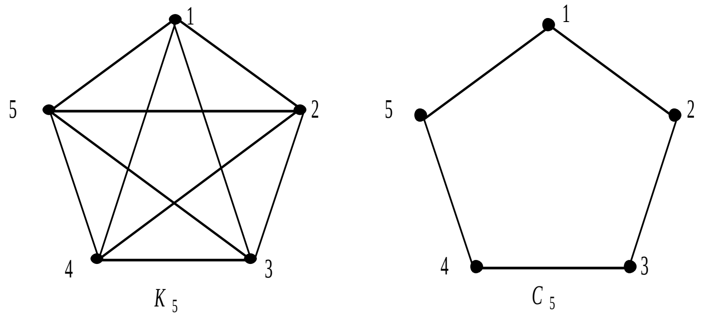
2.16. Jungumas

Jungusis grafas – tai grafas, kuriame bet kuri viršūnių pora sujungta grandine. Pavyzdžiui, (žr. 2.16.1 pav.) K_5 ir C_5 yra jungieji grafai, tačiau intuityviai jaučiame, kad K_5 yra labiau jungus nei C_5.



2.16.1 pav. Jungumas

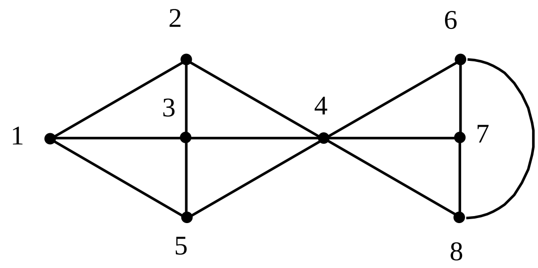
Grafo jungumo klausimas yra aktualus praktikoje. Tarkime, reikia suprojektuoti kompiuterių tinklą. Tinklą sudaro informacijos saugojimo ir apdorojimo centrai, kurie jungiami ryšio kanalais. Informacijos pasikeitimas tarp dviejų centrų vyksta arba tiesiogiai per šiuos centrus jungiantį ryšio kanalą (jei toks yra), arba per kitus centrus ir juos jungiančius ryšio kanalus. Aišku, kad tinklo grafas, kurio viršūnės vaizduoja centrus, o dvi viršūnės jungiamos briauna, jei tarp viršūnėms atitinkančių centrų yra ryšio kanalas, turi būti jungusis. Labai svarbus tinklo parametras yra jo patikimumas: tinklas turi funkcionuoti net ir tuo atveju, jei sugenda keli centrai arba ryšio kanalai. Šį parametrą kiekybiškai galima įvertinti per žemiau įvestas sąvokas.

Viršūninio jungumo skaičius. Tai mažiausias skaičius viršūnių, kurias pašalinus, grafas G tampa arba nejungiuoju grafu arba vienos viršūnės grafu. Šis skaičius žymimas $\kappa(G)$.

Briauninis jungumo skaičius. Tai mažiausias skaičius briaunų, kurias pašalinus, grafas G tampa nejungiuoju grafu. Šis skaičius žymimas $\lambda(G)$.

Pavyzdžiui, $\kappa(K_n) = n-1$, $\kappa(C_n) = 2$.

2.16.2 pav. grafui $\kappa(G) = 1$ (pašalinus 4-ąją viršūnę, grafas suyra), o $\lambda(G) = 3$ (pašalinus briaunas (4, 6), (4, 7), (4, 8) grafas suyra).

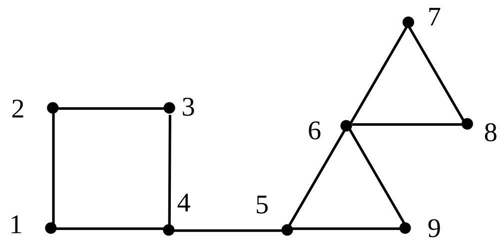


2.16.2 pav. Grafo jungumo kiekybiniai įverčiai

Grafo G viršūnė v vadinama ***sąlyčio tašku***, jei $G - v$ turi daugiau jungiųjų komponenčių nei grafas G.

Grafo G briauna (x, y) vadinama ***tiltu***, jei, ją pašalimus, gautasis grafas turi daugiau jungiųjų komponenčių nei grafas G.

Pavyzdžiui, 2.16.3 pav grafas turi tris sąlyčio taškus (viršūnės 4, 5, 6) ir vieną tiltą (briauna (4, 5)).



2.16.3 pav. Grafo sąlyčio taškai ir tiltai

Grįžtant prie paragrafo pradžioje pateikto pavyzdžio, nesunku pastebėti, kad grafo viršūninio jungumo ir briauninio jungumo skaičius parodo tinklo atsparumą centrų ir ryšio kanalų gedimams, o sąlyčio taškai bei tiltai parodo labiausiai pažeidžiamas tinklo vietas.

Teorema. Kiekvienam grafui

$$\kappa(G) \le \lambda(G) \le \delta(G),$$

čia $\delta(G) = \min_{v \in V} d(v)$, o $d(v)$ – v-osios viršūnės laipsnis.

Teorema. Beveik kiekvienam grafui[6] teisinga lygybė

$$\kappa(G) = \lambda(G).$$

Grafas vadinamas ***k-jungiuoju***, jei $\kappa(G) \geq k$ ir ***briaunomis k-jungiuoju***, jei $\lambda(G) \geq k$.

Jei $\kappa(G) = 2$, tai grafas vadinamas ***dviryšiu***.

2.16.2 pav. pavaizduotas grafas yra 1-jungusis ir briaunomis – 3-jungusis. Aišku, kad šis grafas turi pografius, kurie yra labiau jungieji nei pats grafas. Pavyzdžiui, pografis, kurį indukuoja viršūnių $\{1,2,3,4,5\}$ aibė, yra 3-jungusis. Tokių pografių apibūdinimui įvedama *k*-jungiosios komponentės sąvoka.

Grafo k-jungioji komponentė – tai maksimalus *k*-jungusis pografis. Jis dažnai vadinamas ***k-komponente***.

Pavyzdžiui, 2.16.4 pav. grafas G_1 turi dvi 2-jungiąsias komponentes (pografiai, kuriuos indukuoja viršūnių $\{1,2,3,4\}$ ir $\{3,5,6,7\}$ aibės), o grafas G_2 turi dvi 3-komponentes (pografiai, kuriuos indukuoja viršūnių $\{1,2,3,4,5,6,7\}$ ir $\{4,6,8,9,10,11,12,13\}$ aibės). Tačiau reikia pabrėžti, kad abu grafai G_1 ir G_2 yra 1-jungieji. Be to, nesunku pastebėti, kad dvi 2-komponentės turi vieną bendrą viršūnę (Grafe G_1 – tai 3-ioji viršūnė), o dvi 3-komponentės turi dvi bendras viršūnes (grafe G_2 – tai 4-oji ir 6-oji viršūnės).

Teorema. Dvi skirtingos grafo *G* *k*-komponentės turi ne daugiau nei $(k-1)$ bendrų viršūnių.

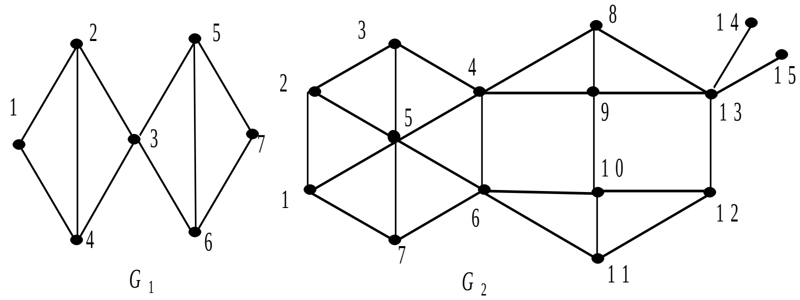
Apibrėžimas. Dvi (a,b)-grandinės vadinamos nesusikertančiomis (viršūnėmis nesusikertančiomis), jei jos neturi bendrų viršūnių, išskyrus *a* ir *b*.

Teorema (H.Witney, 1932). Grafas yra *k*-jungusis tada ir tiktai tada, kai bet kuri nesutampančių viršūnių pora (a,b) sujungta ne mažiau kaip *k* viršūnėmis nesusikertančių grandinių.

Apibrėžimas. Sakoma, kad grafo *G* viršūnių poaibis *S* skiria viršūnes *a* ir *b*, jei grafe $G-S$ viršūnės *a* ir *b* priklauso skirtingoms jungiosioms komponentėms.

---

[6] Primename sąvoką "beveik kiekvienam grafui". Simboliu $\varphi P(n)$ pažymėkime skaičių *n*-viršūninių grafų, turinčių savybę *P*, o simboliu $\varphi(n)$ – visų *n*-viršūninių grafų skaičių. Tada sakysime, kad "beveik visi grafai turi savybę *P*", jei $\lim_{n\to\infty} \frac{\varphi P(n)}{\varphi(n)} = 1$, o jei $\lim_{n\to\infty} \frac{\varphi P(n)}{\varphi(n)} = 0$, tai sakome, kad "beveik nėra grafų, turinčių savybę *P*".



2.16.4 pav. Grafo *k*-komponentės

Teorema (K.Mengeras, 1927). Mažiausias skaičius viršūnų, skiriančių dvi negretimas viršūnes a ir b, yra lygus didžiausiam skaičiui poromis nesusikertančių grandinių, jungiančių a ir b viršūnes.

Įveskime skiriančių briaunų sąvoką.

Apibrėžimas. Briaunų aibė R skiria grafo G a ir b viršūnes, jei grafe $G-R$ viršūnės a ir b priklauso skirtingoms jungiamosioms komponentėms.

Teorema. Mažiausias skaičius briaunų, skiriančių grafo G viršūnes a ir b, yra lygus didžiausiam briaunomis nesusikertančių grandinių, jungiančių a ir b viršūnes, skaičiui.

2.16.1. Dviryšiai grafai

Tiek grafų teorijoje, tiek ir praktikoje svarbią vietą užima 2-jungieji grafai, kurie vadinami dviryšiais grafais.

Kaip buvo minėta aukščiau, grafas $G=(V,U)$ yra dviryšis, jei bet kurią nesutampančių viršūnių a ir b porą jungia bent dvi viršūnėmis nesusikertančios grandinės.

Galima pateikti ir kitą dviryšio grafo apibrėžimą: grafas $G=(V,U)$ vadinamas dviryšiu, jei jis neturi sąlyčio taškų.

Apibrėžimas. Didžiausias galimas grafo G pografis, neturintis sąlyčio taškų, vadinamas ***dvigubo jungumo komponente*** arba ***bloku***.

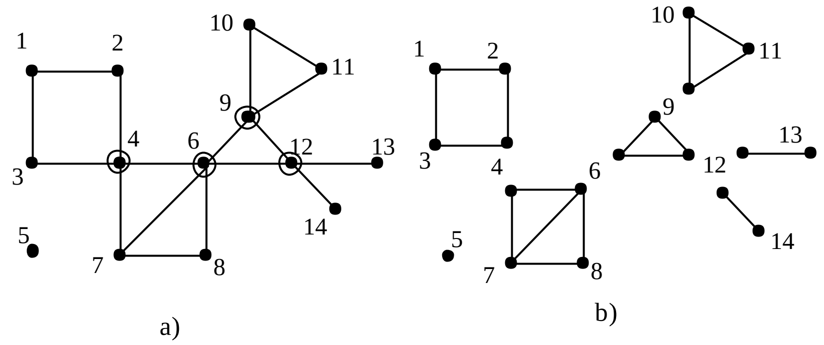
Grafo dvigubo jungumo komponenčių bei sąlyčio taškų ieškojimo uždavinys turi svarbią reikšmę. Pavyzdžiui, visi komunaliniai ir transporto tinklai negali turėti sąlyčio taškų, t.y. mažiausiai jie turi būti dviryšiai.

Dvigubo jungumo komponenčių išskyrimas grafe padeda spręsti nepriklausomų ciklų radimo uždavinį bei nustatyti, ar grafas yra plokštusis.

Aptarsime grafo $G=(V,U)$ dvigubo jungumo komponenčių (dviryšių komponenčių, blokų) radimo uždavinį.

Sąlyčio taško savybė. Neorientuotojo jungiojo grafo *G* viršūnė *a* yra sąlyčio tašku tada ir tiktai tada, kada egzistuoja tokios kitos dvi grafo viršūnės *x* ir *y*, kad bet kuris kelias tarp viršūnių *x* ir y eina par viršūnę *a*.

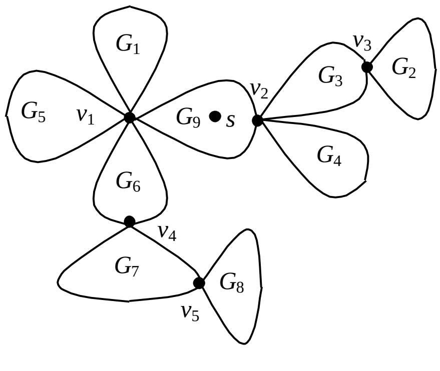
Pavyzdžiui, 2.16.5 a) pav. Pavaizduotas grafas *G*, o 2.16.5 b) pav. – to grafo dvigubo jungumo komponentės.



2.16.5 pav. Grafo dvigubo jungumo komponentės

Ši sąlyčio taško savybė įgalina panaudoti paieškos gilyn metodą, apskaičiuojant grafo *G* dvigubo jungumo komponentes – blokus.

Panagrinėkime pavyzdį (žr. 2.16.6 pav.).



2.16.6 pav. Grafo blokinė schema

2.16.6 pav. schematiškai vaizduoja jungųjį grafą, susidedantį iš 9-ių dvigubo jungumo komponenčių $G_i, i = \overline{1,9}$ bei turintį 5-is sąlyčio taškus v_1, v_2, v_3, v_4, ir v_5.

