TRYLIKTASIS SKYRIUS

Algoritmai su grafais

Dalis svarbiausių grafų teorijos sąvokų, algoritmų, uždavinių klasių, kurių sprendimui taikoma grafų teorija, aprašyti P. Tannenbaumo ir R. Arnoldo knygoje „Kelionės į šiuolaikinę matematiką“ (Vilnius: TEV, 1995). Grafai nagrinėjami šios knygos 5–8 skyriuose. Atkreipkite dėmesį į svarbiausias grafų teorijos sąvokas (133–136 psl., 191–196 psl., 235–237 psl.), Oilerio ciklus (136–144 psl.), Hamiltono ciklus (161–170 psl.), minimalaus tinklo uždavinius (189–201 psl.).

Be to, naudinga išsinagrinėti 41, 66, 69 uždavinius iš V. Dagienės ir J. Skūpienės knygos „Moksleivių informatikos olimpiadų uždaviniai: I–VII olimpiados“ (Vilnius: TEV, 1999) bei 102, 138 uždavinius iš tų pačių autorių knygos „Moksleivių informatikos olimpiadų uždaviniai: VIII–X olimpiados“ (Vilnius: TEV, 2001).

Grafo vaizdavimas kompiuteryje

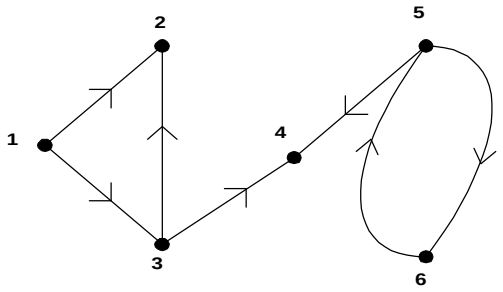
Tarkime, grafas G turi n viršūnių ir m lankų (arba briaunų). Galime parinkti įvairias duomenų struktūras grafui pavaizduoti.

A. Sudarykime dvimatę lentelę (t. y. dvimatį masyvą) A_{nm}, turinčią n eilučių ir m stulpelių, kiekvienai viršūnei skiriama viena eilutė, o kiekvienam lankui (briaunai) – vienas stulpelis.

Jei grafas orientuotas, lentelės elementai apibrėžiami taip:

$$a_{ij} = \begin{cases} -1, \ jei\ iš\ viršūnės\ i\ išeina\ lankas\ j, \\ 1, \ jei\ į\ viršūnę\ i\ ateina\ lankas\ j, \\ 2, \ jei\ iš\ viršūnės\ i\ išeina\ kilpa, t.y. i = j, \\ 0, \ kitais\ atvejais. \end{cases}$$

Pavyzdžiui, čia nubraižytas grafas bus vaizduojamas tokia lentele:



	<1, 2>	<1, 3>	<3, 2>	<3, 4>	<5, 4>	<5, 6>	<6, 5>
1	–1	–1	0	0	0	0	0
2	1	0	1	0	0	0	0
3	0	1	–1	–1	0	0	0
4	0	0	0	1	1	0	0
5	0	0	0	0	–1	–1	1
6	0	0	0	0	0	–1	1

Jei grafas neorientuotas:

$$a_{ij} = \begin{cases} 1, \ jei\ iš\ viršūnės\ i\ išeina\ briauna\ j, \\ 0, \ priešingu\ atveju. \end{cases}$$

Tai yra vienas iš mažiausiai ekonomiškų grafo vaizdavimo būdų. Jis užima $n \times m$ atminties ląstelių. Jei briaunų skaičius yra žymiai didesnis už viršūnių skaičių, toks būdas užima daug atminties. Be to, dauguma ląstelių užpildytos nuliais.

B. Geresnis grafo vaizdavimo būdas yra kvadratinė lentelė (pavadinkime ją B_{nn}), kurios elementai apibrėžiami taip:

$$b_{ij} = \begin{cases} 1, \ jei\ egzistuoja\ lankas, einantis\ iš\ viršūnės\ i\ į\ viršūnę\ j, \\ 0, \ priešingu\ atveju. \end{cases}$$

Jei grafas neorientuotas, ši lentelė visuomet simetrinė, t. y. $b_{ij} = b_{ji}$ kiekvienam $i, j \in [1, ..., n]$.

Anksčiau pavaizduotą grafą atitiktų dešinėje pateikta lentelė.

