$\alpha + \omega$, 2002, Nr. 1, 89–96

Grafų teorijos įžanga Paskalio kalba

Rimgaudas Laucius
rimga@ktl.mii.lt

Grafų teorijos pradininku laikomas L. Oileris, 1736 metais išsprendęs gerai žinomą uždavinį apie Karaliaučiaus tiltus [1], [5].

Šiandien remiantis grafų teorija sprendžiama begalė gyvenimiškų uždavinių (paprasčiausi pavyzdžiai: vandentiekio, elektros tinklų planavimas, srautų juose valdymas). Daugeliui grafų teorijos uždavinių spręsti reikia ilgų algoritmų. Todėl plačiau grafų teorija imta taikyti tik atsiradus kompiuteriams.

Straipsnyje pateikiamos pradinės šios teorijos sąvokos ir algoritmai Paskalio programavimo kalba.

Iš pradžių tai pasirodys sudėtinga, bet
iš pradžių viskas sudėtinga...
Musaši

Grafo sąvoka

Norintiems geriau suprasti grafų teoriją ir giliau ją paanalizuoti rekomenduojame pasinaudoti papildoma literatūra, kuri yra nurodyta straipsnio pabaigoje. Reikia pastebėti, kad „Alfa plius omega“ žurnale šia tema taip pat jau buvo skelbta straipsnių [5], [6]. Čia pateiksime tik keletą pagrindinių grafų teorijos sąvokų, kurių mums vėliau prireiks.

Grafas — struktūra, sudaryta iš taškų, vadinamų *viršūnėmis*, ir atkarpų, vadinamų *briaunomis*.

Viršūnių aibę žymėsime V, briaunų — B. Kiekviena briauna $b \in B$ jungia dvi grafo viršūnes, pavyzdžiui, u, v. Todėl briauną galima žymėti viršūnių, kurias ji jungia, pora, pavyzdžiui, $b = (u, v)$.

Grafo išraiška Paskalio kalba

Pastaba. Algoritmams užrašyti mes vartosime Free Paskalio programavimo kalbą, kuri turi platesnes galimybes nei Turbo Paskalis ir iš dalies yra suderinama su *Delphi Object Pascal*. Be to, Free Paskalio sistema platinama nemokamai, todėl yra prieinama visiems [11].

Grafo apibrėžimą Paskaliu galime išreikšti taip:

```
type
  TViršūnė = string;
  ĮBriauna = record
    u, v:  TViršūnė;
  end;
  ĮGrafas = record
    MViršūnės:   array of TViršūnė;
    MBriaunos:   array of ĮBriauna;
  end;
```

Bet kokio grafo viršūnes galima sunumeruoti (1 pav.). Tai kartais palengvina užduoties sprendimą (tada TViršūnė būtų sveikasis tipas), ypač jei viršūnių pavadinimai sprendimo metu nėra svarbūs.

1 pav.

Ankstesnė išraiška tik aprašo grafą, bet nesuteikia galimybių operuoti jo duomenimis. Dažnai sprendžiant uždavinius tenka šalinti arba įterpti į grafą viršūnes ar briaunas. Todėl šią išraišką šiek tiek praplėsime.

```
KGrafas = class
  protected
    function SkaitykV(Id: integer): TViršūnė;
    procedure Įrašyk(Id: integer; Viršūnė: TViršūnė);
    function SkaitykB(Id: integer): ĮBriauna;
    procedure Įrašyk(Id: integer; Briauna: ĮBriauna);
  public
    lViršūnės: TList;
    lBriaunos: TList;
    procedure Pridėti(Viršūnė: TViršūnė);
    procedure Pridėti(Briauna: ĮBriauna);
    procedure Pašalinti(Viršūnė: TViršūnė);
    procedure Pašalinti(Briauna: ĮBriauna);
    procedure Įterpti(Indeksas: integer; Viršūnė: TViršūnė);
    procedure Įterpti(Indeksas: integer; Briauna: ĮBriauna);
    constructor Create;
    destructor Destroy;
    property Viršūnės[Id: integer]: TViršūnė read SkaitykV write Įrašyk;
    property Briaunos[Id: integer]: ĮBriauna read SkaitykB write Įrašyk;
  end;
implementation
constructor KGrafas.Create;
begin
  lViršūnės:= TList.Create;
  lBriaunos:= TList.Create;
end;
destructor KGrafas.Destroy;
begin
  lViršūnės.Free;
  lBriaunos.Free
end;
procedure KGrafas.Įrašyk(Id:integer; Viršūnė: TViršūnė);
begin
  Try
    lViršūnės[Id]:= @Viršūnė;
  Except
    on Exception do Pranešimas('Klaida priskiriant', ptKlaida);
  end;
end;
```

```
procedure KGrafas.SkaitykV(Id:integer):  TViršūnė;
var
  pViršūnė:^TViršūnė;
begin
  Try
    pViršūnė:= lViršūnės[Id];
    Result := pViršūnė^;
  Except
    on Exception do Pranešimas('Klaida skaitant', ptKlaida);
  end;
end;
procedure KGrafas.Pridėk(Viršūnė:  TViršūnė);
begin
  lViršūnės.Add(@Viršūnė);
end;
...
```

