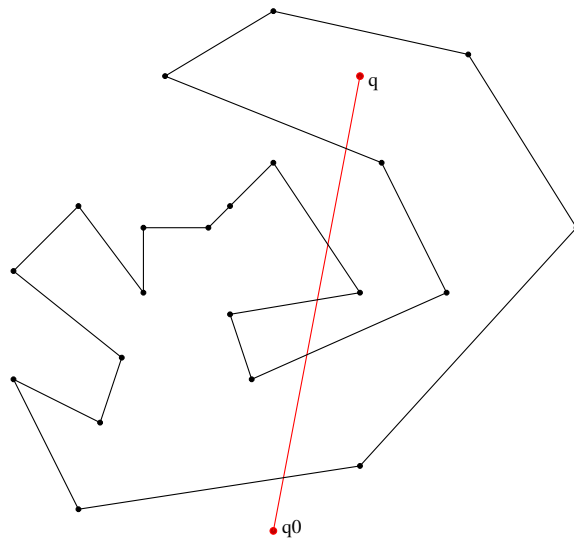
(ForgasIrany(P[i],P[i+1],q0)*ForgasIrany(P[i],P[i+1],Q)<0) And
(ForgasIrany(q0, Q, P[i])*ForgasIrany(q0, Q, P[i+1]) < 0 )

```

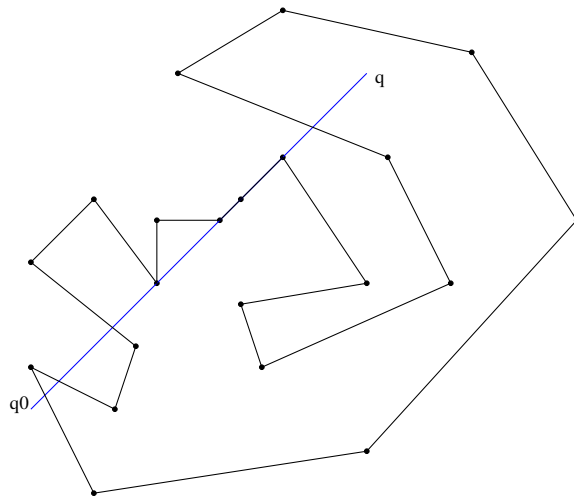
```

1  int Ponthelyete(Pont P[], int n, Pont Q){
2  /Δ
3  Kimenet: -1 ha Q a poligon belsejében van,
4           0 ha az oldalán,
5           1 ha kívül van
6  Δ/

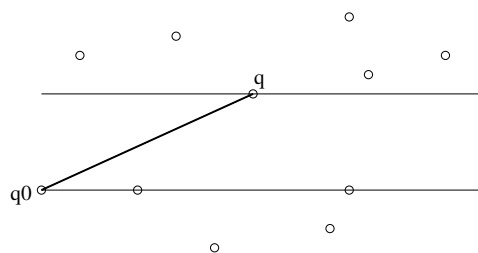
```



9. ábra. A $\overline{q_0q}$ szakasz a poligon páratlan sok oldalát metszi, tehát belső pont.



10. ábra. A poligonnak több csúcsa is a $\overline{q_0q}$ szakaszra esik.



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

```

7   long long y0,x0;
8   int i,ii ,metsz;
9   Pont q0;
10
11  y0=Q.y; x0=P[0].x;
12  for (i=1; i<=n; i++){ //külső pont választás
13      if (P[i].y<Q.y && (P[i].y>y0 || y0==Q.y ) )
14          y0=P[i].y;
15      if (P[i].x<x0) x0=P[i].x;
16  };
17  if (y0==Q.y) //Q kölső pont
18      return 1;
19
20  q0.y=y0; q0.x=x0-1; //q0 a külső pont
21
22  metsz=0;
23  for (i=1; i<=n; i++) {
24      ii=(i+1) % n;
25      if (Szakaszon(P[i], P[ii], Q) ) return 0;
26      if (ForgasIrany(P[i],P[ii],q0)∧ForgasIrany(P[i],P[ii],Q)<0 &&
27          ForgasIrany(q0, Q, P[i])∧ForgasIrany(q0, Q, P[ii]) <0 )
28          metsz++;
29  };
30  if (metsz %2 ==1)
31      return -1;
32  else
33      return 1;
34 } // Ponthelyzete

