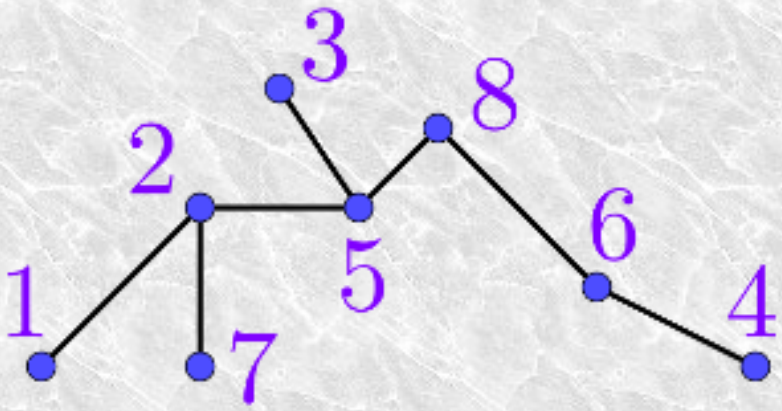


Prüferov kód : 256825

12. Dijkstrov algoritmus, Prüferov kód

V tejto kapitole sa budeme zaoberať dvoma rôznymi problémami. Prvým je hľadanie najlacnejšej cesty Dijkstrovým algoritmom a druhým je Prüferov kód (časť 12.3).

12.1 Hľadanie najlacnejšej cesty Dijkstrovým algoritmom

V kapitole 10 boli definované ohodnotené grafy a popísané dva algoritmy hľadania minimálnej kostry ohodnoteného grafu. Problém hľadania minimálnej kostry daného grafu sa v aplikáciach vyskytuje pomerne často, ale ani zďaleka to nie je jediný problém súvisiaci s ohodnotenými grafmi. Iným častým problémom je napríklad hľadanie najlacnejšej cesty medzi dvoma danými vrcholmi, alebo najlacnejšej cesty z daného vrcholu do všetkých ostatných vrcholov.

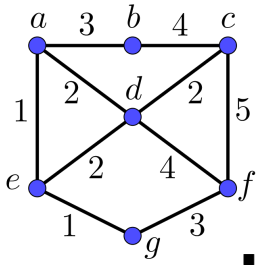
Už v príklade 10.1 sme hľadali „najkratšiu“, alebo lepšie povedané najlacnejšiu, cestu medzi dvoma vrcholmi. Teraz formálne definujeme pojmy *hodnota sledu* a *najlacnejšia cesta*.

Definícia 12.1.1 — Hodnota sledu. Hodnotou sledu v ohodnotenom grafe, nazývame súčet ohodnotení všetkých hrán daného sledu.

P Spomeňme si, že podľa definície 3.3.3 je každá cesta zároveň ťah a každý ťah je zároveň sled, takže predošlá definícia sa rovnako týka aj ťahov a ciest. V prevažnej väčšine praktických aplikácií nás budú zaujímať práve cesty.

▪ Príklad 12.1

Hodnota sledu $s = (a,b,c,f,d,e)$ v grafe, ktorý vidíme na obrázku vpravo, je $3+4+5+4+2 = 18$.



▪

Keď budeme hľadať sled s najmenšou hodnotou medzi dvoma danými vrcholmi grafu G, tak vždy pôjde o taký ohodnotený graf, ktorého všetky ohodnotenia sú kladné čísla. Pri záporných

ohodnoteniach hrán môže byť cena najlacnejšieho sledu zdola neohraničená, a preto záporné ohodnotenia hrán nebudeme pripúšťať. Takže vo všetkých úlohach, v ktorých pôjde o hľadanie sledu s najmenšou hodnotou, budeme uvažovať len grafy s kladnými ohodnoteniami.

Okrem toho, ak hľadáme sled s najmenšou hodnotou, tak vždy pôjde o cestu. Každá cesta je zároveň sled, v ktorom sa neopakujú hrany ani vrcholy. Triviálne sa dá dokázať tvrdenie, že ak má sled medzi vrcholmi u a v v ohodnotenom grafe grafe G najmenšiu možnú hodnotu, tak potom sa v ňom žiadne hrany ani vrcholy neopakujú. Pozri cvičenie 12.1.

Definícia 12.1.2 — Najlacnejšia cesta. Cesta z vrcholu u do vrcholu v, ktorá má najmenšiu hodnotu, sa nazýva najlacnejšia cesta z u do v.

V úlohách, v ktorých ohodnotenia hrán grafov budú zodpovedať vzdialenostiam alebo dĺžkam, napríklad grafy dopravnej siete, budeme niekedy hovoriť o *najkratšej* ceste, prípadne *najmenšej vzdialenosti* medzi vrcholmi, tak ako sme to robili v príklade 10.1. Inak budeme vždy hovoriť o najlacnejšej ceste a ceste s najmenšou hodnotou medzi dvoma vrcholmi.

■ **Príklad 12.2** V grafe z príkladu 12.1 má cesta s najmenšou hodnotou (najkratšia cesta) z vrcholu a do vrcholu c hodnotu (dĺžku) 4 a je to cesta (a,d,c). ■

Úloha nájdenia najlacnejšej cesty medzi danými dvoma vrcholmi v ohodnotenom grafe sa veľmi často vyskytuje v rôznych aplikáciach. Existuje algoritmus, ktorý v ohodnotenom grafe nájde najlacnejšiu cestu medzi dvoma jeho vrcholmi. Jeho autorom je holandský matematik Edsger W. Dijkstra. Existuje viacero modifikácií Dijkstrovho algoritmu. Popis Dijkstrovho algoritmu nájdeme napríklad v [15], [7], alebo [18] a samozrejme nájdeme ho aj na webe, napríklad vo Wikipedii[1]. Vo svojej základnej verzii Dijkstrov algoritmus nájde, pre daný ohodnotený graf G a v ňom pevne určený vrchol v, dĺžky najlacnejších ciest od vrcholu v ku všetkým ostatným vrcholom v grafe G. My tu uvedieme drobnú modifikáciu Dijkstrovho algoritmu, ktorá nám okrem dĺžok najlacnejších ciest od vrcholu v umožní tieto cesty aj jednoducho skonštruovať.

12.1.1 Popis Dijkstrovho algorimu

Pri hľadaní najlacnejšej cesty sa stačí zaoberať obyčajnými grafmi, pretože slučky sa v najlacnejších cestách určite vyskytovať nebudú (pozri cvičenie 12.2). Preto ak by daný graf obsahoval slučky, môžeme ich vynechať. A ak by daný graf obsahoval násobné hrany medzi dvoma vrcholmi, tak stačí brať do úvahy len hranu s najmenšou hodnotou a všetky ostatné hrany medzi týmito dvoma vrcholmi môžeme vynechať.

Majme teda ohodnotený obyčajný graf $G=(V,E)$, v ktorom sú všetky ohodnotenia hrán kladné a pevne daný vrchol v v tomto grafe. Dijkstrov algoritmus nám určí najlacnejšiu cestu medzi každým vrcholom grafu G a vrcholom v. Algoritmus bude každému vrcholu $u \in V$ priraďovať značky. Tieto značky budú najskôr dočasné a potom trvalé. V oboch prípadoch bude mať značka tvar $\big(L(u),\mathbb{P}(u)\big)$. Hodnota $L(u)$ predstavuje pri dočasnej značke momentálne odhadovanú „najmenšiu“ hodnotu cesty medzi vrcholom u a vrcholom v. V prípade trvalej značky je to hodnota najlacnejšej cesty medzi vrcholom u a vrcholom v. Kým má niektorý vrchol len dočasnú značku, tak táto sa môže ešte meniť. Akonáhle je niektorému vrcholu priradená trvalá značka, táto sa už meniť nebude. Množina $\mathbb{P}(u)$ v značke vrcholu sa v pôvodnom Dijkstrovom algoritme nevyskytuje. Je to práve tá modifikácia, ktorá nám po skončení algoritmu umožní skonštruovať najlacnejšie cesty od každého vrcholu grafu G do vrcholu v. Množina $\mathbb{P}(u)$ bude v trvalej značke obsahovať všetkých susedov vrcholu u na cestách s najmenšou hodnotou vedúcich z vrcholu u do vrcholu v.

