Geometriai algoritmusok

Horváth Gyula
Szegedi Tudományegyetem
Informatikai Tanszékcsoport
horvath@inf.u-szeged.hu

2006. január 13.

Számos olyan gyakorlati számítási probléma van, amely megoldható olyan geometriai modellben, amelyben csak egyszerű objektumok, pontok, egyenesek, szakaszok, szögek, szögtartományok, poligonok szerepelnek. Ilyen feladatokat vizsgálunk. Nem foglalkozunk olyan feladatokkal, amelyek lebegőpontos aritmetikát igényelnének.

1. Alapfogalmak

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

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: $e(p_1, p_2)$

2. 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.

Bemeneti specifikáció

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.

Kimeneti specifikáció

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

Példa bemenet és kimenet

poligon.be

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

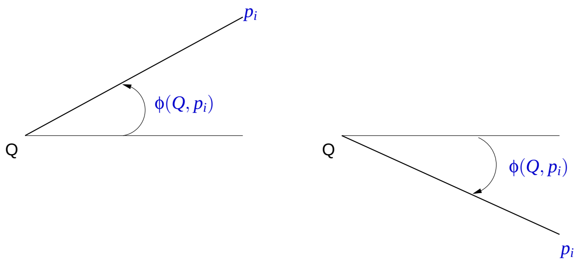
poligon.ki

```
3 1 4 5 6 2
```

Megoldás

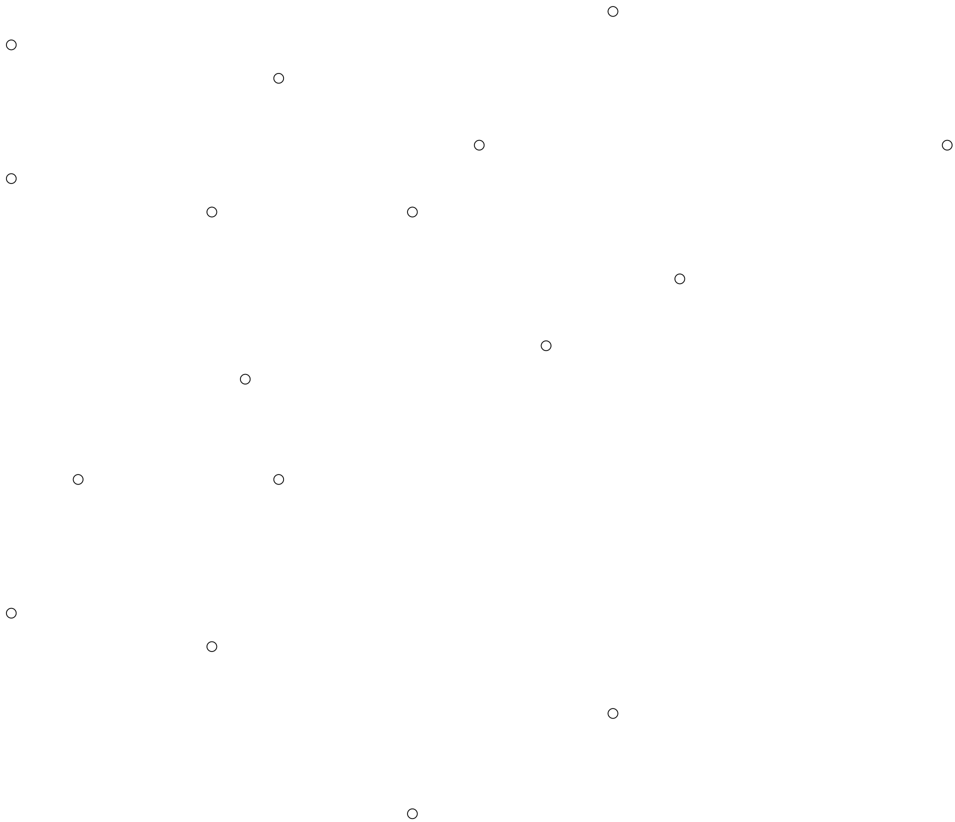
Válasszuk ki a legkisebb x-koordinátájú pontot, ha több ilyen van, akkor ezek közül válasszuk a legkisebb y-koordinátájút. Ezt nevezzük (bal-alsó) sarokpontnak és jelöljük Q-val. Húzzunk (fél) egyenest a Q sarokpontból minden ponthoz.

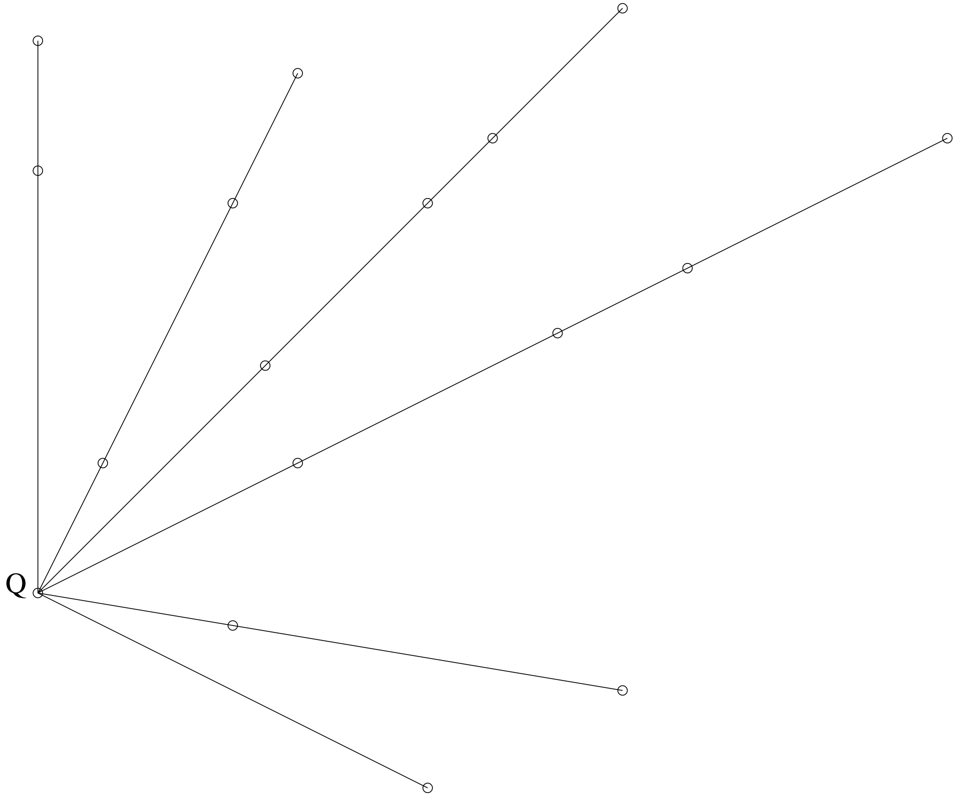
Rendezzük az egyeneseket a Q ponton áthaladó, x-tengellyel párhuzamos egyenessel bezárt (előjeles) szög alapján.

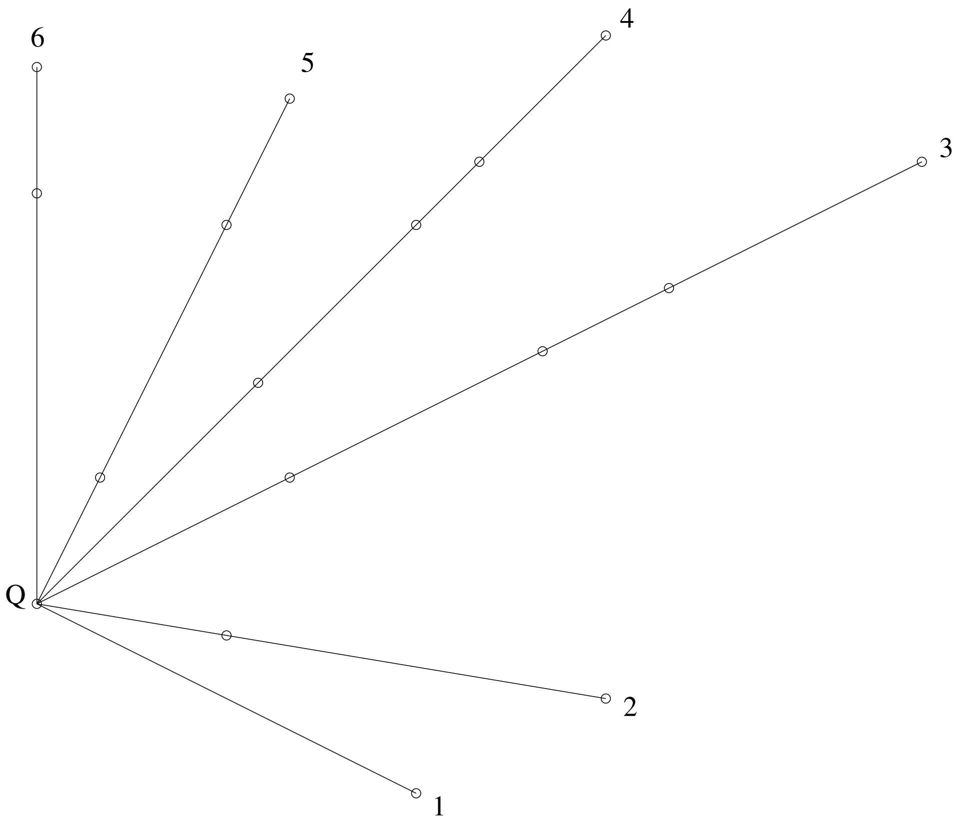


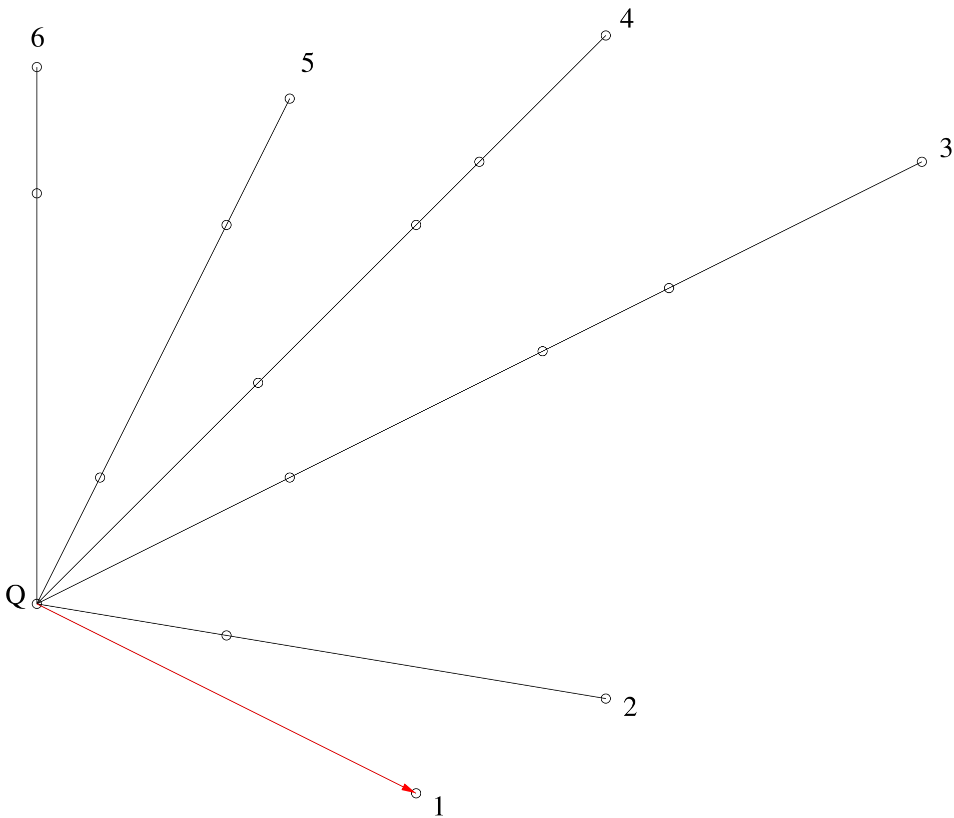
1. ábra. A p_i ponthoz tartozó előjeles szög: $\phi(Q, p_i)$.

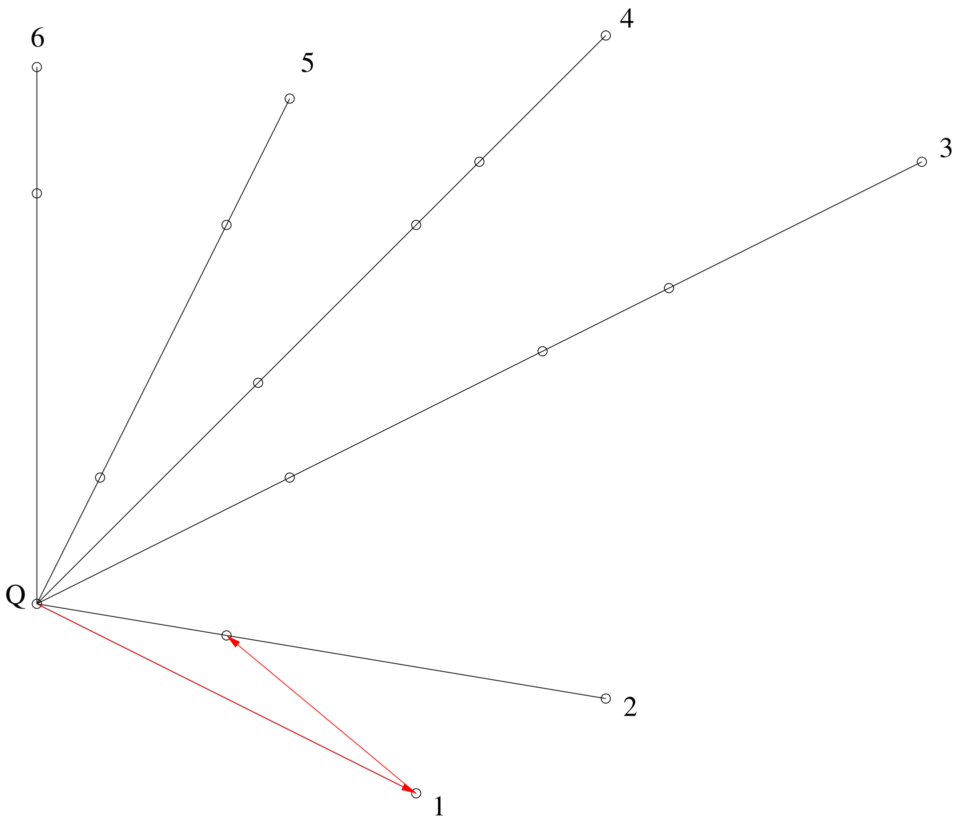
Rendezzük a pontokat úgy, hogy a Q sarokpont legyen az első, és p_i előbb legyen mint p_j akkor és csak akkor, ha
A $\phi(Q, p_i) < \phi(Q, p_j)$, vagy
$\phi(Q, p_i) = \phi(Q, p_j)$ és p_i közelebb van Q-hoz, azaz $p_i.x < p_j.x$, vagy
$\phi(Q, p_i) = \phi(Q, p_j)$ és $p_i.x = p_j.x$ és $p_i.y < p_j.y$.
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.