Bendru atveju b_{ij} turėtų būti lygus lankų, einančių iš viršūnės i į j, skaičiui, nes iš vienos viršūnės į kitą gali eiti keli lankai.

	1	2	3	4	5	6
1	0	1	1	0	0	0
2	0	0	0	0	0	0
3	0	1	0	1	0	0
4	0	0	0	0	0	0
5	0	0	0	1	0	1
6	0	0	0	0	1	0

Šis vaizdavimo būdas patogus tuo, kad iš karto galima sužinoti, ar dvi grafo viršūnes jungia lankas (briauna).

C. Jei grafo lankų (briaunų) skaičius nėra didelis, jį patogu vaizduoti porų, atitinkančių grafo lankus (briaunas), sąrašu.

Duotąjį grafą atitiktų sąrašas dešinėje.

Kai lankų skaičius nedidelis, jis atminties užima nedaug. Tačiau nepatogu, kai reikia nustatyti, su kuriomis viršūnėmis sujungta duota viršūnė.

Situacija pagerėtų, jei sąrašo elementus surikiuotume. Tuomet galima būtų taikyti dvejetainę paiešką.

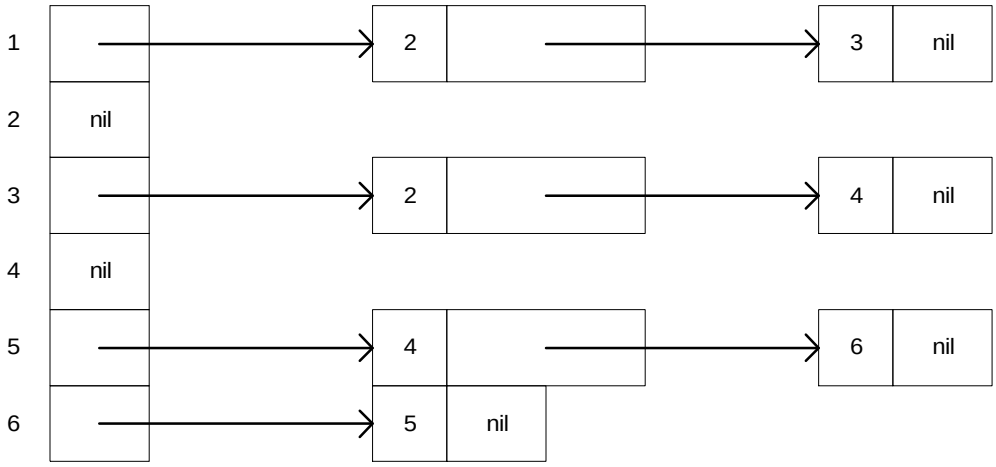
1	3
3	2
3	4
5	4
1	2
6	5
5	6

D. Daugeliu atvejų grafą patogiausia vaizduoti dinaminėmis duomenų struktūromis. Apie jas išsamiau galite paskaityti V. Tumasonio knygoje „Paskalis ir Turbo Paskalis 7.0“ (Vilnius: Ūkas, 1993; p. 134–150) ar kuriame kitame Paskalio kalbos žinyne.

Kiekvienai grafo viršūnei v sudaromas viršūnių, į kurias eina lankai (briaunos) iš viršūnės v, sąrašas.

Kiekvieną šio sąrašo elementą sudarys informacinė dalis (viršūnės, į kurią eina lankas, numeris) ir rodyklė į tolesnį sąrašo elementą.

Anksčiau pateiktas grafas būtų pavaizduotas taip:



Šią duomenų struktūrą galime aprašyti taip:

```
const N = ...; {viršūnių skaičius}
type v_sąrašas = ^elem;
     elem = record
               viršūnė: 1..N;
               tolesnis : v_sąrašas
            end;
     grafas = array [1..N] of v_sąrašas;
```

Jei grafas neorientuotas, kiekviena briauna $\langle u, v \rangle$ sąraše sutinkama du kartus. Vieną kartą – sąraše, atitinkančiame viršūnę v, antrą kartą – sąraše, atitinkančiame viršūnę u.

Jei grafą reikia modifikuoti, lankai (briaunos) įjungiami arba pašalinami iš sąrašo.

