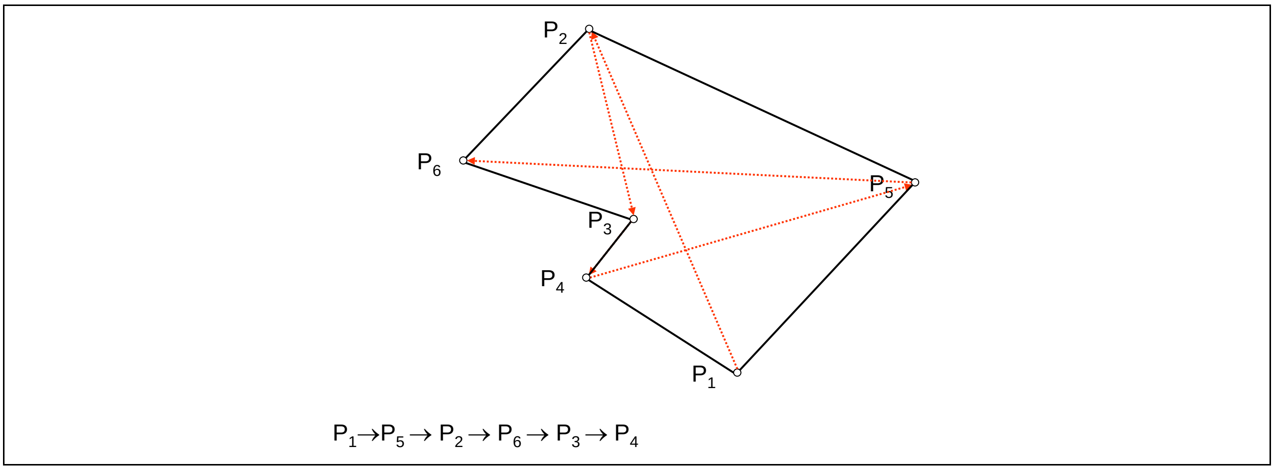
Geometriai Feladatok 1.
(esettanulmány)

Az esettanulmány Horváth Gyula: „*Geometriai algoritmusok*” c., az NJSzT által gondozott „Tehetséggondozó Program” keretén belül megjelent kötete alapján készült.

BEVEZETÉS

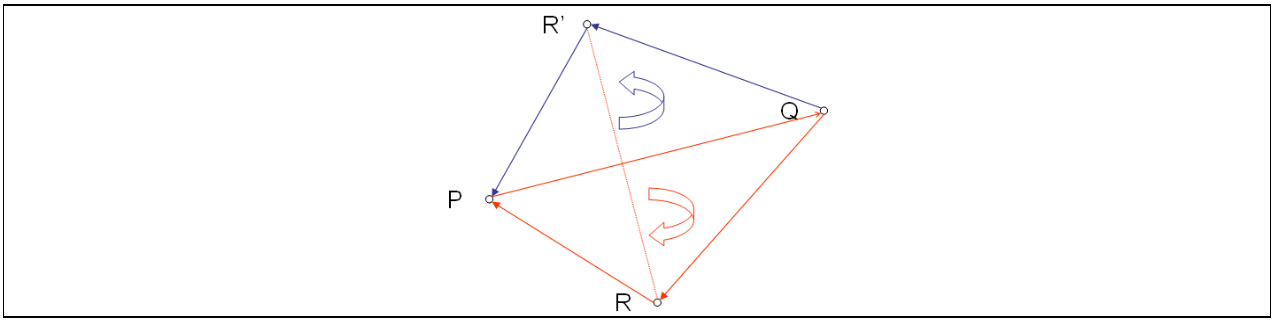
A feladatok köre…

- Pontok összekötése zárt, nem-metsző poligonná.



A nehézség: a sorrend megtalálása a „kézenfekvő" x-, ill. y-koordináták rendezésére építve.

- 3 pont „forgásiránya".



A 3 pont sorrendjének, forgásirányának „könnyű" kifejezése valamilyen „egyszerű" értékkel, Pl. ±1-gyel.

- Egy pont adott poligon belső pontja-e?

Az előzővel már megoldható.

- Egy pont adott szakaszra illeszkedik-e?

Az előzővel már megoldható?

- 2 szakasz metsző-e? Ha igen, mi a metszéspontjuk?

