2.15 Maršrutai

Šiame paragrafe nagrinėsime Oilerio ir Hamiltono maršrutus: grandines ir ciklus. Šie klausimai iškyla ne tiktai sprendžiant įvairius galvosūkius, bet ir praktiniuose uždaviniuose.

Pavyzdžiui, buitinio aptarnavimo mašina turi aplankyti kokio tai rajono visas gatves. Koks turi būti jos maršrutas? Tai Oilerio maršruto uždavinys.

Su Hamiltono maršrutais susijęs garsus komivojažerio arba keliaujančio pirklio uždavinys, kuris formuluojamas taip. Turime *n* miestų, o matrica $C=[c_{ij}]$ $i=\overline{1,n}, \quad j=\overline{1,n}$ yra atstumų matrica: c_{ij} – atstumas tarp miestų *i* ir *j*. Pirklys, išėjęs iš 1-ojo miesto turi apeiti visus miestus po vieną kartą ir grįžti į 1-ąjį miestą. Koks turi būti pirklio maršrutas, kad jo ilgis būtų trumpiausias?

Oilerio maršrutai

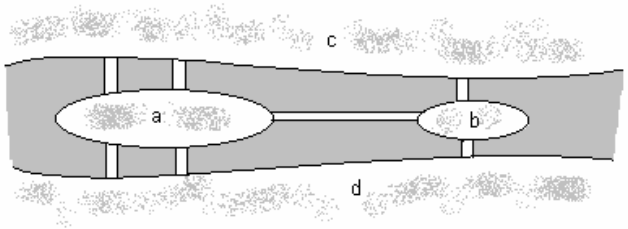
Visa, kas bus kalbama šiame paragrafe, tinka ir multigrafams.

Apibrėžimas. Maršrutas (kelias), apeinantis visas grafo briaunas po vieną kartą, vadinamas ***Oilerio maršrutu***. Jei pradinė ir galinė maršruto viršūnės nesutampa, tai toks maršrutas vadinamas ***Oilerio grandine***, priešingu atveju – ***Oilerio ciklu***.

Grafas, turintis Oilerio maršrutą, vadinamas Oilerio grafu.

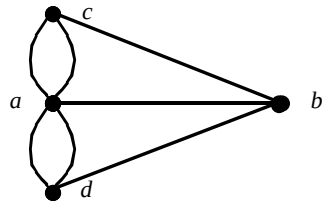
Reikia pastebėti, kad Oilerio maršruto radimo uždavinys yra pirmasis grafų teorijos uždavinys. Tai uždavinys apie Karaliaučiaus tiltus, kurį 1736 m. sėkmingai išsprendė Oileris.

Per Karaliaučių teka upė Prėglis, kurioje yra dvi salos. Šios salos su krantais ir tarpusavyje yra sujungtos tiltais (žr. 2.15.1 pav.).



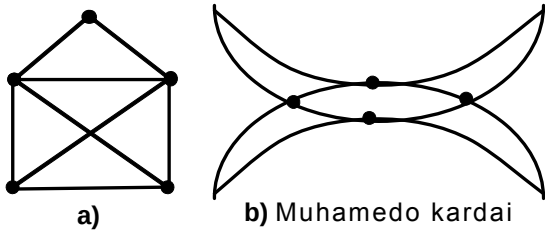
2.15.1 pav. Karaliaučiaus tiltai

Ar galima, išėjus iš namų, apeiti visus tiltus po vieną kartą ir vėl grįžti į namus. Kitaip tariant, ar multigrafas, pavaizduotas 2.15.2 pav. turi Oilerio maršrutą?



2.15.2 pav. Karaliaučiaus tiltų grafas

Su Oilerio maršrutu yra susiję įvairūs galvosūkiai. Pavyzdžiui, ar galima, neatitraukus rankos nuo popieriaus, nupiešti figūras pavaizduotas 2.15.3 a) ir b) pav.?



2.15.3 pav. Oilerio maršrutai

Kaip minėjome, ar grafas turi Oilerio maršrutą, išsprendė Oileris įrodydamas teoremą.

Teorema (Oilerio teorema, 1736). Būtina ir pakankama sąlyga, kad grafas G turėtų Oilerio maršrutą yra:

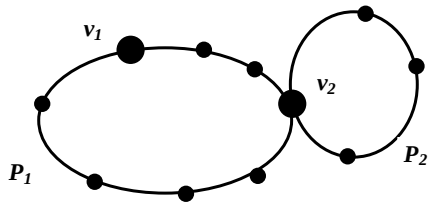
1. G turi būti jungusis,
2. visų jo viršūnių laipsniai turi būti arba lyginiai, arba G turi turėti tik dvi nelyginio laipsnio viršūnes.

Pirmuoju atveju grafas turi Oilerio ciklą, o antruoju – grandinę, prasidedančią ir besibaigiančią nelyginio laipsnio viršūnėse.

