Daugiakampis

Sužinoti, ar galima iš duotų taškų sudaryti daugiakampį yra nesunku: tereikia patikrinti, ar taškų yra daugiau negu du ir ar ne visi taškai yra vienoje tiesėje. Tikrinant taškų priklausomybę vienai tiesei, pasirenkami bet kurie du taškai, parašoma tiesės per du taškus lygtis:

$$\frac{x - x_1}{x_1 - x_2} = \frac{y - y_1}{y_1 - y_2}$$

čia (x_1, y_1) yra pirmojo pasirinkto taško koordinatės, (x_2, y_2) – antrojo.

Į lygtį įstatomos visų likusių taškų koordinatės. Jei visi taškai tenkina šią tapatybę, jie yra vienoje tiesėje.

```
const MAXN = 100;  { didžiausias galimas taškų skaičius }
type taškas = record
                 x, y: integer
              end;
     tdaugiakampis = array [0..MAXN] of taškas;
     { taškai numeruojami nuo 0 iki n }

function ant_tiesės (A, B, C: taškas): boolean;
{ ar taškas C yra vienoje tiesėje su taškais A ir B }
begin
   ant_tiesės := (C.y - B.y) * (B.x - A.x) =
                 (C.x - B.x) * (B.y - A.y)
end;  { ant_tiesės }

function ne_tiesėje (d: tdaugiakampis;
                     n: integer): boolean;
{ patikrina ar ne visi duoti taškai yra vienoje tiesėje }
  var i: integer;
      nėra: boolean;
begin
  nėra := true;
  i := 2;
  { tikrinama, ar yra bent vienas taškas nepriklausantis tiesei, einančiai per d[0] ir d[1] }
  while nėra and (i <= n) do
    begin
      nėra := ant_tiesės (d[i], d[0], d[1]);
      i := i + 1
    end;
    ne_tiesėje := not nėra;
  end;   { ne_tiesėje }
```

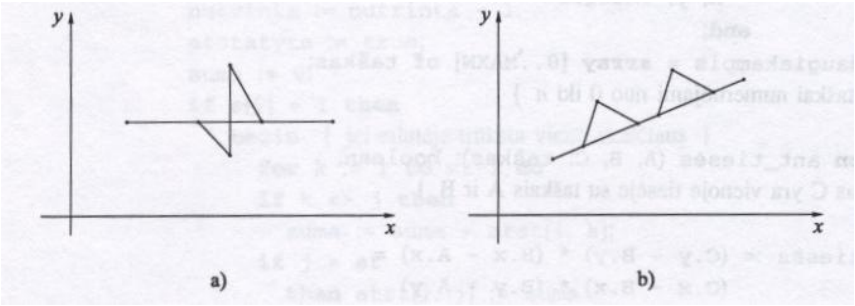
Antroji uždavinio dalis – sunumeruoti taškus taip, kad jie sudarytų daugiakampį (jei galima) yra sunkesnė. Atrodytų, kad bet kokį daugiakampį sudaryti nėra sunku: sąlygoje nepasakyta, kad daugiakampis turėtų būti iškilusis. Vadinasi, daugiakampis gali būti ir neiškilusis, ir žvaigždinis (turėti susikertančių kontūrų) – apie tokius daugiakampius rašoma įvairiuose matematikos žinynuose.

Atrodo, kad tiktų paprasčiausias algoritmas: surikiuoti taškus koordinatės *x* didėjimo tvarka, jei yra keli taškai su vienoda *x* koordinate – surikiuoti juos *y* koordinatės didėjimo tvarka. Taip surikiuotas masyvas rodytų jungiamų taškų tvarką. Paskutiniame masyvo elemente įrašytą tašką sujungtumėme su pirmuoju tašku ir turėtume gauti daugiakampį (kuris gali turėti ir susikertančių kontūrų ir didesnių nei 180^0 kampų). Tačiau nesunku įsitikinti, kad taip jungdami taškus, galime gauti figūras, su įvairiomis „uodegomis“.

Jų negalima pavadinti daugiakampiais. Akivaizdu, kad minėtas algoritmas netinka.

Taigi, uždavinio esmė yra surikiuoti duotus taškus taip, kad jungiant juos iš eilės gautos naujos atkarpos nors dalinai nesutaptų su ankstesnėmis.

