3 skyrius

ALGORITMAI GRAFUOSE

3.1 Trumpiausi keliai grafuose

Kiekvienam ne kartą teko spręsti uždavinį, kaip rasti trumpiausią kelią iš taško A į tašką B. A ir B gali būti du miestai, du pastatai viename mieste ir t.t. Geometrine prasme šis uždavinys atrodo trivialus: tereikia žemėlapyje sujungti taškus A ir B tiesės atkarpa, pasiimti kompasą ir drožti nosies tiesumu. Deja, gyvenime mes naudojamės egzistuojančiu kelių tinklu. Geometriškai trumpiausias maršrutas gali būti nerealus dėl įvairių kliūčių ir kitų priežasčių. Kadangi bet kurį kelių tinklą galime vaizduoti svoriniu orgrafu, tai gauname tokį grafų teorijos uždavinį:

(i) Duotas svorinis orgrafas $G = (V, E, \omega)$ ir dvi grafo viršūnės u ir v. Reikia rasti trumpiausią kelią iš u į v ir to kelio ilgį.

Dažnai mums tenka spręsti bendresnius trumpiausio kelio paieškos uždavinius:

(ii) Duotas svorinis orgrafas $G = (V, E, \omega)$ ir to grafo viršūnė u. Reikia rasti trumpiausius kelius iš u į visas kitas to grafo viršūnes.

(iii) Duotas svorinis orgrafas $G = (V, E, \omega)$. Reikia rasti trumpiausius kelius iš kiekvienos grafo viršūnės į kiekvieną kitą to grafo viršūnę.

Atrodytų, jei grafas turi $n = |V|$ viršūnių, tai antrasis uždavinys yra n kartų sudėtingesnis už pirmąjį, o trečiasis uždavinys savo ruožtu yra n kartų sudėtingesnis už antrąjį. Tačiau iš tikrųjų, norint rasti trumpiausią kelią tarp dviejų fiksuotų grafo viršūnių u ir v, tenka apžiūrėti visas grafo viršūnes, nes jei bent vieną praleisime, tai gali pasirodyti, kad kaip tik eidami per šią viršūnę mes trumpiausiu keliu pateksime iš u į v. Žemiau pateikiame Floyd–Warshall algoritmą, kuris sprendžia trečiąjį uždavinį, o tuo pačiu ir pirmuosius du uždavinius. Nors yra žinoma daug kitų algoritmų (pvz., Deikstros ir Fordo–Belmano) dviems pirmiesiems uždaviniams spręsti, tačiau įdomu tai, kad nė vienas iš tų algoritmų nėra paprastesnis už Floyd–Warshall algoritmą.

Taigi, duotas orgrafas $G = (V, E)$ su viršūnių aibe V (laikome, kad $V = \{v_1, \ldots, v_n\}$), briaunų aibe E ir svorių (atstumų) matrica $A = (\omega(v_i, v_j))$. Reikia rasti trumpiausių kelių ilgių matricą D, t.y.

$$D[i,j] = \min_{K(i,j)} \sum_{e \in K(i,j)} \omega(e).$$

Galime laikyti, kad G yra pilnas grafas, nes jei dvi viršūnės nėra sujungtos briauna, laikome, kad tokios briaunos svoris yra begalybė (∞). Svoriai (atstumai) gali būti ir neigiami, tačiau grafas G negali turėti neigiamo svorio ciklų, t.y. kelių $K(i,i)$: $\sum_{e \in K(i,i)} \omega(e) < 0$ (priešingu atveju mes be galo suktumėmės tokiu ciklu, o nueitas kelias artėtų į $-\infty$).

