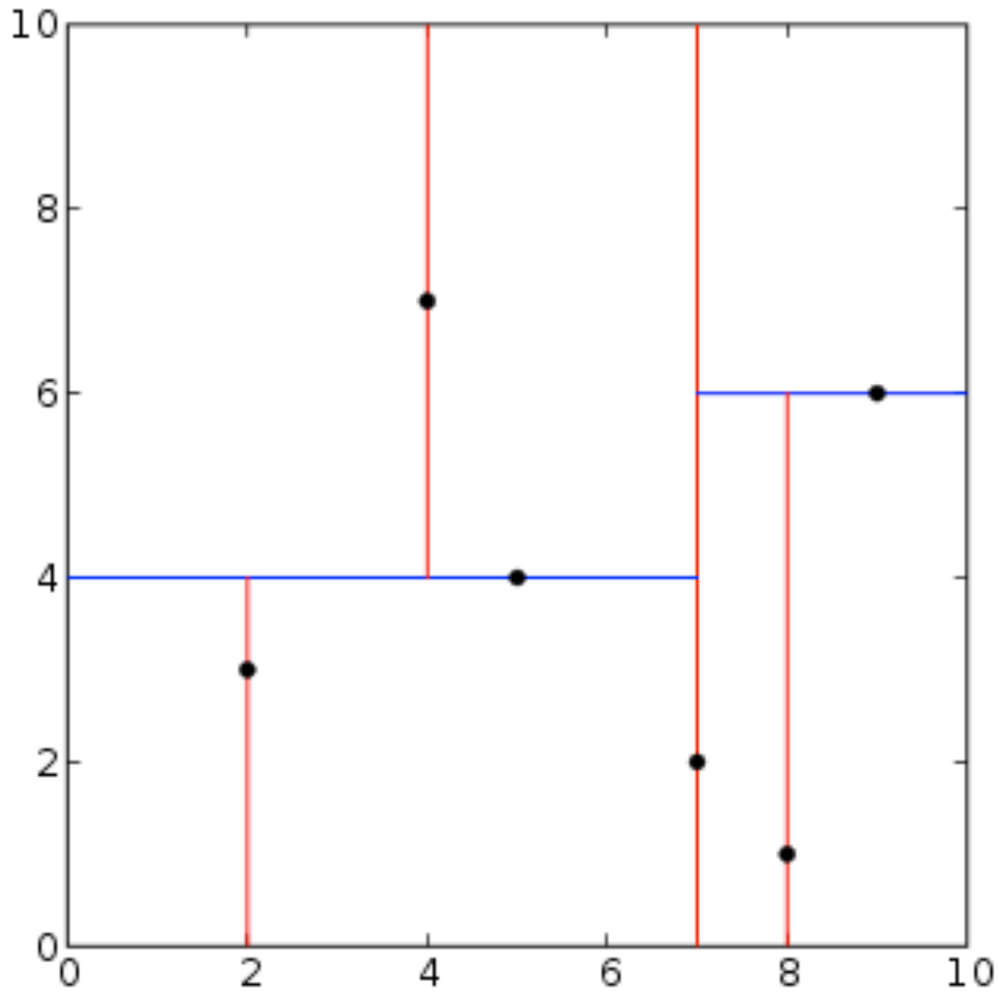
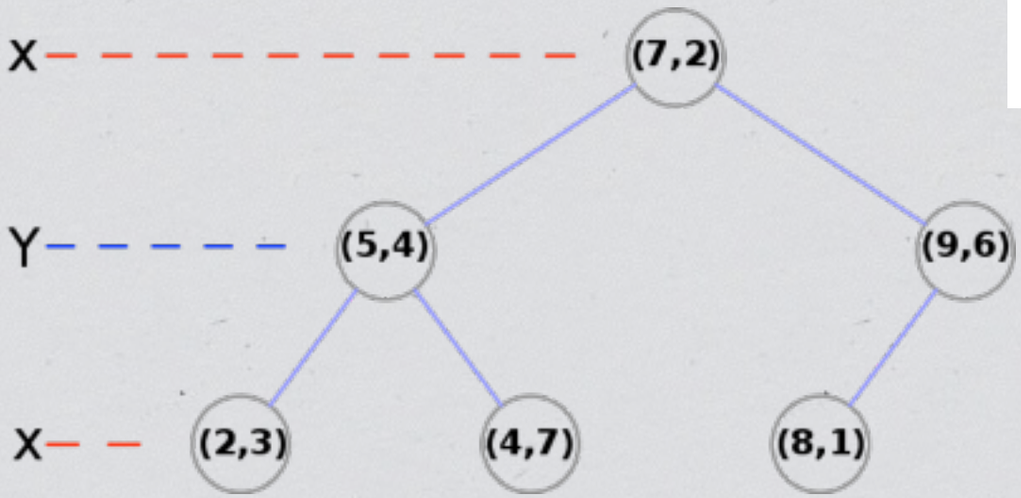
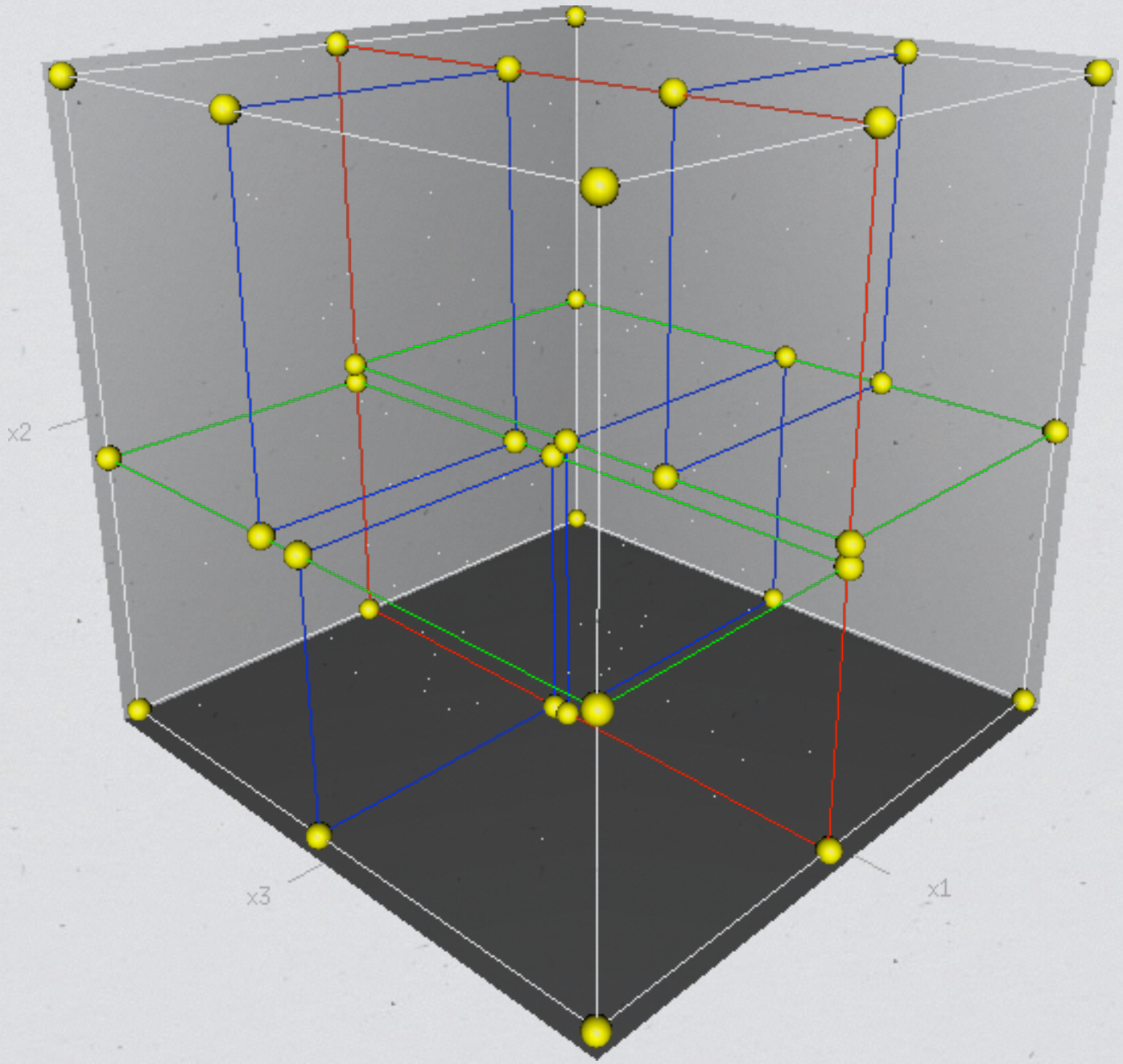
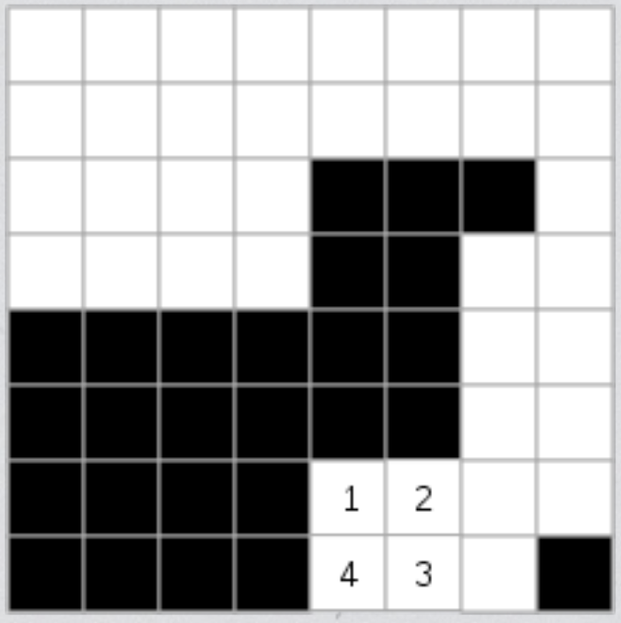
GEOMETRIAI ADATSZERKEZETEK

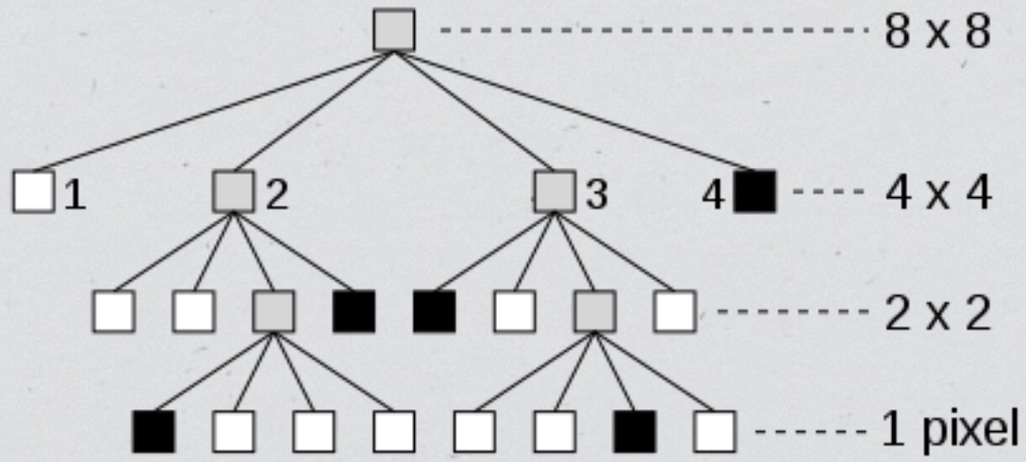
4-es fa, 8-as fa, k-d fa


10
8
6
4
2
0
0
2
4
6
8
10


X
Y
X
(7,2)
(5,4)
(9,6)
(2,3)
(4,7)
(8,1)


x2
x3
x1


1
2
4
3


8 x 8
1
2
3
4
4 x 4
2 x 2
1 pixel

Geometriai algoritmusok

pont helyzete, konvex burok, ponthalmaz legközelebbi és legtávolabbi pontpárja, metsző szakaszpárok keresése

Pont

$p=(x,y)$

$x,y \in R$

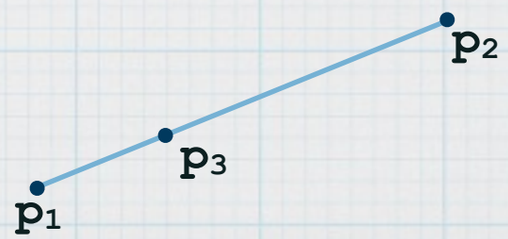
```
class pont {
   double x,y ;
   pont(double ix,iy) {
      x=ix ;
      y=iy ;
   }
}
pont p=new pont(5.2,7.47) ;
```

$p_1=(x_1,y_1)$ $p_2=(x_2,y_2)$ pontok

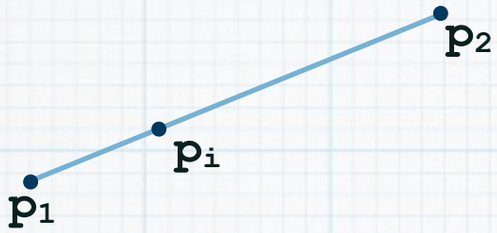
konvex kombinációja $p_3=(x_3,y_3)$ pont, ha

$\exists \alpha \in R : 0 \le \alpha \le 1, x_3=\alpha x_1+(1-\alpha)x_2 \wedge y_3=\alpha y_1+(1-\alpha)y_2$

azaz $p_3=\alpha p_1+(1-\alpha)p_2$



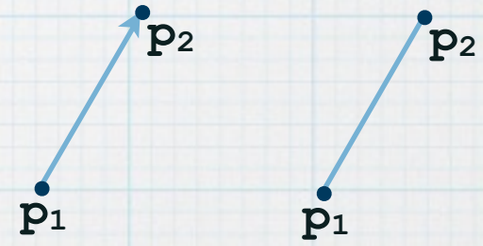
Szakasz



Két különböző p_1 és p_2 pont esetén $\overline{p_1p_2}$ szakasz p_1 és p_2 konvex kombinációinak halmaza.

p_1 és p_2 pontokat a $\overline{p_1p_2}$ szakasz végpontjainak nevezzük.

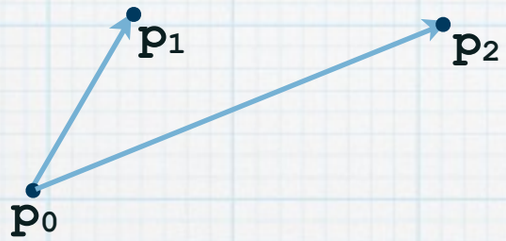
Néha p_1 p_2 sorrendje is számít, ilyenkor $\overrightarrow{p_1p_2}$. irányított szakaszról beszélünk.



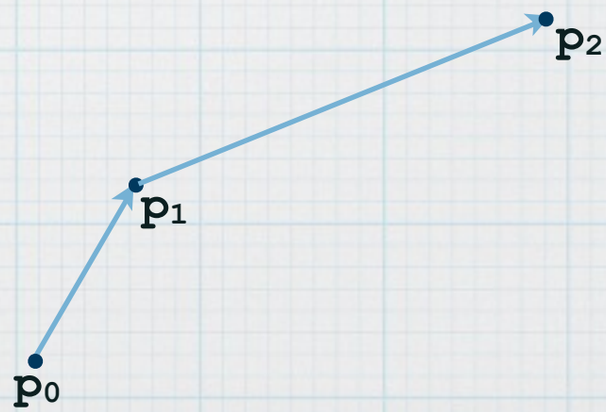

```
class szakasz {
   pont p1,p2 ;
   szakasz(pont ip1,ip2) {
      p1=ip1;
      p2=ip2 ;
   }
}
szaksz s=new szakasz(p15,p26) ;
```

Forgásirány

1. Adott két irányított szakasz: $\overrightarrow{p_0p_1}$ és $\overrightarrow{p_0p_2}$.
   A közös p_0 végpont körül $\overrightarrow{p_0p_1}$-hez viszonyítva
   p_0p_2 jobbra ( ), vagy balra fordul?



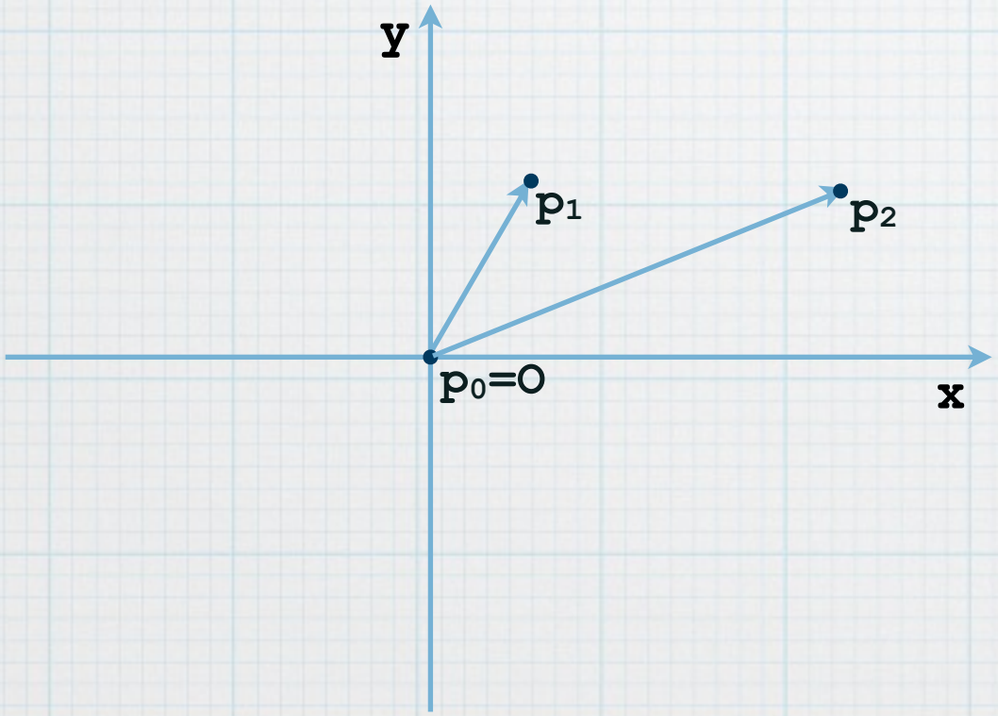
2. Adott $\overrightarrow{p_0p_1}$ és $\overrightarrow{p_1p_2}$ szakasz.
   Ha folyamatosan bejárjuk $\overrightarrow{p_0p_1}$-et majd $\overrightarrow{p_1p_2}$-t,
   balra fordulunk-e a p_1 pontban?

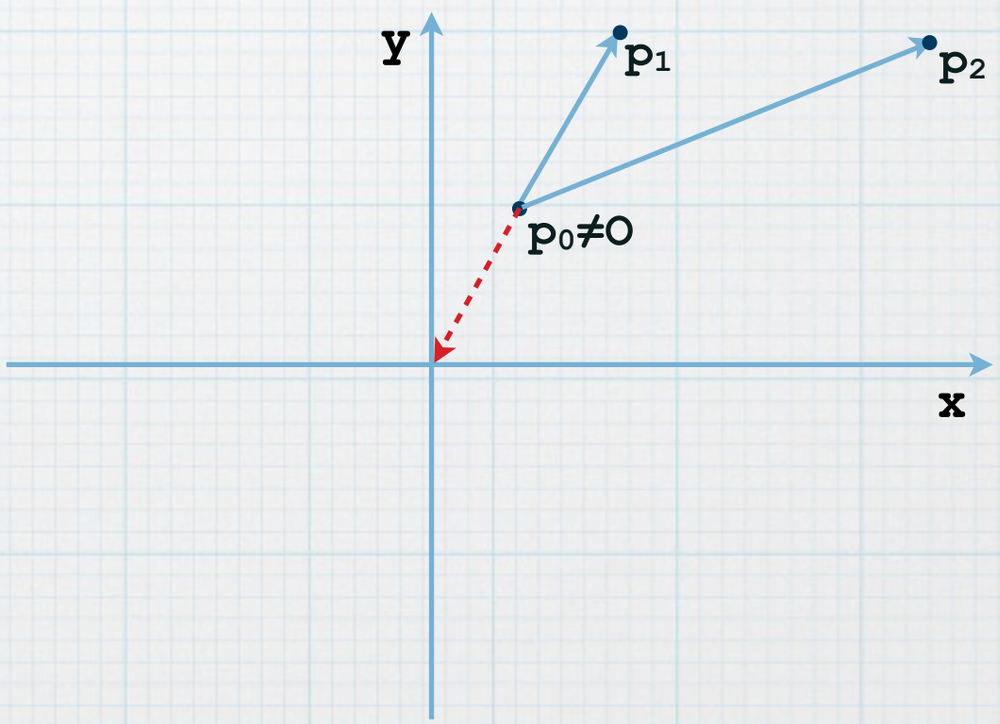


1. Forgásirány

$$\overrightarrow{p_1} \times \overrightarrow{p_2} = \begin{vmatrix} x_1 & x_2 \\ y_1 & y_2 \end{vmatrix} = x_1 y_2 - x_2 y_1$$

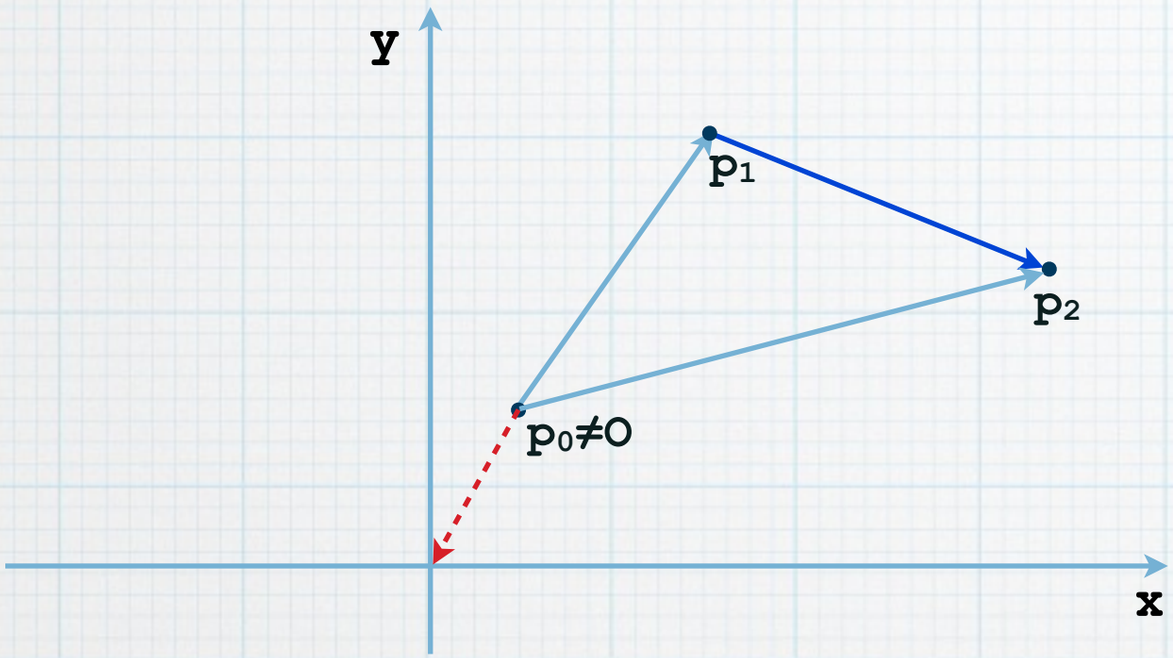
Ha $\vec{p}_1 \times \vec{p}_2$ pozitív, $\vec{p}_1$ jobbra fordul $\vec{p}_2$-höz képest.





$$f = (p_1 - p_0) \times (p_2 - p_0) = (x_1 - x_0)(y_2 - y_0) - (x_2 - x_0)(y_1 - y_0)$$

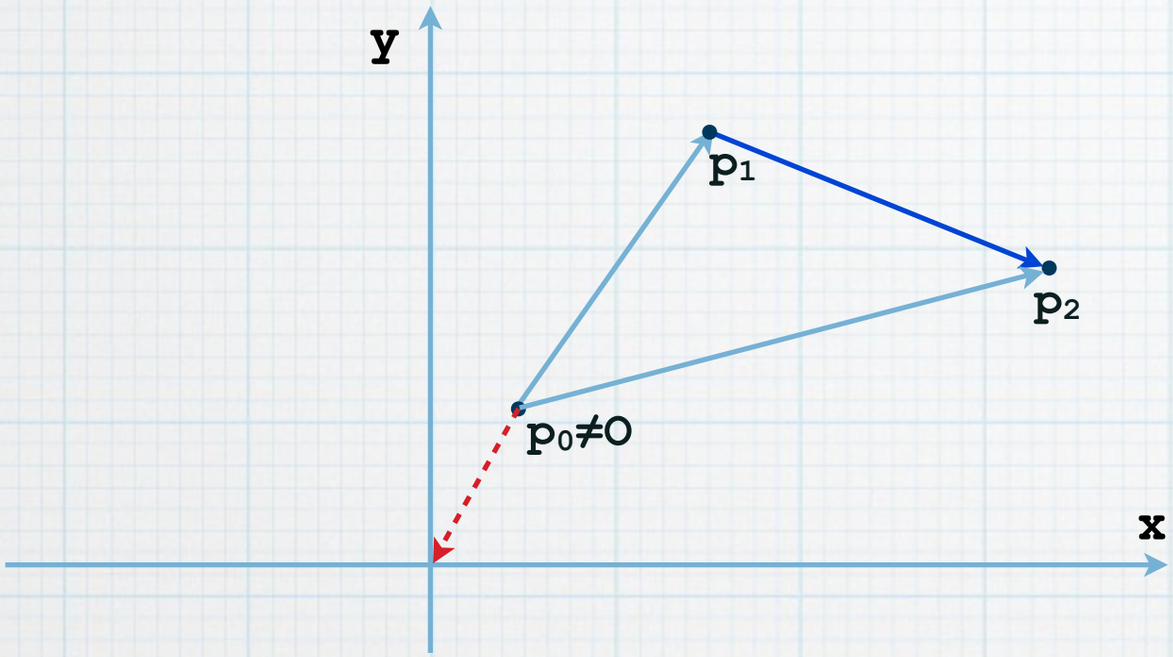
2. Forgásirány



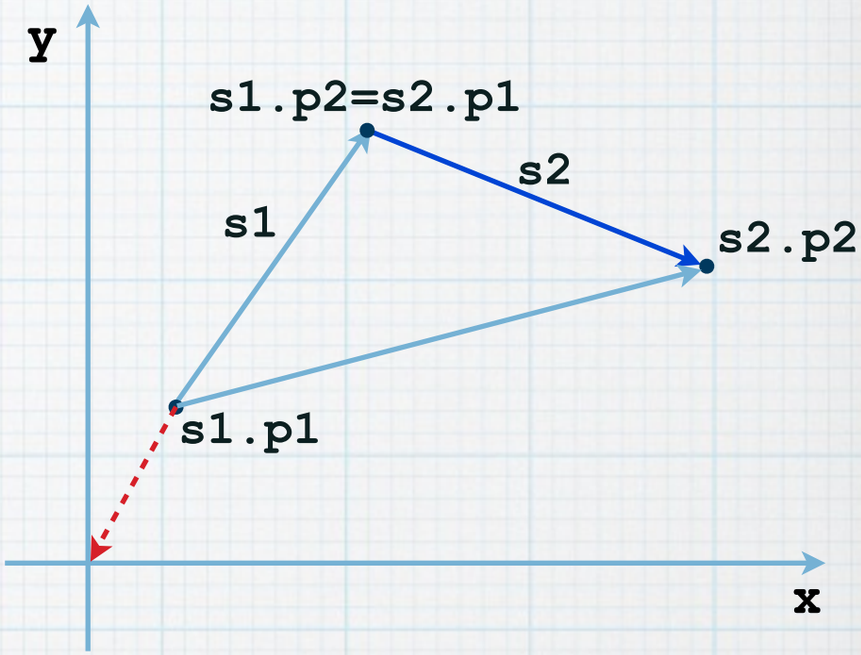
$$f = (p_1 - p_0) \times (p_2 - p_0) = (x_1 - x_0)(y_2 - y_0) - (x_2 - x_0)(y_1 - y_0)$$

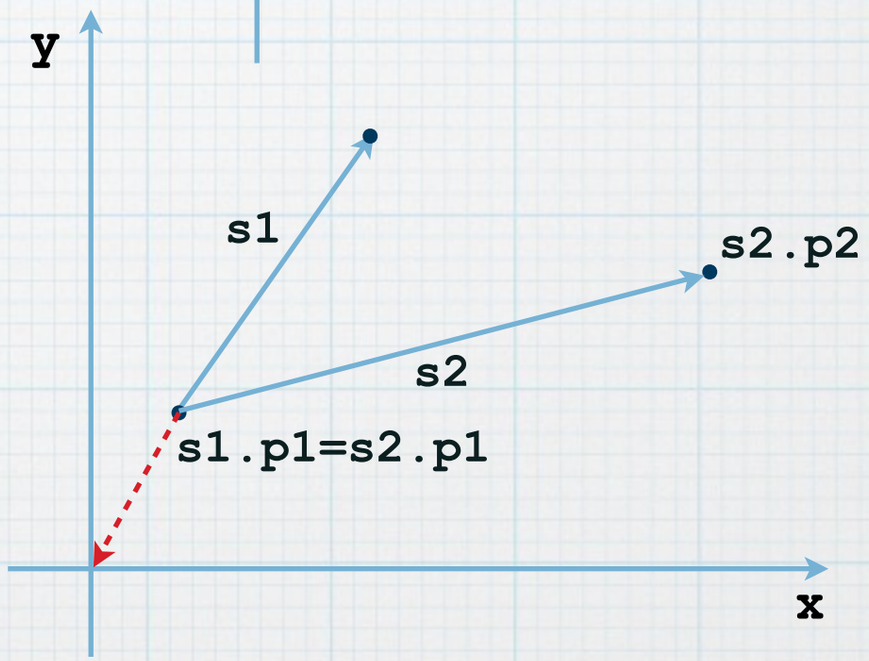
***f>0*⇒balra fordul**
***f<0*⇒jobbra fordul**
***f=0*⇒egy egyenesbe esik (nem fordul)**

Forgásirány



$$f = (p_1 - p_0) \times (p_2 - p_0) = (x_1 - x_0)(y_2 - y_0) - (x_2 - x_0)(y_1 - y_0)$$

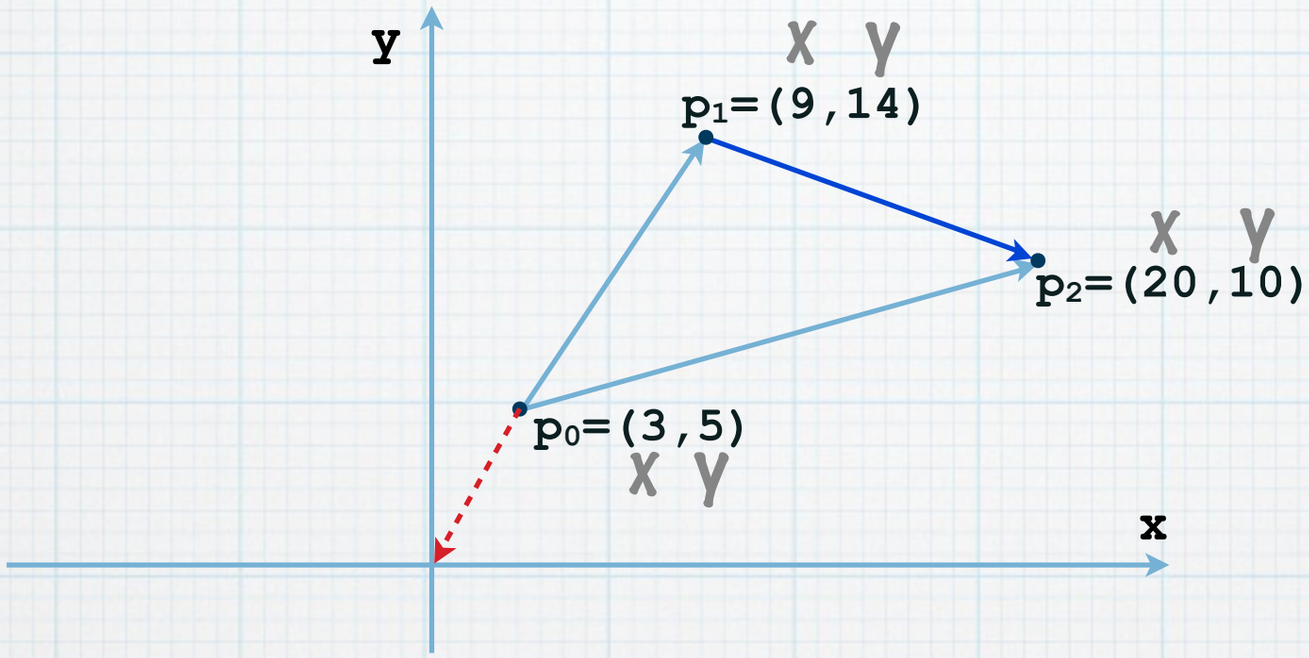





```
forgasirany(pont p0,p1,p2) {
   f=(p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y) ;
   if (f>0) return 1 ;
   if (f<0) retun -1;
   retun 0;
}
```

Forgásirány

példa:



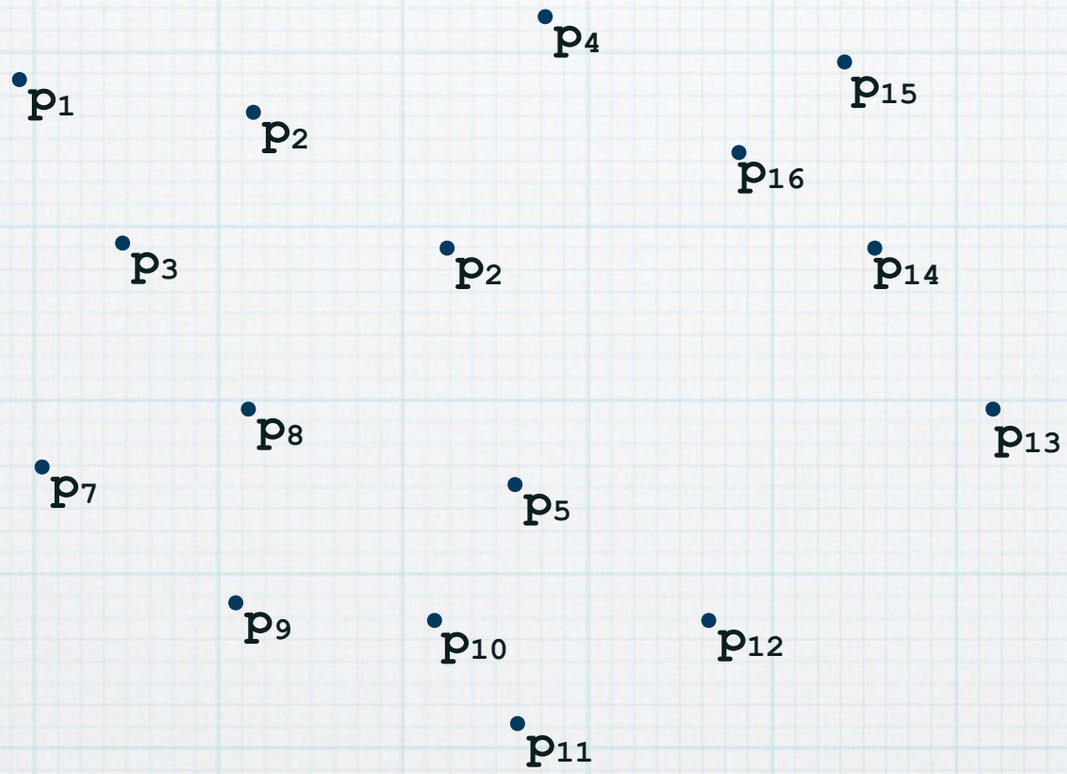

```
forgasirany(pont p0,p1,p2) {
   f=(p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y) ;
   if (f>0) return 1 ; // balra fordul
   if (f<0) retun -1; // jobbra fordul
   retun 0;
}
```

9 3 10 5 20 3 14 5

6 5 17 9

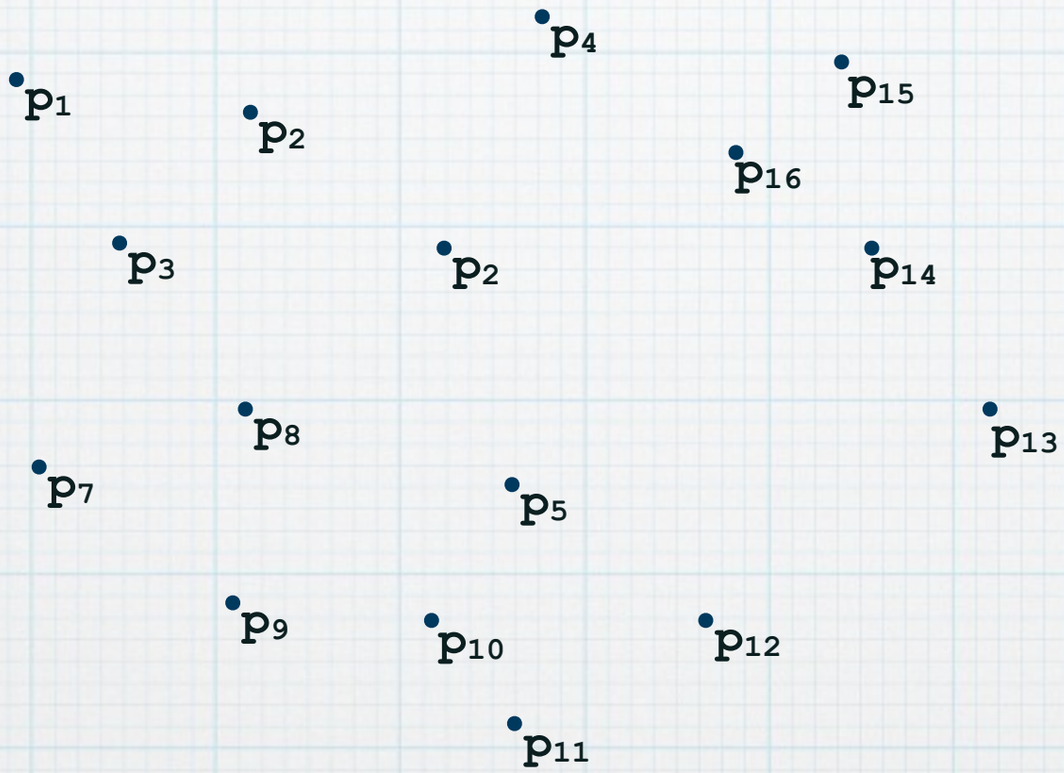
30 153 -123

Adott pontok egy $\{p_1, \ldots, p_n\}$ halmaza.



$\{p_1, \ldots, p_n\}$ ponthalmaz polárszög szerinti rendezése

1. Válasszuk ki a legkisebb x koordinátájú pontot, ha több ilyen van, akkor válasszuk ezek közül a legkisebb y koordinátájút és jelüljük q-val.

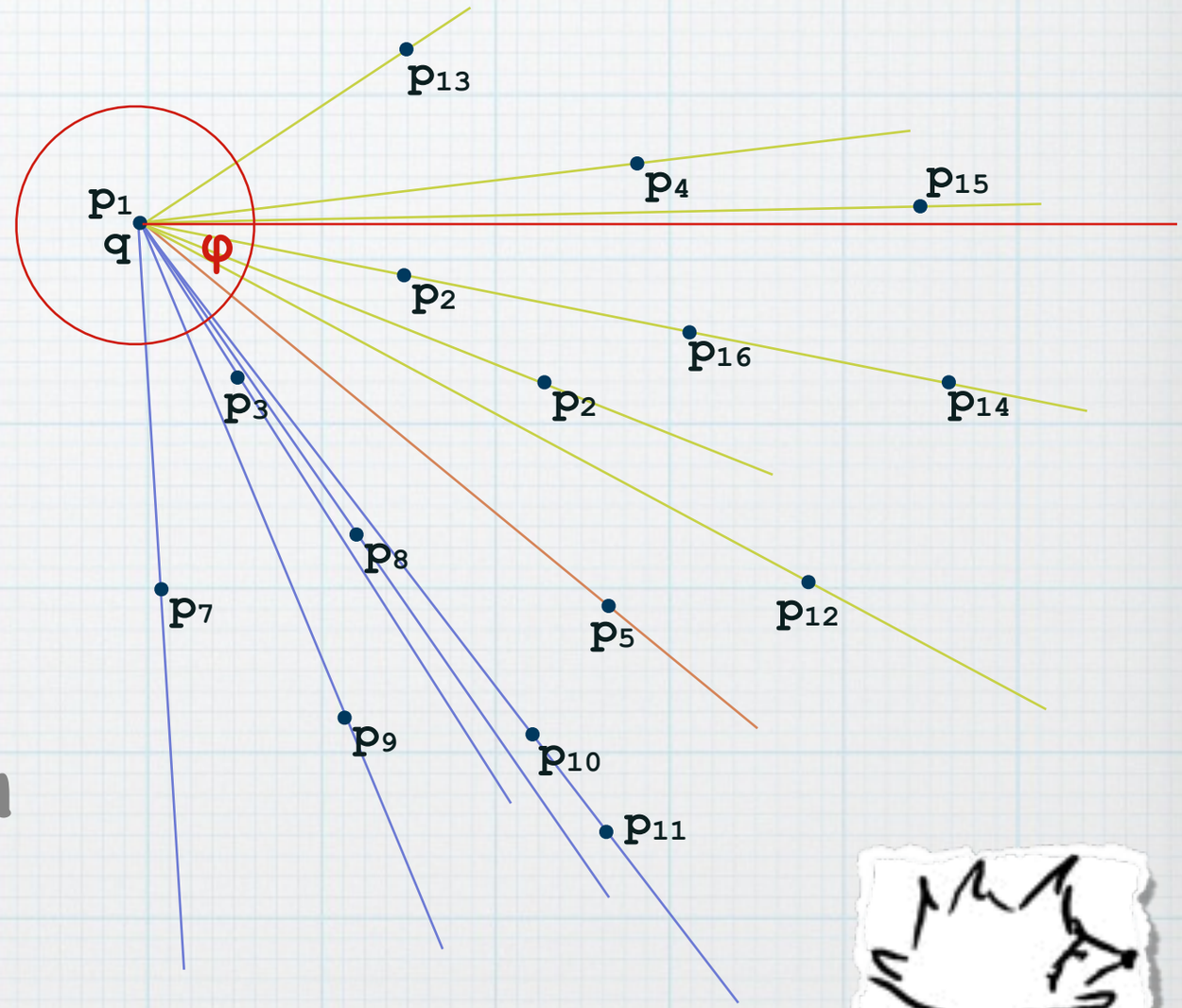


$\{p_1,\ldots,p_n\}$ ponthalmaz polárszög szerinti rendezése

1. Válasszuk ki a legkisebb x koordinátájú pontot, ha több ilyen van, akkor válasszuk ezek közül a legkisebb y koordinátájút és jelüljük **q**-val.