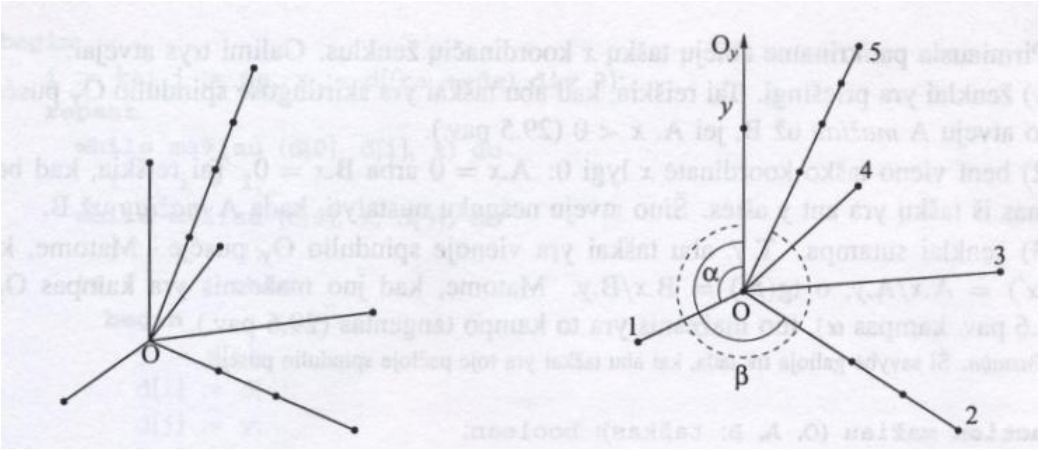
Galimas toks jungimo būdas: pasirenkame vieną iš duotų taškų kaip atskaitos tašką O ir nubrėžiame iš jo spindulius į visus kitus taškus.

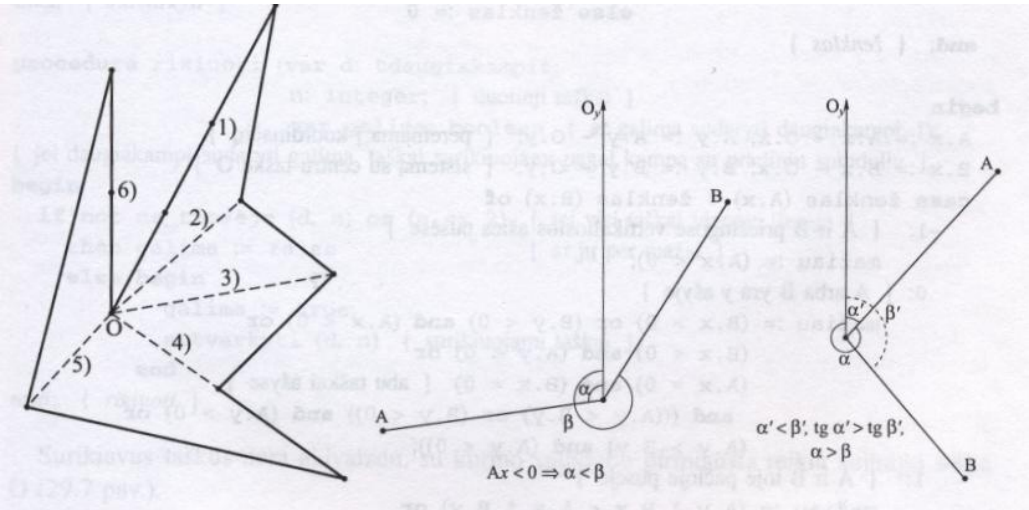


Paimame kažkokį spindulį, pavyzdžiui O_y, kaip pradinį ir surikiuojame spindulius pagal kampą (prieš laikrodžio rodyklę) sudaromą su pradiniu spinduliu. Kelis sutampančius spindulius laikykime vienu.

Tada daugiakampį jungiame taip: brėžiame atkarpą nuo O iki tolimiausio taško pirmame spindulyje, toliau brėžiame atkarpą iki artimiausio taško nuo O antrame spindulyje, toliau vėl brėžiame atkarpą iki tolimiausio taško antrame spindulyje ir taip pereiname visus spindulius iki priešpaskutinio, o paskutiniuoju spinduliu pereiname nuo tolimiausio taško iki artimiausio ir sugrįžtame į tašką O.

Tokiu būdu visada gausime daugiakampį, nes atkarpos tarp gretimų spindulių ir spinduliuose nesikerta.





Rikiuojant taškus, gali kilti keblumų su geometrija. Tarkime, kad turime tris taškus: A, B ir O. Susitarkime, kad taškas O yra koordinačių pradžios taškas. Panagrinėkime, kaip nustatyti, kuris iš spindulių OA ar OB sudaro didesnį kampą su spinduliu O_y. O jei kampas yra vienodas – kuris iš taškų yra arčiau koordinačių pradžios taško. Bus patogu, jei tarp dviejų taškų įvesime sąryšį *mažiau.* Laikysime, kad A *mažiau* už B, jei kampas $O_yA < O_yB$ arba, jei $O_yA = O_yB$ ir $|OA| < |OB|$.