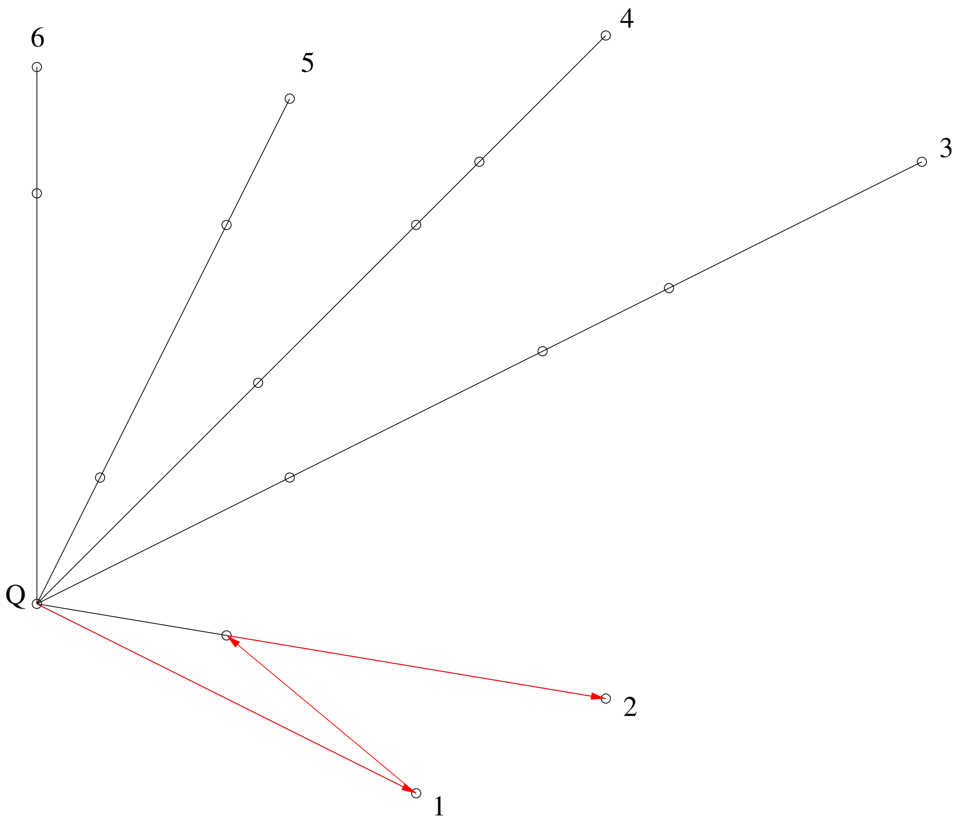

Q

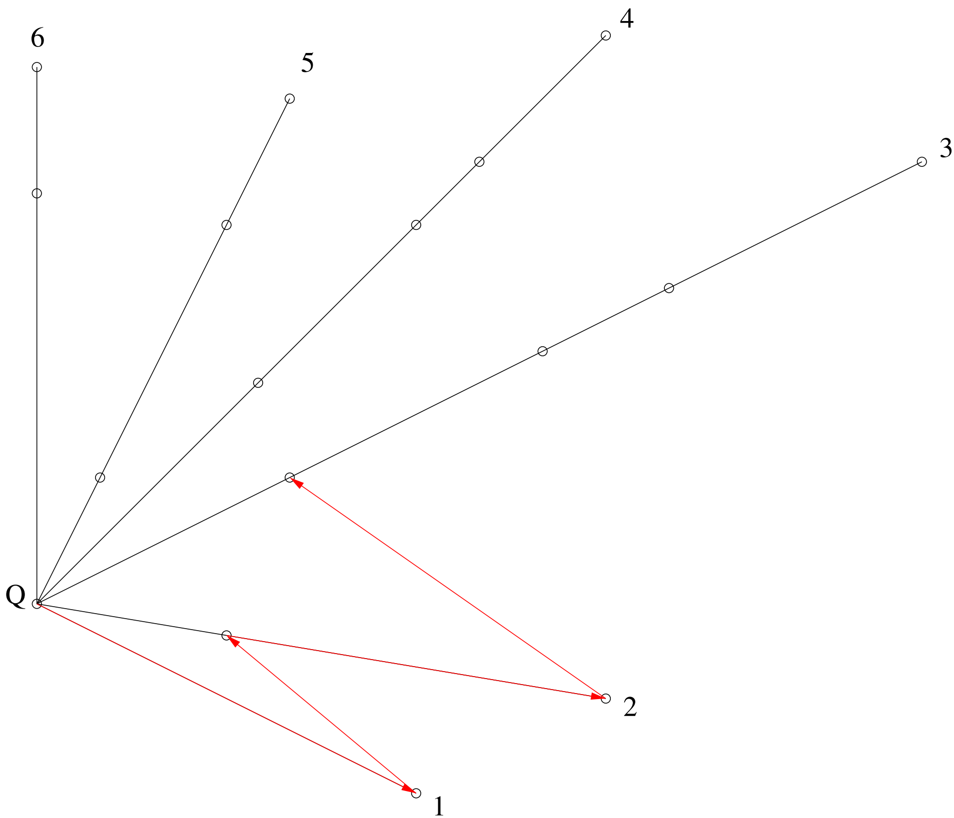

6
5
4
3
Q
2
1

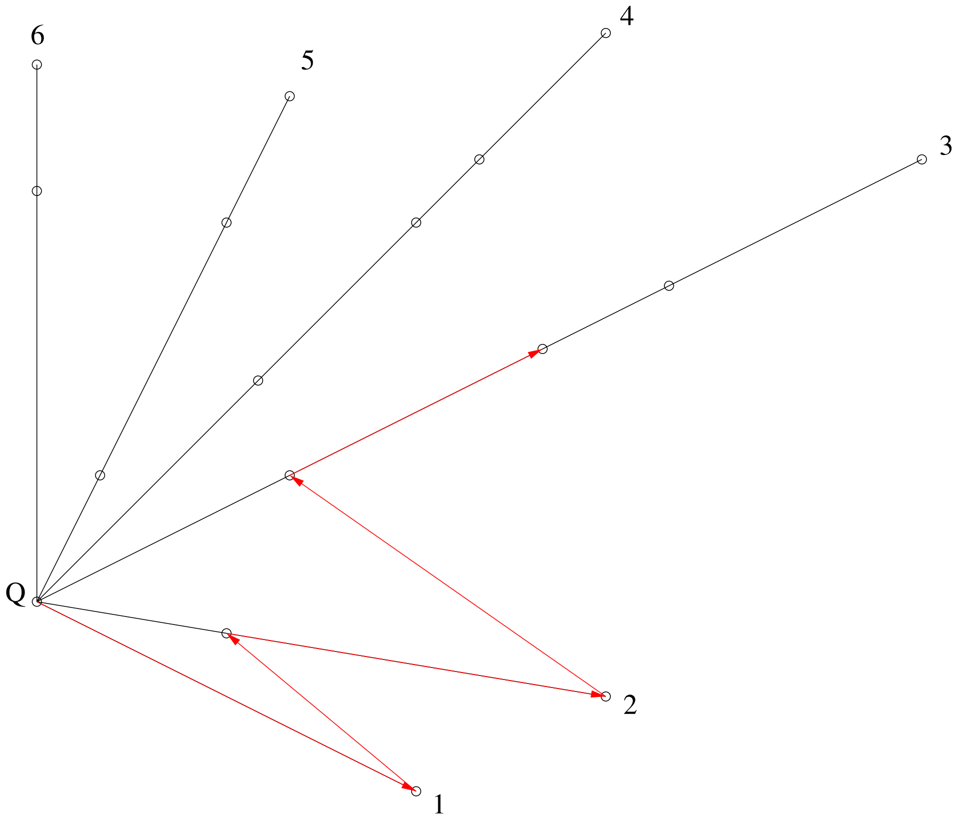

6
5
4
3
Q
2
1

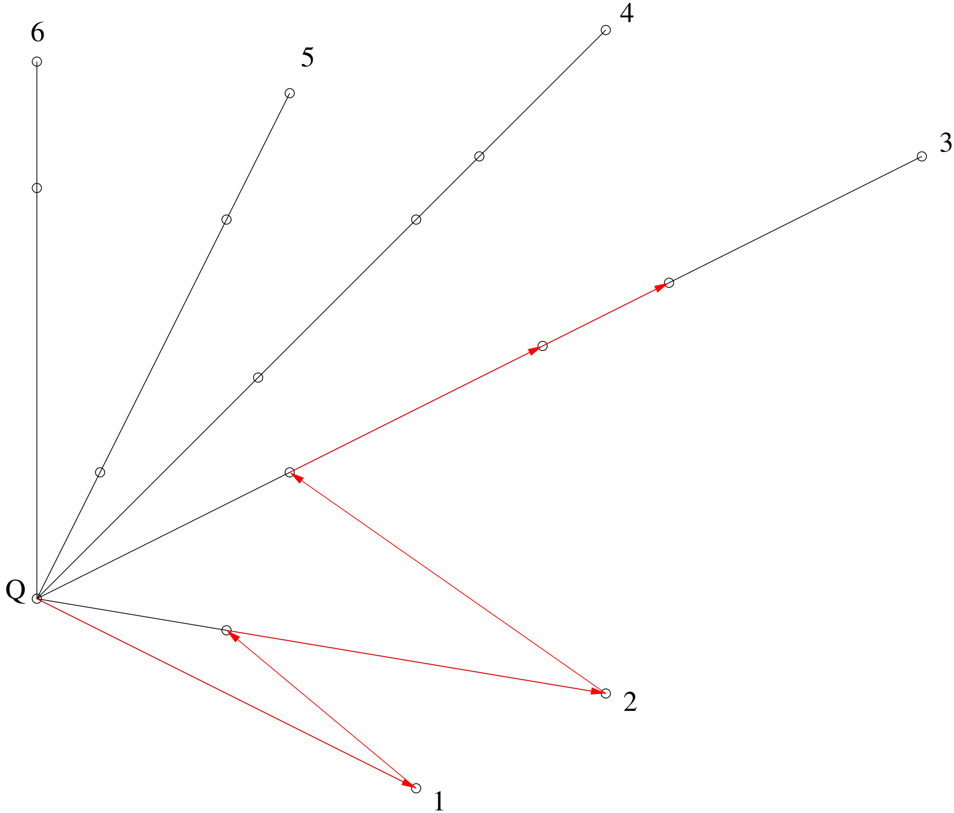


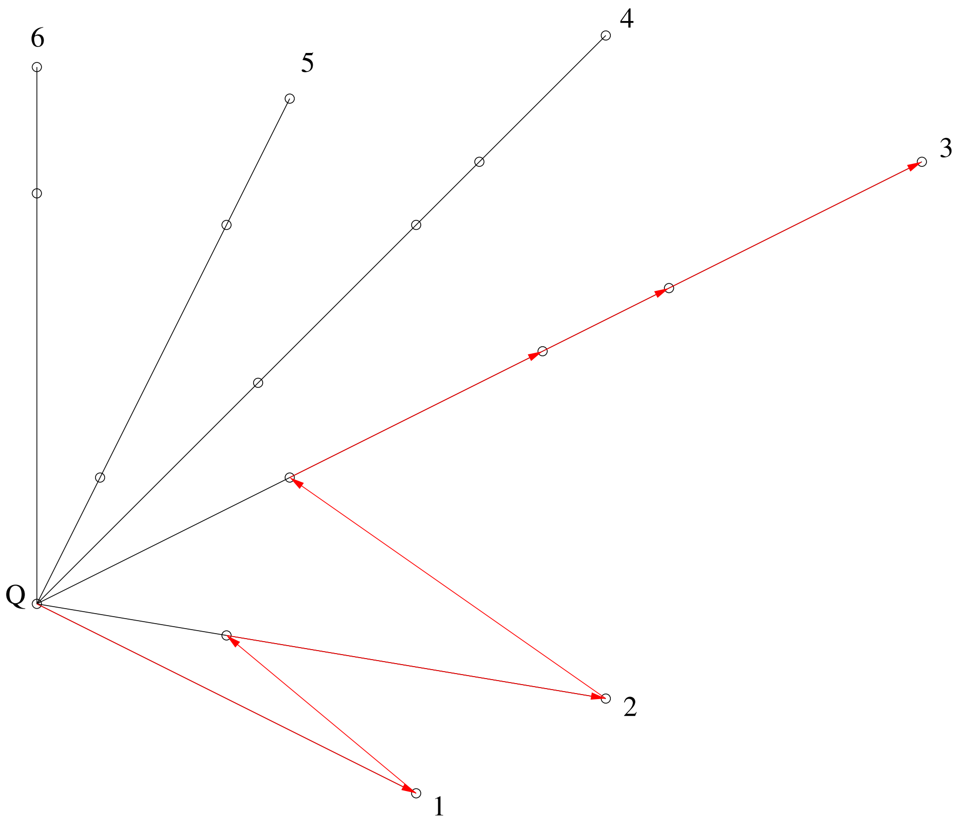
6
5
4
3
Q
2
1

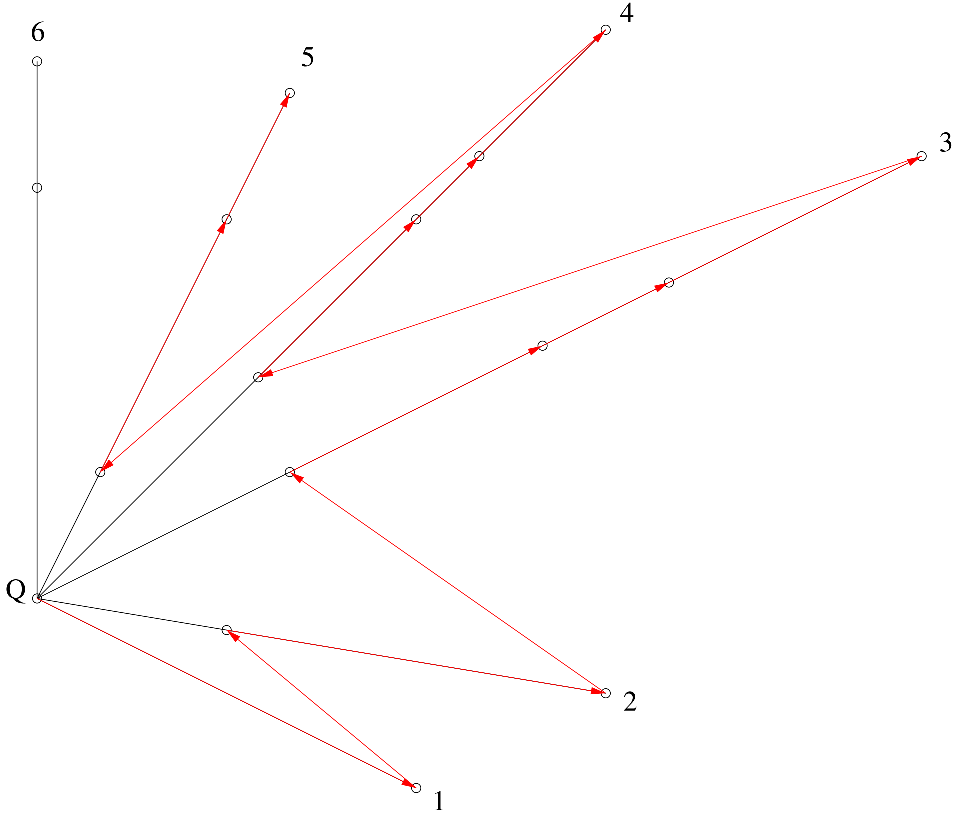

6
5
4
3
Q
2
1

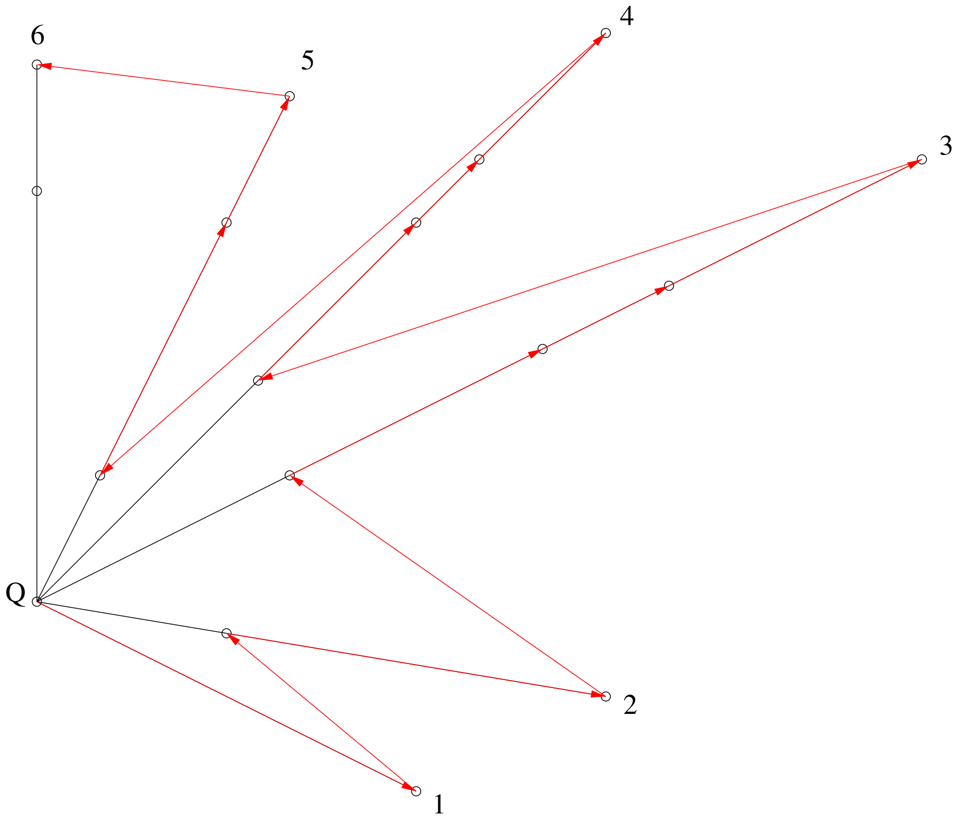

6
5
4
3
Q
2
1

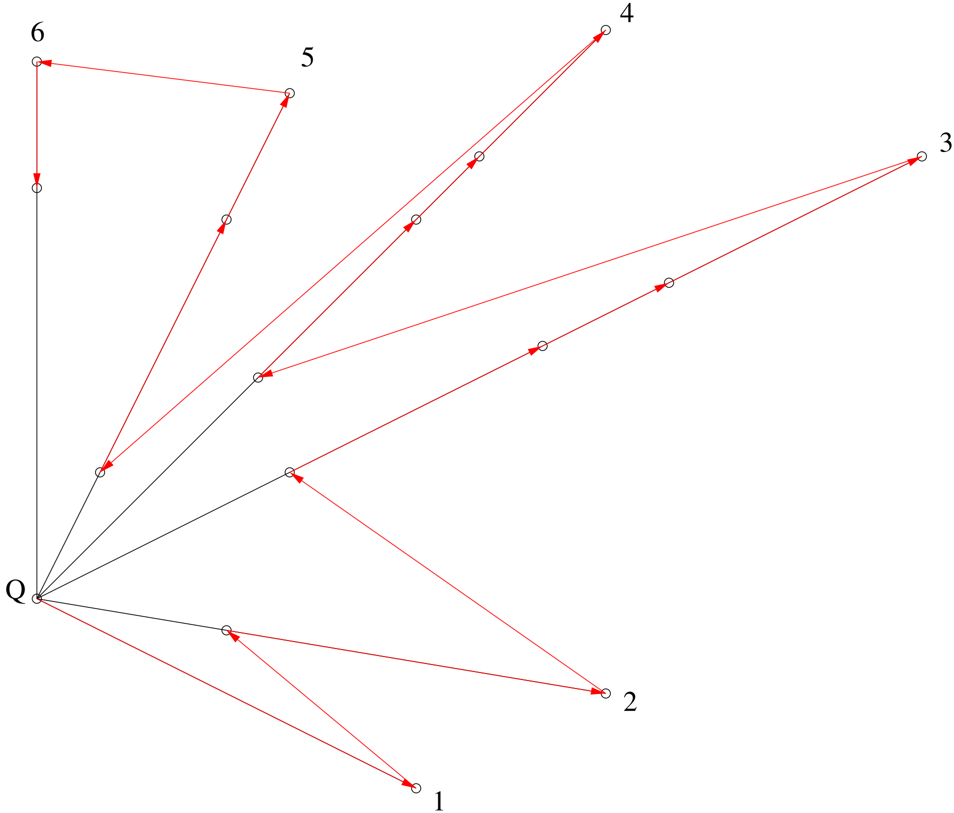

6
5
4
3
Q
2
1

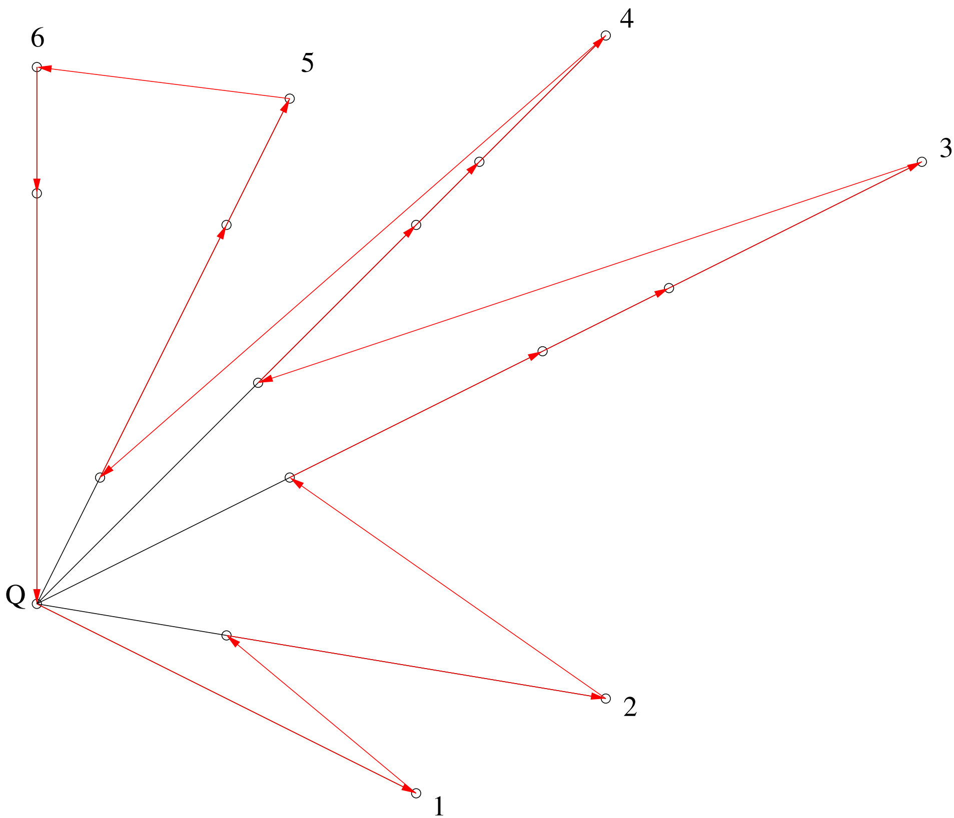

6
5
4
3
Q
2
1


6
5
4
3
Q
2
1

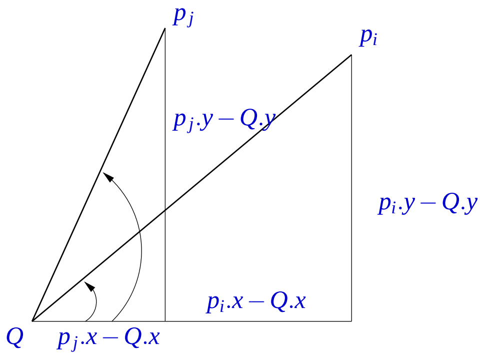

6
5
4
3
Q
2
1


6
5
4
3
Q
2
1


6
5
4
3
Q
2
1


6
5
4
3
Q
2
1

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



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

monoton növekvő, tehát
$\phi(Q, p_i) < \phi(Q, p_j)$ 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

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

```
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;
```

```
Function Feloszt( Bal,Jobb : Word): Word ;
  Var
    E : Pont;
    i,j,Fe : Word;
  Begin
    Fe := (Bal+Jobb) Div 2;
    i := Bal-1; j := Jobb+1;
    While True Do Begin
      Repeat
        Inc(i)
      Until (Fe=i)Or Kisebb(Fe, i);
      Repeat
        Dec(j)
      Until (Fe=j)Or Kisebb(j, Fe);
      If i < j  Then Begin
          E :=P[i]; P[i]:=P[j]; P[j] := E;
      End Else Begin
        Feloszt:= j; Exit
      End;
    End;
  End (* Feloszt *);
```

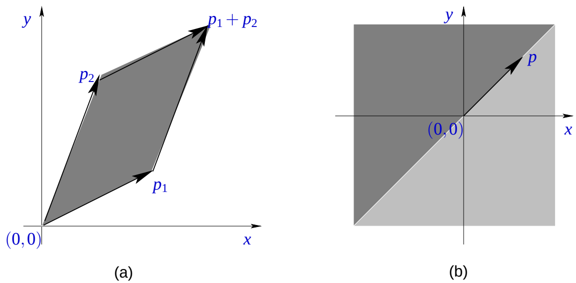
```
Procedure Rendez(Bal,Jobb : Integer);
  Var
    f : Word;
  Begin
    f:= Feloszt(Bal, Jobb);
    If Bal<f Then
      Rendez(Bal, f);
    If f+1 < Jobb Then
      Rendez(f+1, Jobb)
  End (* Rendez *);
Begin{SarokPontRendez}
  i0:=1;             {a bal-alsó sarokpont meghatározása}
  For i:=2 To N Do
    If (P[i].x<P[i0].x)or
       ((P[i].x=P[i0].x)And(P[i].y<P[i0].y)) Then
      i0:=i;
  Q.x:=P[i0].x; Q.y:=P[i0].y; Q.az:=P[i0].az;
  P[i0]:=P[1]; P[1]:=Q;
  Rendez(2, N)
End{SarokPontRendez};
```

```
Procedure Szamol;
{Global: P, N }
  Var
    j:Word;
  Begin{Szamol}
    SarokPontRendez(P,N);

    j:=N-1;
    While ((P[N].y-P[1].y)*(P[j].x-P[1].x)=
           (P[j].y-P[1].y)*(P[N].x-P[1].x)) Do Dec(j);
    KiIr(j);
    {Output: S[1..j],S[N..j+1]}
  End{Szamol};

Begin{Program}
  Beolvas;
  Szamol;
End.
```

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.