Kodėl mes ieškome tik trumpiausių kelių ilgių, o ne pačių kelių ? Pasirodo, kad pačius trumpiausius kelius galima lengvai rasti, naudojant matricas D ir A. Norėdami rasti priešpaskutinę kelio $K(i,j)$ viršūnę v_k, ieškome tokio k, kuriam būtų teisinga lygybė $D[i,j] = D[i,k] + A[k,j]$, t.y. sudedame matricos D i-osios eilutės ir matricos A j-ojo stulpelio atitinkamus elementus, kol gausime lygybę. Aišku, kad $\forall l\ D[i,j] \leqslant D[i,l] + A[l,j]$. Kadangi trumpiausias kelias $K(i,j)$ turi praeiti per kažkurią viršūnę $k \neq j$, tai būtinai atsiras k tokia, kad $D[i,j] = D[i,k] + A[k,j]$. Suradę priešpaskutinę kelio viršūnę, ieškome priešpriešpaskutinės ir t.t., kol grįšime į i.

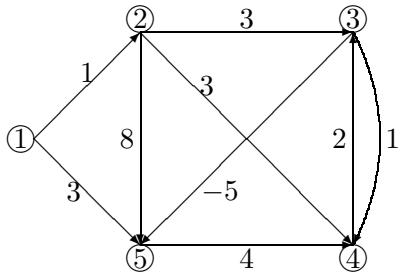
Pagrindinė Floyd–Warshall algoritmo idėja yra paeiliui įterpinėti naujas tarpines viršūnes į visus tuo metu rastus trumpiausius kelius ir tikrinti, ar naudojant naują tarpinę viršūnę gausime naują kelią, trumpesnį už jau turimą kelią. Kadangi grafas neturi neigiamų ciklų, tai bet kuriame trumpiausiame kelyje kiekviena viršūnė pasitaikys ne daugiau kaip 1 kartą. Tarkime, $i, j \in V$ ir K yra trumpiausias kelias iš i į j tarp tokių kelių, kurių visos tarpinės viršūnės priklauso aibei $N_k = \{1, \ldots, k\}$, kur $k \leqslant n$. Jei viršūnė k priklauso keliui K, tai kelio K pirmoji dalis $K_1 = K(i,k)$ bus trumpiausias kelias tarp viršūnių i ir k su tarpinėmis viršūnėmis iš aibės N_k, o kelias $K_2 = K(k,j)$ bus trumpiausias kelias tarp viršūnių k ir j su tarpinėmis viršūnėmis iš aibės N_k (nes priešingu atveju egzistuotų kelias iš i į j, trumpesnis už K). Jei viršūnė k nepriklauso keliui K, tai kelias K bus trumpiausias kelias tarp viršūnių i ir j su tarpinėmis viršūnėmis iš aibės N_{k-1}.

Pažymėję trumpiausio kelio iš i į j su tarpinėmis viršūnėmis iš aibės N_k ilgį $D^{(k)}[i,j]$, gauname rekurentinę sąsają trumpiausių kelių ilgiams:

$$D^{(k)}[i,j] = \begin{cases} A[i,j], & \text{jei } k = 0, \\ \min\{D^{(k-1)}[i,j], D^{(k-1)}[i,k] + D^{(k-1)}[k,j]\}, & \text{jei } k \geqslant 1. \end{cases}$$

Floyd–Warshall algoritmas realizuojamas trigubu ciklu:

function $D =$ shortest_paths(A)
for $i := 1$ **to** n **do**
  for $j := 1$ **to** n **do** $D[i,j] := A[i,j]$
  end
end
for $k := 1$ **to** n **do**



Pav. 3.1: Rasti trumpiausius kelius.

for $i := 1$ **to** n **do**
for $j := 1$ **to** n **do** $D[i,j] := \min\{D[i,j], D[i,k] + D[k,j]\}$
end
end
end

Šio algoritmo sudėtingumas yra $\mathrm{O}(n^3)$.

Pavyzdys 3.1.1. Duota grafas (žr. Pav. **??**) su atstumų matrica

$$A = \begin{pmatrix} 0 & 1 & \infty & \infty & 3 \\ \infty & 0 & 3 & 3 & 8 \\ \infty & \infty & 0 & 1 & -5 \\ \infty & \infty & 2 & 0 & \infty \\ \infty & \infty & \infty & 4 & 0 \end{pmatrix}.$$

Rasti trumpiausių kelių tarp bet kurių dviejų grafo viršūnių ilgius bei trumpiausią kelią iš 1 į 4 viršūnę.