Pateiksime šios teoremos įrodymą, kadangi jis yra konstruktyvus: iš įrodymo išplaukia Oilerio maršruto konstravimo algoritmas.

***Būtinumas*.** Tarkime G – Oilerio grafas. Oilerio ciklas eidamas per kiekvieną viršūnę , patenka į viršūnę viena briauna, o išeina – kita briauna. Tai reiškia, kad kiekviena grafo G viršūnė incidentiška lyginiam Oilerio ciklo briaunų skaičiui. Kadangi Oilerio ciklas apima visas grafo briaunas, tai reiškia, kad grafas yra jungusis ir, kad kiekvienos grafo viršūnės laipsnis yra lyginis.

Pakankamumas. Tarkime, kad visų grafo G viršūnių laipsniai yra lyginiai. Pradėkime kelionę iš viršūnės v_1 ir, eidami grafo briaunomis, laikysimės taisyklės: “niekada neiti ta pačia briauna”. Tai tolygu taisyklei: “pereitą briauną – ištriname”. Atėję į bet kurią viršūnę $v \neq v_1$, iš jos visada galėsime išvykti, nes viršūnės v laipsnis yra lyginis. Jei patekome į viršūnę, iš kurios negalime išvykti, tai reiškia, kad esame pradinėje kelio viršūnėje v_1. Vadinasi mūsų kelias sudarė ciklą P_1. Pašalinę šį ciklą iš grafo G, gausime grafą G', kurio kiekvienos viršūnės laipsnis yra arba lyginis, arba lygus nuliui. Jei visų viršūnių laipsniai yra lygūs nuliui, tai reiškia, kad P_1 yra Oilerio ciklas. Priešingu atveju nagrinėsime grafą G'. Kadangi grafas G yra jungusis grafas, tai grafai P_1 ir G' turi bent vieną bendrą viršūnę v_2. Pradėję kelionę grafe G' iš viršūnės v_2 ir laikydamiesi tos pačios taisyklės, sukonstruosime ciklą P_2. Iš ciklų P_1 ir P_2 sukonstruosime bendrą ciklą taip: iš viršūnės v_1 ciklo P_1 briaunomis nukeliausime į viršūnę v_2, po to apeisime ciklą P_2 ir po to iš viršūnės v_2 ciklo P_1 briaunomis ateisime į viršūnę v_1 (žr. 2.15.4 pav.).



2.15.4 pav. Oilerio ciklo konstravimas

Pašalinkime šį ciklą iš grafo *G*. Jei po ciklo pašalinimo grafas yra tuščiasis, tai šis ciklas yra Oilerio ciklas. Priešingu atveju, šiame grafe atliksime analogiškus veiksmus. Ir taip elgsimės iki sukonstruosime ciklą, einantį per visas grafo *G* briaunas.

Teorema (R.Reidas, 1962). Beveik nėra Oilerio grafų. [EM90].

Primename, kad sąvoka "beveik nėra grafų" apibrėžiama taip. Simboliu *GP*(*n*) pažymėkime skaičių *n*-viršūninių grafų, turinčių Oilerio maršrutą, t.y. Oilerio grafų skaičių. Simboliu *G*(*n*) pažymėkime visų *n*-viršūninių grafų skaičių. Tada $\lim_{n\to\infty}\frac{GP(n)}{G(n)}=0$, ir sakome, kad "beveik nėra Oilerio grafų".

Oilerio ciklo konstravimo algoritmas

Duota: Jungusis grafas *G*, kurio visų viršūnių laipsniai yra arba lyginiai, arba *G* turi dvi nelyginio laipsnio viršūnes. Tarkime, kad grafas G nusakytas gretimumo struktūra: *N*(*v*) – tai aibė viršūnių gretimų viršūnei *v*.

Rasti: Sukonstruoti Oilerio maršrutą.

Įrodyme pateiktą Oilerio maršruto konstravimą patogu realizuoti naudojant du stekus: *STEK* ir *Oiler.*

Priminsime, kad stekas, tai tiesinis sąrašas, kuriame naujo elemento patalpinimas ir pašalinimas atliekamas viename sąrašo gale. Kitaip dar stekas vadinamas sąrašu LIFO (nuo angliškų žodžių "last in first out" (paskutinis į steką – pirmas iš jo)).

Steką patogu organizuoti vienmačiu masyvu *STEK* [1..*n*]. Jį charakterizuoja vienas parametras – rodyklė "*top*", kuri reiškia paskutinio elemento sąraše adresą.