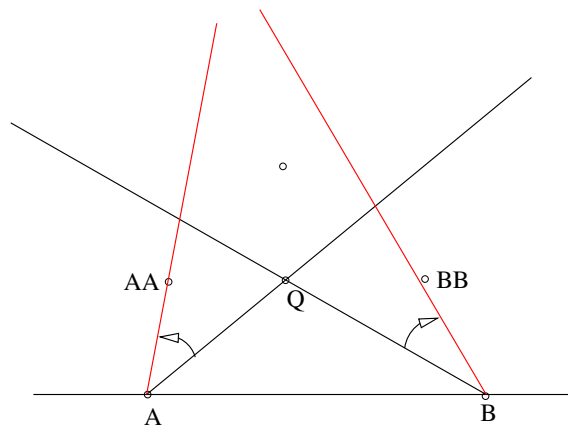
```

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

4. 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.



12. ábra.


```

1  int HaromPont(Pont P[], int n, Pont A, Pont B, Pont Q){
2      int FirAB, FirBQ, FirQA, FirAAA, FirBBB, FirAC, FirBC;
3      Pont AA, BB, C;
4      AA.azon=0; BB.azon=0; C.azon=0;
5      for ( int i=1; i<=n; i++){ //AA és BB meghatározása
6          if (P[i].azon==A.azon || P[i].azon==B.azon) continue;
7          FirAB= ForgasIrand(A,B,P[i]);
8          FirBQ= ForgasIrand(B,Q,P[i]);
9          FirQA= ForgasIrand(Q,A,P[i]);
10         if (FirAB>=0 && FirBQ>=0 && FirQA>=0)
11             return 0;
12
13         if (FirAB>0 && FirQA<0 && FirBQ>0)
14             if (AA.azon==0 || ForgasIrand(A,AA,P[i])<0)
15                 AA=P[i];
16
17         if (FirAB>0 && FirQA>0 && FirBQ<0)
18             if (BB.azon==0 || ForgasIrand(B,BB,P[i])>0)
19                 BB=P[i];
20     };
21
22     for (int i=1; i<=n; i++){ // egy C meghatározása
23         if (P[i].azon==A.azon || P[i].azon==B.azon || P[i].azon==AA.azon || P[i].azon==BB.azon)
24         if (AA.azon>0) FirAAA= ForgasIrand(A,AA,P[i]);
25         if (BB.azon>0) FirBBB= ForgasIrand(B,BB,P[i]);
26         FirQA= ForgasIrand(Q,A,P[i]);
27         FirBQ= ForgasIrand(B,Q,P[i]);
28         if ((AA.azon==0 || FirAAA<0) && (BB.azon==0 || FirBBB>0) &&
29             FirQA<0 && FirBQ<0){
30             C=P[i];
31             break;
32         };
33     };
34     if (C.azon==0) return 0;
35     for (int i=C.azon+1; i<=n; i++){ // C finomítása
36         if (P[i].azon==A.azon || P[i].azon==B.azon || P[i].azon==AA.azon || P[i].azon==BB.azon)
37         FirAB= ForgasIrand(A,B,P[i]);
38         FirAC= ForgasIrand(A,C,P[i]);
39         FirBC= ForgasIrand(B,C,P[i]);
40         if (FirAB>0 && FirAC<0 && FirBC>0) C=P[i];
41     };
42     return C.azon;
43 }

```

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

Adott a síkon n darab pont, amelyek nem esnek egy egyenesre. A pontok (x,y) koordinátáikkal adóttak, amelyek egész számok. A pontokat a $1, \dots, n$ számokkal azonosítjuk. Kössünk össze pontpárokat egyenes szakaszokkal úgy, hogy olyan zárt poligonot kapjunk, amelyben nincs metsző szakaspár. 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

6
2 0
1 4
0 2
3 2
2 4
2 6

kimenet

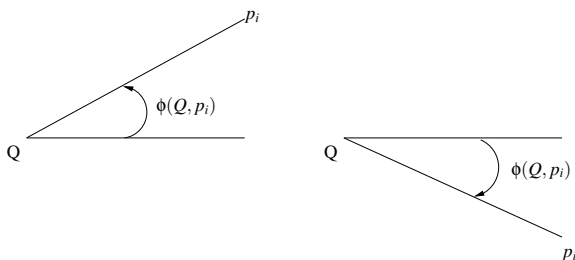
3 1 4 5 6 2

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

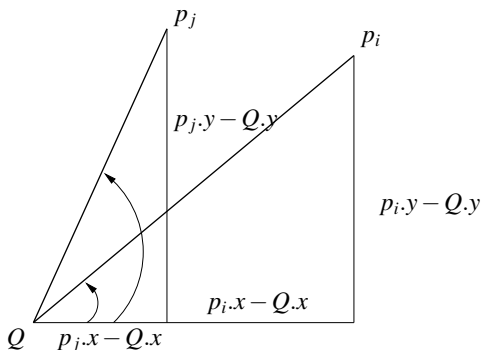
$\phi(Q, p_i) = \phi(Q, p_j)$ és p_i közelebb van Q -hoz, azaz $p_i.x < p_j.x$, vagy

$\phi(Q, p_i) = \phi(Q, p_j)$ 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 FORGASIR-ANY és a KOZEL (vagy KOZTE) műveletekkel:

```
1 #include <stdlib.h>
2 #include <stdio.h>
3 #include <iostream>
4 #include <math.h>
5 #include <limits.h>
6 #define maxN 100000
7 using namespace std;
8
9 typedef struct{
10     int x,y;
11     int azon;
12 }Pont;
13 Pont P[maxN];
14 Pont Q; //sarokpont
15
16 int SarokRend(const void* a, const void* b) {
17     long long ax = ((Pont*) a)->x - Q.x;
18     long long ay = ((Pont*) a)->y - Q.y;
19     long long bx = ((Pont*) b)->x - Q.x;
20     long long by = ((Pont*) b)->y - Q.y;
21     if (ax==bx && ay==by) return 0;
22     long long kereszt=ax*by - bx*ay;
23     if (kereszt > 0) return -1;
24     else if (kereszt < 0) return 1;
25     else {
26         if (abs(ax)<abs(bx) || abs(ay)<abs(by))
27             return -1;
28         else
29             return 1;
30     }
31 }
32
33 int main( int argc, char **argv){
34     int n;
35     int x,y, qaz=0;
36     cin>>n;
```

```

36 Q.x=INT_MAX;
37 for (int i=0; i<n; i++){
38     cin>>x; cin>>y;
39     P[i].azon=i+1;
40     P[i].x=x; P[i].y=y;
41     if (x<Q.x || x==Q.x && y<Q.y)
42         Q=P[i];
43 }
44 qsort((char *)P, n, sizeof(Pont), SarokRend);
45 int j=n-2;
46 while (j>0 && (long long)(P[j].x-Q.x)*(P[n-1].y-Q.y)==
47         (long long)(P[n-1].x-Q.x)*(P[j].y-Q.y)) j--;
48 for (int i=0; i<=j; i++)
49     cout<<P[i].azon<<"_";
50 for (int i=n-1; i>j; i--)
51     cout<<P[i].azon<<"_";
52 }

