2.11 Pagrindiniai grafų teorijos skaičiai

Pagrindiniai grafų teorijos skaičiai yra:

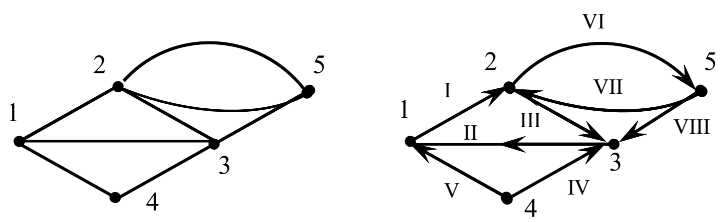
1. **ciklomatinis** skaičius,
2. ***chromatinis*** (spalvinis) skaičius,
3. ***nepriklausomumo*** (vidinio stabilumo) skaičius,
4. ***dominavimo*** (išorinio stabilumo) skaičius.

Ciklomatinis skaičius

Nagrinėsime neorientuotuosius grafus, kurie gali būti ir multigrafai. Grafo $G=(V,U)$ ***ciklomatinis skaičius*** $\nu(G)$ apibrėžiamas formule

$$\nu(G)=m-n+p,$$

čia *m* – briaunų skaičius, *n* – viršūnių skaičius, o *p* – jungiųjų komponenčių skaičius. Pavyzdžiui, grafui, pavaizduotam 2.11.1 pav., ciklomatinis skaičius $\nu(G)=8-5+1=4$.



a) b)

2.11.1 pav. **Grafo ciklomatinis skaičius**

Kiekvienam grafo ciklui galima priskirti *m*-matį vektorių tokiu būdu:

1. sunumeruokime briaunas ir kiekvienai briaunai laisvai suteikime orientaciją (žr. 2.11.1 b) pav.);
2. jei apeinant ciklą, *k*-toji briauna (u,v) rodyklės kryptimi praeinama *s* kartų, o prieš rodyklę – *t* kartų, tai šiam ciklui atitinkančio vektoriaus *k*-oji komponentė lygi $s-t$.

Pavyzdžiui, 2.11.1 b) pav. ciklą $\mu_1=(1,2,5,3,1)$ atitiks vektorius

$$c_1=(\overline{\begin{array}{cccccccc}\scriptstyle 1 & \scriptstyle 2 & \scriptstyle 3 & \scriptstyle 4 & \scriptstyle 5 & \scriptstyle 6 & \scriptstyle 7 & \scriptstyle 8\end{array}} \atop \begin{array}{cccccccc}1 & 1 & 0 & 0 & 0 & 1 & 0 & 1\end{array}),$$

o ciklą $\mu_2=(1,3,4,1)$ – vektorius

$$c_2=(\overline{\begin{array}{cccccccc}\scriptstyle 1 & \scriptstyle 2 & \scriptstyle 3 & \scriptstyle 4 & \scriptstyle 5 & \scriptstyle 6 & \scriptstyle 7 & \scriptstyle 8\end{array}} \atop \begin{array}{cccccccc}0 & -1 & 0 & -1 & 1 & 0 & 0 & 0\end{array}).$$

Nepriklausomiems vektoriams atitinkantys ciklai yra nepriklausomi.

Teorema. Multigrafo *G* ciklomatinis skaičius lygus didžiausiam nepriklausomų ciklų skaičiui.

Išvados.

- Grafas *G* neturi ciklų tada ir tik tada, kai $\nu(G)=0$.
- Grafas *G* turi vienintelį ciklą tada ir tiktai tada, kai $\nu(G)=1$.

Pavyzdžiui, 2.11.1 a) grafui nepriklausomų ciklų aibė (bazė) yra ciklai: $\mu_1=(1,2,3,1)$, $\mu_2=(1,3,4,1)$, $\mu_3=(2,5,2)$, $\mu_4=(2,5,3,2)$.

Aišku, kad nepriklausomų ciklų rinkinys yra nevienintelis. Pavyzdžiui, tam pačiam 2.11.1 a) grafui nepriklausomų ciklų bazę sudaro ir ciklai $\mu_1=(1,2,5,3,1)$, $\mu_2=(2,5,3,2)$, $\mu_3=(2,5,2)$, $\mu_4=(1,3,4,1)$. Šie ciklai yra tiesiškai nepriklausomi, nes kiekvienas iš jų turi bent po vieną unikalią briauną, nepriklausančią likusiems ciklams. Ciklas μ_1 turi unikalią briauną $(1,2)$, μ_2 – $(3,2)$, μ_3 – $(5,2)$ ir μ_4 – $(3,4)$ arba $(4,1)$.

Nepriklausomų ciklų apskaičiavimo uždavinys. Duotas jungusis neorientuotasis (*n,m*)-grafas (ne multigrafas) *G*=(*V,*U). Rasti nepriklausomų ciklų bazę.