Stekas tuščias {*žymime* ***STEK:=*** ∅}
*top: =*0*;*
Elemento patalpinimas į steką {*žymime* ***STEK⇐ v***}*.*
top: = top +1*;*
if top > n then "steko persipildymas"
else STEK[*top*]*: = v;*
Elemento pašalinimas iš steko {*žymime* ***v⇐ STEK***}*.*
if top = 0 *then "stekas tuščias"*
else begin
v: = STEK[*top*]*;*
top: =top -1
end;
Elemento nuskaitymas iš steko {*žymime* ***v: = top***(***STEK***)}*.*
if top=0 *then "stekas tuščias"*
else v: =STEK[*top*]*;*

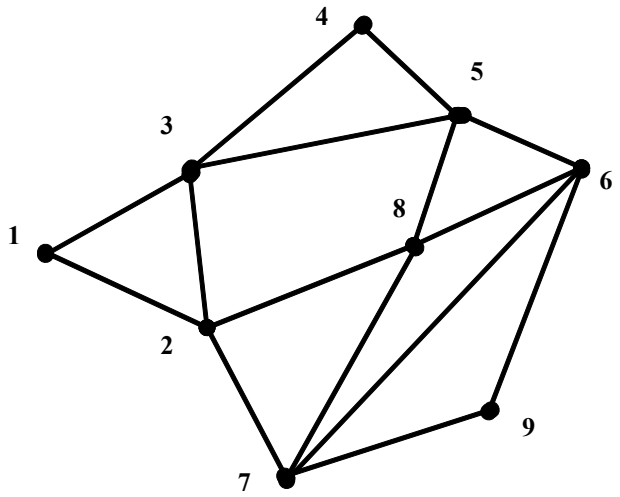
Algoritmo veikimo principas. Tarkime v_1 yra viršūnė, iš kurios pradėsime brėžti Oilerio ciklą. Šią viršūnę patalpinkime į steką *STEK*. Iš viršūnės v_1 keliausime per grafą, talpindami kelio viršūnes į steką *STEK* ir ištrindami kelio briaunas. Jei atėjome į viršūnę, iš kurios negalime išeiti, tai reiškia, kad esame pradinėje kelio

viršūnėje (jei grafo G visų viršūnių laipsniai buvo lyginiai) arba kitoje nelyginio laipsnio viršūnėje (jei v_1 laipsnis buvo nelyginis). Šią viršūnę patalpinsime į steką *Oiler* ir bandysime keliauti iš priešpaskutinės kelio viršūnės. Kelionė galima, nes iš grafo G pašalinome ciklą, einantį per viršūnę v_1, ir likusių viršūnių laipsniai yra arba lyginiai arba lygūs nuliui.

Taip elgsimės iki stekas *STEK* taps tuščias. Tada viršūnės patalpintos steke *Oiler* ir nusakys Oilerio maršrutą.

```
begin
    STEK:= ∅;  Oiler:= ∅;
    v: = bet kuri grafo viršūnė, jei visų grafo viršūnių
       laipsniai yra lyginiai arba nelyginio laipsnio
       viršūnė, jei grafas turi dvi nelyginio laipsnio
       viršūnes.
    STEK⇐ v;
    while STEK ≠ ∅ do
       begin
          v:= top(STEK); {v = viršutinis steko
                          elementas; rodyklė top
                          neperskaičiuojama}
          if  N(v) ≠ ∅
             then
                begin
                      u:= pirmoji aibės N(v) viršūnė;
                      STEK ⇐ u;
                      {pašalinti (v,u) briauną}
                      N(v):= N(v) \ {u} ;
                      N(u):= N(u) \ {v} ;
                end
             else { N(v) = ∅}
                begin
                   v⇐ STEK ;
                   Oiler ⇐ v;
                end;
       end;
end;
```

Pavyzdys. Remdamiesi algoritmu, sukonstruokime Oilerio ciklą grafui, pavaizduotam 2.15.5 pav.



Grafas užrašytas gretimumo struktūra

1: 2, 3;
2: 1, 3, 7, 8;
3: 1, 2, 4, 5;
4: 3, 5;
5: 3, 4, 6, 8;
6: 5, 7, 8, 9;
7: 2, 6, 8, 9;
8: 2, 5, 6, 7;
9: 6, 7.

2.15.5 pav. Oilerio ciklo pavyzdys

Kadangi grafo visų viršūnių laipsniai yra lyginiai, tai kelionę (Oilerio ciklo konstravimą) pradėsime iš 1-osios viršūnės. Tada stekų: *STEK* ir *Oiler* turiniai bus tokie:

STEK: 1, 2, 3, ~~1~~, 4, 5, ~~3~~, 6, 7, 2, 8, ~~5~~, 6, 9, 7, ~~8~~

Oiler: 1, 3, 5, 8, 7, 9, 6, 8, 2, 7, 6, 5, 4, 3, 2, 1.

Nubraukti elementai steke STEK žymi momentus, kai keliones metu atėjome į viršūnes, iš kurių nebuvo galima išeiti.