2. Húzzunk félegyeneseket a **q**-ból minden ponthoz.

   Rendezzük az egyeneseket a **q** ponton áthaladó, x-tengellyel párhuzamos egyenessel bezárt (előjeles) szög alapján. **φ(q,p)**



3. Rendezzük a pontokat úgy, hogy **q** legyen az első és **p** előbb legyen mint **r** ha:
   φ(q,p)<φ(q,r) vagy
   φ(q,p)=φ(q,r) és **p** közelebb van **q**-hoz mint **r**.

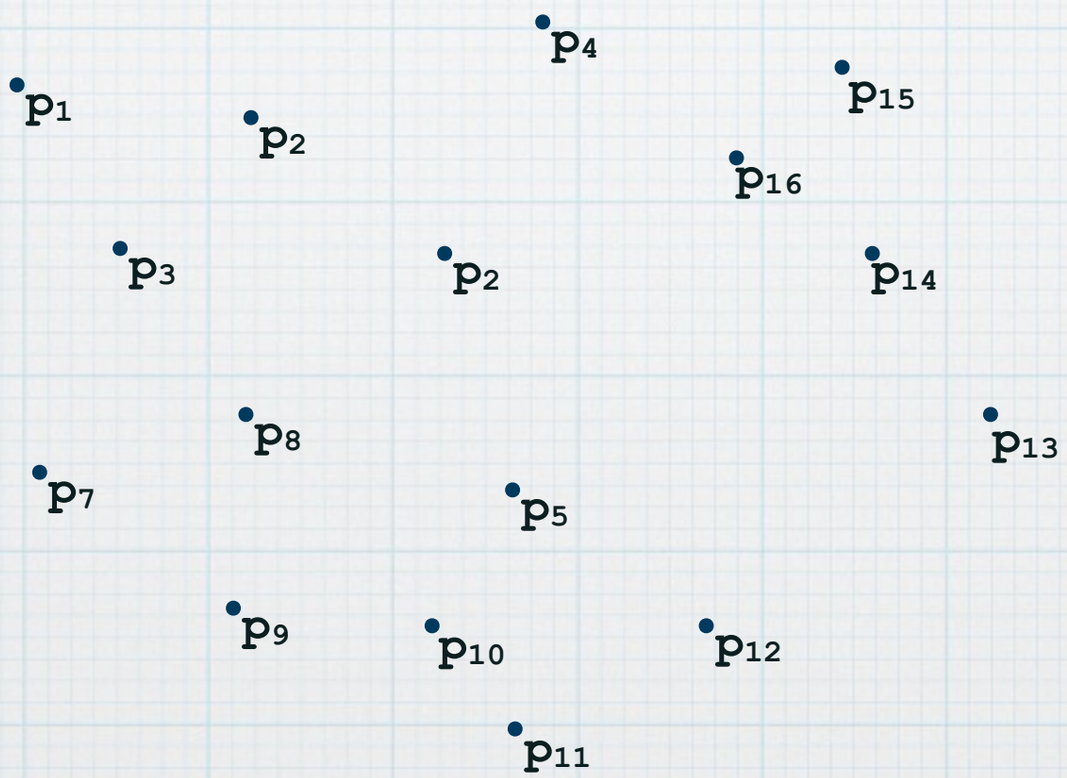
p<r ⇔ `forgasirany(q,p,r)==1`

O(nlogn)

Feladat

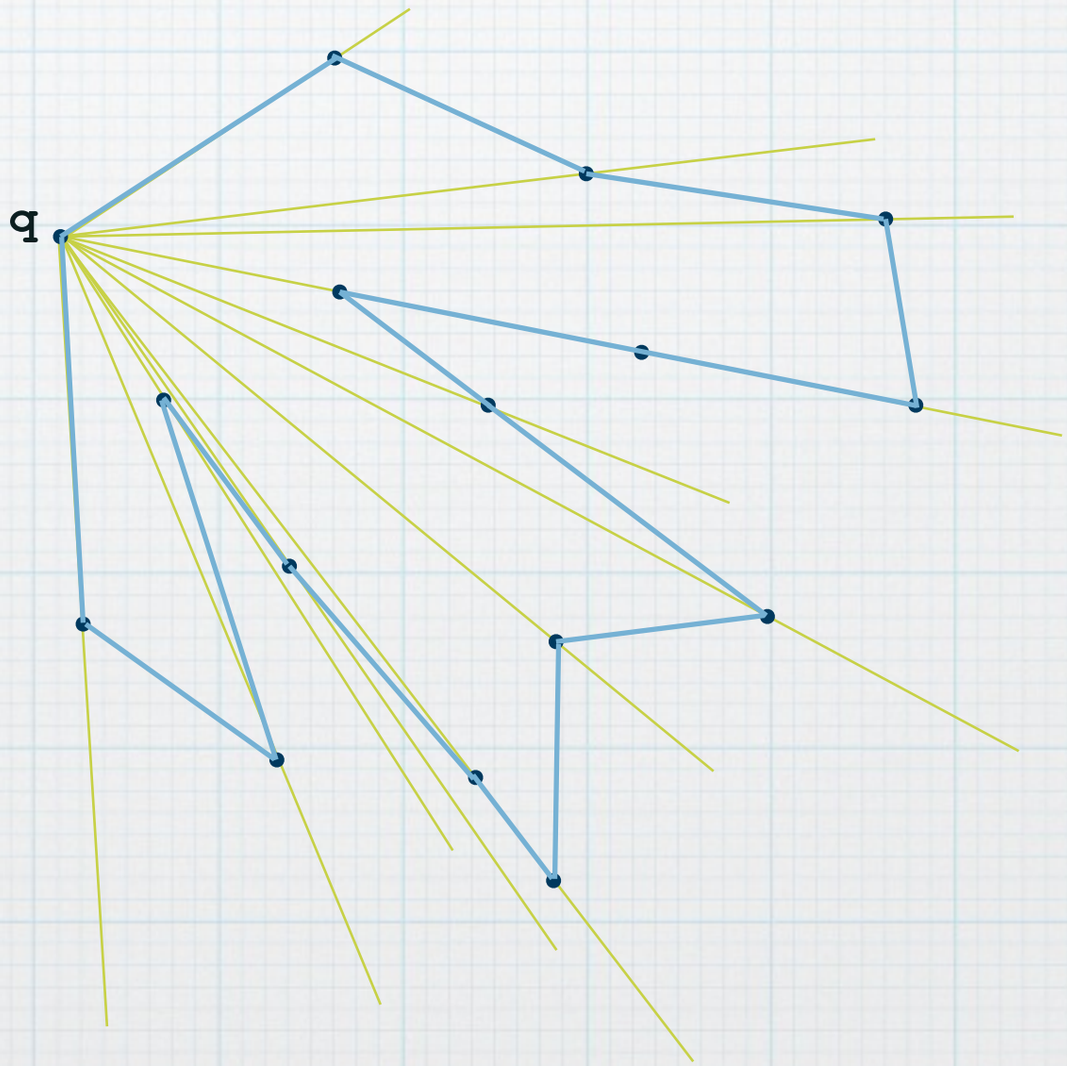
Pontok összekötése zárt nem metsző poligonná

Adott $\{p_1, \ldots, p_n\}$ ponthalmaz.
Kössük össze $p_1, \ldots, p_n$ pontokat zárt, nem metsző poligonná!



Megoldás

1. A ponthalmaz polárszög szerinti rendezése.
2. Kössük össze a pontokat a rendezés szerinti sorrendben.



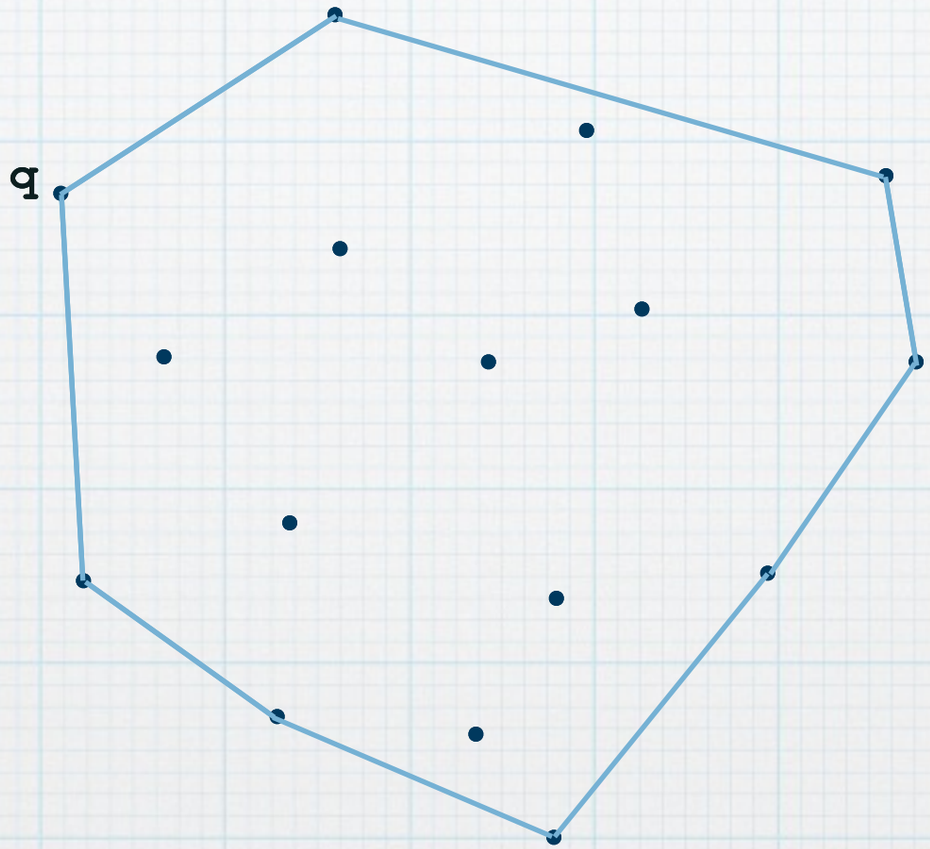
Feladat

Konvex burok meghatározása

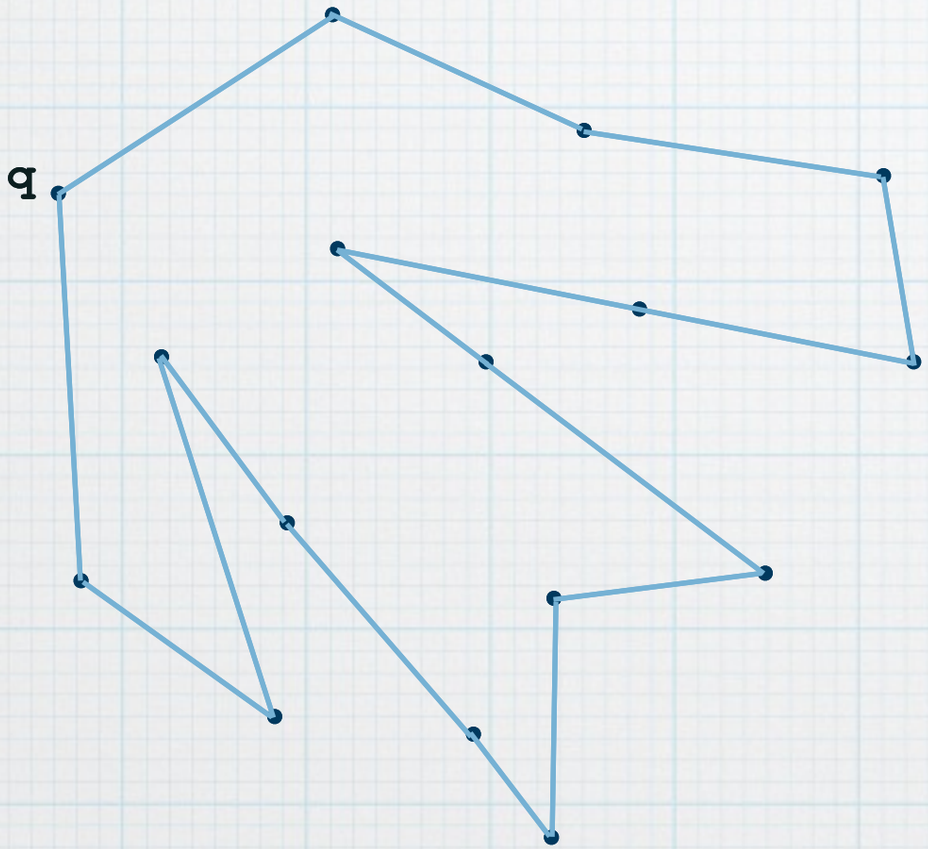
Adott $\{p_1, \ldots, p_n\}$ ponthalmaz.
Határozzuk meg $p_1, \ldots, p_n$ ponthalmaz konvex burkát!

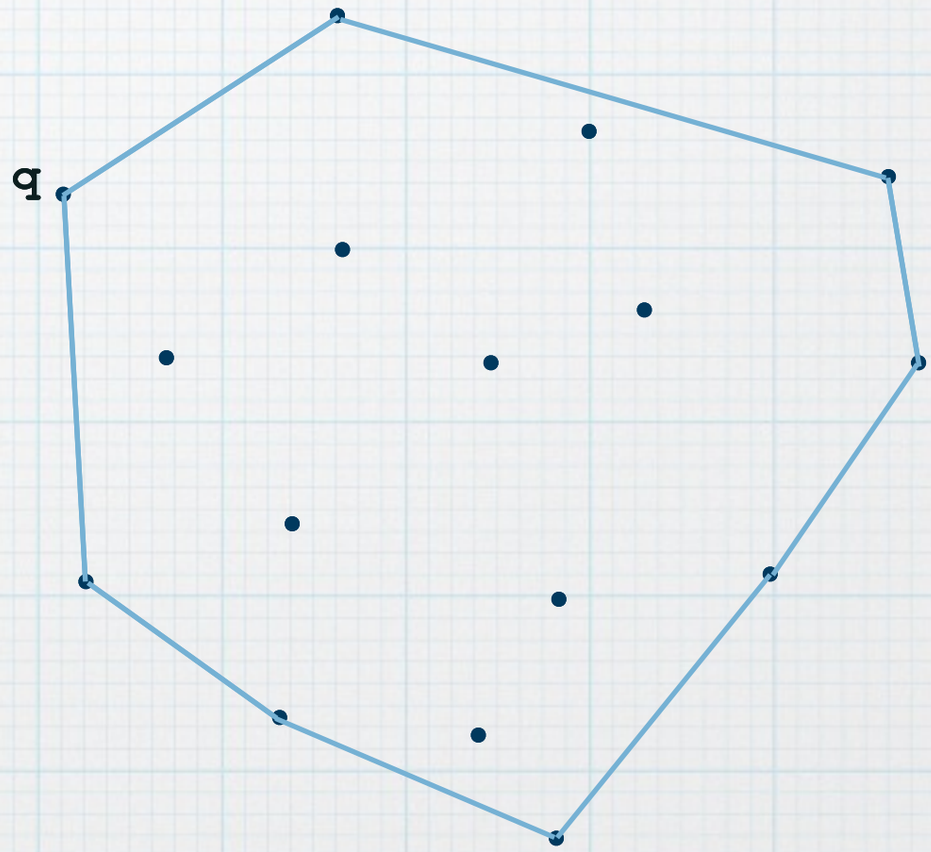
Definíció:
Egy egyszerű (nem metsző) poligon konvex, ha bármely két belső pontját összekötő szakasz minden pontja a poligon belsejében, vagy határán van.

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


q

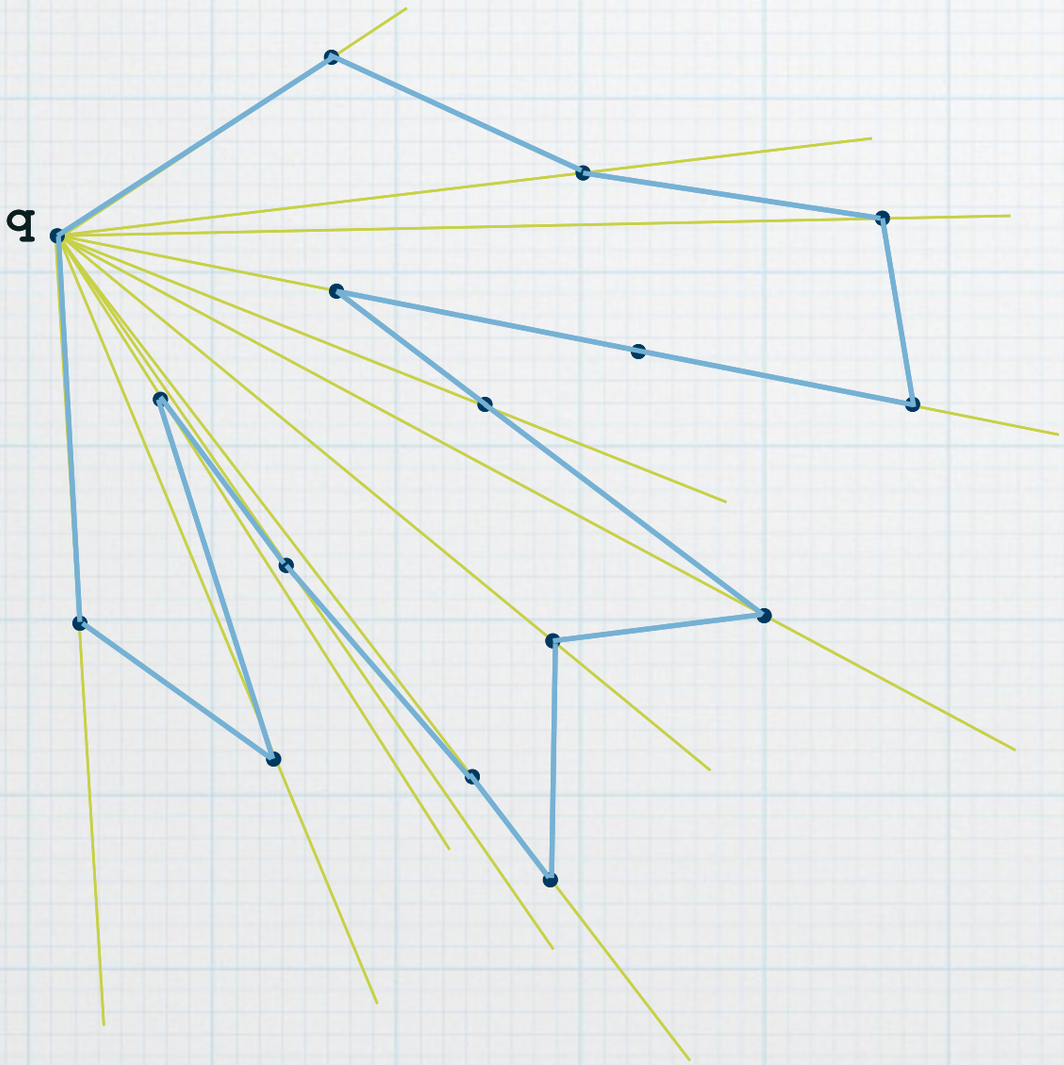
Megoldás

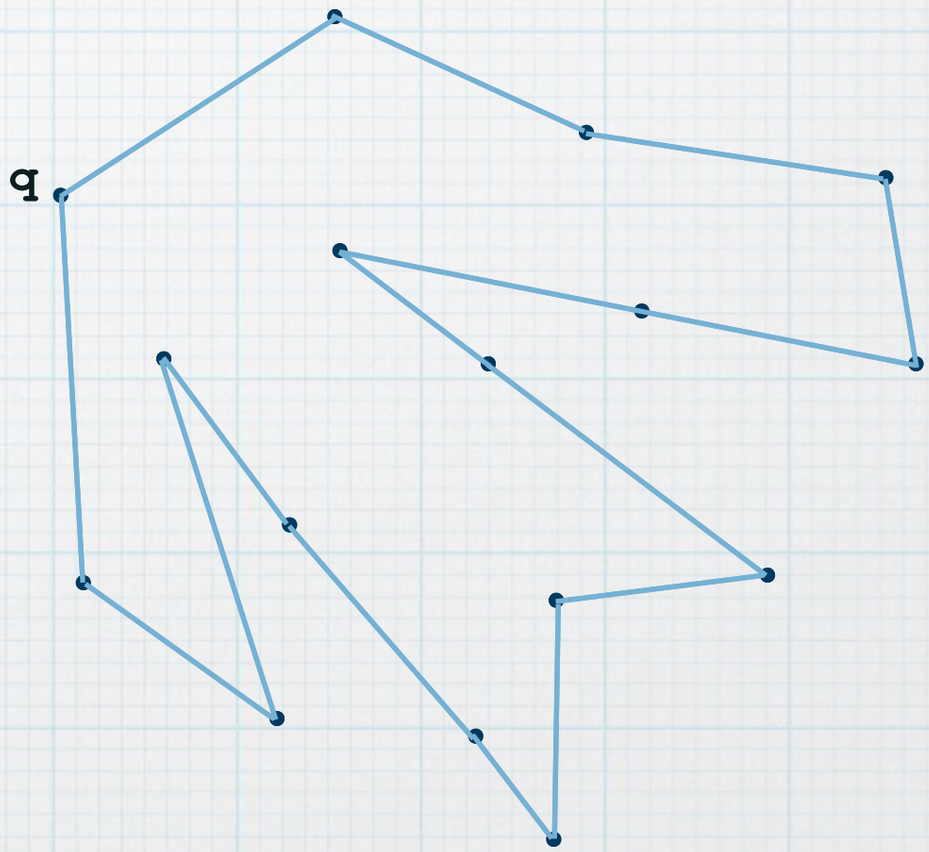




Megoldás

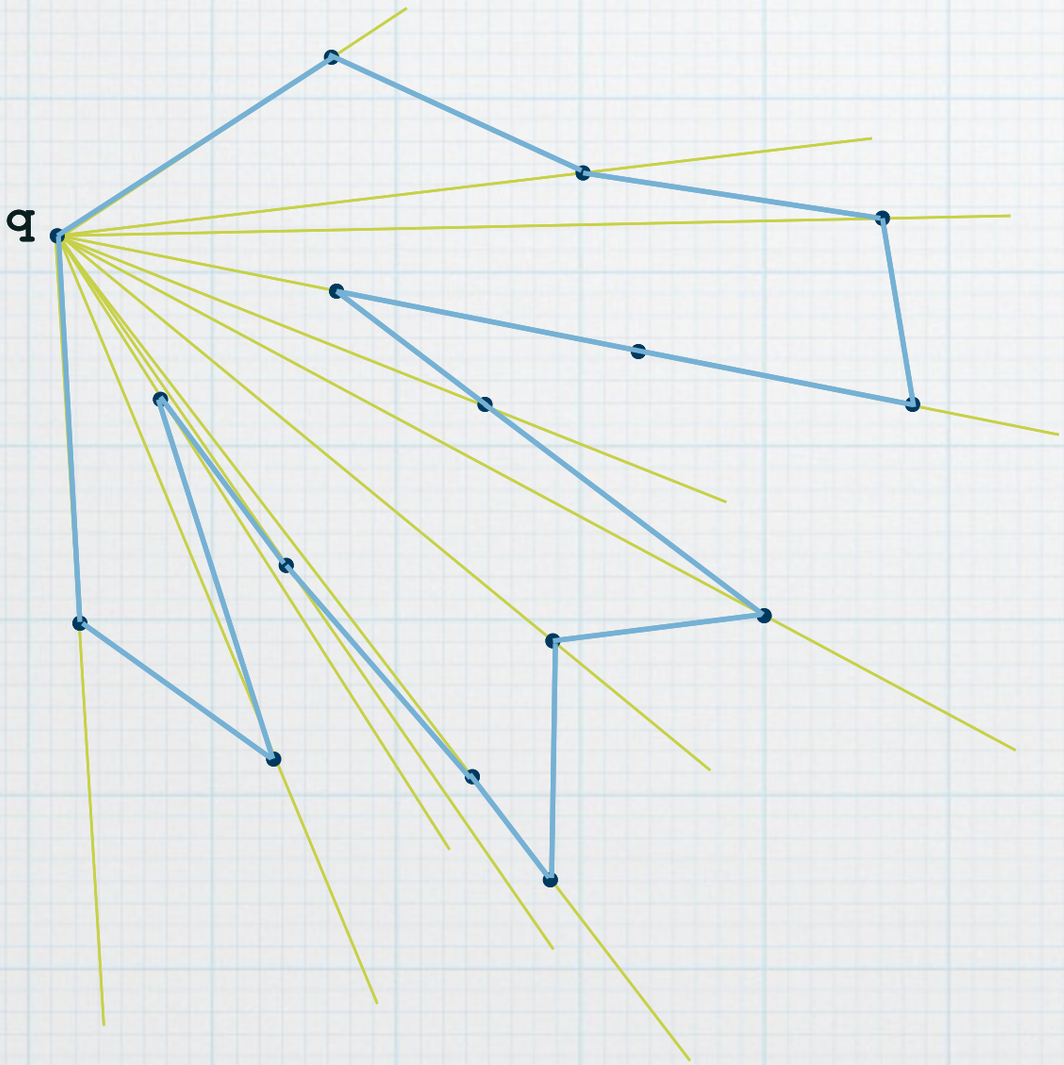
1. a ponthalmaz polárszög szerinti rendezése.

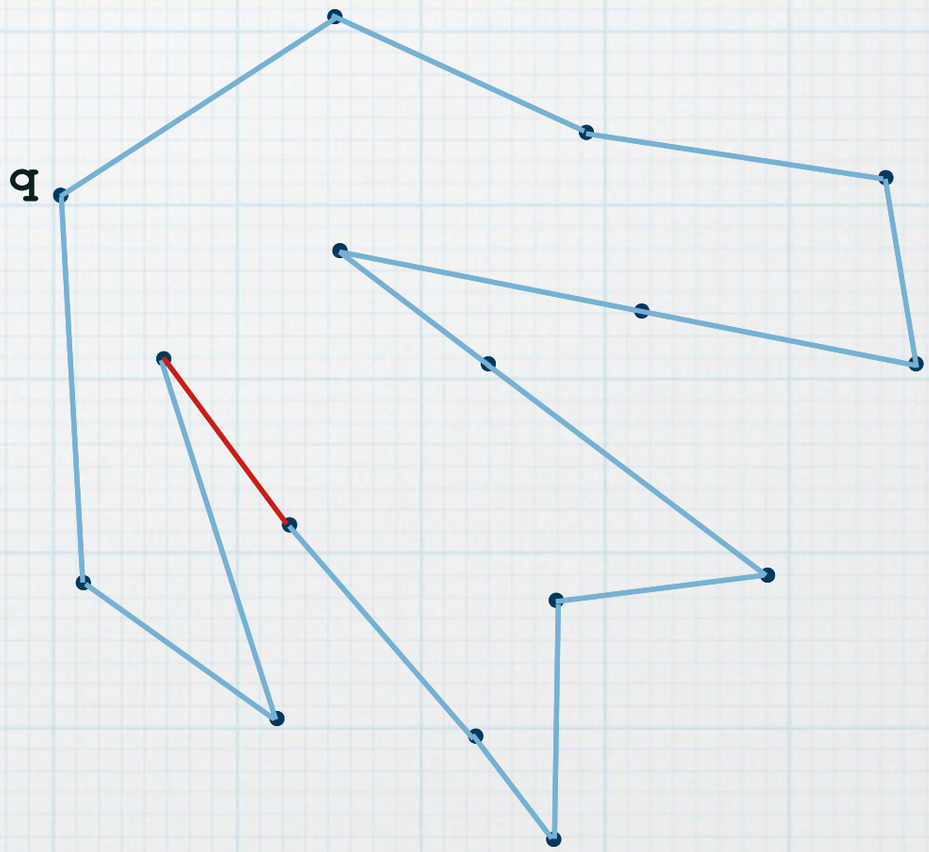




Megoldás

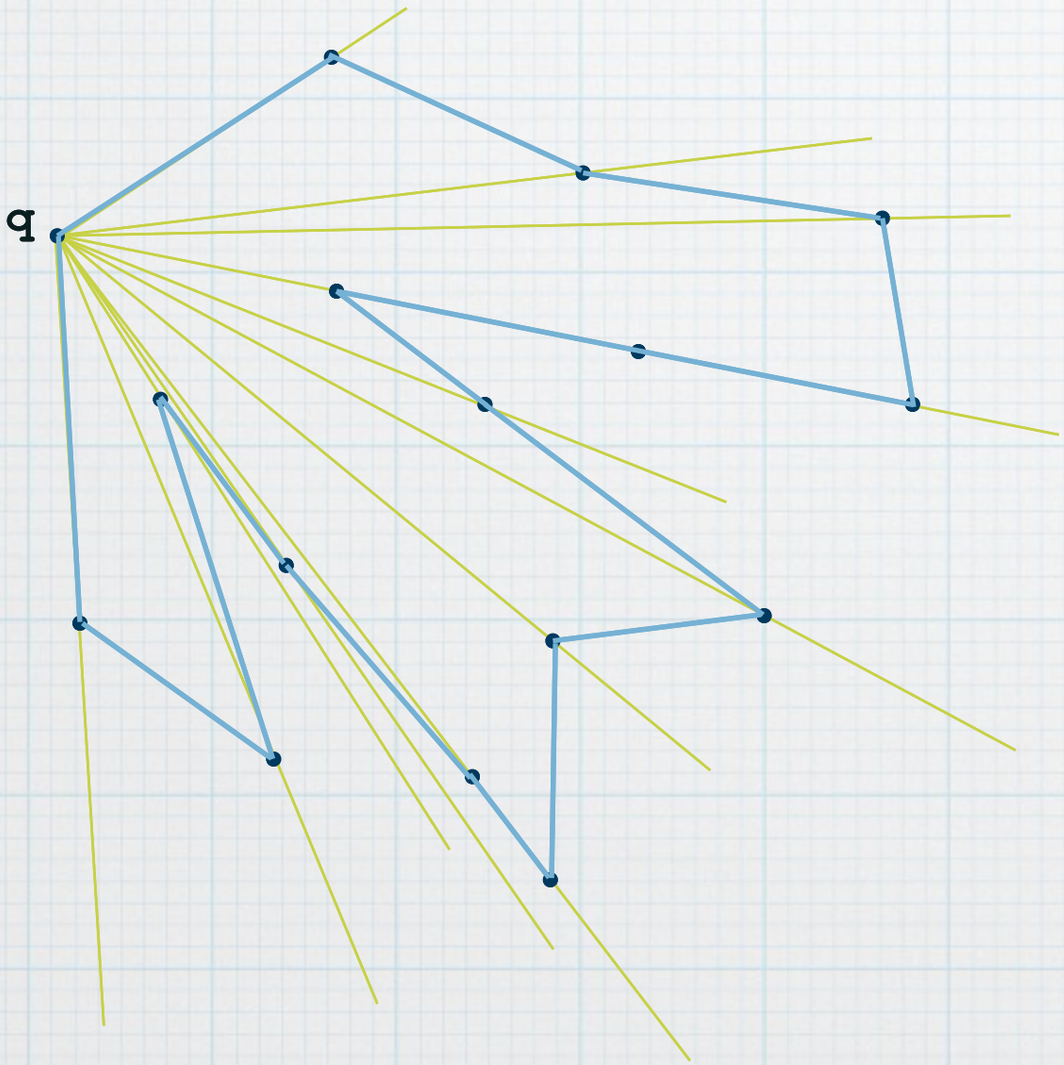
1. a ponthalmaz polárszög szerinti rendezése.

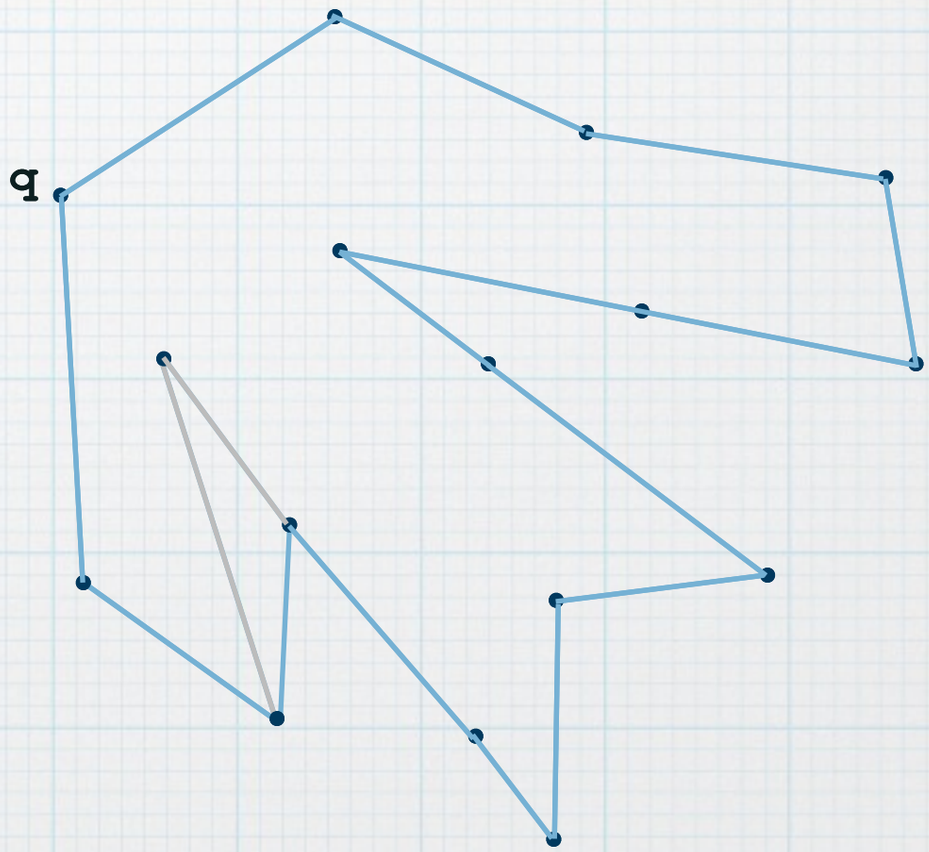




Megoldás

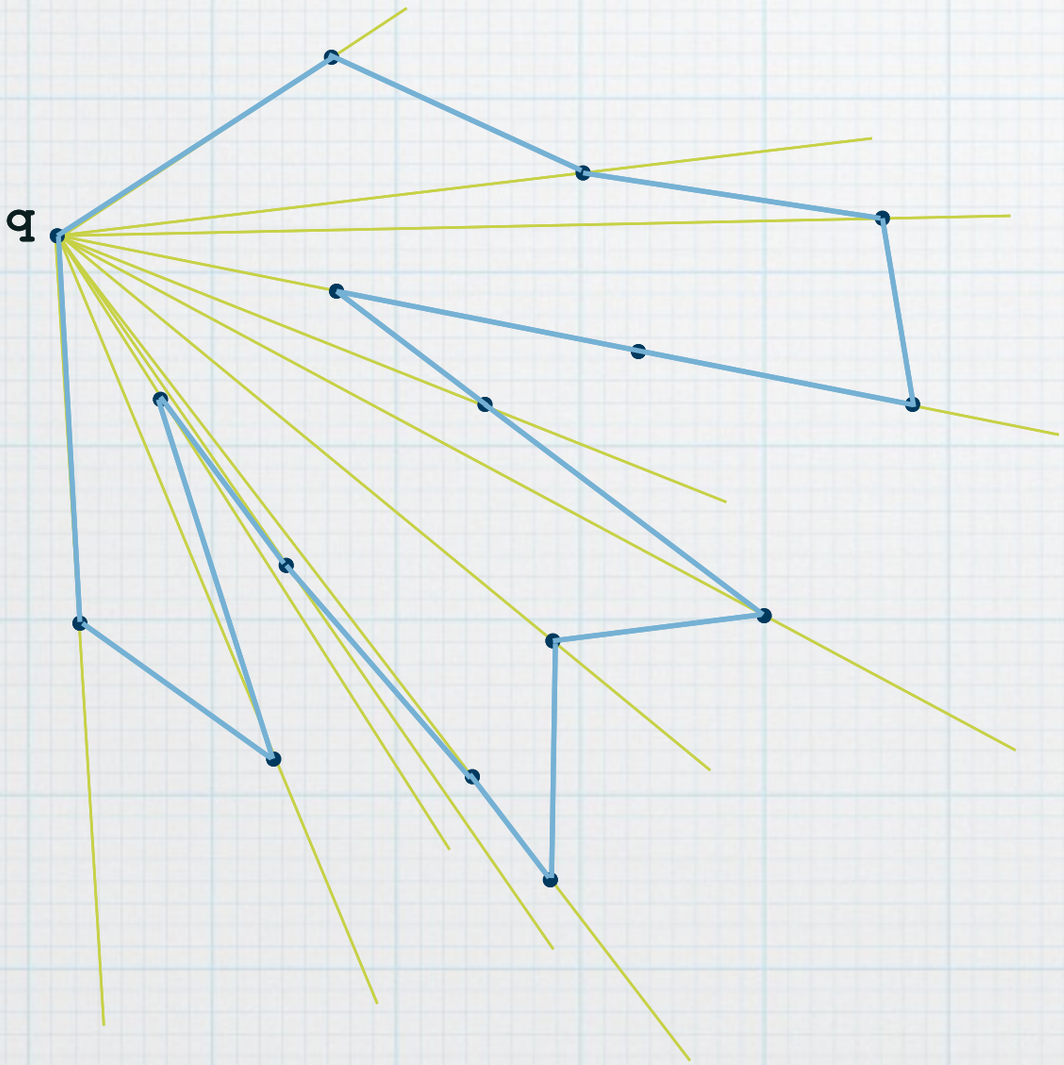
1. a ponthalmaz polárszög szerinti rendezése.

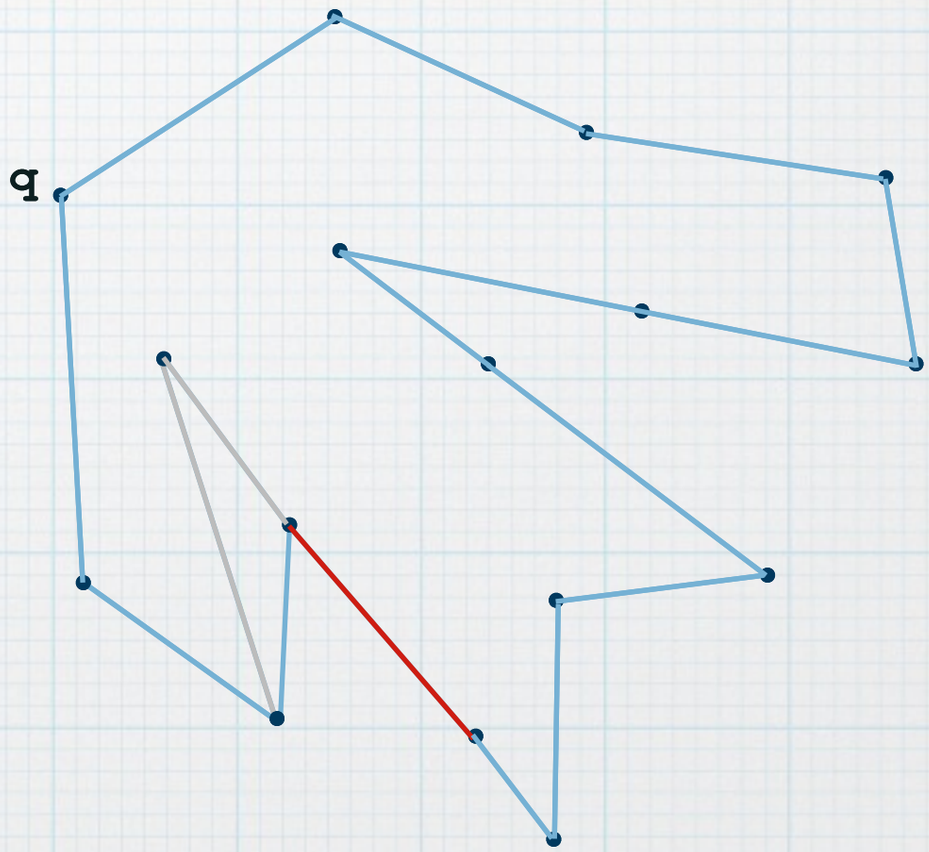




Megoldás

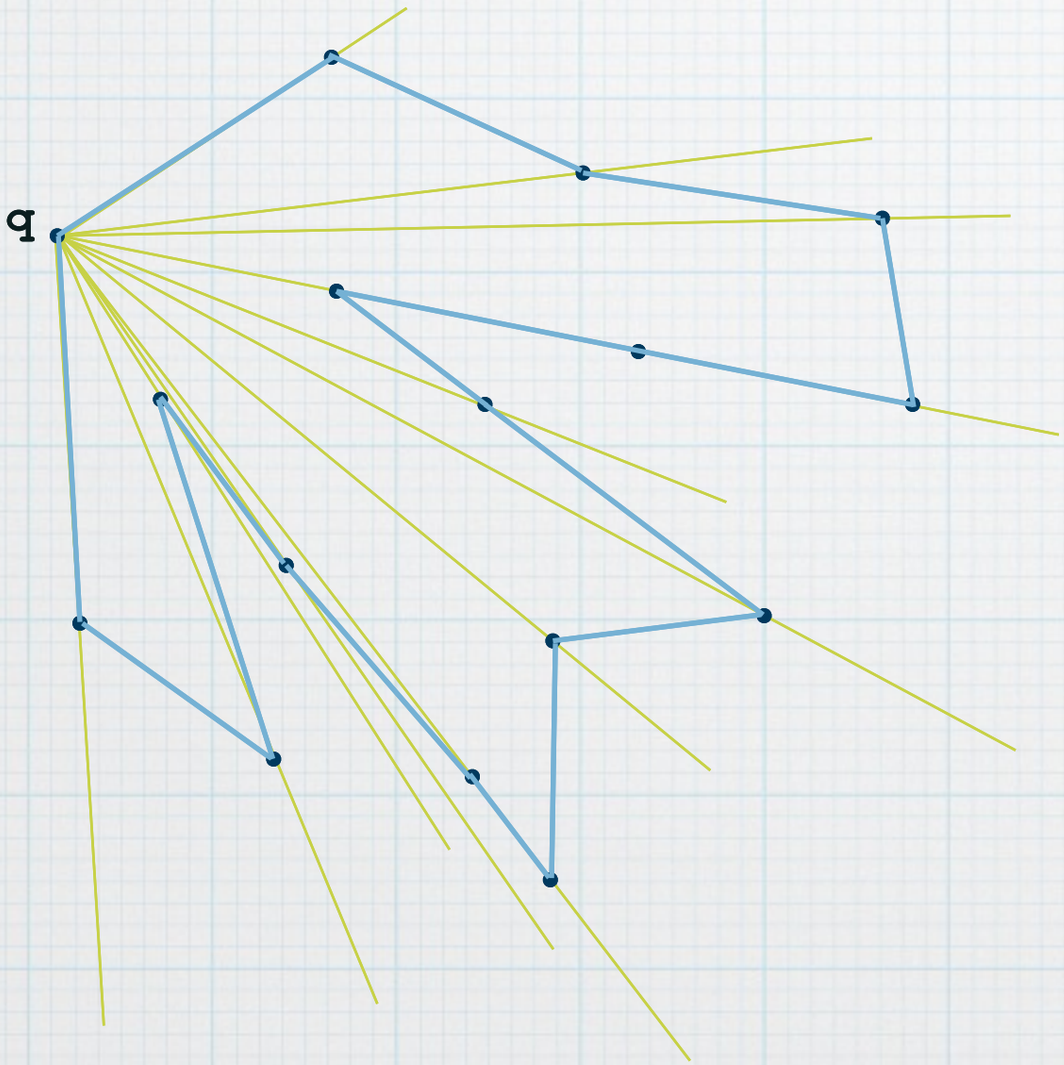
1. a ponthalmaz polárszög szerinti rendezése.