Panašiai aprašę ir likusias procedūras, gautume tipą KGrafas, į kurį jau galima įterpti briaunas arba viršūnes bei jas šalinti. Aprašant viršūnės šalinimo procedūrą, reikia nepamiršti pašalinti ir briaunų, kuriomis ji sujungta su kitomis viršūnėmis. Tačiau nesurūšiuotame grafe tai padaryti yra sunkiau. Kad galėtume tokį grafą surūšiuoti, reikia aprašyti jo elementų palyginimo funkciją.

Tai galime padaryti tokiu būdu:

```
function Palygink_Briaunas_U(Briauna1, Briauna2:  Pointer):  Integer;
begin
  Result:= CompareText(Briauna1^.u, Briauna2^.u);
end;
```

Tada atlikus kreipinį

```
  Grafas.Sort(Palygink_Briaunas_U);
```

grafo briaunos bus surūšiuotos pagal viršūnės u pavadinimą.

Alternatyvūs grafų užrašymo būdai

Deja, nagrinėta grafo išraiška yra gana sudėtinga. Norint ją padaryti universalią įvairiems uždaviniams spręsti, dar reiktų įdėti nemažai darbo, todėl pateikiame alternatyvių išraiškos būdų.

Gretimų viršūnių masyvas. Šis būdas yra tinkamas sunumeruotiems grafams. Užrašant grafą sudaromas dvimatis masyvas, kuriame kiekvienai viršūnei skiriama viena eilutė ir vienas stulpelis. Jei grafe yra n viršūnių, tai masyvas turi n eilučių ir n stulpelių.

```
type
  aGrafas = array [1..n, 1..n] of boolean; //n — viršūnių skaičius
```

Masyvo reikšmės randamos pagal formulę:

`Grafas [u, v] = true`, jei viršūnes u ir v jungia briauna, `false` — priešingu atveju.

Pateikto 1 paveiksle grafo išraišką atitiktų tokia lentelė:

	1	2	3	4
1	false	true	true	false
2	true	false	true	false
3	true	true	false	true
4	false	false	true	false

Šis išraiškos būdas geras tuo, kad iš karto galima sužinoti, ar dvi viršūnes jungia briauna. Jis ypač patogus, kai grafas yra *tankus* — turi daug briaunų arba yra nedidelis.

Porų sąrašas. Jei briaunų skaičius nėra didelis, grafą galima išreikšti porų, atitinkančių grafo briaunas, sąrašu. Turintį n briaunų grafą Paskalio kalba galima užrašyti taip:

```
type
  aGrafas = array [1..2, 1..n] of TViršūnė; //n — viršūnių skaičius
```

Čia grafo viršūnes `Grafas [1, i]`, `Grafas [2, i]` jungia briauna, $\forall i \in [1, \ldots, n]$.

Pateikto 1 paveiksle grafo išraišką atitiktų tokia lentelė:

Grafas $[1, i]$	Grafas $[2, i]$
1	2
1	3
2	3
3	4

Šis išraiškos būdas nepatogus, kai reikia nustatyti, su kuriomis viršūnėmis sujungta duotoji viršūnė. Situacija pagerėtų, jei sąrašo elementus surikiuotume. Tuomet galima būtų taikyti dvejetainę paiešką.