Įveskime ***atvirkštinės briaunos*** sąvoką. Tarkime, kad $H=(V,U_H)$ yra grafo *G* dalinis grafas, neturintis ciklų. Toks dalinis grafas vadinamas dengiančiu medžiu. (Apie medžius bus kalbama 2.13 paragrafe). Dengiantis medis turi $(n-1)$ briauną ir neturi ciklų.

Apibrėžimas. Grafo *G* briauna (u,v) yra ***atvirkštinė briauna* (*styga*)**, jei ji nepriklauso dengiančio medžio *H* briaunų aibei.

Kadangi $|U_H|=n-1$, tai atvirkštinių briaunų skaičius yra lygus $m-n+1$, t.y. kiekviena atvirkštinė briauna $e=(u,v)$ drauge su dengiančio medžio briaunomis, priklausančiomis grandinei, jungiančiai viršūnę *u* su viršūne *v*, sudaro vieną nepriklausomą grafo *G* ciklą. ***Šį ciklą žymėsime*** C_e.

Vadinasi, kiekvienas grafo *G* dengiantis medis apibrėš nepriklausomų ciklų aibę. Be to, neizomorfinių medžių nepriklausomų ciklų aibės bus skirtingos.

Pastaba. Taip apibrėžus nepriklausomus ciklus, visus kitus grafo ciklus galima išreikšti per nepriklausomų ciklų simetrinį skirtumą (sumą moduliu 2).

Apibrėžimas. Grafo $G=(V,U)$ briaunų aibė *C* vadinama ***pseudociklu***, jei grafo (V,C) visų viršūnių laipsniai yra lyginiai.

Tuščia aibė ir bet koks grafo ciklas yra pseudociklas.

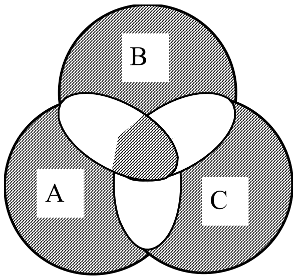
Priminsime aibių ***simetrinio skirtumo*** operaciją:

$$A\oplus B=(A\cup B)\setminus(A\cap B).$$

Šią operaciją taikysime didesniam aibių $A_1,A_2,\dots,A_k$ skaičiui:

$$\bigoplus_{i=1}^{k} A_i = A_1\oplus A_2\oplus\dots\oplus A_k.$$

Nesunku įsitikinti, kad aibių $A_1,A_2,\dots,A_k$ simetrinį skirtumą, nepriklausomai nuo skliaustų, nurodančių veiksmų atlikimo tvarką, sudėjimo sudarys tik tie elementai, kurie priklauso nelyginiam skaičiui poaibių A_i (žr. 2.11.2 pav.)



2.11.2 pav. **Aibių *A, B* ir *C* simetrinis skirtumas *A* ⊕ *B* ⊕ *C***

Lema. Bet kokio skaičiaus pseudociklų simetrinis skirtumas yra pseudociklas.

Įrodymas. Aišku, kad pakanka panagrinėti dviejų pseudociklų C_1 ir C_2 atvejį.

Simboliais $S_1(v)$, $S_2(v)$ ir $S(v)$ pažymėkime viršūnei *v* incidentiškų briaunų skaičių atitinkamai pseudocikluose C_1, C_2ir $C = C_1 \oplus C_2$. Aišku, kad aibių $S_1(v)$ ir $S_2(v)$ elementų skaičius yra lyginis. Tada

$|S(v)| = |S_1(v) \oplus S_2(v)| = |S_1(v)| + |S_2(v)| - 2|S_1(v) \cap S_2(v)|$

taip pat yra lyginis skaičius, ir *C* yra pseudociklas.

Teorema. Tarkime, *G*=(*V,U*) – jungusis neorientuotasis grafas, o (*V,T*) – jį dengiantis medis. Tada bet koks grafo *G* ciklas *C* vienareikšmiškai išreiškiamas per nepriklausomus ciklus C_e taip:

$$C = \bigoplus_{e \in C \setminus T} C_e .$$

Įrodymas. Simetrinis skirtumas $\bigoplus_{e \in C \setminus T} C_e$, remiantis lema, yra pseudociklas, susidedantis iš atvirkštinių briaunų (stygų) aibės *C* \ *T* ir kažkokių medžio *T* briaunų. Tai išplaukia iš to fakto, kad kiekviena atvirkštinė briauna $e \in C \setminus T$ priklauso tik vienam nepriklausomam ciklui C_e.