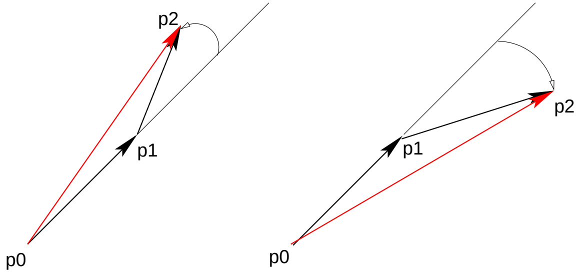


3. ábra. (a) A p_1 és p_2 vektorok keresztszorzata a paralelogramma előjeles területe. (b) A világos tartomány azokat a pontokat tartalmazza, amelyek a p-től órajárással egyező irányba esnek, a sötétebb pedig azokat, amelyek órajárással ellentétes irányba esnek p-től.

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

$p_1 \times p_2 > 0 \Leftrightarrow p_2$ 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, $\overrightarrow{p_0p_1}$ és $\overrightarrow{p_1p_2}$. Milyen irányba fordul $\overrightarrow{p_1p_2}$ a $\overrightarrow{p_0p_1}$-hez viszonyítva?
A válasz $(p_1 - p_0) \times (p_2 - p_0) = (p_1.x - p_0.x)(p_2.y - p_0.y) - (p_2.x - p_0.x)(p_1.y - p_0.y)$ ke-



4. ábra. Csatlakozó szakaszok forgásiránya.

resztszorzat előjele alapján megadható.
$(p_1 - p_0) \times (p_2 - p_0) > 0$: $\overrightarrow{p_1p_2}$ balra fordul,
$(p_1 - p_0) \times (p_2 - p_0) = 0$: $\overrightarrow{p_0p_1}$ és $\overrightarrow{p_1p_2}$ kollineárisak,
$(p_1 - p_0) \times (p_2 - p_0) < 0$: $\overrightarrow{p_1p_2}$ 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:=(P1.x-P0.x)*(P2.y-P0.y)-(P2.x-P0.x)*(P1.y-P0.y);
  If KeresztSzorz < 0 Then
    ForgasIrany:=-1
  Else If KeresztSzorz>0 Then
    ForgasIrany:=1
  Else
    ForgasIrany:=0;
End{ForgasIrany};
```

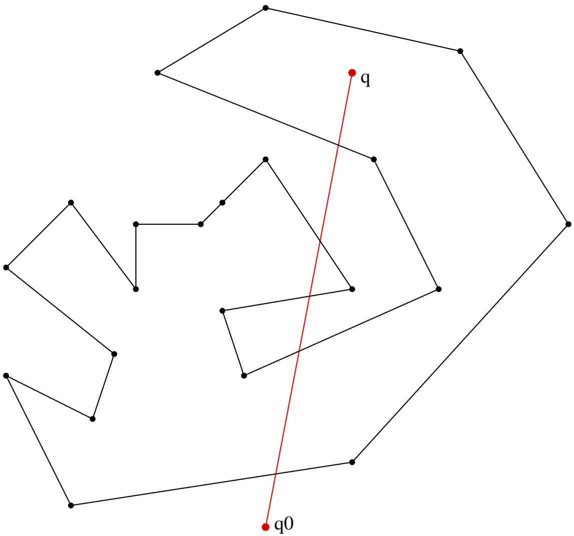
3. Feladat: Pont helyzetének megállapítása.

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.

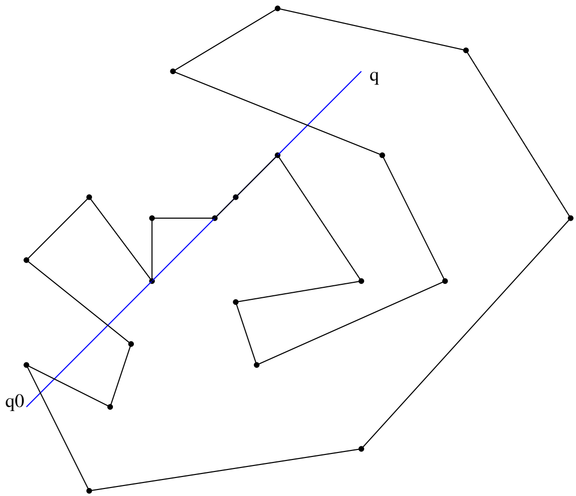
Megoldás

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.

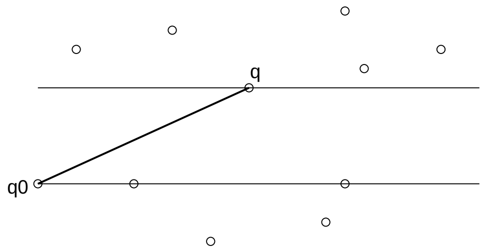


5. ábra. A $\overline{q_0\, q}$ szakasz a poligon páratlan sok oldalát metszi, tehát belső pont.



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

A q_0 külső pont választása.
Legyen $q_0.y = \max\{p_i.y | p_i.y < q.y, i = 1,\dots n\}$ és $q_0.x = \min\{p_i.x | i = 1,\dots n\} - 1$.
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

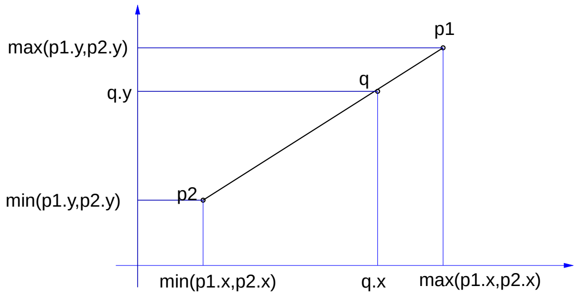


7. ábra. A q_0 pont választása:
$q_0.y = \max\{p_i.y | p_i.y < q.y, i = 1,\dots n\}$ és $q_0.x = \min\{p_i.x | i = 1,\dots n\} - 1$.

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:

```
(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 )
```

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



8. 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)*(P2.y-Q.y)-(P2.x-Q.x)*(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};
```

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

```
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:=0 To N-1 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:=0 To n-1 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};
```

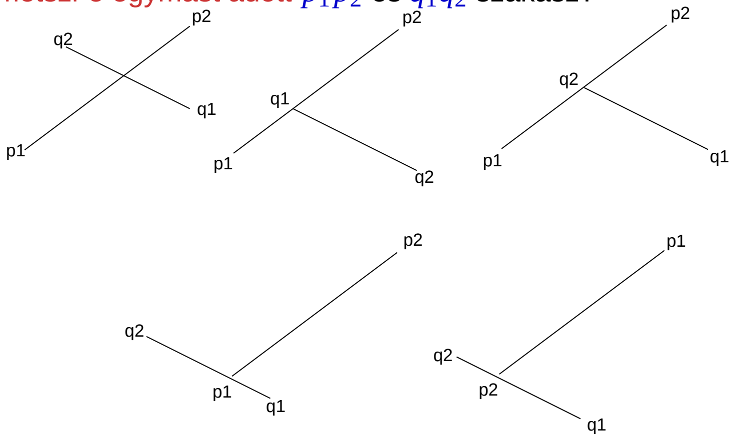
A algoritmus futási ideje legrosszabb esetben is a poligon csúcsainak n számával arányos, tehát $O(n)$.

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]);
```

4. Metsző szakaszpárok

Eldöntendő, hogy metszi-e egymást adott $\overline{p_1 p_2}$ és $\overline{q_1 q_2}$ szakasz?



9. ábra. Szakaszpárok metszésének öt lehetséges esete.