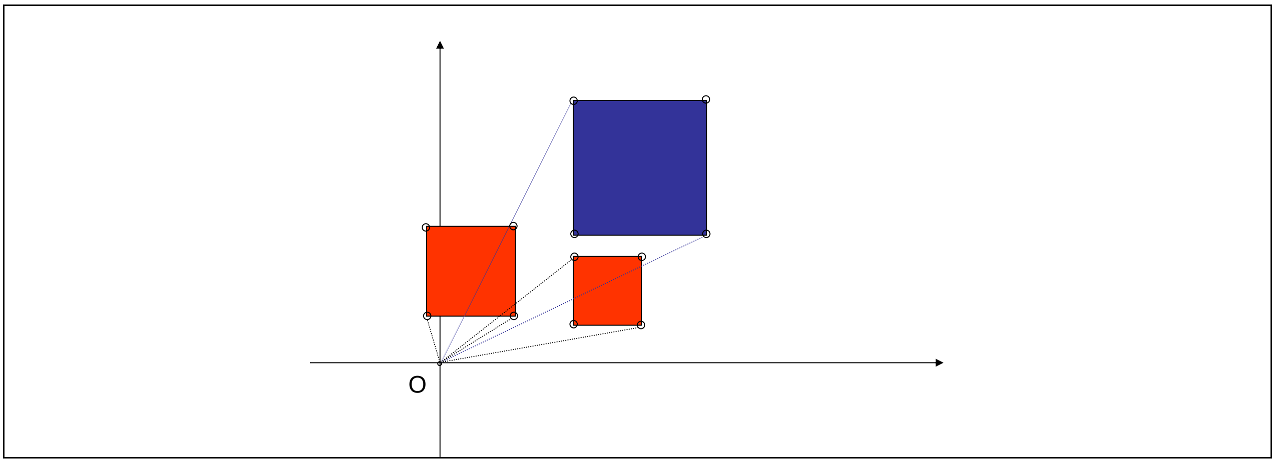
Az előzővel már megoldható?

- Ponthalmaz konvex burka.

A nehézség hasonlatos a legelsőként említett problémáéhoz.

Arra és a forgásirányra építhető is egy megoldás?

- Egy adott pontból teljesen látható négyzetek (pl. megszámlálása).



Az origóból látható négyzetek pirosak.

Alapvető típusok – alapvető műveletek

A fentiek megoldásában szereplő alapvető típusok (ábrázolás+művelethalmaz):

- pont,
- szakasz,
- pontsorozat.

AZ ALAPVETŐ TÍPUSOK MEGVALÓSÍTÁSA

Pont

- Ábrázolás:

```
Típus TPont=Rekord(x,y:Valós, hiba:Logikai)
Konst NullPont:TPont(0,0,Hamis)
```

- Asszociált műveletek szignatúrája:

```
Eljárás PontKiirás(Konst kezd:Szöveg, p:TPont, zár:Szöveg)
[kezd és zár között formázva jeleníti meg a pont koordinátáit;
 lehetővé téve az egy sorba és a külön sorba szervezést egyaránt]
```

Problémamentes.

```
Függvény PontBeolvasás(Konst kérd:Szöveg,
                              olyanE:TJóPontE):TPont
[az olyanE fv állapítja meg, hogy megfelelő-e a beolvasott pont;
 beolvas pontot, és a hiba-flagjében jelzi, hogy megfelelő-e]
```

Problémamentes.

```
Függvény X_koord(Konst p:TPont):Valós
```

Problémamentes.

```
Függvény Y_koord(Konst p:TPont):Valós
```

Problémamentes.

```
Operátor =(Konst p,q:TPont):Logikai
```
[1]

Problémamentes.

```
Operátor +(Konst p,q:TPont):TPont
```

Problémamentes.

```
Operátor –(Konst p,q:TPont):TPont
```

Problémamentes.

```
Operátor *(Konst p:TPont, a:Valós):TPont
['p' vektor az 'a' skalárral való jobbról szorzata]
```

Problémamentes.

```
Operátor *(Konst a:Valós, p:TPont):TPont
['p' vektor az 'a' skalárral való balról szorzata]
```

Problémamentes.

```
Függvény Norma(Konst p:TPont):Valós
[euklideszi-normája, azaz az origótól számított „szokásos” távol-
 sága]
```

Problémamentes.

```
Operátor *(Konst p,q:TPont):Valós
[a definíciót l. alább]
```

Def-x:

$\_ \times \_ : TPont^2 \rightarrow R$

$p_1 \times p_2 := p_1.x * p_2.y - p_2.x * p_1.y$

Megjegyzés:

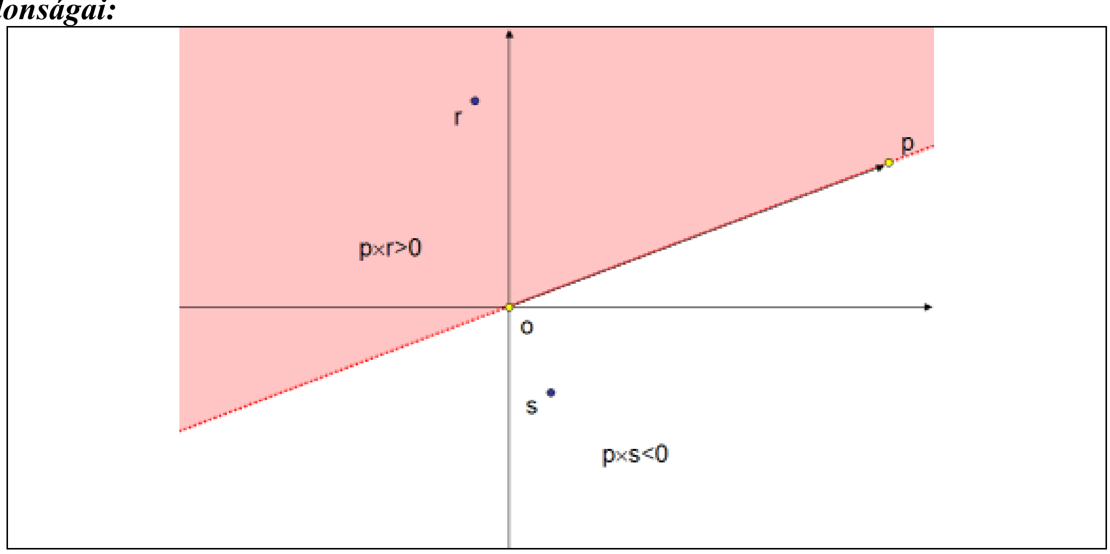
A $p_1 \times p_2$ művelet a $(\varnothing, p_1, p_2, p_1+p_2)$ pontok által kijelölt paralelogramma előjeles területét adja.

[1] Free Pascal-ban a következőt elegendő tudni az operátor készítéséhez:

```
Operator =(Const p,q:TPont) l:Boolean;
//l kapja értékül -az operátor törzsében- az operáció végeredményét
```

Tulajdonságai:

(T0)



Ha a p az 1. vagy a 4. síknegyedben van, akkor az o_p-re illeszkedő egyenes feletti pontokra a p×r>0, alatti pontokra <0;
ha a p a 2. vagy a 3. síknegyedben van, akkor pont fordítva, azaz p×r<0, alatti pontokra >0 .

Azaz a o_p-t irányvektorként tekintve az általa kijelölt irányba nézve a balra eső pontokra pozitív, jobbra eső pontokra negatív e szorzat.

Biz.: p×r=p.x*r.y-r.x*p.y>0 ⇒ p.x*r.y>r.x*p.y

ha p.x>0, akkor r.y>r.x*p.y/p.x, azaz r.y>r.x*iránytangens(p)

ha p.x<0, akkor r.y<r.x*p.y/p.x, azaz r.y<r.x*iránytangens(p)