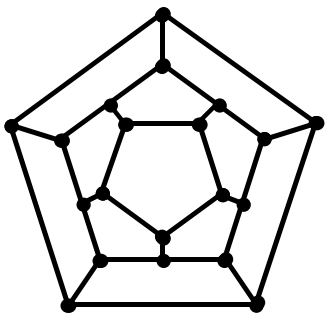
Pastaba. Praktikoje tenka spręsti tokį uždavinį. Duotas svorinis grafas *G*, kuris nėra Oilerio grafas. Kokias šio grafo briaunas pakartoti, kad jis taptų Oilerio grafu ir Oilerio ciklo ilgis būtų trumpiausias. Čia šio uždavinio nenagrinėsime, pažymėdami, kad jo sprendimas pateiktas literatūroje [K78].

Hamiltono maršrutai

Apibrėžimas. Maršrutas (kelias) apeinantis visas grafo viršūnes po vieną kartą vadinamas Hamiltono maršrutu. Jei pradinė ir galinė maršruto viršūnės sutampa, tai šis maršrutas vadinamas Hamiltono ciklu; priešingu atveju – Hamiltono grandine.

Grafas turintis Hamiltono maršrutą vadinamas Hamiltono grafu.

Hamiltono grafo sąvoka susijusi su žymaus airių matematiko W. Hamiltono vardu , kuris 1859 m. pasiūlė žaidimą, pavadintą “Kelionė aplink pasaulį”. Kiekvienai iš 20-ies dodekaedro viršūnių priskiriamas koks nors pasaulio žymaus miesto vardas. Reikia, einant iš vieno miesto į kitą, aplankyti visus miestus po vieną kartą ir grįžti į pradinį miestą. (žr. 2.15.6 pav.)

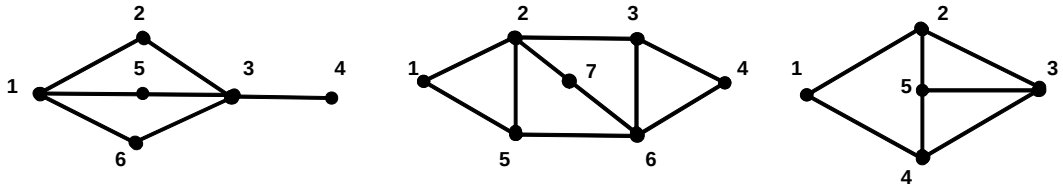


2.15.6 pav. Kelionė aplink pasaulį

Skirtingai nei Oilerio grafo atveju nėra žinoma nei vienos paprastos būtinos ir pakankamos sąlygos, kuri pasakytų ar grafas yra Hamiltono grafas. Negana to, nėra

žinoma nei vieno algoritmo, nustatančio, ar grafas yra Hamiltono grafas ir kurio veiksmų skaičius būtų išreiškiamas polinomu nuo viršūnių skaičiaus *n*. Kitaip tariant, Hamiltono grafo nustatymo uždavinys priklauso *NP* uždavinių klasei [GD82].

2.15.7 pav. pateikti trys grafai, iš kurių grafas a) nėra Hamiltono grafas, grafas b) turi Hamiltono grandinę, o grafas c) – Hamiltono ciklą.



a) Nėra b) 1, 5, 2, 7, 6, 3, c) 1, 2, 3, 5, 4, 1

2.15.7 pav. Hamiltono maršrutai

Aišku, jei grafas yra pilnasis, tai jis yra Hamiltono grafas.

Pateiksime trejetą teoremų, nusakančių pakankamas sąlygas, kad grafas būtų Hamiltono grafas.

Teorema (V. Hvatalas, 1972) [EM90]. Grafas su viršūnių laipsnių seka $d_1 \le d_2 \le \cdots \le d_n$ yra Hamiltono grafas, jei bet kuriam sveikajam *k*, tenkinančiam sąlygą $1 \le k < \lfloor n/2 \rfloor$ teisinga implikacija $(d_k \le k) \Rightarrow (d_k \ge n-k)$.

Teorema (O. Ore, 1960) [EM90]. Jei *n* viršūnių ($n \ge 3$) grafo *G* bet kuriai negretimų viršūnių porai *u* ir *v* teisinga nelygybė: $d(u)+d(v) \ge n$, tai G – Hamiltono grafas, čia *d*(*u*), *d*(*v*) – viršūnių *u* ir *v* laipsniai.

Teorema (G. Dirakas, 1952) [EM90]. Jei *n* viršūnių grafo *G* kiekvienos viršūnės laipsnis nemažesnis nei $n/2$, tai *G* - Hamiltono grafas.

Aptarkime Hamiltono maršrutų radimo uždavinį:

1. nustatyti ar duotasis grafas *G* yra Hamiltono grafas ir
2. duotajam grafui *G* rasti visus Hamiltono maršrutus.

Aišku, kad 1)-asis uždavinys yra 2)-ojo uždavinio atskirasis atvejas, todėl čia nagrinėsime 2)-ąjį uždavinį.

