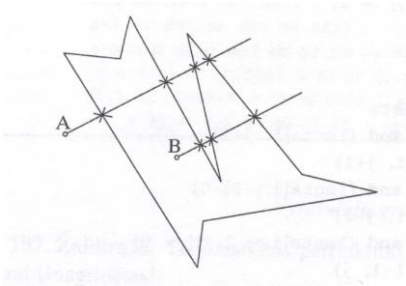
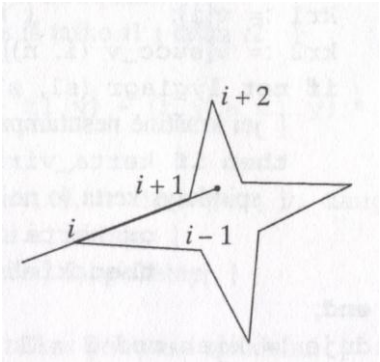
# Nesėkmingas skrydis oro balionu

Norint išspręsti šį uždavinį, reikia žinoti šiek tiek geometrijos. Idėja labai paprasta: iš taško, kuriame nusileido daugiakampis brėžiame spindulį bet kuria kryptimi. Suskaičiuojame kiek kartų spindulys kerta daugiakampį. Jei lyginį kartų skaičių – balionas nusileido valstybės teritorijoje, jei nelyginį – už jos ribų:



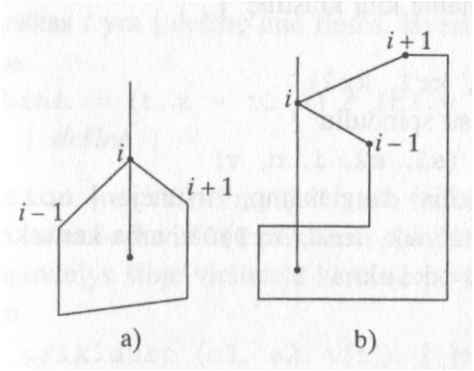

Taigi, imame visas daugiakampio briaunas ir skaičiuojame kelios iš jų kerta spindulį. Reikia nepamiršti keleto atskirų atveju:

a) spindulys yra lygiagretus daugiakampio kraštinei ir ją liečia:




tada situacijoje spindulys daugiakampio nekerta.

b) spindulys kerta daugiakampio viršūnę:




Pirmuoju atveju kirsdamas viršūnę, spindulys tik liečia daugiakampį, antruoju – kerta. Tarkime, kad kertamos viršūnės numeris yra *i*. Tuomet reikia paimti *i*-1-ąją ir *i*+1-mąją viršūnes ir patikrinti ar jos yra toje pačioje tiesės (gautos pratęsus spindulį) pusėje.

Remdamiesi šia idėja parašome algoritmą.

```
const MAX_N = 1000;  { maksimalus daugiakampio viršūnių skaičius }
      type taškas =  record
                       x, y: -1..1000;
                     end;
           daugiak = array [1..MAX_N] of taškas;

function succ_v (i, n: integer): integer;
begin
  if i = n
     then succ_v := 1
     else succ_v := i + 1
end;  { succ_v }

function pred_v (i, n: integer): integer;
begin
  if i = 1
     then pred_v := n
     else pred_v := i - 1
end;  { succ_v }
```

```
function viduje (v: daugiak; n: integer; { daugiakampis }
                 t: taškas { keliautojų koordinatės }): boolean;
  var i,
      kiek: integer;   { kiek briaunų kerta spindulį }
      br1, br2, s1, s2: taškas;
      prikl: boolean;  { ar taškas priklauso daugiakampio briaunai }
begin
  { patikrinsime, ar taškas priklauso daugiakampio briaunai }
  prikl := false;
  for i :=  1 to n do
    prikl := prikl or priklauso(v[i], v[succ_v(i,n)], t);
  if prikl     { jei keliautojai nusileido ant sienos }
  then viduje := true
  else         { priešingu atveju }
    begin
      s1.x := -1; s1.y := -1;  { spindulys tikrai išeis už }
      s2 := t;                 { daugiakampio ribų }
      kiek := 0;
      for i := 1 to n do { skaičiuosime kiek kraštinių kerta spindulį }
        begin
          br1 := v[i];        { imame kitą briauną }
          br2 := v[succ_v (i, n)];
          if not lygiagr (s1, s2, br1, br2)
             { jei briauna nelygiagreti spinduliui }
             then if virš (s1, s2, i, n, v)
             { spindulys kerta (o ne liečia) daugiakampį viršūnėje}
             or kerta (s1, s2, br1, br2) { arba kerta briauną }
             then kiek := kiek + 1;
        end;
      viduje := kiek mod 2 = 1
    end
end; { viduje }
```

Dabar galime detalizuoti procedūras. Tarkime, kad tiesė nusakoma dviem taškas $t1(x_1, y_1)$ ir $t2\ (x_2, y_2)$ ir turime trečią tašką $t\ (x, y)$. Iš geometrijos žinome kad, jei :

$$\frac{(x-x_1)}{(x_2-x_1)} - \frac{y-y_1}{y_2-y_1}: \begin{cases} < 0, & \text{tai taškas } t \text{ yra kairėje tiesės pusėje;} \\ = 0, & \text{tai taškas } t \text{ priklauso tiesei;} \\ > 0, & \text{tai taškas } t \text{ yra dešinėje tiesės pusėje.} \end{cases}$$

jei tiesės kryptis yra iš taško $t_1$ į $t_2$.