Tarkime, kad paiešką gilyn pradėjome iš 9-ojo bloko G_9 viršūnės s, ir visas aplankytas viršūnes (briaunas) talpiname į steką STEK. Pradėję paiešką iš viršūnės s, per viršūnę v_2 galime patekti į bloką G_4. Remiantis paieškos gilyn savybe, kad per viršūnę v_2 grįšime, kai bus aplankytos visos G_4 viršūnės, galime teigti, kad G_4 sudaro viršūnės (briaunos), kurios buvo aplankytos tarp įėjimo ir grįžimo per viršūnę v_2.

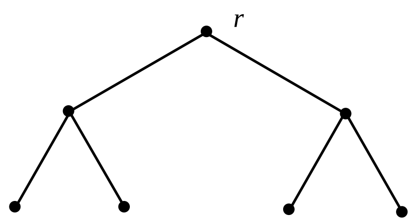
Su kitomis dvigubo jungumo komponentėmis reikalas sudėtingesnis. Iškeliavę iš viršūnės s, galime patekti į jungumo komponentę G_3, o iš G_3 per viršūnę v_3 – patekti į komponentę G_2. Tačiau ir šiuo atveju, kai aplankytos viršūnės (briaunos) saugomos steke STEK, grįžtant per viršūnę v_3, visos aplankytos bloko G_2 viršūnės bus steko viršuje (nuo steko viršaus iki viršūnės v_3). Pašalinus iš steko šias viršūnes, steko viršuje liks bloko G_3 viršūnės, kurios buvo aplankytos iki patekome į bloką G_2. Kai grįšime į viršūnę v_2, steko viršuje bus visos bloko G_3 viršūnės.

Vadinasi, jei mokėtume atpažinti ***sąlyčio taškus***, tai naudodami ***paiešką gilyn*** ir saugodami aplankytas viršūnes (briaunas) ***steke*** ta tvarka, kuria jos buvo aplankytos, galime rasti ***dvigubo jungumo komponentes***: ***viršūnės*** **(*briaunos*)*, esančios steko viršuje grįžimo per sąlyčio tašką metu, sudaro dvigubo jungumo komponentę*.**

Aptarkime formalią sąlyčio taškų radimo sąlygą.

Tarkime, kad $G = (V, U)$ – jungusis neorientuotasis grafas. Tarkime, kad T – jį dengiantis medis su šaknimi r, sukonstruotas paieškos gilyn iš viršūnės r metodu.

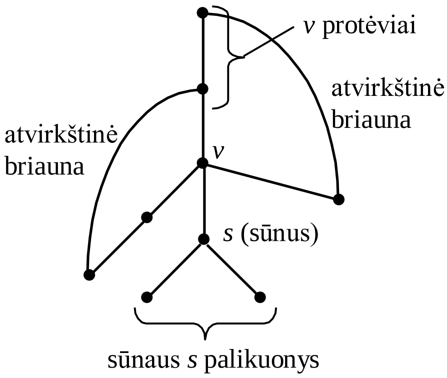
1 *lema*. Šaknis r yra sąlyčio taškas tada ir tiktai tada, kai r turi daugiau nei vieną sūnų (žr. 2.16.7 pav.).



2.16.7 pav. Dengiančio medžio T šaknis r – sąlyčio taškas

2 *lema*. Viršūnė $v \neq r$ yra grafo sąlyčio taškas tada ir tiktai tada, kai yra nors vienas viršūnės *v* sūnus *r*, kuris neturi atvirkštinių briaunų, jungiančių jį patį arba bet kurį jo palikuonį su viršūnės *v* protėviu (žr. 2.16.8 pav.).

Priminsime, kad grafo *G* atvirkštinės briaunos yra visos briaunos, nepriklausančios dengiančiojo medžio *T* briaunų aibei.



2.16.8 pav. Dengiančio medžio *T* viršūnė *v* – sąlyčio taškas

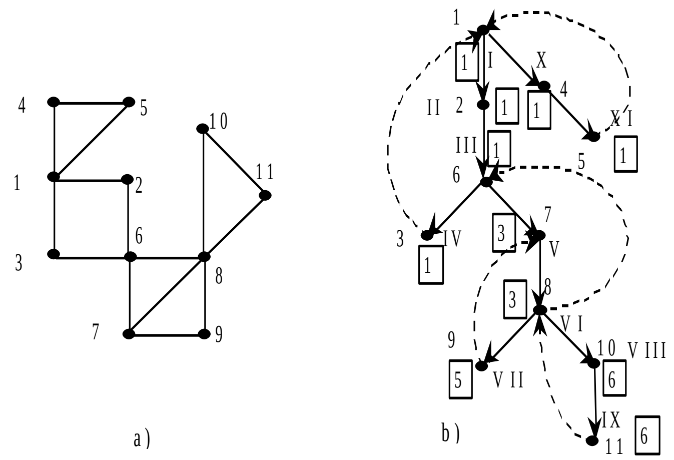
Sąlyčio taškams rasti kiekvienai grafo viršūnei *v* įvesime du parametrus: $nr[v]$ ir $low[v]$.

Parametras $nr[v]$ yra viršūnės *v* aplankymo eilės numeris, atliekant paiešką gilyn iš viršūnės *r*.

Parametras $low[v]$[7] – tai mažiausias aplankymo numeris viršūnės *x*, į kurią galime patekti iš viršūnės *v* grandine, sudaryta iš nulio arba daugiau dengiančio medžio briaunų ir nedaugiau kaip vienos atvirkštinės briaunos (žr. 2.16.9 pav.).

2.16.9 a) pav. pavaizduotas grafas *G*, o 2.16.9 b) pav. pavaizduotas tą grafą dengiantis medis, sukonstruotas paieškos gilyn iš 1-osios viršūnės metodu. Medžio briaunos pavaizduotos ištisine, o atvirkštinės briaunos – punktyrine linija. Romėniškais skaičiais pažymėti viršūnių aplankymo eilės numeriai, o parametro $low[v]$ reikšmės apibrėžtos.

[7] Parametrą $low[v]$ galima apibrėžti ir kitais žodžiais: parametras $low[v]$ – tai mažiausias aplankymo numeris viršūnės *x*, į kurią galime patekti iš viršūnės *v* arba bet kurio jos palikuonio ne daugiau kaip vienos atvirkštinės briaunos pagalba.



2.16.9 pav. Parametrų $nr[v]$ ir $low[v]$ reikšmės

Pasinaudodami parametrais $nr[v]$ ir $low[v]$ galima 2 lemos kriterijų perrašyti kaip teoremą.

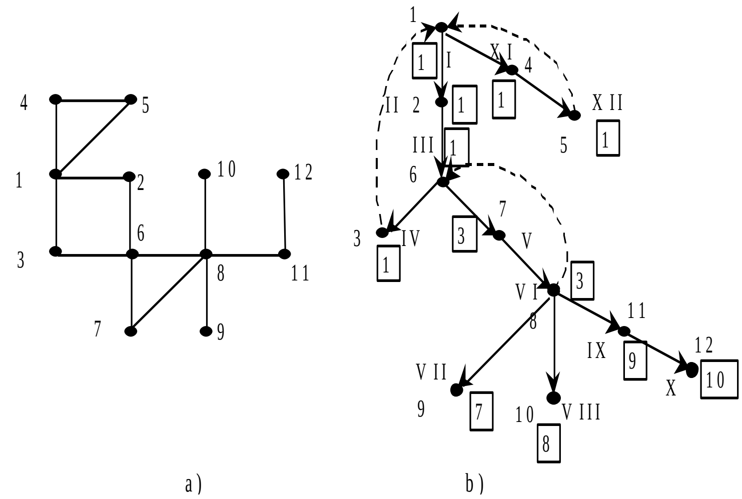
Teorema. Viršūnė $v \neq r$ yra grafo G sąlyčio taškas tada ir tiktai tada, kai v turi sūnų s, kuriam $low[s] \geq nr[v]$.

Pavyzdžiui, 2.16.9 a) pav. grafui sąlyčio taškai yra: 1-oji viršūnė (medžio šaknis, turinti du sūnus), 6-oji viršūnė (ji turi sūnų $s = 7$, kuriam $low[7] = nr[6]$) ir 8-oji viršūnė (ji turi sūnų $s = 10$, kuriam $low[10] = nr[8]$).

Bendresnis pavyzdys pateiktas 2.16.10 paveiksle. Čia 2.16.10 a) pav. duotas grafas, o 2.16.10 b) – jį dengiantis medis, sukonstruotas naudojant paiešką gilyn iš 1-osios viršūnės. Simbolių prie medžio reikšmės tos pačios, kaip ir 2.16.9 b) pav. Remiantis aukščiau pateiktu nagrinėjimu, nesunku apskaičiuoti, kad šio grafo sąlyčio taškai yra: 1-oji viršūnė (šaknis, turinti du sūnus), 6-oji viršūnė (sūnaus $s = 7$ parametras $low[7] = nr[6]$), 8-oji viršūnė (turi du sūnus $s = 9$ ir $s = 10$, kurių $low[s] > nr[8]$) ir 11-oji viršūnė (sūnaus $s = 12$ $low[12] > nr[11]$).

Parametrą $low[v]$ galima apibrėžti rekurentiškai:

$low[v]$ = min ($nr[v]$; $low[s]$, čia s – viršūnės v sūnus; $nr[w]$, čia (v, w) – atvirkštinė briauna).



2.16.10 pav. Sąlyčio taškų apskaičiavimas

Remiantis šiuo apibrėžimu, $low[v]$ reikšmė apskaičiuojama taip.

1. *Kai viršūnė v paieškos gilyn metu **yra aplankoma pirmą kartą**, tai* $low[v] := nr[v]$.
2. *Kai **nagrinėjama atvirkštinė briauna*** (v, w), *tai* $low[v] := \min(low[v], nr[w])$.
3. *Kai **grįžtame į viršūnę v, pilnai apėjus visas briaunas, incidentiškas sūnui s**, tai* $low[v] := \min(low[v], low[s])$.

Kai grįžtame į viršūnę *v*, pilnai apėjus visas briaunas, incidentiškas sūnui *s*, tai $low[s]$ reikšmė nusistovi. Jei šiuo metu $low[s] \geq nr[v]$, tai, pagal teoremą, *v* yra sąlyčio taškas.

Žemiau pateikta dvigubo jungumo komponenčių ieškojimo procedūra, kurios pagrindą sudaro paieškos gilyn algoritmas, išnagrinėtas 2.8.1.2 paragrafe, ir kuris papildytas aukščiau aptartu sąlyčio taškų sąlygos tikrinimu bei blokų formavimu.

```
const c = 500;
type
    mas = array [1..c] of integer;
    matr = array [1..2, 1..c] of integer;
procedure blok (n, m : integer; L, lst : mas; var st : mas);
```

```
{ Procedūra blok randa jungiojo grafo dvigubo jungumo komponentes (blokus)
paieškos gilyn iš pirmosios viršūnės metodu, kai grafas nusakytas L ir lst
masyvais. Rasti sąlyčio taškai ir blokų briaunų aibės atspausdinamos.
Formalūs parametrai:
    n – grafo viršūnių skaičius,
    m – grafo briaunų (lankų) skaičius,
    L, lst – grafą nusakantys tiesioginių nuorodų masyvai;
    st – sąlyčio taškų masyvas:
        jei st [i] = 1, tai viršūnė i yra sąlyčio taškas,
        jei st [i] = 0, tai viršūnė i nėra sąlyčio taškas. }
var i, k, u : integer;
    s, sk, tau : integer;
    t, p, pirma : boolean;
    fst, prec : mas;
    stek : mas;
    nr, low : mas;
begin
{ Inicializacija }
pirma := false;
v := 1;
nr [v] := 1;
low [v] := 1;
sk := 1;
tau := 0;
    for i := 1 to n do begin
        fst [i]:= lst [i] + 1;
        prec [i] := 0;
        st [i] := 0;
    end;
k := v;
if fst [k] <= lst [k + 1] then {yra nenagrinetų briaunų, incidentiskų viršūnei k }
    begin
        t := false;
        p := true;
        prec [k] := k; {k-pradinė paieškos viršūnė }
        { Nagrinėti viršūnę k }
    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;
                        low [u] := nr [u];
                        tau := tau + 1;
                        stek [tau] := k;
                        tau := tau + 1;
                        stek [tau] := u;
                        prec [u] := k;{ į viršūnę u atėjome iš viršūnės k}
                        if fs t [u] <= lst [u + 1] then {viršūnė u neišsemta}
                            k := u
                                                  else {viršūnė u išsemta}
                            p := false;
                    end
                              else
                    begin
                        if (prec [k] <> u) and (nr [k] > nr[u]) then
                        { Briauna (k, u) - atvirkštinė briauna }
                            begin
                                tau := tau + 1;
                                stek [tau] := k;
                                tau := tau + 1;
                                stek [tau] := u;
                                { low[k]=min (low [k], nr[u]) }
                                if nr [u] < low [k] then low [k] := nr [u];
                            end;
                        p := false; {viršūnė u nenauja}
            end;
    end;
{Atgal}
while not p and not t do
    begin
        {Imama nauja, dar nenagrinėta briauna, incidentiska viršūnei k}
        fst [k] := fst [k] + 1;
```

```
if fst [k] <= lst [k + 1] then {tokia briauna egzistuoja}
    begin
        if k = 1 then
        { pirmoji viršūnė turi daugiau nei vieną sūnų }
        pirma := true;
        p := true;
    end
                              else {viršūnė k išsemta}
if prec [k] = k then { pradinė paieškos viršūnė išsemta;
paieškos pabaiga }
t := true
                else { grįžome į viršūnę,
                iš kurios buvome atėję į viršūnę k }
    begin
        s := k;{ viršūnė s išsemta }
        k := prec [k];
        { low [k] = min (low [k], low [s]) }
        if low [s] < low [k] then low[k]:=low[s];
        if ((k <> 1) and (low [s] >= nr [k])) or ((k = 1) and pirma) then
        { k – sąlyčio taškas }
            begin
                st[k] := 1;
                writeln ('sąl. taškas = ', k : 3);
            end;
        if low [s] >= nr [k] then
            begin
                { Spausdinti bloką }
                writeln ('blokas');
                while (stek [tau] <> s) or (stek [tau – 1] <> k) do
                    begin
                        write (stek [tau] : 3,' - ',
                        stek [tau – 1] : 3);
                        tau := tau – 2;
                        writeln;
                    end;
                write (stek [tau] : 3,' – ',
                stek [tau – 1] : 3);
                tau := tau – 2;
```

```
            writeln;
          end;
      end;
    end;
  end;
end;
```

2.17. Srautai tinkluose ir giminingi uždaviniai

2.17.1. Pagrindinės sąvokos

Aptarsime kelis praktinių uždavinių pavyzdžius.