Aišku, kad visus Hamiltono maršrutus rasime išnagrinėję visas maršrutų galimybes, kurių skaičius yra *n*! . Kitaip tariant, kiekvienas n – elementų kėlinys gali būti Hamiltono maršrutas, t.y. kiekvienam kėliniui reikės patikrinti ar jis nusako Hamiltono maršrutą. Šio algoritmo sudėtingumas yra *O*(*n*• *n*!). Kadangi faktorialas auga greičiau nei eksponentinė funkcija, tai šis kelias yra labai neracionalus.

Aptarkime racionalesnį Hamiltono maršrutų apskaičiavimo algoritmą, nors ir šio algoritmo veiksmų skaičius apribotas eksponente. Šis algoritmas pagrįstas “paieška gilyn su grįžimu” (angl. backtracking).

Šio metodo esmė yra labai paprasta: jei paieškos gilyn metu viršūnė *u* tampa išsemta, tai pamirštame, kad šioje viršūnėje buvome, t.y. viršūnė *u* tampa nauja. Šis metodas įgalina perrinkti visus galimus grafo maršrutus.

Žemiau pateikiame šį metodą realizuojantį algoritmą.

Tarkime, kad grafas *G*=(*V,U*) yra jungusis ir nusakytas masyvais *L* ir *lst* (žr. 2.7 paragrafą). Paiešką pradėsime iš 1-osios viršūnės. Paieškos gilyn metu aplankytas viršūnes talpinsime į masyvą *X*[1..*n*+1]. Kitaip tariant, masyvo *X* elementai apibrėš aplankytųjų viršūnių maršrutą. Jei paieškos gilyn metu masyve *X* yra *n* elementų, ir iš viršūnės x_n atėjome į 1-ąją viršūnę, tai reiškia, kad radome Hamiltono ciklą. Jei

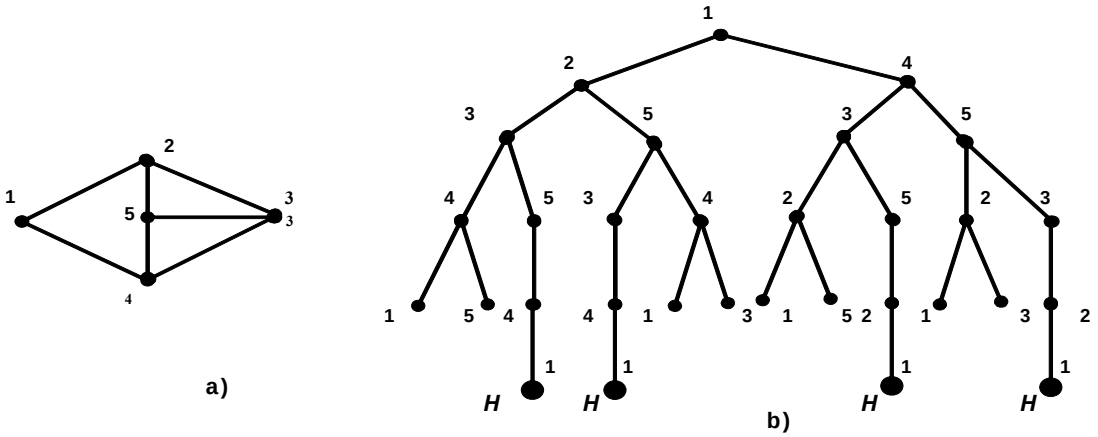
viršūnei x_n gretimų viršūnių tarpe nėra 1-osios viršūnės, tai masyvo *X* elementai nusakys Hamiltono kelią. Žemiau pateiktoje procedūroje pastaroji sąlyga netikrinama.

```
const c = 500;
type mas = array[1..c] of integer;
   matr = array[1..2,1..c] of integer;
var X, fst, prec: mas;
procedure hamil (n,m:integer; L,lst:mas; var alfa:boolean);
   { Procedūra hamil apskaičiuoja Hamiltono ciklus paieškos gilyn iš pirmosios
   viršūnės metodu, kai grafas nusakytas L ir lst masyvais. Rasti ciklai
   atspausdinami.
      Formalūs parametrai:
      n   - grafo viršūnių skaičius,
      m   - grafo briaunų (lankų) skaičius,
      L, lst - grafą nusakantys tiesioginių nuorodų masyvai. }
   var i, k, u, v   : integer;
      j  : integer;
      t,p : boolean;
      fst, prec  : mas;
      x : mas;
begin
   {Inicializacija}
      alfa:=false;
      v:=1;
      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 nenagrinėtų briaunų, incidentiškų  viršūnei k }
            begin
               t:=false;
               p:=true;
               prec[k]:=k; {k-pradinė paieškos viršūnė}
               { Nagrinėti viršūnę k }
               j:=1;
               x[j]:=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
                         j:=j+1;
                         x[j]:=u;
```

```
                    prec[u]:=k; {į viršūnė u atejome iš viršūnės k}
                    if fst[u] <= lst[u+1] then
                      {viršūnė u neišsemta}
                       k:=u
                    else {viršūnė u išsemta}
                       p:=false;
                  end
                else
                  begin
                    { write(u:3);}
                    p:=false; {viršūnė u nenauja}
                    if (j = n) and (u = 1)
                      then { Radome Hamiltono ciklą }
                           begin
                             alfa:=true;
                             x[n+1]:=1;
                            writeln('Ciklas');
                             for i:=1 to n+1 do
                                 write(x[i]: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 į viršūnę, iš kurios buvome atėję į
                      viršūnę k  }
                     begin
                       u:=prec[k];
                       prec[k]:=0;
                      fst[k]:=lst[k]+1;
                      k:=u;
                      j:=j-1;
                     end;
             end;
      end;
end;
```