Dažnai patogu, kai kiekvienas sąrašo elementas turi rodyklę ne tik į tolesnį, bet ir į prieš tai ėjusį sąrašo elementą. Tai labai supaprastina, pavyzdžiui, briaunos pašalinimą iš neorientuoto grafo.

E. Panašiai kaip D atveju, jei grafo lankų (briaunų) skaičius nėra didelis, jį galima vaizduoti masyvu, kurio elementai yra viršūnių, į kurias eina lankai (briaunos) iš viršūnės *v*, aibės.

Viršunė	1	2	3	4	5	6
Viršūnių aibė	{2, 3}	{}	{2, 4}	{}	{4, 6}	{5}

Šį masyvą galima aprašyti taip:

```
const N = ...; {viršūnių skaičius}
type v_aibė = set of 1..N;
     grafas = array [1..N] of v_aibė;
```

Šis būdas patogus, jei grafą reikia modifikuoti – įjungti arba pašalinti lankus (briaunas).

Pavyzdžiui, iš viršūnės 1 išvesime briaunas į viršūnes 4 ir 5:

```
var graf: grafas;
...
graf[1] := graf[1] + [4, 5];
```

Paieška gilyn

Paieška gilyn (anglų k. *Depth-First Search*, *DFS*) – vienas iš grafo apėjimo metodų. Jo esmė: jei esama kažkurioje grafo viršūnėje, ji pažymima kaip aplankyta ir rekursyviai aplankomos visos neaplankytos kaimyninės viršūnės. Algoritmas panašus į grįžimo metodą, bet yra efektyvesnis: apeinamos visos viršūnės, pasiekiamos iš pradinės, bet kiekviena viršūnė aplankoma ne daugiau kaip vieną kartą.

Pateikiame pavyzdį, kuriame grafas apibrėžtas kaimynystės matrica, aplankytoms viršūnėms žymėti naudojamas loginis masyvas (visų elementų pradinė reikšmė `false`):

```
const MAXN = ...; {maksimalus grafo viršūnių skaičius}
type atkarpa = 1..MAXN;
     grafas = record
                n: integer; {viršūnių skaičius}
                briauna: array [atkarpa, atkarpa] of boolean;
              end;
var G: grafas;
    aplankyta: array [atkarpa] of boolean;
procedure dfs (var G: grafas; v: atkarpas);
var i: atkarpa;
begin
  aplankyta[v] := true;
  for i := 1 to G.n do
    if G[v, i] and not aplankyta[i] then
      dfs(i)
end;
```

Paieška gilyn dažnai naudojama grafo jungumui patikrinti: grafas yra jungus tada ir tik tada, jei įvykdžius paiešką gilyn iš bet kurios jo viršūnės, bus aplankytos visos viršūnės.

Paieška platyn

Paieškos platyn (anglų k. *Breadth-First Search*, *BFS*) algoritmas vykdomas taip: iš pradinės viršūnės aplankomos visos viena briauna pasiekiamos viršūnės (kaimynės), po to – pasiekiamos dvejomis briaunomis (kaimynių kaimynės) ir t. t. Paieškos platyn algoritme svarbi viršūnių lankymo tvarka. Tai užtikrins eilės duomenų struktūra:

```
type eilė = record
              duom: array [atkarpa] of atkarpa;
              pradžia, pabaiga: atkarpa
```

```
end;
```

Pirmiausia į eilę įrašoma pradinė viršūnė; kol eilė netuščia, iš jos pradžios imama viršūnė ir visos neaplankytos jos kaimynės įrašomos į eilės galą. Tokiu būdu eilėje pirma atsiduria viršūnės, pasiekiamos viena briauna, vėliau – dviem ir t. t.

Būtina pabrėžti, kad elementai imami tik iš eilės pradžios ir dedami tik į eilės galą. Taip pat, jei paieška bus vykdoma ne vieną kartą, eilė turi būti išvaloma. Todėl reikia aprašyti dvi procedūras ir dvi funkcijas.

Pateikiame antraštes, o reikiamus veiksmus paliekame apibrėžti patiems:

```
procedure įdėk(var eil: eilė; x: atkarpa);
procedure išvalyk(eil: eilė);
function išimk(var eil: eilė): atkarpa;
function tuščia(var eil: eilė): boolean;
```

Naudojanti paieškos platyn metodu randamas skaičius briaunų tarp pradinės ir kiekvienos nagrinėjamos viršūnės. Jeigu grafas yra besvoris (visos briaunos lygiavertės), tai bus trumpiausias kelias tarp jų.