```
Function SzakaszParMetsz(P1,P2, Q1,Q2 : Pont):Boolean;
  Var fpq1,fpq2,fqp1,fqp2:integer;
Begin
  fpq1:=ForgasIrany(P1,P2, Q1);   fpq2:=ForgasIrany(P1,P2, Q2);
  fqp1:=ForgasIrany(Q1,Q2, P1);   fqp2:=ForgasIrany(Q1,Q2, P2);
  SzakaszParMetsz:=(fpq1*fpq2<0) and (fqp1*fqp2<0) or
  Szakaszon(P1,P2, Q1) or Szakaszon(P1,P2, Q2) or
  Szakaszon(Q1,Q2, P1) or Szakaszon(Q1,Q2, P2);
End;
```

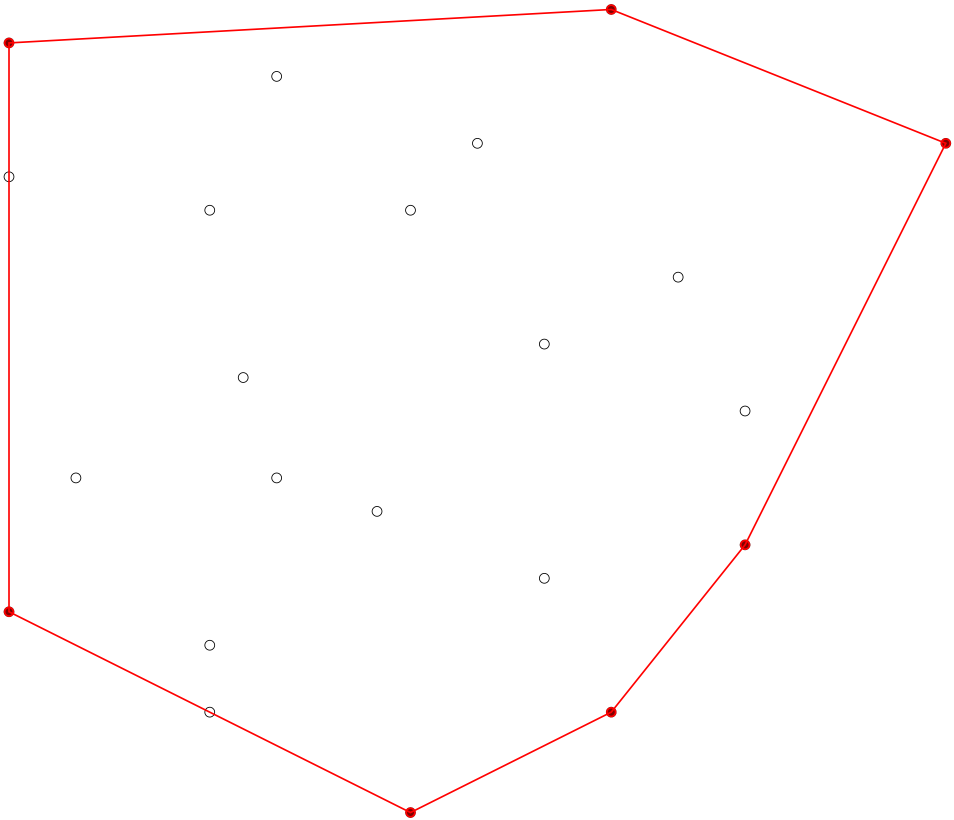
5. Ponthalmaz konvex burka

Definíció. Egy egyszerű (nem metsző) poligon konvex, ha bármely két belső pontját összekötő szakasz minden pontja a poligon belsejében, vagy határán van.

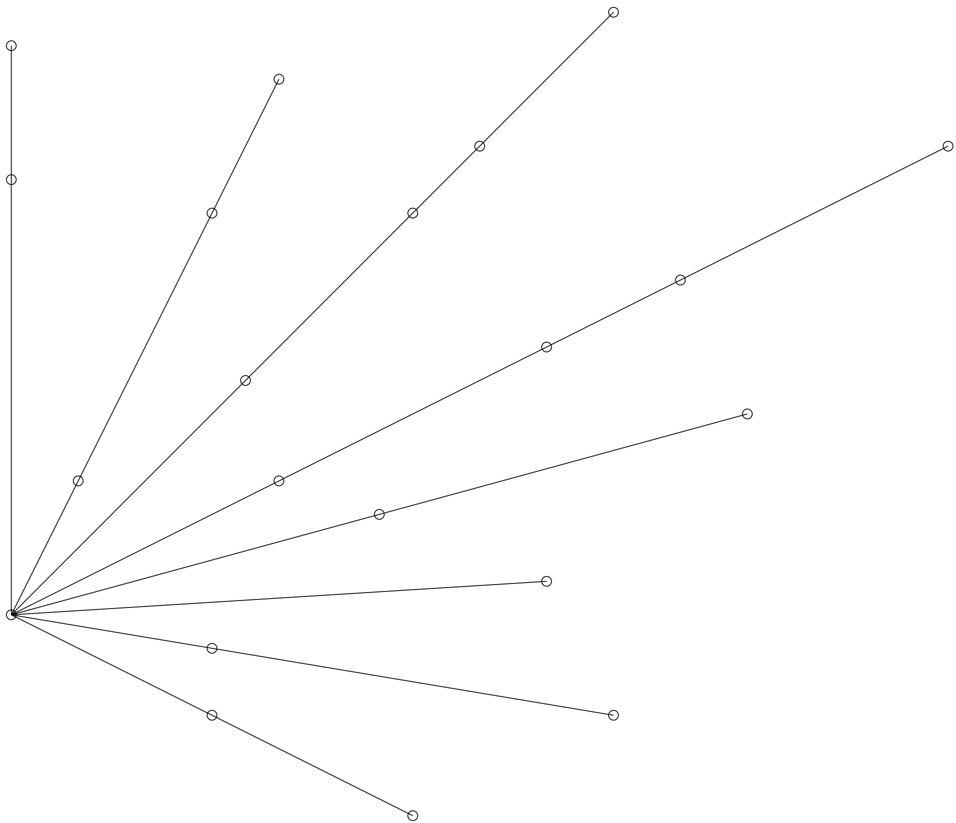
Definíció. Egy $P = \{p_1, \ldots, p_n\}$ ponthalmaz konvex burka az a legszűkebb Q konvex poligon, amely a ponthalmaz minden pontját tartalmazza.

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

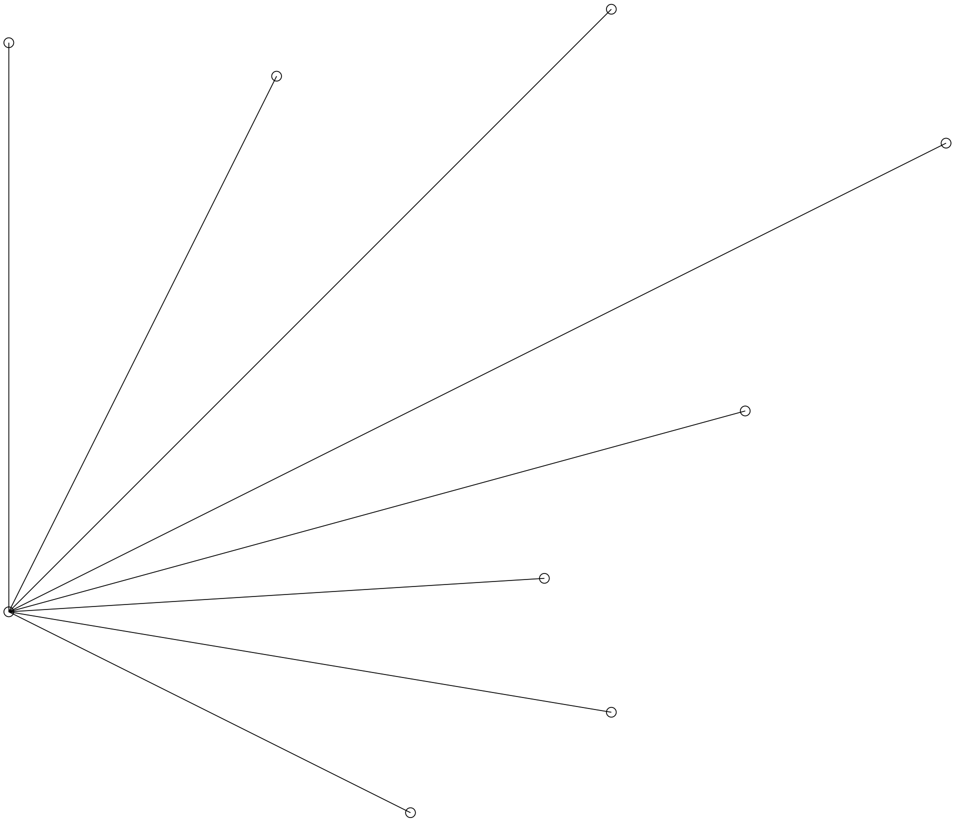
Az algoritmus futási ideje $O(n \lg n)$, ha a polárszög szerinti rendezést olyan algoritmussal valósítjuk meg amelynek futási ideje $O(n \lg n)$.



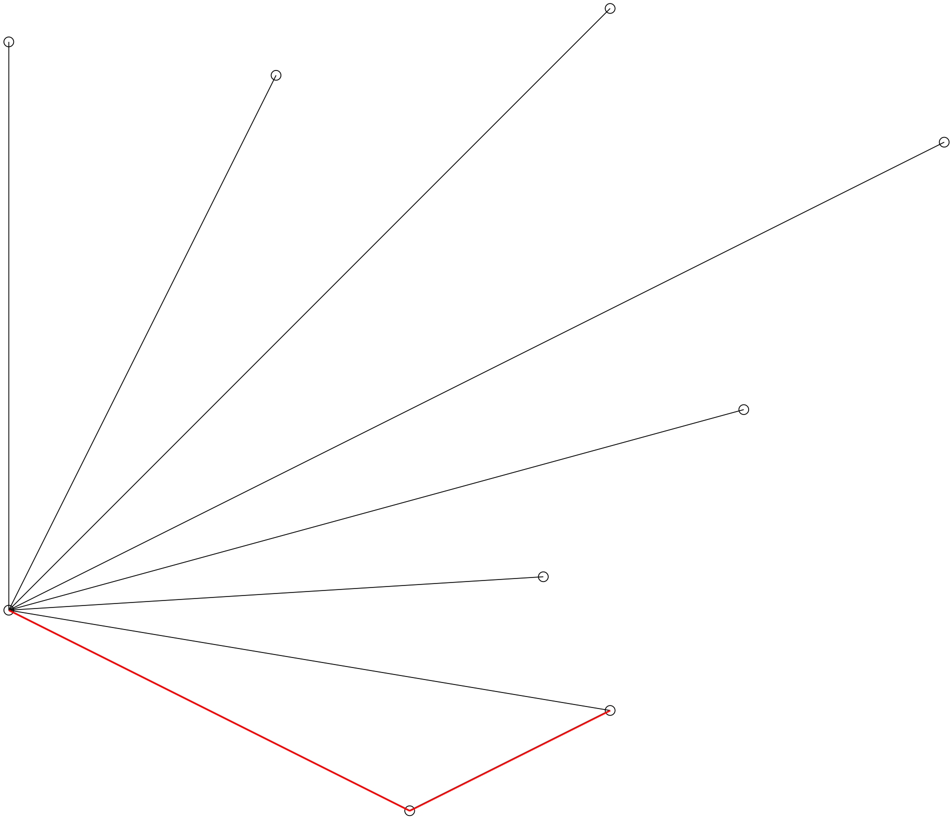
10. ábra. Ponthalmaz konvex burka.

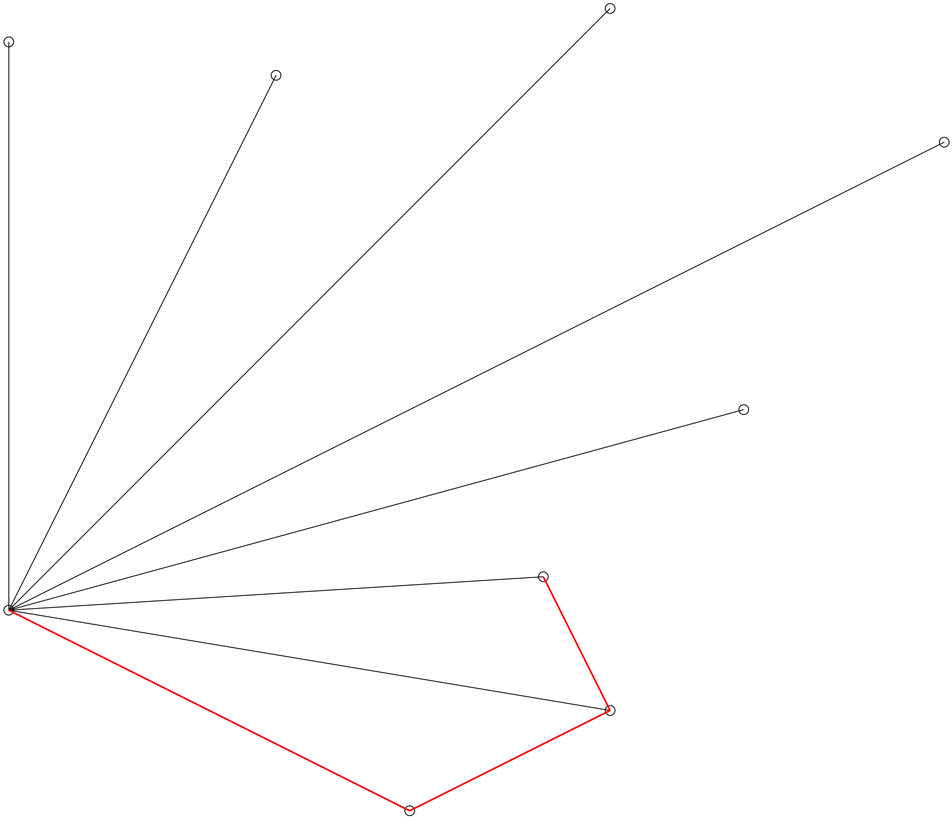


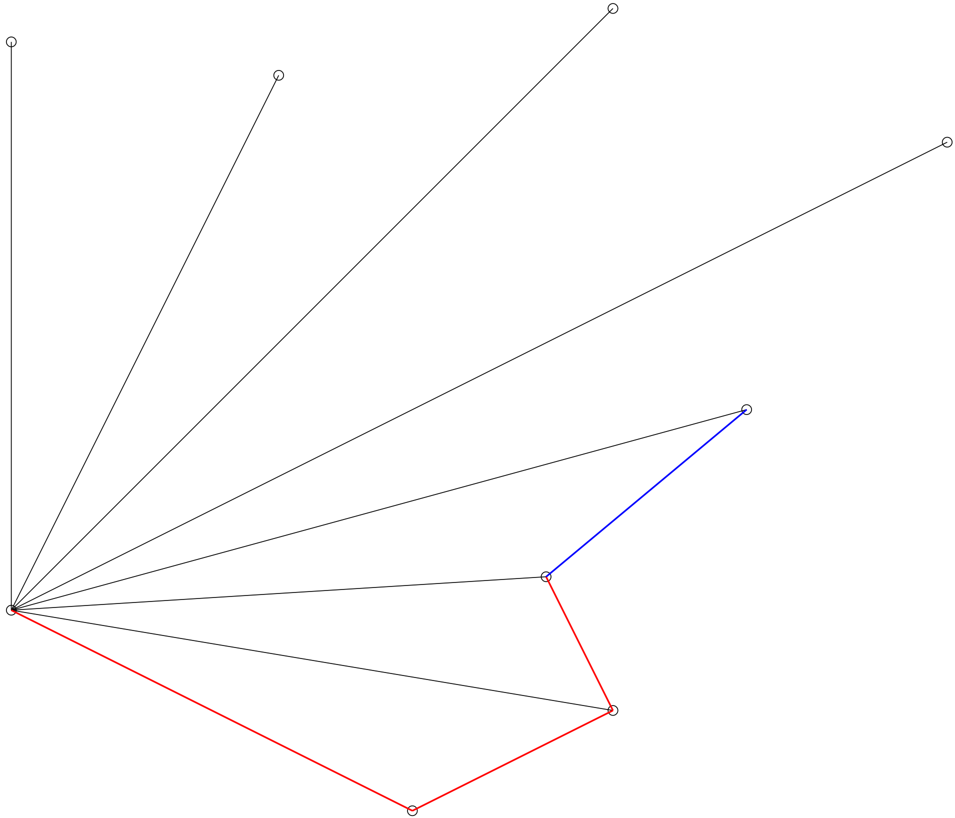
11. ábra. A pontok halmazának rendezése polárszög szerint.

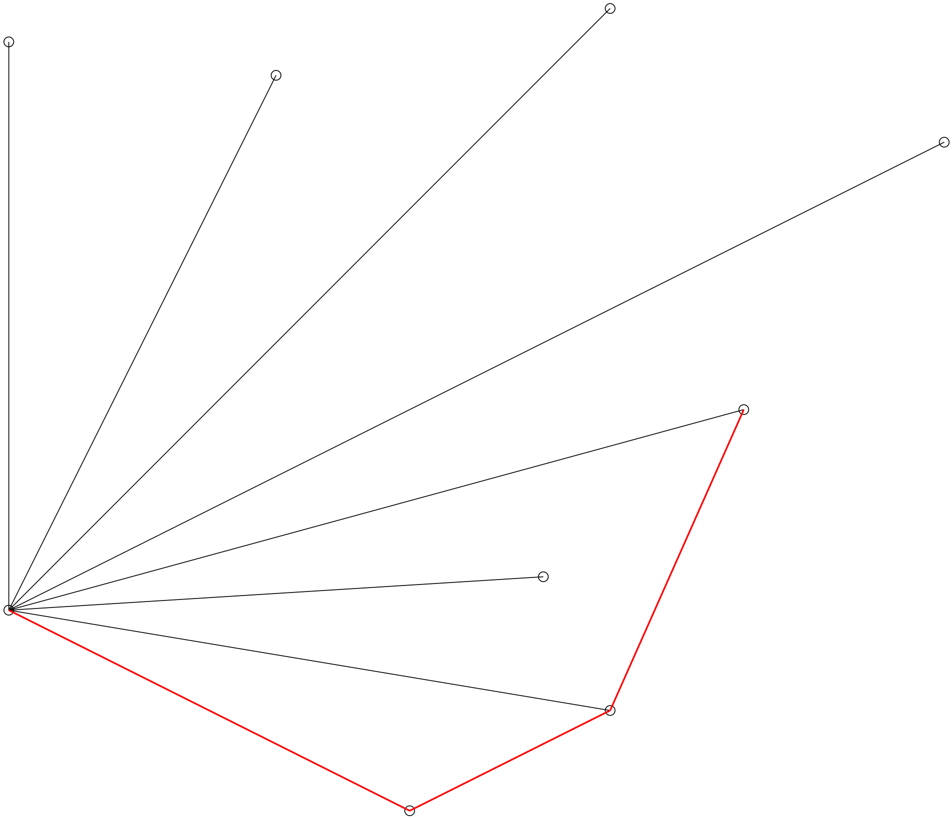


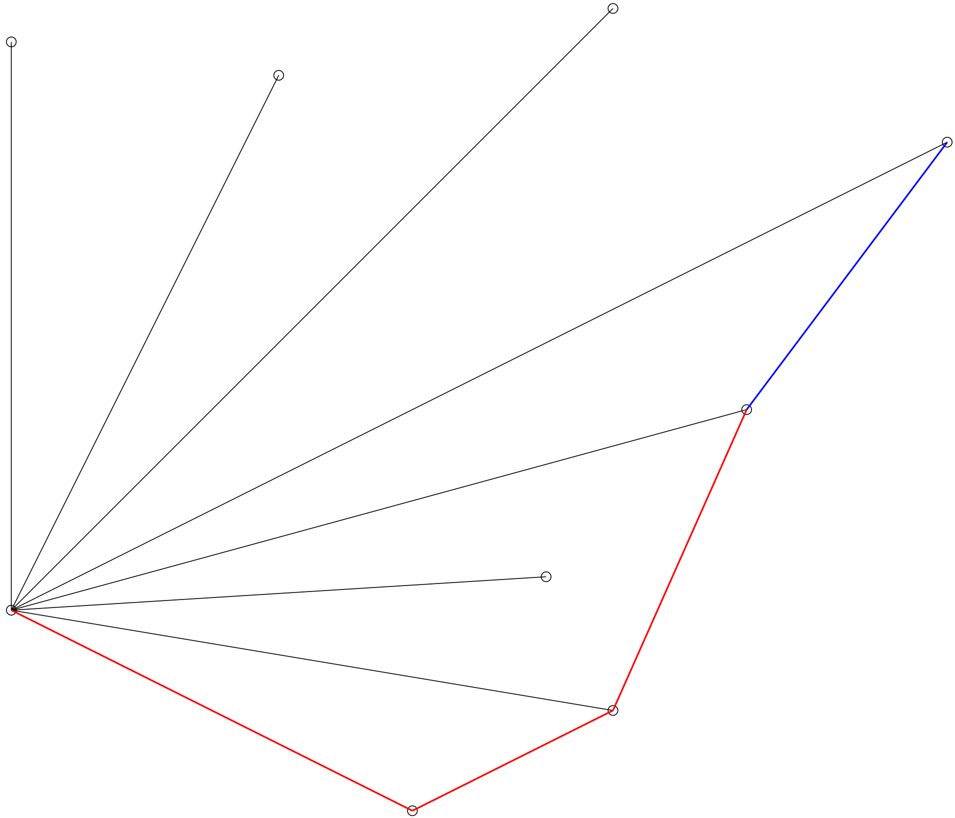
12. ábra. Minden egyenesen csak a legtávolabbi pont marad meg.

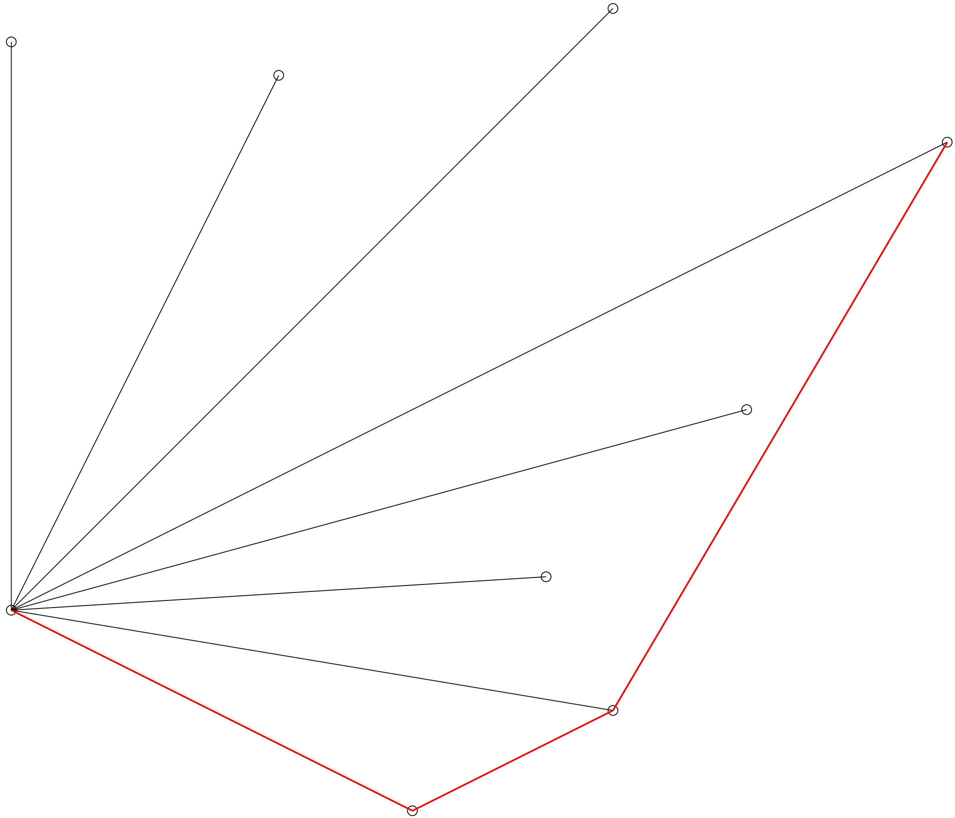


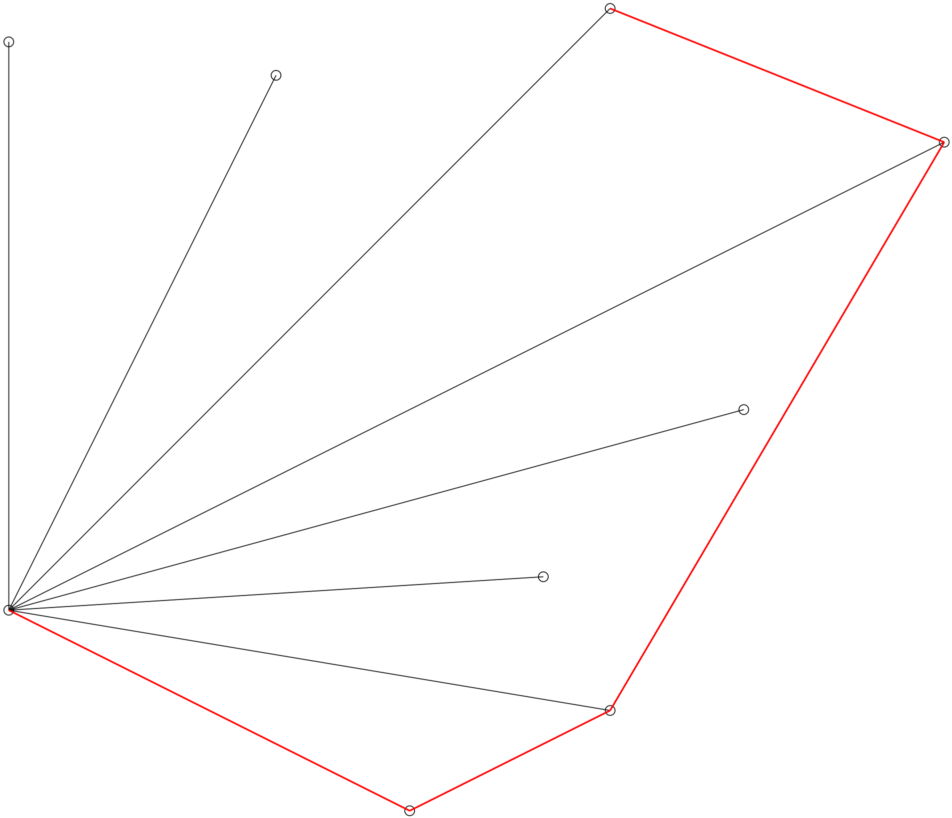


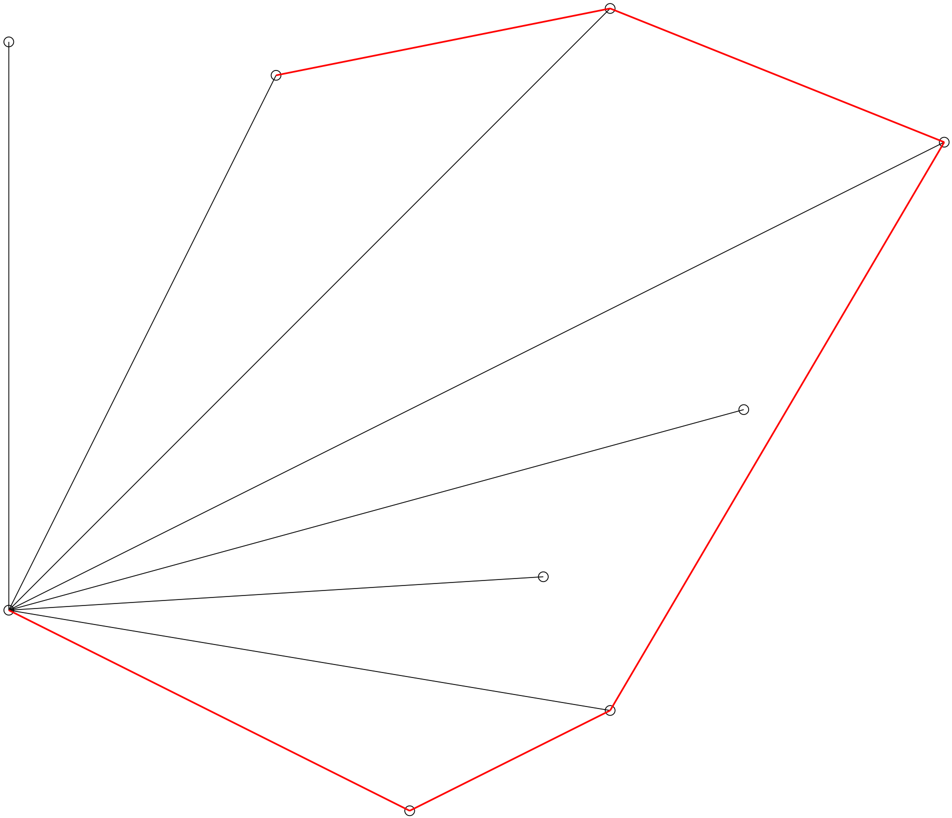


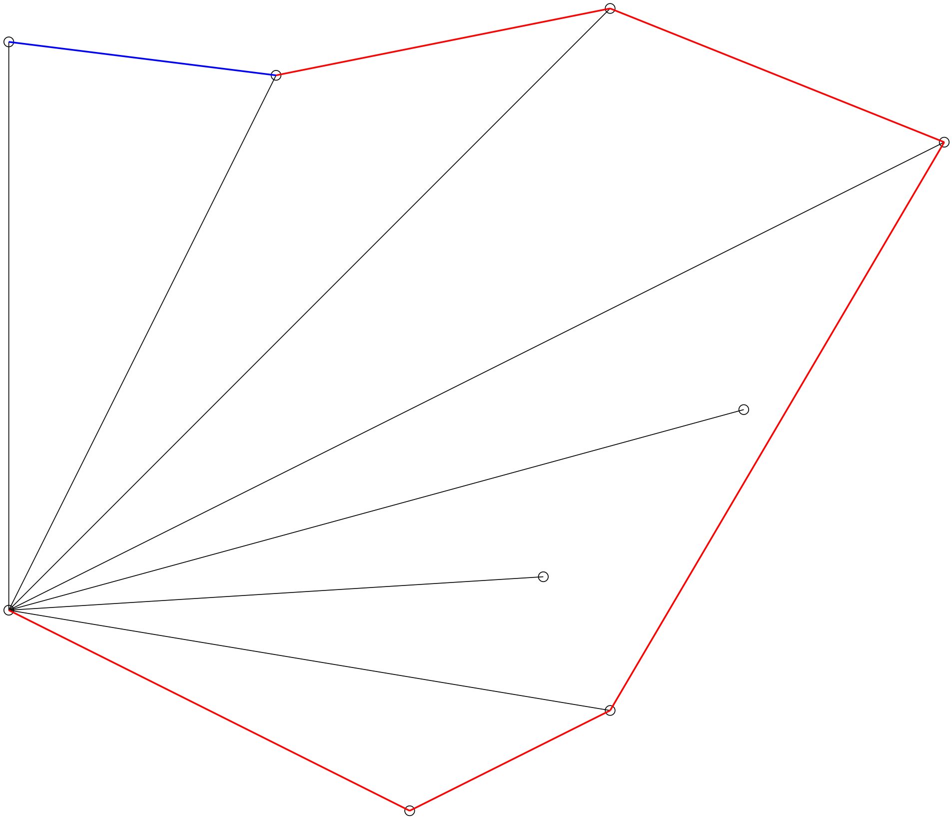


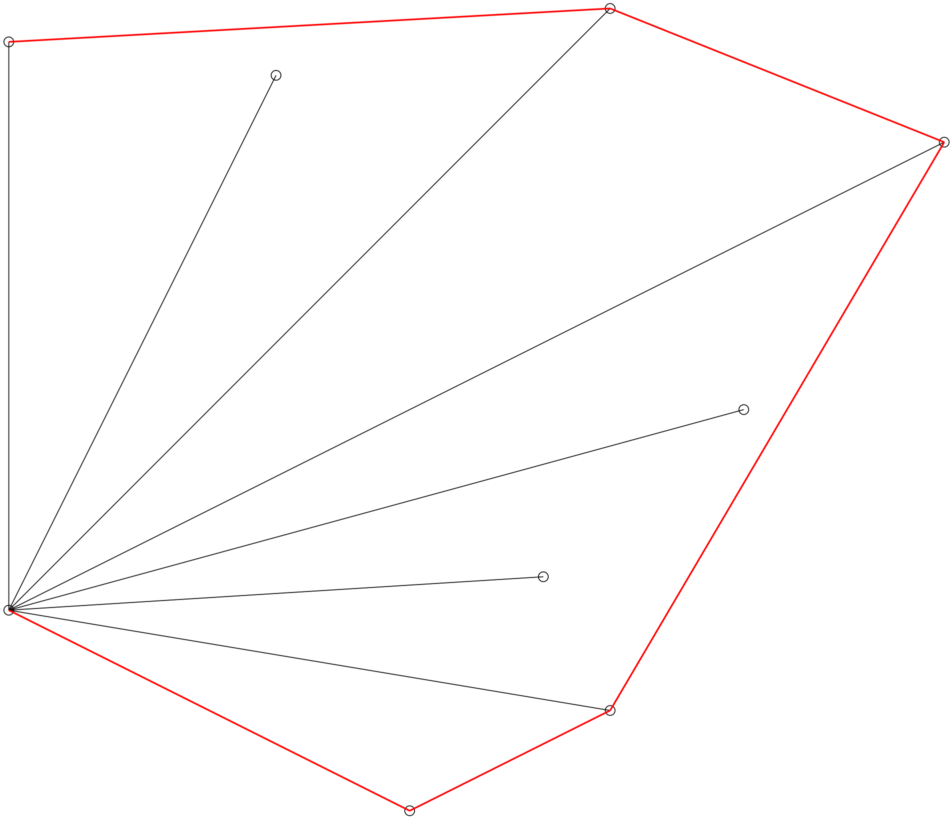


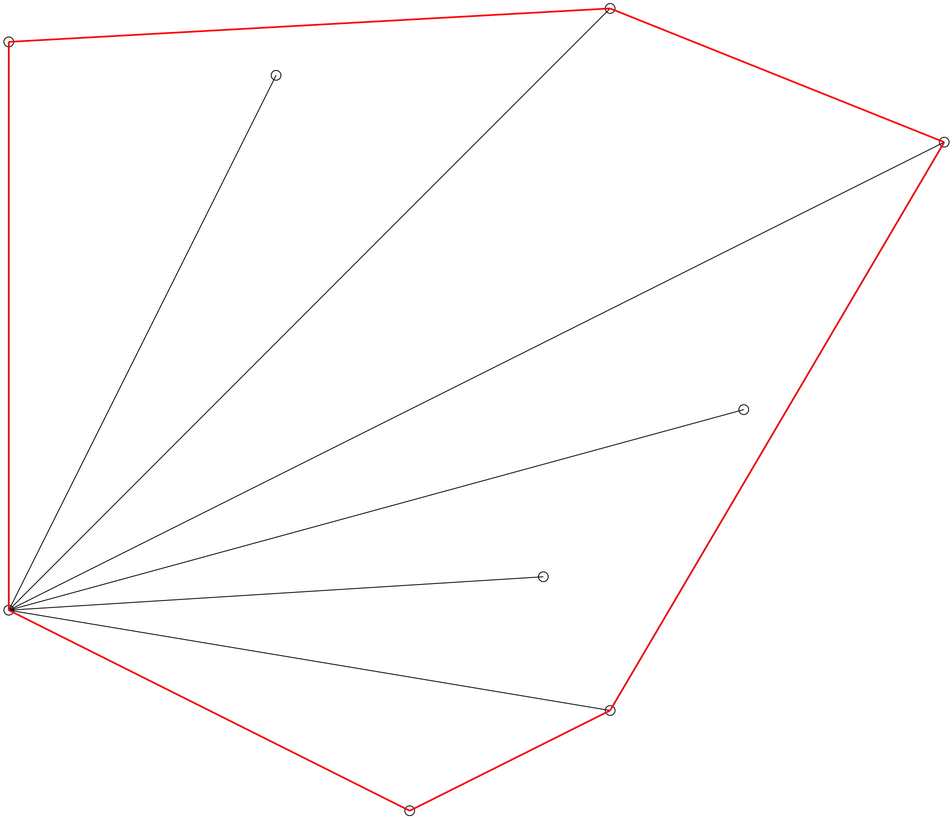