Pirmiausia patikriname abiejų taškų *x* koordinačių ženklus. Galimi trys atvejai:

– ženklai yra priešingi. Tai reiškia, kad abu taškai yra skirtingose spindulio O_y pusėse. Šiuo atveju A *mažiau* už B, jei $A.x < 0$.

– bent vieno taško koordinatė *x* lygi 0: $A.x = 0$ arba $B.x = 0$. Tai reiškia, kad bent vienas iš taškų yra ant *y* ašies. Šiuo atveju nesunku nustatyti, kada A *mažiau* už B.

– ženklai sutampa. T. y. abu taškai yra vienoje spindulio O_y pusėje. Matome, kad $tg(\alpha') = A.x/A.y$, o $tg(\beta') = B.x/B.y$. Matome, kad juo mažesnis yra kampas O_yA (26.6 pav. kampas α), tuo mažesnis yra to kampo tangentas.

Pastaba. Ši savybė galioja tik tada, kai abu taškai yra toje pačioje spindulio pusėje.

```
function mažiau (O, A, B: taškas): boolean;
{ palyginamos taškų A ir B padėtys taško O atžvilgiu: }
{ Oy yra vertikalus spindulys į viršų nuo taško O, k(M) – kampas MOy  matuojamas }
{ prieš laikrodžio rodyklę nuo Oy, čia M – taškas }
{ Taigi, P(A) < P(B), kai k(A) < k(B) arba kai k(A) = k(B) ir |OA| < |OB| }

  function ženklas (x: integer): integer;
  begin
    if x > 0 then ženklas := 1
             else if x < 0 then ženklas := -1
                           else ženklas := 0
  end;  { ženklas }

begin
  A.x := A.x - O.x; A.y := A.y - O.y;  { pereinama į koordinačių }
  B.x := B.x - O.x; B.y := B.y - O.y;  { sistemą su centru taške O }
  case ženklas (A.x) * ženklas (B.x) of
    -1:  { A ir B priešingose vertikaliosios ašies pusėse }
         mažiau := (A.x < 0);
     0:  { A arba B yra ant y ašies }
         mažiau := (B.x > 0) or (B.y < 0) and (A.x < 0) or
                   (B.x < 0) and (A.y < 0) or
                 A.x = 0) and (B.x = 0)  { abu taškai ant ašių }
                 and (((A.y<B.y) or (B.y<0)) and (A.y>0) or
                 (A.y > B.y) and (A.y < 0));
     1:  { A ir B toje pačioje pusėje }
         mažiau := (A.y*B.x<A.x*B.y) or
                   (A.y*B.x=A.x*B.y) and (abs(A.x)<abs(B.x))
   end
end;   { mažiau }
```

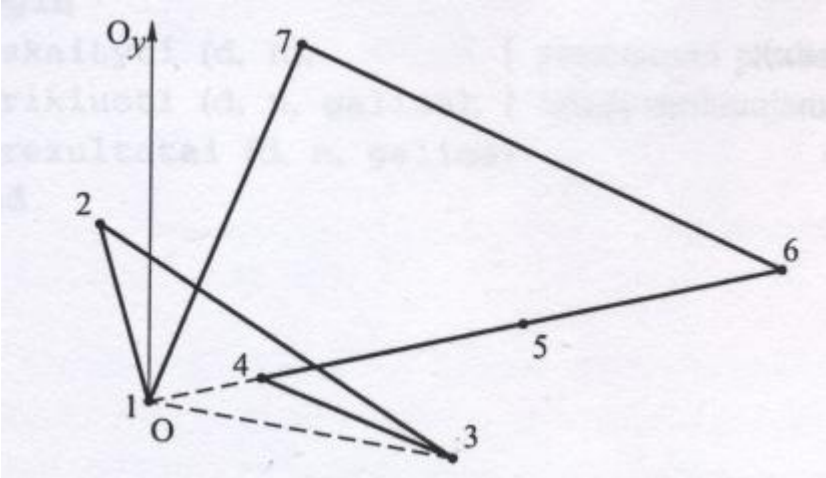
```
procedure sutvarkyti (var d: tdaugiakampis; n: integer);
{ daugiakampio taškai surikiuojami pagal sąryšį mažiau }
{ naudojant Greitojo rikiavimo metodą}

   procedure rūšiuok (ka, de: integer);
     var i, j: integer;
         x, y: taškas;
   begin
      i := ka; j := de; x := d[(ka + de) div 2];
      repeat
        while mažiau (d[0], d[i], x) do
          i := i + 1;
        while mažiau (d[0], x, d[j]) do
          j := j - 1;
        if i <= j then
          begin
            y := d[i];
            d[i] := d[j];
            d[j] := y;
            i := i + 1;
            j := j - 1;
          end
      until i > j;
      if ka < j then rūšiuok (ka, j);
      if i < de then rūšiuok (i, de)
   end; { rūšiuok }

begin
  rūšiuok (1, n)
end; { sutvarkyti }

procedure rikiuoti (var d: tdaugiakampis;
                    n: integer; { duotieji taškai }
                    var galima: boolean
                    { ar galima sudaryti daugiakampį }) ;
{ jei daugiakampį sudaryti galima, taškai surikiuojami pagal kampą su }
{ pradiniu spinduliu }
begin
  if not ne_tiesėje (d,n) or (n<=2) { jei visi taškai vienoje tiesėje }
  then galima := false              { ar jų per mažai }
  else begin
         galima := true;
         sutvarkyti (d, n) { surikiuojami taškai }
       end;
end; { rikiuoti }
```