Dinaminis gretimų viršūnių sąrašas. Dažnai grafą patogu vaizduoti dinaminėmis duomenų struktūromis. Kiekvienai grafo viršūnei u sudaromas viršūnių, su kuriomis ji sujungta, sąrašas. Kiekvieną šio sąrašo elementą sudaro informacinė dalis (viršūnės, su kuria ji sujungta, numeris) ir rodyklė į tolesnį sąrašo elementą.

Čia pateikiamas vienas iš būdų, kaip tokią duomenų struktūrą galima užrašyti Paskalio kalba:

```
type
  rElementas = ^ĮElementas;
  ĮElementas = record
    iViršūnė:  1..n; //n — viršūnių skaičius
    rSekantis:  ĮElementas;
  end;
  aGrafas = array [1..n] of ĮElementas;
```

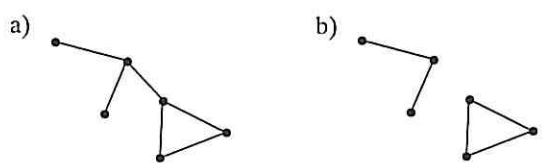
Ši išraiška patogi, kai grafas yra *retas*, t. y. briaunų skaičius daug mažesnis už viršūnių skaičiaus kvadratą. Nepatogi, kai reikia nustatyti, ar dvi viršūnės sujungtos briauna.

Daugiau sąvokų

Pereinant prie algoritmų su grafais mums reikės daugiau šios teorijos sąvokų, todėl jas čia ir paminėsime.

Grafas, kurio briaunos neturi krypties, vadinamas *neorientuotuoju grafu*. Kaip ir iki šiol, nagrinėsime tik neorientuotus grafus.

Grafas yra vadinamas *jungiuoju*, jei iš bet kurios vienos grafo viršūnės galima pasiekti bet kurią kitą. Jei to padaryti negalima, grafas vadinamas *nejungiuoju*.

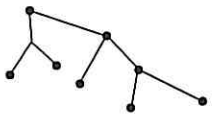


2 pav. Grafai: a) jungusis, b) nejungusis

Gretimų viršūnių seka, einant iš vienos viršūnės į kitą, vadinama *keliu*. Šio kelio ilgis yra lygus jį sudarančių viršūnių skaičiui.

Kelias vadinamas *ciklu*, jei jis prasideda ir baigiasi toje pačioje viršūnėje. Viršūnės *laipsniu* vadinamas iš jos išeinančių briaunų skaičius (kilpa padidina laipsnį dviem).

Jungusis, neturintis ciklų grafas vadinamas *medžiu*.



3 pav. Medžio tipo grafas

Medis, kurio viena viršūnė išskirta iš kitų, vadinamas *šakniniu*, o išskirtoji viršūnė — *šaknimi*. Sakykime, viršūnė u yra kelyje iš šaknies į viršūnę v. Tokiu atveju viršūnė u yra viršūnės v *protėvis*, o viršūnė v — viršūnės u *palikuonis*. Šaknis yra visų medžio viršūnių protėvis ir nėra nė vienos viršūnės palikuonis.

Eilė. Algoritmuose su grafais dažnai tenka pasinaudoti eile. Tai skaičių eilutė, kurią paprastai galima realizuoti kaip vienmatį masyvą. Kaip pavyzdį panagrinėsime FIFO (*first in, first out*) tipo eilę. FIFO reiškia, kad pirmasis atsistojęs į eilę pirmasis iš jos ir pašalinamas. Dirbant su eilute dažnai naudojamos sąvokos *galva* ir *uodega*. Galva rodo anksčiausiai į eilę atsistojusį elementą, uodega — eilės pabaigą, t. y. tuščią vietą masyve, kur bus įrašytas naujai į eilę įtraukiamas elementas. Eilės ilgį žymėsime raide N.

Pavyzdžiui:

	1	2	3	4	5	6	7	8	9	10	11
Eilė				26	9	15	8	41	3		

```
galva = 4;
uodega = 10;
N = 11.
```

Eilėje yra 6 elementai, stovintys 4–9 pozicijose (baltos spalvos langeliuose).

Darbui su eile reikia dviejų procedūrų. Vienos — įtraukti naujiems elementams į eilės pabaigą, kitos — pašalinti iš eilės pirmajam elementui.