Pirmas pavyzdys. Tarkime, kad turime tinklą automobilinių kelių, kuriais galima nuvykti iš punkto *A* į punktą *B*. Keliai gali kirstis tarpiniuose punktuose. Kiekvienai kelio atkarpai yra žinomas maksimalus skaičius automobilių, kuriuos ši kelio atkarpa gali praleisti per laiko vienetą (kelio pralaidumas). Koks didžiausias skaičius automobilių per laiko vienetą gali nuvažiuoti iš punkto *A* į punktą *B*? Šis automobilių skaičius vadinamas nagrinėjamo kelio tinklo automobilių srauto dydžiu. Paparastai mus domina ne vien tik koks didžiausias automobilių skaičius gali nuvykti iš *A* į *B*, bet ir kiekvieno kelio ruožo srautas, t.y. koks skaičius automobilių vyksta šiuo kelio ruožu.

Aišku, kad galimas ir kitas klausimas: kiek ir kokių kelių pralaidumus reikia padidinti, kad maksimalus automobilių srautas per šį kelių tinklą padidėtų nurodytu dydžiu?

Antras pavyzdys. Tarkime, kad naftos verslovė *A* naftotiekių tinklu yra sujungta su perdirbimo įmone *B*. Taip pat žinome kiekvieno naftotiekio tarpo maksimalų pralaidumą – debitą (naftos kiekį per laiko vienetą). Kokį kiekį naftos per laiko vienetą galime perpumpuoti šiuo tinklu?

Trečias pavyzdys. Turime informacijos perdavimo tinklą. Žinomas kiekvienos ryšio linijos pralaidumas. Kokį didžiausią informacijos kiekį šiuo tinklu galima perduoti iš punkto *A* į punktą *B*?

Šie ir daugelis kitų praktinių uždavinių nagrinėjami srautų tinkluose teorijos metodais. Šiame paragrafe spręsime pagrindinį šios teorijos uždavinį – ***maksimalaus srauto radimo uždavinį***.

Apibrėžimas. Tinklas, tai pora $S = (G, C)$, kur pirmasis poros elementas $G = (V, U)$ yra bet koks orientuotasis grafas, kuriame išskirtos dvi viršūnės *s* ir *t*, iš kurių pirmoji viršūnė *s*, neturinti įeinančių lankų, vadinama ***šaltiniu***, o antroji – *t*, neturinti išeinančių lankų, vadinama ***nuotakiu***. Antrasis poros elementas C yra funkcija $U \to R$, kuri kiekvienam lankui (u,v) priskiria neneigiamą realųjį skaičių $c(u,v)$, kuris vadinamas ***lanko pralaidumu***.

Apibrėžimas. Tarkime, kad duota funkcija $f : U \to R$. Tada funkcijos f ***divergencija*** viršūnėje v vadinamas dydis $div_f(v)$, apibrėžiamas formule:

$$div_f(v) = \sum_{u:(v,u)\in U} f(v,u) - \sum_{u:(u,v)\in U} f(u,v) . \qquad (2.17.1)$$

Jei $f(u,v)$ interpretuosime kaip srautą iš viršūnės u į viršūnę v, tai skaičius $div_f(v)$ parodo "srauto kiekį", kuris išeina iš viršūnės v. Šis skaičius gali būti ***teigiamas*** (jei iš viršūnės v daugiau išeina negu įeina, t.y. viršūnė v yra papildomas šaltinis), ***neigiamas*** (jei į viršūnę v daugiau įeina negu išeina, t.y. srautas kaupiasi viršūnėje v) ir ***lygus nuliui***.

Apibrėžimas. Funkcija $f : U \to R$ vadinama ***srautu*** tinkle S, jeigu:

1. kiekvienam grafo G lankui (u,v) galioja nelygybė
$$0 \le f(u,v) \le c(u,v) , \qquad (2.17.2)$$
2. kiekvienos viršūnės v, išskyrus s ir t, divergencija lygi nuliui, t.y.
$$div_f(v) = 0, \quad v \in V \setminus \{s,t\} . \qquad (2.17.3)$$

Skaičius $W(f) = div_f(s)$ vadinamas ***srauto dydžiu***.

Pagal šį apibrėžimą nagrinėjame srautą, kuris nesikaupia ir neatsiranda nei vienoje tarpinėje grafo G viršūnėje (išskyrus viršūnes s ir t).

Toliau spręsime maksimalaus srauto radimo uždavinį: *rasime tokią funkciją* $f : U \to R$, *kuri tenkintų* (*2.17.2*) *ir* (*2.17.3*) *apribojimus ir kuriai skaičius* $W(f) = div_f(s)$ *būtų didžiausias*. Aišku, kad funkcijos f reikšmės $f(u,v)$ kiekvienam lankui (u,v) apibrėš maksimalų srautą per šį lanką.

Maksimalaus srauto apskaičiavimo uždavinys yra tiesinio programavimo uždavinys, todėl jam gali būti taikomi tiesinio programavimo uždavinių sprendimo metodai. Tačiau, įvertinant uždavinio specifiką, yra sukurti žymiai efektyvesni šio uždavinio sprendimi metodai, kuriuos toliau ir nagrinėsime.

Pjūvio apibrėžimas. Tinklo S pjūviu $P(A)$, kurį apibrėžia grafo G viršūnių tikrinis poaibis A, t.y. $A \subset V$ ir $A \neq \varnothing$ ir $A \neq V$, vadiname grafo G tokių lankų $(u,v) \in U$ aibę, kad $u \in A$, o $v \in V \setminus A$.

Kitaip tariant,

$$P(A) = U \cap (A \times (V \setminus A)) .$$

Aišku, kad bet kokiam tinklo S srautui f pjūvio srautas $P(A)$ yra lygus pjūvį sudarančių lankų sumai, t.y.

$$f(A, V \setminus A) = f(P(A)) = \sum_{e \in P(A)} f(e).$$

Lema. Jei $s \in A$, o $t \in V \setminus A$, tai bet kokiam srautui f iš s į t galioja lygybė

$$W(f) = f(A, V \setminus A) - f(V \setminus A, A) \tag{2.17.4}$$

Įrodymas. (2.17.4) lygybės teisingumas išplaukia iš (2.17.1) ir (2.17.3) formulių.

Susumuokime (2.17.3) lygtis visoms grafo viršūnėms $v \in A$. Šią sumą sudarys nariai $f(u,v)$, turintys pliuso arba minuso ženklą.

Jei abi viršūnės u ir v priklauso aibei A, tai $f(u,v)$ turės pliuso ženklą lygtyje $div_f(u) = 0$ ir minuso ženklą lygtyje $div_f(v)$, ir šio dėmens neturės nei viena lygtis $div_f(r) = 0$, kai $r \neq u$ ir $r \neq v$. Vadinasi, sumuodami šiuos dėmenis, gausime 0.

Jei $u \in A$, o $v \in V \setminus A$, tai kiekvienas narys $f(u,v)$ sumoje pasikartos vienintelį kartą su ženklu plius lygtyje $div_f(u) = 0$, ir visumoje gausime $f(A, V \setminus A)$. Analogiški dėmenys $f(u,v)$, kai $u \in V \setminus A$, o $v \in A$, duos (2.17.4) formulės narį $f(V \setminus A, A)$. Kita vertus, nagrinėjama suma lygi $div_f(s) = W(f)$, nes $div_f(v) = 0$ visiems $v \in A \setminus \{s\}$.

Išvada. $div_f(s) = -div_f(t)$.

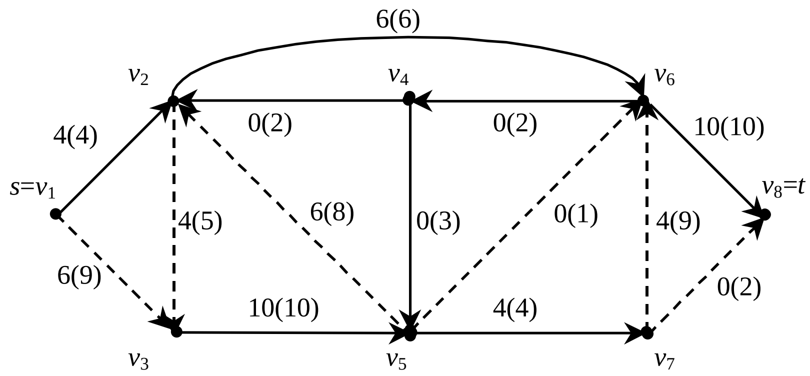
Imdami $A = V \setminus \{t\}$ iš (2.17.4) formulės gausime:

$$div_f(s) = W(f) = f(V \setminus \{t\}, \{t\}) - f(\{t\}, V \setminus \{t\}) =$$
$$= -(f(\{t\}, V \setminus \{t\}) - f(V \setminus \{t\}, \{t\})) = -div_f(t).$$

Gauta tapatybė įrodo intuityviai aiškų faktą: *nuotakio srautas yra lygus šaltinio srautui*.

Įrodyta lema sako tai, kad tinklo srautas gali būti matuojamas bet kuriame pjūvyje, kuris atskiria *s* ir *t* viršūnes.

Pavyzdys. 2.17.1 pav. pavaizduotas tinklas. Skaičiai prie lankų apibrėžia srautą *f*. Lankų pralaidumą žymi skliausteliuose parašyti skaičiai. Panagrinėkime pjūvius, kuriuos nusako viršūnių $A = \{v_1 = s, v_2, v_6, v_7\}$ ir $B = \{v_1 = s, v_2, v_3\}$ poaibiai.



2.17.1 pav. Srautas tinkle: 1-osios lemos ir didinančiosios grandinės iliustracija. Punktyru išskirta didinančioji grandinė.

Aibės A viršūnių divergencija yra:

$W(f) = div_f(v_1) = f(v_1,v_2) + f(v_1,v_3)$,

$div_f(v_2) = f(v_2,v_3) + f(v_2,v_6) - f(v_1,v_2) - f(v_5,v_2) - f(v_4,v_2) = 0$,

$div_f(v_6) = f(v_6,v_8) + f(v_6,v_4) - f(v_2,v_6) - f(v_5,v_6) - f(v_7,v_6) = 0$,

$div_f(v_7) = f(v_7,v_6) + f(v_7,v_8) - f(v_5,v_7) = 0$.

Šias lygtis sudėję, gausime:

$$-W(f) + f(v_1,v_2) + f(v_1,v_3) + f(v_2,v_3) + f(v_2,v_6) - f(v_1,v_2) -$$
$$- f(v_5,v_2) - f(v_4,v_2) + f(v_6,v_8) + f(v_6,v_4) - f(v_2,v_6) -$$
$$- f(v_5,v_6) - f(v_7,v_6) + f(v_7,v_6) + f(v_7,v_8) - f(v_5,v_7) = 0.$$

Sutraukę panašius narius, gausime:

$$W(f) = f(v_1,v_3) + f(v_2,v_3) + f(v_6,v_8) + f(v_6,v_4) + f(v_7,v_8)$$
$$- f(v_5,v_2) - f(v_4,v_2) - f(v_5,v_6) - f(v_5,v_7) =$$
$$= f(A, V \setminus A) - f(V \setminus A, A).$$

Įstatę srauto reikšmes, gausime:

$W(f) = 6+4+10+0+0-6-0-0-4 = 10$.

Pjūviui, kurį nusako poaibis $B = \{v_1, v_2, v_3\}$, analogiškai gausime:

$W(f) = f(v_2,v_6) + f(v_3,v_5) - f(v_5,v_2) - f(v_4,v_2)$.

Įstatę srauto reikšmes, gausime:

$W(f) = 6+10-6-0 = 10$.

Nesunku pastebėti, kad šie pavyzdžiai iliustruoja lemos įrodymą bei lemos išvados teisingumą.

Apibrėžimas. ***Pjūvio***, kurį nusako viršūnių poaibis A, t.y. pjūvio $P(A)$ ***pralaidumas*** yra skaičius, apibrėžiamas formule

$$c(A, V \setminus A) = \sum_{e \in P(A)} c(e) .$$

Kitaip tariant, pjūvio $P(A)$ pralaidumas yra lygus pjūvį sudarančių lankų pralaidumų sumai.

Apibrėžimas. Minimaliu pjūviu vadinsime pjūvį $P(A)\ (s \in A, t \in V \setminus A)$, kurio ***pralaidumas*** yra ***mažiausias*** tarp visų galimų pjūvių, atskiriančių s ir t viršūnes.

Vienas iš pagrindinių srautų tinkluose rezultatų yra nusakomas Fordo ir Falkersono teorema.