[1] `https://en.wikipedia.org/wiki/Dijkstra's_algorithm`

P V popise Dijkstrovho algoritmu budeme používať tieto označenia

- $G = (V,E)$ – ohodnotený obyčajný graf, ktorého všetky ohodnotenia hrán sú kladné,
- V – množina všetkých vrcholov grafu G,
- $w(h)$ – ohodnotenie hrany h v grafe G,
- $\mathbb{A}$ – množina „aktívnych“ vrcholov grafu G, čiže tých, ktoré ešte nemajú trvalú značku.

Dijkstrov algoritmus

VSTUP: Ohodnotený obyčajný graf graf $G = (V,E)$ a vrchol $v \in V$.

VÝSTUP: Pre každé $u \in V$ trvalá značka $\big(L(u), \mathbb{P}(u)\big)$.

INICIALIZÁCIA

$\mathbb{A} := V$;
for $\{$ *každé* $u \in V$ $\}$ **do**
  $L(u) := \infty$;
  $\mathbb{P}(u) := \{u\}$;
end
$L(v) := 0$;
$\mathbb{P}(v) := \{v\}$;

PRIDEĽOVANIE ZNAČIEK VRCHOLOM

while $\{$ *existuje* $x \in \mathbb{A}$; $L(x) < \infty$ $\}$ **do**
  $\delta := \min\{L(u);\ u \in \mathbb{A}\}$;
  $\mathbb{N} := \{u \in \mathbb{A};\ L(u) = \delta\}$;
  $\mathbb{A} := \mathbb{A} \setminus \mathbb{N}$;
  for $\{$ *pre každú hranu* $h = \{u,x\}$, *kde* $u \in \mathbb{N}$ *a* $x \in \mathbb{A}$ $\}$ **do**
    if $\{ L(x) = L(u) + w(h) \}$ **then**
      $\mathbb{P}(x) := \mathbb{P}(x) \cup \{u\}$;
    end
    if $\{ L(x) > L(u) + w(h) \}$ **then**
      $L(x) := L(u) + w(h)$;
      $\mathbb{P}(x) := \{u\}$;
    end
  end
end

V pseudokóde Dijkstrovho algoritmu uvedenom vyššie je podmienka ukončenia algoritmu *pre každé* $x \in \mathbb{A}: L(x) = \infty$. Znamená to, že buď $\mathbb{A} = \emptyset$, alebo sú všetky vrcholy množiny $\mathbb{A}$ nedosiahnuteľné z vrcholu v. V prípade súvislého grafu G môže nastať len prípad $\mathbb{A} = \emptyset$. Po skončení Dijkstrovho algoritmu má každý vrchol grafu G značku $\big(L(u), \mathbb{P}(u)\big)$, kde $L(u)$ je hodnota najlacnejšej cesty medzi vrcholmi u a v a množina $\mathbb{P}(u)$ obsahuje zoznam všetkých susedov vrcholu u, na cestách s najmenšou hodnotou vedúcich z vrcholu u do vrcholu v. Uvedený algoritmus sa dá triviálne modifikovať aj pre orientované ohodnotené grafy.

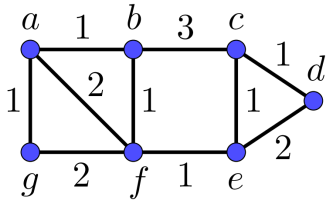
Dôkaz správnosti Dijkstrovho algoritmu sa robí matematickou indukciou a môžeme ho nájsť napríklad v knihe [15]. Slovný popis Dijkstrovho algoritmu je nasledovný:

1. Prvým krokom algoritmu je inicializácia.
   - Množina $\mathbb{A}$ bude obsahovať všetky vrcholy grafu $G = (V,E)$.
   - Pre každý vrchol $u \in V$ bude $\mathbb{P}(u) = \{u\}$.
   - Pre každý vrchol $u \in V$, $u \neq v$ bude $L(u) = \infty$.
   - Pre vrchol v bude $L(v) = 0$.
2. Podmienka ukončenia algoritmu je *pre každé* $x \in \mathbb{A}: \ L(x) = \infty$.
   Čiže ak *existuje* $x \in \mathbb{A}: \ L(x) < \infty$, algoritmus pokračuje.
3. Druhým krokom algoritmu je voľba množiny $\mathbb{N}$ a úprava množiny $\mathbb{A}$.
   - Spomedzi vrcholov $x \in \mathbb{A}$ nájdeme minimálnu hodnotu $L(x)$ a označíme ju δ.
   - Do množiny $\mathbb{N}$ dáme všetky vrcholy $x \in \mathbb{A}$, pre ktoré $L(x) = \delta$.
   - Z množiny $\mathbb{A}$ vyhodíme vrcholy patriace do množiny $\mathbb{N}$.
4. Ďalším krokom algoritmu je úprava dočasných a prideľovanie trvalých značiek vrcholom.
   - Nájdeme všetky hrany $h = \{u,x\}$, ktorých koncový vrchol u patrí do množiny $\mathbb{N}$ a koncový vrchol x patrí do množiny $\mathbb{A}$. Budeme upravovať značky vrcholov x.
   - Ak platí $L(x) = L(u) + w(h)$, znamená to, že z vrcholu x sa cez vrchol u dostaneme do vrcholu v cestou rovnakej hodnoty, ako má doteraz najlacnejšia nájdená cesta. Takže hodnotu $L(x)$ meniť netreba. Ale do množiny $\mathbb{P}(x)$ pridáme vrchol u ako počiatočný vrchol alternatívnej najlacnejšej cesty do vrcholu v.
   - Ak platí $L(x) > L(u) + w(h)$, tak z vrcholu x do vrcholu v sa vieme dostať cez vrchol u lacnejšou cestou, než sme našli doteraz. Preto zmeníme hodnoty $L(x)$ aj $\mathbb{P}(x)$.
   - Návrat na test podmienky ukončenia algoritmu (2. bod).

Použitie Dijkstrovho algoritmu si teraz ilustrujeme na niekoľkých príkladoch.

■ Príklad 12.3

V ohodnotenom grafe, ktorý je na obrázku vpravo, nájdite hodnoty najlacnejších ciest z vrcholu a do všetkých ostatných vrcholov a všetky najlacnejšie cesty z vrcholu a do vrcholu d.