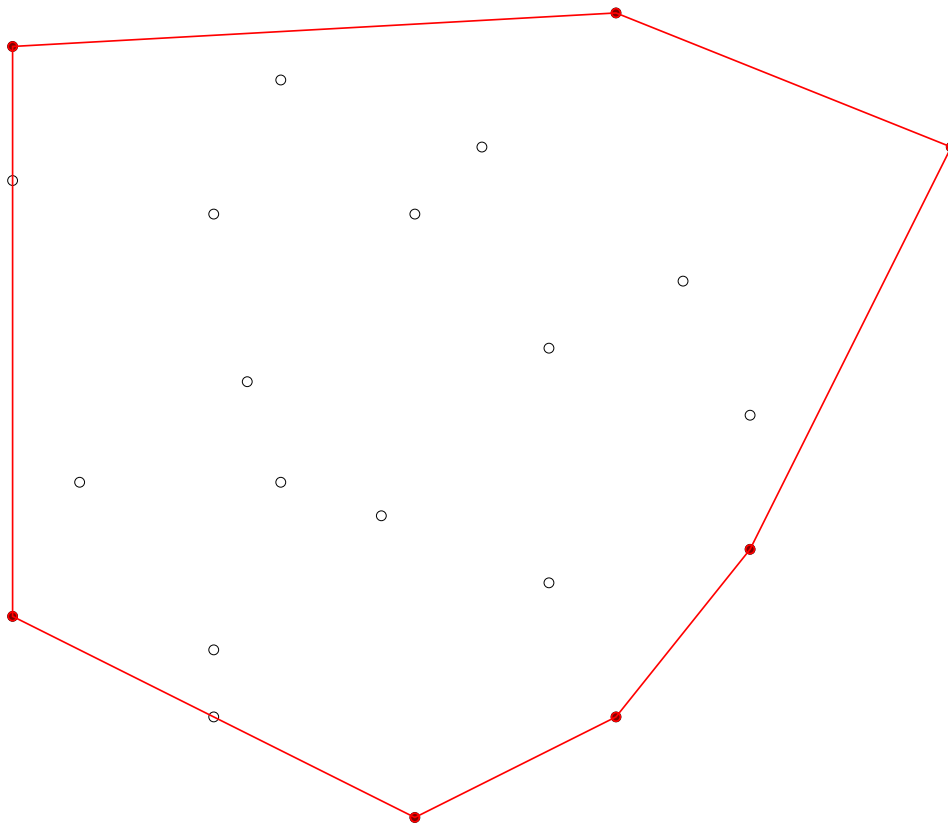
```

6. 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 $P = \{p_1, \dots, p_n\}$ pontthalmaz konvex burka az a legszűkebb Q konvex poligon, amely a pontthalmaz minden pontját tartalmazza.

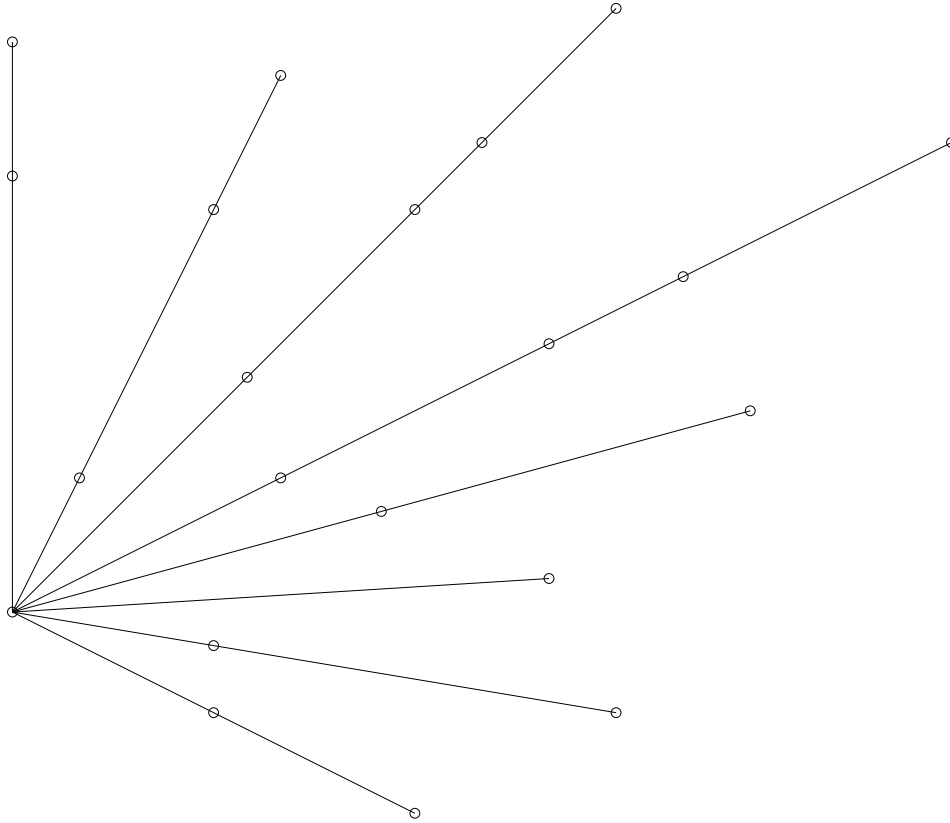
Megoldás. Első lépésként rendezzük a pontthalmazt a bal-alsó sarokpontra vett polárszög szerint, majd minden egyenesen csak