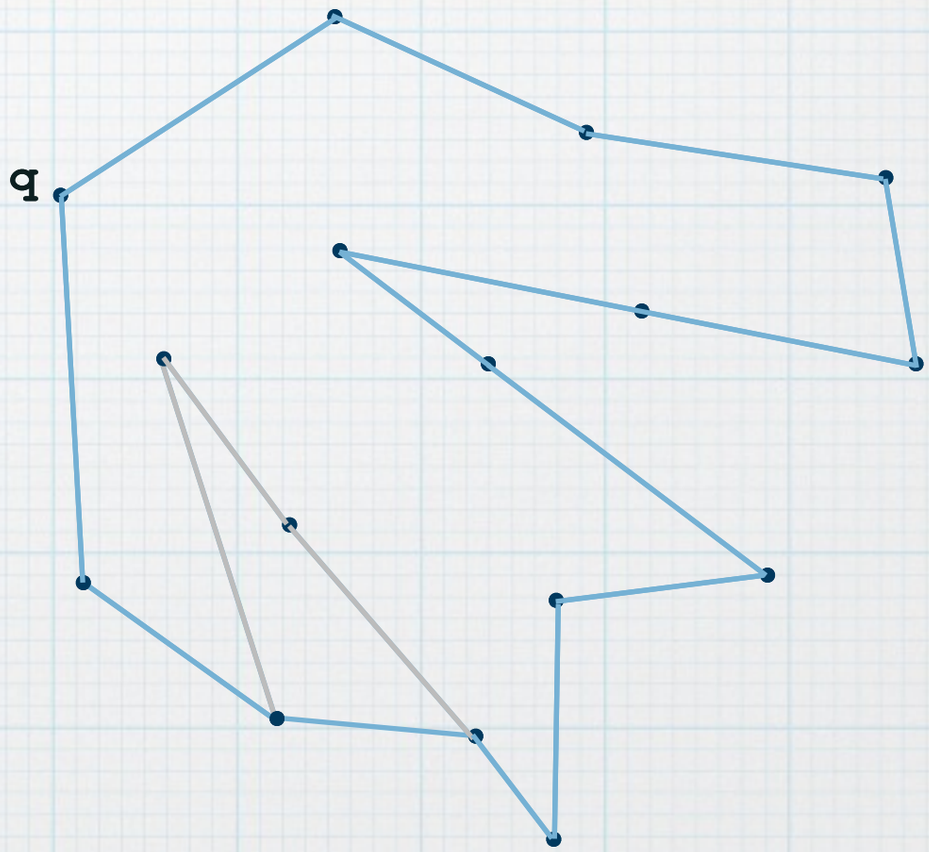




Megoldás

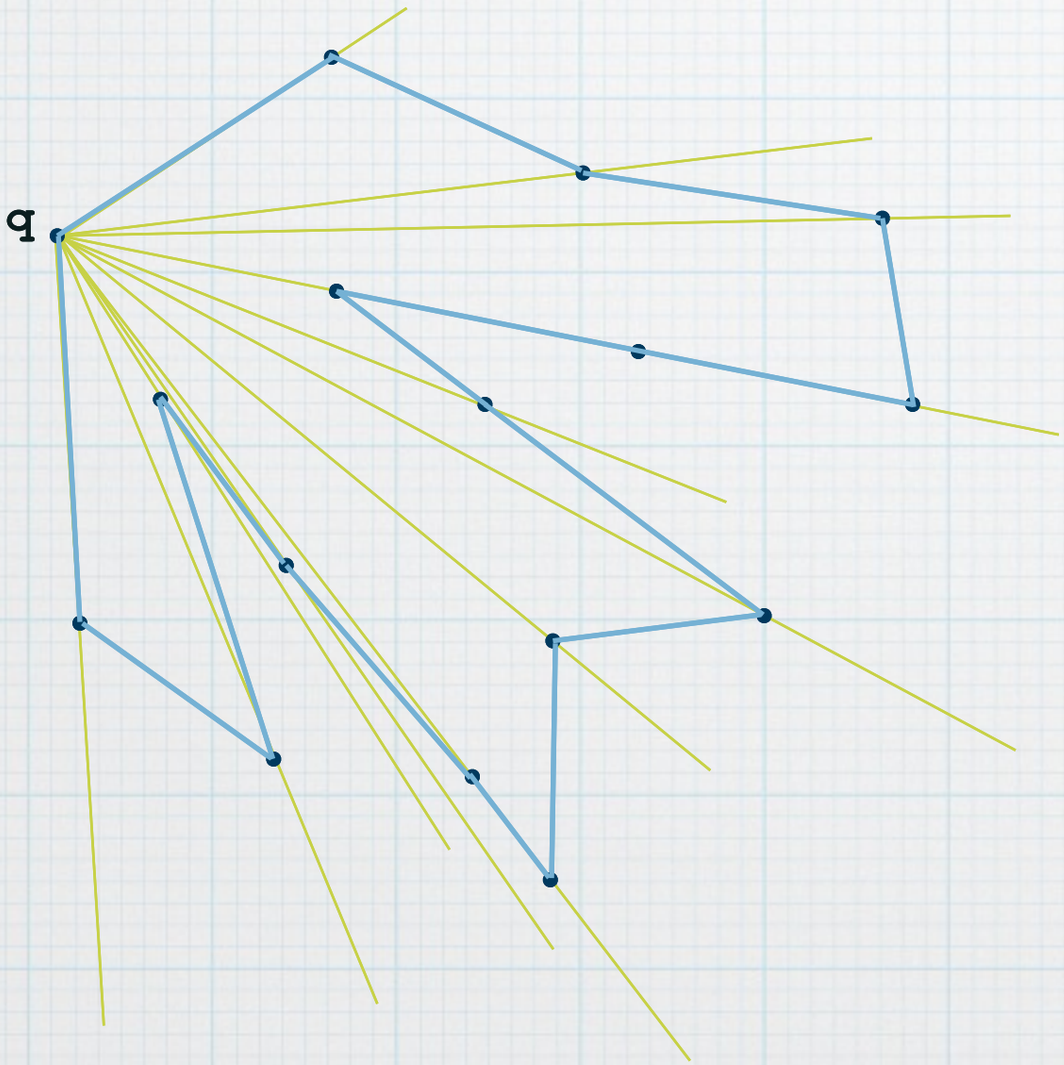
1. a ponthalmaz polárszög szerinti rendezése.

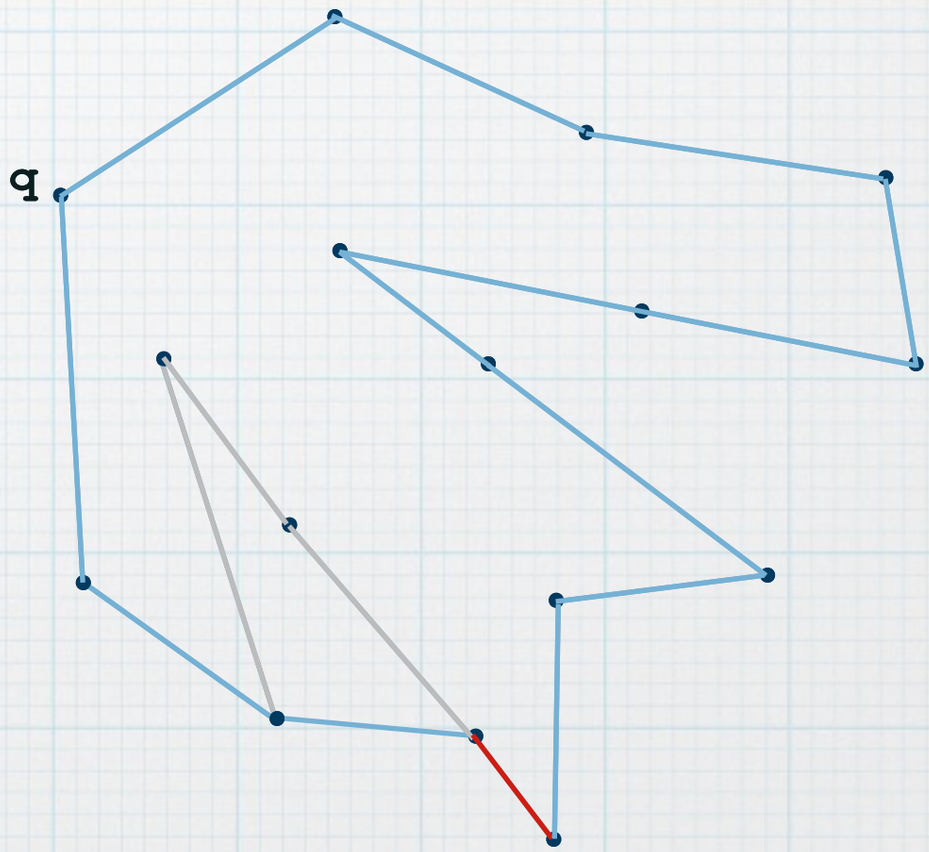




Megoldás

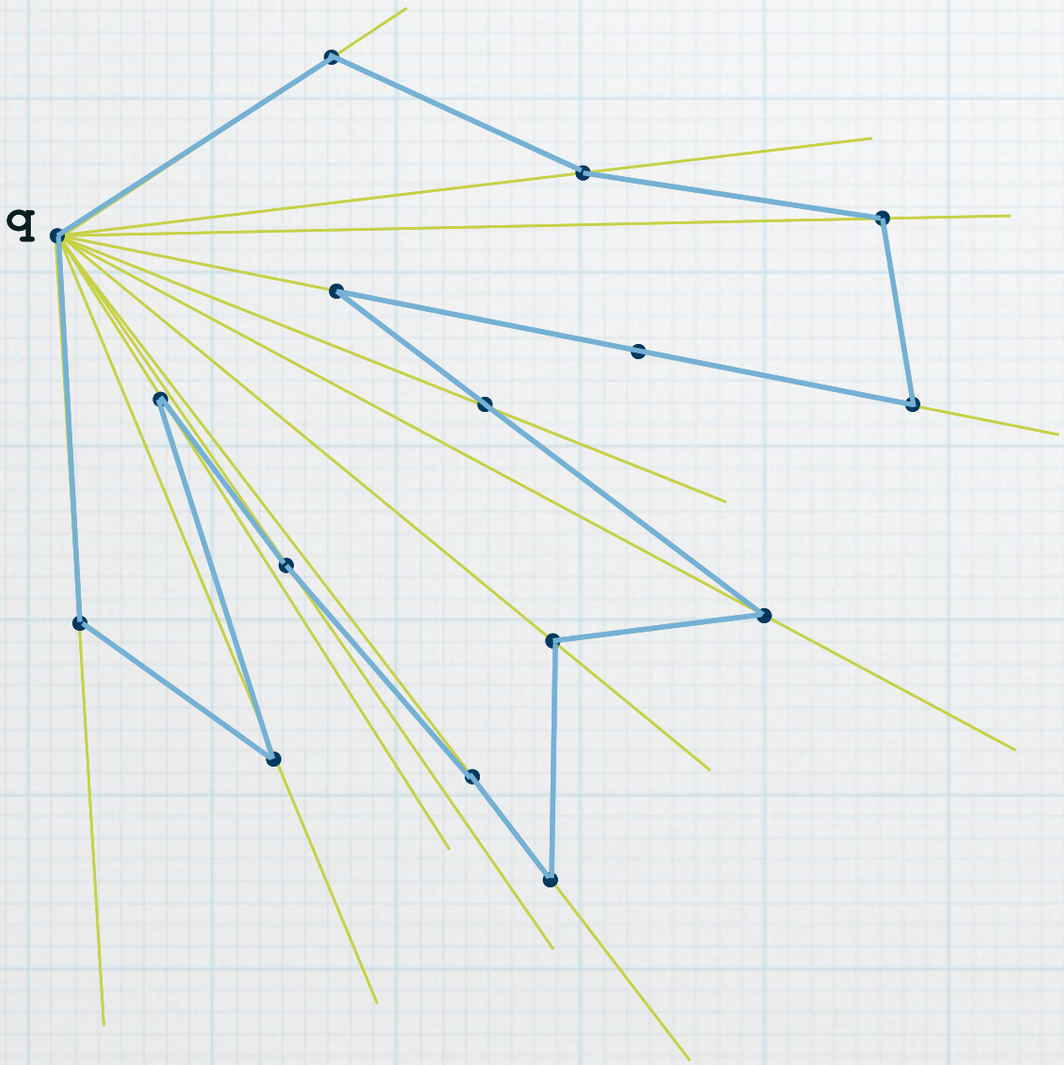
1. a ponthalmaz polárszög szerinti rendezése.

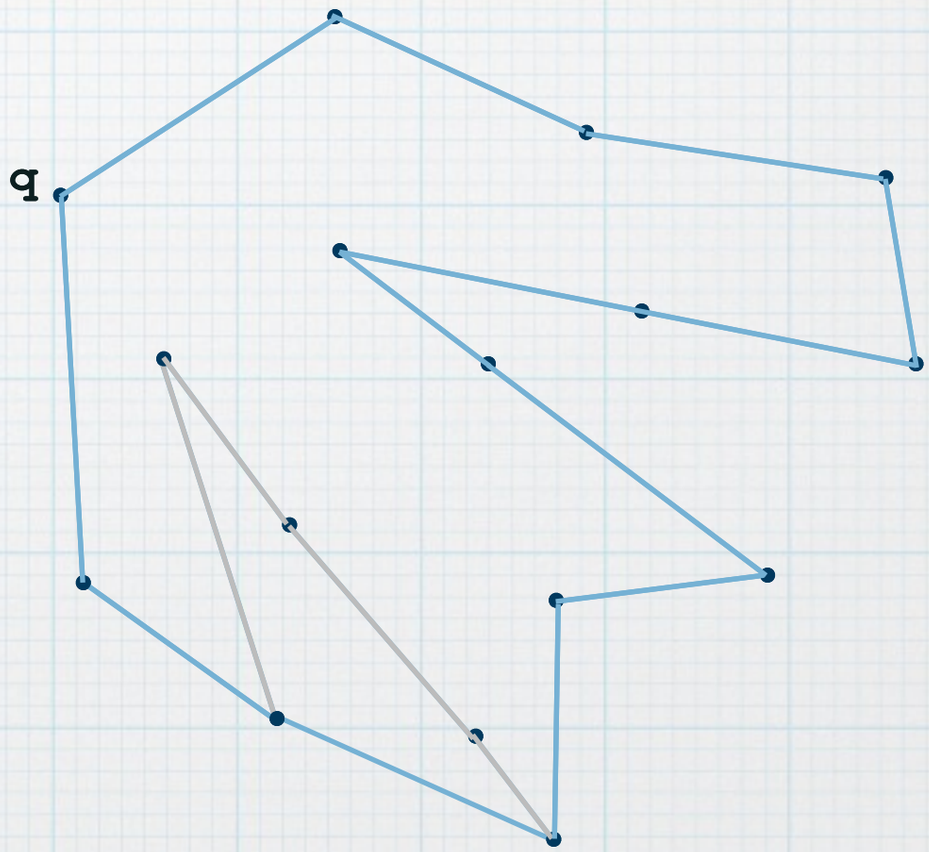




Megoldás

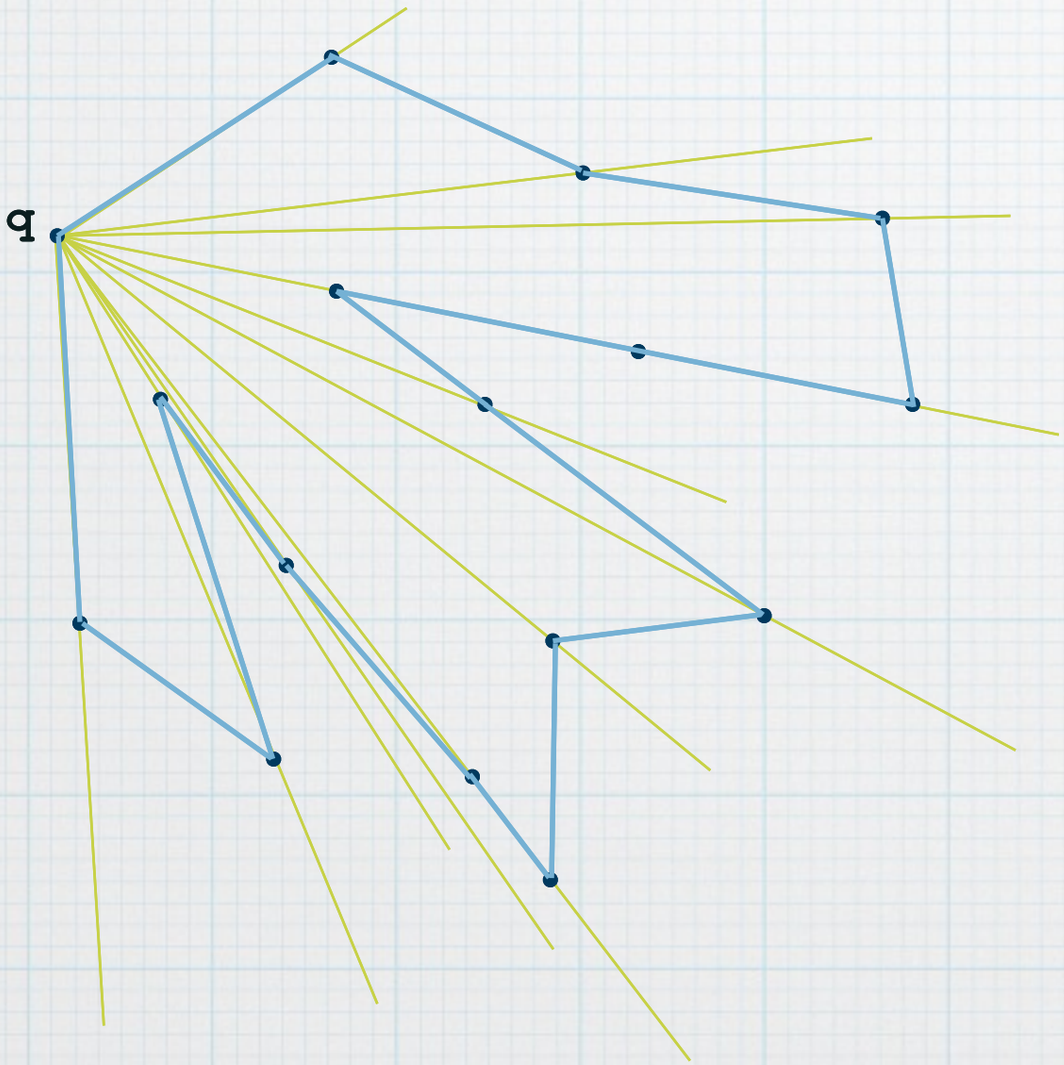
1. a ponthalmaz polárszög szerinti rendezése.

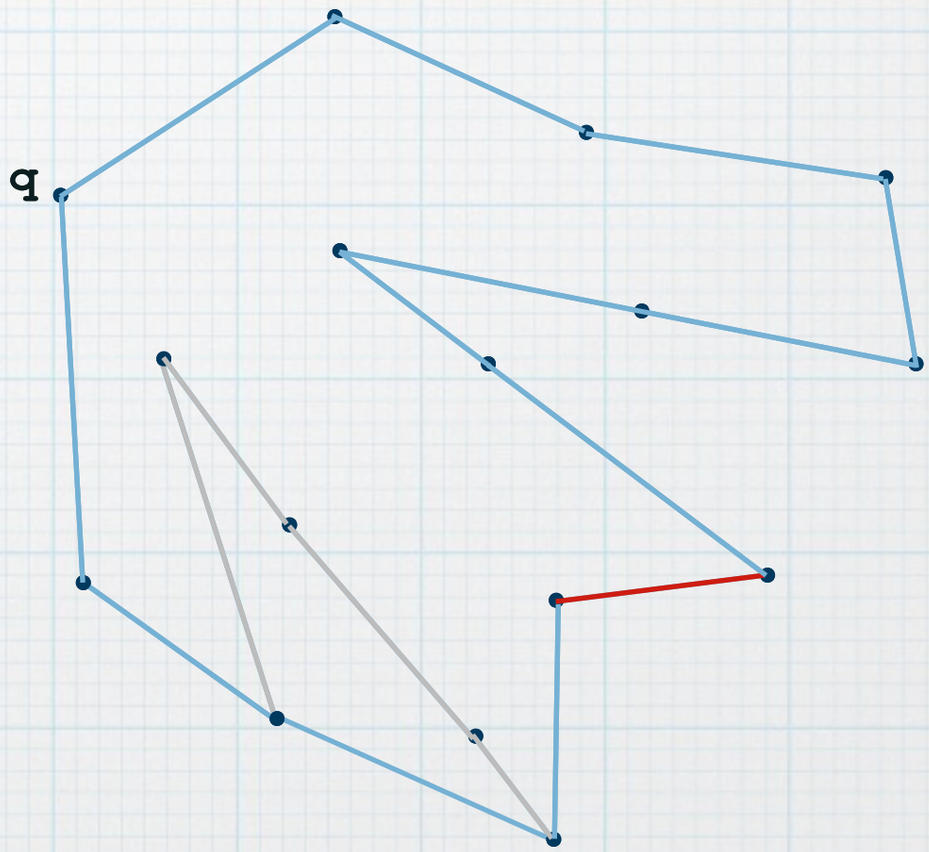




Megoldás

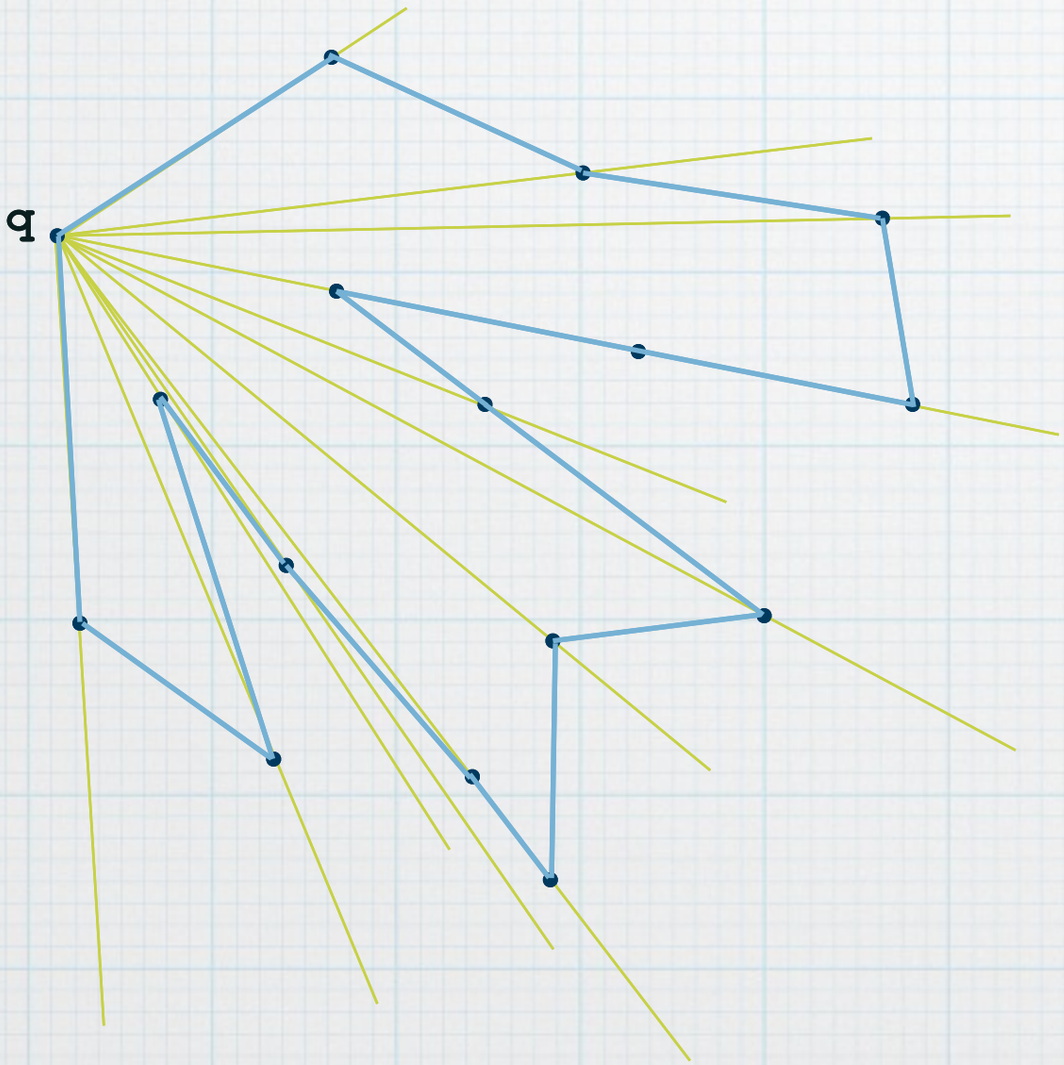
1. a ponthalmaz polárszög szerinti rendezése.

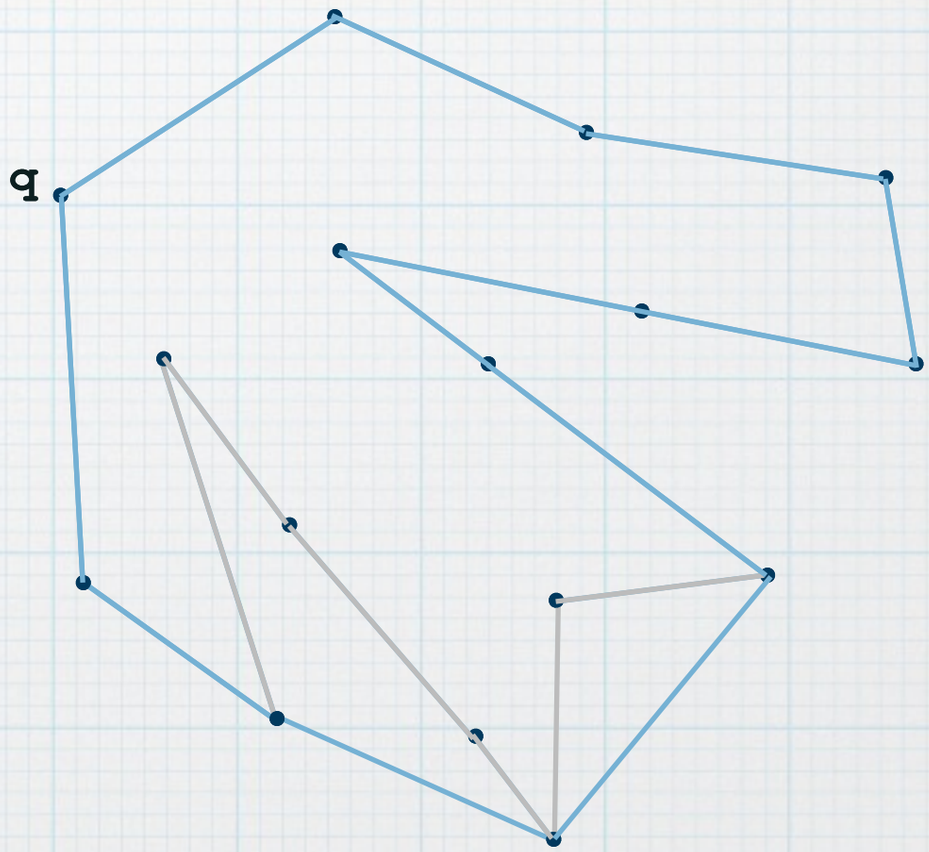




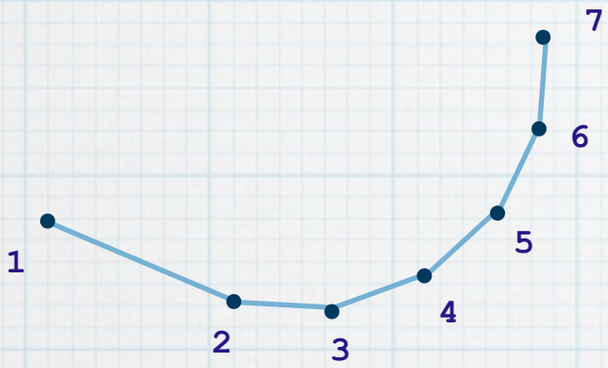
Megoldás

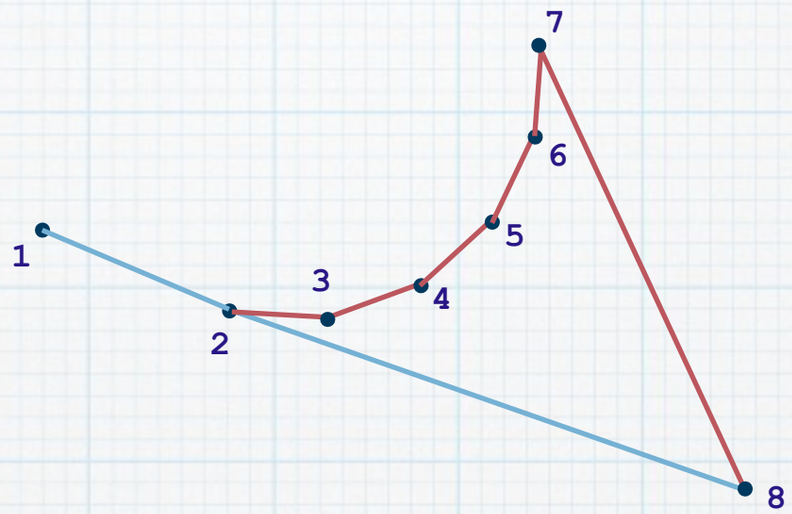
1. a ponthalmaz polárszög szerinti rendezése.

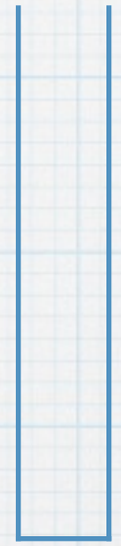


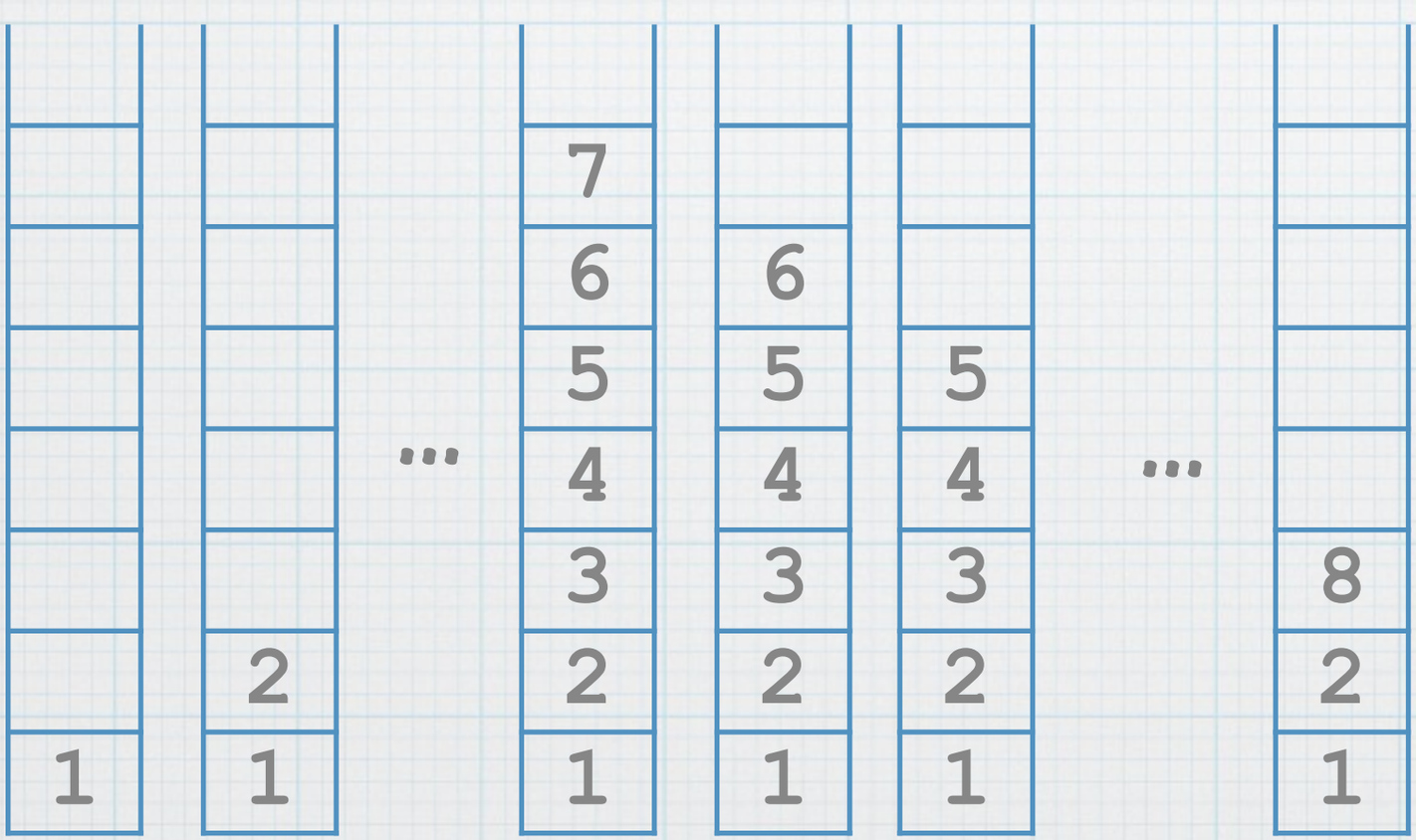


Megoldás

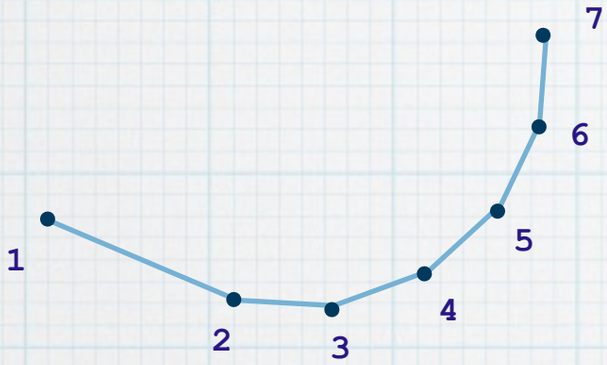


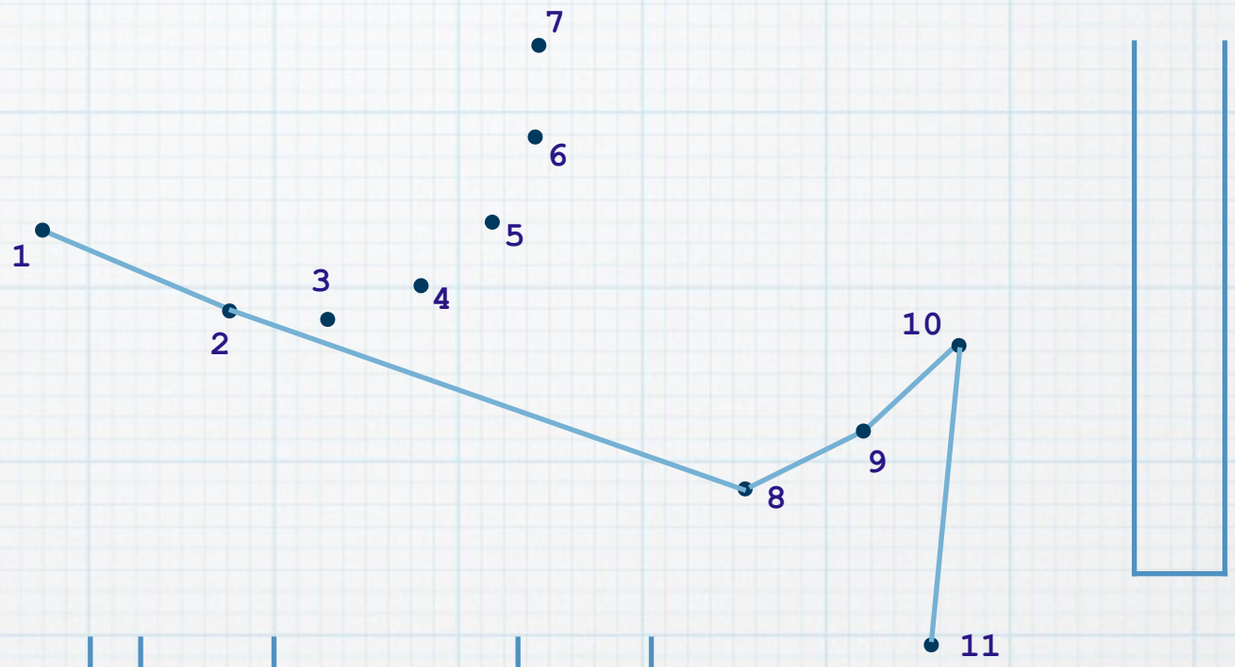


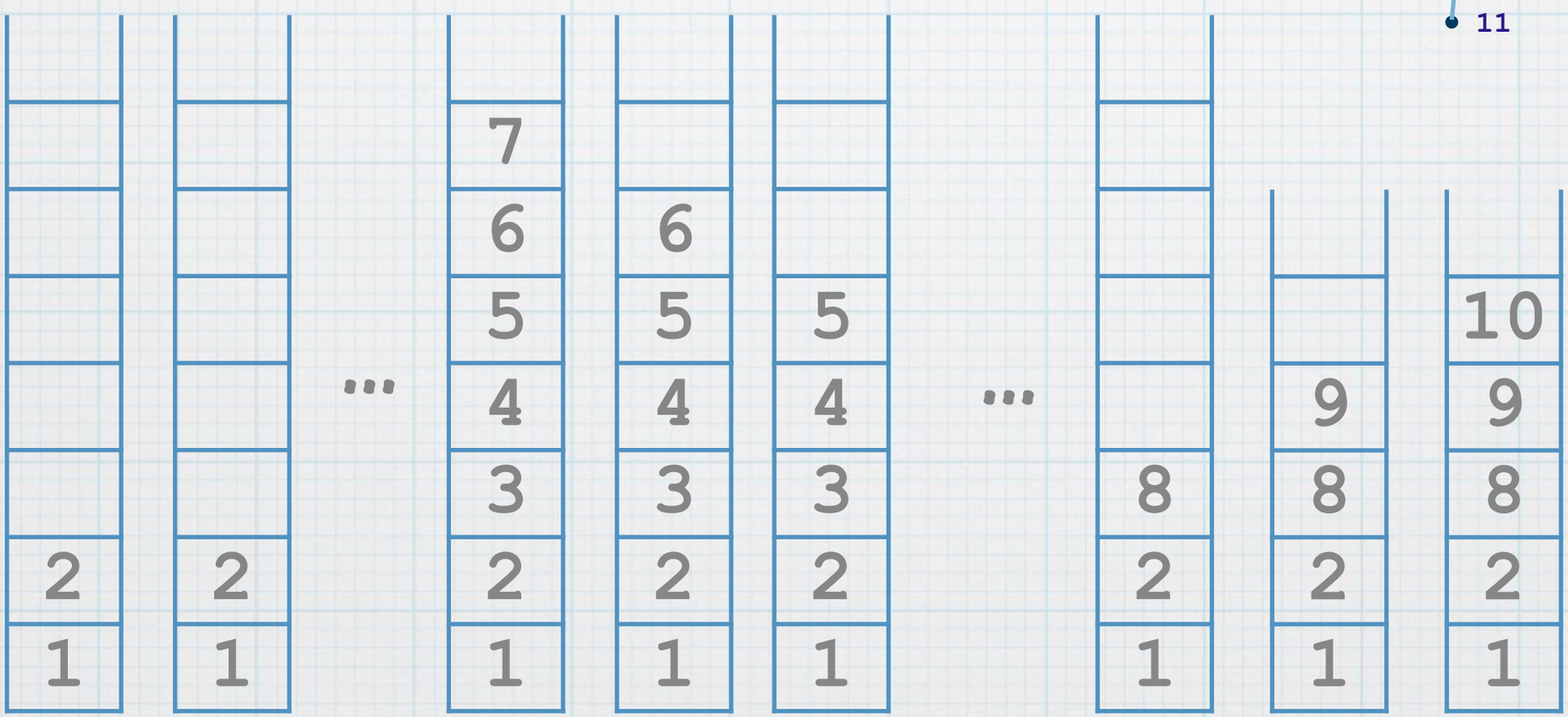




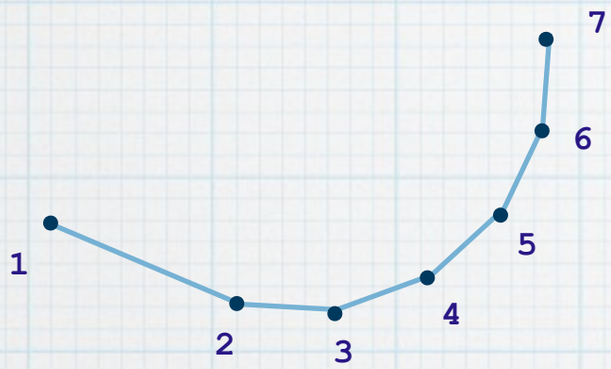
Megoldás


1
2
3
4
5
6
7

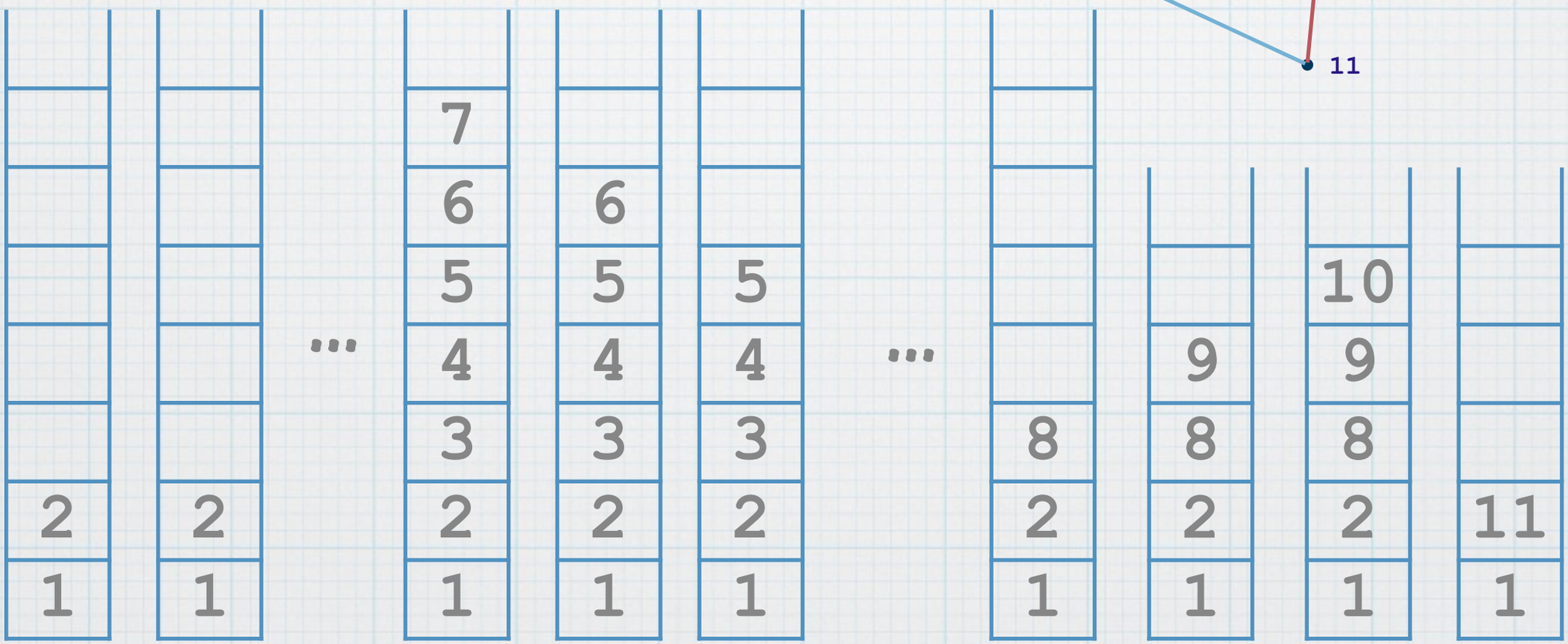

1
2
3
4
5
6
7
8
9
10
11


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

Megoldás


1
2
3
4
5
6
7


0
1
2
3
4
5
6
7
8
9
10
11


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

Megoldás

Konvex burok meghatározása

1. A ponthalmaz polárszög szerinti rendezése. O(nlogn)
2. A pontokat sorra vesszük a rendezés sorrendjében. + O(n)

3. Ha a verem tetején lévő pont, az aktuális és a következő pont forgásiránya + O(n)

   1 vagy 0: az aktuális pontot betesszük a verembe

   -1: kivesszük a pontot a veremből és visza 3.-hoz

4. Az utolsó pont után mégegyszer vesszük q-t, majd megállunk.
   A veremben lévő pontok a konvex burok pontjai.

= O(nlogn)