Teorema (Fordas (Ford L.R.) ir Falkersonas (Fulkerson D.R.) 1962). Bet kokio srauto iš viršūnės s į viršūnę t dydis neviršija minimalaus pjūvio, skiriančio šias viršūnes, pralaidumo; be to egzistuoja toks srautas, kurio dydis yra lygus tokio minimalaus pjūvio pralaidumui.

Įrodymas. Tarkime, kad $P(A)$ – minimalus pjūvis. Remiantis lema, bet kokiam srautui f gausime:

$$W(f) = f(A, V \setminus A) - f(V \setminus A, A) \le f(A, V \setminus A) =$$
$$= \sum_{e \in P(A)} f(e) \le \sum_{e \in P(A)} c(e) = c(A, V \setminus A).$$

Įrodymas, kad maksimalaus srauto dydis yra lygus minimalaus pjūvio pralaidumui, yra sudėtingesnis faktas. Šis įrodymas išplauks iš maksimalaus srauto konstravimo algoritmo, kuris nagrinėjamas kitame paragrafe.

2.17.2. Maksimalaus srauto konstravimo algoritmas

Visi žinomi maksimalaus srauto konstravimo algoritmai pagrįsti nuosekliu srauto didinimo metodu, o visų srauto didinimo metodų pagrindą sudaro ***didinančiųjų grandinių*** teorija.

Apibrėžimas. Tinklo S lankas e iš viršūnės u į viršūnę v vadinamas ***leistinuoju*** lanku srauto f atžvilgiu, jeigu

a) $e = (u, v)$ ir $f(e) < c(e)$

arba

b) $e = (v, u)$ ir $f(e) > 0$.

Jei galioja sąlyga a), tai tokį lanką vadinsime ***suderintuoju*** (soglasovannaja duga). Jei galioja sąlyga b), tai tokį lanką vadinsime ***nesuderintuoju*** (nesoglasovannaja duga).

Apibrėžimas. Ilgio l didinančioji grandinė iš viršūnės s į viršūnę t srauto f atžvilgiu yra seka, sudaryta iš besikeičiančia tvarka surašytų (poromis skirtingų) viršūnių ir lankų:

$$v_0, e_1, v_1, e_2, v_2, ..., v_{l-1}, e_l, v_l, \quad (2.17.5)$$

Čia $v_0 = s$, $v_l = t$ ir kiekvienam $1 \le i \le l$ lankas e_i srauto f atžvilgiu yra leistinasis lankas iš viršūnės v_{i-1} į viršūnę v_i.

Pavyzdžiui, 2.17.1 pav. pavaizduoto tinklo ir srauto per jį atžvilgiu, grandinė:

$$\begin{aligned} &v_1, (v_1, v_3), v_3, (v_2, v_3), v_2, (v_5, v_2), v_5, \\ &(v_5, v_6), v_6, (v_7, v_6), v_7, (v_7, v_8), v_8, \end{aligned} \quad (2.17.6)$$

yra didinančioji grandinė, kurios ilgis yra 6.

Žinodami (2.17.5) pavidalo didinančiąją grandinę, srautą f galime padidinti dydžiu

$$\delta = \min\{\Delta(e_i), 1 \le i \le l\}$$

čia

$$\Delta(e_i) = \begin{cases} c(e_i) - f(e_i), \text{ jei } e_i \text{ yra suderintasis lankas,} \\ f(e_i), \text{ jei } e_i \text{ yra nesuderintasis lankas.} \end{cases}$$

Tam tikslui (pavyzdžiui, žiūrint į 2.17.1 pav.) reikia didinančiosios grandinės kiekvieno suderintojo lanko srautą padidinti dydžiu δ, o kiekvieno nesuderintojo lanko srautą sumažinti tuo pačiu dydžiu δ, t.y.

$$f'(e_i) = \begin{cases} f(e_i) + \delta, \text{ jei } e_i \text{ yra didinančiosios grandinės} \\ \qquad \text{suderintasis lankas,} \\ f(e_i) - \delta, \text{ jei } e_i \text{ yra didinančiosios grandinės} \\ \qquad \text{nesuderintasis lankas,} \\ f(e_i), \quad \text{jei } e_i \text{ nepriklauso didinančiajai} \\ \qquad \text{grandinei.} \end{cases}$$

Aišku, kad taip apskaičiuota funkcija f' yra srautas, nes $0 \le f'(e) \le c(e)$, $e \in U$; ir $div_f(v_i) = 0$, $v_i \in V \setminus \{s, t\}$.

Iš tikro, jei v_i nepriklauso didinančiajai grandinei, tai $div_f(v_i) = div_{f'}(v_i) = 0$.

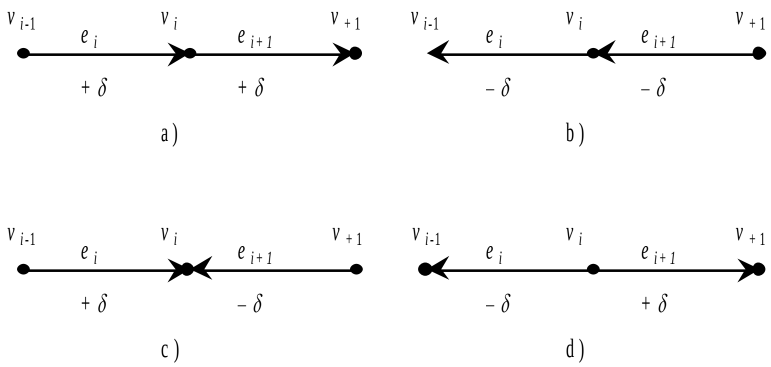
Tarkime, kad v_i priklauso didinančiajai grandinei. Tada (žr. 2.17.2 pav.):

1) jei e_i ir e_{i+1} yra suderintieji lankai, tai srautas, įtekantis į viršūnę v_i ir ištekantis iš jos padidėja tuo pačiu dydžiu δ; tuo būdu $div_{f'}(v_i) = 0$ (žr. 2.17.2 a) pav.),

2) jei e_i ir e_{i+1} yra nesuderintieji lankai, tai srautas, įtekantis ir ištekantis iš viršūnės v_i sumažėja tuo pačiu dydžiu δ ; vadinasi $div_{f'}(v_i) = 0$ (žr. 2.17.2 b) pav.),

3) jei e_i yra suderintasis lankas, o e_{i+1} – nesuderintasis lankas, tai srautas įeinančiu į viršūnę v_i lanku e_i padidėja dydžiu δ, o įeinančiu lanku e_{i+1} sumažėja tuo pačiu dydžiu; vadinasi $div_{f'}(v_i) = 0$ (žr. 2.17.2 c) pav.),

4) jei e_i yra nesuderintasis lankas, o e_{i+1} – suderintasis lankas, tai išeinančiu iš viršūnės v_i lanku e_i srautas sumažėja dydžiu δ, o išeinančiu iš viršūnės lanku e_{i+1} padidėja tuo pačiu dydžiu; tuo būdu $div_{f'}(v_i) = 0$ (žr. 2.17.2 d) pav.).



2.17.2 pav. Didinančiosios grandinės lankų, incidentiškų viršūnei v_i, srauto perskaičiavimas

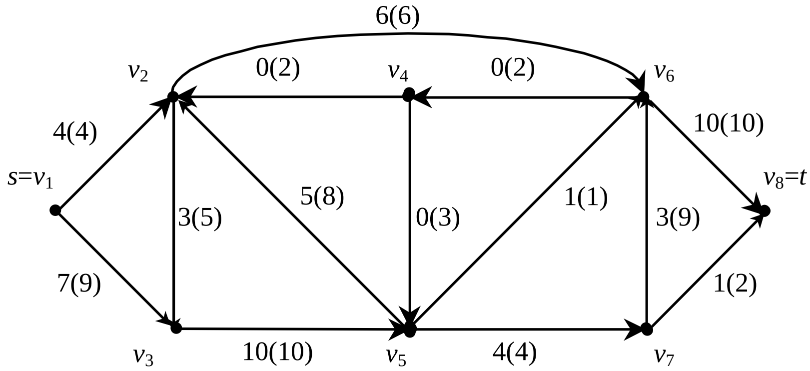
Srauto f' dydis padidėja dėmeniu δ.:

$$W(f') = div_{f'}(s) = div_f(s) + \delta = W(f) + \delta \ .$$

Pavyzdys. Panagrinėkime tinklą su srautu *f*, kuris pavaizduotas 2.17.1 pav. (2.17.6) seka yra šio tinklo ir srauto per jį atžvilgiu didinančioji grandinė. Šiai grandinei

$$\delta = \min\{\Delta(e_i) \,|\, 1 \le i \le 6\} = 1.$$

Vadinasi, 2. 17.1 pav. pavaizduoto tinklo srautą galima padidinti dydžiu 1, t.y. nuo 10 iki 11. 3.17.3 pav. pavaizduotas tas pats 2.17.1 pav. tinklas ir perskaičiuotas srautas.



2.17.3 pav. Tinklo, pavaizduoto 2.17.1 pav., srautas po modifikacijos

Įrodysime teoremą, kuri pagrindžia maksimalaus srauto apskaičiavimo, naudojant didinačiasias grandines, metodą.

Teorema. Žemiau pateiktos trys sąlygos yra ekvivalenčios:

1) srautas iš s į t maksimalus,
2) neegzistuoja didinančios grandinės srauto f atžvilgiu,
3) egzistuoja pjūvis, kurį nusako toks viršūnių poaibis $A\,(A \subset V)$, skiriantis viršūnes s ir t ($s \in A$, $t \in V \setminus A$), kad $W(f) = c(A, V \setminus A)$.

Įrodymas. 1) $\Rightarrow$ 2). Jei srautas maksimalus, tai aišku, kad neegzistuoja didinančiosios grandinės šio srauto atžvilgiu, nes, priešingu atveju, esant didinančiai grandinei, srautą būtų galima padidinti.

2) $\Rightarrow$ 3). Tarkime, kad srautui f neegzistuoja didinančiosios grandinės. Apibrėžkime viršūnių aibę $A\,(A \subseteq V)$, kurią sudaro viršūnės tokios grandinės

$$v_0, e_1, v_1, e_2, v_2, ..., v_{k-1}, e_k, v_k,$$

kad $k \ge 0$, $v_0 = s$, $v_k = v$ ir e_i – leistinasis lankas iš viršūnės v_{i-1} į viršūnę v_i, $1 \le i \le k$. Aišku, kad $s \in A$, o $t \notin A$, nes šiuo atveju egzistuotų didinančioji grandinė srauto f atžvilgiu. Panagrinėkime pjūvį $P(A)$. Iš aibės A ir suderintojo lanko apibrėžimo išplaukia, kad kiekvienam pjūvio lankui turi galioti lygybė $f(e) = c(e)$. Vadinasi, $f(A, V \setminus A) = c(A, V \setminus A)$.

Analogiškai, remiantis aibės A ir nesuderintojo lanko apibrėžimu, gausime, kad $f(V \setminus A, A) = 0$. Remiantis aukščiau įrodyta lema, gausime

$$W(f) = f(A, V \setminus A) - f(V \setminus A, A) = c(A, V \setminus A).$$

3 $\Rightarrow$ 1) išplaukia iš to fakto, kad srauto dydis neviršija $c(A, V \setminus A)$.

Pavyzdys. Nesunku patikrinti, kad srautas 2.17.3 pav. yra maksimalus. “Prisotintas” pjūvis $P(A)$, atsirandantis teoremos įrodyme, yra $A = \{v_1, v_2, v_3, v_5\}$.

Remiantis pateiktais teoriniais samprotavimais, galima sudaryti paprastą maksimalaus srauto apskaičiavimo algoritmą: pradėdami nuo bet kokio, pavyzdžiui, nulinio srauto ($f(e) = 0$, $e \in U$), ieškosime didinančiosios grandinės ir, jei tokia grandinė egzistuoja, srautą didinsime; priešingu atveju srautas *f* yra maksimalus.

Tačiau čia iškyla klausimas, ar algoritmo žingsnių skaičius yra baigtinis. Fordas ir Falkersonas[8] davė neigiamą atsakymą. 1962 m. jie pateikė pavyzdį tokio tinklo, kuriame galima taip parinkinėti didinančiasias grandines, kad procesas niekada nepasibaigtų. Be to, srauto dydis visą laiką bus ketvirtadaliu mažesnis už maksimalaus srauto dydį.

Tačiau 1972 m. Edmondsas (Edmonds J.) ir Karpas (Karp R.M.)[9] įrodė, kad, jei kiekviename žingsnyje srautą didinsime trumpiausios didinančiosios grandinės atžvilgiu, tai maksimalus srautas bus sukonstruotas, panaudojant ne daugiau kaip $m \cdot n / 2$ grandinių (čia $n = |V|$, $m = |U|$). Rasti trumpiausią didinančiąją grandinę patogu, naudojant paiešką platyn (žr. 2.9 paragrafą). Įvertinant tai, kad trumpiausios grandinės konstravimo algoritmo sudėtingumas yra $O(m+n)$, galime įvertinti viso maksimalaus srauto apskaičiavimo algoritmo sudėtingumą, – $O(mn(m+n))$.

Čia nenagrinėsime šio algoritmo detalių, kadangi toliau panagrinėsime efektyvesnį 1970 m. Dinico (Dinic E.A.)[10] pasiūlytą maksimalaus srauto apskaičiavimo algoritmą, kurio sudėtingumas yra $O(n^3)$.

Dinico metodas susideda iš fazių. Jo efektyvumas pagrįstas tuo, kad, nepriklausomai nuo tinklo lankų pralaidumo, fazių skaičius niekada neviršija $(n-1)$, čia *n* – tinklo viršūnių skaičius [Lip88].

