Tiesės

Pirmiausia suskaičiuokime visus tiesių tarpusavio susikirtimo taškus. Jei kurios nors dvi tiesės sutampa, laikome, kad jos neturi bendrų taškų. Norint sudaryti keturkampį, susikirtimo taškų skaičius turi būti ne mažesnis kaip keturi. Jei kiekviena tiesė kertasi su visomis kitomis skirtinguose taškuose, gali būti daugiausia šeši susikirtimo taškai.

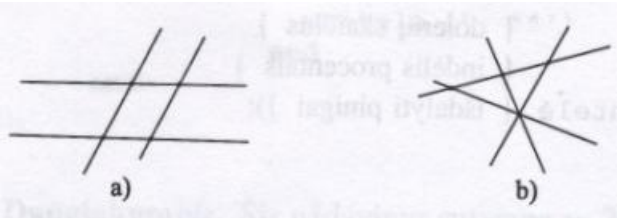
Panagrinėkime šiuos tiesių susikirtimo variantus.

1. *Tiesės kertasi keturiuose taškuose*. Taip gali atsitikti, jei

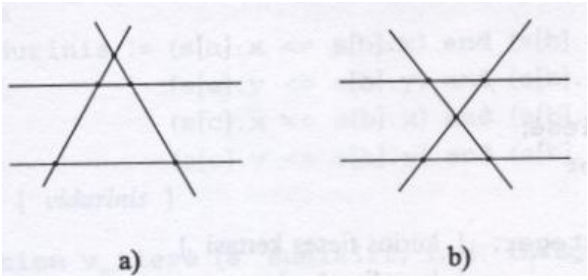
a) tiesės yra poromis lygiagrečios;

b) trys tiesės kertasi viename taške, ir nėra lygiagrečių tiesių.

Pirmuoju atveju visada gausime keturkampį, antruoju atveju jo niekada negausime, nes trys iš keturių taškų bus vienoje tiesėje.

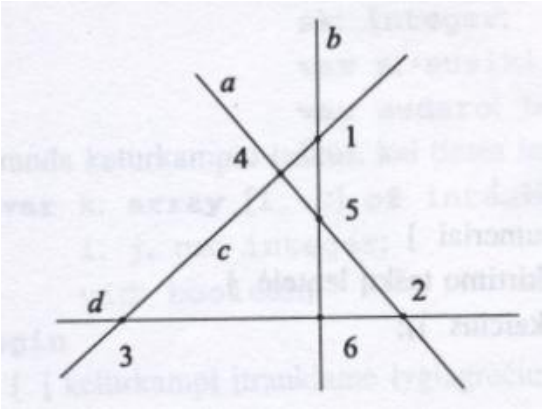


2. *Tiesės kertasi penkiuose taškuose*. Taip gali atsitikti tik tada, kai dvi tiesės yra lygiagrečios, o likusios dvi kertasi:



Matome, kad pirmuoju atveju keturkampį sudaryti galima, antruoju – ne.

3. *Tiesės kertasi šešiuose taškuose*. Taip yra, kai jokios trys tiesės nesikerta viename taške ir jokios dvi tiesės nėra lygiagrečios:



Šiuo atveju visuomet galima sudaryti keturkampį.

Dabar nesunkiai galime nustatyti, ar įmanoma sudaryti keturkampį. Jei yra šeši susikirtimo taškai, tai tiesės sudaro keturkampį. Jei penki – reikia patikrinti, ar dviejų nelygiagrečių tiesių susikirtimo taškas yra lygiagrečių tiesių „išorėje", o jei keturi – pakanka patikrinti, ar yra lygiagrečių tiesių.

Jei žinome, kad tiesės sudaro keturkampį, reikia rasti koordinates. Kai susikirtimo taškai keturi, viskas paprasta – tie taškai yra ieškomieji. Kai yra penki susikirtimo taškai, reikia atmesti nelygiagrečių tiesių susikirtimo tašką, o likę bus keturkampio koordinatės.