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 $\langle q_1, \dots, q_m \rangle$ 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 $\vec{q_{i+1}q_{i+2}}$ nem balra fordul $\vec{q_iq_{i+1}}$ -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 $O(n \lg n)$.

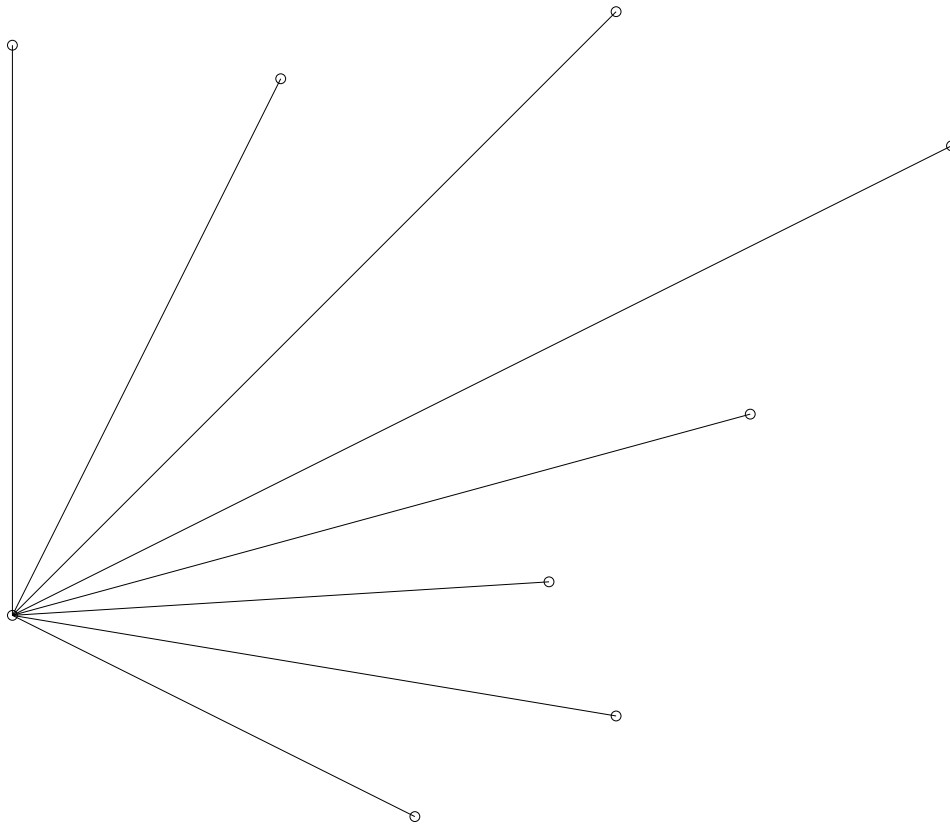


18. ábra. A pontok halmazának rendezése polárszög szerint.