---

[8] Ford L.R., Fulkerson D.R. Flows in networks. Princeton Univ. Press, Princeton, N.J. 1962. Yra vertimas į rusų kalbą. Ford L.R., Falkerson D.R. Potoki v setiach., Moskva: Mir, 1966.

[9] Edmonds J., Karp R.M. Theoretical improvements in algorithmic efficiency for network flow problems. J.ACM, 1972, 19, p.p. 248-264.

[10] Dinic E.A. Algoritm rešenija zadači o maksimalnom potoke v seti so stepennoj ocenkoj. Dokl. Akademii nauk SSSR, serija Mat. Fiz., 1970, 194, 4, s. 754-757.

Panagrinėkime k-osios $(k \le n-1)$ ***fazės veiksmus***. Tarkime, kad fazės pradžioje turime tinklą G ir jame apibrėžtą srautą f. Tokį tinklą žymėsime simboliu G_f.

Remdamiesi tinklu G_f, sudarome pagalbinį tinklą, kuris neturi kontūrų (ciklų) ir kurio struktūra vaizduoja visas trumpiausias didinančiasias tinklo G_f grandines. Pažymėkime šį tinklą simboliu S_f. Tinklas S_f konstruojamas paieškos platyn grafe G_f metodu, ir į tinklą S_f įtraukiami tinklo G_f srauto f atžvilgiu leistinieji lankai. Tuo būdu į tinklą S_f įeina šaltinis s, nuotakis t ir grafo G_f leistinieji lankai (u,v), čia viršūnė u nutolusi nuo šaltinio s atstumu d, o viršūnė v – atstumu d+1, $0 \le d \le l$, o l yra tinklo S_f ilgis, t.y. atstumas nuo s iki t grafe G_f. Šio lanko pralaidumą žymėsime $c_f(u,v)$ ir apibrėšime formule

$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v), \text{ jei } (u,v) \text{ yra suderintasis leistinasis } G_f \text{ lankas}, \\ f(v,u), \text{ jei } (u,v) \text{ yra nesuderintasis leistinasis } G_f \text{ lankas}, \\ c(u,v) - f(u,v) + f(v,u), \text{ jei lankas } (u,v) \text{ vienu metu ir} \\ \text{suderintasis, ir nesuderintasis leistinasis } G_f \text{ lankas}. \end{cases}$$

Sudarius pagalbinį tinklą S_f, toliau jame apskaičiuojamas ***pseudomaksimalus srautas***.

Apibrėžimas. Ilgio l tinklo S_f pseudomaksimalus srautas – tai toks tinklo S_f srautas f^*, prie kurio tinkle S_f neegzistuoja nei viena l ilgio didinančioji grandinė; kitaip tariant, bet kokiam tinklo S_f iš viršūnės s į viršūnę t keliui

$$v_0, v_1, ..., v_l \ (v_0 = s, v_l = t)$$

egzistuoja toks lankas $(v_i, v_{i+1}),\ 0 \le i < l$, kad $f^*(v_i, v_{i+1}) = c_f(v_i, v_{i+1})$.

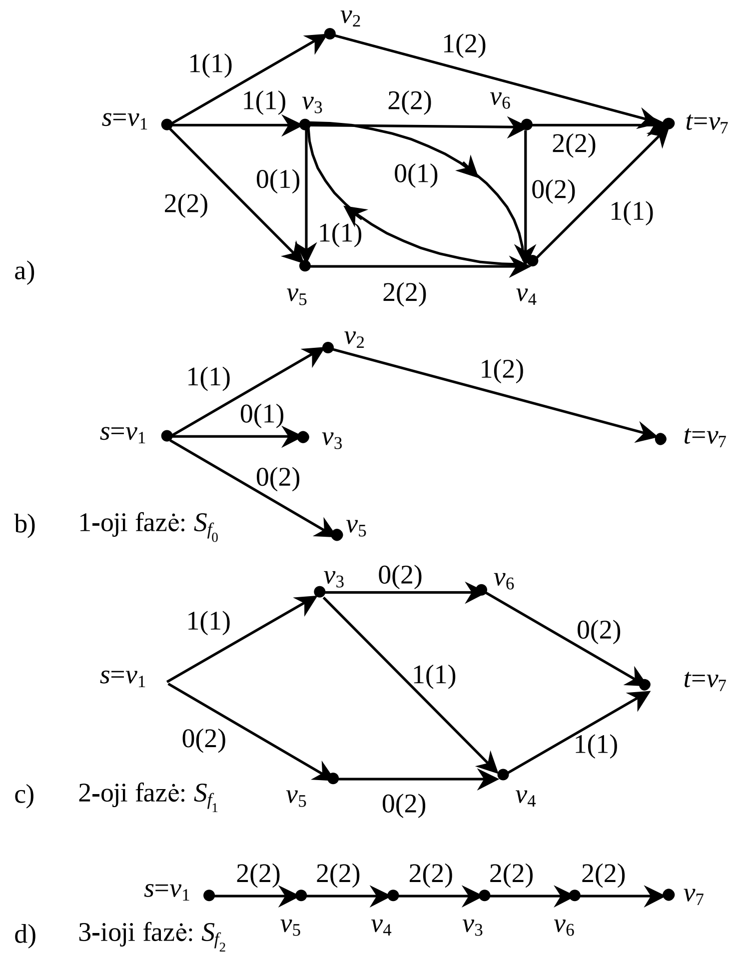
Po to pseudomaksimalus srautas "perkeliamas" į pagrindinį tinklą G_f : srautas $f^*(u,v)$ sumuojamas su G_f srautu $f(u,v)$; jei ši suma iššaukia lanko (u,v) "persipildymą", t.y. suma viršija $c(u,v)$, tai šis srauto "perteklius" kompensuojamas sumažinant srautą $f(v,u)$. Tuo būdu viršūnės u divergencija $div_f(u)$ padidėja dydžiu $f^*(u,v)$, o viršūnės v divergencija $div_f(v)$ tuo pačiu dydžiu sumažėja. Nesunku įsitikinti, kad tokių visų lankų srautų modifikacija apibrėžia naują tinklo G_f srautą f', kurio dydis $W(f') = W(f) + W(f^*)$. Šiuo veiksmu ir baigiama k-oji fazė.

Maksimalaus srauto konstravimas baigiamas, kai iš tinklo G_f nebegalima sukonstruoti pagalbinio tinklo S_f.

Kaip bus matyti iš toliau pateiktų tinklo S_f bei jo pseudomaksimalaus srauto apskaičiavimo algoritmų, vienos fazės sudėtingumas yra $O(n^2)$.

Kadangi, kaip parodyta literatūroje [Lip88], fazių skaičius neviršija $(n-1)$, tai Dinico metodo sudėtingumas yra $O(n^3)$.

Pavyzdys. Panagrinėkime tinklą, pavaizduotą 2.17.4 a) paveiksle. Prie lanko parašytas skaičius reiškia lanko maksimalų srautą, o skaičius skliausteliuose rodo lanko pralaidumą.



2.17.4 pav. Maksimalaus srauto ieškojimas Dinico metodu

Pradėję nuo tinklo nulinio srauto f_0, pirmiausia apskaičiuojame pagalbinį be kontūrų tinklą S_{f_0} (žr. 2.17.4 b) pav.). Po to rastąjį pseudomaksimalų srautą perkeliame į tinklą G_f, gaudami srautą f_1, t.y. tinklą G_{f_1}.

Toliau konstruojame pagalbinį tinklą S_{f_1} (žr. 2.17.4 c) pav.), ir rastąjį pseudomaksimalų srautą perkeliame į tinklą G_{f_1}. Gauname srautą f_2 (tinklą G_{f_2}).

Paskutinėje fazėje konstruojame pagalbinį tinklą S_{f_2} (žr. 2.17.4 d) pav.). Kai šio tinklo pseudomaksimalų srautą perkeliame į G_{f_2}, gauname srautą *f*, pavaizduotą 2.17.4 a) pav., kuris yra maksimalus tinklo *G* srautas, kadangi $W(f) = 4 = c(\{s\}, V \setminus \{s\})$ (žr. 2.17.1 paragrafo Fordo ir Falkersono teoremą).

2.17.4 c) pav. gerai iliustruoja tą faktą, kad pseudomaksomalus srautas nebūtinai yra maksimalus srautas. Iš tikro, šiame paveikslėlyje pavaizduotą pseaudomaksimalų srautą galime padidinti panaudoję ilgio 5 didinančiąją grandinę:

$$v_1, (v_1, v_5), v_5, (v_5, v_4), v_4, (v_3, v_4), v_3, (v_3, v_6), v_6, (v_6, v_7), v_7 .$$

Taip pat pažymėsime, kad tinklo S_{f_2} (žr. 2.17.4 d) pav.) lankas (v_4, v_3) atsirado iš pagrindinio tinklo *G* dviejų priešpriešais nukreiptų lankų.

Aptarkime pagalbinio tinklo S_f ir jo pseudomaksimalaus srauto apskaičiavimo algoritmus.

Pagalbinio tinklo S_f apaskaičiavimo algoritmas

Tarkime, kad pradinio tinklo grafas nusakytas gretimumo struktūromis $N[v]$ ir $PN[v]$, $v \in V$.

$N[v]$ – tai aibė viršūnių, į kurias iš viršūnės *v* išeina lankai.

$PN[v]$ – tai aibė viršūnių, iš kurių į viršūnę *v* įeina lankai.

Lankų pralaidumą žymėsime masyvu $c[u, v]$, $u, v \in V$, o faktinąjį srautą *f* – masyvu $f[u, v]$, $u, v \in V$. Tokia pralaidumo bei srauto vaizdavimo struktūra atminties požiūriu nėra racionali, tačiau prie šios struktūros algoritmas tampa vaizdesnis.

Kintamieji, vaizduojantys pagalbinį tinklą S_f, yra analogiški pradinio tinklo kintamiesiems ir jie žymimi tais pačiais vardais, pridedant prie jų pradinę raidę *S*. Tuo būdu, *SV* žymi tinklo S_f viršūnių aibę; $SN[v]$ ir $SPN[v]$, $v \in SV$, žymi S_f gretimumo struktūrą; *Sc* ir *Sf* atitinkamai žymi S_f lankų pralaidumus ir srautą. Masyvo *d* elementas $d[v]$, $v \in SV$, žymi tinklo S_f viršūnės *v* nuotolį nuo šaltinio *s*.

Kaip buvo minėta aukščiau, tinklas S_f konstruojamas paieškos platyn iš viršūnės *s* tinkle G_f metodu. Paieškos platyn procese aplankytos viršūnės *v* talpinamos į eilę. Kiekviena iš eilės paimta viršūnė *u* nagrinėjama du kartus: pirmą kartą – ieškomi iš jos išeinantys leistinieji suderintieji lankai (u,v); antrą kartą – ieškomi leistinieji nesuderintieji lankai (v,u).

Jei tinkle G_f vienu metu lankas (u,v) yra suderintasis ir nesuderintasis leistinasis lankas, t.y. (u,v) – leistinasis suderintasis lankas, o (v,u) – leistinasis nesuderintasis lankas, tai į tinklą S_f įtraukto lanko (u,v) pralaidumas yra lygus šių lankų pralaidumų srauto *f* atžvilgiu sumai.

```
procedure psa;
{Pagalbinio be kontūrų tinklo Sf konstravimas.
Kintamieji V, N, PN, c, f, SV, SN, SPN, Sc, Sf, d, s, t – globalieji}
begin
   for u ∈ V do {inicializacija}
      begin
         d [u] := ∞;
         SN [u] :=∅; SPN [u] := ∅;
         for v ∈ V do Sc [u, v] := 0;
         for v ∈ V do Sf [u, v] := 0; {nulinio srauto inicializacija}
      end;
   Eilė := ∅; SV := ∅; d [s] := 0;
   {Paieška platyn iš šaltinio s}
   Eilė ⇐ s;
   while Eilė ≠ ∅ do
      begin
         u ⇐ Eilė;
         SV := SV ∪ {u};
         for v ∈ N [u] do {suderintųjų lankų iš viršūnės u paieška}
         if ( d [u] < d [v] ≤ d [t] ) and (f [u, v] < c [u, v] ) then
            begin
               if d [v] = ∞ then {viršūnė v – nauja} Eilė ⇐ v;
               d [v] := d [u] + 1;
               { Laukas (u, v) įjungiamas į tinklą Sf }
               SN [u] := SN [u] ∪ {v};
```

```
            SPN [v] := SPN [v] ∪ {u};
            Sc [u, v] := c [u, v] – f [u, v];
        end;
    for v ∈ PN [u] do {nesuderintųjų lankų paieška}
    if ( d [u] < d [v] ≤ d [t] ) and (f [v, u] > 0 ) then
        begin
            if d [v] = ∞  then {viršūnė v – nauja} Eilė ⇐ v;
            d [v] := d [u] + 1;
            if Sc [u, v] = 0 then {Lankas (u, v) dar neįjungtas į tinklą.
            Lanką (u, v) įtraukti į tinklą}
                begin
                    SN [u] := SN [u] ∪ {v};
                    SPN [v] := SPN [v] ∪ {u};
                end;
            Sc [u, v] := Sc [u, v] + f [v, u];
        end;
  end; {while}
end; {psa}.
```

Tarkime, po procedūros *psa* įvykdymo $d[t] = l$. Jei $l = \infty$, tai reiškia, kad paieškos platyn metu nuotakio viršūnė *t* nebuvo pasiekta. Vadinasi, tinkle G_f nuo *s* iki *t* nėra didinančiosios grandinės. Tada, remiantis 2.17.2 paragrafo teorema, galime teigti, kad srautas *f* yra maksimalus.