Surikiavus taškus nėra akivaizdu, su kuriuo spinduliu pirmiausia reikia sujungti tašką O



Taigi, prieš spausdinant būtina patikrinti, ar nėra dviejų spindulių kampas tarp kurių viršija 180°. Jei yra, iš taško O būtina eiti į tašką, esantį viename šių spindulių.

```
function kitas (x, n: integer): integer;
begin
  if x = n
     then kitas := 1
     else kitas := x + 1
end; { kitas }

function atgal (x, n: integer): integer;
begin
  if x = 1
     then atgal := n
     else atgal := x - 1
end; { atgal }

function mažiau_už_180 (O, A, B: taškas): boolean;
begin
  A.x := A.x - O.x; A.y := A.y - O.y;
  B.x := B.x - O.x; B.y := B.y - O.y;
  mažiau_už_180 := (B.y*A.x > B.x*A.y) or
                   (B.y*A.x=B.x*A.y) and (A.x*B.x>-A.y*B.y)
end;
```

```
procedure rasti_pradžią (d: tdaugiakampis; n: integer;
                         var t: integer);
{ randa tašką, kurį reikia jungti su tašku O }
begin
  t := 1;
  while mažiau_už_180(d[0],d[t],d[kitas(t,n)])and(t<=n)do
    t := t + 1;
  t := kitas (t, n)
end;

procedure rezultatai (d: tdaugiakampis; n: integer;
                      galima: boolean);
{ išveda taškus tokia tvarka, kad jungiant juos paeiliui būtų }
{ gaunamas daugiakampis }
  var f: text;
  procedure rašyk_tašką (i: integer);
  begin
    writeln (f, d[i].x, ' ', d[i].y)
  end;
  var pirm, pask, i, a: integer;
begin { rezultatai }
  assign (f, 'DAUG.REZ');
  rewrite (f);
  if not galima
  then writeln (f,'NEGALIMA')
  else
   begin
     rasti_pradžią (d, n, a); { a – pirmasis pirmojo spindulio taškas }
     { ieškosime tolimiausio paskutinio spindulio taško }
     pask := atgal (a, n);
     { ieškosime pirmo to paties spindulio taško }
      pirm := pask;
      while ant_tiesės (d[0], d[pirm], d[pask]) do
        pirm := atgal (pirm, n);
      pirm := kitas (pirm, n);
      rašyk_tašką (0);
      { taškas O yra tarp taškų a ir pirm }
      while a <> pirm do
        begin
          rašyk_tašką (a);
          a := kitas (a, n)
        end;
      for i := pask downto pirm do
        rašyk_tašką (i);
   end;
  close (f)
end; { rezultatai }
```

Kadangi gan sudėtingas šio uždavinio išvedimas, pateikiame ir pagrindinę programos dalį. Trūksta tik duomenų įvedimo procedūros, kurią rasite kartu su testais. Duomenų įvedimas šiam uždaviniui – labai nesudėtingas.

```
program daugiakampis;

  const MAXN = 100;  { didžiausias galimas taškų skaičius }
  type taškas = record
                  x, y: integer
                end;
       tdaugiakampis = array [0..MAXN] of taškas;

  var n: integer;
      d: tdaugiakampis;
      galima: boolean;
begin
  skaityti (d, n);   { perskaitomi pradiniai duomenys }
  rikiuoti (d, n, galima);  { taškai surikiuojami pagal kampą }
                            { sudaromą su pradiniu spinduliu }
  rezultatai (d, n, galima)
end.
```