```

1  Pont Q;
2  int SarokRend(const void& a, const void& b) {
3      long long ax =((Pont&) a)->x - Q.x;
4      long long ay =((Pont&) a)->y - Q.y;
5      long long bx =((Pont&) b)->x - Q.x;
6      long long by =((Pont&) b)->y - Q.y;
7      if (ax==bx && ay==by) return 0;
8      long long kereszt=ax&by - bx&ay;
9      if (kereszt > 0) return -1;
10     else if (kereszt < 0) return 1;
11     else {
12         if (abs(ax)<abs(bx) || abs(ay)<abs(by))
13             return -1;
14         else
15             return 1;
16     }
17 }
18
19 void SarokRendez(Pont P[], int n){
20     Q=P[0];
21     for (int i=1; i<n;i++)
22         if (P[i].x<Q.x || Q.x==P[i].x && P[i].y<Q.y)
23             Q=P[i];

```



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

```

24     qsort((char *)P, n, sizeof(Pont), SarokRend);
25 }

26 int KonvexBurok(Pont P[], int n){
27 //Bemenet: P[0],...,P[n-1],
28 // Kimenet: P[0],...,P[m] a konvex burok csúcsai }
29     int m, i;
30     Pont xy;
31
32     SarokRendez(P, n); //A ponthalmaz bal–alsó sarokpont szerinti rendezése}
33     m=1;
34     for (i=2; i<n; i++){
35         while (m>2 && ForgasIrany(P[m-1], P[m], P[i])<=0) m--;
36         m++;
37         xy=P[m]; P[m]=P[i]; P[i]=xy;
38     };
39     return m;
40 }

```

7. 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 metszék egymást! A pontokat a $1, \dots, n$ 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 $P = \{p_1, \dots, p_n, \dots, p_{2n}\}$ a pontok halmaza.

7.1. 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 $d_i, i = 1, \dots, 2n$, 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 véget ér. ■

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;

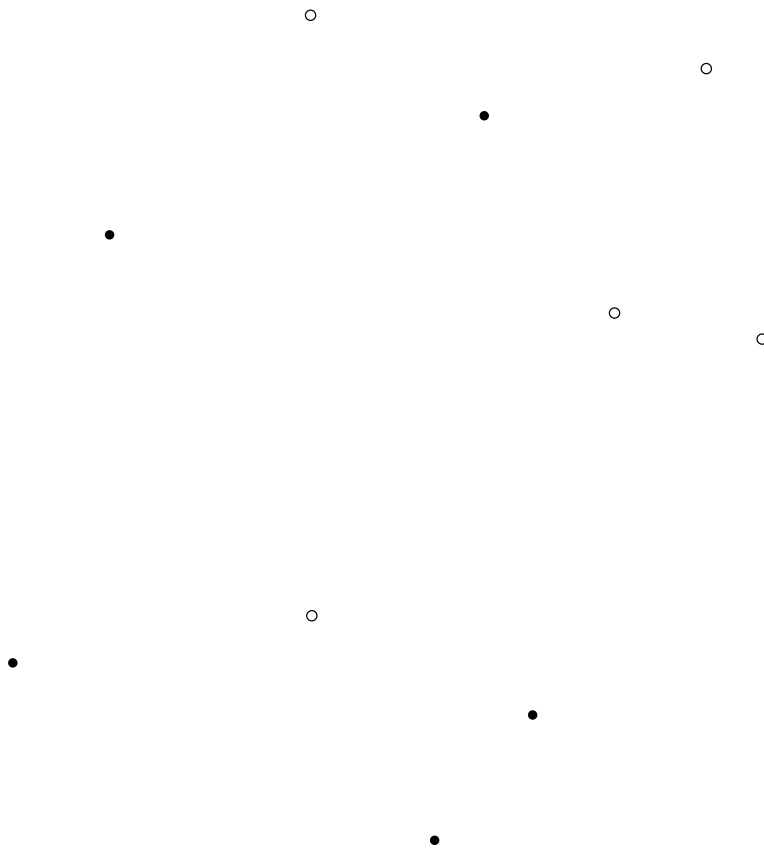
if (d=0)and (P[bal].az>n) Xor (P[i].az>n) **then break**;

 i:=i+1;

end {while}

KiIr(bal, i);

if bal+1 < i-1 **then** Parosit(bal+1, i-1);



20. ábra. Párosítandó pontok

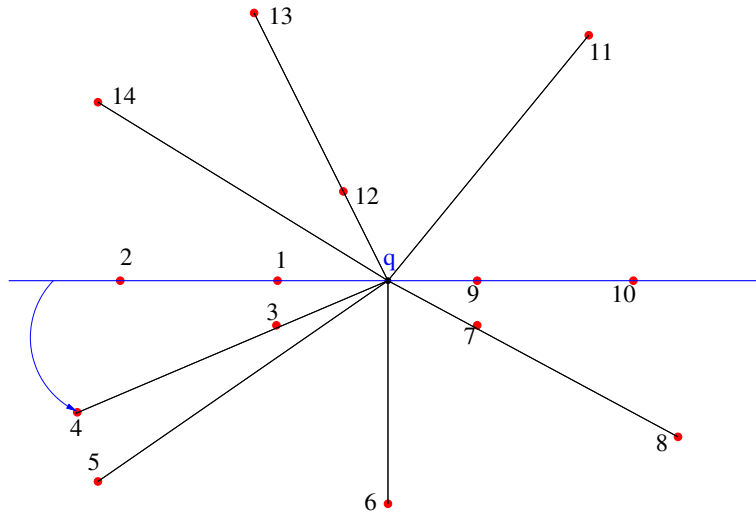
```