Uždaviniai

1. Jeigu jungiajame grafe yra tokia briauna, kurią ištrynus *grafas* tampa nejungiuoju, ji vadinama *tiltu*.

Parašykite dviejų parametrų loginę funkciją, nustatančia, ar viršūnes *i* ir *j* jungianti briauna yra tiltas.

2. *Parašykite algoritmą (programą)*, kuris nustatytų, iš kokių komponenčių (atskirų dalių) sudarytas grafas. Komponentę nusako viršūnių, priklausančių jai, sąrašas didėjimo tvarka.

Pradiniai duomenys ir *rezultatai* saugomi atskiruose failuose. Kokiais formatais išdėstomi duomenys failuose JPM klausytojas parenka pats.

3. *Parašykite loginę funkciją*

```
function ArMedis (var G: grafas): boolean,
```

nustatančią, ar besvoris jungus grafas yra *medis*. Medis – jungus beciklis grafas.

4. Laikykime, kad atstumas tarp dviejų besvorio grafo viršūnių yra minimalus jas jungiantis kelias. *Parašykite programą*, randančią viršūnių, atstumas tarp kurių yra ilgiausias, porą. Jei yra kelios tokios poros, galima pateikti bet kurią.

Pradiniai duomenys ir *rezultatai* saugomi atskiruose failuose. Kokiais formatais išdėstomi duomenys failuose JPM klausytojas parenka pats.

Šio skyriaus 1–4 uždavinių sprendimai turi būti pateikti JPM interneto svetainėje (http://ims.mii.lt/jpm/) iki 2009 m. balandžio 5 d. 24 val.

Failo vardas turi būti *?????13.** (čia * – *doc, txt, odt arba rtf*). Vietoj klaustukų įrašykite pirmąsias penkias savo pavardės raides (be diakritinių ženklų).
Jei kartais būtų pavardžių, trumpesnių negu penkios raidės, trūkstami simboliai keičiami pabraukimo brūkšniais „_“.

Elektroninio pašto adresas JPM antrosios dalies klausytojams: jpm.2kursas@gmail.com.

5. Duota *n* miestų ir *m* šiuos miestus jungiančių kelių. Kiekvienas miestas gali būti sujungtas su bet kuriuo iš kitų miestų. Du miestus gali jungti keletas kelių. Gali egzistuoti keliai, vedantys iš to paties miesto į tą patį. Visuose keliuose eismas abipusis. Keliautojas panoro apvažiuoti visus kelius, bet tik po vieną kartą.

Parašykite: **a**) *šio uždavinio sprendimo idėją* ir

b) *programą*, kuri nustatytų, ar pavyks keliautojui tai padaryti.

Pradiniai duomenys pateikti faile *duom.txt*. Pirmoje eilutėje atskirti tarpu įrašyti skaičiai n ir m ($1 \le n \le 50$, $1 \le m \le 3000$). Kiekvienoje iš likusių m eilučių įrašyti duomenys apie vieną kelią: nurodomi miestai, kuriuos jis jungia.

Rezultatą – žodį PAVYKS arba NEPAVYKS – spausdinkite į failą *rez.txt*.

Pradinių duomenų ir rezultatų pavyzdys

Pradiniai duomenys	*Rezultatas*
4 5 1 2 2 3 3 4 4 1 1 3	PAVYKS

6. N mokinių nori dalyvauti varžybose. Juos reikia suskirstyti į kaip galima daugiau komandų. Komandose turi būti tik tarpusavyje galintys susipažinti mokiniai, kurie nieko nepažįsta kitose komandose. Vienas komandos narys gali turėti kelis draugus, kurie tarpusavyje nėra pažįstami, bet gali susipažinti. *Pavyzdžiui*, A pažįsta B ir C, nors B ir C nėra pažįstami. Tokiu atveju A, B ir C bus vienoje komandoje. Tokiu būdu vienoje komandoje gali būti labai daug mokinių.

Parašykite *programą*, kuri nustatytų, į kiek daugiausiai komandų galima suskirstyti mokinius, kad kiekvienas asmuo būtų tik vienoje komandoje.