Pavyzdys. Panagrinėsime 2.15.7 c) pav. grafą. Aprašytos procedūros pagalba gausime tokius Hamiltono ciklus: 1, 2, 3, 5, 4, 1; 1, 2, 5, 3, 4, 1; 1, 4, 3, 5, 2, 1; 1, 4, 5, 3, 2, 1**.** 2.15.8 a) pav. pavaizduotas 2.15.7 c) pav. grafas, 2.15.8 b) pav.

pavaizduotas paieškų medis, t.y. paieškos gilyn su grįžimu perrinkti maršrutai. Raide “H” pažymėti Hamiltono ciklai.



2.15.8 pav. Hamiltono ciklų ieškojimas

Reikia pažymėti, nors pateiktos procedūros veiksmų skaičius yra žymiai mažesnis nei pilno perrinkimo veiksmų skaičius, kuris lygus $O(n \bullet n!)$, tačiau, nepalankiausiu atveju, ir šios procedūros veiksmų skaičius auga eksponentiškai nuo viršūnių skaičiaus. Tai bus teisinga ir tuo atveju, jei algoritmą nutrauksime, radę pirmąjį Hamiltono ciklą. Jei grafas neturi Hamiltono ciklo, tai ir šiuo atveju reikės peržiūrėti visus galimus kelius.

Keliaujančio pirklio uždavinys

Kaip buvome minėję, su Hamiltono ciklu yra susijęs keliaujančio pirklio uždavinys. Priminsime šio uždavinio formulavimą.

Turime n miestų ir žinoma atstumų tarp šių miestų matrica $C = [c_{ij}] \quad i = \overline{1,n}, \quad j = \overline{1,n}$, čia c_{ij} – atstumas tarp miestų i ir j. Pirklys, išėjęs iš 1-ojo miesto, turi apeiti visus miestus po vieną kartą ir grįžti į pirmąjį miestą. Koks turi būti pirklio maršrutas, kad jo ilgis būtų trumpiausias.

Šis uždavinys taip pat priklauso NP uždavinių klasei ir visų tikslių šio uždavinio sprendimo algoritmų sudėtingumas išreiškiamas eksponente nuo kintamųjų skaičiaus n. Todėl, praktiškai sprendžiant šį uždavinį, naudojami įvairūs euristiniai algoritmai. Dažniausiai naudojamos euristikos yra:

3. artimiausio kaimyno metodas ir
4. miestų įterpimo metodas.

Artimiausio kaimyno metodas

Šis metodas pagrįstas taisykle: “jei pirklys yra mieste k, tai pirklys toliau keliaus į artimiausią miestui k neaplankytą miestą l; aišku, - $l \neq 1$, jei yra kita galimybė”. Po šio veiksmo iš matricos C pašalinama k-oji eilutė (iš miesto k pirklys daugiau neišeis) ir l-asis stulpelis (į l-ąjį miestą pirklys niekada nebeužeis).

Pavyzdys. Taikydami artimiausio kaimyno metodą, apskaičiuokime pirklio maršrutą, esant tokiai atstumų matricai:

$$C = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 10 & 7 & 4 & 3 \\ 2 & 10 & 0 & 7 & 2 & 6 \\ 3 & 7 & 7 & 0 & 5 & 4 \\ 4 & 4 & 2 & 5 & 0 & 8 \\ 5 & 3 & 6 & 4 & 8 & 0 \end{array}.$$

Iš pirmojo miesto eisime į 5-ąjį miestą, nes $c_{15} = \min\limits_{\substack{1\le j\le 5\\ j\ne 1}} (c_{1j} / c_{1j} \ne 0)$. Iš atstumų matricos pašalinę 1-ąją eilutę ir 5-ąjį stulpelį, gausime:

$$C = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 2 & 10 & 0 & 7 & 2 \\ 3 & 7 & 7 & 0 & 5 \\ 4 & 4 & 2 & 5 & 0 \\ 5 & 3 & 6 & 4 & 8 \end{array}.$$