Kebliau, kai yra 6 susikirtimo taškai. Galima ieškoti tokiu būdu. Kiekviena tiesė kertasi su kitomis trimis. Vienas susikirtimo taškas ant konkrečios tiesės yra tarp kitų dviejų (27.3 pav. žr. 4,5 ir 6 taškus). Iš šių trijų taškų tiesei A priklauso taškai 4 ir 5, tiesei B – 5 ir 6, tiesei C – 4, tiesei D – 6. Taigi paskutinysis taškas bus tiesių C ir D susikirtimo taškas.

Gali kilti klausimas, kaip nustatyti, ar tiesės yra lygiagrečios. Tarkime, kad duotos dvi tiesės $y = a_1x + b_1$, $y = a_2x + b_2$. Jos yra lygiagrečios, kai $a_1 = a_2$.

Jei dvi tiesės kertasi, jų susikirtimo taško (x, y) koordinatės apskaičiuojamos pagal formulę:

$x = (b_2 - b_1) / (a_1 - a_2)$;
$y = (a_1b_2 - a_2b_1) / (a_1 - a_2)$

```
const MAX = 6;  { maksimalus susikirtimų skaičius }
      N = 4;    { maksimalus tiesių skaičius }
type tiesė = record
               a, b: integer
             end;
     tiesės = array [1..N] of tiesė;
     susikirt = array [1..MAX] of
                  record
                    nr1, nr2: integer;  { kurios tieses kertasi }
                    x, y: real;         { koordinatės }
                    prikl: boolean      { ar priklauso keturkampiui }
                  end;
      aibė = set of 1..4;

function lyg (t1, t2: tiese): boolean;
{ nustato, ar dvi tiesės yra lygiagrečios }
begin
  lyg := t1.a = t2.a
end; { lyg }

procedure kertasi (t: tiesės;          { tiesės }
                   i, j: integer;      { jų numeriai }
                   var s: susikirt;    { susikirtimo taškų lentelė }
                   var sk: integer     { jų skaičius });
{ randa tiesių i ir j susikirtimo tašką }
  const eps = 0.00000001;
  var buvo: boolean;
begin
  if t[i].a <> t[j].a then { jei tiesės nelygigagrečios }
   begin
     sk := sk + 1;
     s[sk].nr1 := i;
     s[sk].nr2 := j;
     s[sk].prikl := false;
     s[sk].x := (t[j].b - t[i].b) / (t[i].a - t[j].a);
     s[sk].y := (t[i].a*t[j].b - t[j].a*t[i].b) /
                (t[i].a - t[j].a);
     { tikrinsime, ar tokio taško nebuvo }
     buvo := false;
     for i := 1 to sk-1 do
       if (abs (s[i].x - s[sk].x) < eps) and
          (abs (s[i].y - s[sk].y) < eps)
          then buvo := true;
            if buvo then
               sk := sk - 1
   end
```

```
end;  { kertasi }

function vidurinis (s: susikirt; a, b, c: integer):
boolean;
{ patikrina, ar taškas b yra tarp taškų a ir c, kai }
{ žinoma, kad tie taškai tikrai yra vienoje tiesėje }
begin
  vidurinis :=
    (s[a].x <= s[b].x) and (s[b].x <= s[c].x) and
    (s[a].y <= s[b].y) and (s[b].y <= s[c].y) or
    (s[c].x <= s[b].x) and (s[b].x <= s[a].x) and
    (s[c].y <= s[b].y) and (s[b].y <= s[a].y)
end;  { vidurinis }

function v_tiesė (s: susikirt; i, j: integer): boolean;
{ patikrina, ar susikirtimo taškai i ir j yra vienoje tiesėje }
begin
  v_tiesė := (s[i].nr1 = s[j].nr1) or (s[i].nr1 =
s[j].nr2) or
             (s[i].nr2 = s[j].nr1) or (s[i].nr2 =
s[j].nr2)
end;  { v_tiesė }

procedure penki (t: tiesės;
                 sk: integer;       { susikirtimų skaičius }
                 var s: susikirt;   { susikirtimo taškai }
                 var sudaro: boolean);
{ randa keturkampio taškus, kai tiesės kertasi penkiuose taškuose }