Priebeh Dijkstrovho algoritmu budeme zapisovať do tabuľky. Pokiaľ sa v niektorom kroku značka vrcholu nebude meniť, tak do príslušného políčka zapíšeme —. Trvalé značky vrcholov budú označené modrou farbou a vrcholmi, ktoré už majú pridelenú trvalú značku, sa nemusíme viac zaoberať.

	a	b	c	d	e	f	g
1.	$(0,\{a\})$	$(\infty,\{b\})$	$(\infty,\{c\})$	$(\infty,\{d\})$	$(\infty,\{e\})$	$(\infty,\{f\})$	$(\infty,\{g\})$
2.	$(0,\{a\})$	$(1,\{a\})$	—	—	—	$(2,\{a\})$	$(1,\{a\})$
3.		$(1,\{a\})$	$(4,\{b\})$	—	—	$(2,\{a,b\})$	$(1,\{a\})$
4.			—	—	$(3,\{f\})$	$(2,\{a,b\})$	
5.			$(4,\{b,e\})$	$(5,\{e\})$	$(3,\{f\})$		
6.			$(4,\{b,e\})$	$(5,\{c,e\})$			
7.				$(5,\{c,e\})$			

Hodnoty najlacnejších ciest z vrcholu a do všetkých ostatných vrcholov grafu odčítame z trvalých značiek vrcholov. Napríklad najlacnejšia cesta do vrcholu c má hodnotu 4 a najlacnejšia cesta

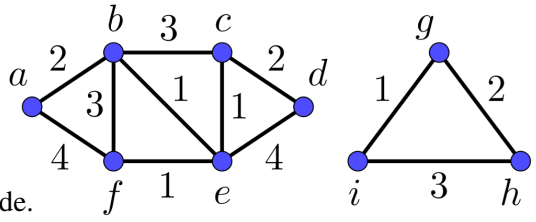
do vrcholu e má hodnotu 3. Ak chceme nájsť najlacnejšie cesty medzi vrcholom a a vrcholom d, tak začneme vo vrchole d, z jeho značky odčítame predposledný vrchol na najlacnejšej ceste, ideme do tohto vrcholu a opäť z jeho značky odčítame ďalší vrchol cesty atď. Takže najlacnejšie cesty medzi vrcholmi a a d sú

$$d \leftarrow \begin{cases} c \leftarrow \begin{cases} b \leftarrow a \\ e \leftarrow f \leftarrow \begin{cases} a \\ b \leftarrow a \end{cases} \end{cases} \\ e \leftarrow f \leftarrow \begin{cases} a \\ b \leftarrow a \end{cases} \end{cases}$$

Z obrázku vidíme, že existuje 5 najlacnejších ciest z vrcholu a do vrcholu d. Sú to cesty (a,b,c,d), (a,f,e,c,d), (a,b,f,e,c,d), (a,f,e,d) a (a,b,f,e,d). Všetkých päť ciest má hodnotu 5. ■

■ Príklad 12.4

V ohodnotenom grafe, ktorý je na obrázku vpravo, nájdite hodnoty najlacnejších ciest z vrcholu a do všetkých ostatných vrcholov a najlacnejšie cesty z vrcholu a do vrcholu d.



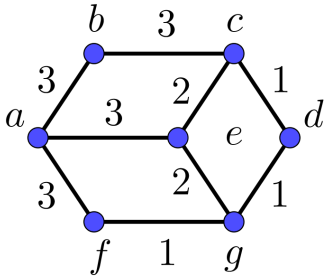
Postupovať budeme rovnako ako v predošlom príklade.

	a	b	c	d	e	f	g	h	i
1.	$(0,\{a\})$	$(\infty,\{b\})$	$(\infty,\{c\})$	$(\infty,\{d\})$	$(\infty,\{e\})$	$(\infty,\{f\})$	$(\infty,\{g\})$	$(\infty,\{h\})$	$(\infty,\{i\})$
2.	$(0,\{a\})$	$(2,\{a\})$	—	—	—	$(4,\{a\})$	—	—	—
3.		$(2,\{a\})$	$(5,\{b\})$	—	$(3,\{b\})$	—	—	—	—
4.			$(4,\{e\})$	$(7,\{e\})$	$(3,\{b\})$	$(4,\{a,e\})$	—	—	—
5.			$(4,\{e\})$	$(6,\{c\})$		$(4,\{a,e\})$	—	—	—
6.				$(6,\{c\})$			—	—	—

Algoritmus v 6. kroku skončil, pretože v aktívnej množine zostali už len vrcholy x so značkou $L(x) = \infty$. To znamená, že tieto vrcholy sú z vrcholu a nedosiahnuteľné. Najlacnejšia cesta z vrcholu a do vrcholu d je len jedna. Je to cesta $d \leftarrow c \leftarrow e \leftarrow b \leftarrow a$. ■

■ Príklad 12.5

V ohodnotenom grafe, ktorý je na obrázku vpravo, nájdite hodnoty najlacnejších ciest z vrcholu a do všetkých ostatných vrcholov a najlacnejšie cesty z vrcholu a do vrcholu d.



Postupovať budeme rovnako ako v predošlých príkladoch.

	a	b	c	d	e	f	g
1.	$(0,\{a\})$	$(\infty,\{b\})$	$(\infty,\{c\})$	$(\infty,\{d\})$	$(\infty,\{e\})$	$(\infty,\{f\})$	$(\infty,\{g\})$
2.	$(0,\{a\})$	$(3,\{a\})$	—	—	$(3,\{a\})$	$(3,\{a\})$	—
3.		$(3,\{a\})$	$(5,\{e\})$	—	$(3,\{a\})$	$(3,\{a\})$	$(4,\{f\})$
4.			—	$(5,\{g\})$			$(4,\{f\})$
5.			$(5,\{e\})$	$(5,\{g\})$			

Najlacnejšia cesta z vrcholu a do vrcholu d je len jedna. Je to cesta $d \leftarrow g \leftarrow f \leftarrow a$ a má dĺžku 5. ■

12.2 Problém obchodného cestujúceho

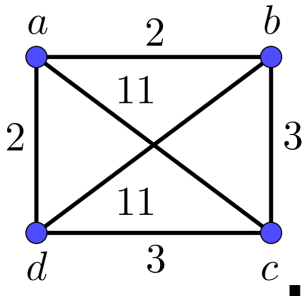
Problém obchodného cestujúceho je podobný problému hamiltonovského cyklu. Je to v podstate problém nájdenia najlacnejšieho hamiltonovského cyklu v ohodnotenom grafe. Rovnako ako problém existencie hamiltonovského cyklu v danom grafe, vznikol aj problém obchodného cestujúceho už v 19. storočí a súvisí s čiste praktickým problémom. Máme niekoľko miest, medzi nimi sú cesty. Tieto cesty sú rôznej dĺžky a kvality a obchodný cestujúci má prejsť všetkými mestami. Aby šetril čas a náklady na cestu, chce prejsť každým mestom práve raz.

Situáciu si môžeme opäť modelovať grafom. Mestám budú zodpovedať vrcholy grafu a cestám budú zodpovedať hrany grafu. Dĺžka ciest, časová náročnosť ciest, alebo cestovné náklady zodpovedajúce jednotlivým cestám budú vyjadrené ohodnotením hrán grafu. Tým sa problém obchodného cestujúceho transformuje na problém hľadania najlacnejšieho hamiltonovského cyklu v ohodnotenom grafe. Úlohou je nájsť najlacnejší, v zmysle dĺžky, dopravných nákladov, času, alebo ďalších kritérií vyjadrených ohodnotením hrán, hamiltonovský cyklus.

Keďže už samotný problém existencie hamiltonovského cyklu v neohodnotenom grafe je NP-úplný, je taký aj problém obchodného cestujúceho. To znamená, že vo všeobecnosti neexistuje jeho efektívne riešenie. Existuje však viacero heuristík, pomocou ktorých sa tento problém dá riešiť. Pri malom počte miest, hrán a vhodných ohodnoteniach, je riešenie pomerne jednoduché. Ukážeme si to na príklade.