Aibė $C \oplus \bigoplus_{e \in C \setminus T} C_e$, remiantis lema, taip pat nusakys pseudociklą, kurį gali sudaryti tik medžio *T* briaunos. (Pseudociklo *C* atvirkštinės briaunos *e* priklauso ir pseudociklui $\bigoplus_{e \in C \setminus T} C_e$, todėl jos nepriklausys aibei $C \oplus \bigoplus_{e \in C \setminus T} C_e$). Kadangi medis T neturi ciklų, tai šis pseudociklas yra tuščioji aibė. Vadinasi, $C = \bigoplus_{e \in C \setminus T} C_e$. šios formulės vienareikšmiškumas išplaukia iš to, kad ciklas C_e yra vienintelis nepriklausomas ciklas, turintis atvirkštinę briauną *e*.

Iš šio nagrinėjimo išplaukia, kad grafo nepriklausomus ciklus patogu ieškoti, naudojant paieškos gilyn metodą, aptartą 2.8.2 paragrafe.

Kaip paieškos gilyn metodu formaliai nustatyti ***atvirkštinę briauną*** (***stygą***)?

Tam tikslui reikia žinoti viršūnių aplankymo eilės numerius. Įveskime masyvą $nr[1..n]$, čia $nr[i]$ yra *i*-osios viršūnės aplankymo eilės numeris.

Tarkime, kad paieškos gilyn metu iš viršūnės *u* atėjome į viršūnę *v*. Briauna (u,v) bus atvirkštinė briauna (styga) ir iššauks nepriklausomą ciklą, jei bus teisinga sąlyga: "***viršūnė v nenauja***" ***and*** "***į viršūnę u buvome atėję ne iš viršūnės v***" ***and*** "$\mathbf{nr}[\mathbf{v}] < \mathbf{nr}[\mathbf{u}]$, ***t.y. viršūnė v buvo aplankyta anksčiau nei viršūnė u***".

2.8.1.2 paragrafe aprašytam paieškos gilyn algoritme ši sąlyga bus užrašoma taip: $(\mathbf{prec}[\mathbf{v}] \neq \mathbf{0})$ ***and*** $(\mathbf{prec}[\mathbf{u}] \neq \mathbf{v})$ ***and*** $(\mathbf{nr}[\mathbf{v}] < \mathbf{nr}[\mathbf{u}])$.

Žinant šią atvirkštinės briaunos sąlygą nesunku modifikuoti 2.8.1.2 paragrafe aprašytą paieškos gilyn metodą taip, kad jis apskaičiuotų nepriklausomus cilus. Žemiau ir pateikta ši nepriklausomų ciklų apskaičiavimo procedūra.

```
const c = 500;
type
   mas = array [1..c] of integer;
procedure ciklai(v, n, m : integer; L, lst : mas);
{ Procedūra ciklai, naudodama paiešką gilyn iš viršūnės v, kai grafas nusakytas L ir
lst masyvais, apskaičiuoja grafo nepriklausomus ciklus.
   Formalūs parametrai:
   v – pradinė paieškos viršūnė,
   n – grafo viršūniu skaičius,
   m – grafo briaunų (lanku) skaičius,
   L, lst – grafą nusakantys tiesioginių nuorodų masyvai. }
var i, k, u   : integer;
   sk, j      : integer;
   t, p       : boolean;
   fst, prec    : mas;
   nr         : mas;
begin
   { Inicializacija }
   for i:=1 to n do begin
   fst [i]:= lst [i] + 1;
   prec [i] := 0;
   end;
   k := v;
   if fst [k] <= lst [k+1] then {yra nenagrinetų briaunų, incidentiškų viršūnei k}
      begin
         t := false;
         p := true;
         prec [k] := k; { k – pradinė paieškos viršūnė }
         { Nagrinėti viršūnę k }
         nr[k] := 1;
         sk := 1; { Aplankytų viršūnių skaičius }
      end
            else {viršūnė k yra arba izoliuota viršūnė, arba neturi išeinančių lankų
            (orientuotojo grafo atveju); paieškos pabaiga }
            t := true;
while not t do { paieška nebaigta}
   begin
   { Pirmyn }
      while p do
         begin
            u := L [fst [k]];
            if prec [u] = 0 then {viršūnė u nauja}
               begin
                  { Nagrinėti viršūnę u }
                  sk := sk + 1;
                  nr [u] := sk;
                  prec [u] := k;{ į viršūnę u atėjome iš viršūnės k }
                     if fst [u] <= lst [u + 1] then { viršūnė u neišsemta }
                        k:=u
```

```
                        else {viršūnė u išsemta} p:=false;
            end
                else
            begin
              p := false; { viršūnė u nenauja}
              if (prec [k] <> u) and (nr [k] > nr [u]) then
              { Briauna (k, u) yra atvirkštinė briauna. Spausdinti ciklą }
                begin
                  writeln;
                  write (u : 3);
                  write (k : 3);
                  j := k;
                  while prec [j] <> u do
                    begin
                      j := prec [j];
                      write (j : 3);
                    end;
                  write(u : 3);
                  writeln;
                end;
            end;
        end;
  { Atgal }
    while not p and not t do
      begin
        {Imama nauja, dar nenagrinėta briauna, incidentiška viršūnei k}
        fst [k] := fst [k] + 1;
        if fst [k] <= lst[k + 1] then { tokia briauna egzistuoja }
          p := true
              else { viršūnė k išsemta }
                if prec [k] = k then { pradinė paieškos viršūnė išsemta; paieškos
                pabaiga } t := true
                  else { grįžome i viršūnę, iš kurios buvome atėję į viršūnę k } k
                  := prec [k];
      end;
  end;
end;
```

Aptarkime dar vieną nepriklausomų ciklų apskaičiavimo, naudojant rekursiją, procedūrą.