```
Function KonvexBurok(Var P:PontHalmaz; N:Longint):Longint;
{Bemenet: P[0],...,P[N-1],
 Kimenet: P[0],...,P[M-1] a konvex burok csúcsai }
Var
  M,i:Longint;
  xy:Pont;
Begin{KonvexBurok}
  SarokRendez(P,N); {A ponthalmaz bal-alsó sarokpont szerinti rendezése}
  M:=1;
  For i:=2 To N-1 Do Begin
    While (M>1)And(ForgasIrany(P[M-1], P[M], P[i])<=0) Do
      Dec(M);
    Inc(M);
    xy:=P[M]; P[M]:=P[i]; P[i]:=xy;
  End{for};
  KonvexBurok:=M+1;
End{KonvexBurok};
```

6. 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.

Bemeneti specifikáció

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.

Kimeneti specifikáció

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

paros.be

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

paros.ki

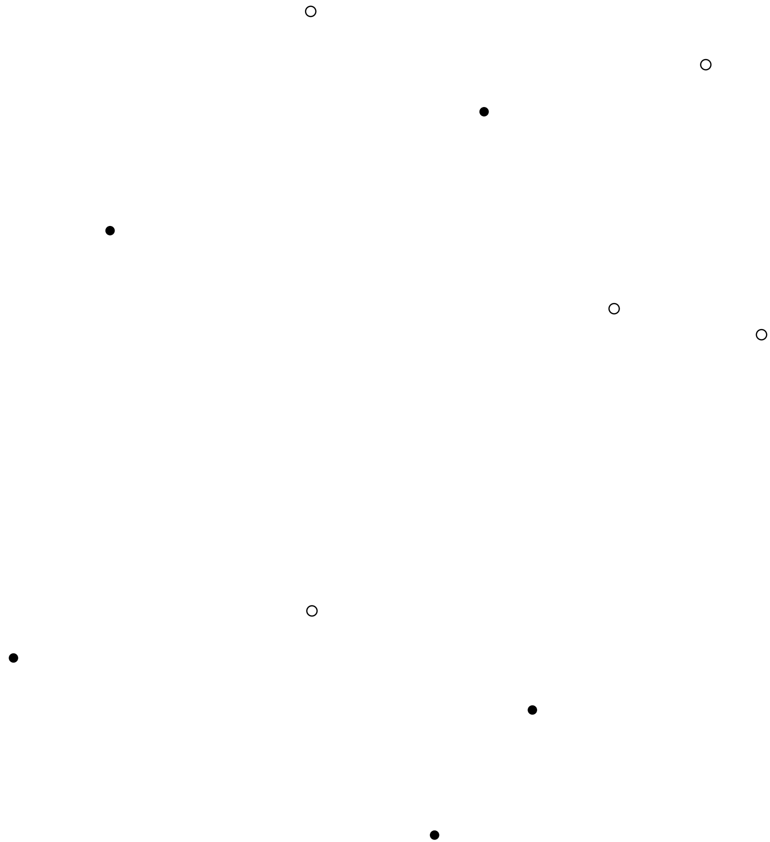
```
1 2
2 4
3 2
4 1
5 5
```

Megoldás

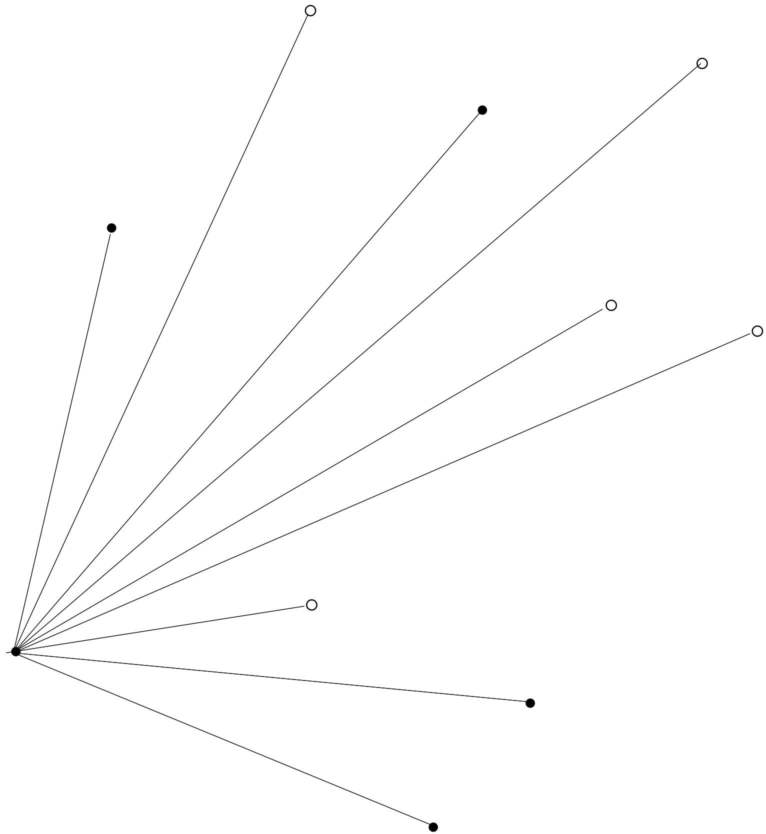
Legyen $P = \{p_1, \ldots, p_n, \ldots, p_{2n}\}$ a pontok halmaza.

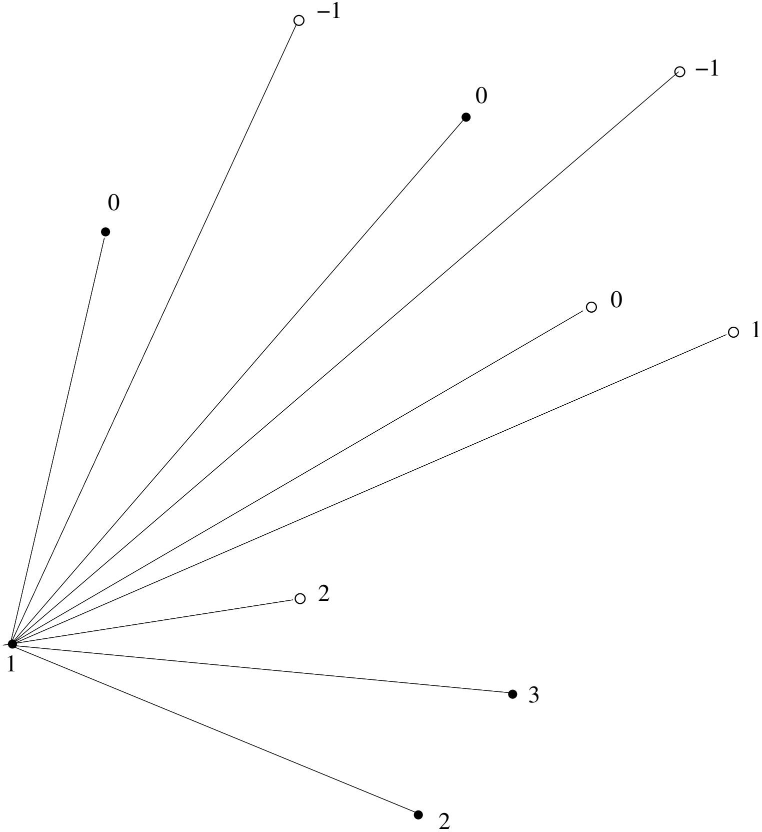
6.1. lemma. *Létezik olyan p_i és p_j különböző színű pontpár, hogy az $e(p_i, p_j)$ egyenes mindkét oldalán a fehér pontok száma megegyezik a fekete pontok számával.*

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. ■

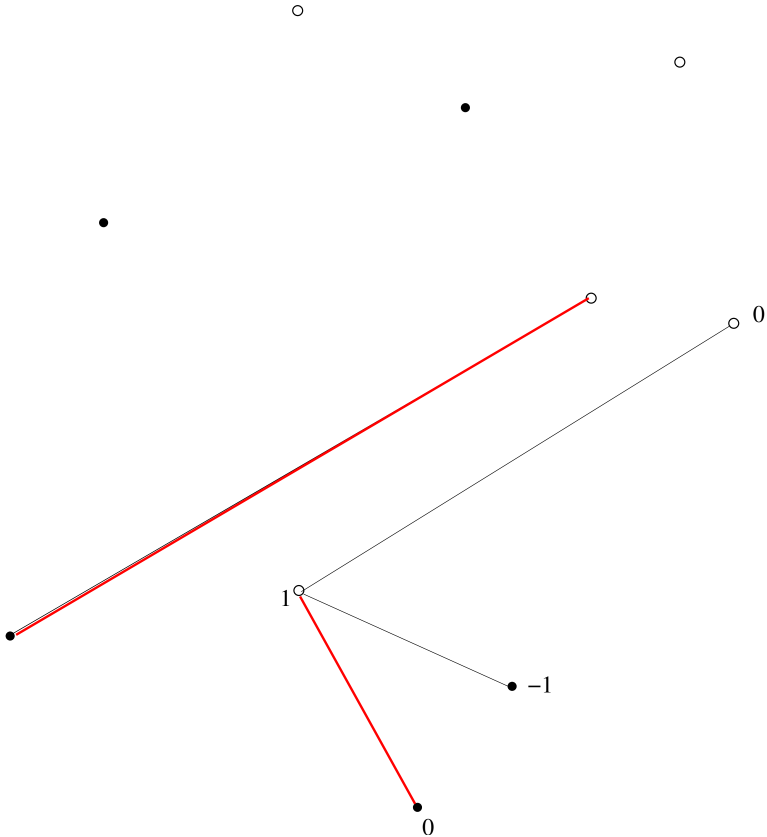


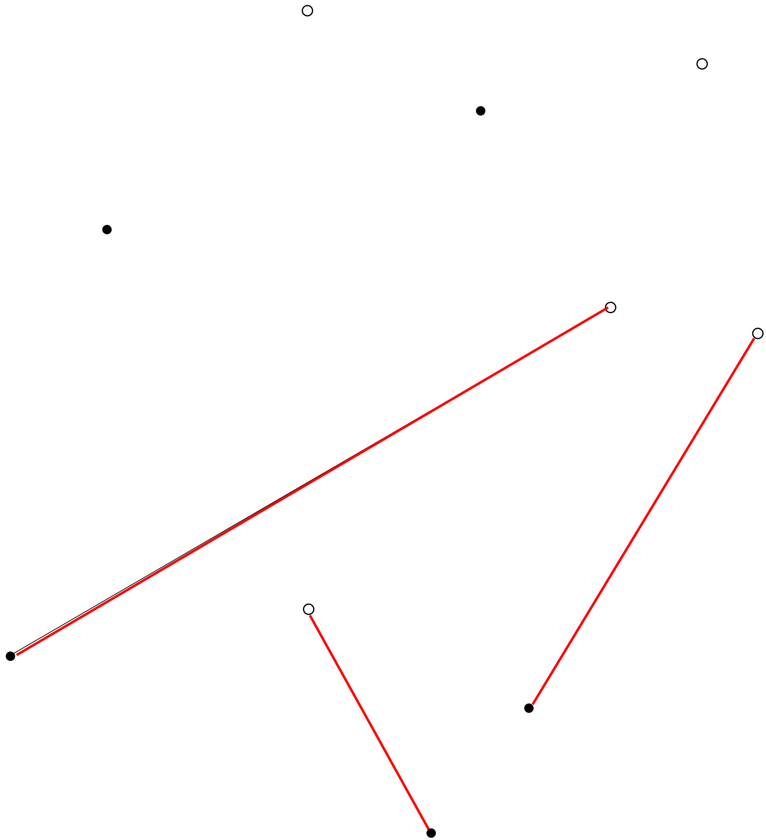
13. ábra. Párosítandó pontok

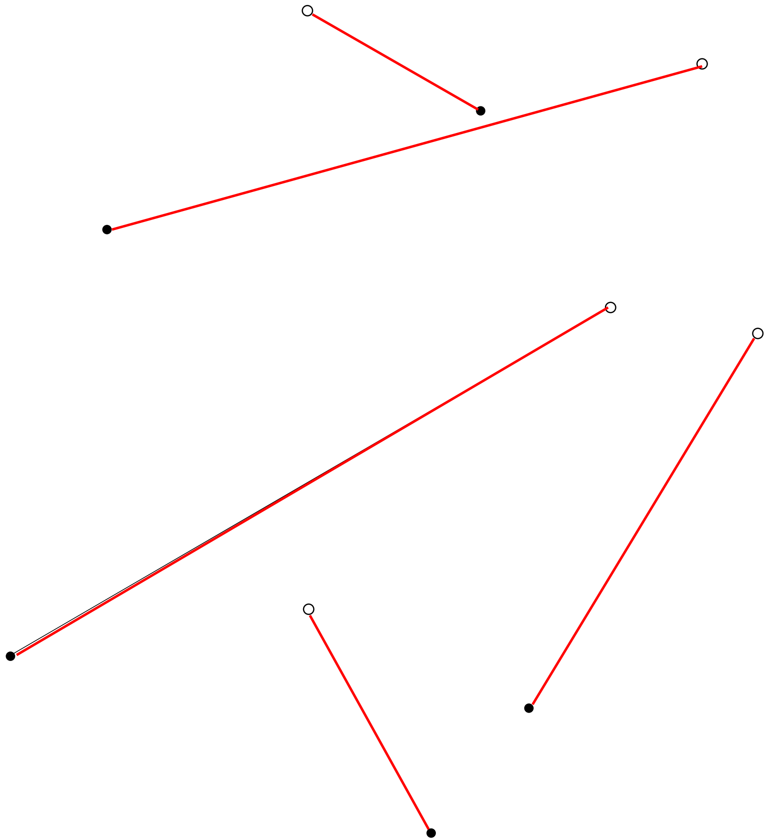




−1
−1
0
0
0
1
2
1
3
2


0
1
−1
0