■ Príklad 12.6

Problém obchodného cestujúceho budeme riešiť na grafe z obrázku vpravo. Cyklus $c = (a,b,c,d)$ je hamiltonovský cyklus v danom grafe a jeho hodnota je 10. Akýkoľvek cyklus obsahujúci hrany $\{a,c\}$, alebo $\{b,d\}$, ktoré majú dĺžku 11, by mal väčšiu hodnotu. Vidíme teda, že cyklus dĺžky 10 je najlacnejší možný. Preto bude cyklus c riešením problému obchodného cestujúceho v danom grafe.



■

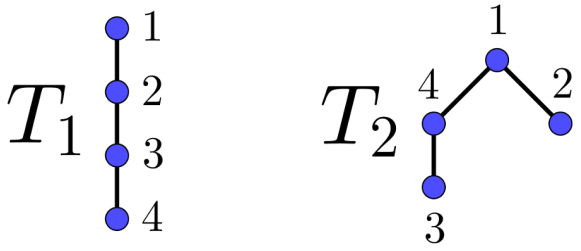
12.3 Prüferov kód

Prüferov kód je efektívny spôsob kódovania očíslovaných stromov. Pod *očíslovaným stromom* rozumieme strom, ktorého vrcholy sú označené číslami[2]. Napríklad očíslované stromy T_1 a T_2 na nasledovnom obrázku

sú rôzne, aj keď ako grafy sú izomorfné. Očíslovaný strom si môžeme predstaviť ako koreňový strom, ktorého koreň je vždy vrchol 1. Stromy T_1 a T_2 sa potom dajú nakresliť aj tak, ako to vidíme na obrázku 12.1 (strana 235). Pri takejto interpretácii môžu mať aj dva izomorfné grafy úplne odlišnú podobu, ako to vidno z uvedeného príkladu.

Pomocou Prüferovho kódu vieme každý očíslovaný strom na n vrcholoch, jednoznačne reprezentovať postupnosťou $(p_1, p_2, \ldots, p_{n-2})$, čiže ako usporiadanú $(n-2)$-ticu. Uvedieme si pseudokódy dvoch algoritmov. Pomocou prvého algoritmu budeme z daného očíslovaného stromu

[2] Niekedy sa namiesto čísel používajú aj písmená, alebo akékoľvek iné symboly s daným lineárnym usporiadaním. V anglickej terminológii sa očíslovaný strom označuje ako *„labeled tree“* a v slovenskej terminológii sa občas používa aj pojem *„označený strom“*.



Obr. 12.1: Očíslované stromy

vytvárať Prüferov kód a pomocou druhého algoritmu budeme z daného Prüferovho kódu rekonštruovať pôvodný strom. V definícii 4.2.3 (strana 77) bol definovaný pojem *list* (stromu). V pseudokóde algoritmu na konštrukciu Prüferovho kódu budeme používať funkciu $\texttt{sused}(i)$, ktorá nám pre list so značkou i vráti značku jeho suseda. Keďže list je vrchol stromu, ktorého stupeň je 1, je táto funkcia dobre definovaná. Okrem toho budeme používať aj operáciu $\biguplus$, ktorá nám pridá jeden prvok na koniec už vytvorenej postupnosti. Napríklad:

$$(2,6,1,3,4) \biguplus 7 = (2,6,1,3,4,7)\,.$$

Konštrukcia Prüferovho kódu z daného očíslovaného stromu je veľmi jednoduchá. V každom kroku algoritmu nájdeme v strome list s najmenšou značkou, do kódu pridáme značku jeho suseda a tento list potom z pôvodného stromu odstránime aj s hranou, ktorá s ním inciduje. Toto opakujeme dovtedy, kým nám z pôvodného stromu nezostane len cesta dĺžky 1, t. j. len jedna hrana. Pseudokód algoritmu vytvárania Prüferovho kódu z očíslovaného stromu vyzerá takto

Prüferov kód z očíslovaného stromu

VSTUP: Očíslovaný strom $T = (V,E)$, kde $V = \{1,\ldots,n\}$.

VÝSTUP: Postupnosť $P = (p_1,\ldots,p_{n-2})$, kde každé $p_i \in \{1,\ldots,n\}$.

INICIALIZÁCIA

$P := ();$

VYTVÁRANIE PRÜFEROVHO KÓDU STROMU T

while $\{\, |P| < (n-2) \,\}$ **do**

  $i := \min\{\, x;\ x \text{ je list aktuálneho stromu } T \,\};$

  $P := P \biguplus \texttt{sused}(i);$

  MODIFIKÁCIA STROMU T

  $V := V \setminus \{i\};$

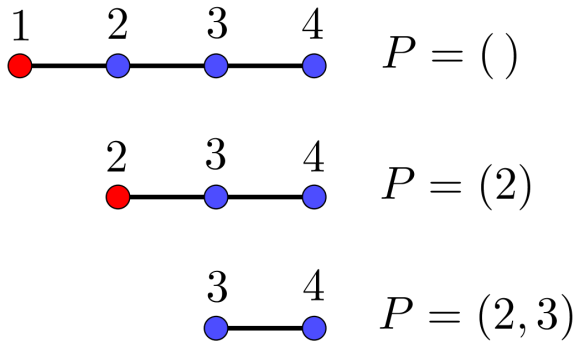
  $E := E \setminus \{i, \texttt{sused}(i)\};$

end

Tvorbu Prüferovho kódu z očíslovaného stromu si ilustrujeme na príkladoch.

■ **Príklad 12.7** Nájdite Prüferov kód očíslovaného stromu T_1 z obrázku 12.1 (strana 235).

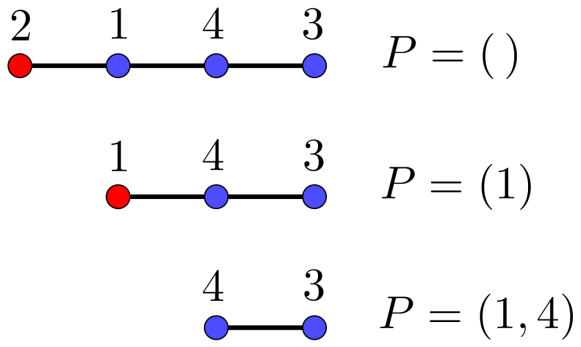
Riešenie: Strom T_1 má 4 vrcholy, takže bude mať dvojciferný Prüferov kód. Postup jeho tvorby je zobrazený na nasledovnom obrázku



List s najmenšou značkou je 1, takže prvou číslicou kódu bude značka jeho suseda, čo je 2. Potom vrchol 1, aj so s ním incidujúcou hranou, odstránime a pokračujeme ďalej. V druhom kroku je list s najmenšou značkou 2, takže druhá číslica kódu bude značka jeho suseda, čo je 3. Vrchol 2, aj so s ním incidujúcou hranou, odstránime a keďže nám zostala už len jedna hrana, skončili sme. Prüferov kód grafu T_1 z obrázku 12.1 je $P = (2,3)$. ■

■ **Príklad 12.8** Nájdite Prüferov kód očíslovaného stromu T_2 z obrázku 12.1 (strana 235).