Jei $l < \infty$, tai trumpiausios didinančiosios grandinės iš *s* į *t* ilgis yra *l*. Be to, visų kelių nuo *s* iki *t* ilgiai tinkle S_f yra lygūs *l*, ir jie žymi *l* ilgio didinančiasias grandines tinkle G_f. Reikia pažymėti, kad, pasiekus nuotakį *t*, $d[t]$ tampa lygus $l < \infty$, ir viršūnės *v*, kurioms $d[v] \geq l$, procedūroje *psa* nenagrinėjamos, kadangi nei viena tokia viršūnė negali priklausyti *l* ilgio keliui, jungiančiam *s* ir *t*.

Nesunku įvertinti procedūros *psa* sudėtingumą. Kiekviena viršūnė *v* į eilę talpinama ir šalinama ne daugiau kaip vieną kartą. Kiekvienas viršūnei *v* incidentiškas lankas analizuojamas vieną kartą, be to, analizės veiksmų skaičius apribotas konstanta. Vadinasi, bendras veiksmų skaičius be inicializacijos yra $O(n+m)$. Inicializacijos veiksmų skaičius yra $O(n^2) > O(n+m)$. Tuo būdu, procedūros *psa* sudėtingumas yra $O(n^2)$.

Belieka aptarti tinklo S_f pseudomaksimalaus srauto efektyvaus apskaičiavimo algoritmą. Žemiau pateiktas Malchotros, Kumaro ir Mahešvario

(Malhotra V.M., Kumar P.M., Makeshvari S.N. An $O(n^3)$ algorithm for finding maximum flows in networks. Information Processing Lett. 1978, 7, s. 277 – 278) pseudomaksimalaus srauto apskaičiavimo algoritmą, kurio sudėtingumas yra $O(n^2)$.

Tinklo S_f pseudomaksimalaus srauto apskaičiavimo procedūra maxpsa

Šio algoritmo aprašymui įveskime tinklo ***viršūnės potencialo*** sąvoką.

Apibrėžimas. Tinklo viršūnės v potencialas yra maksimalus srauto kiekis, kurį galima praleisti per viršūnę v. Viršūnės v potencialas $P(v)$ apibrėžiamas formule

$$P(v) = \min(Pin(v), Pout(v)),$$

$$Pin(v) = \begin{cases} \sum_{u:u\to v} c(u,v), & \text{jei } v \neq s, \\ \infty, & \text{jei } v = s, \end{cases}$$

$$Pout(v) = \begin{cases} \sum_{u:v\to u} c(v,u), & \text{jei } v \neq t, \\ \infty, & \text{jei } v = t. \end{cases}$$

Žemiau pateikta ***maxpsa*** procedūra pirmiausia apskaičiuoja pagalbinio tinklo S_f visų viršūnių potencialus ir nulinio potencialo viršūnes patalpina į steką STEK. Aišku, kad nulinio potencialo viršūnes drauge su joms incidentiškais lankais galima pašalinti iš tinklo S_f, ir šis pašalinimas neturės įtakos jokiam tinklo S_f srautui. Tokių viršūnių, o tikslaiu joms incidentiškų lankų pašalinimas gali paveikti kitų tinklo S_f viršūnių potencialus, ir jie gali tapti lygūs nuliui. Jei taip atsitinka, tai naujos nulinio potencialo viršūnės šalinamos iš tinklo.

Jei, pasibaigus nulinio potencialo viršūnių šalinimo procesui, tinklas S_f neturi nenulinio potencialo viršūnių, tai tinklo S_f pseudomaksimalus srautas surastas. Priešingu atveju randame mažiausio potencialo viršūnę r ir jos potencialą p. Tada iš viršūnės r į viršūnę t ir iš viršūnės s į viršūnę r konstruojame p dydžio srautą. Tuo būdu apskaičiuojamas p dydžio srautas iš šaltinio s į nuotakį t. Šis srautas naudos tik leistinuosius suderintuosius lankus.

Aptarsime, kaip konstruojamas p dydžio srautas iš viršūnės r į viršūnę t. Srauto iš viršūnės s į viršūnę r konstravimas yra analogiškas.

Kaip buvo minėta, p dydžio srautas iš viršūnės r į viršūnę t naudos tik suderintuosius leistinuosius lankus, ir jį patogiausia konstruoti naudojant paiešką platyn iš viršūnės r. Įsivaizduokime, kad viršūnėje r patalpintas krovinys $Q[r] = p$, ir mes norime "perkelti" šį krovinį į viršūnę *t*. Kiekviena paieškos platyn metu aplankyta viršūnė *v* "perkelia" krovinį $Q[v]$ į viršūnes *u*, į kurias eina lankai iš viršūnės *v*. Be to, į viršūnę *u* "perkeliamo" krovinio dalis yra lygi lanko (v,u) pralaidumui, t.y. $Sf[v,u] = Sc[v,u]$. Aišku, kad tokie lankai gali būti šalinami iš tinklo. Lankų šalinimas iššaukia viršūnių potencialų perskaičiavimą, ir, jei viršūnės *v* potencialas tampa lygus nuliui, tai viršūnė *v* talpinama į steką STEK.

Tuo būdu, krovinys $Q[r] = p$ iš viršūnės *r*, kuri nuo šaltinio s nutolusi atstumu *d*, pirmiausia bus "perkeltas" į viršūnes, nutolusias nuo šaltinio *s* atstumu $d+1$, po to, iš šių viršūnių, į viršūnes, nutolusias nuo šaltinio *s* atstumu $d+2$ ir t.t. iki visas krovinys bus "perkeltas" į viršūnę *t*. Kadangi *p* yra mažiausias viršūnės potencialas, tai toks krovinio "perkėlimas" visada yra galimas.

Kaip minėjome, srautas iš *s* į *r* konstruojamas analogiškai.

p dydžio srauto iš viršūnės *s* į *t* konstravimu baigiasi pagrindinio ciklo veiksmai, ir, kaip buvo minėta aukščiau, pagrindinis ciklas bus kartojamas, kol tinklas S_f turės nenulinio potencialo viršūnių.

```
procedure maxpsa;
{Pseudomaksimalaus srauto pagalbiniame tinkle Sf apskaičiavimas
Malchotros, Kumaro ir Mahešvario metodu.
Kintamieji SV, SN, SPN, Sc, Sf, s ir t yra globalieji.}
begin
  STEK := ∅; { Steke STEK bus saugomos nulinio potencialo viršūnės. }
  for v ∈ SV do {viršūnės v potencialo apskaičiavimas}
    begin
        Pin [v] := 0; Pout [v] := 0;
        if v = s then Pin [v] := ∞
                else
            for u∈ SPN [v] do
                Pin [v] := Pin [v] +Sc [u, v];
        if v = t then Pout [v] := ∞
                else
            for u∈ SN [v] do
                Pout [v] := Pout [v] + Sc [v, u];
        P [v] := min (Pin [v], Pout [v]);
        if P [v] = 0 then STEK ⇐ v;
    end;
for v ∈ SV do Q [v] := 0; { Krovinių inicializacija tinklo Sf viršūnėse }
```

```
XN := SV;{ Aibėje XN bus saugomas tinklo S_f nenulinio potencialo viršūnės }
while XN ≠ ∅ do { Pagrindinis ciklas }
    begin
        { Nulinio potencialo viršūnių šalinimas }
        while STEK ≠ ∅ do
            begin
                v ⇐ STEK; XN := XN \ {v};
                { Lankų, įeinančių į viršūnę v, šalinimas }
                for u ∈ SPN [v] do { Lanko (u, v) šalinimas }
                    begin
                        Pout [u] := Pout [u] – (Sc [u, v] – Sf [u, v]);
                        SN [u] := SN [u] \ {v};
                        SPN [v] := SPN [v] \ {u};
                        if P [u] ≠ 0 then { Potencialo P [u] modifikacija }
                            begin
                                P [u] := min (Pin [u], Pout [u]);
                                if P [u] = 0 then STEK ⇐ u;
                            end;
                    end; { for }
                { Lankų, išeinančių iš viršūnės v, šalinimas }
                for u ∈ SN [v] do { Lanko (v, u) šalinimas }
                    begin
                        Pin [u] := Pin [u] – (Sc [v, u] –Sf [v, u]);
                        SPN [u] := SPN [u] \ {v};
                        SN [v] := SN [v] \ {u};
                        if P [u] ≠ 0 then { P [u] modifikacija }
                            begin
                                P [u] := min (Pin [u], Pout [u]);
                                if P [u] = 0 then STEK ⇐ u;
                            end;
                    end; { for }
            end; { while STEK ≠ ∅ }
                { XN – tai nenulinio potencialo viršūnių aibė }
                if XN ≠ ∅ then { Srautas dar nepseudomaksimalus }
                    begin
                        { Minimalaus potencialo viršūnės r apskaičiavimas }
                        p := ∞;
                        for v ∈ XN do
                        if P [v] < p then
                            begin
```

```
            r :=v;
            p := P [r];
        end;
{ Dydžio p srauto iš viršūnės r į viršūnę t konstravimas }
Eilė := ∅; Eilė :⇐ r; Q [r] := p;
repeat
    v ⇐ Eilė;
    Pin [v] := Pin [v] – Q [v];
    Pout [v] := Pout [v] – Q [v];
    P [v] := P [v] – Q [v];
    if P [v] = 0 then STEK ⇐ v;
    if v = t then Q [v] := p
            else
        begin { Viršūnės v “iškrova” }
            u := pirmoji aibės SN [v] viršūnė;
            while Q [v] > 0 do
                begin
                    if Q [u]  = 0 then Eilė :⇐ u;
                    delta := min (Q [v], Sc [v, u] – Sf [v, u]);
                    Sf [v, u] := Sf [v, u] + delta;
                    Q [v] := Q [v] – delta;
                    Q [u] := Q [u] + delta;
                    if Sf [v, u] = Sc [v, u] then { Lanko (v, u)
                    šalinimas }
                        begin
                            SN [v] := SN [v] \ {u};
                            SPN [u] := SPN [u] \ {v};
                        end;
                    if Q [v] > 0 then
                        u := po viršūnės u einanti kita aibės SN [v]
                        viršūnė;
                end; { while Q [v] > 0 }
        end; { viršūnės v “iškrovos” pabaiga }

until v = t; { srautas iš viršūnės r į t surastas }

{ Dydžio p srauto iš šaltinio s į viršūnę r konstaravimas }
Eilė := ∅; Eilė :⇐ r; Q [r] := p;
    repeat
        v ⇐ Eilė;
```

```
      if v ≠ r then { P [v] dar nesumažintas }
         begin
            Pin [v] := Pin [v] – Q [v];
            Pout [v] := Pout [v] – Q [v];
            P [v] := P [v] – Q [v];
            if P [v] = 0 then STEK ⇐ v;
         end;
      if v = s then Q [v] := 0
            else
         begin { Viršūnės v “iškrova” }
            u := pirmoji aibės SPN [v] viršūnė;
            while Q [v] > 0 do
               begin
                  if Q [u] = 0 then Eilė :⇐ u;
                  delta := min (Q [v], Sc [u, v] – Sf [u, v]);
                  Sf [u, v] := Sf [u, v] + delta;
                  Q [v] := Q [v] – delta;
                  Q [u] := Q [u] + delta;
                  if Sf [u, v] = Sc [u, v] then { Lanko (u, v)
                  šalinimas }
                     begin
                        SPN [v] := SPN [v] \ {u};
                        SN [u] := SN [u] \ {v};
                     end;
                  if Q [v] > 0 then
                     u := po viršūnės u einanti kita aibės
                     SPN [v] viršūnė;
               end; { while Q [v] > 0 }
         end; { viršūnės v “iškrovos” pabaiga }

      until v = s; { srautas iš viršūnės s į r surastas }

   end; { if XN ≠ ∅ }
 end; { pagrindinio ciklo while XN ≠ ∅  pabaiga }
end; { procedūros maxpsa pabaiga }
```

Aptarkime procedūros ***maxpsa*** sudėtingumą. Pradinis viršūnių potencialų apskaičiavimas reikalauja $O(n+m)$ veiksmų, kadangi kiekvienas grafo lankas analizuojamas ne daugiau kaip du kartus: apskaičiuojant *Pout* [*v*] ir *Pin* [*v*].

Pagrindinio ciklo nulinio potencialo viršūnių šalinimo sudėtingumas taip pat yra $O(n+m)$, kadangi kiekviena viršūnė į steką STEK talpinama vieną kartą, o šalinant viršūnę iš steko pašalinamos visos jai incidentiškos briaunos, tuo būdu kiekvienas lankas šalinamas ne daugiau kaip vieną kartą.

Kiekviena pagrindinio ciklo iteracija iššaukia ne mažiau kaip vienos nulinio potencialo viršūnės atsiradimą (viršūnės *r* potencialas visada bus lygus nuliui). Vadinasi, pagrindinio ciklo pasikartojimų skaičius neviršija *n*.

Pagrindiniame cikle, be minėtų nulinio potencialo viršūnių šalinimo veiksmų, naudojant paiešką platyn konstruojamas *p* dydžio srautas iš *r* į *t* ir iš *s* į *r*. Šių dalių sudėtingumas yra lygus lankų analizavimo skaičiaus eilei.

Lanko analizė gali būti “naikinamoji”, kai srautas per lanką yra lygus to lanko pralaidumui, ir toks lankas šalinamas iš tinklo.

Lanko analizė gali būti “nenaikinamoji”, kai srauto per lanką dydis yra mažesnis už to lanko pralaidumą. Toks lankas iš tinklo nešalinamas ir gali būti analizuojamas kartojant pagrindinį ciklą.