Šią procedūrą patogu organizuoti naudojant steką (žr. 2.8.1.3 paragrafą).

Tarkime, kad grafas $G=(V,U)$ nusakytas gretimumo struktūra, t.y. $N(v)$ –aibė viršūnių, gretimų viršūnei *v*.

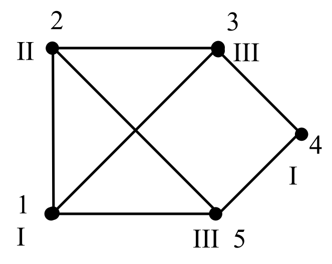
```
const c = 500;
type
  mas = array [0..c] of integer;
  matr = array [1..2, 1..c] of integer;
procedure ncikl (v : integer);
{ Grafo G jungiosios komponentės, turinčios savyje viršūnę v, nepriklausomų ciklų
apskaičiavimas.
Kintamieji num, top, stec, nr, L, lst – globalieji }
var u, i, top1 : integer;
```

```
begin
top := top +1;
stek [top] := v;
num := num + 1;
nr [v] := num;
for i := lst [v] + 1 to lst [v + 1] do
    begin
        u: = L [i];
        if nr [u] = 0 then ncikl (u)
            else
                if (nr [v] > nr [u]) and (u <> stek [top – 1]) then
                { Briauna (v,u) – atvirkštinė briauna. Įsiminti ciklą, einantį per viršūnes: u,
                stek [top], stek [top – 1], ..., stek [c], čia stek [c] = u. }
                    begin
                        writeln;
                        write (u : 3);
                        top1 := top;
                        while stek [top1] <> u do
                            begin
                                write (stek[top1]:3);
                                top1:=top1-1;
                            end;
                        write (u : 3);
                        writeln;
                    end;
    end;
top := top – 1; { Išsemta viršūnė v šalinama iš steko. }
end; { ncikl }
```

Chromatinis skaičius

(*n,m*)-grafo *G*=(*V,U*) ***chromatinis skaičius*** yra mažiausias skaičius spalvų, kurioms grafo viršūnes galima nudažyti taip, kad bet kokios dvi gretimos viršūnės būtų nudažytos skirtinga spalva. Šį skaičių žymėsime $\gamma(G)$.

Pavyzdžiui, 2.11.3 pav. grafui chromatinis skaičius yra 3. Aišku, kad šio grafo negalima nudažyti su mažesniu spalvų skaičiumi, nes jis turi ilgio 3 ciklų.



2.11.3 pav. **Grafo chromatinis skaičius**

Aišku, kad K_n grafo chromatinis skaičius lygus *n*, o bet koks dvidalis grafas $G=(A,B,U)$ yra bichromatusis: aibės *A* viršūnės dažomos pirmąja spalva, o aibės *B* viršūnės – antrąja spalva.

Perfrazavus Kionigo teoremą, galime teigti, kad grafas yra bichromatusis tada ir tiktai tada, kai jis neturi nelyginio ilgio ciklų.

Chromatinio skaičiaus apskaičiavimo uždavinys yra diskrečiojo optimizavimo uždavinys. Formaliai jį galima užrašyti taip.

$\min\ z = p$, čia p – spalvų skaičius,

esant apribojimams:

a) kiekviena viršūnė turi būti dažoma viena spalva,
b) bet kokios gretimos viršūnės turi būti nudažytos skirtinga spalva.

Šiems apribojimams užrašyti įvesime kintamuosius

$$x_{ij} = \begin{cases} 1, \text{jei } i\text{-toji viršūnė dažoma } j\text{-ąją spalva,} \\ 0, \text{priešingu atveju,} \end{cases}$$

$i = \overline{1,n}$, $j = \overline{1,p}$.

Tada a) apribojmas bus užrašomas taip:

$$\sum_{j=1}^{p} x_{ij} = 1, \quad i = \overline{1,n},$$

o b) apribojimas –

$$\sum_{i=1}^{n} x_{ij} \cdot a_{ik} \le 1, \quad j = \overline{1,p}, i = \overline{1,m},$$

čia a_{ik} – incidencijų matricos (žr. 2.7 paragrafą) elementas.