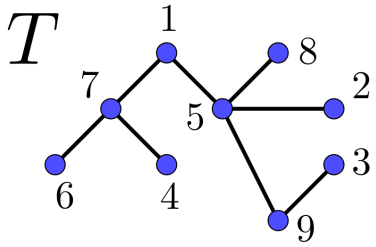
Riešenie: Strom T_2 má 4 vrcholy, takže bude mať dvojciferný Prüferov kód. Postup jeho tvorby je zobrazený na nasledovnom obrázku



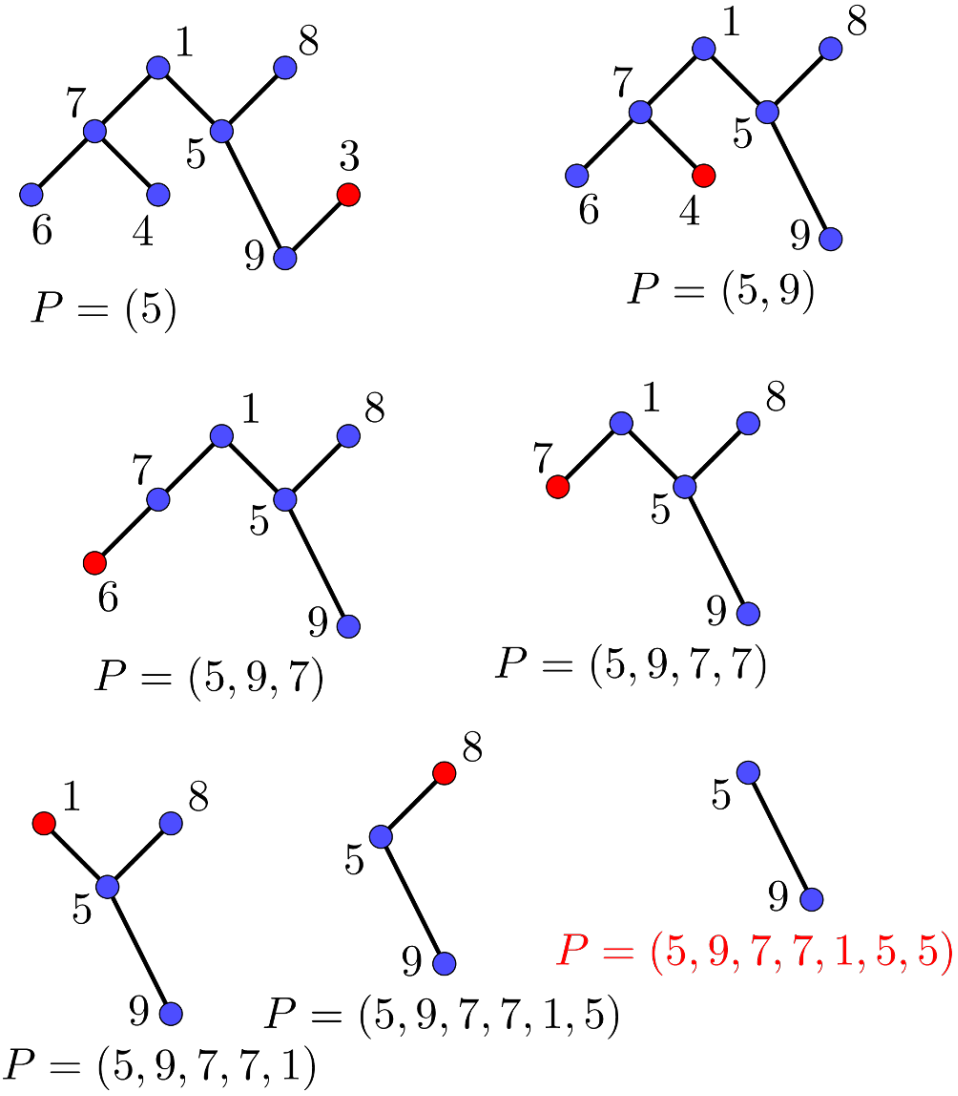
List s najmenšou značkou je 2, takže prvou číslicou kódu bude značka jeho suseda, čo je 1. Potom vrchol 2 aj so s ním incidujúcou hranou odstránime a pokračujeme ďalej. V druhom kroku je list s najmenšou značkou 1, takže druhá číslica kódu bude značka jeho suseda, čo je 4. Vrchol 1, aj so s ním incidujúcou hranou, odstránime, a keďže nám zostala už len jedna hrana, skončili sme. Prüferov kód grafu T_2 z obrázku 12.1 je $P = (1,4)$. ■

Na predošlých dvoch príkladov sme videli, že dva rôzne očíslované stromy, aj keď sú izomorfné, majú rôzne Prüferové kódy. Tvorbu Prüferovho kódu si ilustrujeme ešte na jednom príklade.

■ **Príklad 12.9** Nájdite Prüferov kód očíslovaného stromu T z nasledujúceho obrázku



Riešenie: Daný strom má 9 vrcholov, a preto jeho Prüferov kód bude usporiadaná sedmica. Postup riešenia máme zobrazený na nasledovnej sérii obrázkov. Séria sa začína stromom, ktorý už má odstránený vrchol 2 a s ním incidujúcu hranu. Listy stromu T budeme postupne odstraňovať v poradí



Obr. 12.2: Postup konštrukcie Prüferovho kódu stromu z príkladu 12.9

$2,3,4,6,7,1,8$. Samozrejme nie všetky uvedené vrcholy sú listy pôvodného stromu T. Vrcholy 7 a 1 sa stanú listami až po odstránení niektorých vrcholov pôvodného stromu. Výsledný Prüferov kód očíslovaného stromu T je usporiadaná sedmica $P = (5,9,7,7,1,5,5)$. ■

Ďalej si ukážeme, ako z $(n-2)$ číselného Prüferovho kódu zrekonštruujeme pôvodný n vrcholový očíslovaný strom. Najskôr si musíme uvedomiť dva fakty

1. Ak je Prüferov kód usporiadaná $(n-2)$-tica, tak pôvodný strom má n vrcholov so značkami $\{1,\ldots,n\}$.
2. Pre všetky $i \in \{1,\ldots,n\}$, ktoré sa nenachádzajú v Prüferovom kóde, je i značka listu pôvodného stromu.

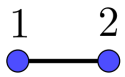
Pri rekonštrukcii očíslovaného stromu z Prüferovho kódu budeme postupovať presne opačným spôsobom, ako pri vytváraní Prüferovho kódu z očíslovaného stromu. Celý postup si najskôr ilustrujeme na príkladoch a až potom si zapíšeme pseudokód rekonštrukčného algoritmu. Budeme rekonštruovať očíslované stromy z príkladov 12.7, 12.8 a 12.9. Podrobný popis postupu rekonštrukcie uvedieme v texte príkladov.

■ **Príklad 12.10** Zrekonštruujte očíslovaný strom, ktorého Prüferov kód je $P = (2,3)$.

Riešenie: Prüferov kód je usporiadaná dvojica, takže pôvodný strom má 4 vrcholy so značkami $\{1,2,3,4\}$. Pri rekonštrukcii pôvodného očíslovaného stromu budeme používať pomocnú množinu M. Túto množinu budeme zapisovať ako usporiadanú k-ticu a jej prvky budeme písať vždy v lexikografickom usporiadaní. Na začiatku bude množina M obsahovať[3] všetky listy pôvodného stromu, čiže $M = (1,\ldots,n) \setminus P$, kde $P = (p_1,\ldots,p_{n-2})$ je Prüferov kód. Okrem toho budeme pracovať s množinou L, do ktorej budeme postupne pridávať „listy“, ktoré sme už spojili hranou so s nimi susediacimi vrcholmi. Úvodzovky sú v predošlej vete použité preto, lebo nie všetky vrcholy z množiny L budú listami pôvodného očíslovaného stromu. Pozri poznámku v riešení príkladu 12.9. Na začiatku bude množina L prázdna. Postup rekonštrukcie budeme písať po jednotlivých krokoch. Vždy si uvedieme aktuálny stav množín P, M a L a následne obrázok aktuálne zrekonštruovaného stromu.