```
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}
```

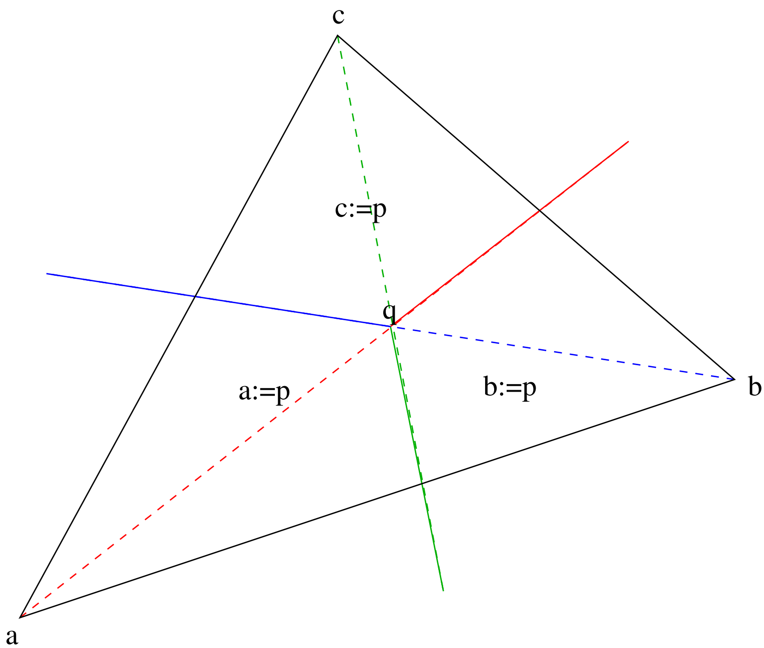
Az algoritmus futási ideje $O(n^2 \lg n)$.

7. 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!

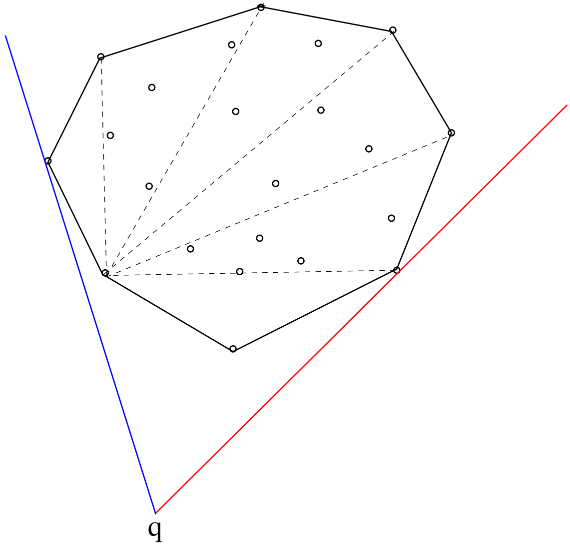
Megoldás.

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.



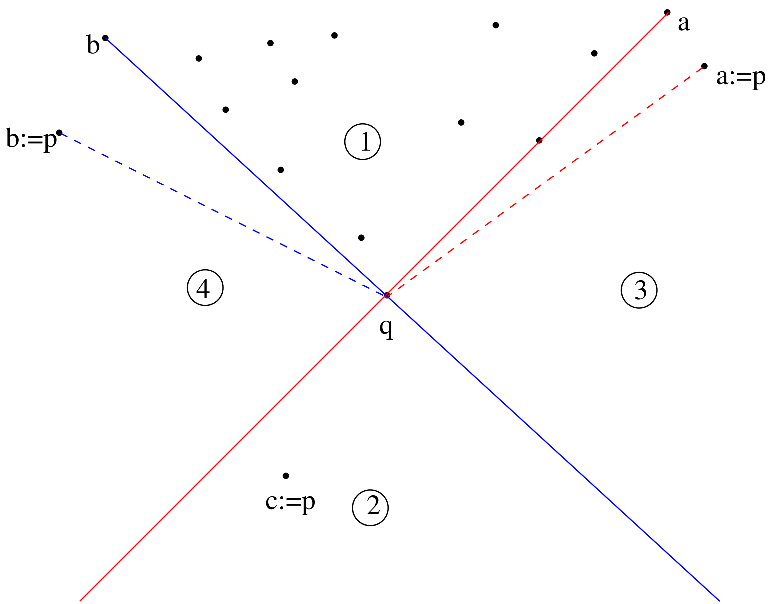
14. ábra. A háromszög finomítása. Minden $p \in P$ pontra, amely az $\triangle(a,b,c)$ háromszögbe, vagy oldalára esik, $q \in \triangle(a,b,p)$, vagy $q \in \triangle(b,c,p)$ vagy $q \in \triangle(c,a,p)$ teljesül.

2. Állítás. Akkor és csak akkor van megoldás, ha a q pont a P ponthalmaz konvex burkán belül, vagy oldalán van.



15. ábra.

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.



16. ábra. Az eddig vizsgált pontok a $\sphericalangle(b,q,a)$ 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 $ForgasIrany(q,b,p) > 0$): $b := p$.

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.

```
Program Kerites;
Const
  MaxN=100000;          {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
  P:Ponthalmaz;
  N:Longint;
  Q:Pont;
```

```
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:=(P1.x-P0.x)*(P2.y-P0.y)-(P2.x-P0.x)*(P1.y-P0.y);
  If KeresztSzorz < 0 Then
    ForgasIrany:=-1
  Else If KeresztSzorz>0 Then
    ForgasIrany:=1
  Else
    ForgasIrany:=0;
End{ForgasIrany};

Function Haromszogben(a,b,c,q:pont):boolean;
begin{Haromszogben}
  Haromszogben:=(ForgasIrany(a,b,q)>=0) and
                (ForgasIrany(b,c,q)>=0) and
                (ForgasIrany(c,a,q)>=0)
end{Haromszogben};
```

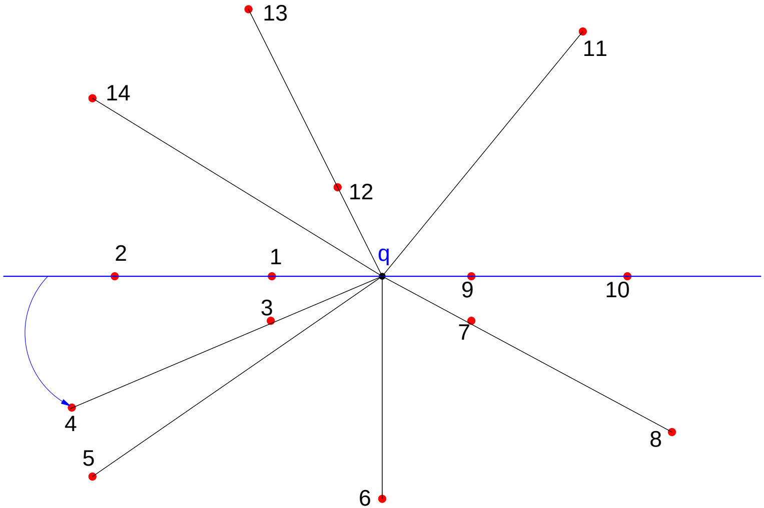
```
procedure BeOlvas;
var
  BeF:Text;
  i:integer;
begin{BeOlvas}
  assign(BeF,'kerites.be'); reset(BeF);
  readln(BeF,Q.x,Q.y);
  readln(BeF,N);
  for i:=1 to N do begin
    readln(Bef, P[i].x, P[i].y);
    P[i].az:=i;
  end;
  close(BeF);
end{BeOlvas};
```

```
 Var
  a,b,c:Pont;
  i:longint;
  firqap,firqbp:integer;
  firaqp,firbqp,fircqp:integer;
begin{program}
  Beolvas;
  a:=P[1];
  b.az:=0; c.az:=0;
  for i:=2 to N do
    if ForgasIrany(Q,a,P[i])<>0 then begin
      b:=P[i]; break;
    end;
  if b.az=0 then begin
    writeln('Nincs megoldás1'); halt;
  end;
  if ForgasIrany(Q,a,b)<0 then begin
    a:=b; b:=P[1];
  end;
```

```
{befoglaló háromszög keresés}
for i:=1 to N do begin
  firqap:=ForgasIrany(Q,a,P[i]);
  firqbp:=ForgasIrany(Q,b,P[i]);
  if (firqap>=0)and(firqbp<=0) then begin
    {semmi}
  end else if (firqap<=0)and(firqbp>=0) then begin
    c:=P[i]; break;
  end else if firqap<0 then begin
    a:=P[i];
  end else begin
    b:=P[i]
  end;
end{for i};
if c.az=0 then begin
   writeln('Nincs megoldás'); halt;
end;
```

```
  {Háromszög finomítás}
  for i:=1 to N do
    if (i<>a.az)and(i<>b.az)and(i<>c.az)and
       Haromszogben(a,b,c,P[i]) then begin
       firaqp:=ForgasIrany(a,q,P[i]);
       firbqp:=ForgasIrany(b,q,P[i]);
       fircqp:=ForgasIrany(c,q,P[i]);
       if firaqp>=0 then begin
         if firbqp<=0 then
           c:=P[i]
         else
           a:=P[i]
       end else begin
         if fircqp>=0 then
           b:=P[i]
         else
           a:=P[i]
       end;
    end;
  writeln(a.az,' ',b.az,' ',c.az);
end.
```

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



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