♦

(T1) p×p=0

Biz.: p×p=

=p.x*p.y-p.x*p.y=0

♦

(T2) $p_1 \times p_2 = -p_2 \times p_1$

Biz.: $p_1 \times p_2 = p_1.x * p_2.y - p_2.x * p_1.y =$

$= -(-p_1.x * p_2.y + p_2.x * p_1.y) = -p_2 \times p_1$

♦

(T3) $(a*p_1) \times (b*p_2) = a*b*(p_1 \times p_2)$

Biz.: $(a*p_1) \times (b*p_2) = (a*p_1.x)*(b*p_2.y) - (b*p_2.x)*(a*p_1.y) =$

$= a*b*(p_1.x*p_2.y - p_2.x*p_1.y) = a*b*(p_2 \times p_1)$

♦

(T4) $(p_1+q) \times p_2 = p_1 \times p_2 + q \times p_2$

Biz.: $(p_1+q) \times p_2 = (p_1.x+q.x)*p_2.y - p_2.x*(p_1.y+q.y) =$

$= p_1.x*p_2.y + q.x*p_2.y - p_2.y*p_1.y - p_2.x*q.y =$

$= p_1 \times p_2 + q \times p_2$

♦

(T5) $p_1\times(p_2+q)=p_1\times p_2+p_2\times q$

Biz.: $p_1\times(p_2+q)=$

(T2)⇒ $=-(p_2+q)\times p_1=$

(T3)⇒ $=-p_2\times p_1-q\times p_1=$

(T2)⇒ $=p_1\times p_2+p_1\times q$

♦

(T6) $p\times\varnothing=\varnothing\times p=0$ $(\varnothing=NullPont)$

Biz.: $p\times\varnothing=p.x*0-0*p.y=0$

$\varnothing\times p=0*p.x-p.y*0=0$

♦

Megjegyzés:

(T4)&(T5) ⇒ a × disztributív a + műveletre nézve

(T3)&(T4)&(T5) ⇒ a × a vektortéren linearitást tartó leképezés

Függvény ForgásIrány(**Konst** p,q,r:TPont):{-1,0,+1}[2]

[p→q→r balforgású[3], kollineáris[4], jobbforgású esetben]

Def-KSz:

KeresztSzorzat: $TPont^3\to R$

KeresztSzorzat(p,q,r):=(q-p)×(r-p)=

=(q.y-p.y)(r.x-p.x)-(r.y-p.y)*(q.x-p.x)*

Állítás:

Ha a KeresztSzorzat(p,q,r)>0 , akkor a KeresztSzorzat(p,q,r')<0,

ahol r' az r tükörképe a p_q-ra illeszkedő egyenesre nézve.

Bizonyítás:

(Def-KSz) ⇒ KeresztSzorzat(p,q,r)=(q-p)×(r-p)

Az világos, hogy

az r a p_q-ra illeszkedő egyenes alatt van ⇔

ha r-p a 0_q-p-re illeszkedő egyenes alatt van

(T0) ⇒ (q-p)×(r-p)>0 ⇔

ha r-p a 0_q-p-re illeszkedő egyenes alatt van

No már most az r' éppen úgy helyezkedik el a p_q egyeneshez képest, mint az r'-p a 0_q-p egyeneshez képest: azaz mindkét esetben a megfelelő egyenes felett lesz.

(T0) ⇒ (q-p)×(r'-p)<0 ⇒

(q-p)×(r'-p)=KeresztSzorzat(p,q,r')<0

♦

---

[2] Elegánsabb, ha a `TFordulás=(Balra,Rajta,Jobbra)` felsorolástípussal valósítjuk meg.

[3] Azaz az óra járásával ellentétes irányúak.

[4] Azaz egy egyenesre illeszkednek.

Állítás:

a) Ha p=q vagy p=r vagy q=r, akkor

KeresztSzorzat(p,q,r)=0.

b) Ha p≠q és p≠r és q≠r, és p, q, r egy egyenesen vannak, akkor

KeresztSzorzat(p,q,r)=0

Bizonyítás:

a)

KeresztSzorzat(p,p,r):=(p-p)×(r-p)=∅×(r-p)=0 ⇐(T6)

KeresztSzorzat(p,q,p):=(q-p)×(p-p)=(r-p)×∅=0 ⇐(T6)

KeresztSzorzat(p,r,r):=(r-p)×(r-p)=0 ⇐(T1)

b)

ha p, q, r egy egyenesen vannak, akkor r' (azaz az r tükörképe a p_q-ra illeszkedő egyenesre nézve) r-rel egybe esik, így

KeresztSzorzat(p,q,r)=KeresztSzorzat(p,q,r')

másrészt az előbbi állítás miatt

KeresztSzorzat(p,q,r)=-KeresztSzorzat(p,q,r')

Vagyis

KeresztSzorzat(p,q,r)=-KeresztSzorzat(p,q,r)

Amiből már rögvest következik, hogy

KeresztSzorzat(p,q,r)=0

♦

Az utóbbi két állítás következménye, hogy a `ForgásIrány`-számítást alapozni lehet a `Kereszt-Szorzat` képletére. Írja meg a függvény algoritmusát!

Függvény HibásE(**Vált** p:TPont):Logikai

[vissza adja a p pont hiba-flagjét, s közben alaphelyzetbe hozza]

Problémamentes.