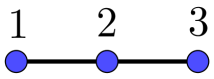
▷ **0. krok**
$P = (2,3) \qquad L = () \qquad M = (1,2,3,4) \setminus (2,3) = (1,4)$
List, ktorý bude spojený s vrcholom so značkou 2, čo je prvé číslo v Prüferovom kóde, musí byť list pôvodného stromu, ktorý mal najmenšiu značku. Čiže prvý vrchol z množiny M. V ďalšom kroku teda spojíme hranou vrcholy 1 a 2, vrchol 1 pridáme do množiny L, vrchol 2 vyhodíme z Prüferovho kódu P a novú množinu M dostaneme ako $M = (1,2,3,4) \setminus \big(P \cup L\big)$. List, ktorý budeme spájať s ďalším vrcholom z Prüferovho kódu, musí byť vrchol s najmenšou značkou, ktorá sa nenachádza v aktuálnom Prüferovom kóde, ani medzi „listami“, ktoré sme už spojili s ich susedom.

▷ **1. krok**
$P = (3) \qquad L = (1) \qquad M = (1,2,3,4) \setminus \big((3) \cup (1)\big) = (2,4)$



S vrcholom 3, čo je momentálne prvé číslo v Prüferovom kóde, bude spojený vrchol so značkou 2, pretože je to najmenšia značka, ktorá sa nenachádza ani v aktuálnom Prüferovom kóde, ani medzi už použitými listami.

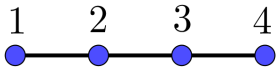
▷ **2. krok**
$P = () \qquad L = (1,2) \qquad M = (1,2,3,4) \setminus \big(\emptyset \cup (1,2)\big) = (3,4)$



Teraz už máme Prüferov kód prázdny. Je ale zrejmé, že z pôvodného stromu nám ešte chýba jedna hrana, čo je hrana, ktorá zostala po skončení tvorby Prüferovho kódu. Jeden vrchol tejto hrany musí byť posledné číslo, ktoré sa nachádzalo v Prüferovom kóde, čiže 3. A druhý vrchol tejto hrany je vrchol z množiny M, ktorý má najmenšiu značku rôznu od 3. Takže bude to vrchol 4.

[3] Nasledujúci zápis používa symbol odčítania množín, ale M aj P sú usporiadané množiny. Tejto drobnej nezrovnalosti sme si vedomí. Pri odčítaní budeme s M a P pracovať ako s množinami.

▷ **3. krok**
V poslednom kroku už len spojíme vrchol, ktorý bol ako posledný v Prüferovom kóde, s najmenším, od neho rôznym vrcholom v poslednej množine M.



■

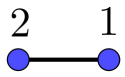
Na predošlom príklade sme videli, že v poslednom kroku rekonštrukcie stromu pridávame hranu, ktorá nám ako posledná zostala pri tvorbe Prüferovho kódu. Jedným vrcholom tejto hrany musí byť posledný vrchol, uvedený v Prüferovom kóde. V ďalších príkladoch preto budeme zapisovať množinu P tak, že jej posledný prvok zopakujeme dvakrát, čo nám zjednotí popis rekonštrukčného algoritmu pre všetky kroky, vrátane toho posledného. Čiže Prüferov kód P budeme zapisovať takto $P = (p_1, \ldots, p_{n-2}, \boldsymbol{p_{n-2}})$. Okrem toho pre väčšiu prehľadnosť už nebudeme písať ako sme dostali aktuálnu množinu M, čiže $M = (1, \ldots, n) \setminus (P \cup L)$, ale len jej výslednú podobu.

■ **Príklad 12.11** Zrekonštruujte očíslovaný strom, ktorého Prüferov kód je $P = (1,4)$.

Riešenie: Prüferov kód je usporiadaná dvojica, takže pôvodný strom má 4 vrcholy so značkami $\{1,2,3,4\}$. Postup riešenia bude rovnaký ako v predošlom príklade a opäť ho budeme zapisovať po krokoch.

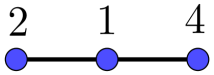
▷ **0. krok**
$P = (1,4,\boldsymbol{4})$ $L = ()$ $M = (2,3)$
V ďalšom kroku spojíme vrcholy 2 a 1.

▷ **1. krok**
$P = (4,\boldsymbol{4})$ $L = (2)$ $M = (1,3)$



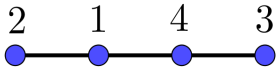
V ďalšom kroku spojíme vrcholy 1 a 4.

▷ **2. krok**
$P = (\boldsymbol{4})$ $L = (2,1)$ $M = (3)$



V ďalšom kroku spojíme vrcholy 3 a 4.

▷ **3. krok**
Po pridaní poslednej hrany, spojením vrcholov 3 a 4, sme dostali pôvodný očíslovaný strom.



■

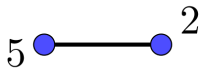
■ **Príklad 12.12** Zrekonštruujte očíslovaný strom, ktorého Prüferov kód je $P = (5,9,7,7,1,5,5)$.

Riešenie: Prüferov kód je usporiadaná sedmica, takže pôvodný strom má 9 vrcholov so značkami $\{1,2,3,4,5,6,7,8,9\}$. Na začiatku rekonštrukcie budeme mať množinu $P = (5,9,7,7,1,5,5,\boldsymbol{5})$, množina L bude prázdna a aktuálnu množinu M dostaneme vždy ako $M = (1, \ldots, 9) \setminus (P \cup L)$. V každom kroku spojíme a, prvý vrchol množiny M, s vrcholom b, prvým vrcholom množiny P. Potom vrchol a pridáme do množiny L, vrchol b vyhodíme z množiny P a určíme si novú množinu M. Jednotlivé kroky budú vyzerať takto

▷ **0. krok**
$P = (5,9,7,7,1,5,5,\boldsymbol{5})$ $L = ()$ $M = (2,3,4,6,8)$
V ďalšom kroku spojíme vrcholy 2 a 5.

▷ **1. krok**

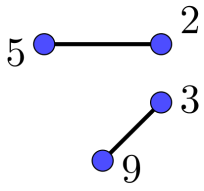
$P = (9,7,7,1,5,5,\mathbf{5})$ $L = (2)$ $M = (3,4,6,8)$



V ďalšom kroku spojíme vrcholy 3 a 9.

▷ **2. krok**

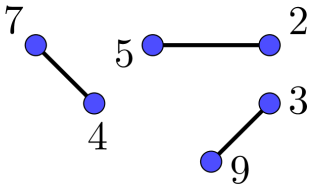
$P = (7,7,1,5,5,\mathbf{5})$ $L = (2,3)$ $M = (4,6,8,9)$



V ďalšom kroku spojíme vrcholy 4 a 7.

▷ **3. krok**

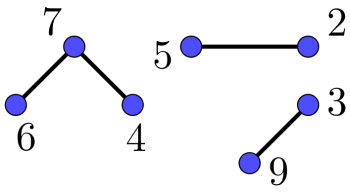
$P = (7,1,5,5,\mathbf{5})$ $L = (2,3,4)$ $M = (6,8,9)$



V ďalšom kroku spojíme vrcholy 6 a 7.