Aišku, kad per visus pagrindinio ciklo pasikartojimus “naikinamuoju” būdu analizuojame $O(m)$ lankų, o kiekviename pagrindiniame cikle “nenaikinamuoju” būdu analizuojame ne daugiau kaip *n* lankų (po vieną kiekvienai aplankytai viršūnei). Vadinasi, per visus pagrindinio ciklo pasikartojimus “nenaikinamuoju” būdu bus analizuota ne daugiau kaip n^2 lankų. Tuo būdu, bendras pagrindinio ciklo veiksmų skaičius yra $O(m+n^2)$, t.y. $O(n^2)$.

Iš viso nagrinėjimo darome išvadą, kad procedūros ***maxpsa*** sudėtingumas yra $O(n^2)$.

Dabar aprašytas ***psa*** ir ***maxpsa*** procedūras galima sukomponuoti į vieną bendrą maksimalaus srauto apskaičiavimo algoritmą.

Maksimalaus srauto apskaičiavimo algoritmas

Duota: tinklas, nusakytas gretimumo struktūromis $N(v)$ ir $PN(v)$, $v \in V$; laukų pralaidumu $c[u,v]$, $u,v \in V$; šaltiniu *s* ir nuotakiu *t*.

Rezultatas: maksimalus srautas $f[u,v]$, $u,v \in V$.

```
begin
   for u ∈ V do
      for v ∈ V do f [u, v] :=0; { Pradinis nulinis srautas }.
   repeat
      psa; { Pagalbinio  be kontūrų tinklo Sf formavimas }
      if d [t] ≠ ∞ then { srautas f – nemaksimalus }
```

```
begin
    maxpsa; { Pseudomaksimalaus srauto tinkle Sf apskaičiavimas }
    { Pseudomaksimalaus srauto perkėlimas į pagrindinį tinklą }
    for u ∈ SV do { SV – tinklo Sf viršūnių aibė }
    for v ∈ SV do
        begin
            f [u, v] := f [u, v] + Sf [u, v];
            if f [u, v] > c [u, v] then
                begin
                    f [v, u] := f [v, u] – (f [u, v] – c [u, v]);
                    f [u, v] := c [u, v];
                end;
        end;
end; { fazės pabaiga }

  until d [t] = ∞; { srautas f – maksimalus }
end; { algoritmo pabaiga }
```

Išvada. Iš procedūrų ***psa*** ir ***maxpsa*** sudėtingumo analizės bei to, kad fazių skaičius neviršija $(n-1)$ išplaukia, kad pateikto maksimalaus srauto algoritmo sudėtingumas yra $O(n^3)$. Tuo pačiu šių algoritmų analizė parodė, kad bet kokiam tinklui egzistuoja maksimaus srautas. O tai ir yra Fordo ir Falkersono teoremos pilno įrodymo trūkstama grandis.

Šį paragrafą baigsime akivaizdžia, bet svarbia pateikta maksimalaus srauto radimo algoritmo savybe.

Teorema. Jeigu tinklo visų lankų pralaidumai yra sveikieji skaičiai, tai maksimalus srautas, apskaičiuotas pagal pateiktą algoritmą, yra sveikaskaitinis, t.y. $f(u,v)$ yra sveikasis skaičius kiekvienam tinklo lankui (u,v).

2.17.3. Maksimalaus suporavimo uždavinys dvidaliame grafe

Apibrėžimas. Suporavimas neorientuotajame grafe $G=(V,U)$ – tai toks grafo G briaunų poaibis $M(M \subseteq U)$, kad bet kokios dvi poaibio M briaunos tarpusavyje nėra gretimos, t.y. bet kokios dvi poaibio M briaunos nėra incidentiškos vienai ir tai pačiai viršūnei.

Jei briauna $(u,v) \in M$, tai sakome, kad M ***suporuoja u ir v viršūnes***.

Jei viršūnė v nėra incidentiška nei vienai poaibio M briaunai, tai viršūnė v vadinama ***laisvąja viršūne***, priešingu atveju – ***suporuotąja***.

Maksimalaus suporavimo neorientuotame grafe uždavinys yra plačiai paplitęs, ir yra žinomi efektyvūs šio uždavinio sprendimo algoritmai. Visi jie pagrįsti alternuojančių grandinių teorija (teorija čeredujuščichsia cepei).

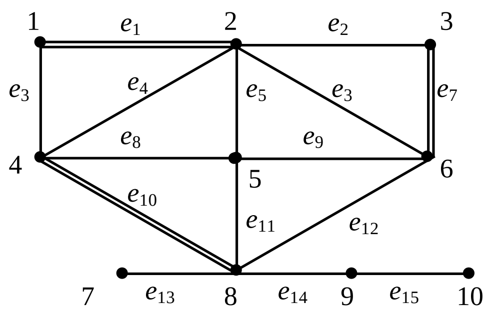
Tarkim, M – grafo G suporavimas.

Apibrėžimas. Grafo G grandinė, kurią sudaro biaunos, pakaitomis nepriklausančios ir priklausančios poaibiui M, vadinama suporavimo M atžvilgiu alternuojančia grandine.

Grandinė, kurios ilgis lygus 1, pagal apibrėžimą taip pat yra alternuojanti.

Alternuojančios grandinės briaunos, priklausančios suporavimui M, vadinamos ***tamsiomis briaunomis***, o nepriklausančios M – ***šviesiomis briaunomis***.

Pavyzdys. Panagrinėkime 2.17.5 pav. pavaizduotą grafą.



2.17.5 pav. Suporavimo uždavinys

Aibė $M = \{e_1, e_7, e_{10}\}$ yra suporavimas grafe G; grandinė $P = (7,8,4,1,2,5)$ yra suporavimo M atžvilgiu alternuojanti grandinė; $e_1 = \{1,2\}$, $e_{10} = \{4,8\}$ – tamsiosios grandinės P briaunos; $e_3 = \{1,4\}$, $e_5 = \{2,5\}$, $e_{13} = \{7,8\}$ – šviesiosios grandinės P briaunos; aibės $\{1,2,3,4,6,8\}$ ir $\{5,7,9,10\}$ – atitinkamai suporuotųjų ir laisvųjų viršūnių aibės.

Aišku, kag jei grafe G suporavimo M atžvilgiu egzistuoja grandinė, jungianti dvi laisvąsias viršūnes, tai grafe G galima sukonstruoti suporavimą, turintį didesnį briaunų skaičių nei M. Aišku, kad tokios alternuojančios grandinės šviesiųjų briaunų skaičius yra vienetu didesnis nei tamsiųjų. Pašalinę iš M visas tamsiasias grandinės briaunas, ir į M įvedę visas šviesiasias tos grandinės briaunas, gausime suporavimą, kuriame briaunų skaičius bus vienetu didesnis nei M. Dėl tos priežasties alternuojanti grandinė, jungianti dvi laisvasias viršūnes, vadinama didinančiaja grandine.

Teorema (K.Beržas). Neorientuotojo grafo G suporavimas M yra didžiausias tada ir tiktai tada, kai šiame grafe suporavimo M atžvilgiu nėra didinančiosios grandinės.

Šiai teoremai iliustruoti panagrinėkime aukščiau 2.17.5 pav. pateiktą grafą. Imkime suporavimą $M = \{e_4, e_7, e_{10}\}$ ir didinančiąją grandinę $P = \{7,8,4,1,2,5\} = \{e_{13}, e_{10}, e_3, e_1, e_5\}$. Dabar galima sudaryti didesnį suporavimą $M' = \{M \setminus \{e_1, e_{10}\}\} \cup \{e_3, e_5, e_{13}\} = \{e_3, e_5, e_7, e_{13}\}$.

Suporavimas M' taip pat nėra didžiausias, nes šio suporavimo atžvilgiu egzistuoja grandinė $P = \{9,10\} = \{e_{15}\}$.

Gausime suporavimą $M'' = M' \cup \{e_{15}\} = \{e_3, e_5, e_7, e_{13}, e_{15}\}$, kuris yra didžiausias, nes jo atžvilgiu grafe nėra didinančiosios grandinės.

Tuo būdu, Beržo teorema nusako tokią didžiausio suporavimo radimo strategiją.

1. Randame bet kokį pradinį suporavimą *M*.
   Pavyzdžiui, pradinis suporavimas gali būti būti kuri grafo briauna.
   Didesnio pradinio suporavimo galima ieškoti, naudojant "godaus" algoritmo strategiją:
   a) rasti briauną, kurios incidentiškų viršūnių laipsnių suma yra mažiausia, ir įtraukti šią briauną į suporavimą;
   b) iš grafo pašalinti į suporavimą įtrauktą briauną drauge su jai gretimomis briaunomis.

   Aišku, kad punktai a) ir b) bus kartojami tol, kol grafas *G* turės bent vieną briauną.
2. Nuosekliai konstruoti suporavimų seką $M_1 = M, M_2, M_3, ..., M_k, M_{k+1}, ...$, kurioje M_{k+1} gaunamas iš M_k radus grafe G suporavimo M_k atžvilgiu didinančiąją grandinę. Kadangi $|M_{k+1}| = |M_k| + 1$, tai sekos ilgis bus nedidesnis nei $\lfloor n/2 \rfloor$. Todėl norint rasti efektyvų šio uždavinio sprendimo algoritmą, pagrįstą išnagrinėta strategija, reikia sukonstruoti efektyvų didinančiosios grandinės radimo algoritmą. Nors tokie efektyvūs algoritmai egzistuoja, čia apsiribosime didžiausio suporavimo dvidaliame grafe nagrinėjimu.

Dažniausiai suporavimo uždavinys sprendžiamas dvidaliuose grafuose (žr. 2.2, 2.10 paragrafus). Tai klasikinis kombinatorikos uždavinys, žinomas "uždavinio apie sutuoktinių poras" vardu.

Primename, kad (n, m)-grafas $G = (V, U)$ yra dvidalis, jei jo viršūnių aibę *V* galima išskaidyti į du poaibius *X* ir *Y* taip, kad visų grafo *G* briaunų galai priklausytų skirtingiems poaibiams. Todėl dvidalis grafas žymimas $G = (X, Y, U)$.

"Uždavinys apie sutuoktinių poras" turi tokią interpretaciją. Tarkime, X ir Y atitinkamai yra vaikinų ir merginų aibės. Sudarykime dvidalį grafą $G = (X, Y, U)$, čia $(x, y) \in U$, jei jaunuolis x draugauja su mergaite y. Tada kiekvienas suporavimas M atitinka aibę galimų sutuoktinių porų, kiekviena iš kurių sudaryta iš tarpusavyje draugaujančių jaunuolio ir merginos; be to, kiekvienas žmogus dalyvauja ne daugiau kaip vienoje poroje.

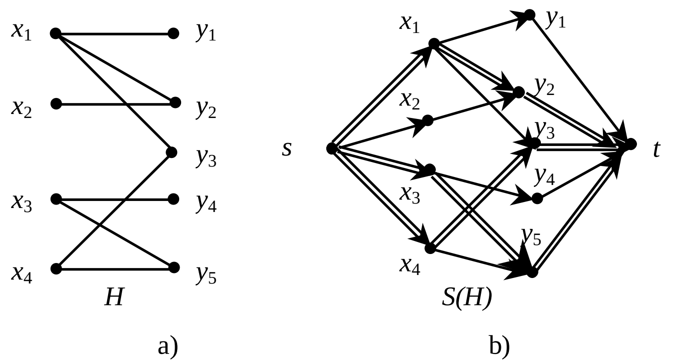
Pasirodo, kad maksimalaus suporavimo uždavinį dvidaliame grafe galima lengvai suvesti į maksimalaus srauto tinkle ieškojimo uždavinį.

Tarkime, $H = (X, Y, U)$ – dvidalis grafas. Sudarykime tinklą $S(H)$, kuris turėtų šaltinį s, nuotakį t ($s \neq t$ ir $s, t \notin X \cup Y$), viršūnių aibę $V^* = \{s, t\} \cup X \cup Y$, lankų aibę

$$E^* = \{(s, x), x \in X\} \cup \{(y, t), y \in Y\}$$
$$\cup \{(x, y) : x \in X \wedge y \in Y \wedge (x, y) \in U\}$$

ir kiekvieno lanko $u, v \in E^*$ pralaidumas $c(u, v) = 1$.

Pavyzdys. 2.17.6 a) pav. pavaizduotas dvidalis grafas H, o 2.17.6 b) pav. – jam atitinkantis tinklas $S(H)$.



2.17.6 pav. Dvidalis grafas H ir jam atitinkantis tinklas $S(H)$. Suporavimas $M = \{(x_1, y_2), (x_3, y_5), (x_4, y_3)\}$ ir jam atitinkantis srautas f_M.

Teorema. Kiekvienam grafo H k galios suporavimui $M = \{(x_1, y_1), (x_2, y_2), ..., (x_k, y_k)\}$, ($x_i \in X$, $y_i \in Y$ visiems $i = \overline{1, k}$)

egzistuoja vienintelis tinklo $S(H)$ dydžio k sveikaskaitinis (0,1) srautas f_M apibrėžiamas taip:

$$f_M(s, x_i) = f_M(x_i, y_i) = f_M(y_i, t) = 1 \text{ visiems } i = \overline{1, k},$$

$$f_M(e) = 0 \text{ visiems likusiems tinklo lankams } e;$$

ir atvirkščiai, kiekvienam tinklo $S(H)$ dydžio k (0,1) srautui f atitinka vienintelis grafo H suporavimas M_f, $|M_f| = k$, apibrėžiamas taip:

$$M_f = \{(x, y) : x \in X \wedge y \in Y \wedge f(x, y) = 1\}.$$

2.17.6 pav. pavaizduotas suporavimas $M = \{(x_1, y_2), (x_3, y_5), (x_4, y_3)\}$ grafe H ir jam atitinkantis srautas f_M tinkle $S(H)$.