○ Szakasz

- Ábrázolás:

Típus TSzakasz=**Rekord**(k[=kezdőpont],v[=végpont]:TPont)

- Asszociált műveletek szignatúrája:

Eljárás SzakaszKiírás(**Konst** kezd:Szöveg, s:TSzakasz, zár:Szöveg)

[kezd és zár között formázva jeleníti meg a pont koordinátáit; lehetővé téve az egy sorba és a külön sorba szervezést egyaránt]

Problémamentes.

Függvény SzakaszBeolvasás(**Konst** kérd:Szöveg,
olyanE:TSzakaszE):TSzakasz
[az olyanE állapítja meg, hogy megfelelő-e a beolvasott szakasz; beolvassa a szakaszt, és a pontok hiba-flagjében jelzi, hogy megfelelő-e]

Problémamentes.

Függvény KezdőPont(**Konst** s:TSzakasz):TPont

Problémamentes.

Függvény VégPont(**Konst** s:TSzakasz):TPont

Problémamentes.

Függvény Hossz(**Konst** s:TSzakasz):Valós

Problémamentes.

Operátor =(**Konst** s1,s2:TSzakasz):Logikai
[egyenlők-e, azaz hosszuk és irányuk megegyezik-e]

Problémamentes.

Függvény Irányszög(**Konst** s:TSzakasz):Valós

Állítás:

Az s szakaszhoz jelölje R:=s.v-s.k:TPont „irányvektort”, akkor

1. Irányszög(sz)=π/2, ha R.x=0 és R.y≥0 (az „Y-tengelyen felfelé”)

*2. Irányszög(sz)=3*π/2, ha R.x=0 és R.y<0 (az „Y-tengelyen lefelé”)*

3. Irányszög(sz)=arcTg(R), ha R.x>0 és R.y≥0 (a 1. síknegyedben)

4. Irányszög(sz)=π-arcTg(R’), ha R.x<0 és R.y≥0 (a 2. síknegyedben), ahol R’:=(-R.x,R.y) [azaz az R 1. síknegyedbeli „tükörképe”]

5. Irányszög(sz)=π+arcTg(R), ha R.x<0 és R.y<0 (a 3. síknegyedben)

*6. Irányszög(sz)=2*π-arcTg(R’), ha R.x>0 és R.y<0 (a 4. síknegyedben), ahol R’:=(R.x,-R.y) [azaz az R 1. síknegyedbeli „tükörképe”]*

Bizonyítás:

Nyilvánvaló.

♦

Ezen állításból adódik már a függvény algoritmusa. Algoritmizálja!

Függvény SzakaszonE(**Konst** s:TSzakasz; r:TPont):Logikai

Állítás:

p,q,r∊TPont egy egyenesen vannak ⇔ KeresztSzorzat(p,q,r)=0

Bizonyítás:

(⇒)

p,q,r∈TPont egy egyenesen vannak ⇒ ∃λ∈R: r=p+λ(q-p) ⇒*

KeresztSzorzat(p,q,r)=(q-p)×(r-p)=(q-p)×(p+λ(q-p)-p)=*

=(q-p)×(λ(q-p))=*

(T3) ⇒ =λ(q-p)×(q-p)=*

(T1) ⇒ =λ(q-p)×(q-p)=0*

(⇐)

KeresztSzorzat(p,q,r)=0 ⇒

(q.y-p.y)(r.x-p.x)-(r.y-p.y)*(q.x-p.x)=0 ⇒*

(q.y-p.y)(r.x-p.x)=(r.y-p.y)*(q.x-p.x) ⇒*

1. ha (q.x-p.x)≠0 és (r.x-p.x)≠0 ⇒

(q.y-p.y)/(q.x-p.x)=(r.y-p.y)/(r.x-p.x) ⇒

Iránytangens(q-r)=Iránytangens(r-p) ⇒

p,q,r∈TPont egy egyenesen vannak.

2. ha (q.x-p.x)=0 ⇒

p_q-n átmenő egyenes az x-tengelyre merőleges és

(r.x-p.x)=0 vagy (q.y-p.y)=0 ⇒

2a.ha (r.x-p.x)=0 ⇒

p_r-n átmenő egyenes az x-tengelyre merőleges ⇒

p,q,r∈TPont egy egyenesen vannak.

2b. ha (q.y-p.y)=0 ⇒

p=q ⇒

p,q,r∈TPont egy egyenesen vannak.