▷ **4. krok**

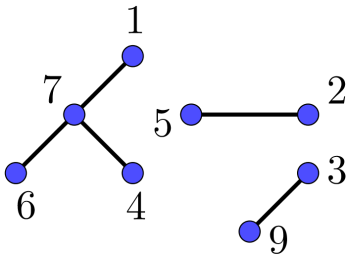
$P = (1,5,5,\mathbf{5})$ $L = (2,3,4,6)$ $M = (7,8,9)$



V ďalšom kroku spojíme vrcholy 7 a 1.

▷ **5. krok**

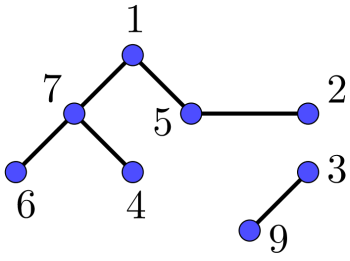
$P = (5,5,\mathbf{5})$ $L = (2,3,4,6,7)$ $M = (1,8,9)$



V ďalšom kroku spojíme vrcholy 1 a 5.

▷ **6. krok**

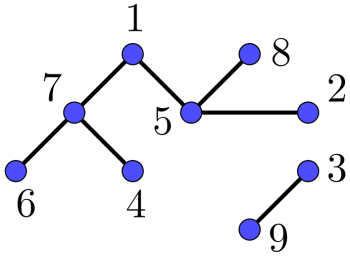
$P = (5,\mathbf{5})$ $L = (2,3,4,6,7,1)$ $M = (8,9)$



V ďalšom kroku spojíme vrcholy 8 a 5.

▷ **7. krok**

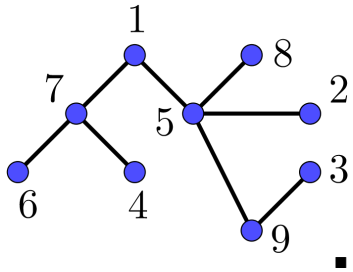
$P = (\mathbf{5})$ $L = (2,3,4,6,7,1,8)$ $M = (9)$



V ďalšom, už poslednom, kroku spojíme vrcholy 9 a 5, čím dostaneme pôvodný očíslovaný strom. V poslednom kroku, keď už množina P je prázdna, nepotrebujeme ani určovať novú množinu M.

▷ **8. krok**

$P = ()$ $L = (2,3,4,6,7,1,8,9)$ $M = (5)$



■

Pri popise pseudokódu konštrukcie očíslovaného stromu z Prüferovho kódu budeme používať funkciu `prvy` (P), ktorá nám z usporiadanej k-tice P vráti jej prvý prvok a zároveň tento prvok z P odstráni. Rovnakú funkciu sme používali aj v pseudokódoch Kruskalovho algoritmu (strana 202) a algoritmu na konštrukciu prehľadávacieho binárneho stromu (strana 215).
Korektnosť algoritmov konštrukcie očíslovaného stromu z Prüferovho kódu a naopak, sa dokazuje ľahko. Treba ukázať dve veci. Po prvé treba dokázať, že uvedeným algoritmom vždy vytvoríme očíslovaný strom a po druhé treba dokázať, že zo zrekonštruovaného stromu spätne dostaneme pôvodný Prüferov kód. Dôkaz korektnosti je uvedený napríklad v knihe [15] v časti 7.4.

Očíslovaný strom z Prüferovho kódu

VSTUP: Postupnosť $P = (p_1, \ldots, p_{n-2})$, kde každé $p_i \in \{1, \ldots, n\}$.

VÝSTUP: Očíslovaný strom $T = (V, E)$, kde $V = \{1, \ldots, n\}$.

INICIALIZÁCIA

$V := \{1, \ldots, n\}$;
$E := \{\}$;
$P := (p_1, \ldots, p_{n-2}, p_{n-2})$;
$L := \{\}$;
$M := V \setminus (P \cup L)$;

REKONŠTRUKCIA OČÍSLOVANÉHO STROMU T

while $\{ |P| > 0 \}$ **do**
  $i := \min\{ x;\ x \in M \}$;
  $j :=$ `prvy` (P);
  $E := E \cup \{i, j\}$;
  $L := L \cup \{i\}$;
  $M := V \setminus (P \cup L)$;
end

Ako sme mali možnosť vidieť z uvedených príkladov, konštrukcia Prüferovho kódu z očíslovaného stromu, rovnako ako rekonštrukcia očíslovaného stromu z Prüferovho kódu, je veľmi jednoduchá. Prüferov kód je pritom veľmi efektívny spôsob reprezentácie stromov. Autorom tohto kódu je nemecký matematik Heinz Prüfer, ktorý tento kód navrhol v roku 1918 v súvislosti s dôkazom Cayleyho[4] vety. Znenie Cayleyho vety je nasledovné.

[4] Arthur Cayley bol slávny britský matematik z 19. storočia, zaoberajúci sa diskrétnou matematikou, predovšetkým algebrou a teóriou grafov.

Veta 12.3.1 — Cayleyoho veta. Počet rôznych kostier očíslovaného kompletného grafu na n vrcholoch je n^{n-2}. Alebo inak sformulované:
Počet rôznych koreňových stromov s vrcholmi $V = \{1,\dots,n\}$ a koreňom 1 je n^{n-2}.

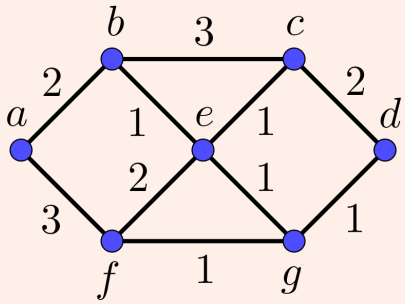
Dôkaz Cayleyho vety je s použitím Prüferovho kódu triviálny. Prenechávame ho ako cvičenie 12.8.

12.4 Cvičenia

Cvičenie 12.1 Majme ohodnotený graf G, ktorého všetky ohodnotnia hrán sú kladné. Nech s je najlacnejší sled medzi vrcholmi u a v v tomto grafe. Dokážte potom, že sled s je cesta, t. j. neopakujú sa v ňom ani hrany, ani vrcholy. ■

Cvičenie 12.2 Dokážte, že najlacnejší sled medzi dvoma vrcholmi daného grafu s kladnými ohodnoteniami hrán neobsahuje slučky. ■

Cvičenie 12.3



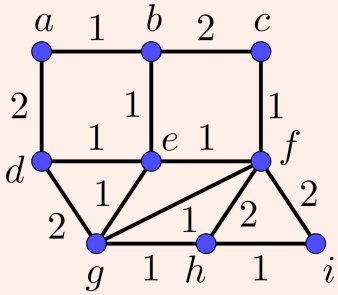
(a) Nájdite hodnoty najlacnejších ciest od vrcholu d ku všetkým ostatným vrcholom grafu.

(b) Nájdite všetky najlacnejšie cesty z vrcholu d do vrcholu a. ■

Cvičenie 12.4

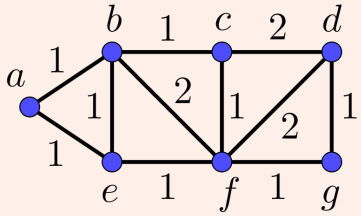
(a) Nájdite hodnoty najlacnejších ciest od vrcholu d ku všetkým ostatným vrcholom grafu.

(b) Nájdite všetky najlacnejšie cesty z vrcholu a do vrcholu g.