Pademonstruosime kaip keičiasi trumpiausių kelių ilgių matrica D (pradiniu momentu $D^{(0)} = A$):

$$D^{(1)} = \begin{pmatrix} 0 & 1 & \infty & \infty & 3 \\ \infty & 0 & 3 & 3 & 8 \\ \infty & \infty & 0 & 1 & -5 \\ \infty & \infty & 2 & 0 & \infty \\ \infty & \infty & \infty & 4 & 0 \end{pmatrix},$$

$$D^{(2)} = \begin{pmatrix} 0 & 1 & 4 & 4 & 3 \\ \infty & 0 & 3 & 3 & 8 \\ \infty & \infty & 0 & 1 & -5 \\ \infty & \infty & 2 & 0 & \infty \\ \infty & \infty & \infty & 4 & 0 \end{pmatrix},$$

$$D^{(3)} = \begin{pmatrix} 0 & 1 & 4 & 4 & -1 \\ \infty & 0 & 3 & 3 & -2 \\ \infty & \infty & 0 & 1 & -5 \\ \infty & \infty & 2 & 0 & -3 \\ \infty & \infty & \infty & 4 & 0 \end{pmatrix},$$

$$D^{(4)} = \begin{pmatrix} 0 & 1 & 4 & 4 & -1 \\ \infty & 0 & 3 & 3 & -2 \\ \infty & \infty & 0 & 1 & -5 \\ \infty & \infty & 2 & 0 & -3 \\ \infty & \infty & 6 & 4 & 0 \end{pmatrix},$$

$$D^{(5)} = \begin{pmatrix} 0 & 1 & 4 & 3 & -1 \\ \infty & 0 & 3 & 2 & -2 \\ \infty & \infty & 0 & -1 & -5 \\ \infty & \infty & 2 & 0 & -3 \\ \infty & \infty & 6 & 4 & 0 \end{pmatrix} = D.$$

Liko rasti trumpiausią kelią iš 1 į 4. Kadangi $D[1,4] = 3 = D[1,5] + A[5,4]$, tai priešpaskutinė šio kelio viršūnė yra 5. Kadangi $D[1,5] = -1 = D[1,3] + A[3,4]$, tai prieš 5 viršūnę eina viršūnė 3. Kadangi $D[1,3] = 4 = D[1,2] + A[2,3]$, tai prieš 3 viršūnę eina viršūnė 2. Gauname kelią $1 \to 2 \to 3 \to 5 \to 4$ ilgio 3.

3.2 Grafų izomorfizmas

Duoti du grafai (orientuoti arba neorientuoti) $G_1 = (V_1, E_1)$ ir $G_2 = (V_2, E_2)$. Reikia nustatyti, ar šie grafai yra izomorfiški, t.y. ar egzistuoja bijekcija $f : V_1 \to V_2$ tokia, kad $(v_i, v_j) \in E_1$ tada ir tik tada, kai $(f(v_i), f(v_j)) \in E_2$. Šiam uždaviniui kol kas nėra rasta jokio algoritmo, kurio sudėtingumas polinomiškai priklausytų nuo grafų viršūnių bei briaunų skaičiaus.

3.2.1 Grafų invariantai

Tam, kad du grafai būtų izomorfiški, turi sutapti įvairios jų charakteristikos. Pavyzdžiui, kadangi turi egzistuoti bijekcija $f : V_1 \to V_2$, kuri skirtingas grafo G_1 briaunas atvaizduotų į skirtingas grafo G_2 briaunas, tai abu grafai privalo turėti tiek pat viršūnių ir tiek pat briaunų. Tokios grafo skaitinės charakteristikos yra vadinamos grafų invariantais. Išvardinsime keletą pagrindinių grafo invariantų:

1. Viršūnių skaičius.
2. Briaunų skaičius.

3. Komponenčių skaičius.

4. Viršūnių laipsnių seka, išdėstyta nedidėjančia tvarka.

5. Visų grafo paprastų ciklų ilgių seka, išdėstyta nedidėjančia tvarka.

Jei įtariame, kad du grafai nėra izomorfiški, iš pradžių verta patikrinti, ar sutampa kai kurie jų invariantai. Deja, niekam nepavyko surasti tokios baigtinės invariantų sekos, kad sutampant visiems šios sekos invariantams galėtume teigti, kad grafai yra izomorfiški. Todėl patikrinę keletą grafo invariantų, jei jie visi sutampa, naudodami paiešką su grįžimu konstruosime bijekciją $f : V_1 \to V_2$.

3.2.2 Paieška gylyn su grįžimu

Spręsime grafų izomorfizmo uždavinį orientuotiems grafams. Tam, kad nereiktų operuoti su grafo viršūnių įėjimo ir išėjimo laipsnių sekomis, iš pradžių koduojame grafo viršūnės išėjimo ir įėjimo laipsnius vienu skaičiumi. Tarkime,

$$V_1 = \{v_1, v_2, \ldots, v_n\}, \qquad V_2 = \{w_1, w_2, \ldots, w_n\}$$

ir $t = \lceil \log_{10}(n+1) \rceil$. Viršūnės v_i laipsnį apibrėžę kaip

$$\mathrm{dg}(v_i) = 10^t \mathrm{indg}(v_i) + \mathrm{outdg}(v_i),$$

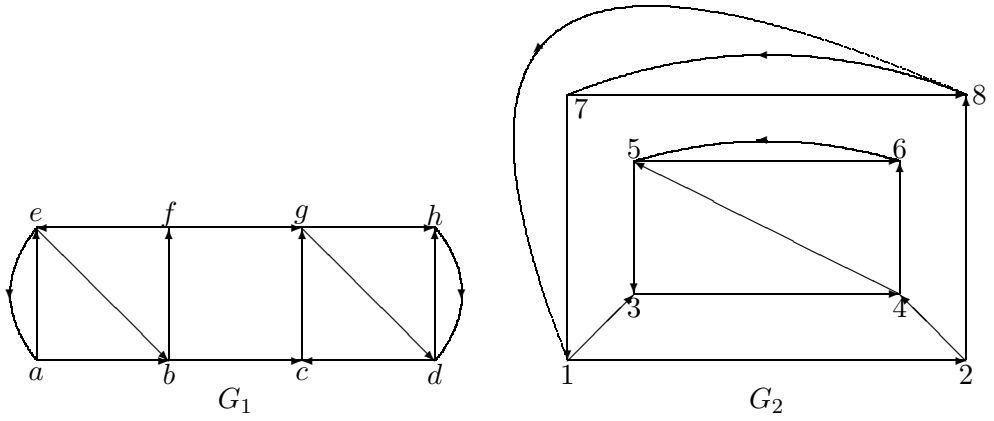
gauname, kad skirtingas poras $(\mathrm{indg}(v), \mathrm{outdg}(v))$ atitiks skirtingi natūralieji skaičiai $\mathrm{dg}(v)$.

Pagrindiniai algoritmo etapai yra šie:

1. Abiejų grafų viršūnių aibėms V_1 ir V_2 apskaičiuojame viršūnių laipsnius $\mathrm{dg}(v_i)$ ir $\mathrm{dg}(w_i)$ sugrupuojame viršūnes su vienodais laipsniais į grupes. Taigi, viršūnių aibės suskyla į k grupių: $V_1 = V_1^1 \cup \cdots \cup V_k^1$ ir $V_2 = V_1^2 \cup \cdots \cup V_k^2$, kur $|V_j^1| = |V_j^2| \ \forall j = 1, \ldots, k$.