Pavyzdžiui, 2.11.3 pav. grafui optimalus šio uždavinio sprendinys yra:

$$X = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\!\!\begin{array}{c} \begin{array}{ccc} \mathrm{I} & \mathrm{II} & \mathrm{III} \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array} .$$

Kaip buvo minėta aukščiau, šis uždavinys yra NP pilnasis [GD82] ir neturi efektyvių sprendimo algoritmų. todėl jis sprendžiamas naudojant įvairias euristikas.

Dažniausiai naudojami grafų dažymo algoritmai remiasi tokiomis euristikomis:

- pirma spalva, po to viršūnė,
- pirma viršūnė, po to spalva.

Detaliau aptarkime šiuos euristinius algoritmus.

Euristinis algoritmas, paremtas principu "pirma spalva, o po to viršūnė"

Šios euristikos idėja yra labai paprasta. Pagal kokį nors požymį sudaroma grafo viršūnių seka. Po to imama nauja spalva ir šia spalva iš eilės dažomos, jeigu galima, sekos viršūnės. Taip elgiamės iki nudažome visas grafo viršūnes.

Tuo būdu šis algoritmas susideda iš dviejų etapų.

1. *Grafo viršūnės išrikiuojamos jų laipsnių mažėjimo tvarka* $v_1, v_2, \dots, v_n$, čia $v_i \in V$ ir $d(v_{i+1}) \le d(v_i)$, $i = \overline{1, n-1}$.

2. $p := 0$; {*spalvų skaičius*}

while "yra nenudažytų viršūnių" do

begin

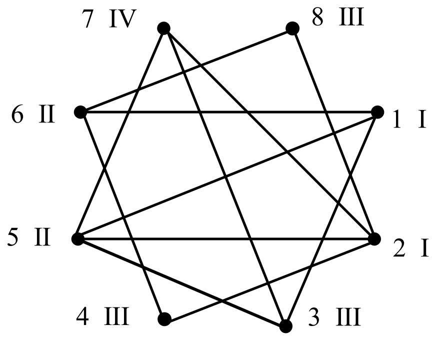
$p := p + 1$;

Pradedant pirmąja nenudažyta sekos viršūne, **p** spalva nuosekliai viena po kitos dažomos, jei galima, nenudažytos sekos viršūnės.

end;

Pavyzdys. Panagrinėkime grafo, pavaizduoto 2.11.4 pav., dažymą, laikantis principo “pirma spalva, o po to viršūnė”.

2.11.4 pav. pavaizduotam grafui seka viršūnių, išrikiuotų jų laipsnių mažėjimo tvarka, yra: 2, 5, 1, 3, 6, 7, 4, 8.



2.11.4 pav. **Grafo dažymas laikantis principo “pirma spalva, o po to viršūnė”**

Pirmąja spalva dažome 2-ąją viršūnę, po to pirmąja spalva galime dažyti tik 1-ąją viršūnę. Gausime:

2,	5,	1,	3,	6,	7,	4,	8
I		I					.

Imame antrąją spalvą ir dažome pirmąją nenudažytą sekos viršūnę – 5-ąją, po to antrąja spalva galime dažyti 6-ąją viršūnę. Gausime:

2,	5,	1,	3,	6,	7,	4,	8
I	II	I		II			.

Trečiąja spalva pirmiausia dažome 3-ąją viršūnę, po to – 4-ąją viršūnę ir, galiausiai, 8-ąją viršūnę. Gausime:

2,	5,	1,	3,	6,	7,	4,	8
I	II	I	III	II		III	III.

Kadangi dar yra nenudažytų viršūnių, tai imame IV-ąją spalvą ir dažome 7-ąją viršūnę, t.y. gausime:

2,	5,	1,	3,	6,	7,	4,	8
I	II	I	III	II	IV	III	III.

Gautas sprendinys nėra optimalus. Šį grafą galima nudažyti trimis spalvomis (žr. euristiką “pirma viršūnė, o po to spalva”).