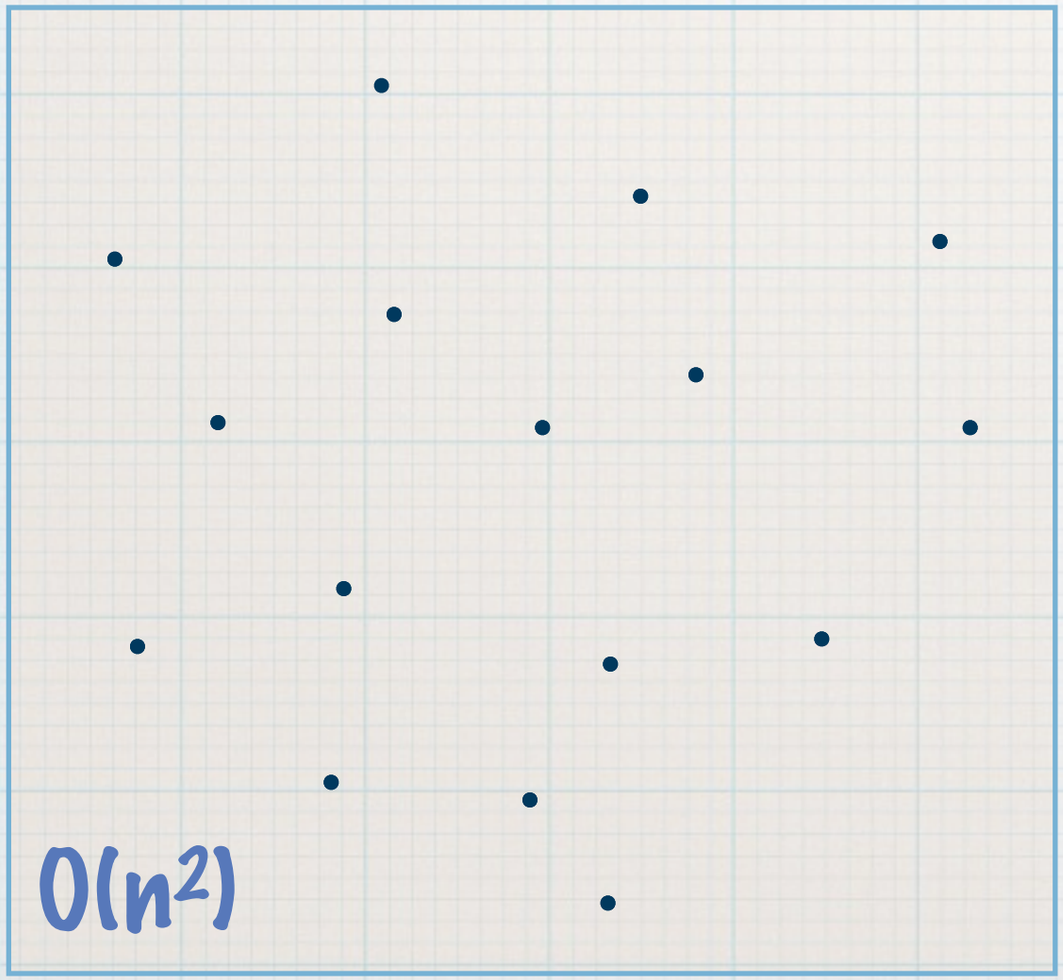
Feladat

Legtávolabbi pontpár meghatározása

Adott egy $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ ponthalmaz a síkon. Szokásos módon, a $p_i = (x_i, y_i)$ és $p_j = (x_j, y_j)$ pontok távolsága

$$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}.$$

Adjunk hatékony algoritmust a két legtávolabbi pont meghatározására!



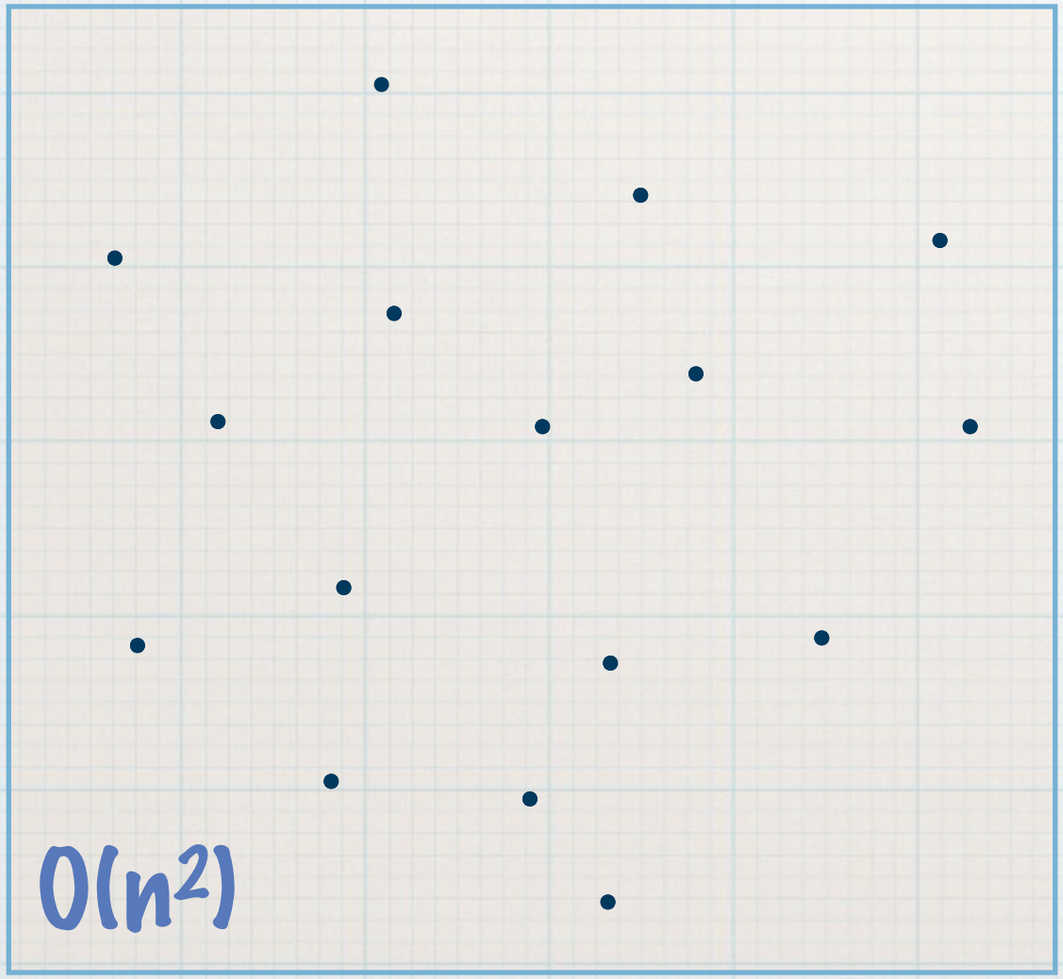
Feladat

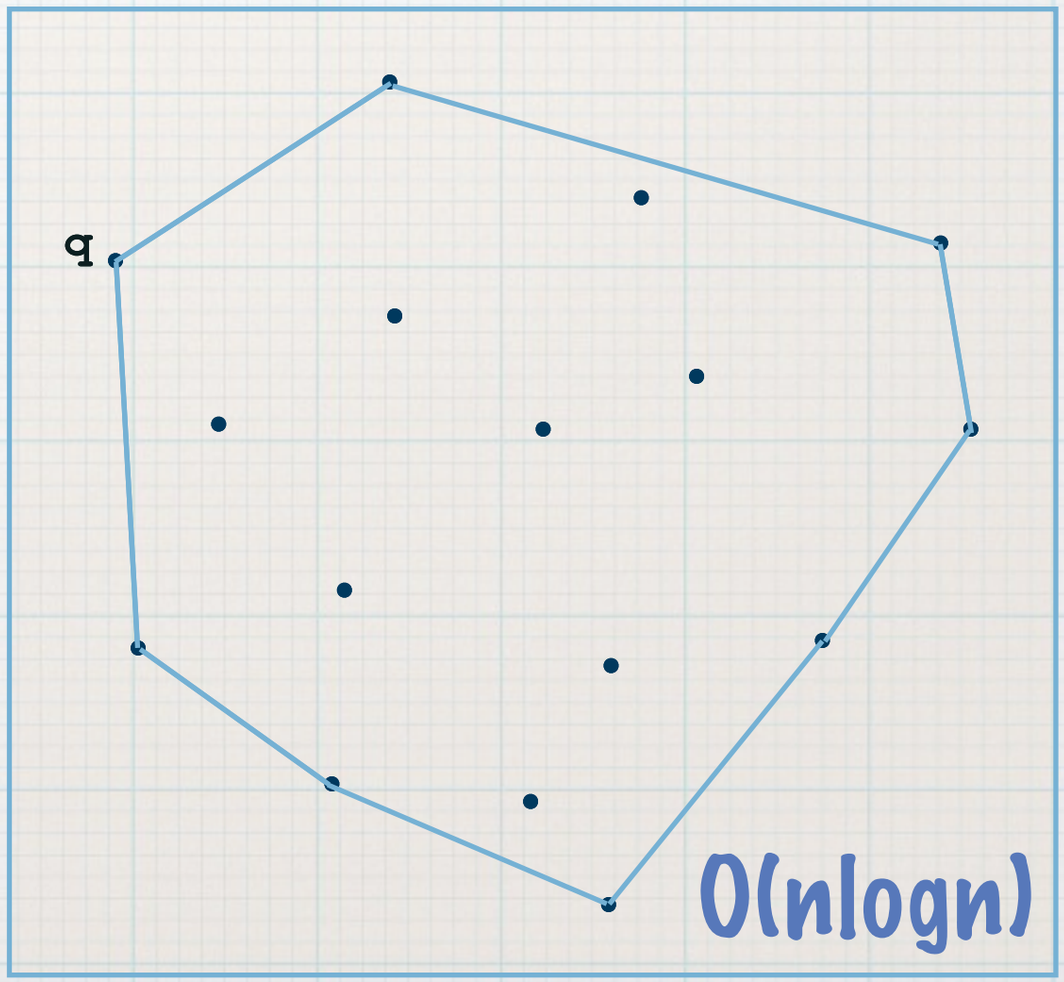
Legtávolabbi pontpár meghatározása

Adott egy $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ ponthalmaz a síkon. Szokásos módon, a $p_i = (x_i, y_i)$ és $p_j = (x_j, y_j)$ pontok távolsága

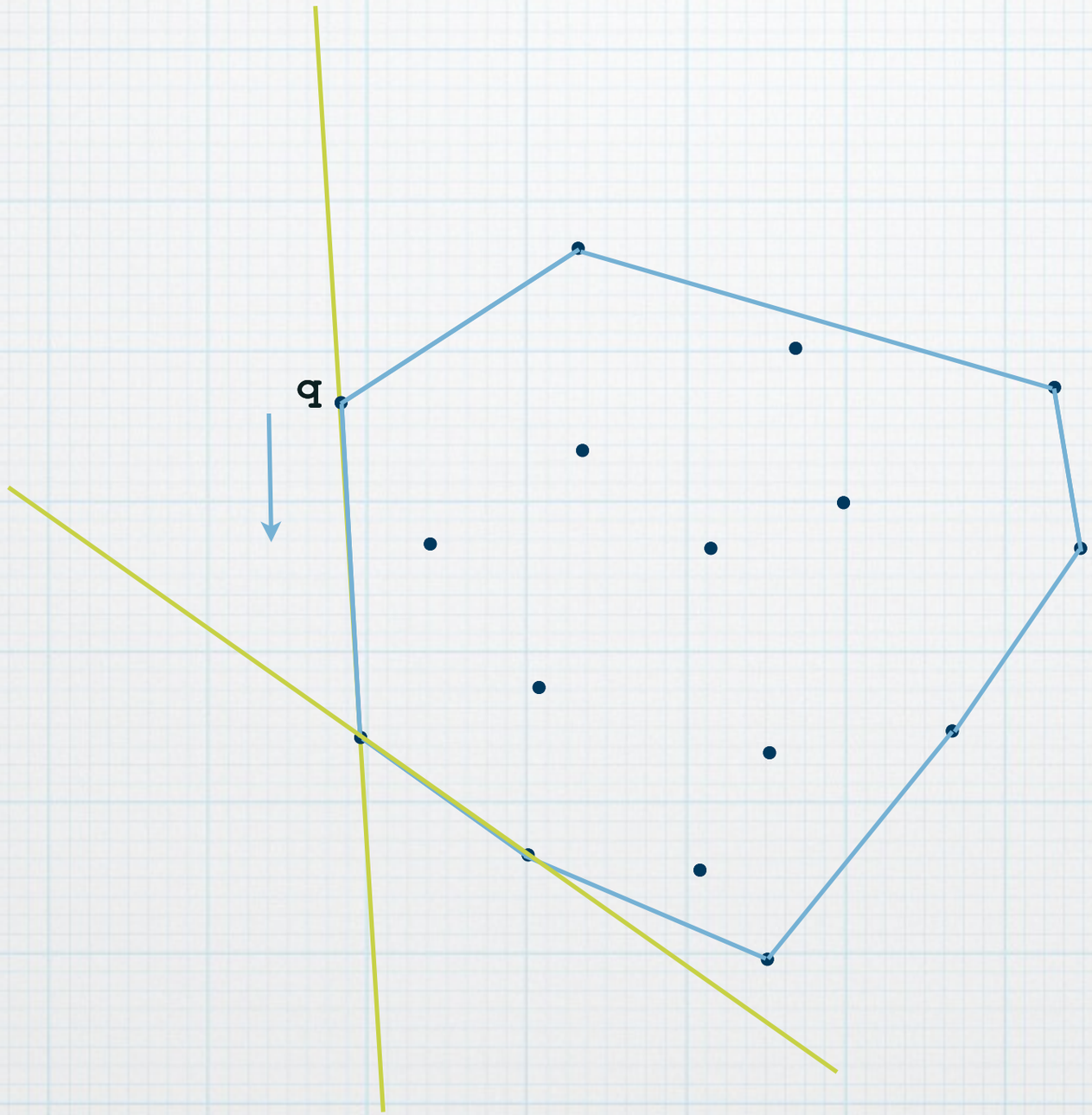
$$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}.$$

Adjunk hatékony algoritmust a két legtávolabbi pont meghatározására!