2. Mažiausiai viršūnių turinčioje grupėje V_j^1 pasirenkame bet kurią viršūnę $v_1 \in V_j^1$ ir iš šios viršūnės grafe G_1 vykdome paiešką gylyn. Tarkime, kad šioje paieškoje grafo G_1 viršūnės yra apeinamos tvarka $v_1, v_2, \ldots, v_n$ ir pažymėkime $G_1(k)$ grafo G_1 pografį, kurį sudaro viršūnės $v_1, \ldots, v_k$ ir visos briaunos, kurių abu galai priklauso aibei $v_1, \ldots, v_k$.

3. Tuo pačiu metu su paieška gylyn konstruojame bijekciją $f(v_1) = w_1$, $f(v_2) = w_2$, …, $f(v_k) = w_k$, taip, kad pografiai $G_1(k)$ ir $G_2(k)$ būtų izomorfiški. Čia $G_2(k)$ yra pografis, kurį grafe G_2 sudaro viršūnės $w_1, \ldots, w_k$. Kai negalime rasti $f(k+1)$ reikšmės tokios, kad pografiai $G_1(k{+}1)$ ir $G_2(k{+}1)$ būtų izomorfiški, mes grįžtame prie pografio $G_1(k{-}1)$ ir renkamės kitą reikšmę $f(k)$. Algoritmas baigs darbą, kai gausime, kad pografis $G_1(n) = G_1$ yra izomorfiškas pografiui $G_2(n) = G_2$.



Pav. 3.2: Nustatyti, ar šie grafai yra izomorfiški.

Pavyzdys 3.2.1. Nustatysime, ar Pav. **??** vaizduojami orgrafai yra izomorfiški. Abu grafai turi po 8 viršūnes ir 14 lankų. Taigi, $t = 1$. Viršūnių aibė V_1 suskyla į 3 grupes:

$$V_1 = \{a, f\} \cup \{c, h\} \cup \{b, d, e, g\} = V_1^1 \cup V_2^1 \cup V_3^1,$$

kur grupės V_1^1 viršūnės yra laipsnio 12, grupės V_2^1 — laipsnio 21 ir grupės V_3^1 — laipsnio 22. Atitinkamas antrojo grafo viršūnių aibės skaidinys yra

$$V_2 = \{2, 7\} \cup \{3, 6\} \cup \{1, 4, 5, 8\} = V_1^2 \cup V_2^2 \cup V_3^2.$$

Kadangi ieškoma bijekcija $f : V_1 \to V_2$ turi išlaikyti viršūnės laipsnį, tai lieka $2! \cdot 2! \cdot 4! = 96$ galimos bijekcijos.

Grafe G_1 pradedame paiešką gylyn iš viršūnės a. Tarkime, kad $f(a) = 2$. Iš a išeina vienintelis lankas (a, b). Kadangi $b \in V_3^1$, tai $f(b) \in V_3^2$. Kadangi grafe G_2 yra briauna $(2, 4)$, tai galime pasirinkti $f(b) = 4$. Iš b einame į f. Viršūnę f gali atitikti tik viršūnė 7, nes viršūnė 2 jau užimta. Taigi, $f(f) = 7$. Tačiau grafe G_2 nėra briaunos $(4, 7)$. Tai reiškia, kad pasirinkimas $f(b) = 4$ buvo neteisingas.

Grįžtame į viršūnę b ir renkamės $f(b) = 8$. Vėl einame iš b į f ir gauname $f(f) = 7$. Kadangi grafe G_2 yra lankas $(8, 7)$, tai viskas gerai. Tačiau grafe G_2 yra ir atvirkščiai orientuotas lankas $(7, 8)$, tuo tarpu grafe G_1 lanko (f, b) nėra. Vadinasi, $f(b) \neq 8$. Kadangi kitų galimybių parinkti $f(b)$ nebėra, tai reikia grįžti į viršūnę a.

Dabar imame $f(a) = 7$. Tada galima pasirinkti $f(b) = 1$ ir $f(f) = 2$. Dar po keleto paieškos žingsnių gauname ieškomą bijekciją f, kuri priskiria viršūnėms a, b, c, d, e, f, g, h atitinkamai viršūnes $7, 1, 3, 5, 8, 2, 4, 6$. Taigi, grafai G_1 ir G_2 yra izomorfiški.