if i+1 < jobb then Parosit(i+1,jobb);
end {Parosit}
begin {Parositas}
  Parosit(1, 2*n);
end {Parositas}

```

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

Tetszőleges q pontra vonatkozó polárszög szerinti rendezés rendezési relációja is megadható a FORGASIRANY és a KOZEL felhasználásával.



21. ábra. A q pontra vett polárszög szerinti rendezés. A pont mellé írt szám a pontnak a rendezésbeli sorszáma.

```

41 Function PolarRend(Q,P1,P2:Pont): Integer ;
42 {A Q ponthoz viszonyított polárszög szerinti rendezésben P1 és P2 viszonya}
43 Var Fir: Integer; Begin
44   If (P1.x=P2.x) and (P1.y=P2.y) Then begin
45     PolarRend:=0; Exit; End;
46   Fir:=Forgasirany(Q,P1,P2);
47   Case Fir of
48     -1: If (p1.y<=q.y) and (p2.y>q.y) Then
49       PolarRend:=-1
50     Else
51       PolarRend:=1;
52     0 : If Kozel(q,p1,p2) and Kozel(p2,p1,q) or
53       Kozel(p1,q,p2) and Kozel(p2,q,p1) and (p1.y<=q.y)
54     Then
55       PolarRend:=-1
56     Else
57       PolarRend:=1;
58     +1: If (p1.y<=q.y) or (p2.y>q.y) Then
59       PolarRend:=-1
60     Else
61       PolarRend:=1;
62   End{ case };
63 End{ PolarRend };

```

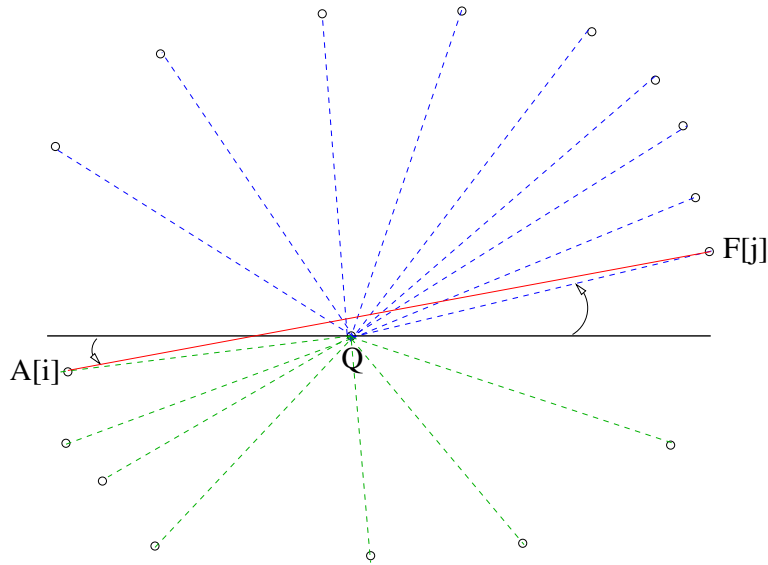
8. Feladat: Pontlefedés szakasszal

Adott a síkon egy P ponthalmaz és egy kitüntetett Q pontja. Kiszámítandó a P ponthalmaz két olyan A és B pontja, hogy a Q pont az $\overline{A, B}$ szakaszra esik.

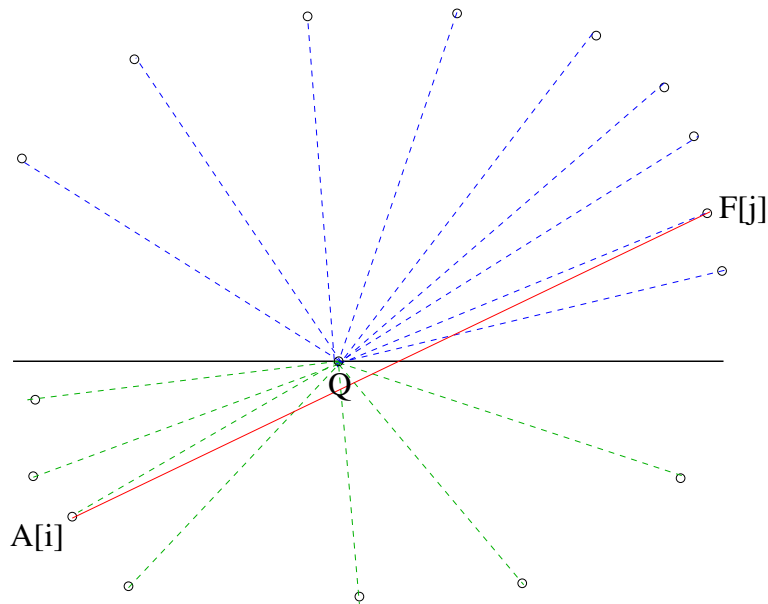
Követelmény: $O(n \log n)$ futási idejű algoritmus.

Megoldás

Osszuk két diszjunkt részhalmazba P pontjait (már a beolvasáskor) aszerint, hogy a Q -n átmenő, x-tengellyel párhuzamos



22. ábra. $i := i + 1$



23. ábra. $j := j + 1$

egyenes melyik oldalára esnek. Ha erre az egyenesre esik egy pont, akkor annak megfelelően, hogy Q előtt van-e, F -be kerüljenek az egyenes felettiak, A -be az alattiak. Majd rendezzük a két ponthalmazt a Q pontra vett polárszög szerint. Legyen $i := 1; j := 1$. Ha $Q \in \overline{A[i]F[j]}$ -től jobbra van, akkor az $A[i]$ -hez nincs olyan pont F -ben, amely megoldás. Hasonlóan, ha $Q \in \overline{A[i]F[j]}$ -től balra van, akkor az $F[j]$ -hez nincs olyan pont A -ben, amely megoldás. Tehát vagy i , vagy j növelhető és nem veszünk el lehetséges megoldást, azaz minden $ii < i$ és $jj < j$ esetén a $\overline{A[ii]F[jj]}$ nem lehet megoldás.