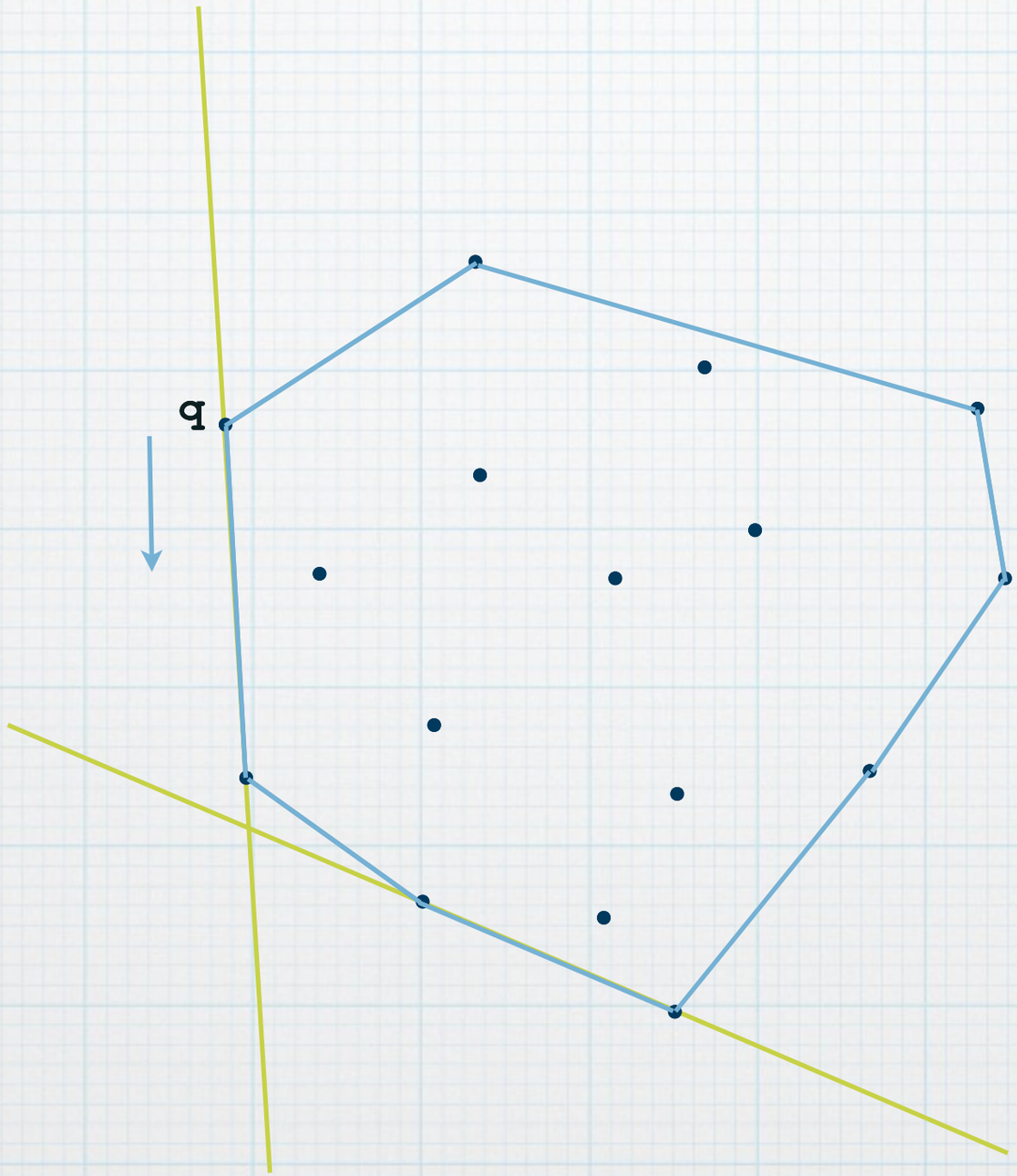




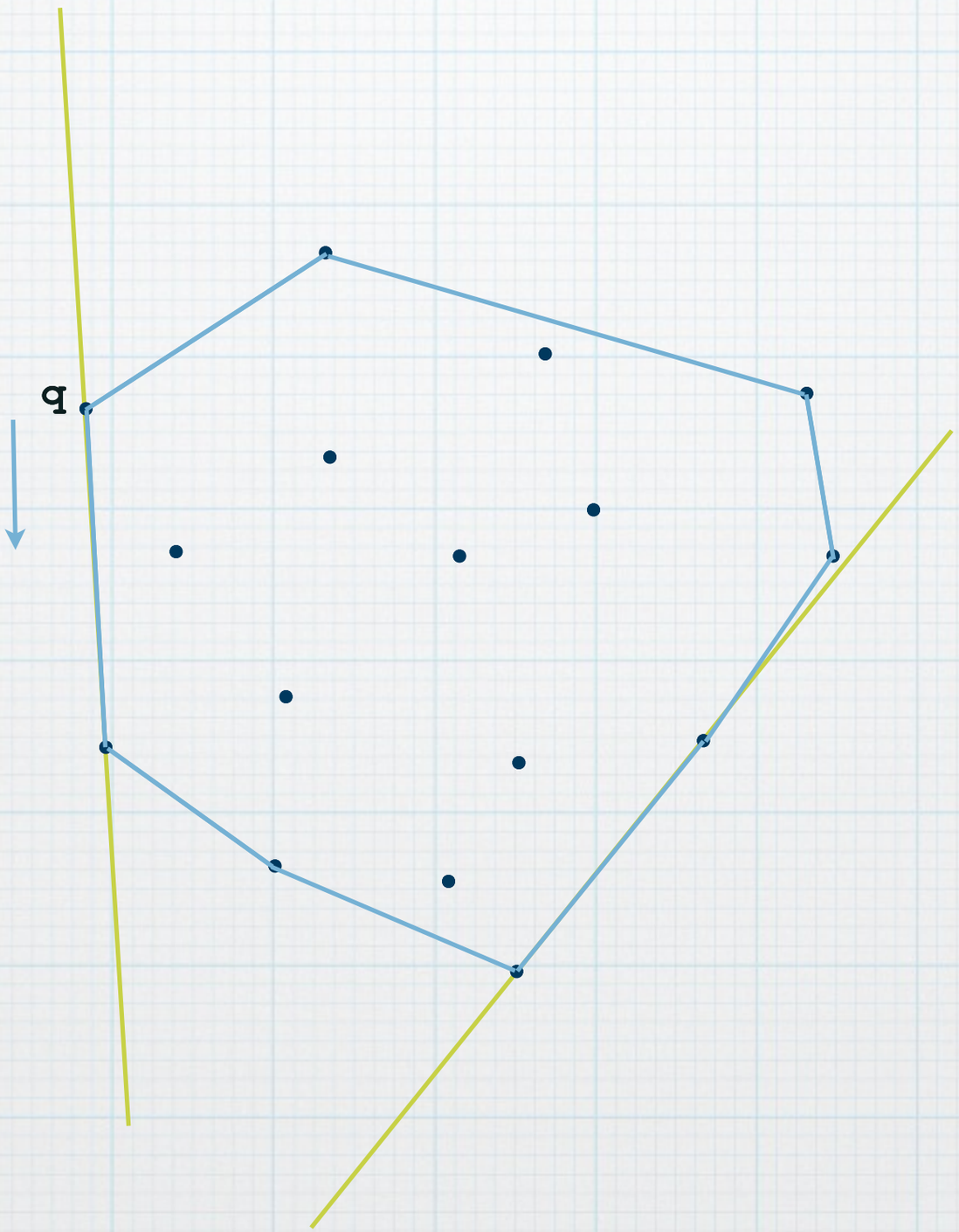
Megoldás



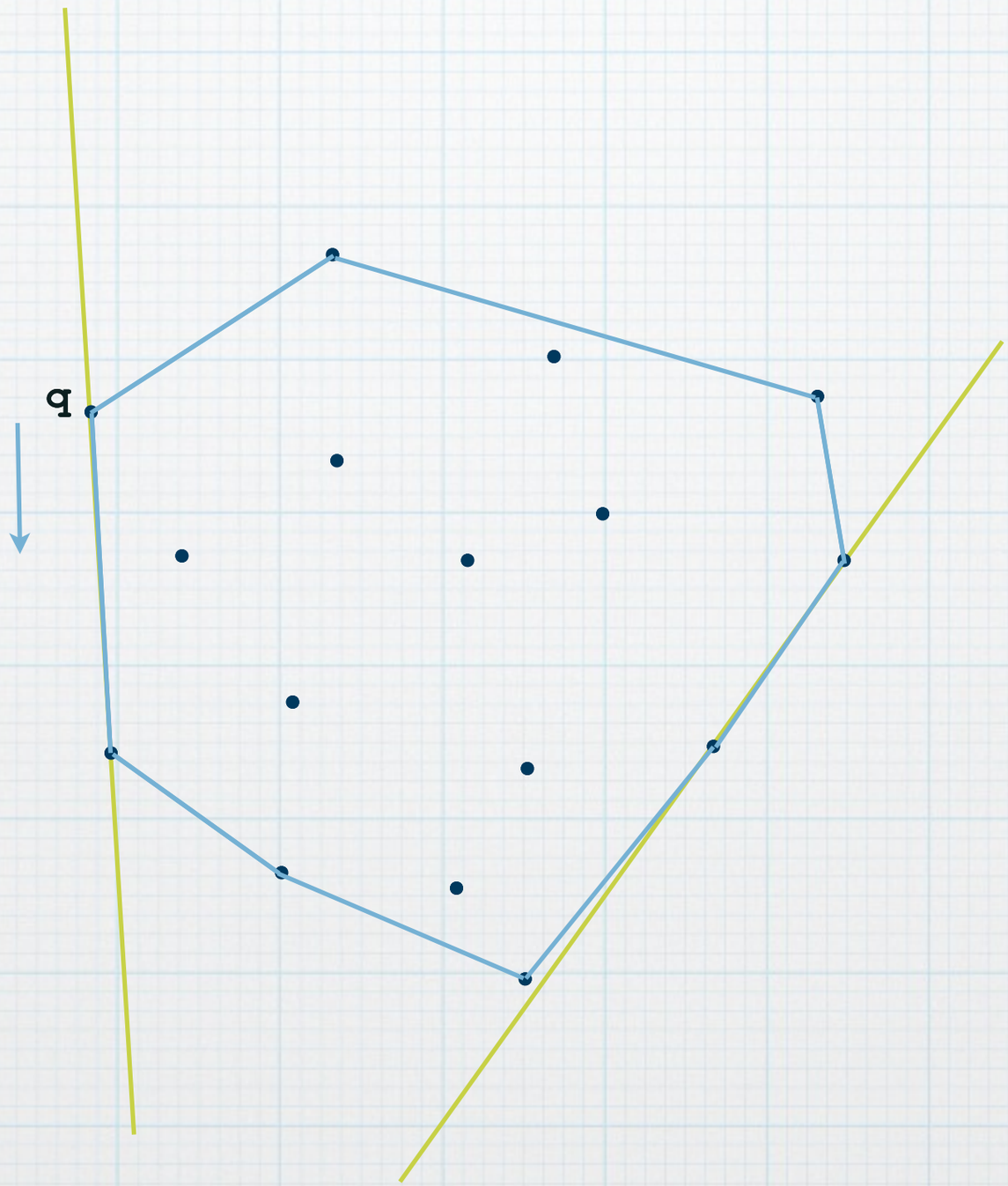
Megoldás



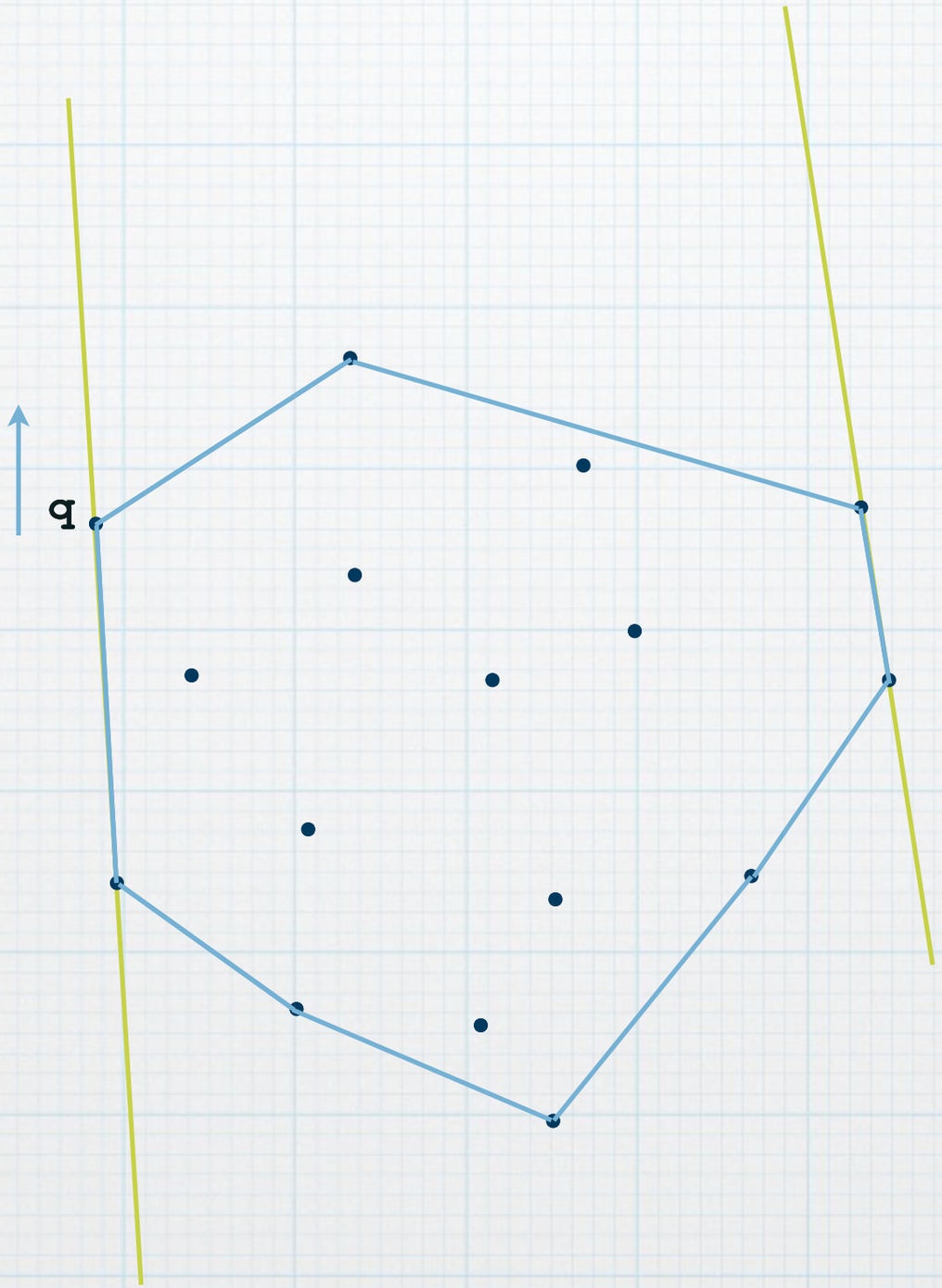
Megoldás



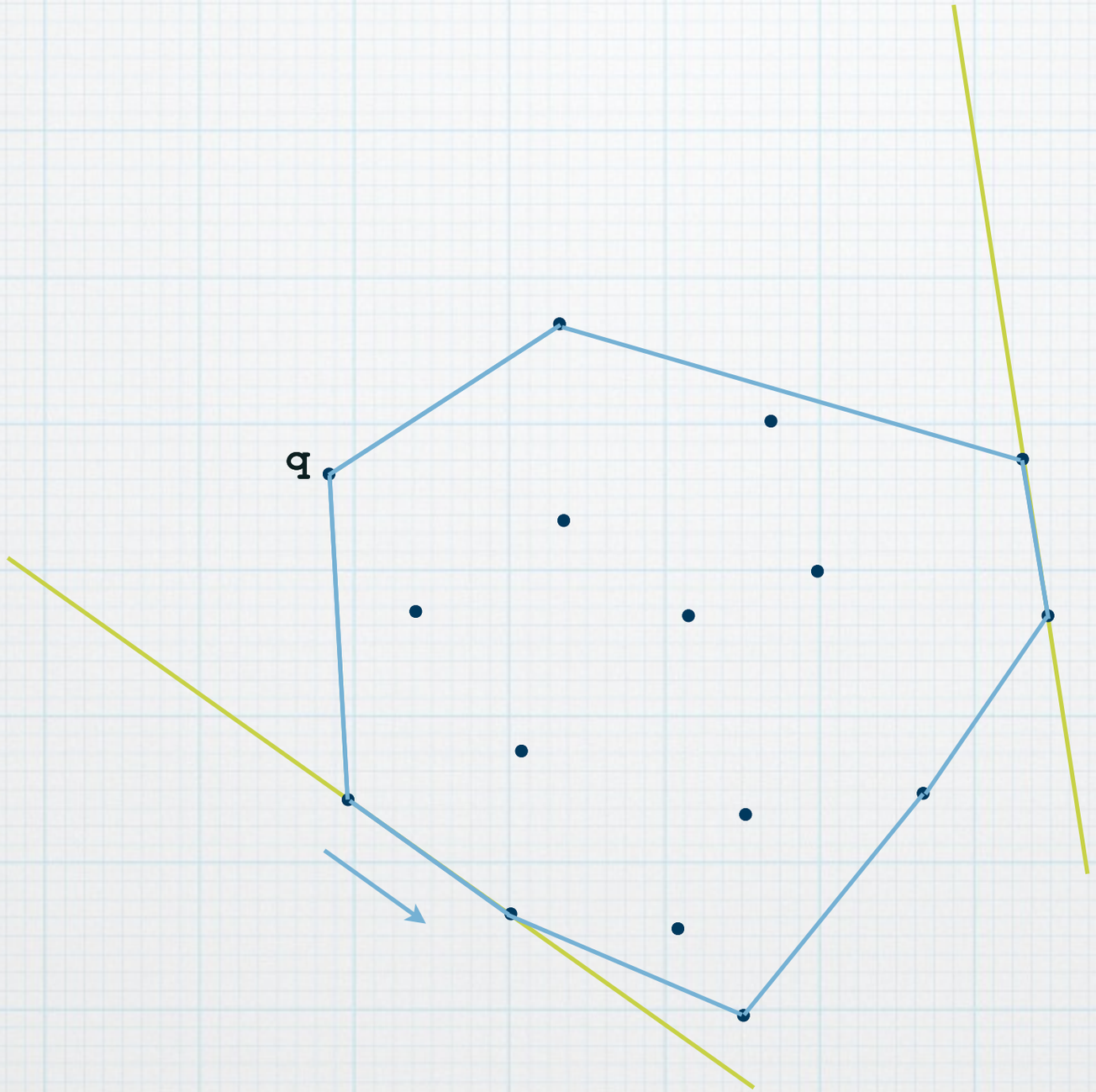
Megoldás



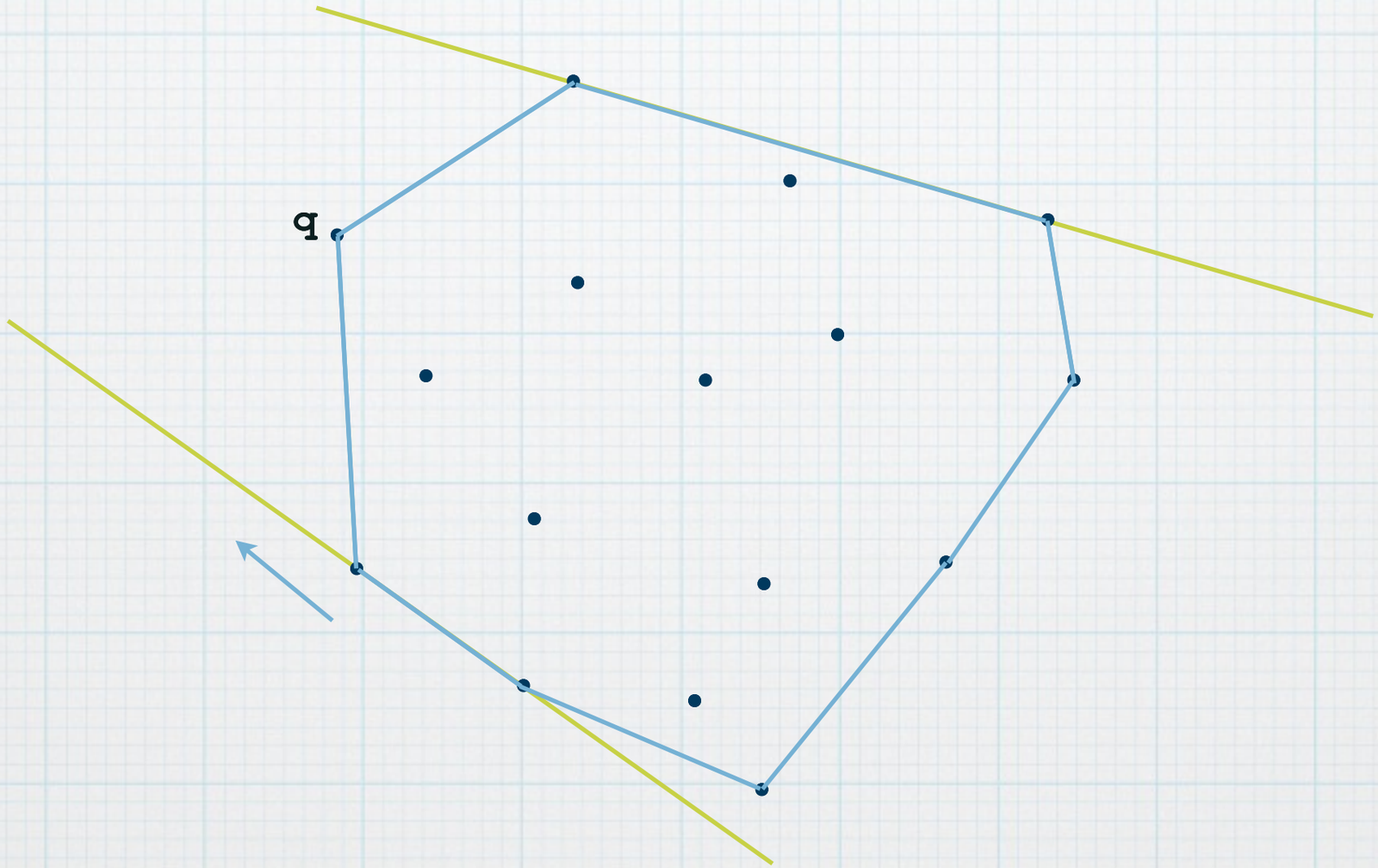
Megoldás



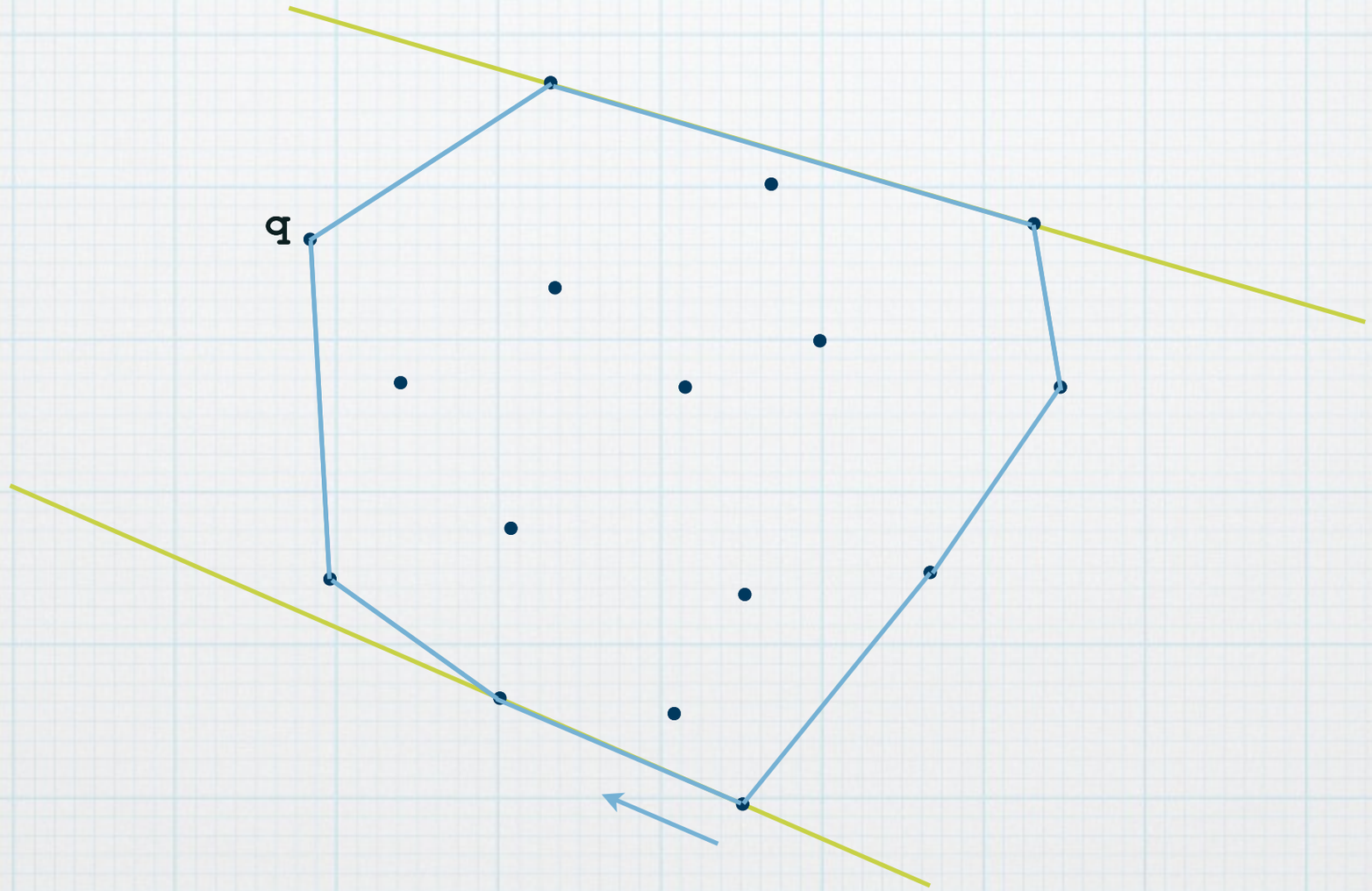
Megoldás



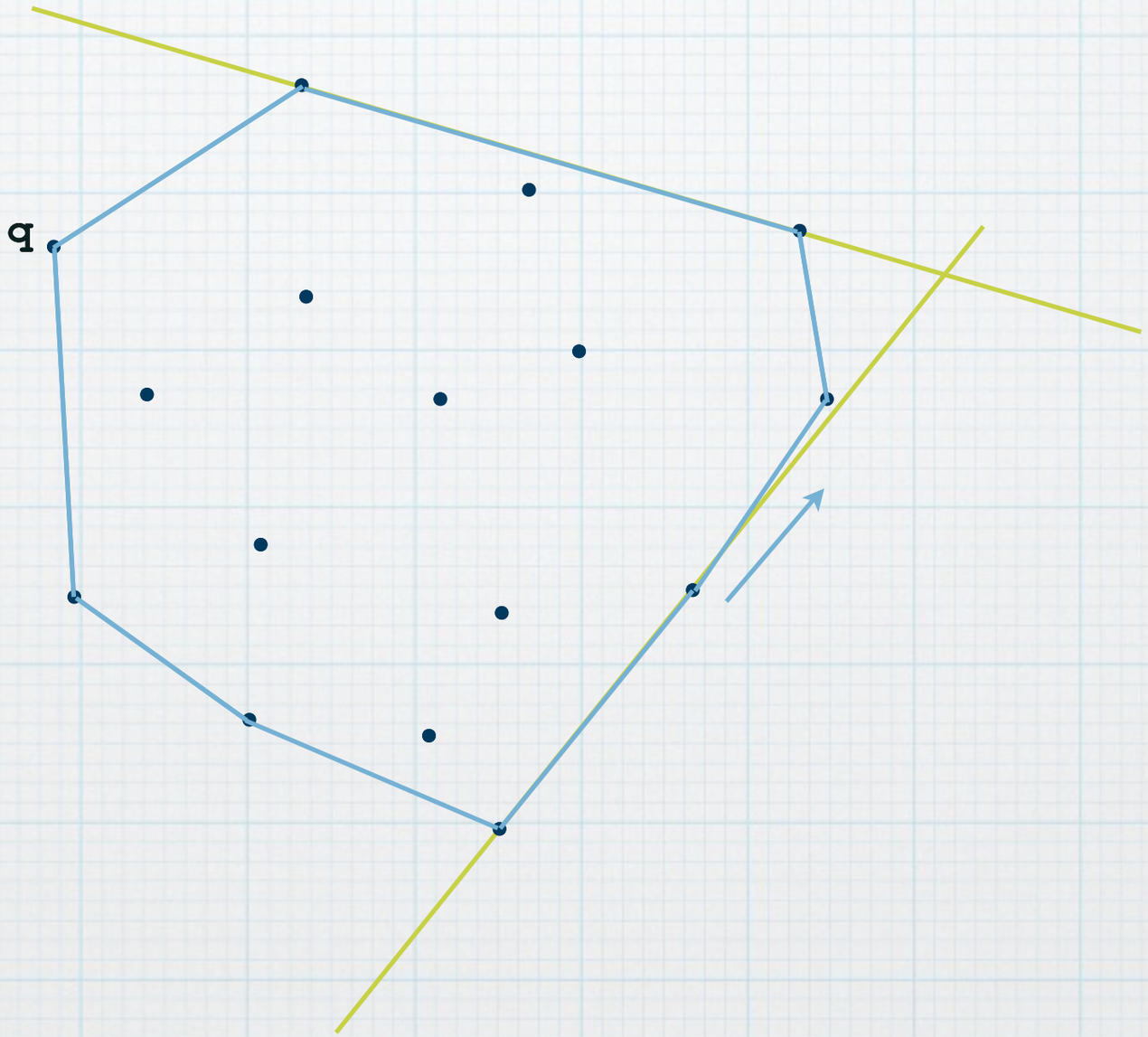
Megoldás



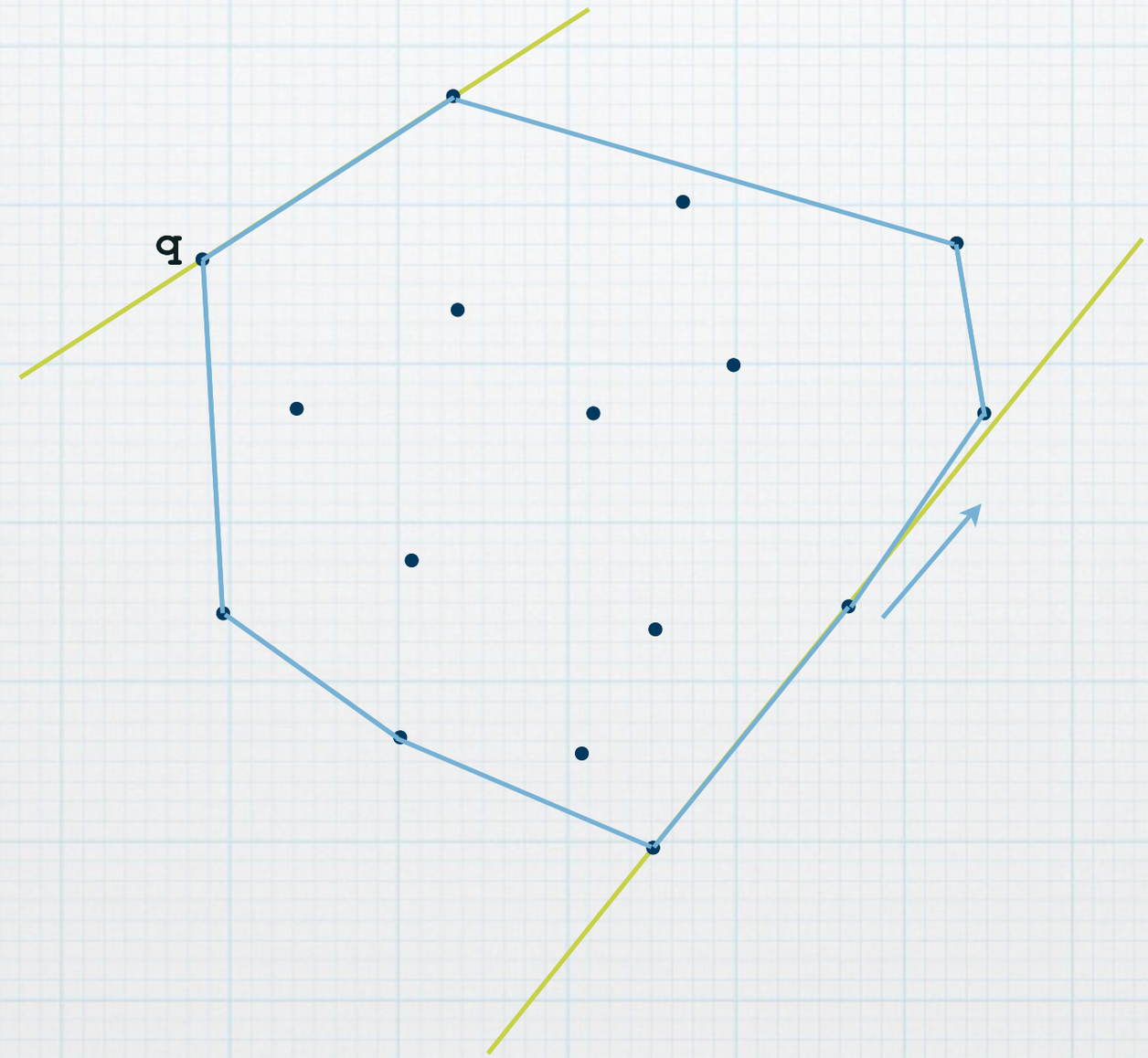
Megoldás



Megoldás

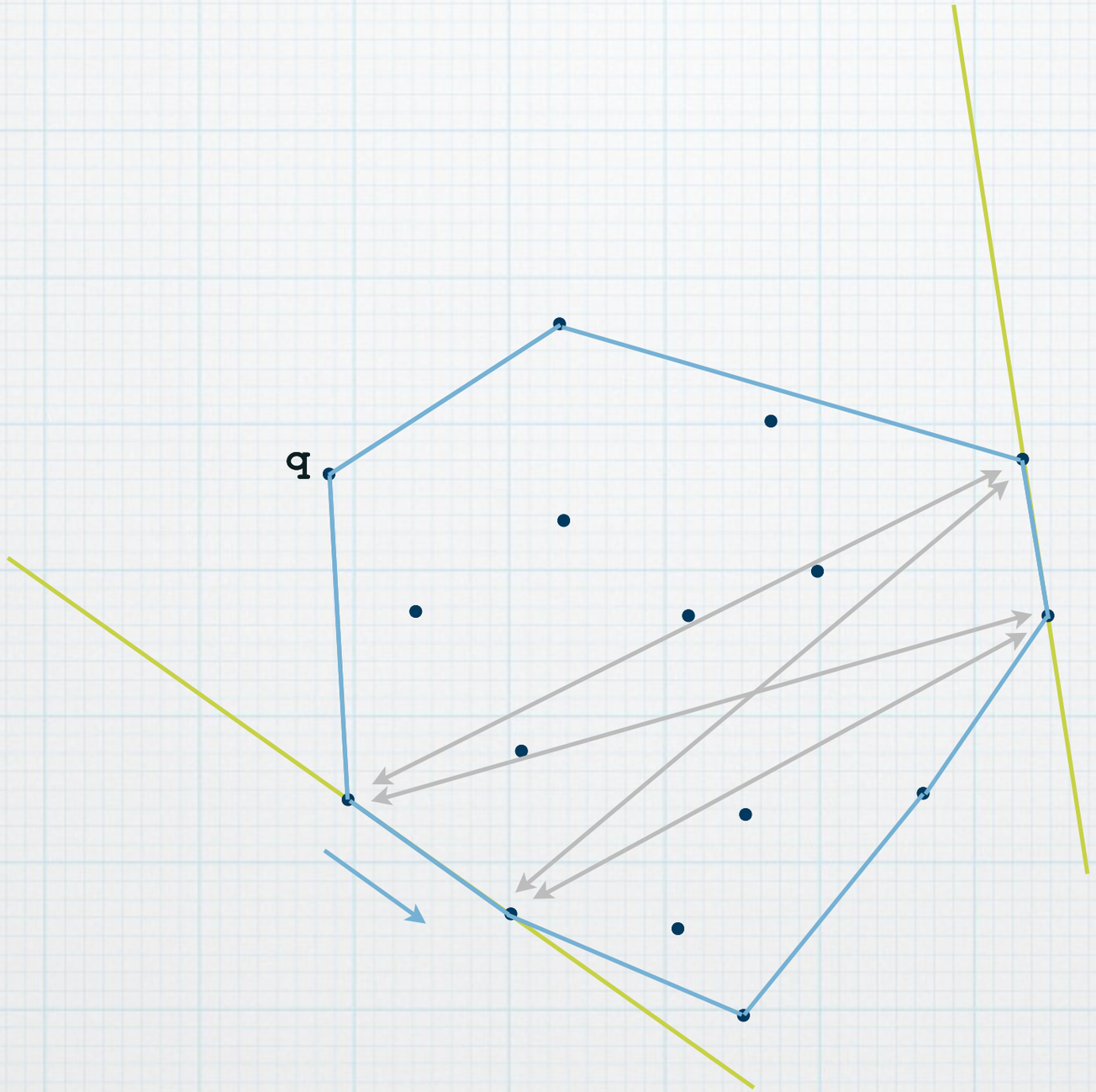


Megoldás



O(n)

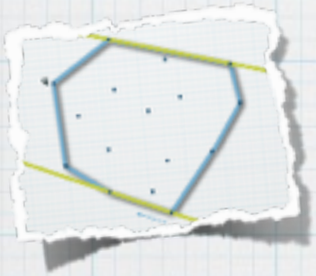
Megoldás



Megoldás

Legtávolabbi pontpár meghatározása

1. A konvex burok meghatározása. O(nlogn)



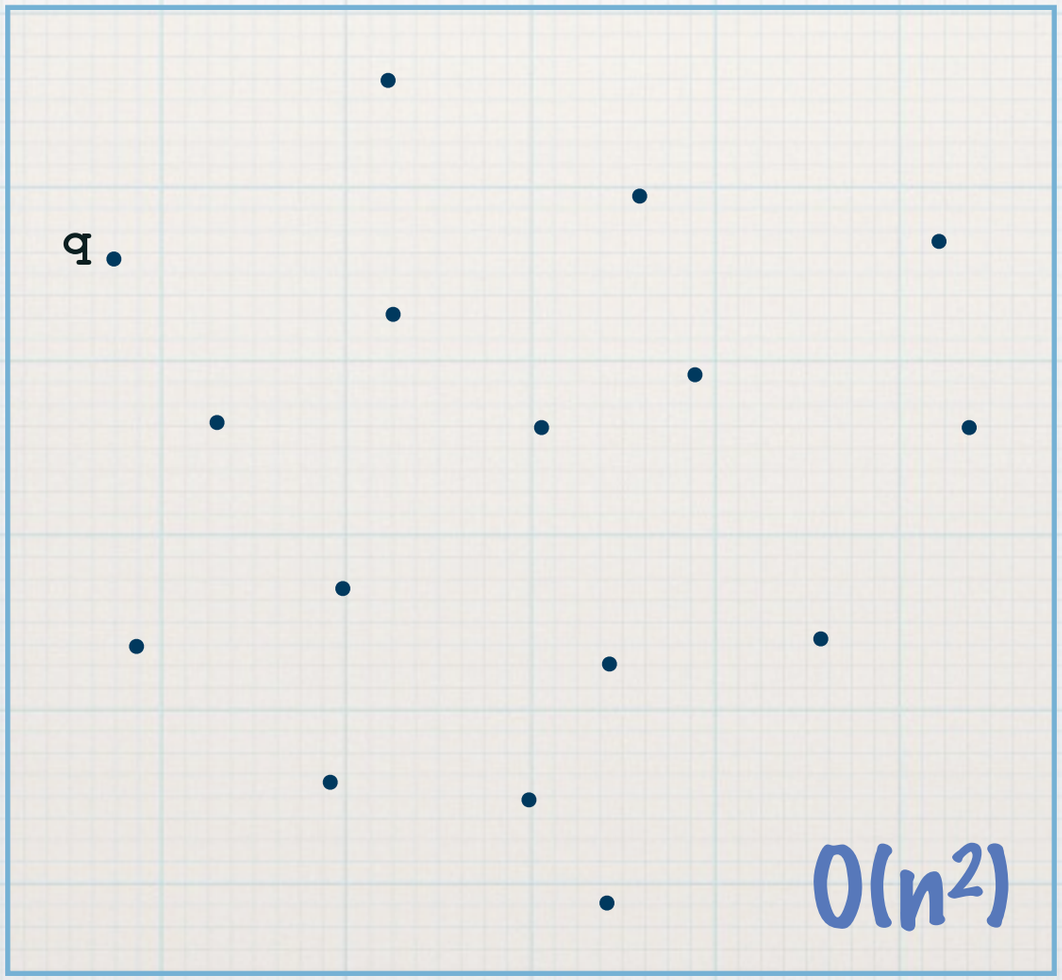
2. A 2 egyenes végiggörgetése és közben maximumkeresés az érintett pontok távolságnégyzetein. + O(n)

O(nlogn)

Feladat

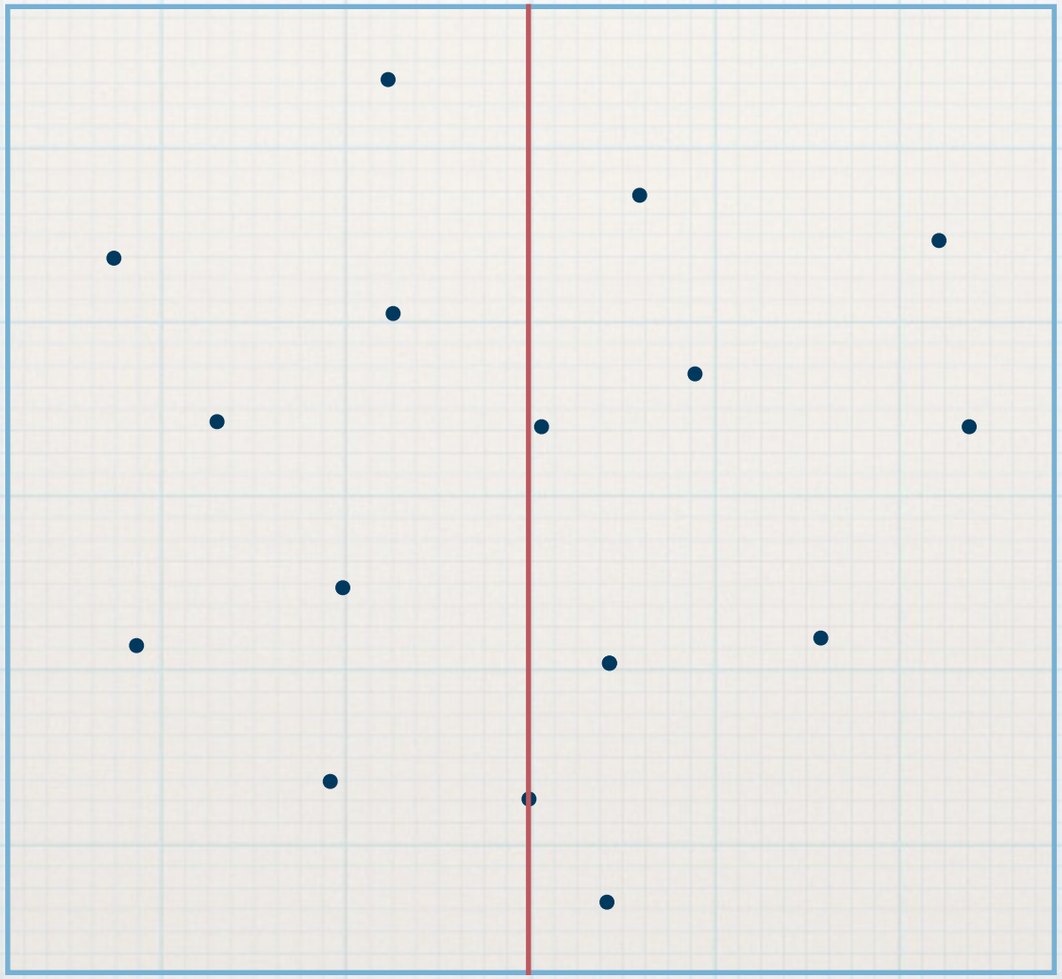
Legközelebbi pontpár meghatározása

Nézzük végig az összes pontpárt és válasszuk ki hol lesz a távolság (négyzete) minimális?

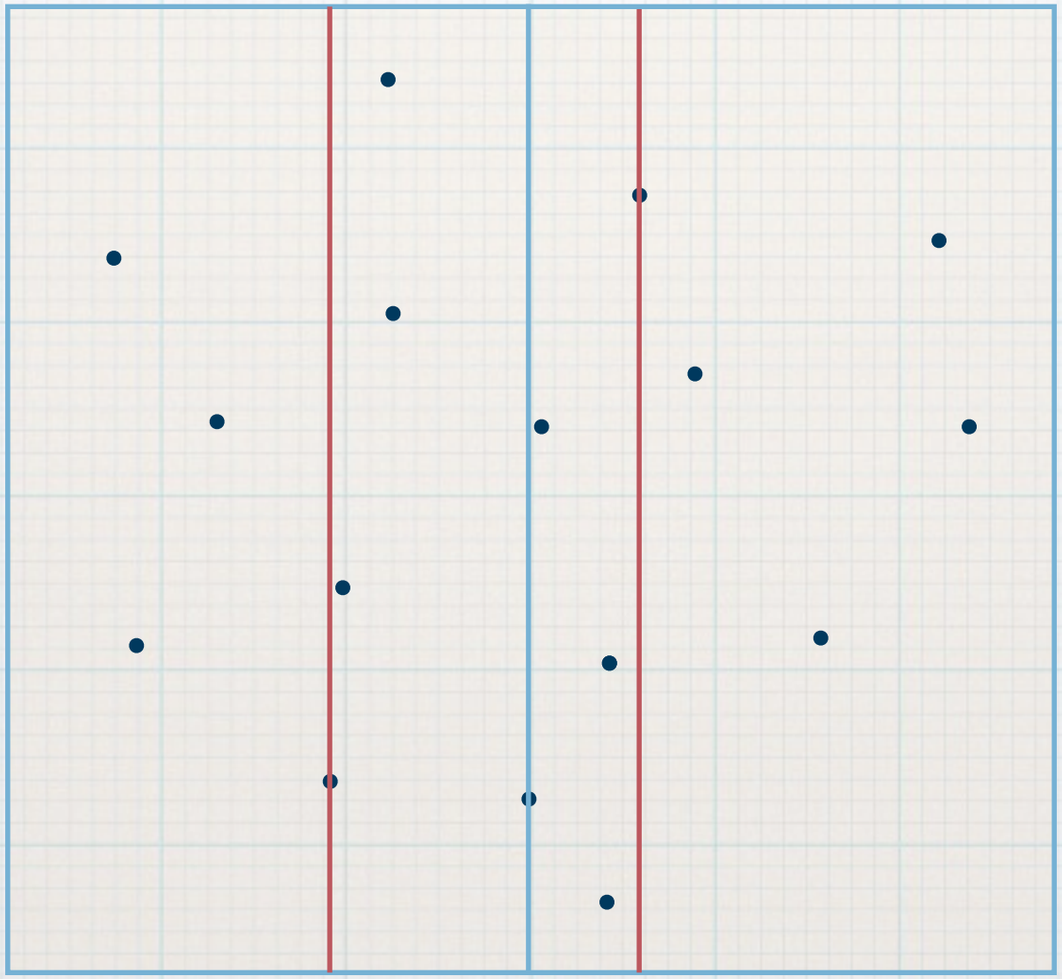


Megoldás

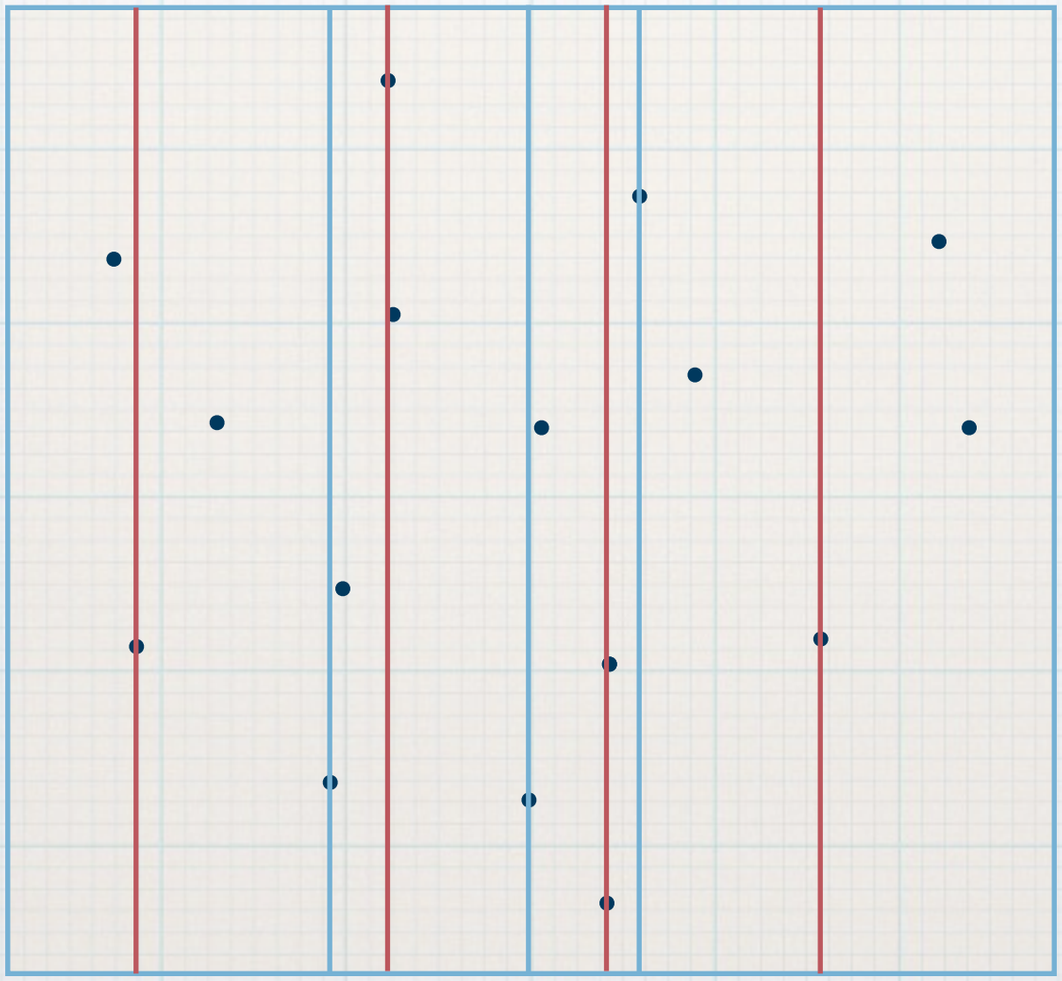
Oroszlán keresése a sivatagban



Megoldás



Megoldás



Minden rekurzív hívásnál átadásra kerülnek a P ponthalmaz egy Q részhalmazának pontjai egy az x koordináták szerint monoton növekvően rendezett X tömbben (azonos x koordinátájú pontok esetén a kisebb y koordinátájú jön előbb), valamint egy az y koordináták szerint monoton növekvően rendezett Y tömbben (azonos y koordinátájú pontok esetén a kisebb x koordinátájú jön előbb).
Hogy ne kelljen minden egyes rekurzív hívás előtt rendezésre pazarolni az időt, ezért már az algoritmus elején előállítjuk P pontjainak az x, illetve az y koordináták szerint monoton növekvően rendezett tömbjeit, így a rekurzív hívások előtt csak kiválogatásokat kell végezni.

Egy adott rekurzív lépés a következő.
Ha |Q| < 3, akkor páronkénti vizsgálattal határozzuk meg a minimális távolságot.
Természetesen az érdekes eset, amikor |Q| > 3, ilyenkor az alábbiak szerint járunk el:

Először egy olyan e függőleges egyenest keresünk, amely Q-t két olyan Q_L és Q_R részre osztja, amelyekre:

- $|Q_L| = \lceil |Q|/2 \rceil$ és $|Q_R| = \lfloor |Q|/2 \rfloor$
- Q_L minden pontja az egyenes bal oldalán van, vagy illeszkedik az egyenesre
- Q_L minden pontja az egyenes jobb oldalán van, vagy illeszkedik az egyenesre

Válogassuk szét a pontokat az X tömbből az X_L és X_R tömbökbe: X_L-be kerüljenek a Q_L-beli pontok, X_R-be pedig a Q_R-beliek. Természetesen az X_L és X_R tömbök is rendezettek az x koordináták szerint.
Válogassuk szét a pontokat az Y tömbből is az Y_L és Y_R tömbökbe a fentiekhez hasonlóan!

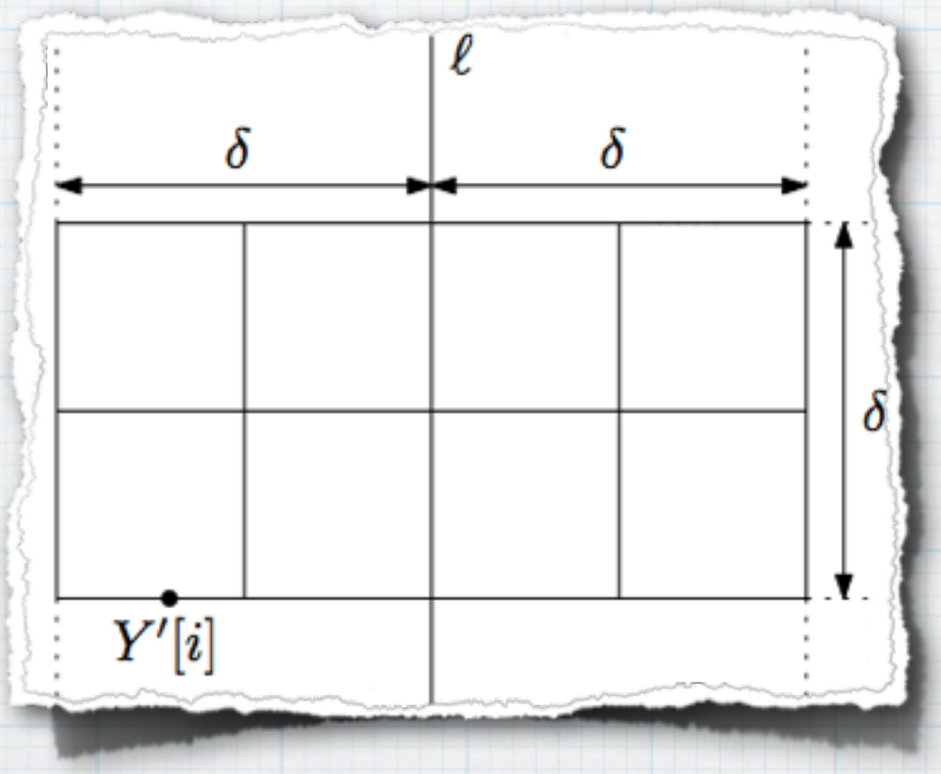
Ezután keressük meg rekurzívan a Q_L és Q_R halmazokban a legközelebbi pontpárt.
Az első hívás bemenete az X_L és Y_L tömbök, míg a második hívásé az X_R és Y_R tömbök. Legyen Q_L-ben, illetve Q_R-ben a minimális távolság δ_L, illetve δ_R, és legyen $\delta = \min(\delta_L, \delta_R)$

Most a legközelebbi pontpár:

- vagy az egyik rekurzív hívás által megtalált δ távolságú páros,
- vagy egy olyan pár, amelynek egyik pontja Q_L-ben, a másik Q_R-ben van.

A következő lépés annak meghatározása, hogy ez utóbbi párok között van-e olyan, amelyben a pontok távolsága kisebb, mint δ. Vegyük észre, hogy ha van ilyen pár, akkor annak egyik pontja sem lehet δ-nál nagyobb távolságra e-től.
Válogassuk ki az Y tömbből azokat a pontokat, amelyek e-től mért távolsága legfeljebb δ. Jelölje a kapott tömböt Y'.
Természetesen az Y' tömb is rendezett az y koordináták szerint.

Most minden egyes $p \in Y'$ ponthoz keressük meg Y' azon pontjait, melyek p-től mért távolsága legfeljebb δ, és amelyek a p-n átmenő vízszintes egyenesen vagy az fölött vannak.
Legfeljebb 7 ilyen pont jöhet számításba:
Ha $i < j$ és $Y'[i]$ valamint $Y'[j]$ távolsága legfeljebb δ, akkor ez a két pont az ábrán látható $\delta \times 2\delta$ oldalú téglalapban van.



Mivel a téglalapban lévő 8 kis négyzet egyikében sem lehet egynél több pont Y'-ből, az állítás adódik.

így minden $p \in Y'$ pontra elég kiszámolni p távolságát az Y'-ben utána következő 7 ponttól. Az így kapott távolságok minimuma legyen δ'. Ha $\delta' < \delta$, akkor Q-ban a minimális távolság δ', egyébként a minimális távolság δ.

$$T(n) = \begin{cases} 2T(n/2) + O(n) & \text{ha } n > 3, \\ O(1) & \text{ha } n \leq 3. \end{cases}$$

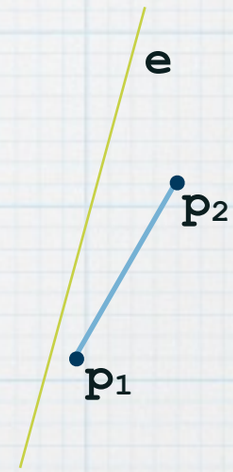
$$T(n) = O(n \log n).$$

Mivel az előrendezések is megvalósíthatók $O(n \log n)$ költséggel, ezért az algoritmus teljes költsége szintén $O(n \log n)$.

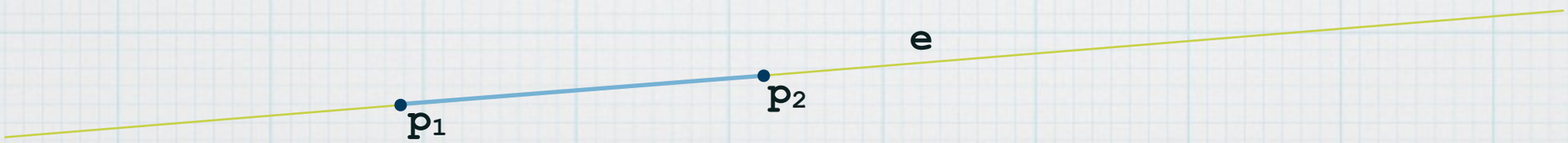
```
TAVKERES(P,X,Y) {
    if |P| ≤ 3 then return NAIV(P)
    Legyen PL, P ⌈|P|/2c⌉ darab legkisebb x koordinátájú eleme.
    Legyenek XL,YL a PL beli pontok rendezett halmazai.
    Legyen PR P többi eleme
    Legyenek XR,YR a PR beli pontok rendezett halmazai.
    Legyen x0 PL-t és PR szeparáló x koordináta
    dL := TAVKERES(PL,XL,YL)
    dR := TAVKERES(PR,XR,YR)
    d := min(dL,dR)
    Yd := azon P-beli pontok y koordináta szerint rendezett
halmaza, amelyek x-koordinátájára |x0-x| ≤ d (max. 7 pont)
    Ha van Yd-ben d-nél közelebbi pár, akkor azt adjuk vissza,
    egyébként d-t a megfelelo pontpárral
}
```

Szakaszok metszése

A p_1p_2 szakasz átfog egy egyenest,
ha p_1 az egyenes egyik, p_2 az egyenes másik oldalára esik.