♦

Állítás:

Ha p,q,r∈TPont egy egyenesen vannak és

r.x∈[Min(p.x,q.x)..Max(p.x,q.x)], r.y∈[Min(p.y,q.y)..Max(p.y,q.y)]

akkor az r a p_q szakaszon

Bizonyítás:

Nyilvánvaló.

♦

Ezen állításokból adódik már a függvény algoritmusa. Algoritmizálja!

Függvény SzakaszPárMetszőE(**Konst** s1,s2:TSzakasz):Logikai

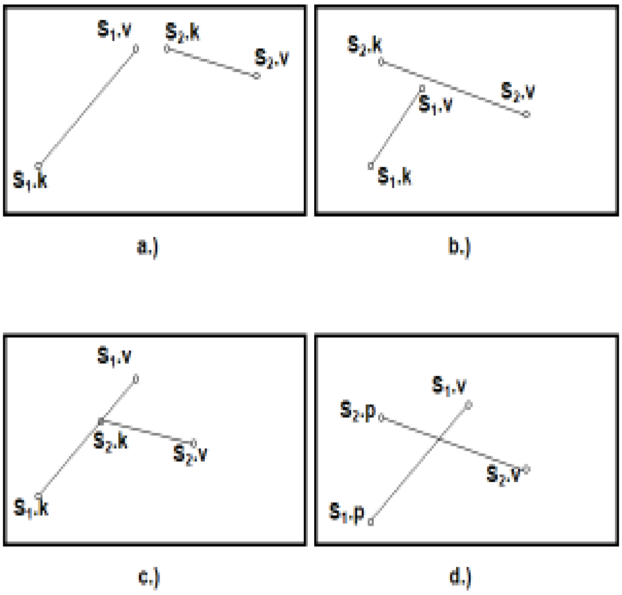
Állítás:

A szakaszok metszőség-vizsgálatát a ForgásIrány-vizsgálatra lehet alapozni.

Bizonyítás:

Az alábbi alapesetek képzelhetők el:

Alapesetek:



Az alapesetek könnyedén felírhatók formálisan a Forgásirány fogalmára visszavezetve:

a.) ForgásIrány(s_1.k,s_1.v,s_2.k)=ForgásIrány(s_1.k,s_1.v,s_2.v) és
ForgásIrány(s_2.k,s_2.v,s_1.k)=ForgásIrány(s_2.k,s_2.v,s_1.v)

b.) ForgásIrány(s_2.k,s_2.v,s_1.k)=ForgásIrány(s_2.k,s_2.v,s_1.v) és
ForgásIrány(s_1.k,s_1.v,s_2.k)=-ForgásIrány(s_1.k,s_1.v,s_2.v) vagy
ForgásIrány(s_1.k,s_1.v,s_2.k)=ForgásIrány(s_1.k,s_1.v,s_2.v) és
ForgásIrány(s_2.k,s_2.v,s_1.k)=-ForgásIrány(s_2.k,s_2.v,s_1.v)

c.) SzakaszonE(s_1.k,s_1.v,s_2.k) vagy SzakaszonE(s_1.k,s_1.v,s_2.v) vagy
SzakaszonE(s_2.k,s_2.v,s_1.k) vagy SzakaszonE(s_2.k,s_2.v,s_1.v)

d.) ForgásIrány(s_2.k,s_2.v,s_1.k)*ForgásIrány(s_2.k,s_2.v,s_1.v)<0 és
ForgásIrány(s_1.k,s_1.v,s_2.k)*ForgásIrány(s_1.k,s_1.v,s_2.v)<0

Világos, hogy a c.) és a d.) eset jelenti a megoldást; azzal a feltétel-lel, hogy egyik szakasz sem „elfajuló” (egy pontból álló), és nem esnek egy egyenesre.

♦

Ezen állításból adódik már a függvény algoritmusa. Algoritmizálja!

Eljárás SzakaszPárMetszéspont(**Konst** s1,s2:TSzakasz):TPont

Állítás:

Ha $\emptyset \neq s_1 \neq s_2 \neq \emptyset$ és van metszéspontjuk, akkor az s_1 s_2 szakaszok r metszéspontja megkapható az alábbi lineáris vektoregyenlet megoldásával:

$r_1(t_1)=r_2(t_2)$, *ahol*

$t_i \in [0..1]$ *és* $r_i(t_i)=s_i.k+t_i*(s_i.v-s_i.k)$ $(i=1,2)$.