Dabar, iš 5-ojo miesto eisime į 3-ąjį, nes $c_{53} = \min\limits_{j\ne 1}(c_{5j} / c_{5j} \ne 0)$. Iš perskaičiuotos atstumų matricos pašaliname 5-ąją eilutę ir 3-ąjį stulpelį. Gauname:

$$C = \begin{array}{c|ccc} & 1 & 2 & 4 \\ \hline 2 & 10 & 0 & 2 \\ 3 & 7 & 7 & 5 \\ 4 & 4 & 2 & 0 \end{array}.$$

Toliau iš 3-ojo miesto keliausime į 4-ąjį miestą, nes $c_{34} = \min\limits_{j\ne 1}(c_{3j} / c_{3j} \ne 0)$. Pašalinus 3-ąją eilutę ir 4-ąjį stulpelį, gausime:

$$C = \begin{array}{c|cc} & 1 & 2 \\ \hline 2 & 10 & 0 \\ 4 & 4 & 2 \end{array}.$$

Tada iš 4-ojo miesto keliausime į 2-ąjį miestą, o iš 2-ojo į 1-ąjį.

Tuo būdu, artimiausio kaimyno metodu apskaičiuotas maršrutas yra: 1, 5, 3, 4, 2, 1, ir jo ilgis yra $c_{15}+c_{53}+c_{34}+c_{42}+c_{21}$= 3+4+5+2+10=24.

Įterpimo metodas

Metodo idėja yra labai paprasta. Pirklio maršrutą konstruosime nuosekliai, pradėdami nuo ciklo v_1,v_2,v_1, ir kiekviename žingsnyje šį ciklą praplėsime įterpdami po vieną naują miestą, t.y. po antrojo žingsnio turėsime ciklą, jungiantį tris miestus, po trečiojo – keturis miestus ir t.t., kol po (n-1)-ojo žingsnio turėsime ciklą, einantį per visus miestus. Praplėsdami ciklą, t.y. įterpdami naują miestą, elgsimės taip, kad praplėsto ciklo ilgio padidėjimas būtų minimalus. Be to pradinį ciklą v_1,v_2,v_1 parinksime taip, kad jo ilgis būtų mažiausias iš visų galimų to ilgio ciklų.

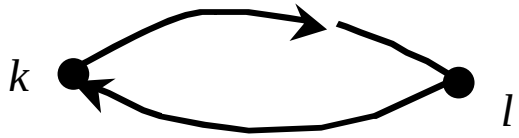
Formaliai aprašysime įterpimo metodą.

Pradinio ciklo parinkimas.

1. Apskaičiuoti $p = \min\limits_{\substack{1\le i\le n\\ 1\le j\le n\\ i\ne j}}(c_{ij} + c_{ji})$.

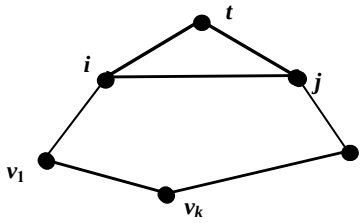
Pastaba. Jei atstumų matrica yra simetrinė, tai pakanka rasti $\min\limits_{\substack{1\le i\le n\\ 1\le j\le n\\ i\ne j}} c_{ij}$.

2. Tarkime, kad $p = c_{kl} + c_{lk}$. Tada pradinis ciklas yra k,l,k (žr. 2.15.9 pav.).



2.15.9 pav. Pirklio pradinis ciklas

Ciklo praplėtimas. Tarkime po (k-1)-ojo žingsnio turime ciklą $v_1,v_2,v_3,\ldots,i,j,\ldots,v_k,v_1$. Norėdami praplėsti ciklą, bandysime įterpti vieną iš likusių miestų tarp dviejų gretimų ciklo miestų i ir j. Iš visų galimų (i,j) porų pasirinksime tokią porą ir tokį įterpiamą miestą, kad ciklo pailgėjimas būtų minimalus, t.y. reikia apskaičiuoti $d = \min\limits_{\forall(i,j):(i,j)\in ciklui} \min\limits_{t\in V-ciklas} (c_{it} + c_{tj} - c_{ij})$ (žr. 2.15.10 pav.).



2.15.10 pav. Miesto įterpimas

Tarkime, kad mažiausia *d* reikšmė yra miestų (*i*,*j*) porai, o įterpiamas miestas - *t*. Tada gausime ciklą $v_1,v_2,\ldots, i,t,j,\ldots,v_k,v_1$.