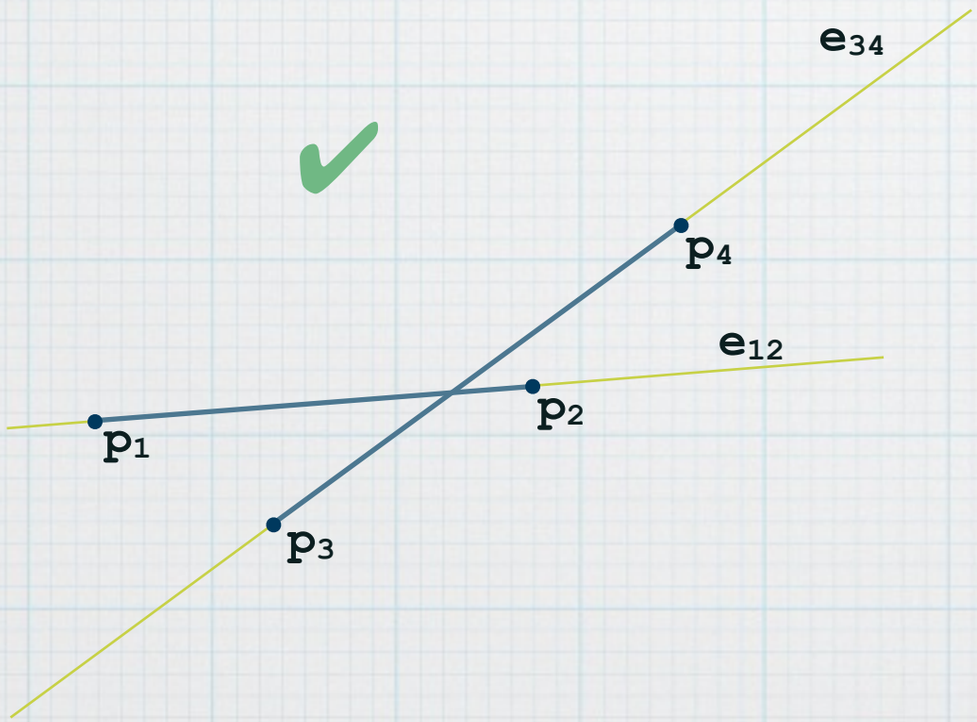
A p_1p_2 szakasz egyenese e, ha $p_1 \in e \ \wedge \ p_2 \in e$

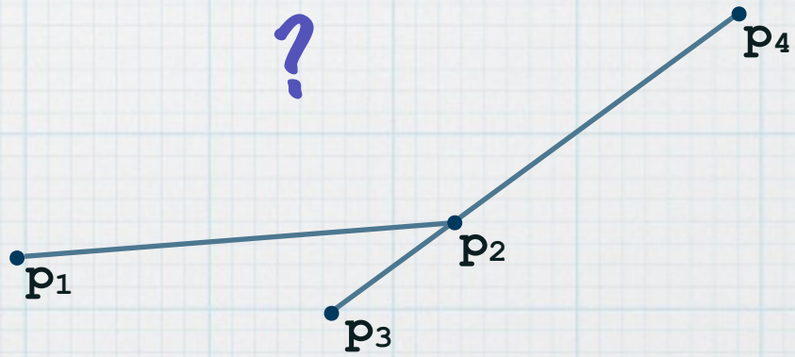


Szakaszok metszése

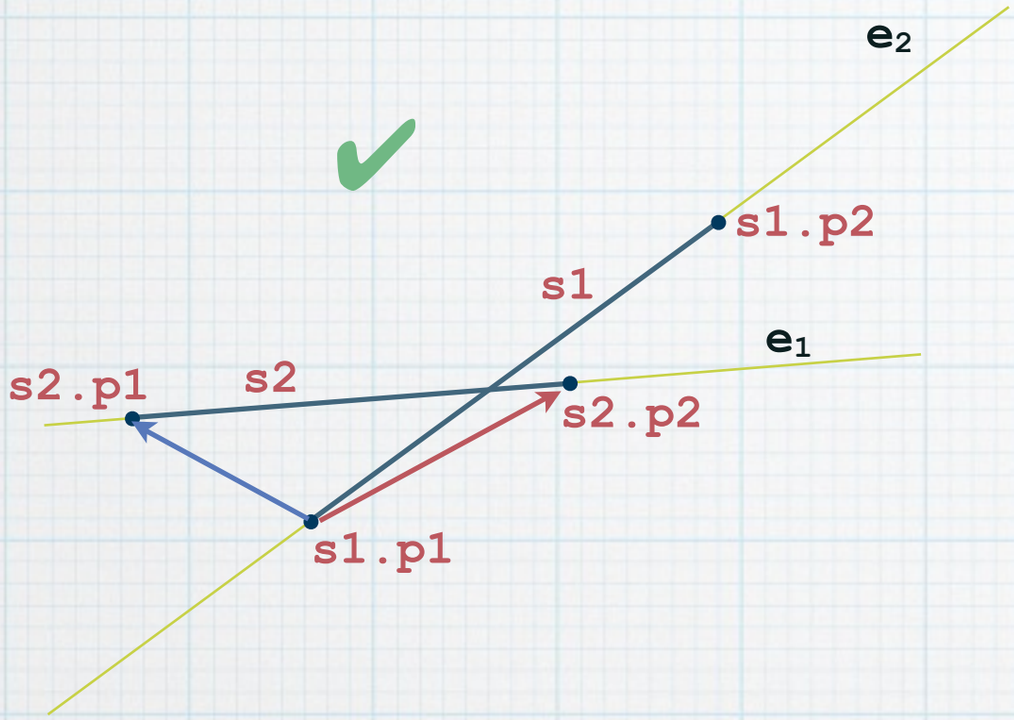
Két szakasz pontosan akkor metszi egymást, ha az alábbi feltételek egyike teljesül:

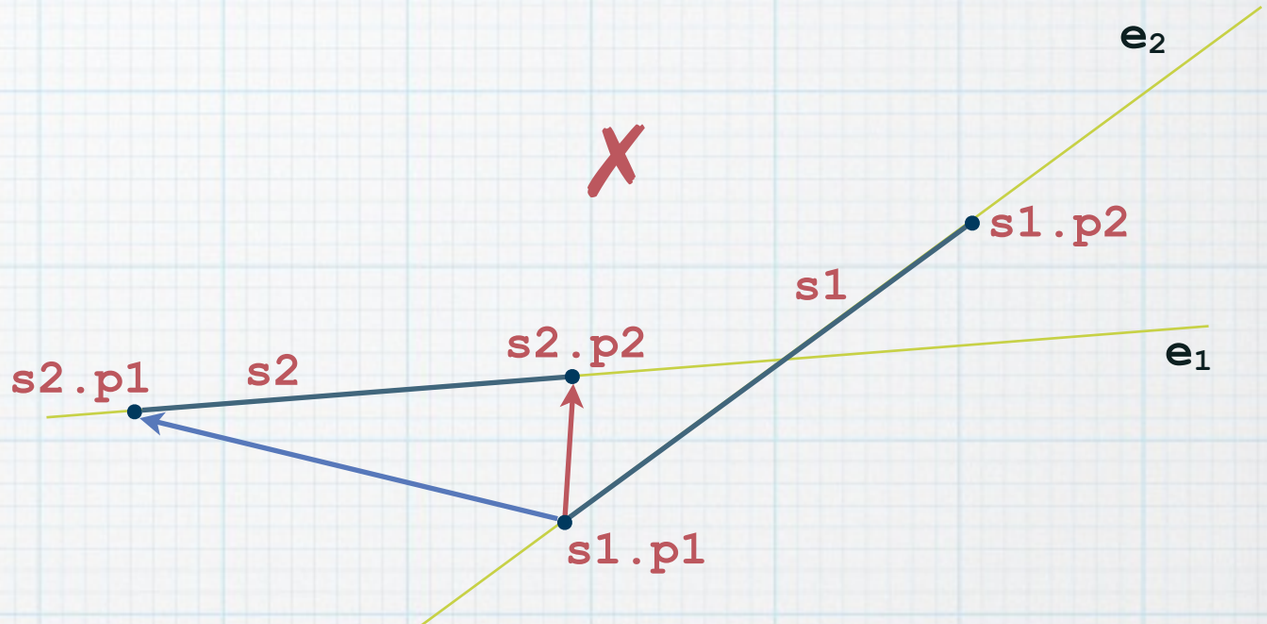
- ✔ mindkét szakasz átfogja a másik egyenesét
- ? az egyik szakasz valamelyik végpontja illeszkedik a másik szakaszra





Szakaszok metszése





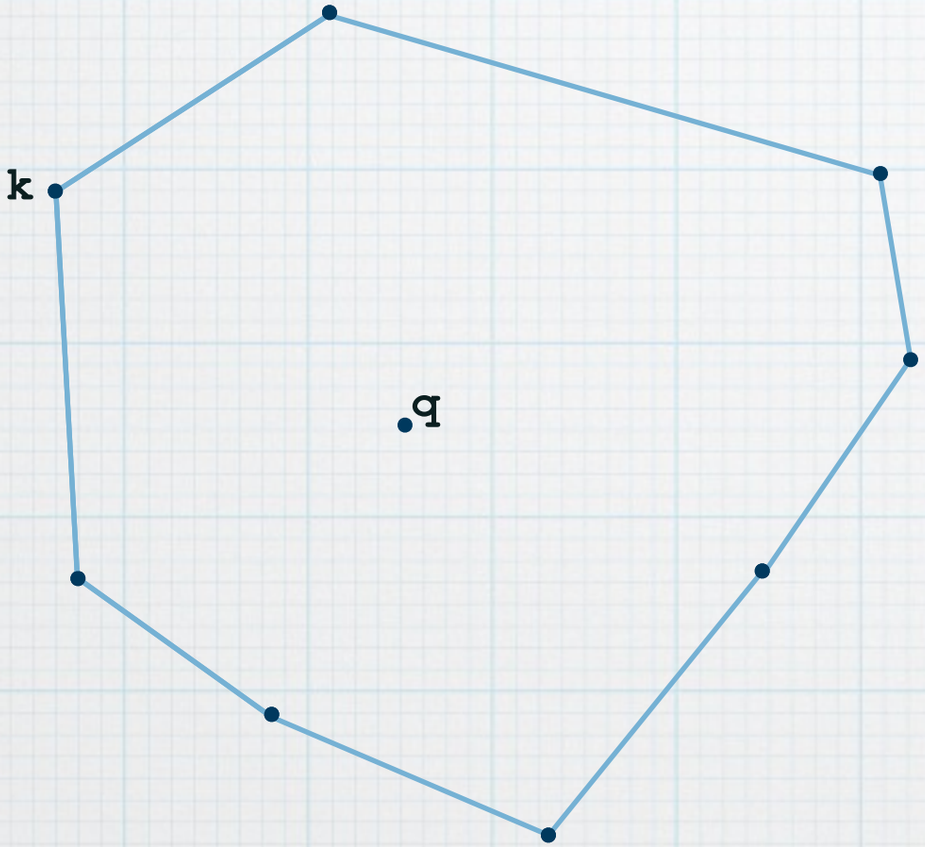

```
szmetsz(szakasz s1,s2) {
    int e=forgasirany(s1.p1,s1.p2,s2.p1) ;
    int f=forgasirany(s1.p1,s1.p2,s2.p2) ;
    int g=forgasirany(s2.p1,s2.p2,s1.p1) ;
    int h=forgasirany(s2.p1,s2.p2,s1.p2) ;
    if (e*f<0 and g*h<0) return true ;

    ... // és ha rajta van is return true!?

    return false ;
}
```

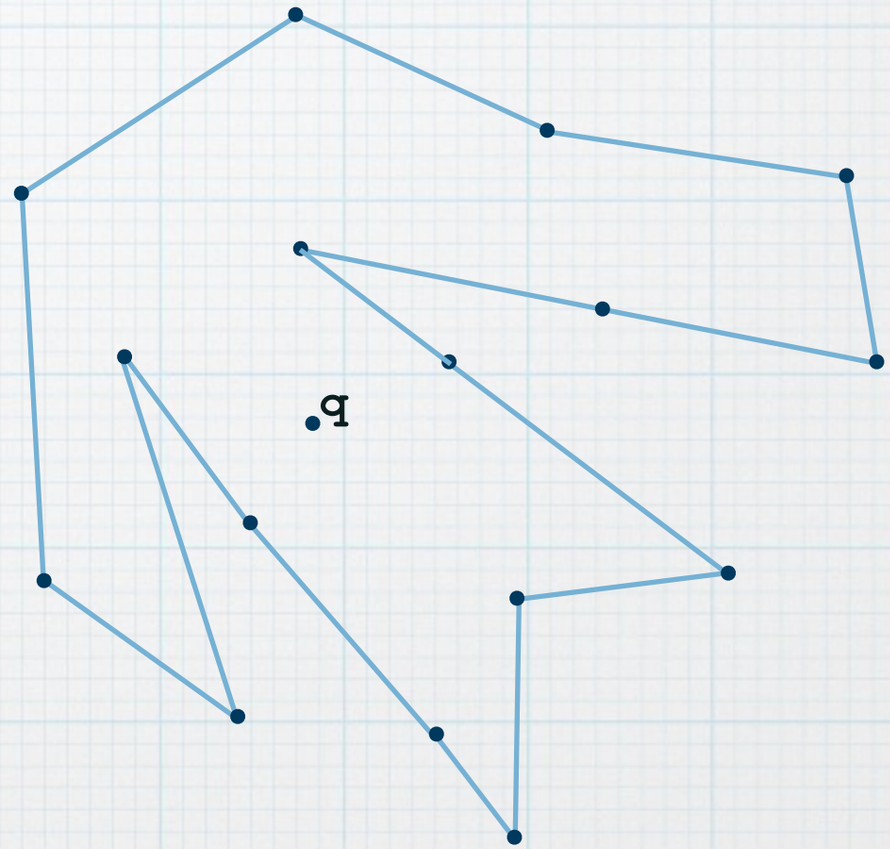
Feladat

Pont helyzetének meghatározása

Adott a síkon egy zárt, nem-metsző poligon a csúcsainak $\{p_1, \ldots, p_n\}$ órajárással ellentétes felsorolásával és adott egy q pont.
Eldöntendo a q pontnak a poligonhoz viszonyított helyzete, azaz, hogy a belsejéhez tartozik-e, az oldalán van-e, avagy külső pont?

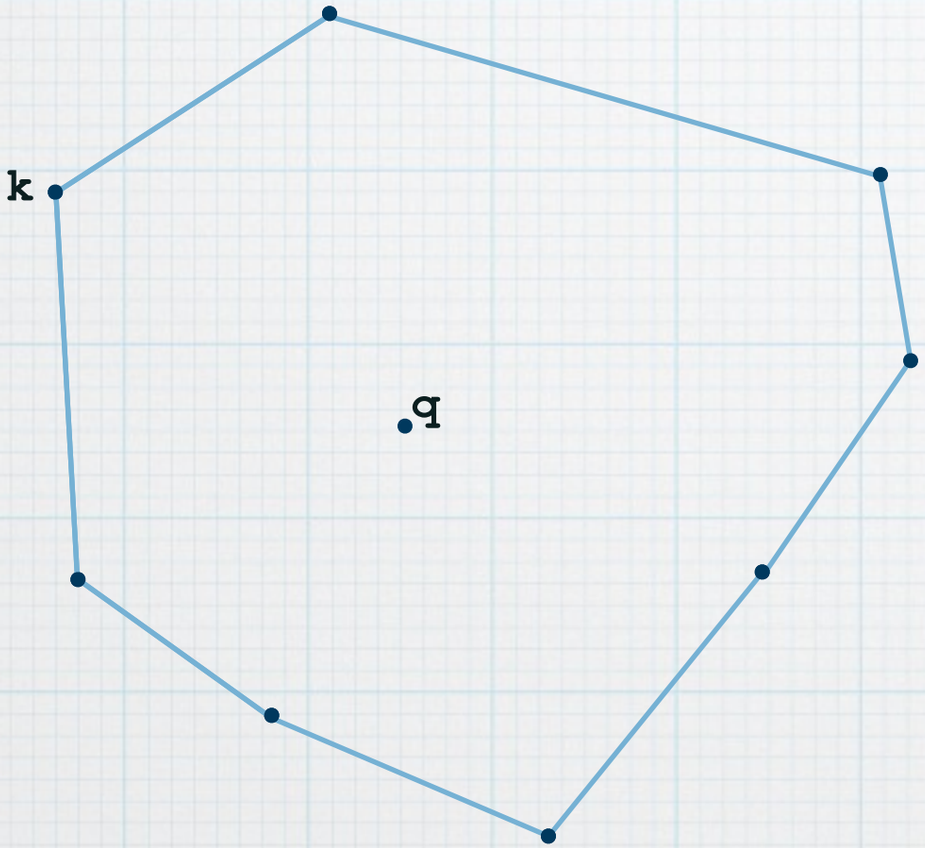
Megoldás



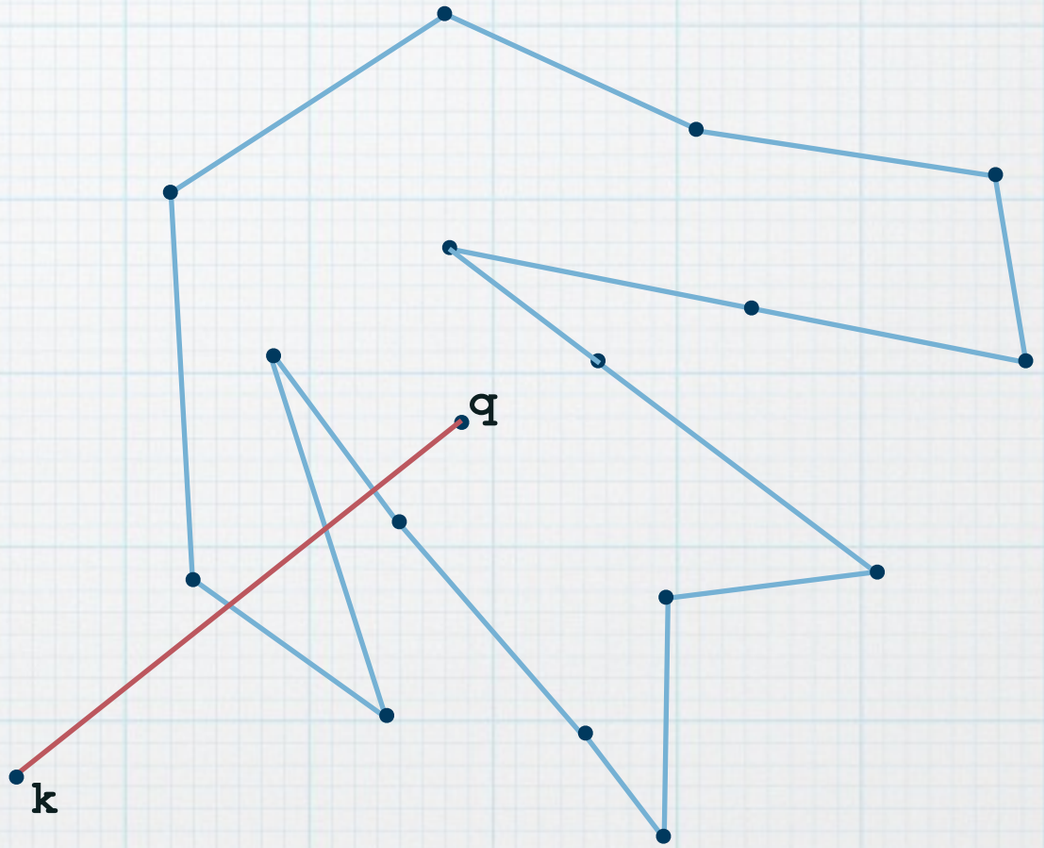


O(n)

Megoldás



O(n)



O(n)

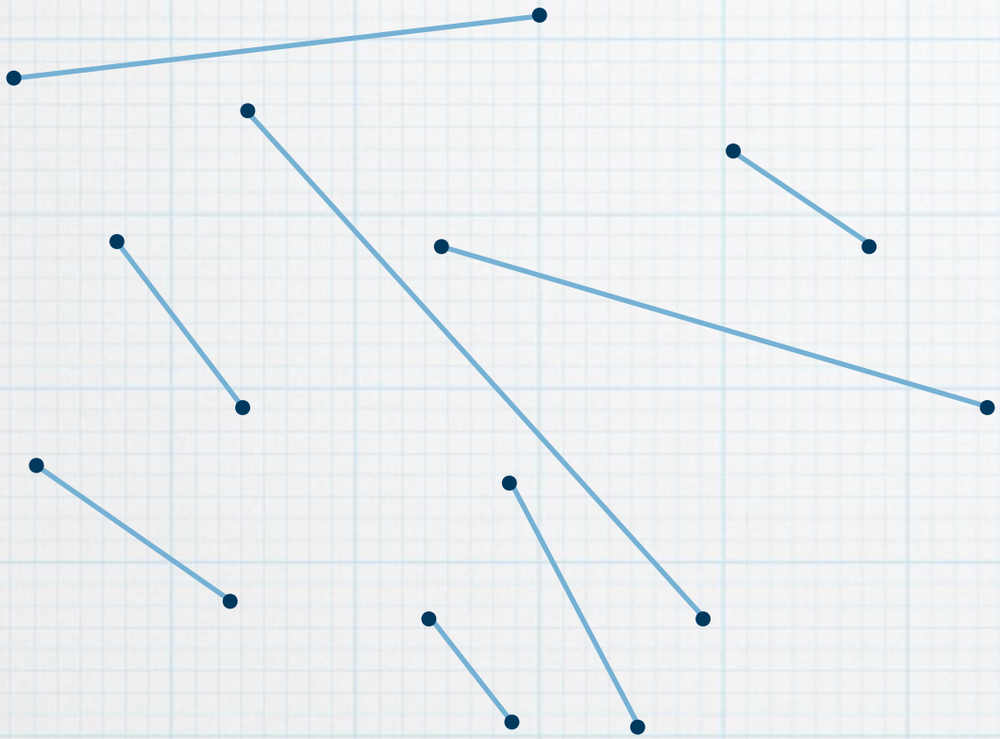
Feladat

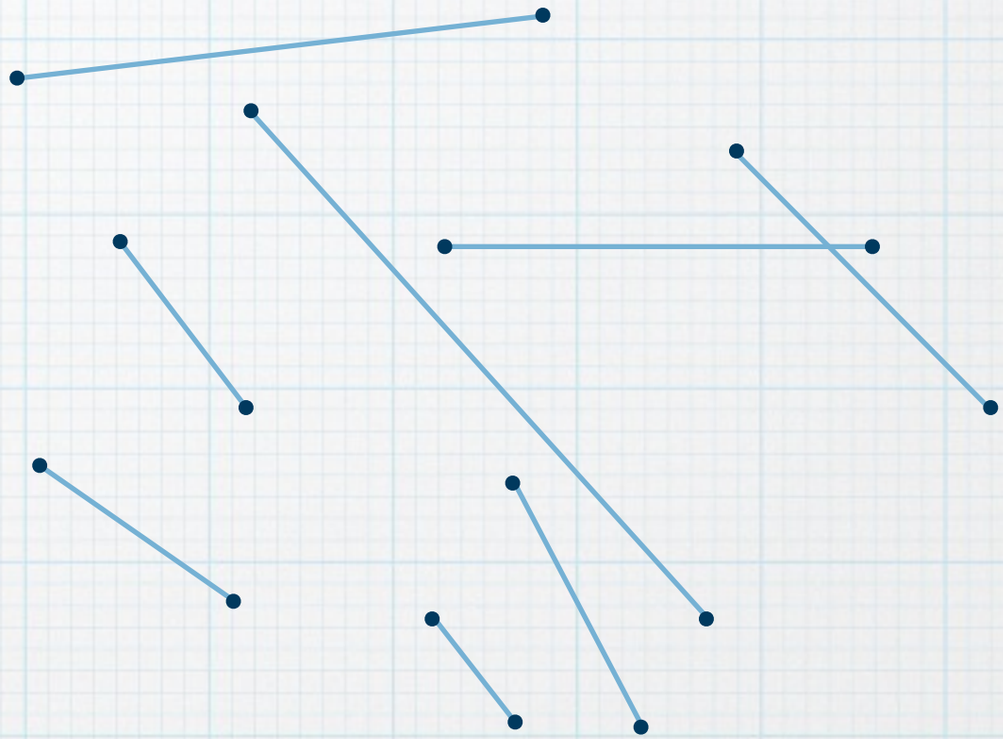
Metsző szakaszpárok keresése

A feladatban adott n darab szakasz $\{s_1, \ldots, s_n\}$ (a szakaszok az s_i.bal és s_i.jobb végpontjaik által adottak) és meg kell határozni vane-e közöttük kettő olyan, amely metszi egymást.

Egyszerűsítő feltételek:
1. Egyik bemeneti szakasz sem függőleges.
2. Nincs három olyan szakasz, amelyek egy pontban metszenék egymást

Megoldás

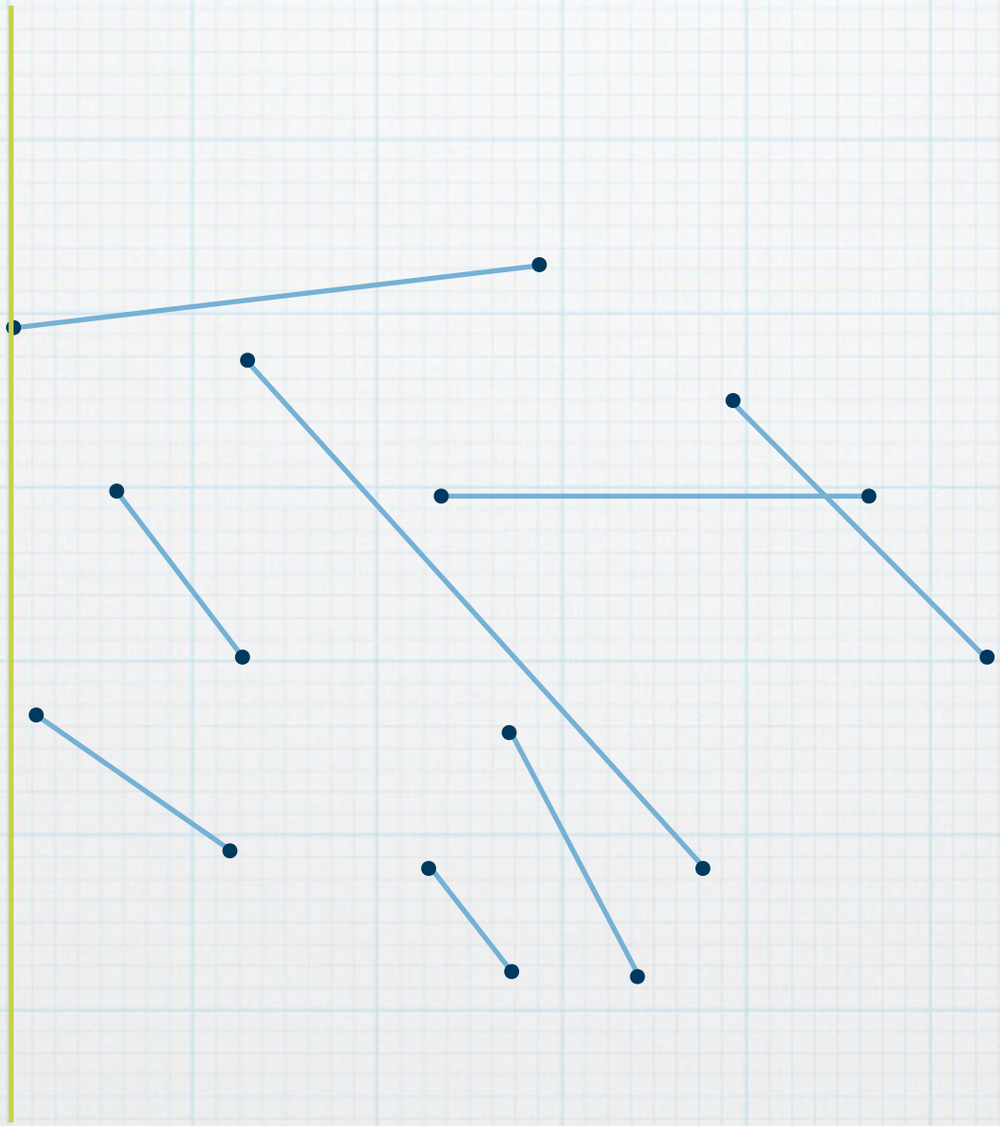




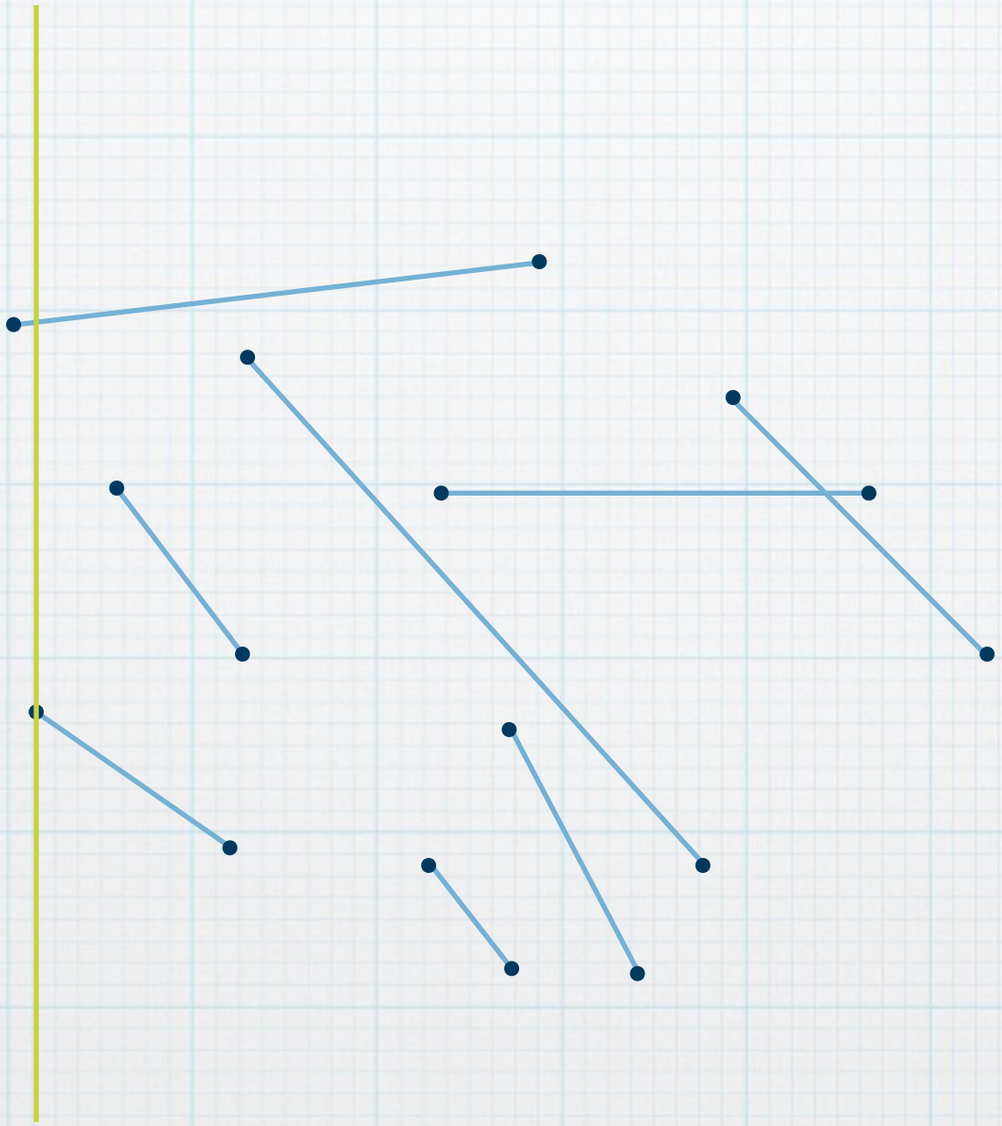
Seprő egyenes

Seprés során egy képzeletbeli seprő egyenest mozgatunk a geometriai elemek halmazán. Azt a dimenziót, amelyben a seprő egyenes halad, idődimenziónak hívjuk.
Jelen esetben a seprő egyenes az y-tengellyel párhuzamos egyenes, az idodimenzió az x-koordináta lesz.