Žemiau pateikta grafo dažymo, naudojant euristiką “pirma spalva, o po to viršūnė” procedūra.

```
const c = 500;
type
   mas  = array [1..c] of integer;
   matr = array [1..2, 1..c] of integer;
procedure grfdaz1 (n, m : integer; b : matr; var p: integer; var d : mas);
{ Procedūra grafdaz1 dažo grafo viršūnes, remdamasi euristika “pirma spalva, o po to
viršūnė”.
```

```
Formalūs parametrai:
   n – grafo viršūnių skaičius,
   m – grafo briaunų (lankų) skaičius,
   b – grafo briaunų matrica ,
   p – spalvų skaičius,
   d [1..n] – viršūnių spalvų masyvas;
   jei d [i] = k, tai reiškia, kad i-oji viršūnė dažoma k-ąja spalva. }
var i, k, z, sv, u,j ,x : integer;
   t : boolean;
   L, lst, v, s : mas;
begin
BLlst(n,m,b,L,lst);
{ Pradinių reikšmių suteikimas darbo masyvams ir kintamiesiems }
for i := 1 to n do
   begin
      v [i] := i; { Masyve v iš eilės surašomi viršūnių numeriai }
      s [i] := lst [i + 1] – lst [i]; {s [i] – i-tosios viršnės laipsnis }
      d[i] := 0;
   end;
{ Viršūnes masyve v išrikiuojame jų laipsnių mažėjimo tvarka }
for k := 1 to n – 1 do
for i: = 1 to n – k do
if s [i] < s [i + 1] then { Keičiame vietomis s [i] su s [i + 1] ir v [i] su v [i + 1] }
   begin
      z := s [i]; s [i] := s [i + 1]; s [i + 1] := z;
      z := v [i]; v [i] := v [i + 1]; v [i + 1] := z;
   end;
p := 0; { p – spalvų skaičius }
sv := 0; {sv - nudažytų viršūnių skaičius }
while sv < n do
   begin
      p := p + 1;
      for i:=1 to n do
         begin
            u := v [i];
            if d [u] = 0 then {Viršūnė u – nenudažyta.
            Ar ją galima dažyti p-ąja spalva? }
               begin
                  { Ar viršūnių, gretimų viršūnei u, tarpe yra viršūnė,nudažyta p-ąja
                  spalva? }
                  j := lst [u] + 1; t := false;
                  { Jei t = false, tai viršūnę u galima dažyti p-ąja spalva }
                  while (j <= lst [u + 1]) and not t do
                     begin
                        x := l [j];
                        if d[x] = p then t:=true
                                    else j:=j+1;
                     end;
                  if not t then
                     begin
                        d [u] := p;
```

```
                sv := sv + 1;
              end;
          end;
      end;
   end;
end;
```

Euristinis algoritmas, paremtas principu "pirma viršūnė, o po to spalva"

Šios euristikos idėja yra ta, kad pirmiausia pagal kažkokį kriterijų išrenkama grafo viršūnė, o po to ji dažoma, jei galima, viena iš anksčiau panaudotų spalvų; jei išrinktos viršūnės negalima nudažyti nei viena iš anksčiau panaudotų spalvų, tai ji dažoma nauja spalva.

Viršūnės išrinkimo kriterijus

Apibrėžimas. Viršūnės *h* laipsnis – tai skaičius spalvų, kuriomis šios viršūnės negalima dažyti, t.y. šiai viršūnei negalimų spalvų skaičius.

Apibrėžimas. Viršūnės *k* laipsnis – tai šiai viršūnei gretimų nenudažytų viršūnių skaičius.

Pirmiausia pasirinksime viršūnę, kurios *h* laipsnis yra didžiausias.

Jei yra kelios viršūnės su tuo pačiu didžiausiu *h* laipsniu, tai iš jų išsirinksime viršūnę, kurios *k* laipsnis yra didžiausias.

Jei ir šiuo atveju bus daugiau nei viena viršūnė, imsime viršūnę, kurios numeris mažiausias.

Tuo būdu algoritmas aprašomas taip.

begin
p:=0;
while "yra nenudažytų viršūnių" do
begin

1) *pagal aukščiau aprašytą kriterijų parenkama nenudažyta viršūnė v;*
2) *jei galima, viršūnę v dažome viena iš anksčiau panaudotų spalvų (paprastai, pirmąja iš galimų), t.y. viena iš aibės {1,2,...,p} galimų spalvų;*
3) *jei viršūnės v negalima nudažyti nei viena iš anksčiau panaudotų spalvų, tai p:=p+1 , ir viršūnę v dažome p-ąją spalva;*
4) *visoms nenudažytoms viršūnėms perskaičiuojame h ir k laipsnius;*

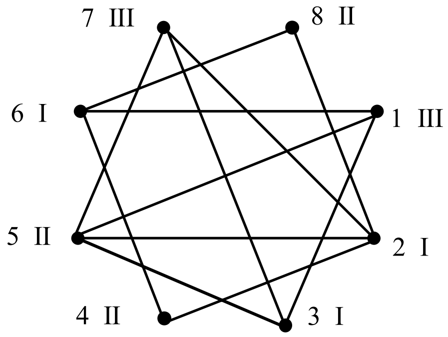
end;
end;

Pavyzdys. Panagrinėkime, kaip bus dažomos 2.11.4 pav. grafas. Skaičiavimai parodyti 2.11.1 lentelėje, o 2.11.5 pav. pavaizduotas nudažytas grafas.