Pavyzdys. Sukonstruokime pirklio maršrutą, esant tai pačiai atstumų matricai *C* (žr. artimiausio kaimyno metodą).

$$C=\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 10 & 7 & 4 & 3 \\ 2 & 10 & 0 & 7 & 2 & 6 \\ 3 & 7 & 7 & 0 & 5 & 4 \\ 4 & 4 & 2 & 5 & 0 & 8 \\ 5 & 3 & 6 & 4 & 8 & 0 \end{array}.$$

Pradinis ciklas bus: $2,4,2$ (žr. 2.15.11 a) pav.). Simboliu $d_{i,l,j}$ pažymėkime ciklo pailgėjimą, kai tarp miestų *i* ir *j* įterpiamas miestas *l*. Tada

$$d_{2,1,4}=c_{21}+c_{14}-c_{24}=10+4-2=12\ ,$$
$$d_{2,3,4}=c_{23}+c_{34}-c_{24}=7+5-2=10\ ,$$
$$d_{2,5,4}=c_{25}+c_{54}-c_{24}=6+8-2=12\ .$$

Kadangi matrica *C* yra simetrinė, tai, terpiant miestą tarp 4-ojo ir 2-ojo, gausime tuos pačius atstumus. Vadinasi, 3-ąjį miestą įterpsime tarp 2-ojo ir 4-ojo miestų. Gausime ciklą $2,3,4,2$ (žr. 2.15.11 b) pav.).

Apskaičiuosime

$$d_{2,1,3}=c_{21}+c_{13}-c_{23}=10+7-7=10,$$
$$d_{2,5,3}=c_{25}+c_{53}-c_{23}=6+4-7=3,$$
$$d_{3,1,4}=c_{31}+c_{14}-c_{34}=7+4-5=6,$$
$$d_{3,5,4}=c_{35}+c_{54}-c_{34}=4+8-5=7,$$
$$d_{4,1,2}=c_{41}+c_{12}-c_{42}=4+10-2=12,$$
$$d_{4,5,2}=c_{45}+c_{52}-c_{42}=8+6-2=12.$$

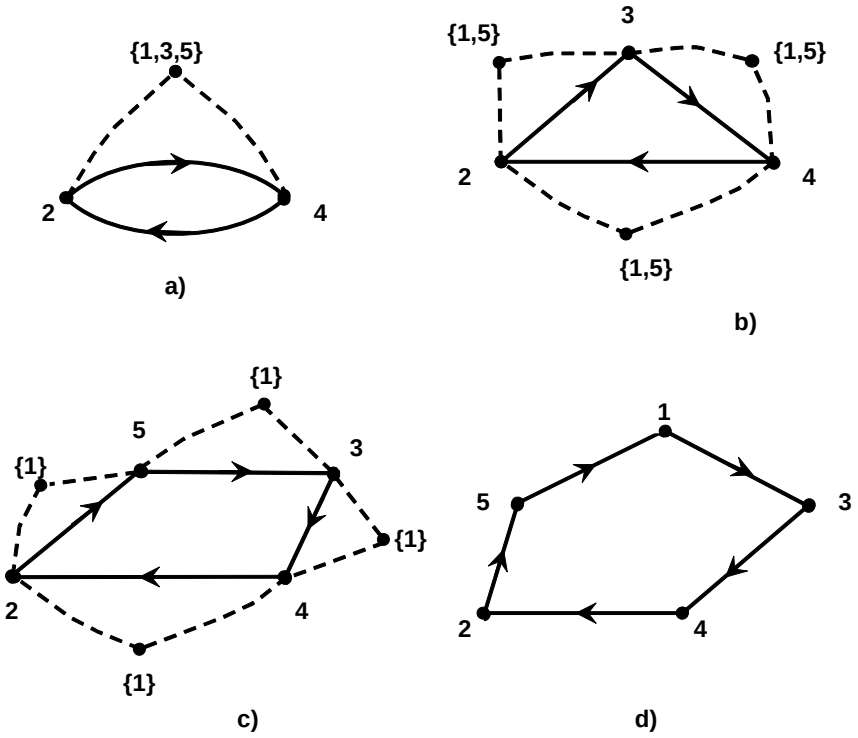
Iš šių skaičių mažiausias yra 3. Vadinasi, pailgėjęs ciklas yra: $2,5,3,4,2$ (žr. 2.15.11 c) pav.).

Apskaičiuosime

$$d_{2,1,5}=c_{21}+c_{15}-c_{25}=10+3-6=7,$$
$$d_{5,1,3}=c_{51}+c_{13}-c_{53}=3+7-4=6,$$
$$d_{3,1,4}=c_{31}+c_{14}-c_{34}=7+4-5=6,$$
$$d_{4,1,2}=c_{41}+c_{12}-c_{42}=4+10-2=12.$$

Yra du lygūs mažiausi skaičiai. Pasirenkame pirmąjį $d_{5,1,3}$. Tada pirklio maršrutas yra: $2,5,1,3,4,2$ ir jo ilgis yra $c_{25}+c_{51}+c_{13}+c_{34}+c_{42}=6+3+7+5+2=23$ (žr. 2.15.11 d) pav.). Šis pirklio maršrutas yra trumpesnis už maršrutą, gautą artimiausio

kaimyno metodu. Tas paaiškinama tuo, kad įterpimo metode buvo atliktas didesnis perrinkimas.



2.15.11 pav. Pirklio maršruto konstravimas įterpimo metodu