Ekkor $r:=r_1(t_1)=r_2(t_2)$.

Bizonyítás:

Mivel egy (=nem végtelenszámú) metszéspontjuk van –a feltételek szerint–, ezért egyértelműen létezik $r_1 \in s_1$-nek és $r_2 \in s_2$-nek is olyan pontja, amelyek egybeesnek: $r_1 = r_2$.

Egy s szakasz belső pontjait befutja a következő függvény:

$r: [0..1] \to R \times R$

$r(\lambda) = s.k + \lambda * (s.v - s.k)$

Tehát egyértelműen létezik olyan $t_1, t_2 \in [0..1]$, amelyre

$r_1(t_1) = s_1.k + t_1 * (s_1.v - s_1.k) = s_2.k + t_2 * (s_2.v - s_2.k) = r_2(t_2)$.

A t_1, t_2 megkapható a fenti vektoregyenletet x- és y-koordinátára kifejtve, majd így mint két ismeretlenes lineáris egyenletrendszert megoldva.

♦

Ebből az állításból adódik már a függvény algoritmusa. Algoritmizálja!

Világos, hogy ha a `s1,s2` pontok elfajult voltát vizsgálni kell! Ekkor olyan pont kell legyen az eredmény, amelyen „látszik” az elfajultság, amelynek a hiba-flagje hibát jelez.

Függvény MerőlegesekE(**Konst** s1,s2:TSzakasz):Logikai

Állítás:

Ha $\langle s_1.v - s_1.k, s_2.v - s_2.k \rangle = 0$, akkor s_1 s_2 merőlegesek egymásra, ahol $\langle q,r \rangle := q.x*r.x + q.y*r.y$ [az a „két vektor skaláris szorzata” művelet].

Ezen állításból adódik már a függvény algoritmusa. Algoritmizálja!

Függvény Merőleges(**Konst** s:TSzakasz, r:TPont):TSzakasz

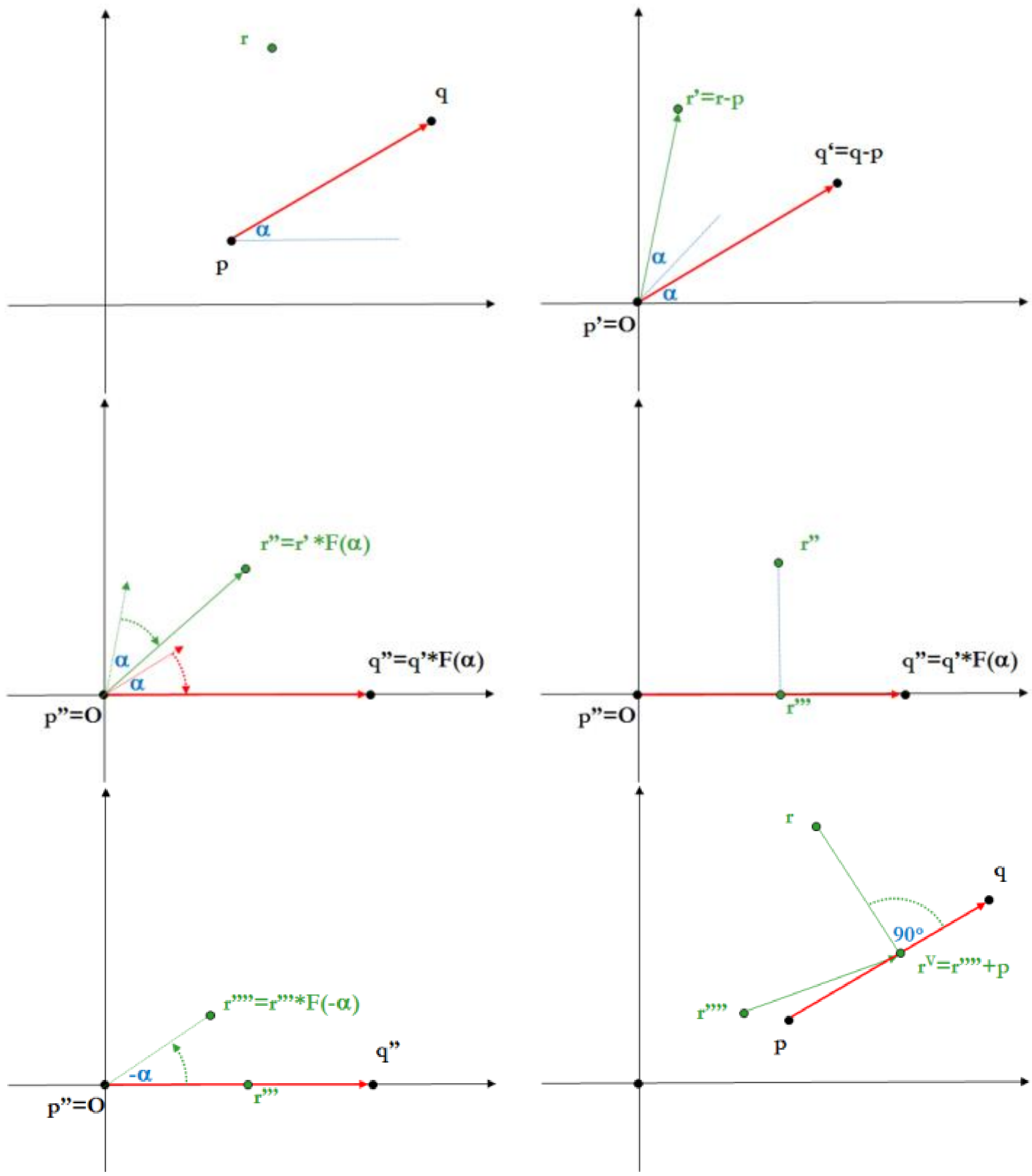
[Uf: az s szakaszt tartalmazó egyenesre merőleges szakasz, amely kezdőpontja az r]