```

1  #include <stdlib.h>
2  #include <stdio.h>
3  #include <iostream>
4  using namespace std;
5  #define maxN 50000
6
7  typedef struct{
8      long long x,y;
9      int azon;
10 }Pont;
11
12 Pont A[maxN];
13 Pont F[maxN];
14 Pont Q;
15
16 int ForgasIrandy(Pont P0, Pont P1,Pont P2){
17     long long Kereszt=(P1.x-P0.x)^(P2.y-P0.y)-(P2.x-P0.x)^(P1.y-P0.y);
18     if (Kereszt<0)
19         return -1;
20     else if (Kereszt>1)
21         return 1;
22     else
23         return 0;
24 }
25
26 int SarokRend(const void^ a, const void^ b) {
27     long long ax =((Pont^ a)->x - Q.x;
28     long long ay =((Pont^ a)->y - Q.y;
29     long long bx =((Pont^ b)->x - Q.x;
30     long long by =((Pont^ b)->y - Q.y;
31     if (ax==bx && ay==by) return 0;
32     long long kereszt=ax^by - bx^ay;
33     if (kereszt > 0) return -1;
34     else if (kereszt < 0) return 1;
35     else {
36         if (abs(ax)<abs(bx) || abs(ay)<abs(by))
37             return -1;
38         else
39             return 1;
40     }
41 }
42
43 int main( int argc, char ^^argv){
44     int n, i, x,y;
45     cin>>n;
46     cin>>x; cin>>y;
47     Q.x=x; Q.y=y;
48     int n1=0, n2=0;
49     for (i = 0; i < n; i++){
50         cin>>x; cin>>y;
51         if (y<Q.y || y==Q.y && x<Q.x){
52             A[n1].x=x; A[n1].y=y; A[n1].azon=i;
53             n1++;
54         } else {
55             F[n2].x=x; F[n2].y=y; F[n2].azon=i;
56             n2++;
57         }
58     }
59 }

```