  var k: array [1..N] of integer;  { kiek taškų kerta i-ąją tiesę }
      i, j, nm: integer;
      vid: boolean;
begin
  { į keturkampį įtraukiame lygiagrečių tiesių taškus }
  for i := 1 to N do
    k[i] := 0;
  for i := 1 to sk do
    begin
      k[s[i].nr1] := k[s[i].nr1] + 1;
      k[s[i].nr2] := k[s[i].nr2] + 1;
    end;
  for i := 1 to sk do
    if k[s[i].nr1] = 2
       then s[i].prikl := true
       else nm := i;  { penktojo taško numeris }
```

```
  { tikriname, ar tinka penktasis, t. y. ar jis nėra viduryje tarp kurių nors dviejų taškų }
   vid := false;
   for i := 1 to sk do
    for j := i+1 to sk do
      if (i <> nm) and (j <> nm) and
         v_tiesė (s, i, j) and v_tiesė (s, i, nm)
         { jei visi trys taškai yra vienoje tiesėje }
         and vidurinis (s, i, nm, j)
         then vid := true;
   sudaro := not vid
end;  { penki }

procedure vid (var s: susikirt; sk, i: integer;
               var aib: aibė);
{ randa i-osios tiesės vidurinįjį susikirtimo tašką }
{ ir jį įtraukia į keturkampį }
  var num: array [1..3] of integer;
      k, j, nm: integer;
begin
  k := 1;  j := 0;
  while j < 3 do { surandame visus tris i-osios tiesės susikirtimo }
    begin         { taškus }
      if (s[k].nr1 = i) or (s[k].nr2 = i)
         then begin
                j := j + 1;
                num[j] := k
              end;
      k := k + 1;
    end;
  if vidurinis (s, num[1], num[2], num[3])
     then nm := num[2]
     else if vidurinis (s, num[2], num[1], num[3])
             then nm := num[1]
             else nm := num[3];
  { tašką įtrauksime į keturkampį }
  s[nm].prikl :=  true;
  if s[nm].nr1 in aib
     then aib := aib - [s[nm].nr1]
     else aib := aib + [s[nm].nr1];
  if s[nm].nr2 in aib
     then aib := aib - [s[nm].nr2]
     else aib := aib + [s[nm].nr2];
end;  { vid }
```

```
procedure šeši (t: tiesės;
                sk: integer { jų skaičius };
                var s: susikirt { susikirtimo taškai });
{ randa keturkampio taškus, kai tiesės kertasi šešiuose taškuose }
  var aib: aibė;
      i: integer;
begin
  aib := [];
  { rasime tiesių vidurinius taškus ir juos įjungsime į keturkampį }
  for i := 1 to N do
    vid (s, sk, i, aib);
  { rasime ketvirtą tašką }
  for i := 1 to sk do
    if not s[i].prikl and (s[i].nr1 in aib) and
       (s[i].nr2 in aib)
       then s[i].prikl := true
end; { šeši }

procedure keturkampis (t: tiesės; { duotosios tiesės }
                 var s: susikirt;{ keturkampio viršūnių koordinatės }
                 var sudaro: boolean; { ar sudaro keturkampį }
                 var sk: integer { susikirtimo taškų skaičius });
  var i, j: integer;
begin
  { rasime visus tiesių tarpusavio susikirtimo taškus }
  sk := 0;
  for i := 1 to N do
    for j := i + 1 to N do
      kertasi (t, i, j, s, sk);
  case sk of
    0, 1, 2, 3: sudaro := false;
    4: begin
         sudaro := lyg (t[1], t[2]) or
                   lyg (t[1], t[3]) or lyg (t[1], t[4]);
         if sudaro
            then for i := 1 to sk do
                   s[i].prikl := true
       end;
    5: penki (t, sk, s, sudaro);
    6: begin
         sudaro := true;
         šeši (t, sk, s)
       end
   end
end; { keturkampis }
```