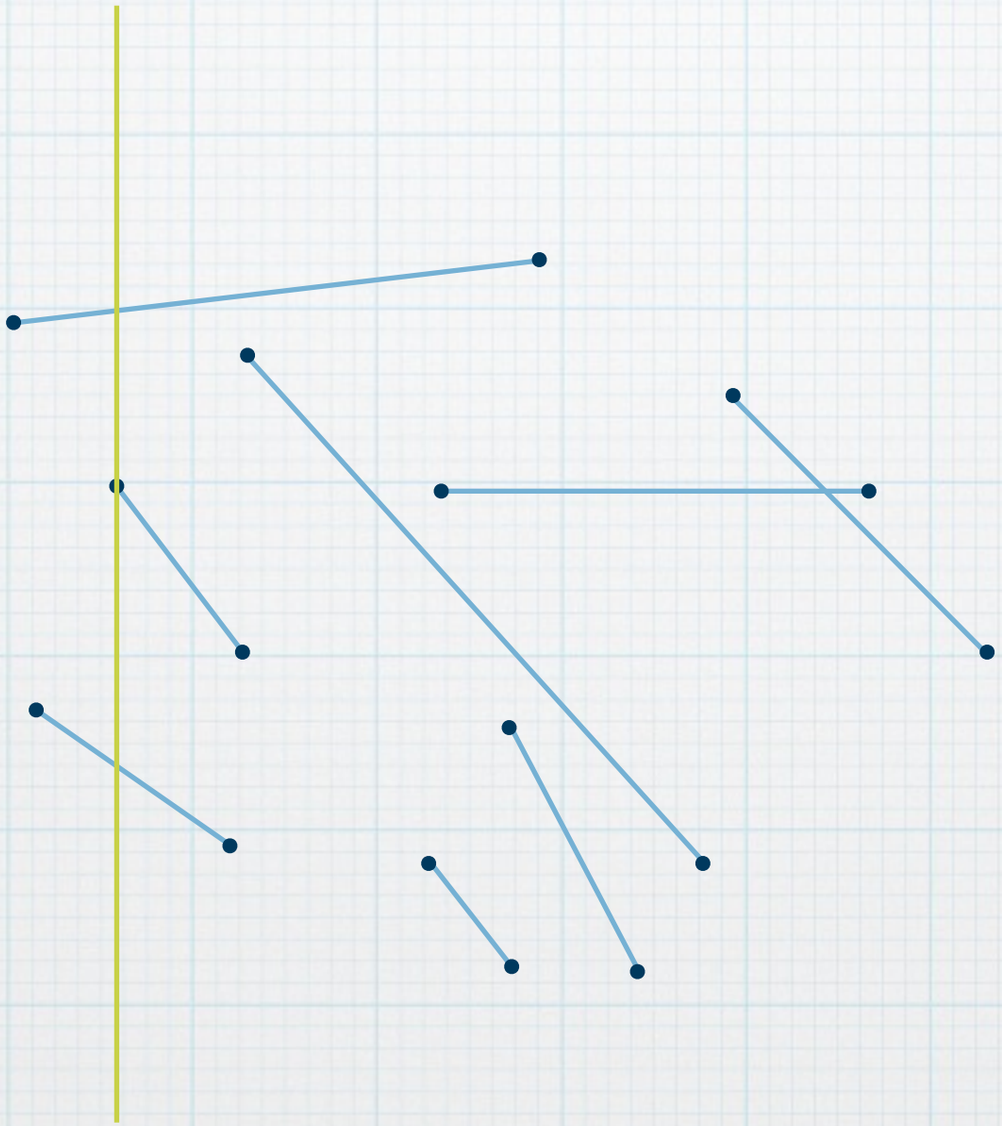
Megoldás



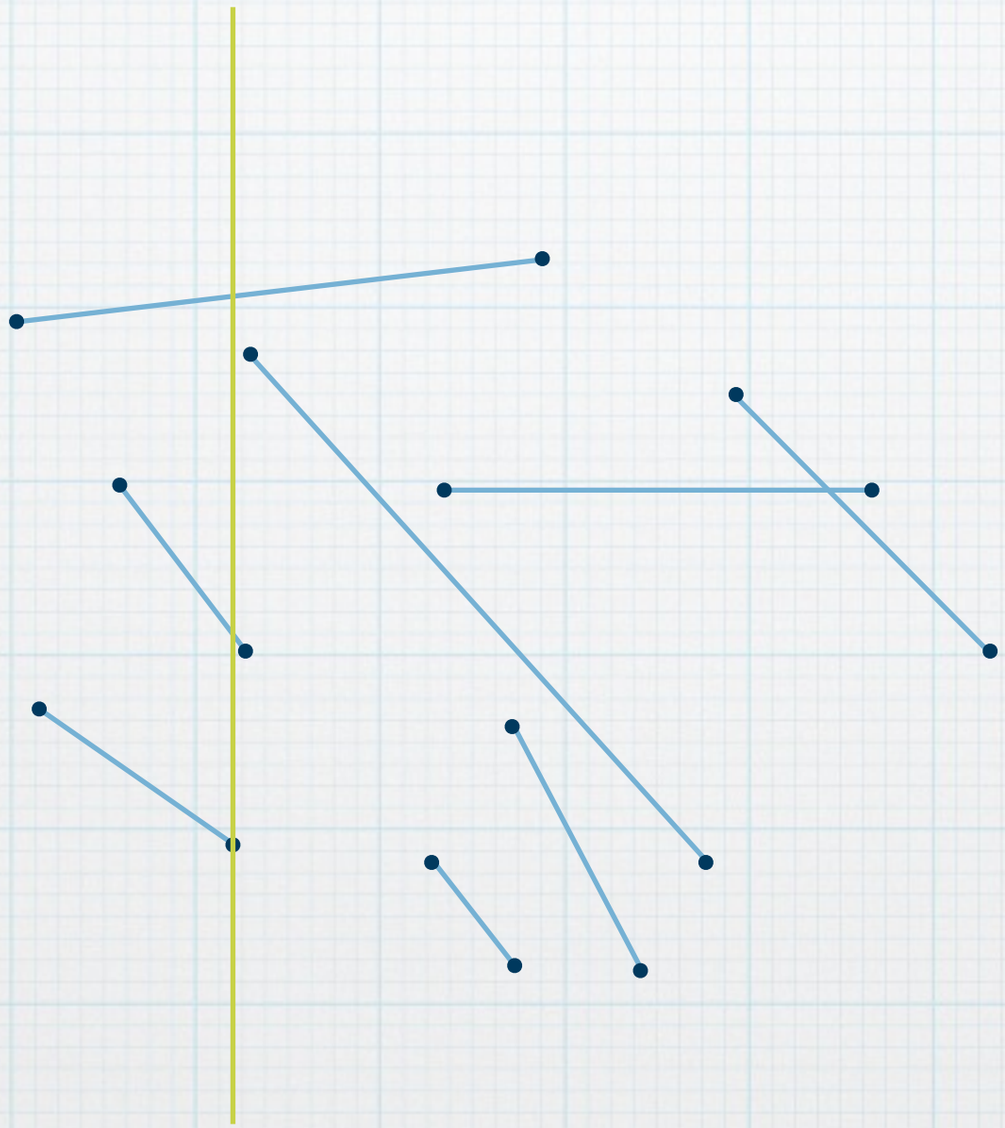
Megoldás



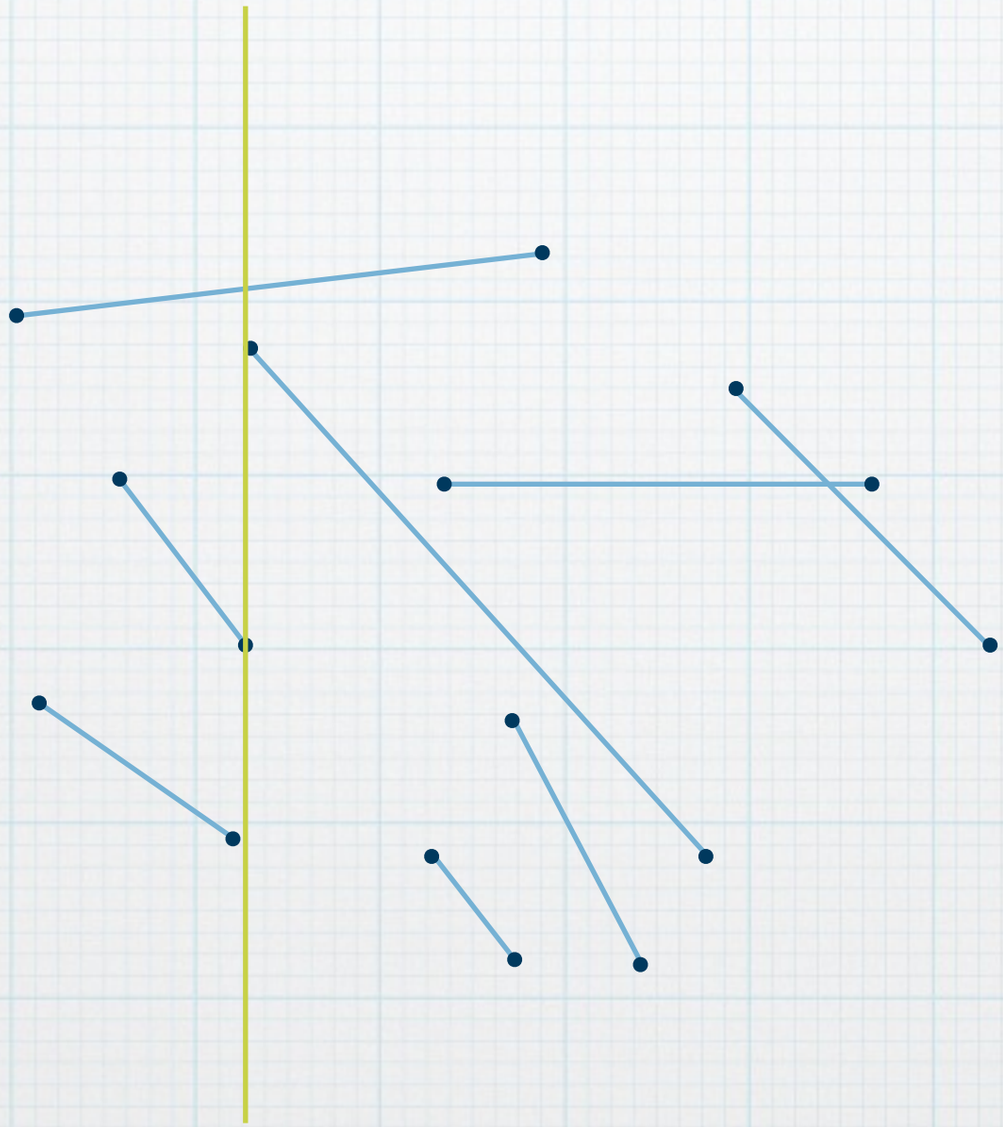
Megoldás



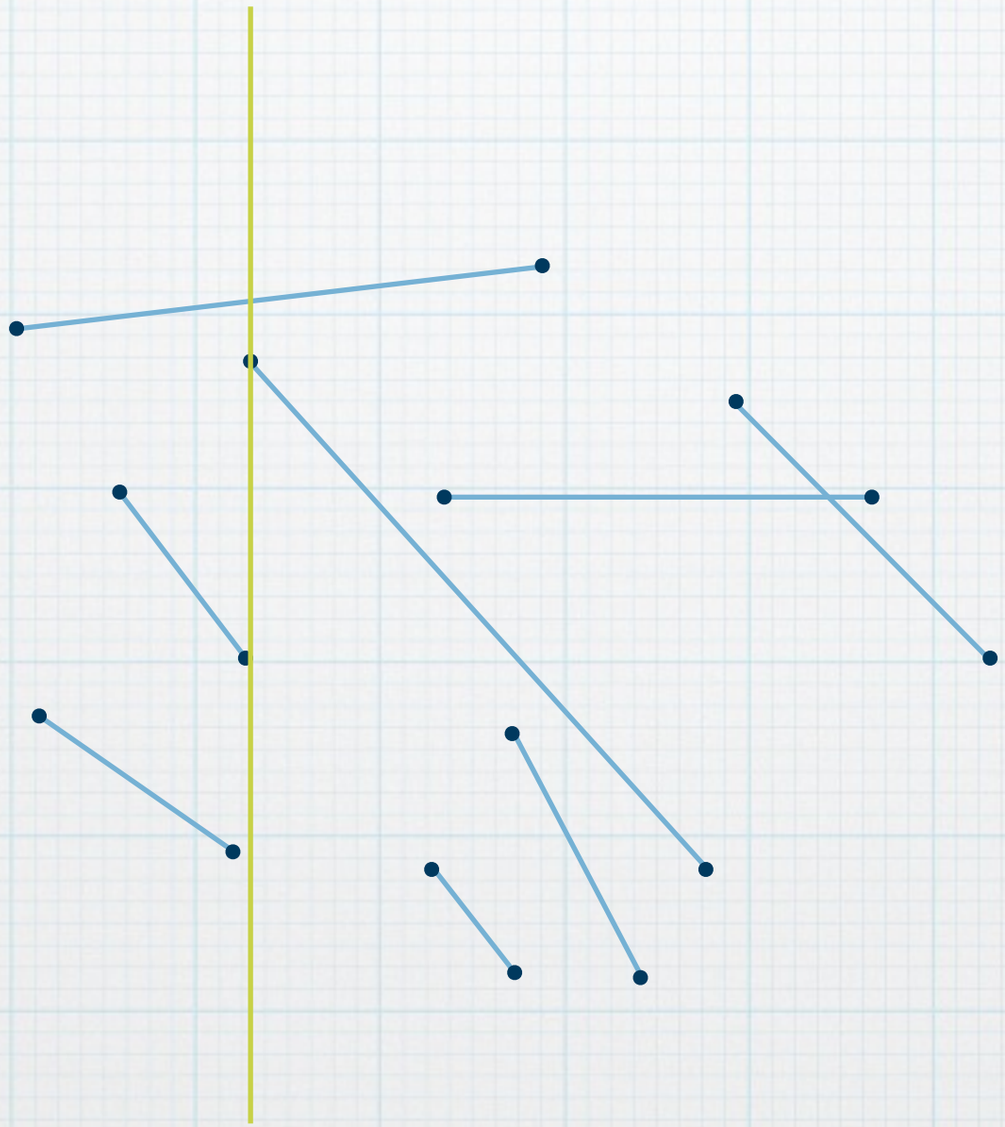
Megoldás



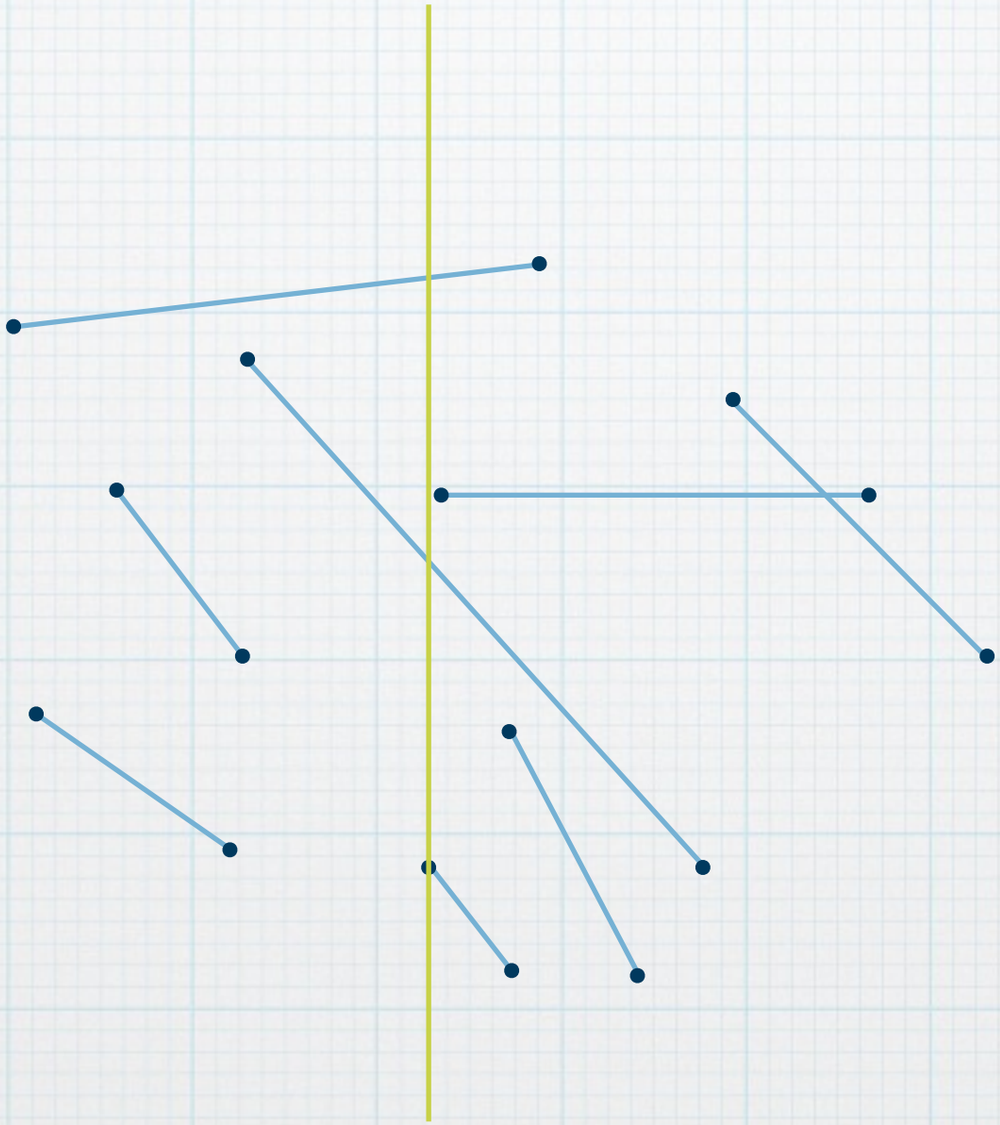
Megoldás



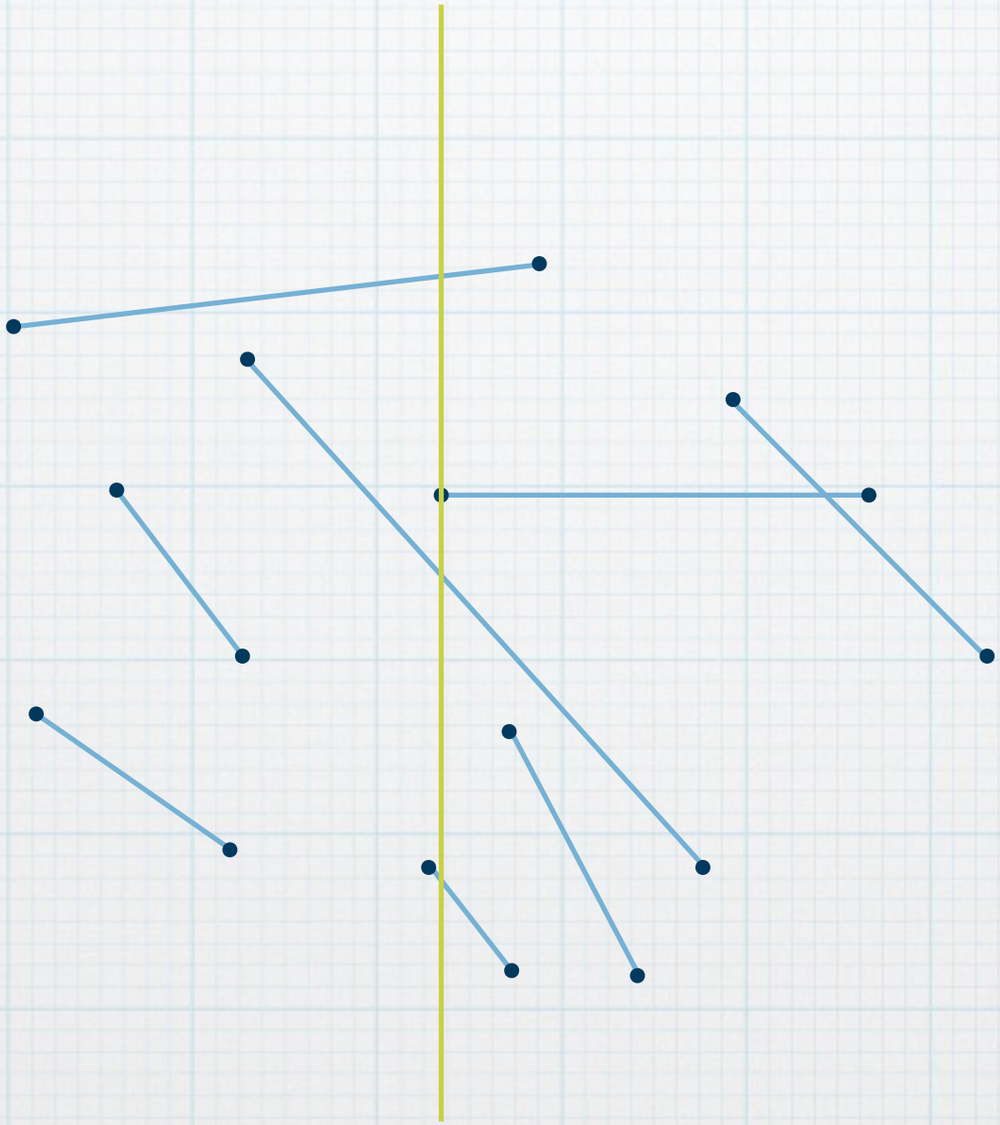
Megoldás



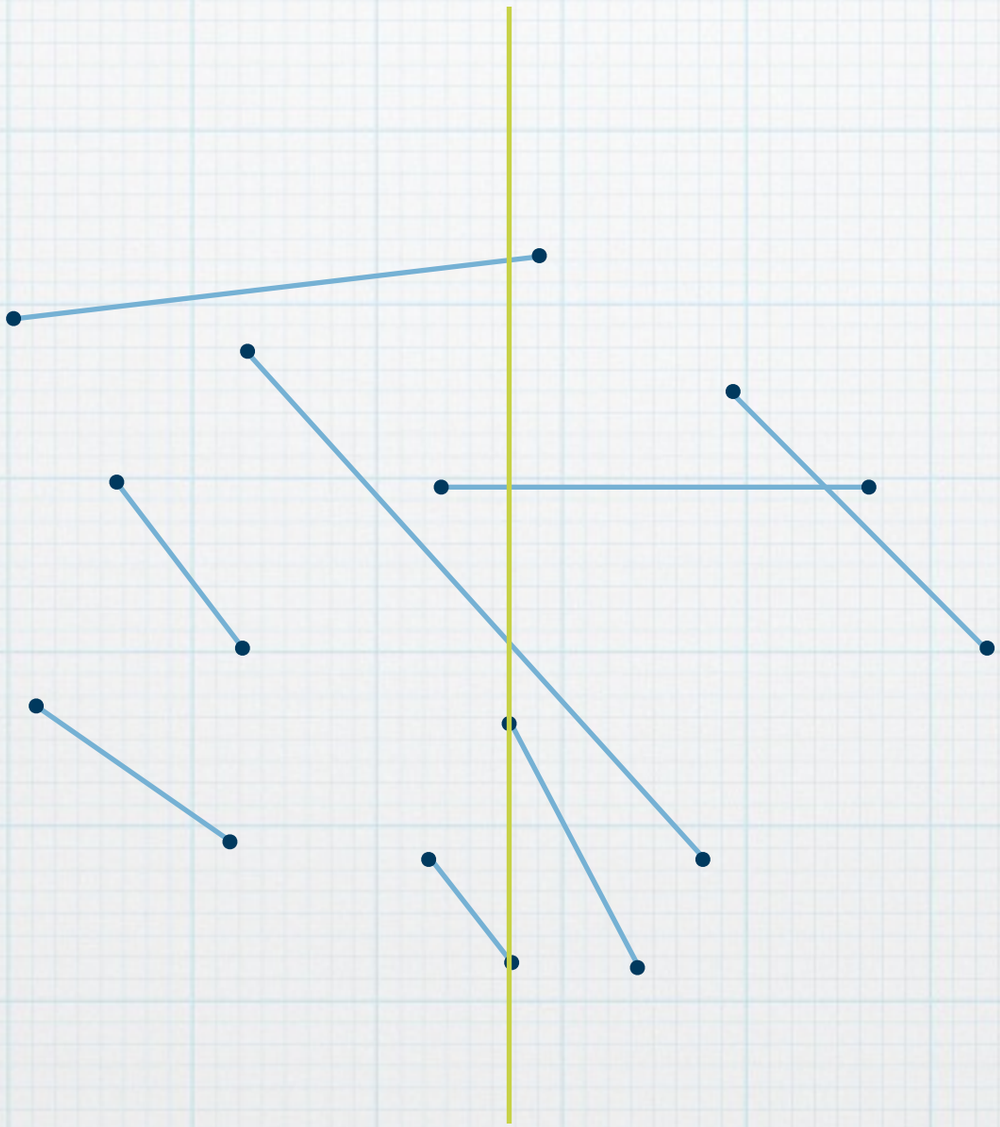
Megoldás



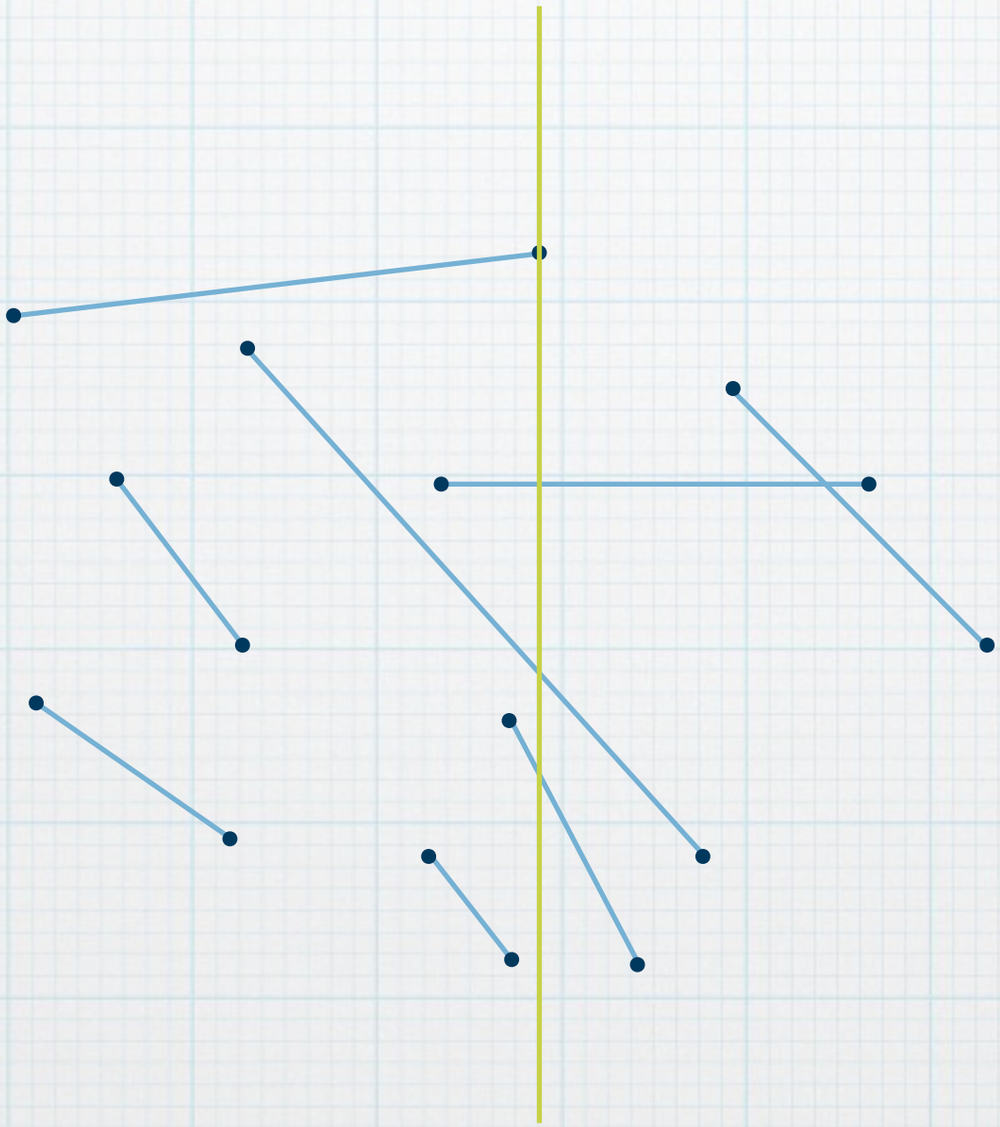
Megoldás



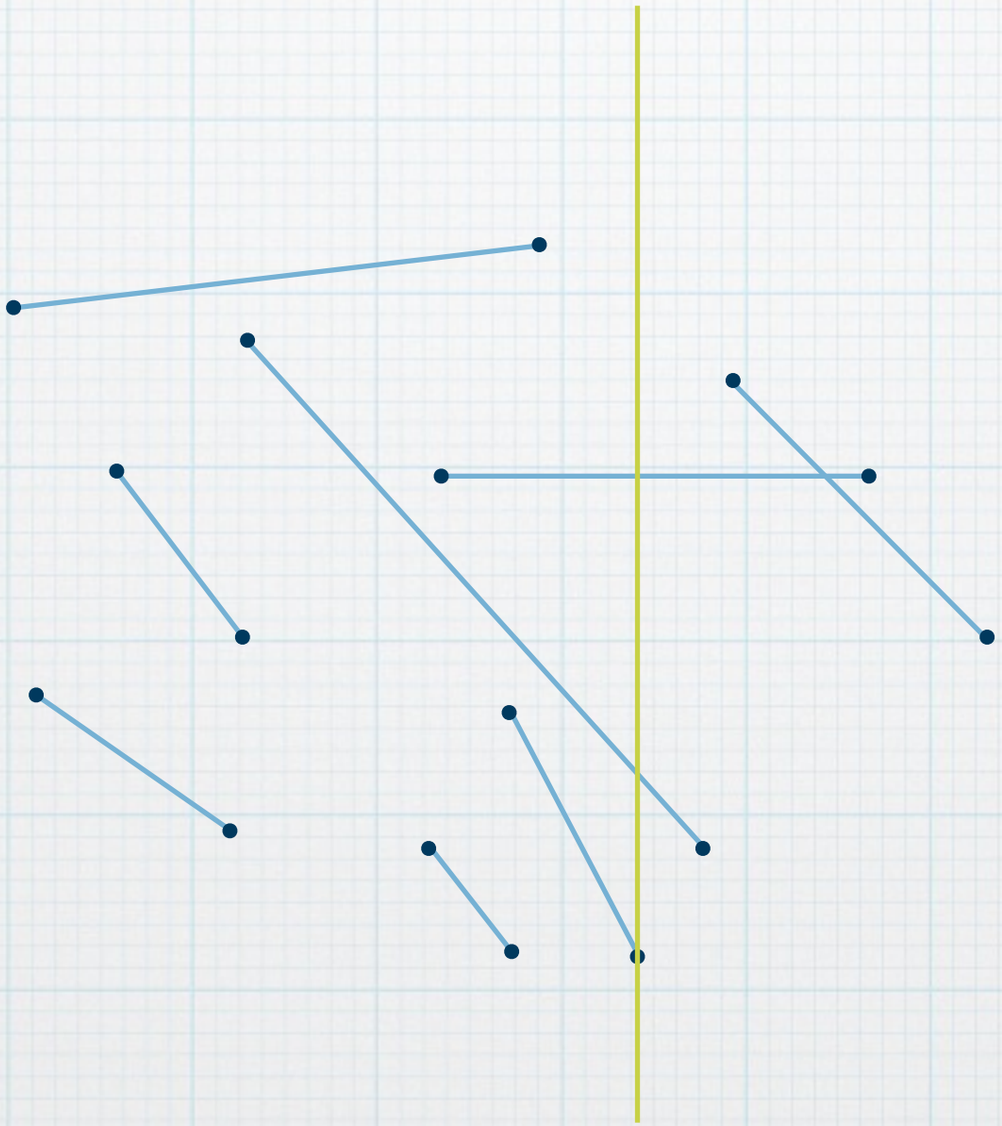
Megoldás



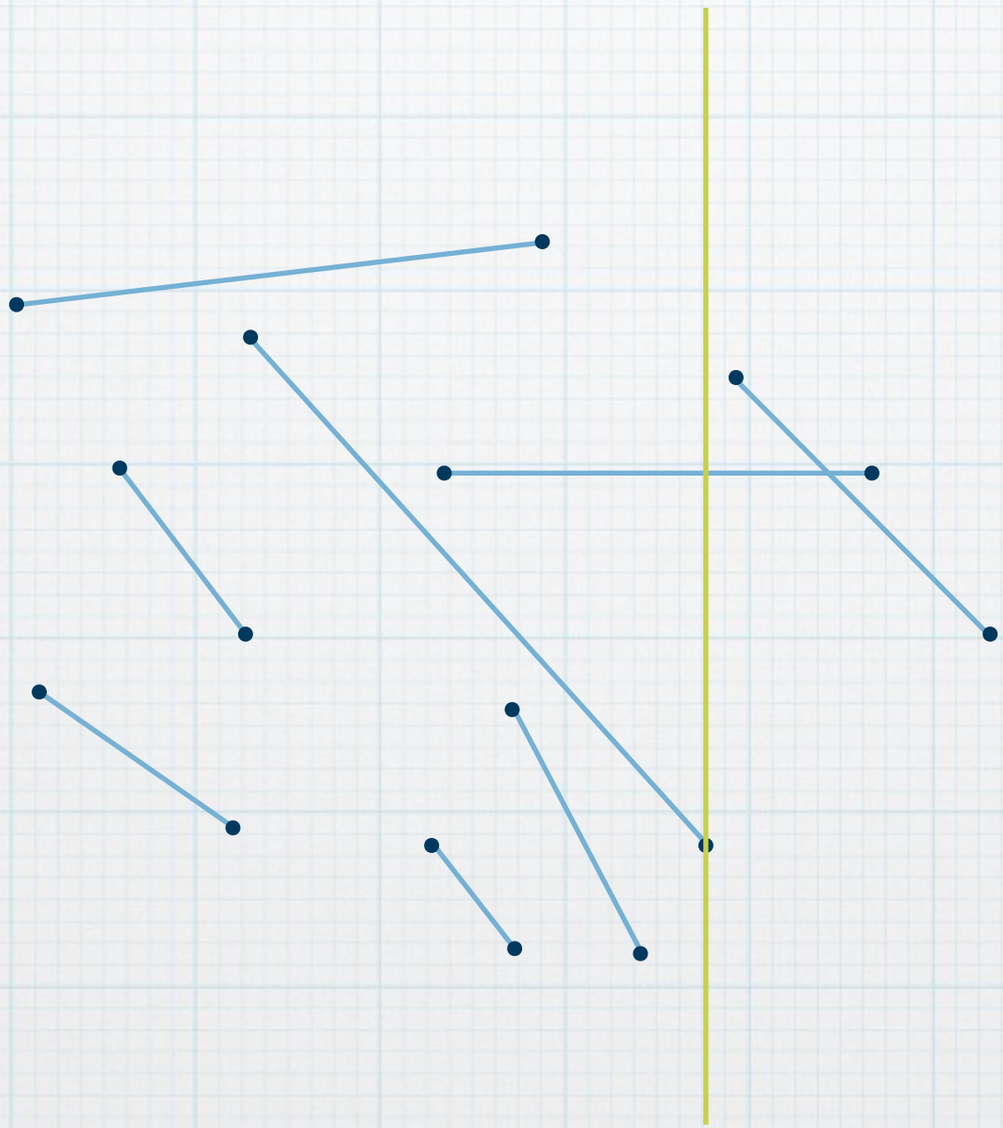
Megoldás



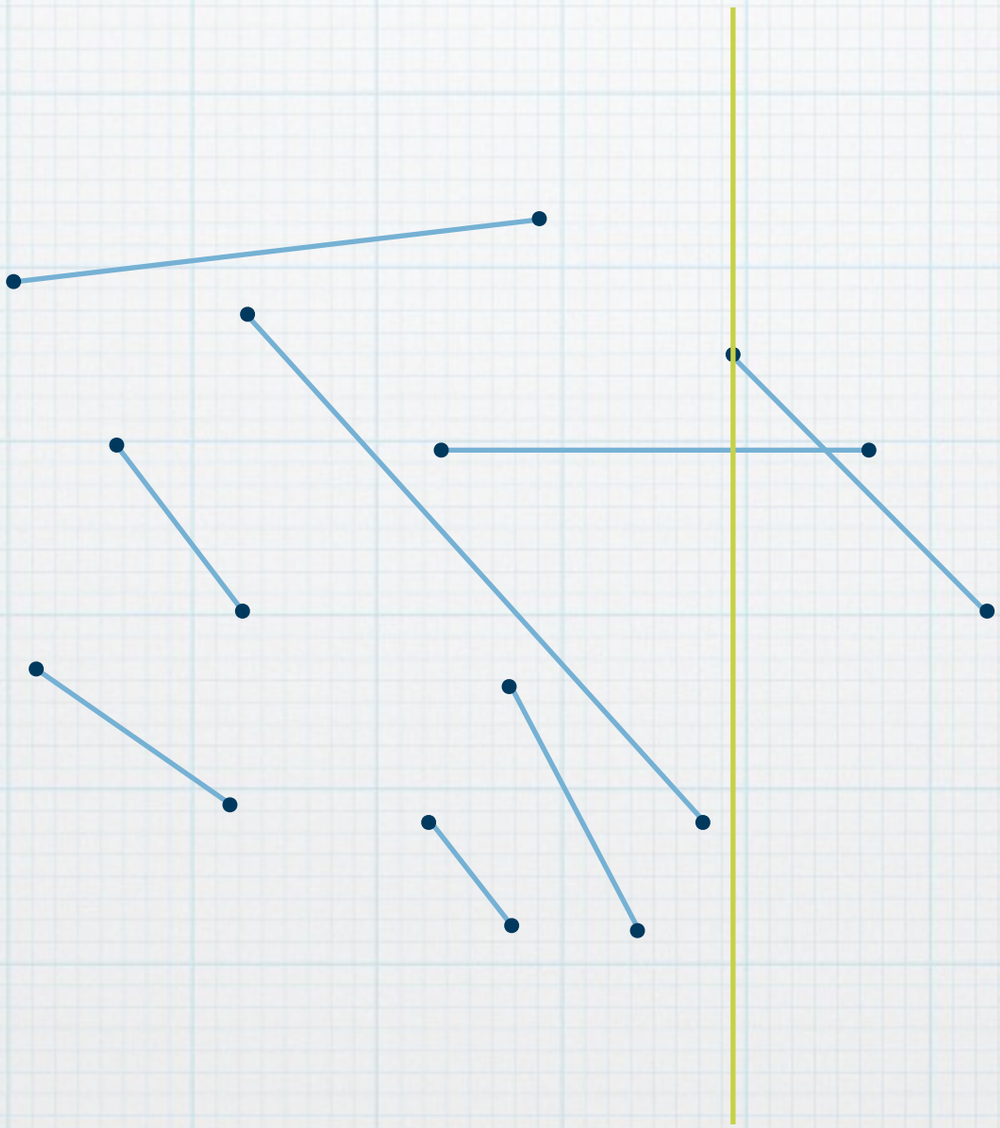
Megoldás



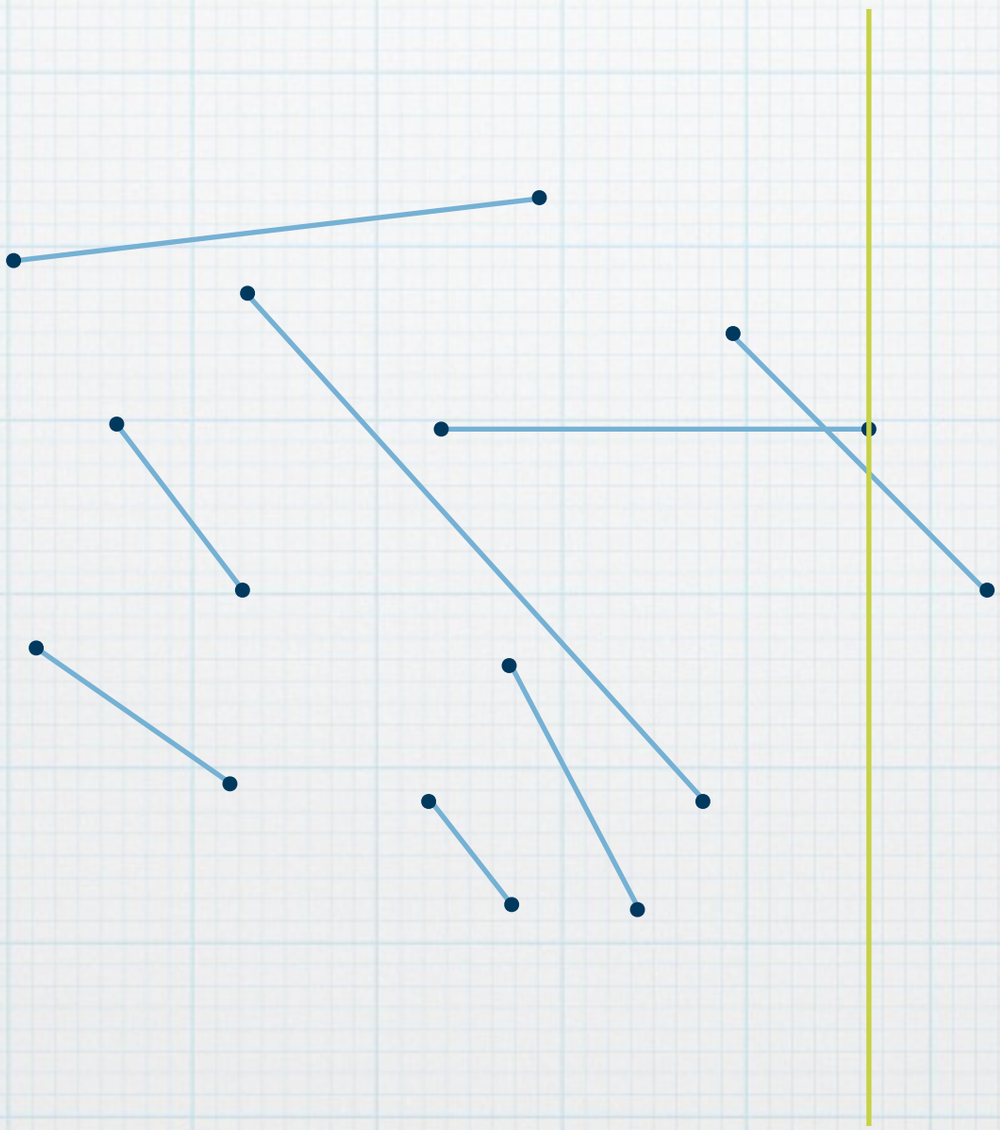
Megoldás



Megoldás



Megoldás



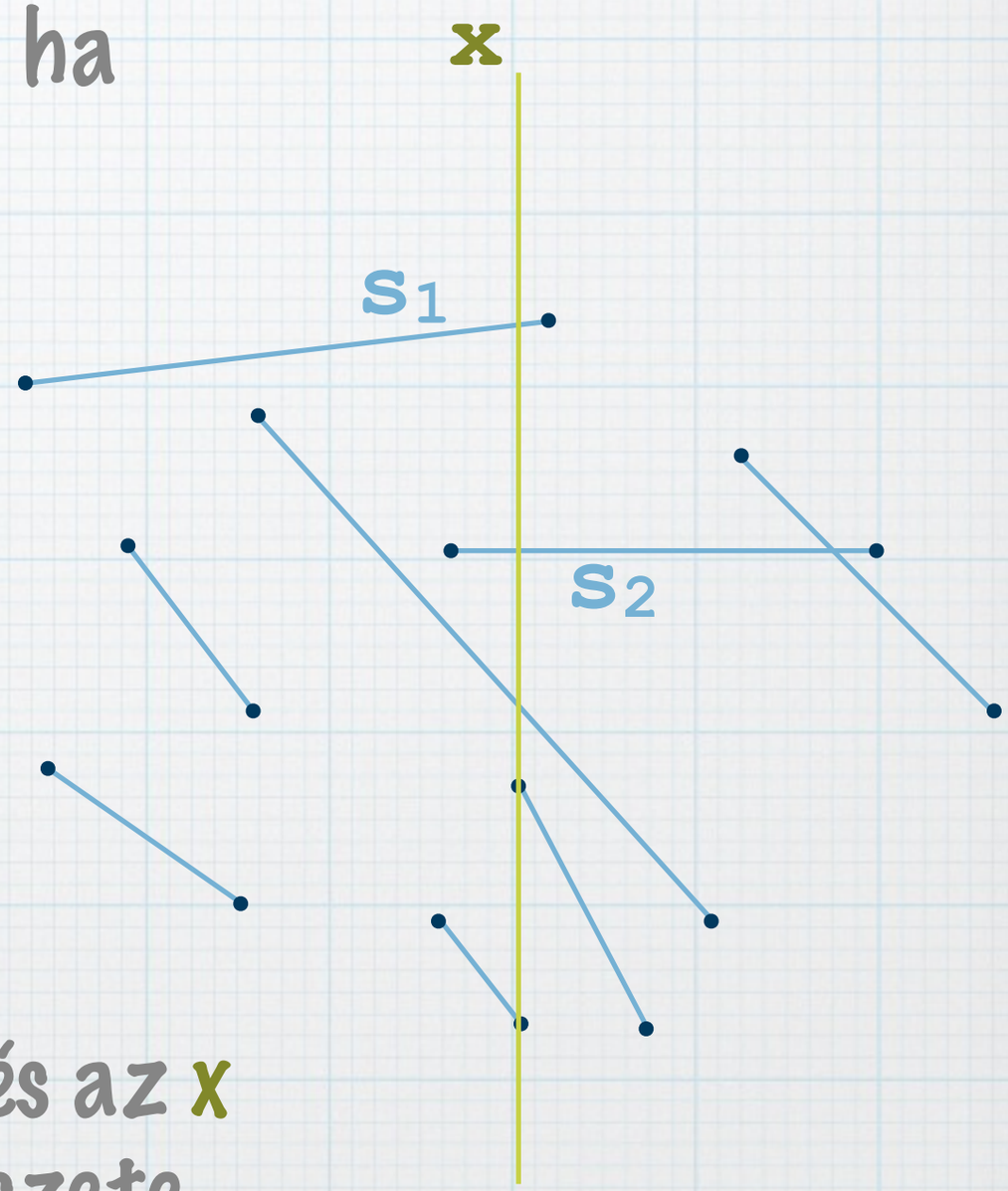
Legyen s_1 és s_2 szakasz és $x \in R$

s_1 összehasonlítható s_2-vel x-nél, ha

$$s_1.bal.x \leq x \leq s_1.jobb.x$$

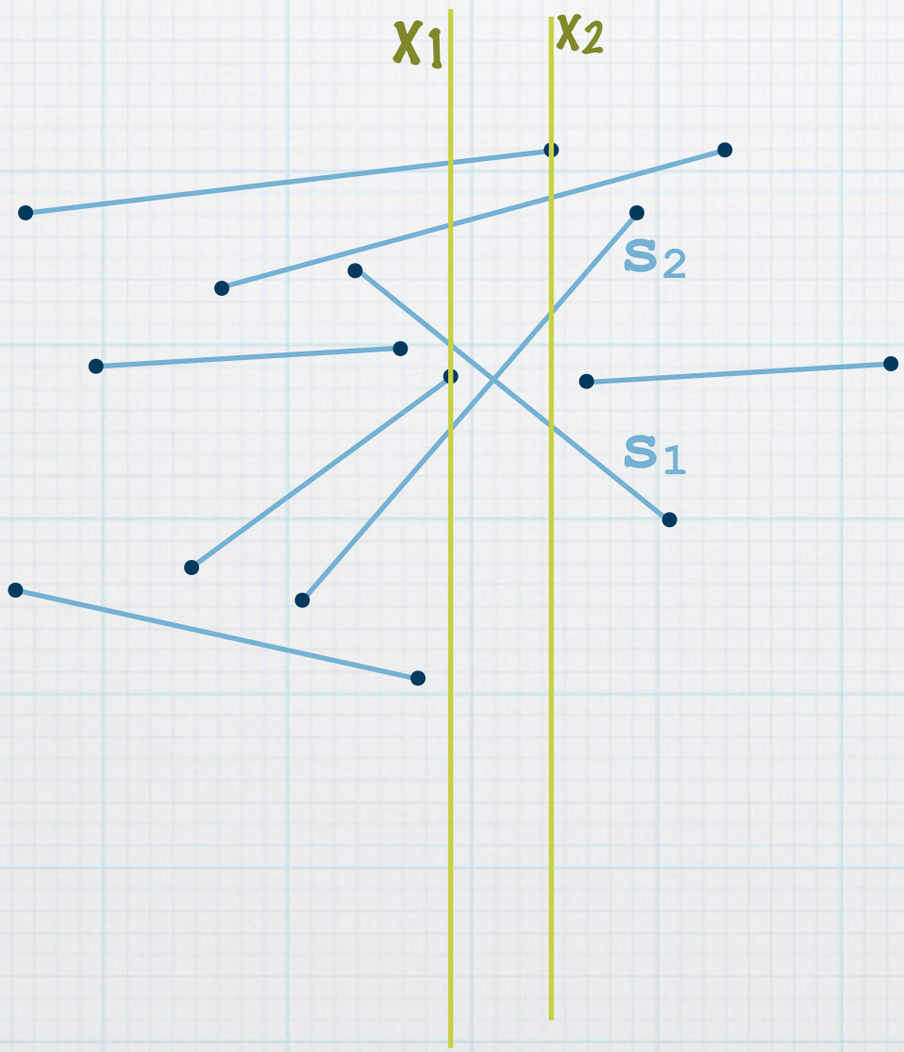
és

$$s_2.bal.x \leq x \leq s_2.jobb.x$$

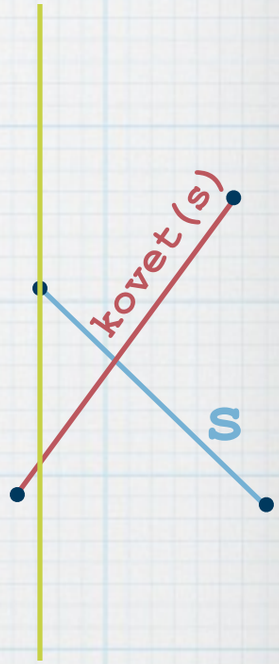


Az s_1 szakasz felette van az s_2 szakasznak, ($s_1 >_x s_2$), ha s_1 összehasonlítható s_2-vel x-nél és az x seperő egyenesnek s_1-el vett metszete magasabban van, mint s_2-vel vett metszete.

Ha $x_1 < x_2$-re $s_1 >_{x_1} s_2$ és $s_2 >_{x_2} s_1$, akkor az s_1 és s_2 szakasz metszi egymást x_1 és x_2 között. Továbbá, ha x_1 és x_2 a seprő egyenes két egymást követő megállási helye, akkor x_1-ben (után) a két szakasz egymást követő lesz a rendezésben.

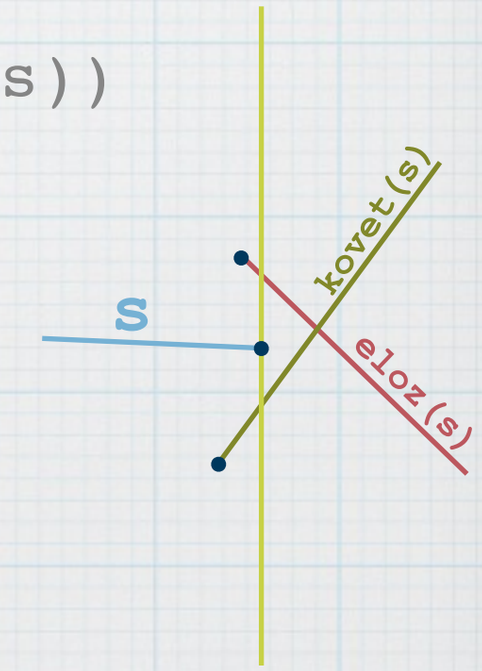


Tehát elegendő ellenőrizni minden szakasz berakásakor, hogy a rendezésben őt megelőző, és követő szakasz metszi-e a berakottat, és kivételkor ellenőrizni, hogy a kivett szakaszt megelozó és követő szakaszok metszik-e egymást.