```
Function PolarRend(Q,P1,P2:Pont):Integer;
 {A Q ponthoz viszonyított polárszög szerinti rendezésben P1 és P2 viszonya}
Var  Fir:Integer; Begin
  If (P1.x=P2.x)and(P1.y=P2.y) Then begin
    PolarRend:=0;   Exit;  End;
  Fir:=Forgasirany(Q,P1,P2);
  Case Fir of
    -1: If (p1.y<=q.y) and (p2.y>q.y) Then
          PolarRend:=-1
        Else
          PolarRend:=1;
    0 : If Kozel(q,p1,p2)and Kozel(p2,p1,q) or
           Kozel(p1,q,p2)and Kozel(p2,q,p1)and(p1.y<=q.y)
        Then
          PolarRend:=-1
        Else
          PolarRend:=1;
    +1: If (p1.y<=q.y) or (p2.y>q.y) Then
          PolarRend:=-1
        Else
          PolarRend:=1;
  End{case};
End{PolarRend};
```

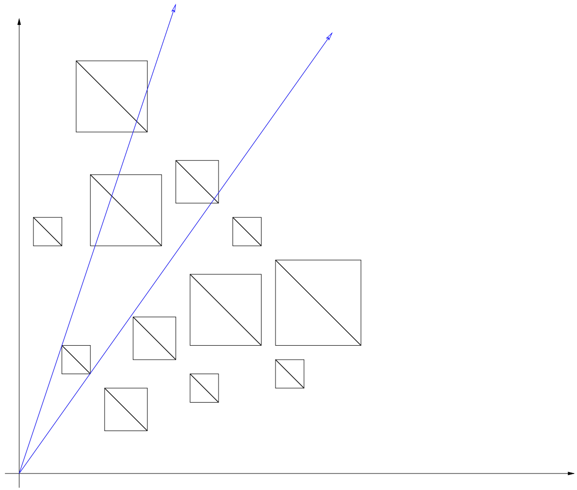
8. Feladat: Látható négyzetek

Adott a síkon n négyzet, amelyeknek oldalai párhuzamosak a koordinátarendszer tengelyeivel. Minden négyzet sarokpontjainak koordinátái pozitív egész számok. A négyzetek nem fedik és nem is érintik egymást, azaz oldalaiknak sincs közös pontja. Ki kell számítani, hogy hány olyan négyzet van, amelynek legalább egy pontja látható az origóból!

18. ábra. Négyzetek.

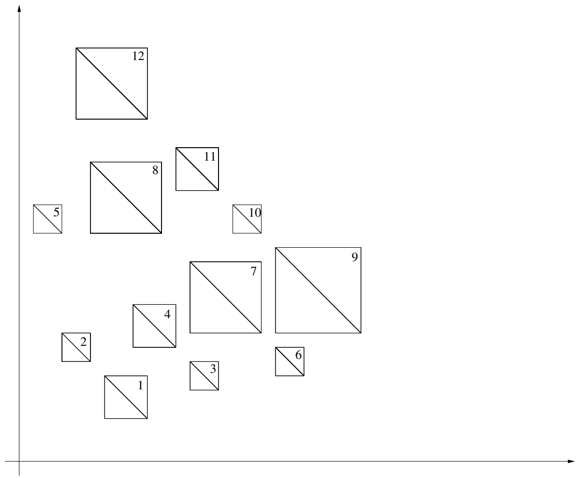
Megoldás.

Vegyük észre, hogy négyzet helyett a négyzet bal-felső és jobb alsó sarkát összekötő átlóval dolgozhatunk a megoldás során. Ugyanis, bármely négyzet akkor és csak akkor látható, ha az átlójának valamely pontja látható.



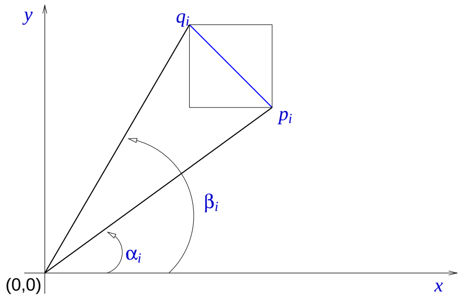
19. ábra. A négyzetek helyett átlók.

Első lépésként rendezzük a négyzeteket (átlós szakaszokat) a következőképpen. Az a négyzet akkor és csak akkor előzze meg a b négyzetet, ha a átlóján átmenő egyenes közelebb van az origóhoz, mint b átlóján átmenő egyenes, vagy
a és b átlóján átmenő egyenes megegyezik és a jobb alsó sarkának y-koordinátája kisebb, mint b jobb alsó sarkának y-koordinátája.



20. ábra. Négyzetek rendezése.

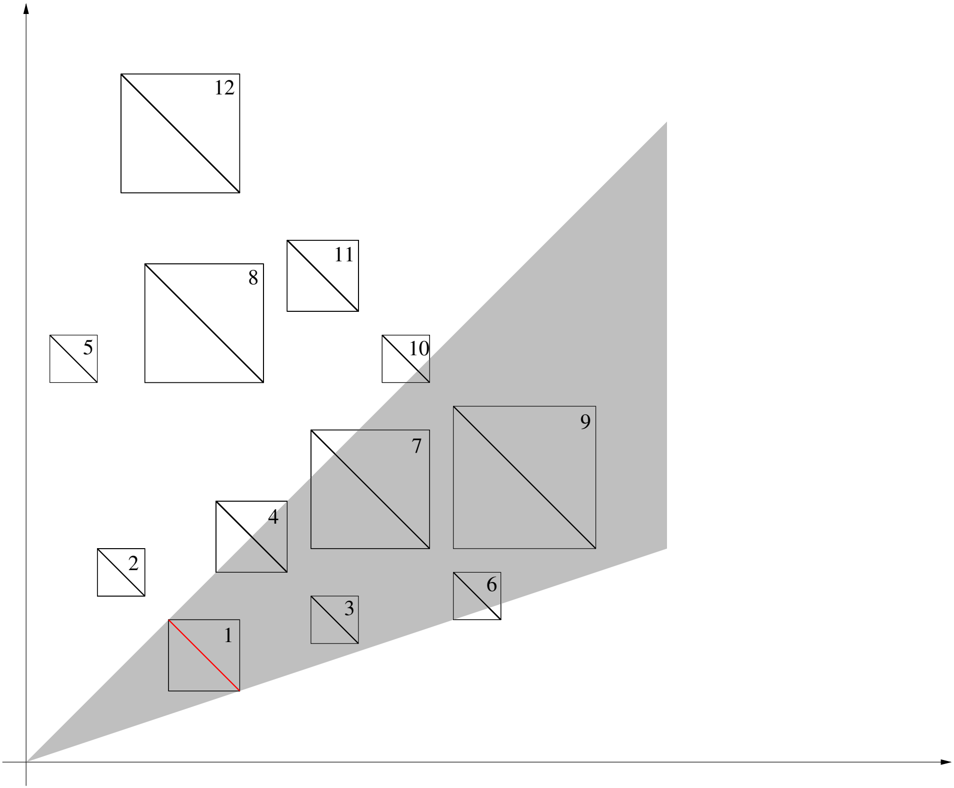
Jelölje p_i a rendezésben i-edik négyzet jobb alsó sarokpontját, q_i pedig a bal felső sarokpontját. Továbbá legyen α_i az x-tengely valamint az origón $(0,0)$ és a p_i ponton átmenő egyenesek által alkotott szög. Hasonlóan, β_i legyen az x-tengely valamint a $(0,0)$ origón és a p_i ponton átmenő egyenesek által alkotott szög. Tehát a rendezésben i-edik négyzet átlója által



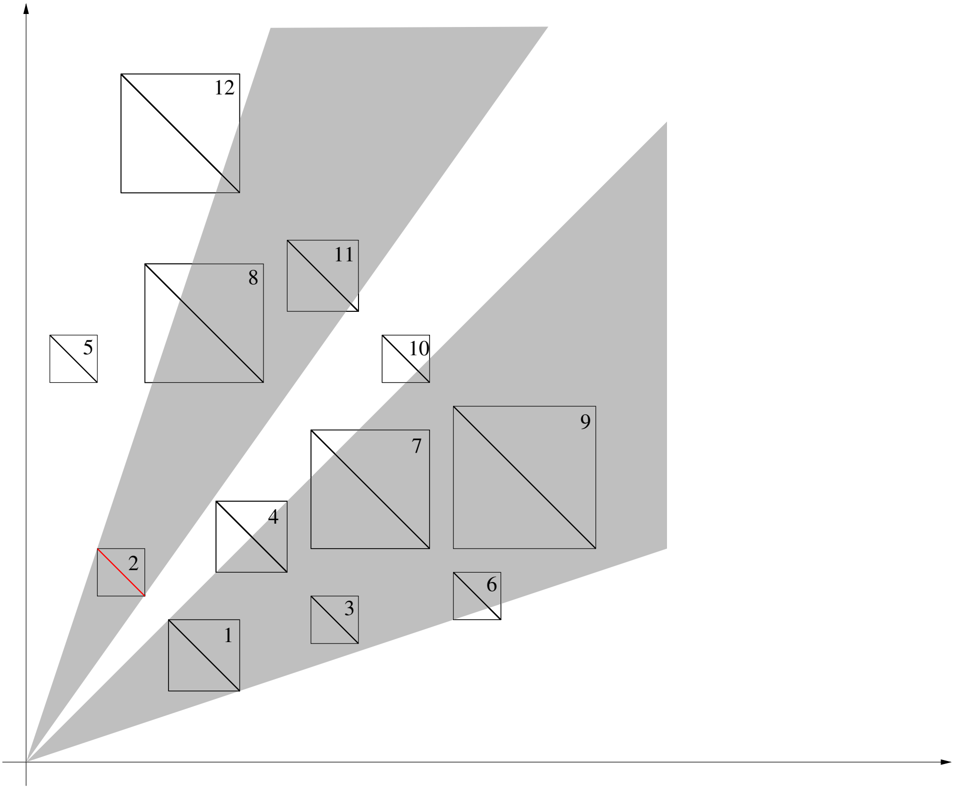
21. ábra. Négyzethez tartozó szögtartomány.

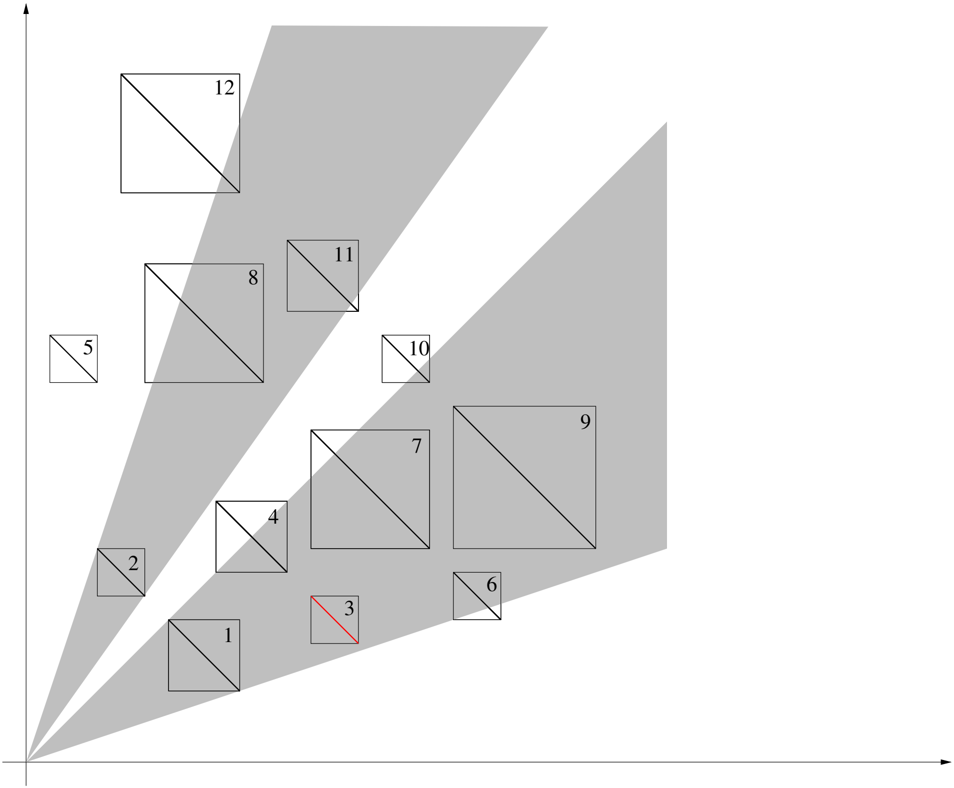
meghatározott szögtartomány az $[\alpha_i, \beta_i]$ intervallum.

Állítás. Az i-edik négyzet akkor és csak akkor nem látható, ha az $[\alpha_1, \beta_1], \ldots, [\alpha_{i-1}, \beta_{i-1}]$ szögtartományok egyesítése tartalmaz olyan $[\phi_j, \psi_j]$ szögtartományt, amely tartalmazza az $[\alpha_i, \beta_i]$ intervallumott, azaz $\phi_j \leq \alpha_i$ és $\beta_i \leq \psi_j$.

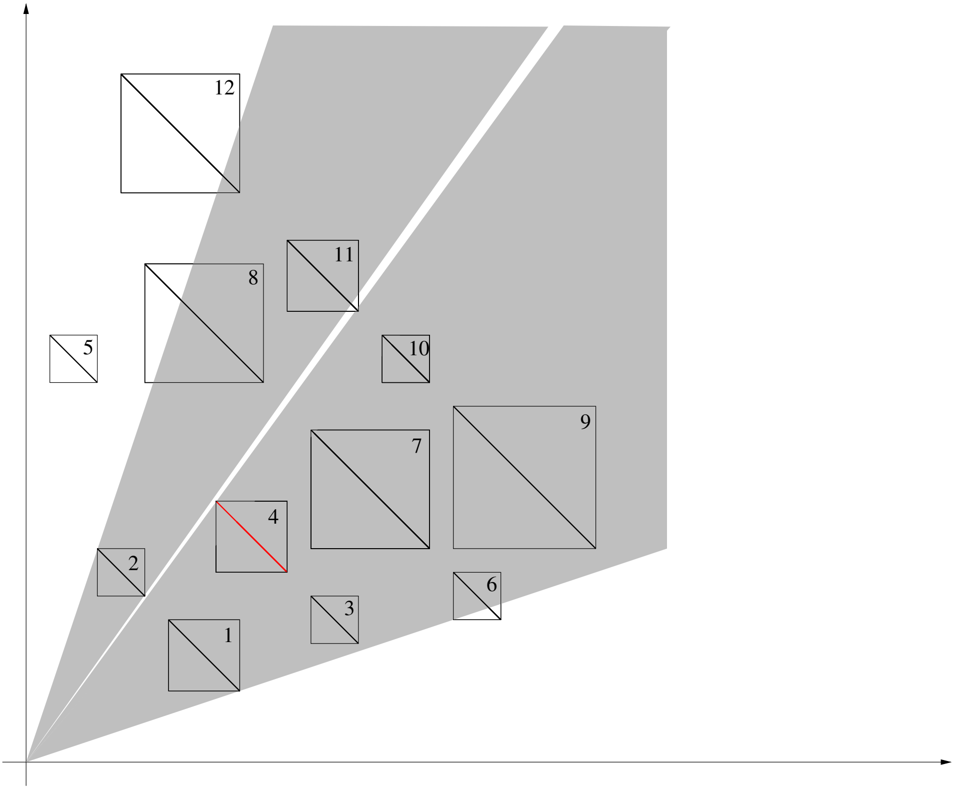


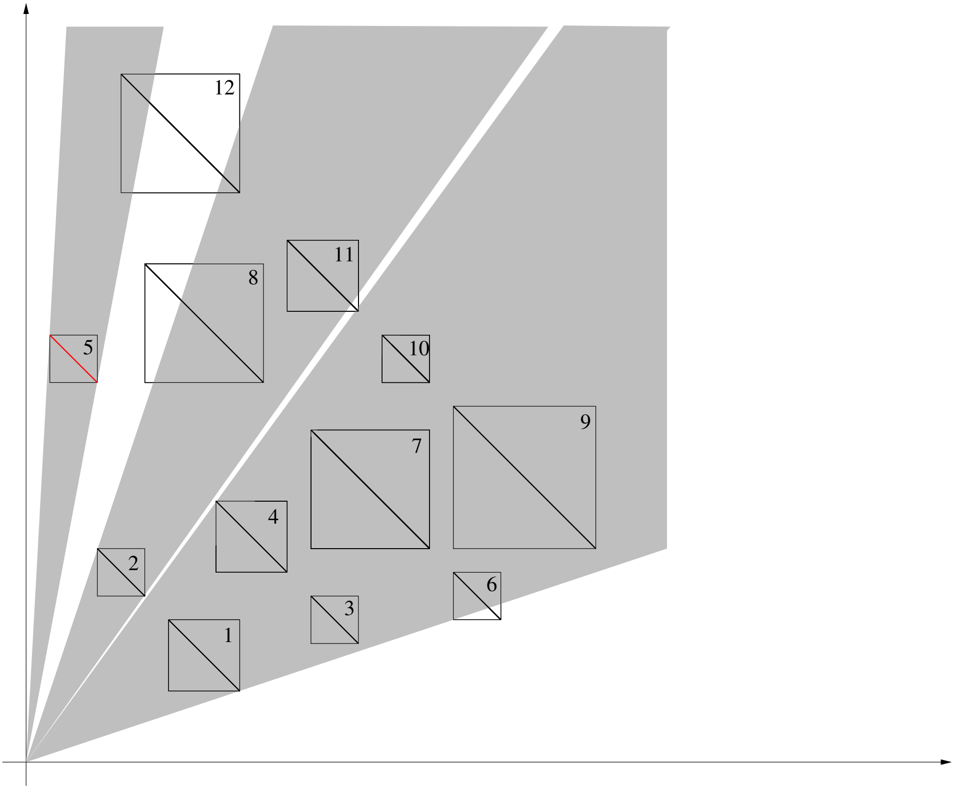
12
11
8
5
10
9
7
4
2
6
3
1


12
11
8
5
10
9
7
4
2
6
3
1

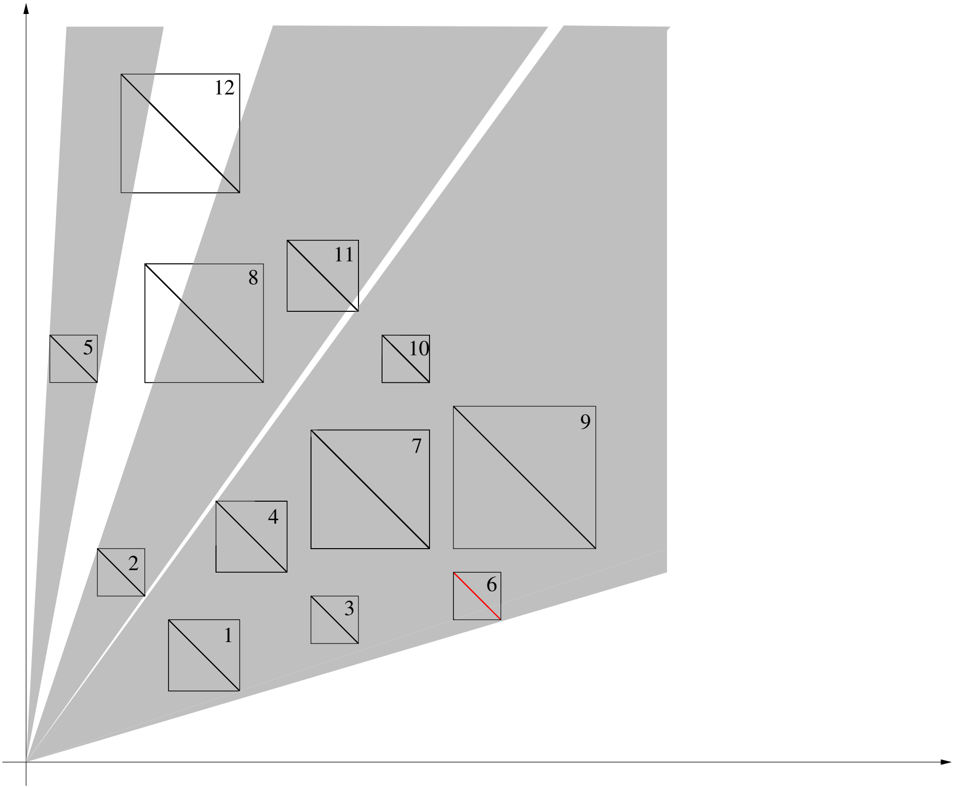


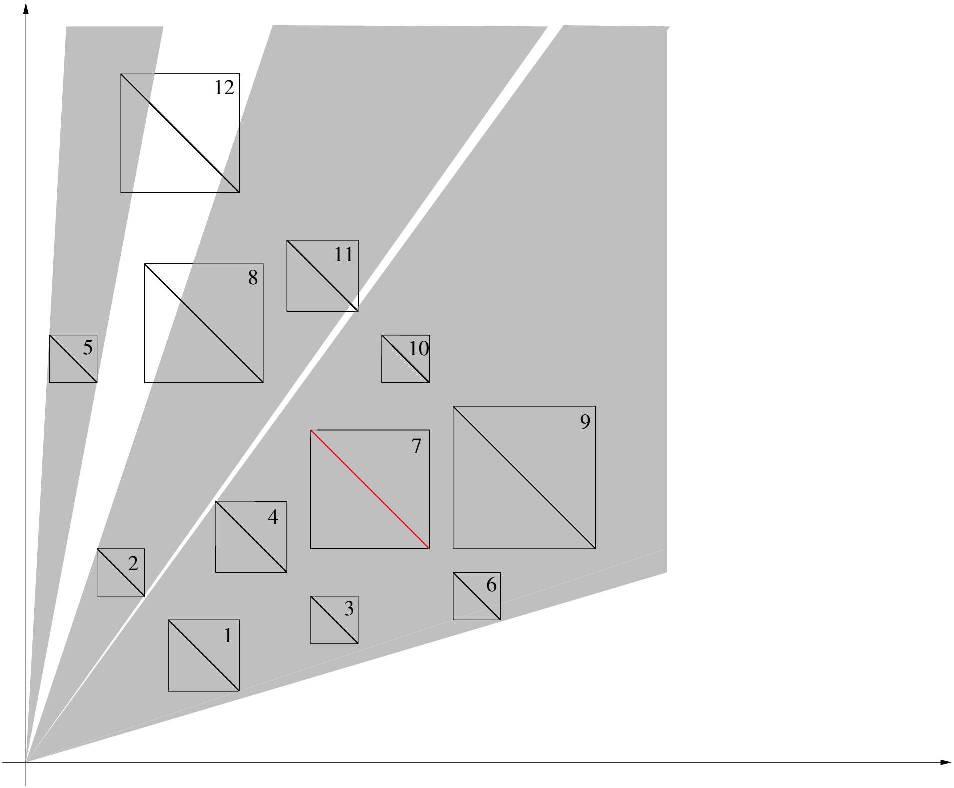
12
11
8
5
10
9
7
4
2
6
3
1

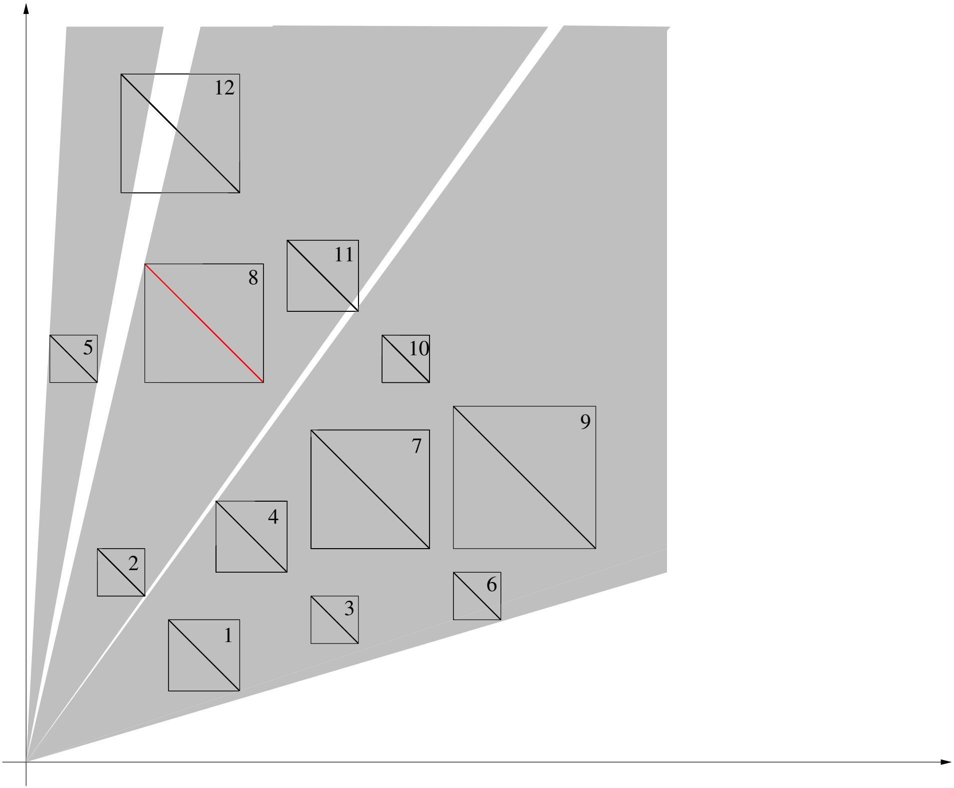

12
11
8
5
10
9
7
4
2
6
3
1

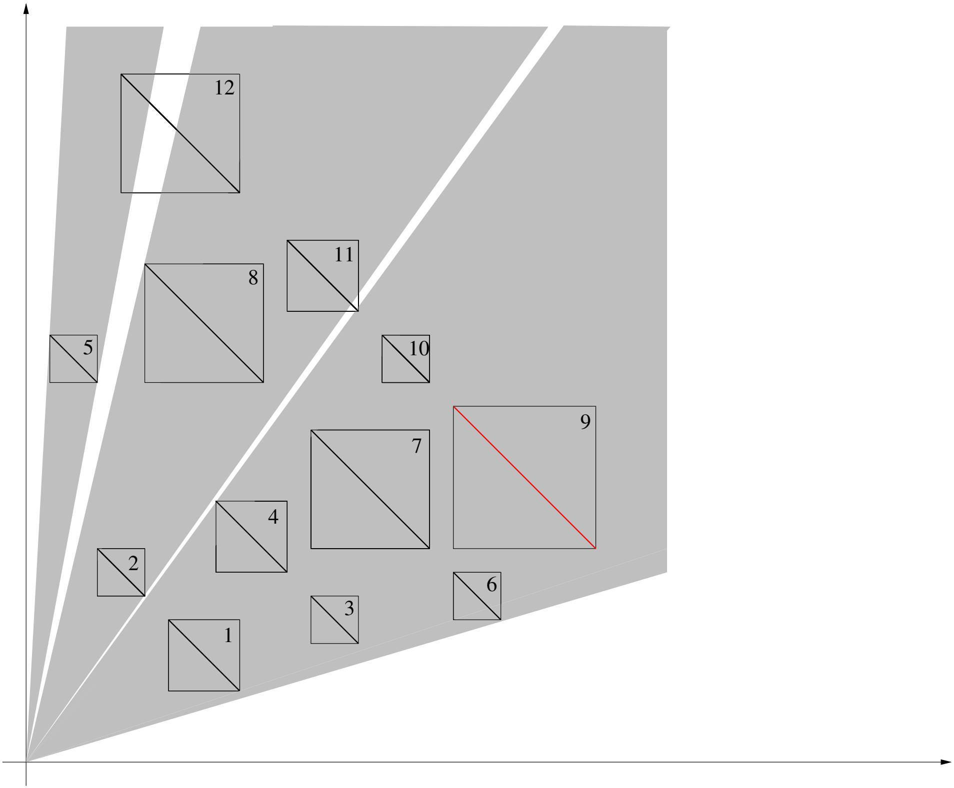


12
11
8
5
10
9
7
4
2
6
3
1

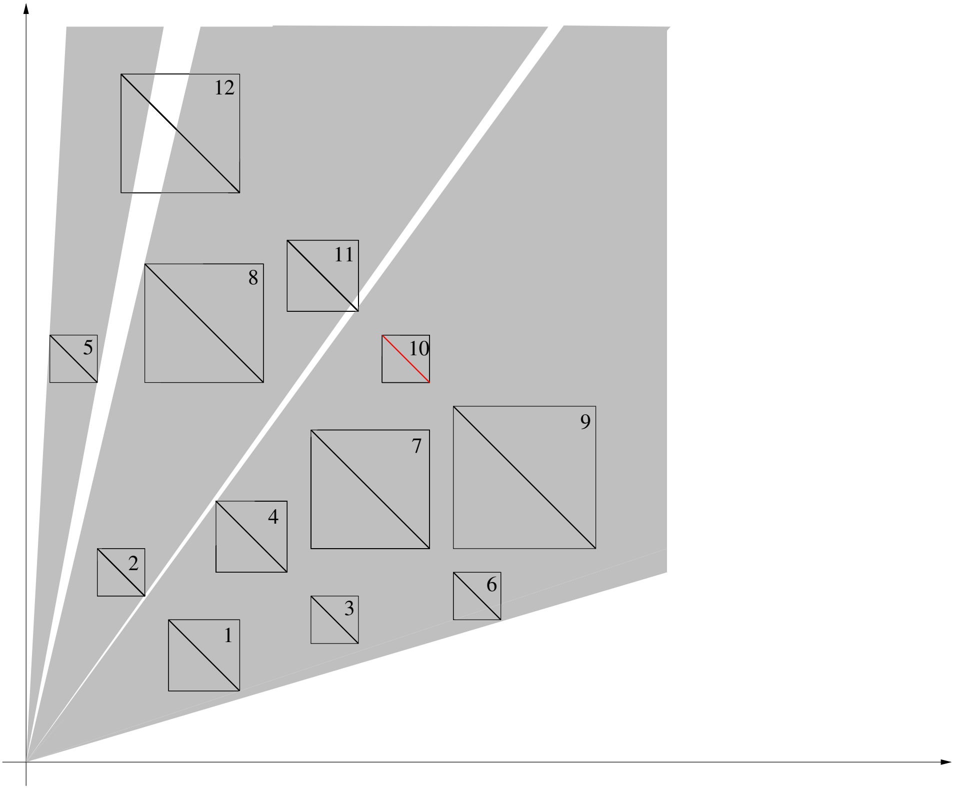

12
11
8
5
10
9
7
4
2
6
3
1

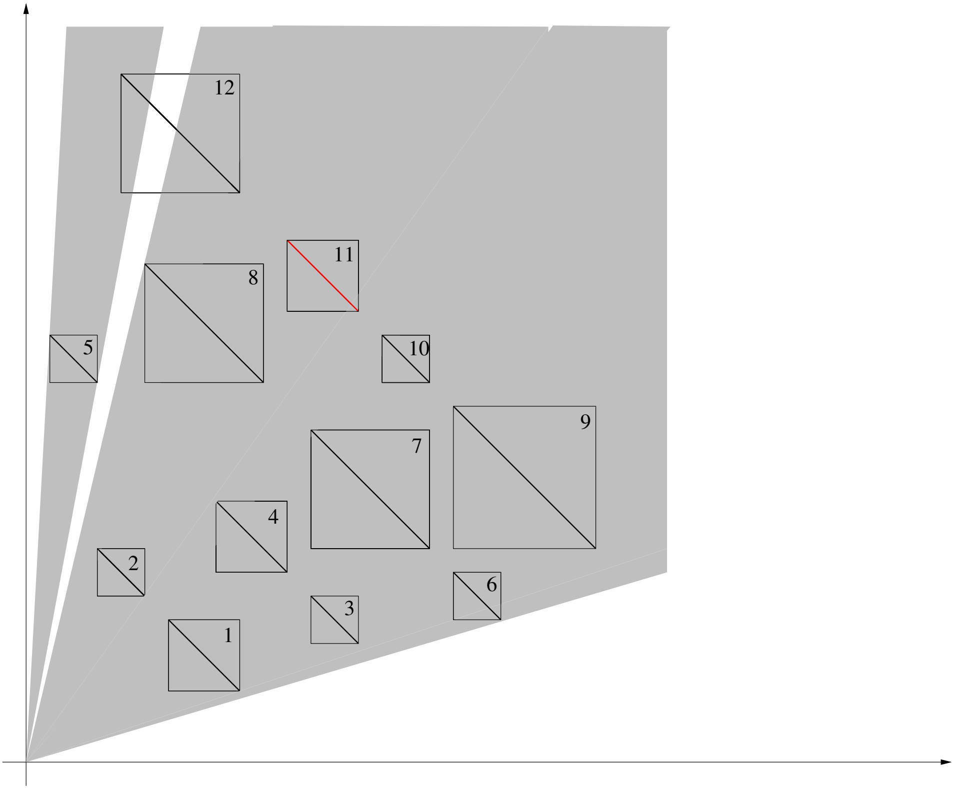

12
11
8
5
10
9
7
4
2
6
3
1

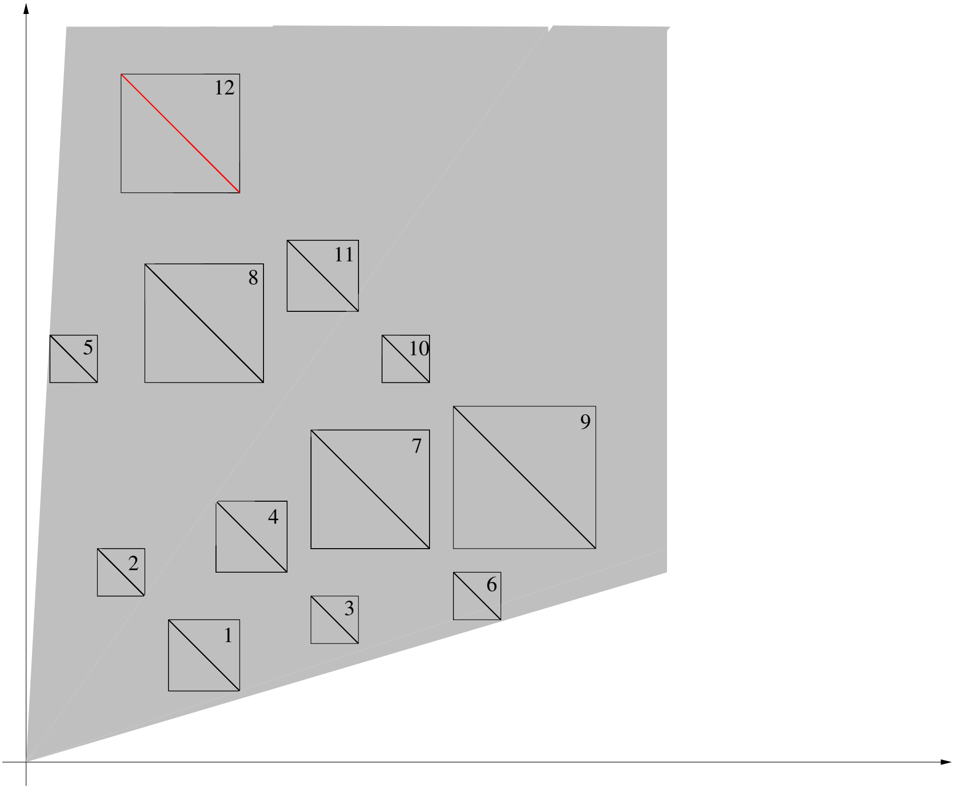

12
11
8
5
10
9
7
4
2
6
3
1



12
11
8
5
10
9
7
4
2
6
3
1


12
11
8
5
10
9
7
4
2
6
3
1


12
11
8
5
10
9
7
4
2
6
3
1

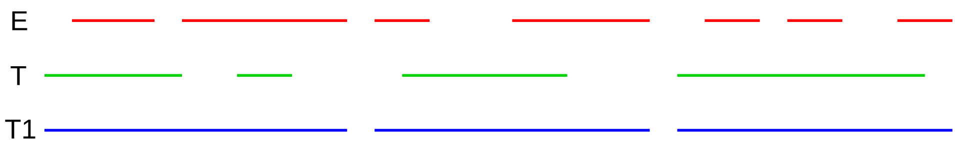

12
11
8
5
10
9
7
4
2
6
3
1

Tehát az elvi algoritmus a következő.

begin
A négyzetek rendezése;
$T := \emptyset$;
szaml:=0;
for i:=1 **to** n **do begin**
$\{T = \bigcup_{k=1}^{i-1}[\alpha_k, \beta_k]\}$
$M := \{[\phi, \psi] \in T : [\phi, \psi] \cap [\alpha_i, \beta_i] \neq \emptyset\}$;
if $M = \{[\phi, \psi]\}$ és $[\alpha_i, \beta_i] \subseteq [\phi, \psi]$ **then**
{az i-edik négyzet nem látható}
else begin{az i-edik négyzet látható}
Inc(szaml);
{$[\alpha_i, \beta_i]$ és az M-beli intervallumok egyesítése}
$\gamma := \min\{\phi : [\phi, \psi] \in M, \alpha_i\}$;
$\delta := \max\{\psi : [\phi, \psi] \in M, \beta_i\}$;
$T := T - M \cup \{[\gamma, \delta]\}$;
end;
end;
end;

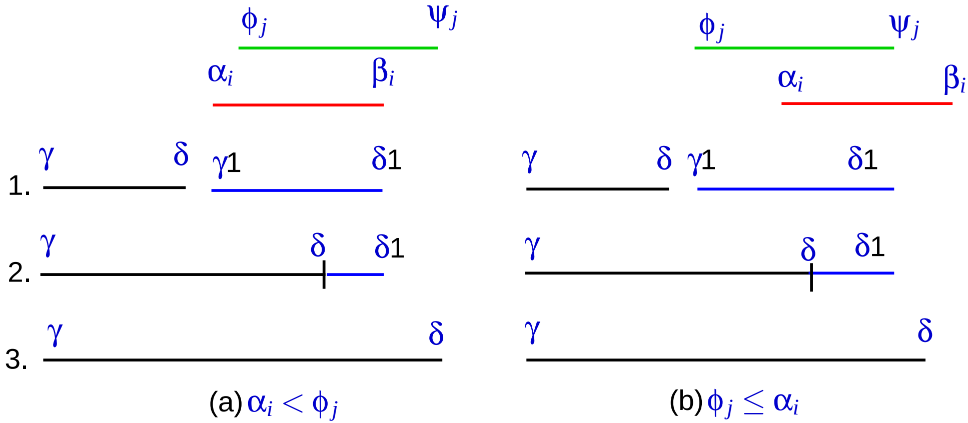
Az algoritmus futási ideje legroszabb esetben $O(\sum_{i=1}^{n}(i + \lg i))$ ha a szögtartományokat rendezetten tároljuk tömbben és bináris keresést alkalmazunk.

Gyorsabb algoritmust kaphatunk, ha az azonos átlós egyenesre eső szakaszokhoz tartozó szögtartományokat menetenként olvasztjuk össze, egy menetben az azonos átlós egyenesen lévőket olvasztjuk az addig kapott takaró szögtartományok rendezett sorozatával. Jelölje E



22. ábra. Két rendezett intervallum-sorozat összeolvasztása.

az aktuális átlós egyenesre eső szakaszok (szögtartományok) rendezett sorozatát, legyen T az eddig összeolvasztott szögtartományok rendezett sorozata. A két sorozat összeolvasztása a $T1$ rendezett sorozatot eredményezi. Majd áttérünk a következő átlós egyenesen lévő szakaszokra, de ekkor T szerepét a $T1$ veszi át; $T := T1; T1 := \emptyset$. Két rendezett intervallum-sorozat összeolvasztása a rendezett sorozatok összefésüléséhez hasonlóan végezhető. Az E és T-beli intervallumokat balról-jobbra haladva vesszük, mindig az lesz az aktuális összeolvasztandó, amelyiknek a bal végpontja kisebb. Jelölje $[\gamma, \delta]$ az utolsónak össszeolvasztott intervallumot, $[\alpha_i, \beta_i]$ az E sorozat aktuális elemét, $[\phi_j, \psi_j]$ pedig a T sorozat aktuális elemét.



23. ábra. Szögtartományok összeolvasztása. (a) eset: $\alpha_i < \phi_j$; 1. $\delta < \alpha_i$, 2. $\alpha_i \leq \delta < \beta_i$, 3. $\delta \geq \beta_i$. (b) eset: $\alpha_i \geq \phi_j$; 1. $\delta < \phi_j$, 2. $\phi_j \leq \delta < \psi_j$, 3. $\delta \geq \psi_j$.