Ši teorema įgalina maksimalaus srauto apskaičiavimo algoritmus, išnagrinėtus 2.17.6 paragrafe, panaudoti apskaičiuojant maksimalų suporavimą dvidaliame grafe.

Naudodami Dinico metodą, maksimalų suporavimą galima apskaičiuoti atlikus $O(n^3)$ veiksmų. Tačiau, atsižvelgiant į tinklo $S(H)$ specifiką, yra sukurti efektyvesni maksimalaus suporavimo apskaičiavimo algoritmai. Čia panagrinėsime Hopkrofto ir Karpo algoritmą, kurio sudėtingumas yra $O(n^{5/2})$ (žr. Hopcroft J., Karp R.M. An $n^{5/2}$ algorithm for maximum Matchings in bipartite graphs. SIAM J. Comput., 1973, 2, s. 225-231).

Šis algoritmas naudoja bendrą Dinico metodo schemą. Vienok, dėl tinklo $S(H)$ specifikos galima sumažinti tiek fazių skaičių, tiek ir sukurti efektyvesnį pseudomaksimalaus srauto kiekvienoje fazėje apskaičiavimo algoritmą.

Pirmiausia pašymėsime, kad tinklo $S(H)$ kiekvienos didinančiosios grandinės ilgis yra nelyginis skaičius, ir, iš jos atmetus pirmąją ir paskutiniąją grandis, ši grandinė bus alternuojanti, prasidedanti ir pasibaigianti šviesiaja briauna (leistinaisiais suderintaisiais lankais).

Apibrėžimas. Duotajam suporavimui M ilgio $l = 2k + 1$, $k > 0$ iš X į Y grandinę $P \subseteq U$:

$$P = \{(x_0, y_1), (y_1, x_1), (x_1, y_2), ..., (y_k, x_k), (x_k, y_{k+1})\},$$

kurioje visos viršūnės $x_0, x_1, ..., x_k, y_1, y_2 ..., y_{k+1}$ yra skirtingos, x_0 – laisvoji X aibės viršūnė, o y_{k+1} – laisvoji Y aibės viršūnė, o kiekviena antroji briauna priklauso M, t.y.

$$P \cap M = \{(y_1, x_1), (y_2, x_2), ..., (y_k, x_k)\}$$

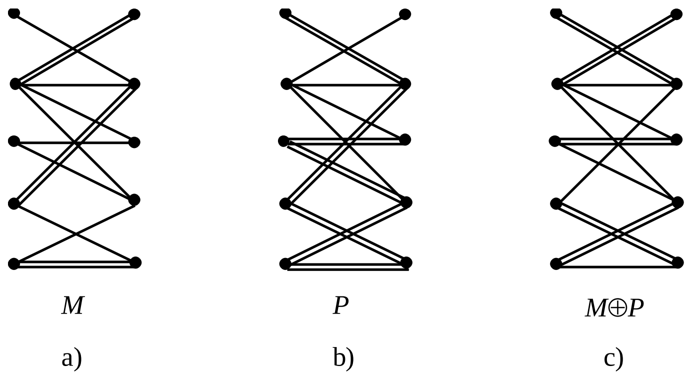
vadinsime alternuojančia grafo H grandine.

Aišku, kad alternuojančią grandinę galima nusakyti viršūnių $x_0, x_1,..., x_k, y_1, y_2..., y_{k+1}$ seka. Ši seka vieninteliu būdu apibrėžia srauto f_M atžvilgiu didinančiąją grandinę:

$$s,(s,x_0),x_0,(x_0,y_1),y_1,(y_1,x_1),x_1,(x_1,y_2),y_2,...,$$
$$y_k,(y_k,x_k),x_k,(x_k,y_{k+1}),y_{k+1},(y_{k+1},t),t.$$

Be to, šios grandinės iššauktas srauto f_M padidėjimas nusako suporavimą, gautą susumavus aibes *A* ir *P* moduliu 2: $M \oplus P$.

Pavyzdžiui, 2.17.7 a) pav. pavaizduotas dvidalis grafas ir suporavimas M; 2.17.7 b) pav. pavaizduota didinančioji alternuojančioji grandinėP, o 2.17.7 c) pav. pavaizduotas suporavimas $M \oplus P$.



2.17.7 pav. Suporavimo *M* padidinimas, panaudojant alternuojančią grandinę *P*

Aišku, kad ir dvidalio grafo *H* atveju yra teisinga Beržo teorema: grafo *H* suporavimas *M* yra didžiausias tada ir tiktai tada, kai šiame grafe suporavimo *M* atžvilgiu nėra didinančiosios grandinės.

Aprašysime Hopkrofto ir Karpo algoritmą. Šio algoritmo schema yra analogiška Dinico algoritmo schemai: kiekvienoje fazėje tinklui $S(H)$ apskaičiuosime pagalbinį bekontūrinį tinklą, jo pseudomaksimalų srautą ir jį "perkelsime" į pagrindinį tinklą. Kaip buvo minėta aukščiau, dėl tinklo $S(H)$ specifikos šie algoritmai yra efektyvesni nei Dinico metode naudojamas analogiškas algoritmas, o fazių skaičius – mažesnis.

Pirmiausia aptarsime pagalbinio bekontūrinio tinklo apskaičiavimo algoritmą ***PGA***.

Tarkime, $H = (X,Y,U)$ yra dvidalis grafas, o *M* – suporavimas šiame grafe. Grafą *H* pakeiskime orientuotuoju grafu H_M tokiu būdu: kiekvienai suporavimo M briaunai $e \in M$ suteikime orientaciją nuo Y į X, o visoms likusioms briaunoms – orientaciją nuo X į Y. Aišku, kad tada kelias nuo laisvosios viršūnės $x \in X$ į laisvąją viršūnę $y \in Y$ bus srauto M atžvilgiu didinančioji grandinė.

Procedūra ***PGA*** pradžioje konstruoja bekontūrinio tinklo lankus nuo viršūnės *s* į visas laisvąsias aibės *X* viršūnes. Po to iš šių laisvųjų viršūnių grafe H_M organizuojama paieška platyn. Jei paieškos platyn metu pasiekiame laisvąją viršūnę $y \in Y$, tai į bekontūrinį tinklą įtraukiame lanką (y,t). Be to, nuo to momento, kai į tinklą įtraukiamas pirmasis toks lankas, daugiau nenagrinėjamos viršūnės, kurios nutolę nuo viršūnės *s* nemažiau kaip kelio nuo *s* iki *t* ilgis.

Panagrinėkime pseudomaksimalų srautą procedūros ***PGA*** sukonstruotame pagalbiniame tinkle. Šiame pagalbiniame $(l+2)$ ilgio tinkle *l* ilgio (atmetame 1-ąjį ir paskutinįjį lankus) alternuojančios didinančiosios grandinės visų viršūnių potencialai yra lygūs 1. Pagal šią grandinę padidinus srautą, visos grandinės viršūnės eliminuojamos iš tolesnio nagrinėjimo. Vadinasi, pagalbinio tinklo pseudomaksimalų srautą apibrėš didžiausia aibė ilgio *l* poromis nesusikertančių alternuojančių grandinių. Tokių grandinių ieškojimą realizuoja procedūra ***fazė***, kuri naudoja paieškos gilyn iš viršūnės *s* pagalbiniame tinkle metodą. Naujos aplankytos viršūnės talpinamos į steką STEK. Kiekvieną kartą, kai pasiekiama viršūnė *t*, steke esančios viršūnės apibrėžia alternuojančią didinančiąją grandinę. Tada visos viršūnės, išskyrus viršūnę *s*, šalinamos iš steko kartu didinant srautą (suporavimą), kurį nusako masyvas *pora*. Žemiau pateikti procedūrų ***PGA***, ***fazė*** ir ***pagrindinės programos*** tekstai.

Hopkrofto ir Karpo didžiausio suporavimo algoritmas

Duota: Dvidalis grafas $H=(X,Y,U)$, nusakytas gretimumo struktūra $N(x)$, $x \in X$, čia $N(x)$ – aibė viršūnių, gretimų viršūnei *x*.

Rasti: Didžiausią suporavimą, kurį nusako masyvas *pora* [1..*n*], $n = |X| + |Y|$, kurios elementas

$$pora[v] = \begin{cases} u, \text{ jei pora } (u,v) \text{ priklauso didžiausiam suporavimui,} \\ 0, \text{ jei viršūnė } v \text{ - laisva,} \end{cases}$$

$v \in V = X \cup Y$.

procedure ***PGA***;

{ *Pagalbinio bekontūrinio tinklo S* (*H*) *konstravimas.*

Kintamieji V, SV, X, Y, pora, N, SN, s ir t – globalieji. Čia, kaip ir maksimalaus srauto apskaičiavimo algoritme, (*žr. 2.17.2 paragrafą*) *SV – tinklo S* (*H*) *viršūnių aibė, o SN – tinklo S* (*H*) *gretimumo struktūra.* }

```
begin
   for u ∈ V ∪ {s, t} do { inicializacija }
      begin
         d [u] = ∞; { viršūnės v nuotolis nuo viršūnės s }
         SN [u] := ∅;
      end;
   { Laisvąsias viršūnes x ∈ X patalpinti į eilę ir į SN [s]. }
   Eilė := ∅; SV := ∅;
   SV := SV ∪ {s}; { SV – pagalbinio tinklo viršūnių aibė }
   d [s] := 0;
   for x ∈ X do
      if  pora [x] = 0  then { x – laisvoji viršūnė }
         begin;
            Eilė ⇐ x;
            SN [s] := SN [s] ∪ {x};
            d [x] := 1;
         end;
   { Paieška platyn iš laisvųjų viršūnių x ∈ X }
   while Eilė ≠ ∅ do
      begin
         u ⇐ Eilė;
         SV := SV ∪ {u};
         if  u ∈ Y then
            if pora [u] = 0 then {u – laisvoji viršūnė; prijungti lanką (u, t).}
            begin
               SN [u] := SN [u] ∪ t;
               if d [t] = ∞ then
                  begin
                     SV := SV ∪ {t};
                     d [t] := d [u] + 1;
                  end;
            end
                         else { pora [u] ≠ 0 }
                           begin
                              x := pora [u];
                              if d [t] = ∞  then { prijungti lanką (u, x) }
```

```
                              begin
                                  Eilė ⇐ x;
                                  SN [u] := SN [u] ∪ {x};
                                  d [x] := d [u] + 1;
                              end;
                      end
              else { u ∉ Y, t.y. u ∈ X }
          for y ∈ N [u] do
              if d [u] < d [y] then { d [y] = d [u] + 1 arba d [y] = ∞. }
                  begin
                      if d [y] = ∞ then { y – nauja viršūnė } Eilė ⇐ y;
                      SN [u] := SN [u] ∪ {y};
                      d [y] := d [u] + 1;
                  end;
    end; { while }
end; { PGA }

procedure fazė;
{ Turimo suporavimo padidinimas, remiantis didžiausia aibe poromis
nesusikertančių kelių iš s į t pagalbiniame tinkle S (H).
Kintamieji SV, pradžia, SN, pora, s ir t – globalieji.
Procedūra fazė – tai modifikuota paieškos gilyn procedūra gylis3 (žr. 2.8.1.3
paragrafą). }
begin
    for u ∈ SV do { inicializacija }
        begin
            naujas [u] := true;
            p [u] := pradžia [u]; { pradžia [u] = rodyklė į sąrašo pradžią, –
            pirmąjį elementą; p [u] = rodyklė į pirmąjį analizuojamąjį sąrašo
            SN [u] elementą }
        end;
    { Paieška gilyn iš viršūnės s pagalbiniame tinkle (žr. 2.8.1.3 paragrafą }
    STEK := ∅; STEK ⇐ s; naujas [s] := false;
    while STEK ≠ ∅ do { pagrindinis ciklas }
        begin
            u := top (STEK); { u = viršutinis steko elementas }
            { Sąraše SN [u] ieškome pirmos naujos viešūnės }
            if p [u] = nil then { sąraše SN [u] naujų viršūnių nėra } b := false
                           else b := not naujas [p [u]↑.viršūnė];
                while b do
```

```
                begin
                    p [u] := p [u] ↑.pėdsakas;
                    if p [u] = nil then b := false
                                    else b := not naujas [p [u]↑.viršūnė];
                end;
        if p [u] ≠ nil then { Sąraše SN [u] randame naują viršūnę }
            if p [u]↑.viršūnė = t then { radome alternuojančią grandinę }
            { Turimo suporavimo didinimas, remiantis steke saugoma
            alternuojančia grandine }
            while top (STEK) ≠ s do
                begin
                    y ⇐ STEK;
                    x ⇐ STEK;
                    pora [x] := y; pora [y] := x;
                end; { while top (STEK) ≠ s }
                { Steke liko tik šaltinis s }
                                else { p [u]↑.viršūnė ≠ t }
                                    begin
                                        v := p [u]↑.viršūnė;
                                        STEK ⇐ v;
                                        naujas [v] := false;
                                    end
                        else { p [u] = nil. Sąraše SN [u] nėra naujų viršūnių }
        u ⇐ STEK; { Šalinamas viršutinis steko elementas }
    end; { while STEK ≠ ∅ }
end; { fazė }

begin { pagrindinė programa }
    for v ∈ V do pora [v] := 0;
    { Inicializacija. Suporavimas – tuščioji aibė. }
    PGA; { Pradinio pagalbinio tinklo sudarymas }
    while t ∈ SV do
        begin
            fazė;
            PGA;
        end;
end;
```

Reikia pažymėti, kad didelis Hopkrofto ir Karpo algoritmo efektyvumas gaunamas todėl, kad fazių skaičius yra nedidesnis nei $\sqrt{n}$ [Lip88].