Šias procedūras užrašysime Paskalio kalba.

```
procedure Įtraukti_Į_Eilę(x);
{ įtraukia į eilės pabaigą elementą x }
begin
  Eilė[uodega]:= x;
  if uodega = N
    then uodega := 1
    else uodega := uodega + 1
end;

procedure Pašalinti_Iš_Eilės;
{ pašalina iš eilės pirmąjį elementą }
begin
  if galva = N
    then galva := 1
    else galva := galva + 1
end;
```

Procedūrose netikrinama, ar duomenys telpa į leistinus rėžius, t. y. ar nebandoma įrašyti į eilę daugiau nei N elementų.

$\alpha + \omega$

Atlikus kreipinius

```
Įtraukti_Į_eilę(21),
Įtraukti_Į_eilę(3),
Įtraukti_Į_eilę(8),
Įtraukti_Į_eilę(18),
```

eilė atrodo taip:

	1	2	3	4	5	6	7	8	9	10	11
Eilė	8	18		26	9	15	8	41	3	21	3

```
galva = 4;
uodega = 3;
```

o atlikus kreipinį `Pašalinti_iš_eilės`:

	1	2	3	4	5	6	7	8	9	10	11
Eilė	8	18			9	15	8	41	3	21	3

```
galva = 5;
uodega = 3.
```

Trumpiausio kelio paieškos į plotį algoritmas

Formali uždavinio sąlyga tokia: reikia rasti trumpiausią kelią nuo grafo viršūnės a iki viršūnės f.

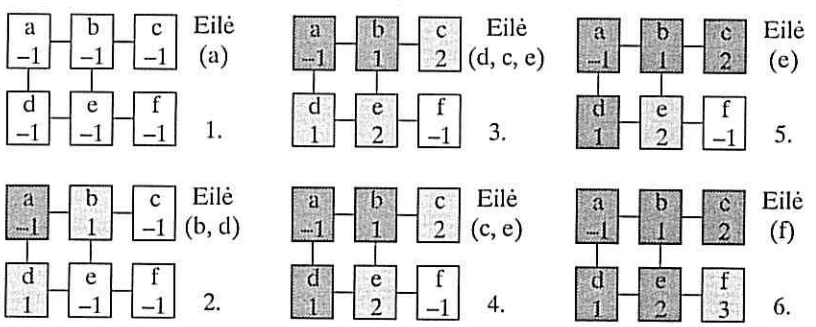
Grafo viršūnes žymėsime kvadratais ir mažosiomis raidėmis (4 pav.). Kvadratuose taip pat įrašysime rastą trumpiausią atstumą nuo viršūnės a. Iš pradžių dar nežinome, koks yra trumpiausias atstumas nuo a iki likusių viršūnių, todėl į visus kvadratus įrašome sutartą skaičių -1. Trumpiausio kelio pradėsime ieškoti nuo viršūnės a, todėl ją įrašome į eilę (4 pav., 1.).

Viršūnė a sujungta su viršūnėmis b ir d. Trumpiausias kelias iki jų yra lygus 1. Į langelius, žyminčius šias viršūnes, tai ir įrašome. Kadangi jau išnagrinėjome visas briaunas, einančias iš viršūnės a, ją išmetame iš eilės ir nuspalviname juodai. Trumpiausio kelio į f dabar reikia ieškoti iš viršūnės b arba d. Todėl abi šias viršūnes įtraukiame į eilę ir nuspalviname pilkai (4 pav., 2.).

Toliau išnagrinėjame briaunas, einančias iš viršūnės b (4 pav., 3.). Po to išnagrinėjame briaunas, einančias iš viršūnės d. Jos jungia ją su viršūnėmis a ir e, kurios jau nuspalvintos. Todėl viršūnę d tereikia pašalinti iš eilės ir nuspalvinti juodai (4 pav., 4.).

Taip tęsiame tol, kol išnagrinėjame visas viršūnes, pasiekiamas iš viršūnės a. Algoritmui baigus darbą, visos viršūnės, kurias galima pasiekti iš a, nuspalvinamos juodai. Jei kurių nors viršūnių pasiekti negalima, jos lieka nenuspalvintos.

Paveiksle pateikti 6 nagrinėjamo algoritmo žingsniai; paskutiniu — 7-uoju yra nuspalvinama juodai ir pašalinama iš eilės viršūnė f.



4 pav.