2.11.1 lentelė. **Grafo dažymas**

NR	*h* – laipsnis	*k* – laipsnis	spalva	*
1	~~0~~ ~~1~~ 2	~~3~~ ~~2~~ 1	III	5
2	0	4	I	1
3	~~0~~ ~~1~~ 2	~~3~~ ~~2~~ 1	I	4
4	~~0~~ 1	~~2~~ 1	II	6
5	~~0~~ 1	4 3	II	2
6	~~0~~ ~~1~~ 2	~~3~~ ~~2~~ 1	I	7
7	~~0~~ ~~1~~ 2	~~3~~ ~~2~~ 1	III	3
8	~~0~~ 1	~~2~~ ~~1~~ 0	II	8

čia * – viršūnės išrinkimo eilės numeris.



2.11.5 pav. **Grafo dažymas laikantis principo "pirma viršūnė, o po to spalva"**

Žemiau pateikta grafo dažymo, naudojant euristiką "pirma viršūnė, o po to spalva" procedūra.

```
const c = 500;
type
   mas = array [1..c] of integer;
   matr = array [1..2, 1..c] of integer;
procedure grfdaz2(n, m : integer; b : matr; var p : integer; var d : mas);
{ Procedūra grafdaz2 dažo grafo viršūnes, remdamasi euristika "pirma viršūnė, o po
to spalva".
Formalūs parametrai:
   n – grafo viršūnių skaičius,
   m – grafo briaunų (lankų) skaičius,
   b – grafo briaunų matrica ,
   p – spalvų skaičius,
   d [1..n] – viršūnių spalvų masyvas;
   jei d [i] = k, tai reiškia, kad i-oji viršūnė dažoma k-ąja spalva. }
var i, sv, max, maxka, v, u, j, x, sp : integer;
   t,st : boolean;
   L,lst,h,ka : mas;
begin
BLlst (n, m, b, L, lst);
{ Pradinių reikšmių suteikimas darbo masyvams ir kintamiesiems }
for i := 1 to n do
   begin
      ka [i] := lst [i + 1] – lst [i]; { ka [i] – i-tosios viršūnės laipsnis }
      d[i] := 0;
      h[i] := 0;
   end;
p := 0; { p – spalvų skaičius }
sv := 0; {sv – nudažytų viršūnių skaičius }
while sv < n do
   begin
      { Viršūnės išrinkimas }
      max := –1;
      for i := 1 to n do
      if (d [i] = 0) and (max < h [i]) then max := h [i];
      maxka := –1;
      for i:=1 to n do
```

```
    if (d [i] = 0) and (h[i] = max) and (maxka < ka [i]) then
        begin
            maxka := ka [i];
            v:=i;
        end;
    { Kuria spalva dažyti viršūnę v ? }
    sv := sv + 1; { Dažome viršūnę v }
    t := false; { t = false, jei viršūnei v spalva neparinkta }
    i := 1; { Bandome viršūnę v dažyti spalva i }
    while (i <= p) and (not t) do
    begin
        { Tikriname, ar viršūnei v gretimų viršūnių tarpe yra viršūnė, kuri nudažyta i-
        ąja spalva }
        j:=lst [v] + 1;
        st := false; { Jei st = false, tai reiškia, kad viršūnei v gretimų viršūnių tarpe
        spalvos i nėra }
        while (j <= lst [v + 1]) and (not st) do
            begin
                u := L [j];
                sp:=d[u];
                if i = sp then st := true
                           else j:=j+1;
            end;
        if st then i := i + 1 { Bandome naują spalvą }
        else t:=true; { Viršūnė v gali būti dažoma i-ąja spalva }
    end;
if t then d [v] := i { Viršūnę v dažome i-ąja spalva }
    else begin
        p := p + 1; { Įvedame naują spalvą }
        d [v] := p; { Viršūnę v dažome nauja spalva }
    end;
{ h ir ka laipsnių perskaičiavimas nenudažytoms viršūnėms, gretimoms viršūnei v }
for i:=lst[v]+1 to lst[v+1] do
    begin
        u := L [i];
        if d [u] = 0 { Viršūnė u , gretima viršūnei v, – nenudažyta } then
            begin
                ka[u] := ka [u] – 1;{ Viršūnei u gretimų nenudažytų viršūnių skaičius
                sumažėjo }
                j := lst [u] + 1;
                t := false; { Nei viena iš viršūnei u gretimų viršūnių nenudažyta d [v]
                spalva }
                while (j <= lst [u + 1]) and ( not t) do
                    begin
                        x:=L[j];
                        if (x <> v) and (d [x] = d [v]) then t := true
                                else j := j + 1;
                    end;
                if not t { Viršūnei u gretimų nudažytų viršūnių x tarpe nėra spalvos,
                lygios d[v] }
```

```
          then h [u] := h [u] + 1;
        end;
    end;
  end;
end;
```

Uždavinys, analogiškas grafo viršūnių dažymo uždaviniui, yra briaunų dažymo uždavinys.