Pradiniai duomenys surašyti faile *duom.txt*. Pradinius duomenis sudaro pažįstamų mokinių poros. Asmenys koduojami skaičiais iš intervalo [1, 100]. Pirmoje eilutėje nurodytas pažįstamų porų skaičius. Kiekviena tolesnė eilutė skiriama vienai pažįstamų mokinių porai. Joje du skaičiai – mokinių kodai.

Asmenų sąrašas atskirai nepateikiamas. Laikoma, kad kiekvienas jų paminėtas porų sąraše.

Rezultatas – sveikasis skaičius, reiškiantis komandų skaičių, spausdinamas į failą *rez.txt*.

Pradinių duomenų ir rezultatų pavyzdys

Duomenys	*Rezultatas*
5 5 10 13 12 18 10 7 40 7 2	3

7. Į tarptautinę konferenciją turėjo atvykti N asmenų. Asmenys įvardyti skaičiais nuo 1 iki N. Yra žinoma, kurie asmenys su kuriais buvo pažįstami iki renginio pradžios. Du asmenys renginio metu būtinai susipažins, jei jie turi arba renginio metu įsigijo bendrą pažįstamą.

Renginio organizatoriai, gerai žinodami, kas ką pažįsta, buvo tikri, kad renginio metu visi asmenys susipažins. Tačiau vienas asmuo į renginį neatvyko ir todėl visiems susipažinti nepavyko. Reikia nustatyti, kas yra šis asmuo.

Parašykite: **a**) *šio uždavinio sprendimo idėją* ir

b) *programą*, kuri nustatytų, kuris asmuo neatvyko į renginį.

Pradiniai duomenys pateikti faile *duom.txt*. Pirmoje eilutėje įrašytas dalyvių skaičius N ($2 < N \le 200$). Antroje eilutėje pateikti numeriai tų asmenų, kuriuos pažįsta pirmasis asmuo: pirmas šios eilutės skaičius rodo, kiek asmenų iš viso pažįsta pirmasis asmuo, toliau pateikiami pažįstamų asmenų numeriai. Trečioje eilutėje analogiškai pateikti numeriai tų asmenų, kuriuos pažįsta antrasis asmuo ir t. t.

Rezultatą – neatvykusio asmens numerį – spausdinkite į failą *rez.txt*.

Pastaba. Jeigu sprendinių (neatvykusių asmenų) bus keletas, tai užtenka pateikti bet kurį vieną.

Pradinių duomenų ir rezultatų pavyzdys

Duomenys	*Rezultatas*
7 2 2 7 2 1 4 2 4 5 4 2 3 6 7 2 3 6 2 4 5 2 1 4	4

Kartu su 5-o, 6-o ir 7-o uždavinio sprendimu turi būti pateikta *programa, generuojanti **atsitiktinį** sąlygą atitinkantį pradinių duomenų rinkinį (atsitiktinį testą), kuriame **būtinai** turi būti **nors vienas** ieškomas objektas.*

Vertinant šių uždavinių sprendimus programos bus tikrinamos (testuojamos) su sąlygose pateiktais pradiniais duomenimis. *Tik tos programos,* kurios su šiais duomenimis duos teisingus rezultatus, bus tikrinamos su specialiai parengtais kontroliniais testais ir vertinamos balais.

Programai surinkus *bent pusę balų* (5), bus vertinami *5-o ir 7-o uždavinių sprendimų idėjos aprašai*, kurie turi būti parašyti kaip *komentaras sprendimo failuose prieš programą*, bei *6-o uždavinio* sprendimo *programavimo stilius (kultūra)*.

Šio skyriaus 5–7 uždavinių sprendimai turi būti pateikti JPM interneto svetainėje (http://ims.mii.lt/jpm/) iki 2009 m. balandžio 19 d. 24 val.

Sprendimų failų vardai turi būti *?????135.pas*, *?????136.pas* ir *?????137.pas*, generuojančių programų (generatorių) – *?????135gen.pas*, *?????136gen.pas* ir *?????137gen.pas*. Vietoj klaustukų įrašykite pirmąsias penkias savo pavardės raides (be diakritinių ženklų). Jei kartais būtų pavardžių, trumpesnių, negu penkios raidės, trūkstami simboliai keičiami pabraukimo brūkšniais „_“.

Elektroninio pašto adresas JPM antrosios dalies klausytojams: jpm.2kursas@gmail.com.