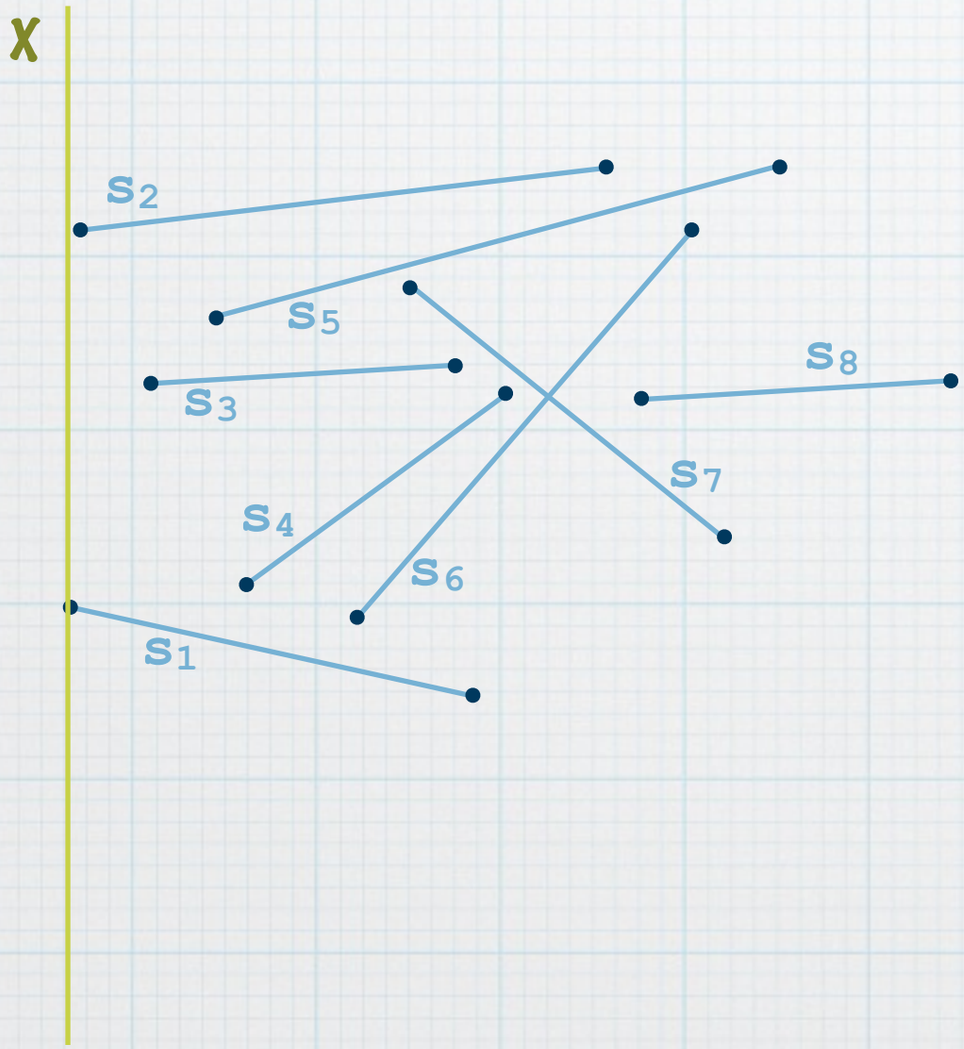
Használjunk Rhalmaz adattípust! (pl. piros-fekete fa)

```
MSZPkeres() {
    minden p-re az R beli sorrendben
    if (p==s.bal) {
       betesz(s) ;
       if (szmetsz(kovet(s),s) or szmetsz(eloz(s),s))
          return True ;
    }
    if (p=s.jobb) {
       if szmetsz(kovet(s),eloz(s)) return True ;
       kivesz(s) ;
    }
}
```



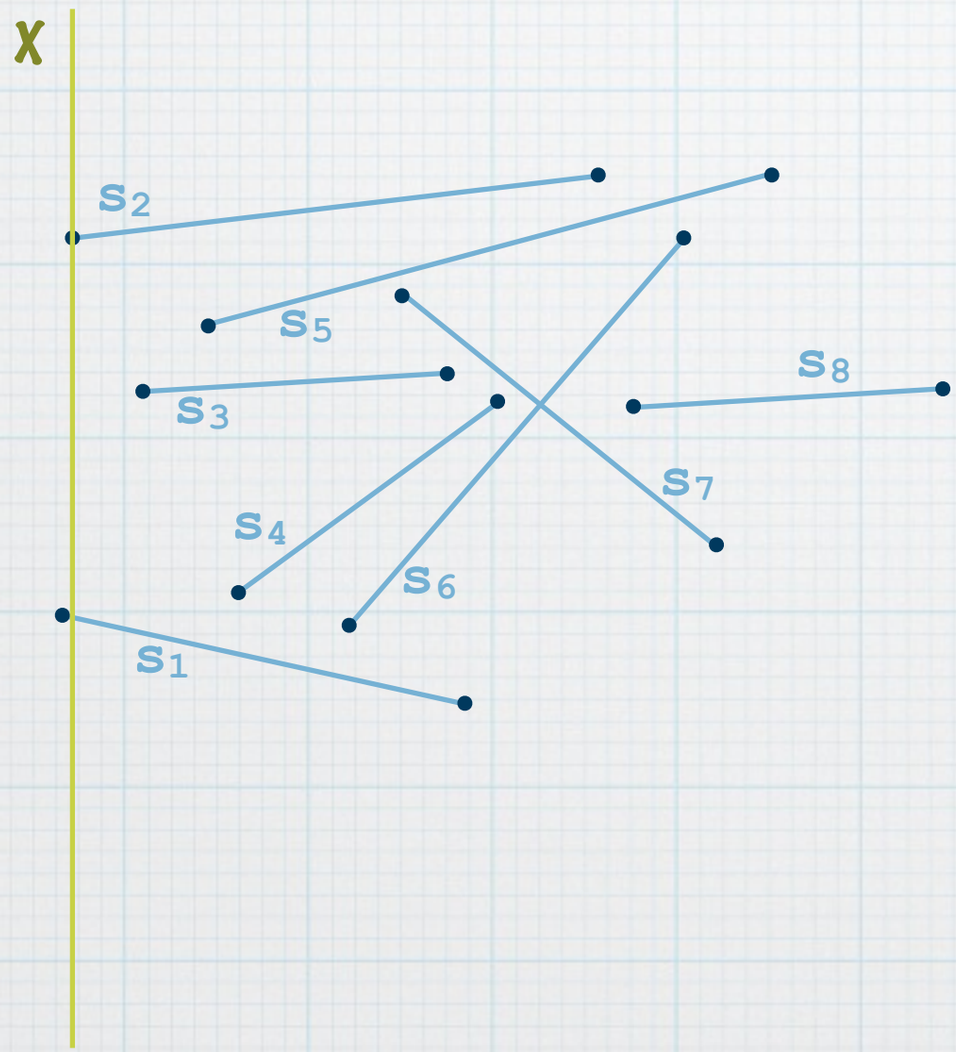
O(nlogn)

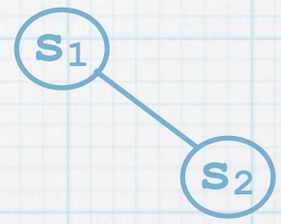
Példa



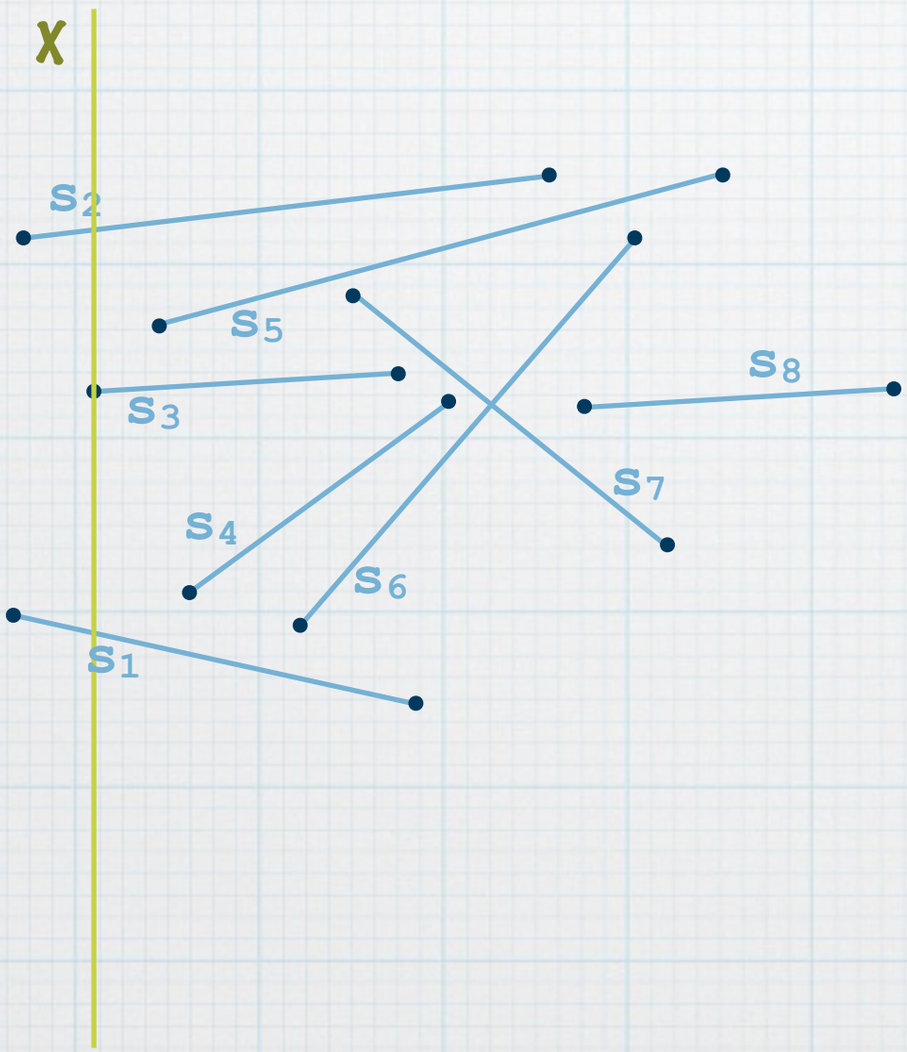
s1

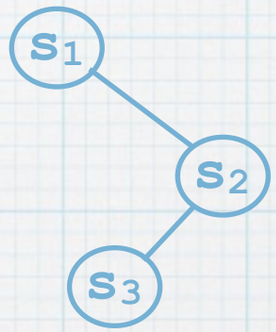
Példa



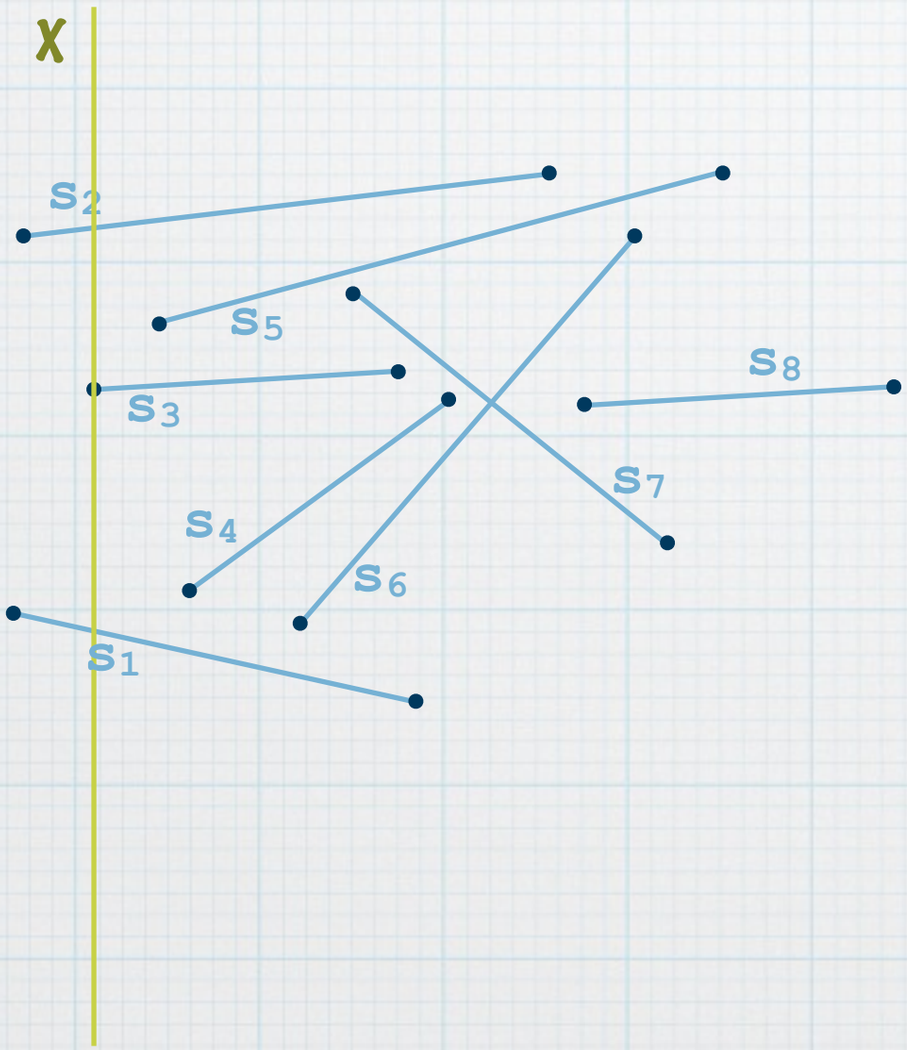


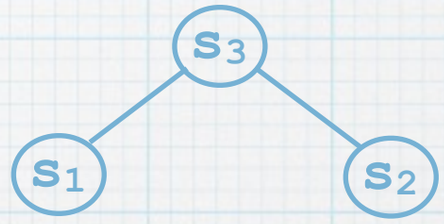
Példa



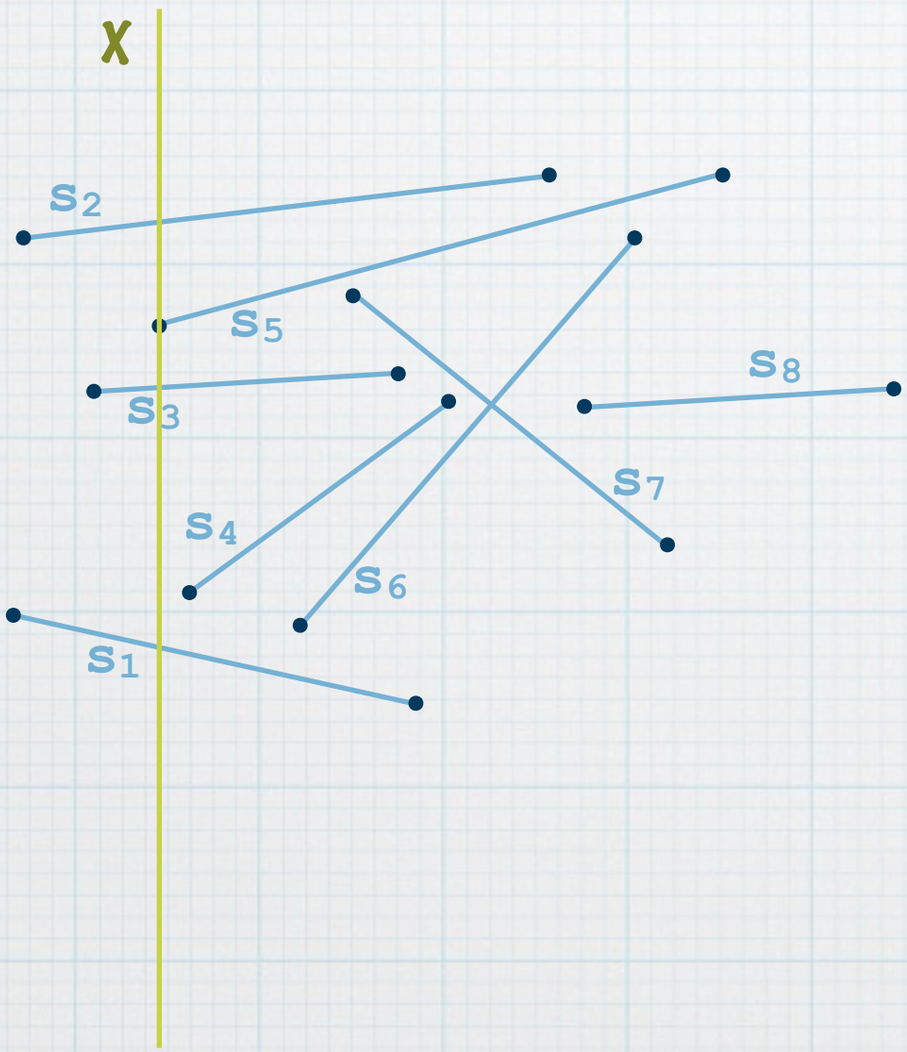


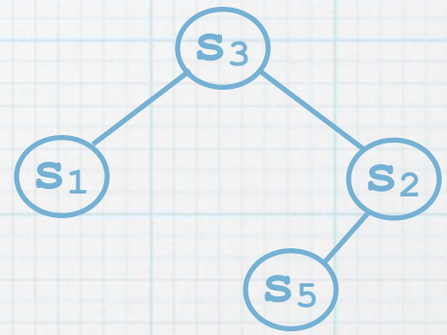
Példa



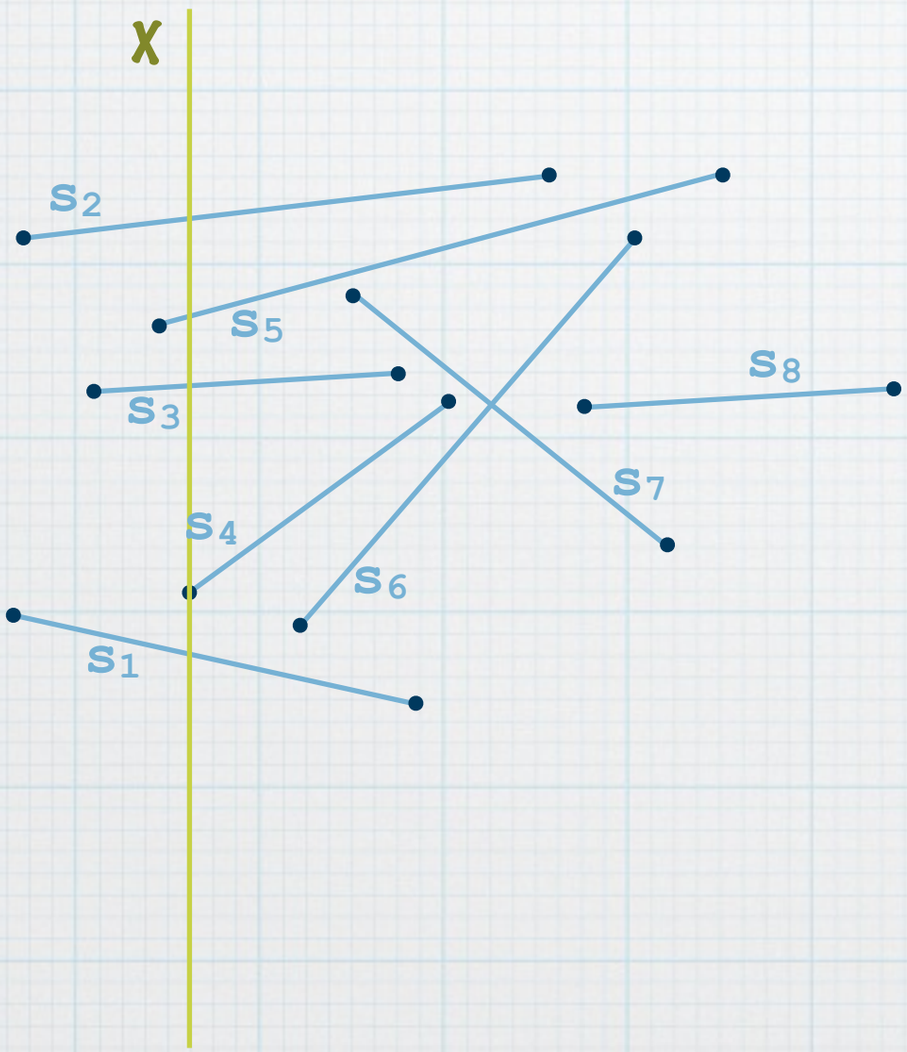


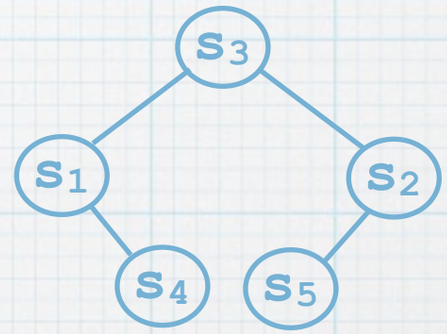
Példa



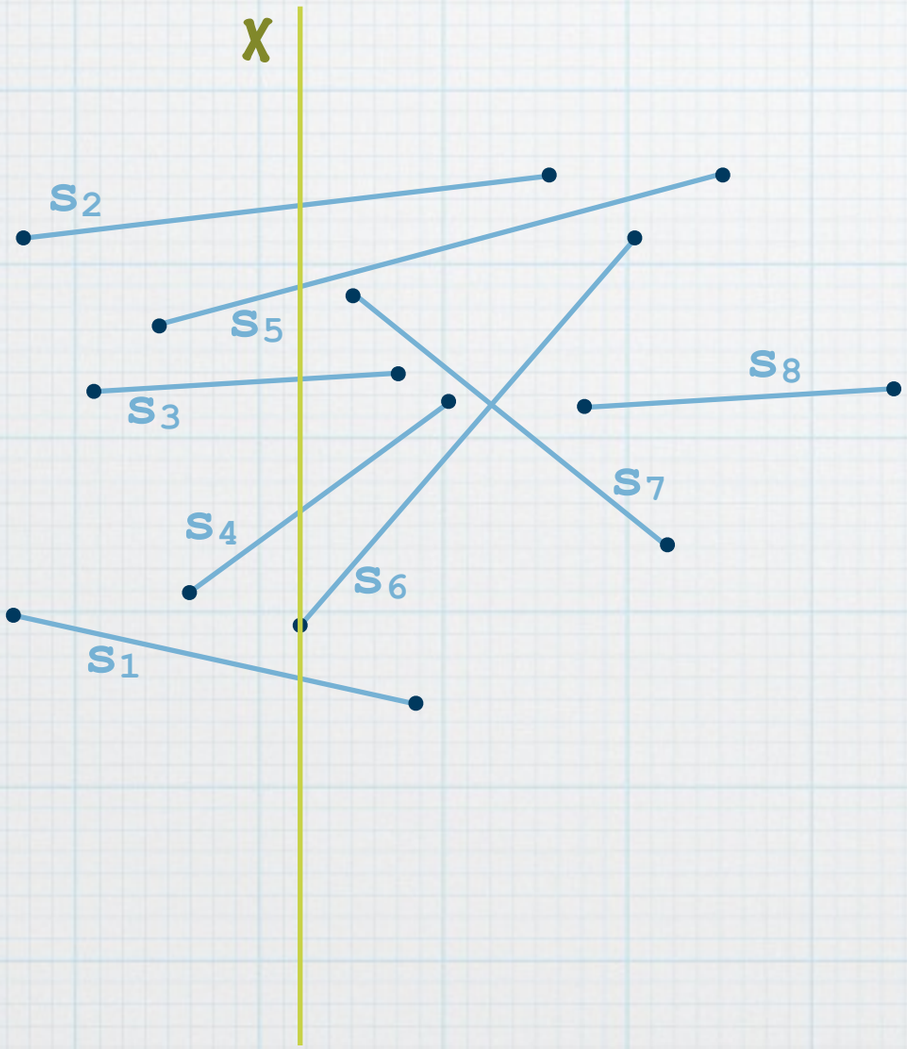


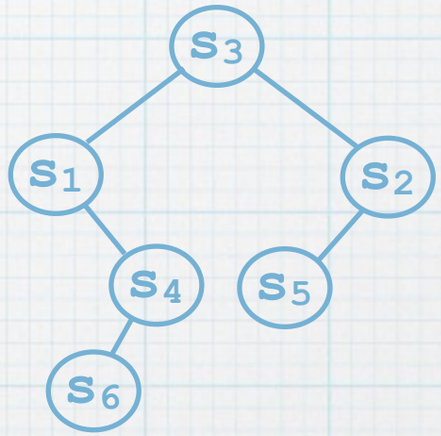
Példa



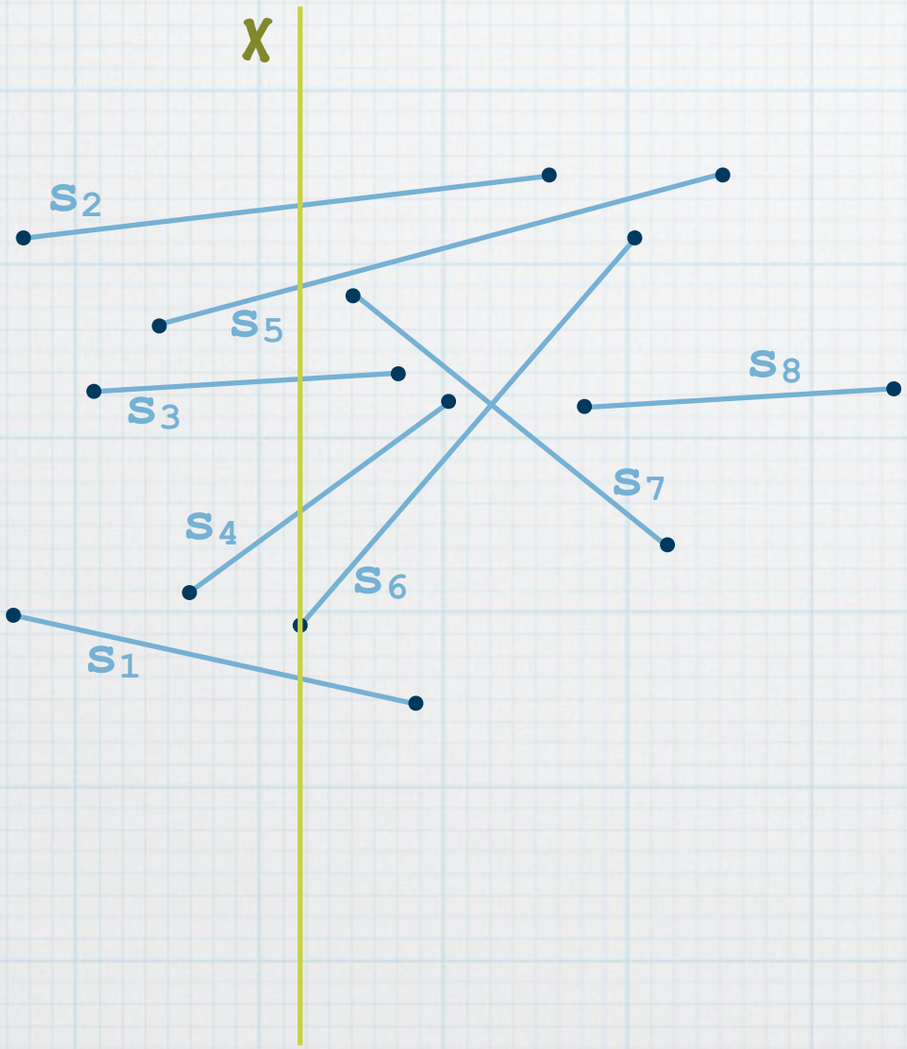


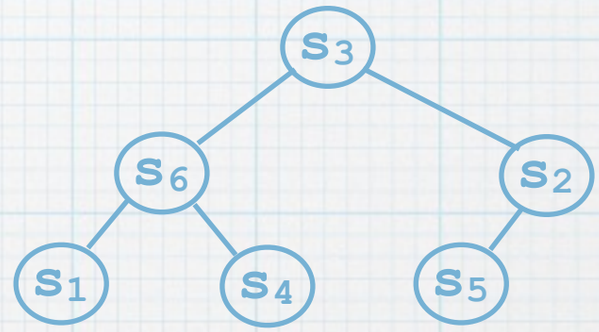
Példa



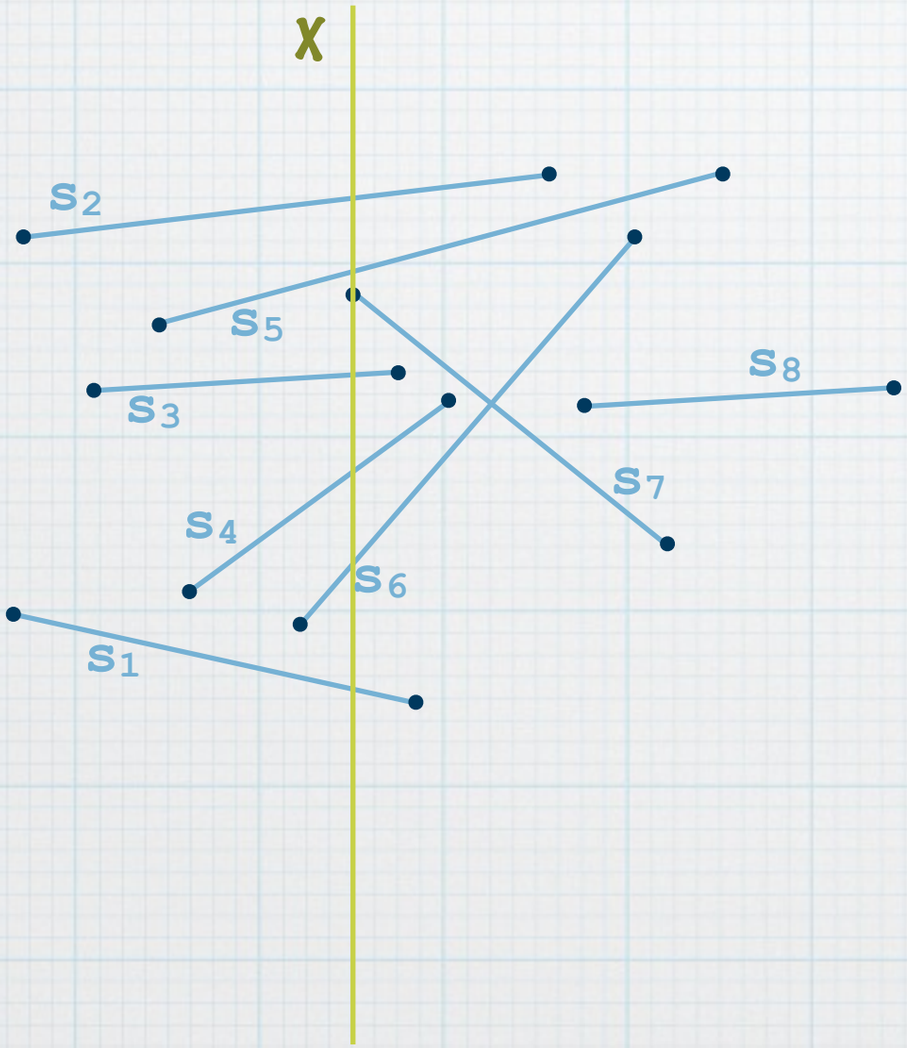


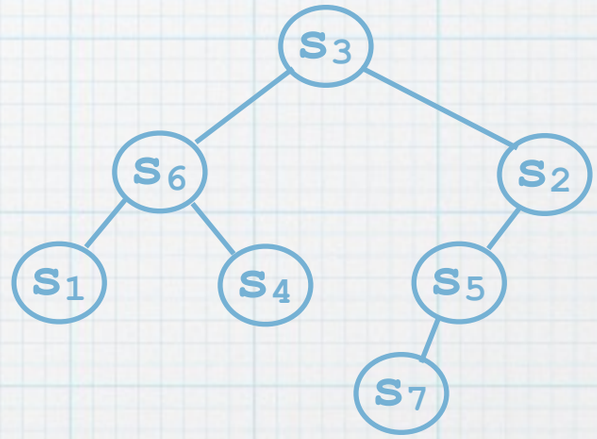
Példa



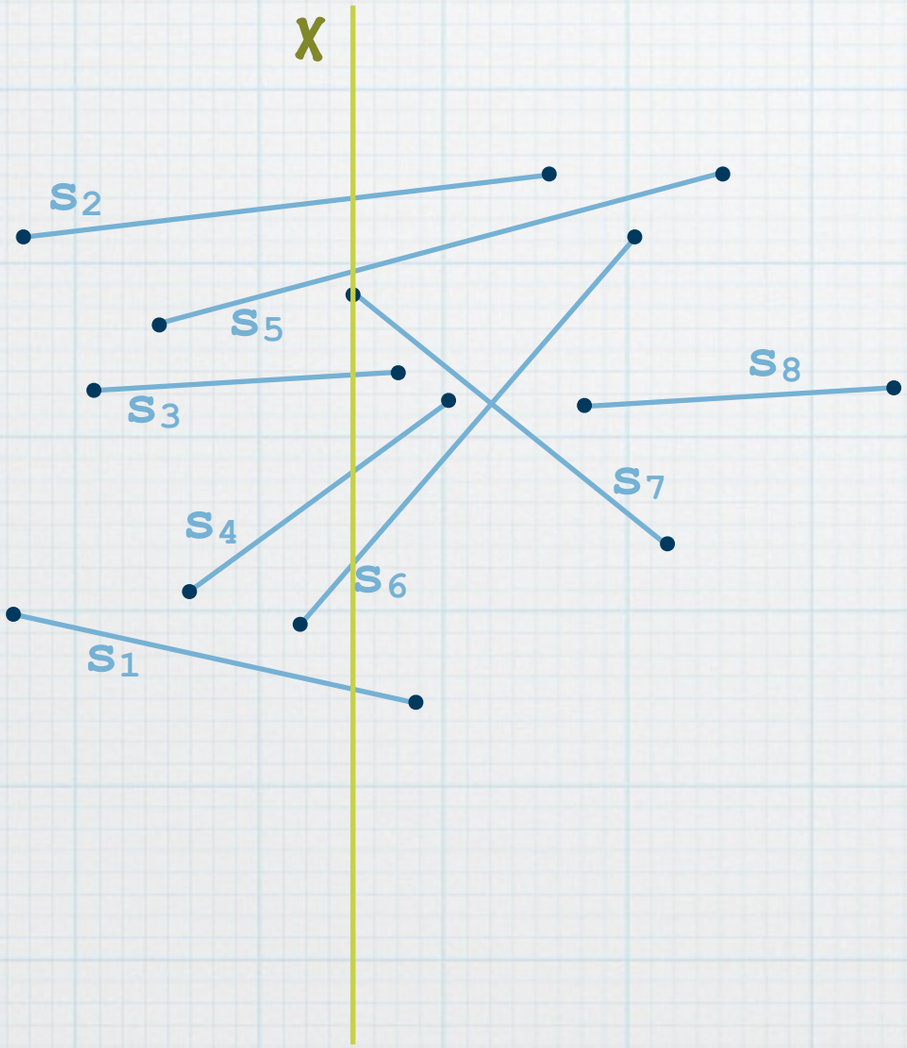


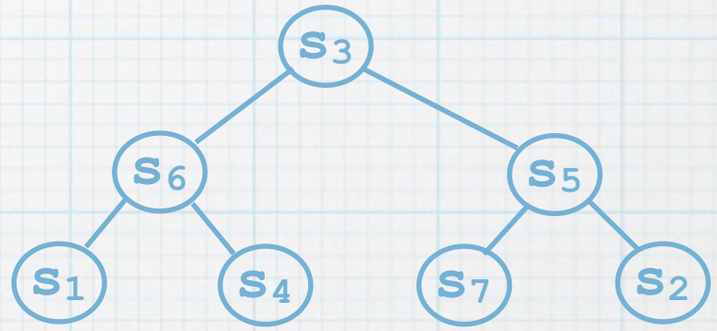
Példa



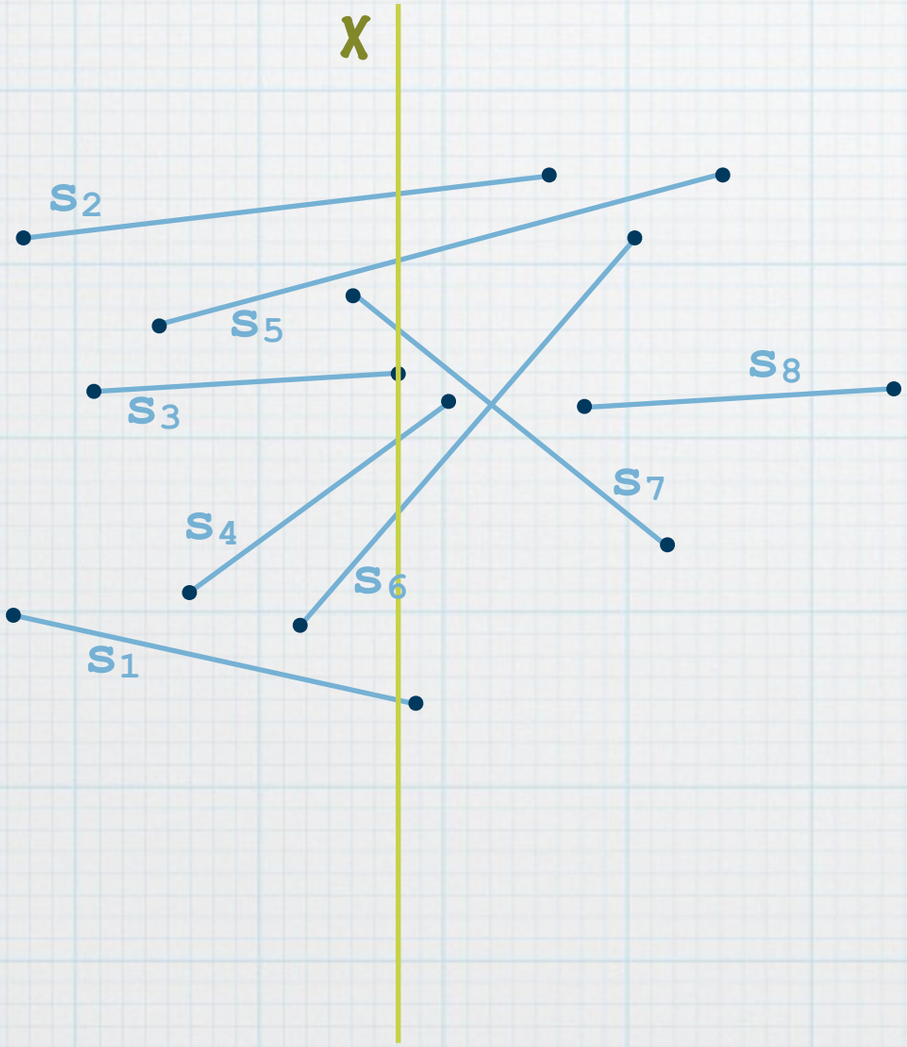


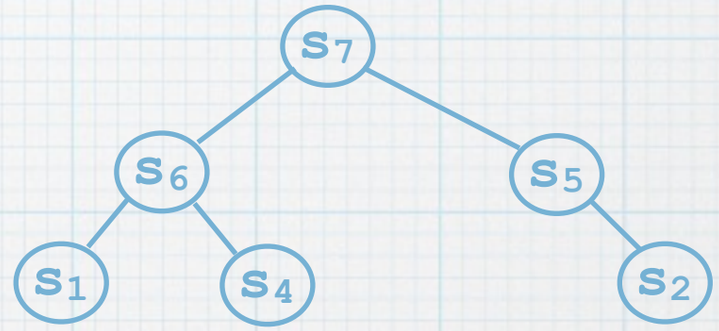
Példa



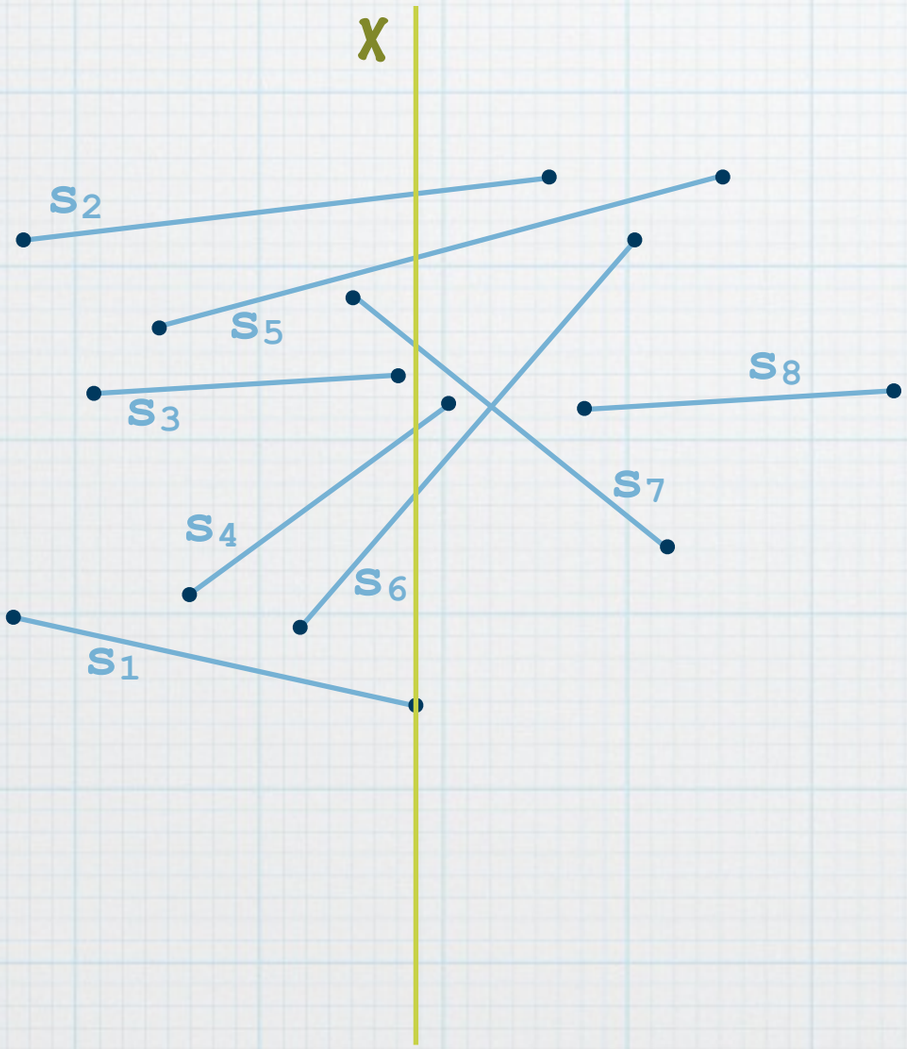


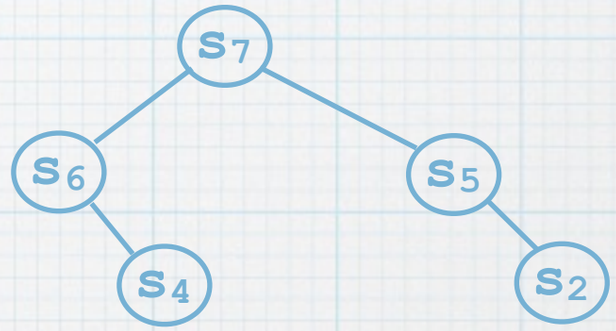
Példa



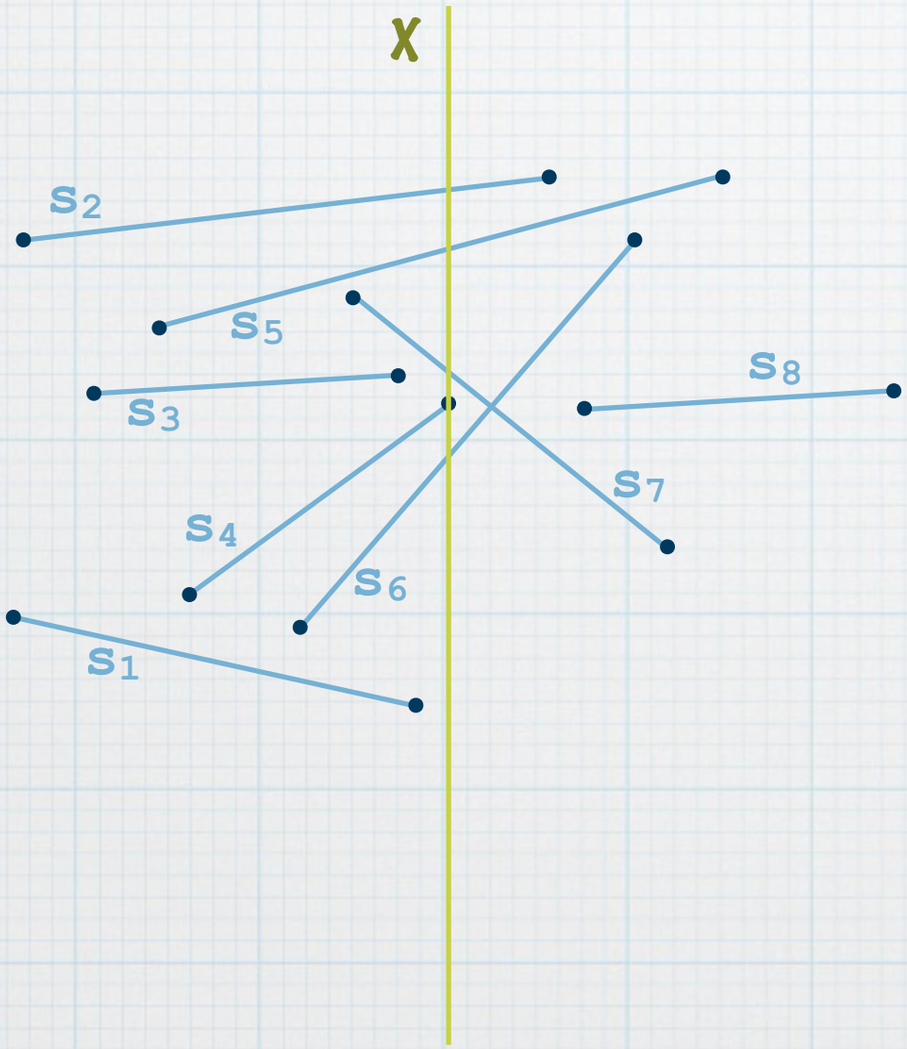


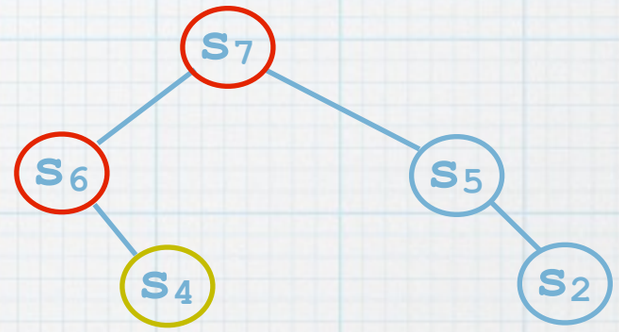
Példa





Példa





[1] T. H. Cormen, C. E. Leiserson, R.L. Rivest: Új algoritmusok
Scolar Informatika, 2003.

[2] Imreh Csanád: Algoritmusok és Adatszerkezetek II. 2009

http://www.inf.u-szeged.hu/~cimreh/n10alg211el.pdf

[3] Szabó László: Algoritmusok Feladatgyűjtemény

https://inf.nyme.hu/~lszabo/konyvek/alg/alg.pdf