```
begin
    A négyzetek rendezése;
    T := {[0,0],[∞,∞]}; {strázsa a sorozat elejére és végére}
    T1 := ∅;
    [γ,δ] := [0,0];
    szaml:=0; i := 1; j := 2;
    while True do begin
        if α_i < φ_j then begin
            if δ < φ_i then begin
                inc(szaml);
                tegyük [γ,δ]-t T1 végére;
                [γ,δ] := [α_i,β_i];
            end else begin
                if δ < β_i then begin
                    inc(szaml);
                    δ := β_i;
                end
            end;
            inc(i); if i > n then break;
            if α_i.x + α_i.y ≠ α_{i−1}.x + α_{i−1}.y then begin {áttérés a köv. átlóra}
                T maradékának átmásolása T1-be;
                T := T1; T1 := ∅; j := 2;
                [γ,δ] := [0,0];
            end;
```

```
        end else begin {αi ≥ φj}
            if δ < φj then begin
                tegyük [γ,δ]-t T1 végére;
                [γ,δ] := [φj,ψj];
            end else begin
                if δ < ψj then
                    δ := ψj;
            end;
            inc(j);
        end
    end {while};
end
```

Az algoritmus futási ideje legroszabb esetben $O(n^2)$.
Hogyan ábrázoljuk a szögeket? Látható, hogy az algoritmus szögekkel csak összehasonlítást végez. Ezért minden α szögeket ábrázolhatunk a sík egy pontjával, pontosabban bármely olyan p ponttal, amelyre az x tengely és a p ponton átmenő egyenes α szöget zár be. Tehát a p_1 pont által ábrázolt szög akkor és csak akkor kisebb, mint a p_2 pont által ábrázolt szög, ha $p_1.y * p_2.x < p_2.y * p_1.x$. Így az algoritmusban elkerülhető az osztás művelet, és a mivel a bemeneti adatok (a négyzetek sarokpontja és oldalhossza) egész számok, nem kell lebegőpontos aritmetikát használni.

```
Program Negyzetek; Const
  MaxN=1000;
Type
  Pont=Record
         x,y:Longint
       End;
  SzogTart=Record p,q:Pont End;
  Sorozat=Array[1..MaxN] Of SzogTart;
Var
  N:Word;          { a négyzetek száma }
  S:Sorozat;       { a szögtartományok (átlók) sorozata}
  Eredmeny:Word;   { a megoldás  }
```

```
Procedure Beolvas; {Global: N, S}
  Var
    BeF:Text;
    x,y,   { a négyzet bal alsó sarka }
    L,     { a négyzet oldalhossza }
    i:word;
  Begin
    Assign(BeF,'negyzetek.be'); Reset(BeF);
    ReadLn(BeF,N);

    For i:=1 To N Do Begin
      ReadLn(BeF,x,y,L);
      S[i].p.x:=x+L;   S[i].p.y:=y;
      S[i].q.x:=x; S[i].q.y:=y+L;
    End;
    Close(BeF);
  End {Beolvas};
```

```
Procedure KiIr; {Global: Eredmeny}
  Var
    KiF:Text;
  Begin
    Assign(KiF,'negyzetek.ki'); Rewrite(KiF);
    WriteLn(KiF,Eredmeny);
    Close(KiF);
  End{KiIr};
```

```
Procedure SzogRendez(Var S:Sorozat); {Global: S, N}
  Function Megelozi(i,j:Word):Boolean;
  Begin{Megelozi}
    Megelozi:=
    (S[i].p.x+S[i].p.y < S[j].p.x+S[j].p.y) Or
    (S[i].p.x+S[i].p.y = S[j].p.x+S[j].p.y) And (S[i].p.y < S[j].p.y)
    End{Megelozi};

  Function Feloszt( Bal,Jobb : Word): Word ;
  {Kimenet: S[bal..f-1] < S[f] < S[f+1..jobb]}
    Var
      Fe, i,f : Word;
    Begin
      Fe := Bal; {a feloszt˘ pont indexe}
      f:=Bal;
      For i:=Bal+1 To Jobb Do
        If Megelozi(i, Fe) Then Begin
            S[f]:=S[i];
            Inc(f);
            S[i]:=S[f]
        End;
      S[f]:=S[Fe];
      Feloszt:= f;
    End (* Feloszt *);
```

```
  Procedure Rendez(Bal,Jobb : Integer);
    Var
      f : Word;
    Begin
      f:= Feloszt(Bal, Jobb);
      If Bal<f-1 Then
        Rendez(Bal, f-1);
      If f+1<Jobb Then
        Rendez(f+1, Jobb)
    End (* Rendez *);
Begin{Rendez}
  Rendez(1, N)
End{SzogRendez};

Function Kisebb(P,Q:Pont):Boolean; {A szögek rendezési feltétele} Begin{Kise
  Kisebb:=P.y*Q.x<Q.y*P.x
End{Kisebb};
```

```
Function Szamol:Word; {Global: N, S }
  Var
    Szamlalo:Word;              { a látható négyzetek száma }
    T:Array[boolean] of Sorozat;{ az uj takaró szakaszok sorozata}
    M,i,j,ii,jj:Word;
    Nulla,Vegt:SzogTart;
    Regi,Uj:Boolean;
    G,D:Pont;
  Begin{Szamol}
    Nulla.p.x:=1; Nulla.p.y:=0;  Nulla.q:=Nulla.p;
    Vegt.p.x:=0; Vegt.p.y:=1;  Vegt.q:=Vegt.p;
    Regi:=False; Uj:=True;
    T[regi,1]:=Nulla; T[regi,2]:=Vegt;
    Szamlalo:=0;

    G:=Nulla.p; D:=Nulla.p;
    {az összevont szögtartomány: [G,D]}
    i:=1; {az aktuális szögtartomány}
    j:=2; {az aktuális takaró szögtartomány}
    M:=2; {a takaró szögtartományok száma}
    jj:=0;{az új takaró szögtartományok száma}
```

```
While True Do Begin
  If Kisebb(S[i].p, T[regi][j].p) Then Begin
    If Kisebb(D,S[i].p) Then Begin
      Inc(Szamlalo);
      Inc(jj);
      T[uj][jj].p:=G; T[uj][jj].q:=D;
      G:=S[i].p; D:=S[i].q;
    End Else Begin
      If Kisebb(D,S[i].q) Then Begin
        Inc(Szamlalo);
        D:=S[i].q;
      End;
    End;
    Inc(i); If i>N Then Break;
    If (S[i-1].p.x+S[i-1].p.y<S[i].p.x+S[i].p.y) Then Begin{uj atlo}
      Inc(jj);
      T[uj][jj].p:=G; T[uj][jj].q:=D;
      For ii:=j To M Do  T[uj,jj+ii-j+1]:=T[regi,ii];
      M:=jj+M-j+1;
      G:=Nulla.p; D:=Nulla.p;
      uj:=Regi; regi:=Not uj;
      j:=2;
      jj:=0;
    End;
```

```
      End Else Begin
        If Kisebb(D,T[regi,j].p) Then Begin
          Inc(jj);
          T[uj][jj].p:=G; T[uj][jj].q:=D;
          G:=T[regi,j].p; D:=T[regi,j].q;
        End Else Begin
          If Kisebb(D,T[regi,j].q) Then
            D:=T[regi,j].q;
        End;
        Inc(j);
      End;
    End{While};

    Szamol:=Szamlalo;
  End{Szamol};

Begin{Program}
  Beolvas;
  SzogRendez(S);
  Eredmeny:=Szamol;
  KiIr;
End.
```