Nagrinėtame uždavinyje norėjome rasti trumpiausią kelią nuo viršūnės a iki f. Tačiau radome daugiau, negu ieškojome.

Apskritai paieškos į plotį grafe algoritmą, kai ieškome trumpiausio kelio nuo viršūnės s iki kurios nors kitos viršūnės, galima nusakyti taip:

1) randamos visos viršūnės, kurias galima pasiekti iš viršūnės s,
2) randami trumpiausi keliai nuo s iki visų iš jos pasiekiamų viršūnių,
3) sukonstruojamas medis, kurio šaknis yra viršūnė s ir kuriame yra visos iš s pasiekiamos viršūnės.

Dabar pademonstruosime, kaip šį algoritmą galima užrašyti Paskalio kalba.

Pakomentuosime algoritme naudojamus kintamuosius:

`Spalva[v]` — viršūnės v spalva,
`Atstumas[v]` — trumpiausias atstumas nuo viršūnės s iki viršūnės v,
`Tėvas[v]` — viršūnės v konstruojamame medyje tėvas,
`Eilė` — atrastų, bet dar nenagrinėtų viršūnių eilė.

```
procedure Paieška_Į_Plotį;
var
  u, i:  integer;
begin
  Įtraukti_Į_Eilę(Startas); //įtraukiame į eilę pradžios viršūnę
  while (Galva - Uodega) <> 0 do //kol eilėje yra elementų
  begin
    u := Eilė[Galva];
    for i := 1 to n do //n — viršūnių skaičius
     if Grafas[u, i] then //grafą aprašėme kaip gretimų viršūnių masyvą
      if Spalva[i] = balta then
      begin
        Spalva[i]:= pilka;
        Atstumas[i]:= Atstumas[u]+ 1;
        Tėvas[i]:= u;
        Įtraukti_Į_Eilę(i);
      end;
    Pašalinti_Iš_Eilės;
    Spalva[u]:= juoda;
  end;
end;
```

Norint išspausdinti patį kelią nuo pradinės viršūnės s iki viršūnės v, galima naudoti tokią procedūrą:

```
procedure Spausdinti_Kelią(s, v:  integer);
begin
  if v = s then
   WriteLn(s)
  else
   if Tėvas[v] = -1 then
    WriteLn('Kelias neegzistuoja')
   else
    begin
      Spausdinti_Kelią(s, Tėvas[v]);
      WriteLn(v);
    end;
end;
```

$\alpha + \omega$

1. P. Tannenbaum, R. Arnold, *Kelionės į šiuolaikinę matematiką*, TEV, Vilnius, 1995.
2. B. Bollob, *Graph Theory*, Springer, 1979.
3. B. Bollob, *Modern Graph Theory*, Springer, 1998.
4. R. Diestel, *Graph Theory*, Springer, 1997.
5. L. Maliaukienė, Grafų teorijos įžanga, *Alfa plius omega*, **1**, 28–35, 2000.
6. E. Manstavičius, Medžiai ir miškai — ne miškininkystė, bet kombinatorika, *Alfa plius omega*, **2**, 39–48, 2000.
7. R. Wilson, *Introduction to Graph Theory*, Longman, 1985.
8. L. Volkmann, *Fundamente der Graphentheorie*, Springer, 1996.
9. C. Berge, *Graphs*, North-Holland, 1985.
10. V. N. Sačkov, *Įvadas į kombinatorinius diskrečios matematikos metodus*, Nauka, Maskva, 1982.
11. Free Paskalio interneto puslapis *http://www.freepascal.org*. Informacijos apie Free Paskalį taip pat galite rasti lietuviškame interneto puslapyje *http://aldona.mii.lt/pms/fps*
12. Jaunųjų matematikų mokyklos interneto puslapis *http://www.mif.vu.lt/ljmm/kursa/sept/septuzd.html*

ALFA PLIUS OMEGA, 2002, Nr. 1(14)

2002 05 02. 12 sp. l. Tiražas 1000 egz. Užs. Nr. 1372
Leidykla TEV, Akademijos g. 4, LT-2600 Vilnius
Spausdino UAB „Mokslo aidai"
A. Goštauto g. 12, LT-2600 Vilnius
Viršelį spausdino P. Kalibato IĮ „Petro ofsetas"
Žalgirio g. 90, LT-2600 Vilnius