Állítás:

Az r-n átmenő s-re[5] merőleges egyenest a következő lépésekben kaphatjuk meg (az ábra jelöléseivel):

1. α:=IránySzög(s)
2. Forgatásmátrix: $FM(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}$
3. rr forgatása: $rr := (r-p)*FM(\alpha)$
4. rr projekciója az x-tengelyre: $rr.y := 0$
5. VisszaForgatásMátrix: $FM(-\alpha)$
6. rr visszaforgatása: $p := rr*FM(-\alpha)+p$

[5] Pontosabban: …*s-n átmenő egyenesre…*



Az r-n átmenő p_q-ra merőleges egyenes, p_q metszéspontjának (r^V) „megszerkesztése”.

Ezen állításból adódik már a függvény algoritmusa. Algoritmizálja!

```
Függvény HibásE(Vált sz:TSzakasz):Logikai
[vissza adja a sz szakasz pontjainak hiba-flagjét (vagyolva),
 s közben alaphelyzetbe hozza azokat]
```

Problémamentes.

- **Pontsorozat**
  - Ábrázolás:

```
Típus TPontSor=Rekord( db:Egész,
                       pontok:Tömb(1..MaxN:TPont),
                       hiba:Logikai )
```

- Asszociált műveletek szignatúrája:

```
Konst UresPontSor:TPontSor(db:0,pontok:(),hiba:Hamis)
```
Problémamentes.

```
Függvény ElemSzám(Konst ps:TPontSor):Egész
```
Problémamentes.

```
Függvény Elem(Konst ps:TPontSor, Konst i:Egész):TPont
[p=Elem(ps,i): i∈[1..ps.db], akkor p a ps pontsort i. pontja,
 különben p.hiba=Igaz]
```
Problémamentes.

```
Eljárás ElemMódosít(Vált ps:TPontSor,
                    Konst i:Egész, p:TPont):
[ps.pontok[i]=p: i∈[1..ps.db], különben ps.hiba=Igaz]
```
Problémamentes.

```
Operátor +(Konst ps:TPontSor, p:TPont):TPontSor
[bővíti a pontsort újabb ponttal, ha ps.db<MaxN;
 különben (ps+p).hiba=Igaz]
```
Problémamentes.

```
Operátor –(Konst ps:TPontSor, i:Egész):TPontSor
[elhagyja a pontsorból az i. pontot, ha i∈[1..ps.db];
 különben (ps-p).hiba=Igaz]
```
Problémamentes.

```
Függvény HibásE(Vált ps:TPontSor):Logikai
[hibás volt-e a ps; törli a hiba-flaget]
```
Problémamentes.

1. FELADAT

Készítse el a geometriai típusok teljes, egyesített modulját!

2. FELADAT

Készítse el a GeomUnit-ot tesztelő programot!