Mažiausias skaičius spalvų, kuriomis grafo briaunas galima nudažyti taip, kad bet kurios dvi gretimos briaunos būtų nudažytos skirtinga spalva, vadinamas grafo ***chromatine klase***.

Aišku, kad (n,m)-grafo *G* chromatinė klasė yra lygi šiam grafui atitinkančio briauninio grafo (žr. 2.5 paragrafą) chromatiniam skaičiui.

Vadinasi, aukščiau aptarti chromatinio skaičiaus apskaičiavimo algoritmai tinka ir grafo chromatinei klasei apskaičiuoti. Tik šiuo atveju reikia nagrinėti grafui *G* atitinkanti briauninį grafą.

Reikia pabrėžti, kad eilė praktinių uždavinių yra formuluojami kaip grafo dažymo uždaviniai. Šiems uždaviniams būdinga tai, kad kai kurie objektai dėl vienokių ar kitokių priežasčių negali būti jungiami į grupes.

Pirmas pavyzdys.

Tarkime, kad per galimai trumpiausią laiką reikia perskaityti keletą paskaitų. Kiekvienos paskaitos trukmė yra 1 valanda ir kai kurias paskaitas skaito tas pats lektorius. Sudarykime grafą *G*, kurio viršūnės vaizduoja paskaitas, o dvi viršūnės yra gretimos, jei jų negalima skaityti vienu metu, t.y. jei jas skaito tas pats lektorius. Aišku, kad bet koks teisingas grafo nudažymas apibrėžia paskaitų skaitymo tvarkaraštį, t.y. paskaitos, atitinkančios viršūnėms, kurios nudažytos ta pačia spalva, gali būti skaitomos vienu metu. Vadinasi, grafo chromatinis skaičius bus lygus mažiausiam valandų skaičiui, kuris reikalingas perskaityti paskaitų ciklą.

Antras pavyzdys.

Duota darbų $V = \{v_1, v_2, ..., v_n\}$ aibė ir mechanizmų $S = \{s_1, s_2, ..., s_m\}$ aibė. Visų darbų trukmės yra lygios, o darbams atlikti naudojami aibės *S* mechanizmai. Aišku, kad tas pats mechanizmas vienu metu negali būti naudojamas keliems darbams atlikti. Mechanizmus reikia paskirstyti taip, kad bendras visų darbų atlikimo laikas būtų trumpiausias.

Sudarykime grafą *G*, kurio viršūnių aibė yra darbų aibė *V*. Viršūnės v_i ir v_j $(i \neq j)$ yra gretimos, jei šių darbų atlikimui reikia vieno ir to paties mechanizmo. Aišku, kad teisingai nudažius grafą, darbai, nudažyti ta pačia spalva, gali būti vykdomi tuo pačiu metu.

Chromatinio skaičiaus įverčiai

[EM90]. Kadangi chromatinio skaičiaus apskaičiavimo uždavinys yra sunkus, tai svarbu žinoti chromatinio skaičiaus įverčius.

Simboliais $\delta(G)$ ir $\Delta(G)$ atitinkamai pažymėkime grafo *G* mažiausią ir didžiausią viršūnių laipsnį, t.y. $\delta(G) = \min_{v \in V} d(v)$, o $\Delta(G) = \max_{v \in V} d(v)$. Tada jungiajam grafui *G* galima nurodyti tokius chromatinio skaičiaus įverčius.

1. Grafo *G* chromatinis skaičius tenkina nelygybę $\gamma(G) \leq 1 + \Delta(G)$.
2. ***Brukso teorema*** (1941). Jei *G* – jungusis ir nepilnasis grafas, kuriam $\Delta(G) \geq 3$, tai $\gamma(G) \leq \Delta(G)$.

3. $\gamma(G) \le \min\limits_{1\le i\le n} \max(d(i)+1, i)$ (Welsh, Powell).

4. $\gamma(G) \ge \dfrac{n}{n - \dfrac{1}{2m}\sum\limits_{i=1}^{n} d^2(i)}$ (Elphick).

5. ***Teorema*** (A.P.Eršovas, G.I.Kožuchinas, 1962).

$$-\left[-n\Big/\left[\frac{n^2-2m}{n}\right]\cdot\left(1-\left\{\frac{n^2-2m}{n}\right\}\Big/\left(1+\left[\frac{n^2-2m}{n}\right]\right)\right)\right] \le$$

$$\le \gamma(G) \le \left[\frac{3+\sqrt{9+8(m-n)}}{2}\right],$$

čia simbolis [·] žymi skaičiaus sveikąją dalį, o {·} – skaičiaus trupmeninę dalį.

Šio paragrafo pabaigoje paminėsime G.Hadvigerio 1943 m. iškeltą hipotezę.

Hadvigerio hipotezė. Bet koks jungusis *n*-chromatinis grafas gali būti sutrauktas į K_n grafą.

Ši hipotezė įrodyta, kai $n \le 5$.