Tikrinant, ar keliautojai nusileido ant sienos reikia patikrinti, ar taškas priklauso atkarpai. Taip bus jei taškas priklausys tiesei, gautai pratęsus atkarpą ir jei taško koordinatės bus apribotos atkarpos galų koordinatėmis:

```
function priklauso (t1, t2, t: taškas): boolean;
{ ar taškas t priklauso atkarpai, išvestai per taškus t1 ir t2 }
begin
  priklauso := ((t.x-t1.x) * (t2.y-t1.y) - { priklauso tiesei }
                (t.y-t1.y) * (t2.x-t1.x) = 0) and
             ((t1.x <= t.x) and (t.x <= t2.x) or { ir atkarpai }
              (t2.x <= t.x) and (t.x <= t2.x)) and
             ((t1.y <= t.y) and (t.y <= t2.y) or
              (t2.y <= t.y) and (t.y <= t2.y))
end;  { priklauso }
```

Dvi tiesės, išvestos per taškus ($t1$, $t2$) ir ($t3$, $t4$) yra lygiagrečios, jei:

$$\frac{(x_2 - x_1)}{(y_2 - y_1)} = \frac{x_4 - x_3}{y_4 - y_3}.$$

```
function lygiagr (t1, t2, t3, t4: taškas): boolean;
{ ar tiesė, išvesta per taškus t1 ir t2 lygiagreti tiesei }
{ išvestai per kitus du taškus }
begin
  lygiagr := (t2.x - t1.x) * (t4.y - t3.y) =
             (t2.y - t1.y) * (t4.x - t3.x)
end;  { lygiagr }
```

Pereiname prie funkcijos *virš*. Ar *i*-toji viršūnė priklauso spinduliui, galima nustatyti pasinaudojant funkcija *priklauso*. Jei taip, imame dvi gretimas viršūnes (i-1-mą ir iž1-mą) ir patikriname ar jos abi yra priešingose tiesės (gautos pratęsus spindulį) pusėse. Jei taip – daugiakampis kertamas, jei ne – liečiamas. Patikrinti galima pagal (1) formulę:

```
function dešinė (t1, t2, t: taškas): boolean;
{ ar taškas t yra dešinėje tiesės išvestos per taškus }
{ t1 ir t2 pusėje. Tiesės kryptis iš t1 į t2 }
begin
  dešinė := (t.x - t1.x) * (t2.y - t1.y) -
            (t.y - t1.y) * (t2.x - t1.x) > 0
end; { dešinė }

function virš (s1, s2: taškas; i, n: integer; v: daugiak)
                                                  : boolean;
{ tikrina, ar i-toji viršūnė priklauso spinduliui ir jei taip, }
{ ar spindulys šioje viršūnėje kerta (o ne liečia) daugiakampį }
begin
  if priklauso (s1, s2, v[i]) { jei taškas priklauso spinduliui }
     then virš := dešinė (s1, s2, v[succ_v(i, n)]) <>
                  dešinė (s1, s2, v[pred_v(i, n)])
     else virš := false
end; { virš }
```

Liko patikrinti, ar spindulys kerta (o ne liečia!) atkarpą. Atrodytų, kad tai gan paprasta: spindulį pratęskime iki tiesės ir patikrinkime, ar abu atkarpos galai yra priešingose tiesės pusėse. Tačiau jei tiesė kerta atkarpą, dar nebūtinai spindulys ją kirs (žr 69.3 pav.). Todėl reikia pratęsti atkarpą iki tiesės, parinkti tikrai daugiakampiui nepriklausantį spindulio tašką ir patikrinti ar spindulio pradžia ir pasirinktasis taškas tikrai yra priešingose tiesės pusėse.

T. y. faktiškai paimamos dvi atkarpos ir tikrinama, ar jos nesikerta.

```
function kerta (a1, a2, b1, b2: taškas): boolean;
{ ar atkarpa |a1, a2| kerta atkarpą |b1, b2| }
{ jei atkarpos nėra lygiagrečios }
begin
  kerta := (dešinė(a1, a2, b1) <> dešinė(a1, a2, b2)) and
           (dešinė(b1, b2, a1) <> dešinė(b1, b2, a2)) and
           not priklauso(a1, a2, b1) and
           not priklauso(a1, a2, b2)
end; { kerta }
```