■

Cvičenie 12.5



(a) Nájdite hodnoty najlacnejších ciest od vrcholu a ku všetkým ostatným vrcholom grafu.

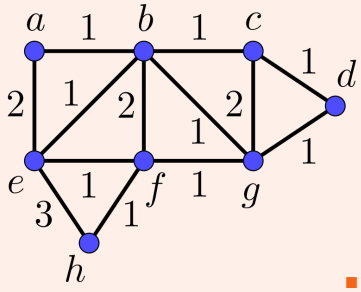
(b) Nájdite všetky najlacnejšie cesty z vrcholu a do vrcholu d. ■

Cvičenie 12.6 Naprogramujte Dijkstrov algorimus v Pythone alebo v C. ■

Cvičenie 12.7

(a) Nájdite hodnoty najlacnejších ciest od vrcholu a ku všetkým ostatným vrcholom grafu.

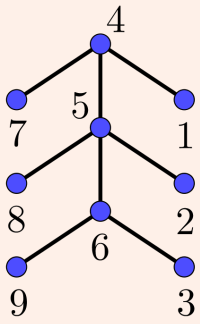
(b) Nájdite všetky najlacnejšie cesty z vrcholu a do vrcholov d a h.



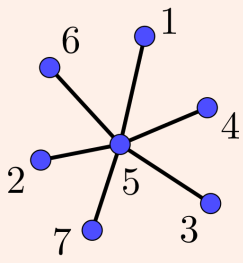
Cvičenie 12.8 Dokážte Cayleyho vetu (strana 242).
Návod: použite Prüferov kód.

Cvičenie 12.9 Nájdite Prüferov kód pre nasledujúce očíslované stromy

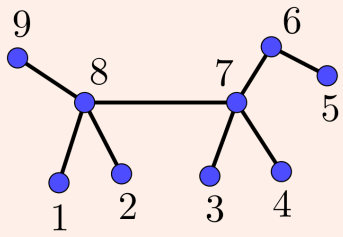
(a)



(b)



(c)



Cvičenie 12.10 Pre nasledujúce Prüferové kódy zrekonštruujte pôvodné očíslované stromy

(a) $(3,3,3,3,3)$
(b) $(1,2,3,4,5)$
(c) $(2,2,3,3,5)$
(d) $(4,4,4,2,2,2,3)$
(e) $(5,4,3,2,1)$
(f) $(1,2,2,3,3,3)$
(g) $(1,1,1,2,2,3)$
(h) $(3,2,1,4,2,4)$
(i) $(5,3,2,4,2,5,2)$

Cvičenie 12.11 Vrcholy cesty dĺžky 4 očíslujte číslami z množiny $\{1,\ldots,5\}$ tak, aby jej Prüferov kód obsahoval čo najmenej rôznych cifier.

Cvičenie 12.12 V Prüferovom kóde nejakého stromu, ktorý je usporiadanou päticou čísel, sa jedno číslo opakovalo trikrát a druhé dvakrát. Koľko má tento strom vrcholov, a aké sú ich stupne? Uveďte príklad takého stromu.

Cvičenie 12.13 Nájdite graf, v ktorom má každý vrchol stupeň aspoň 2 a ktorý má kostru získanú prehľadávaním do šírky izomorfnú

(a) s cestou dĺžky n,
(b) s hviezdou na n vrcholoch $\big($graf $K_{1,(n-1)}\big)$,
(c) so stromom, ktorého Prüferov kód je $(3,3,3,1,4,4,2)$,
(d) so stromom, ktorého Prüferov kód je $(2,1,3,1,4)$.

Cvičenie 12.14 Nájdite graf, v ktorom má každý vrchol stupeň aspoň 2 a ktorý má kostru získanú prehľadávaním do hĺbky izomorfnú

(a) s cestou dĺžky n,

(b) s hviezdou na n vrcholoch – graf $K_{1,(n-1)}$,

(c) so stromom, ktorého Prüferov kód je $(3,3,3,1,4,4,2)$,

(d) so stromom, ktorého Prüferov kód je $(2,1,3,1,4)$. ■

Cvičenie 12.15 Nájdite graf, ktorého každá kostra získaná prehľadávaním do šírky, je izomorfná so stromom, ktorého Prüferov kód je $(2,2,1,3,3,1,4,4)$. ■

Cvičenie 12.16 V Pythone alebo v C naprogramujte vytvorenie Prüferoveho kódu ku zadanému očíslovanému stromu. ■

Cvičenie 12.17 V Pythone alebo v C naprogramujte rekonštrukciu očíslovaného stromu zo zadaného Prüferoveho kódu. Strom môžete zapísať napríklad pomocou matice susednosti. ■

Použitá a odporúčaná literatúra

[1] Martin Aigner: **Combinatorial Theory**, SPRINGER VERLAG, HEIDELBERG 1979.

[2] Bohuslav Balcar, Petr Štěpánek: **Teorie množin**, ACADEMIA, PRAHA 1986.

[3] Berežný, Draženská, Kravecová: **Zbierka úloh z diskrétnej matematiky**, FEI TU, KOŠICE 2005. `http://web.tuke.sk/fei-km/sites/default/files/prilohy/10/Zbierka-DM.pdf`

[4] J. A. Bondy, U. S. Murty: **Graph Theory With Applications**, NORTH-HOLLAND 1976.

[5] Reinhard Diestel: **Graph Theory**, SPRINGER 2000.

[6] Harary, Frank: **Graph theory**, ADDISON-WESLEY 1969.

[7] Richard Johnsonbaugh: **Discrete Mathematics**, PEARSON 2017.

[8] Josef Kaucký: **Kombinatorické identity**, VEDA, VYDAVATEĽSTVO SAV, BRATISLAVA 1975.

[9] Martin Knor: **Úvod do matematickej logiky**, FIIT STU, BRATISLAVA 2016. `http://www.math.sk/jmkollar/literatura/Knor-Uvod_do_matematickej_logiky.pdf`

[10] Martin Knor: **Teória grafov**, SvF STU, BRATISLAVA 2008. `http://www.svf.stuba.sk/docs/dokumenty/skripta/teoria_grafov_martin_knor.pdf`

[11] Martin Knor, Jozef Kollár: **Matematika pre architektov**, STU BRATISLAVA, 2002.

[12] Donald E. Knuth: **The Art of Computer Programming, Volume 1**, ADDISON-WESLEY, 1997.

[13] Sergei K. Lando: **Lectures on Generating Functions**, AMERICAN MATHEMATICAL SOCIETY, STUDENT MATHEMATICAL LIBRARY 2003.

[14] Milan Mareš: **Příběhy matematiky**, PISTORIUS & OLŠANSKÁ, PŘÍBRAM 2011.

[15] J. Matoušek, J. Nešetřil: **Kapitoly z diskrétní matematiky**, KAROLINUM, PRAHA 2002.

[16] М. В. Меньшиков, А. М. Ревякин, А. Н. Копылова, Ю. Н. Макаров, Б. С. Стечкин: **Комбинаторный Анализ – Задачи и упражнения**, Издательство »НАУКА«, Москва 1982.

[17] В. А. Носов: **Комбинаторика и теория графов**, Москва 1999. `http://intsys.msu.ru/staff/vnosov/combgraph.htm` (PDF verzia)

[18] Stanislav Palúch: **Algoritmická teória grafov**, Žilinská univerzita, 2008. `http://frcatel.fri.uniza.sk/users/paluch/grafy.pdf`

[19] Ján Plesník: **Grafové algoritmy**, VEDA, Bratislava 1983.

[20] Franco P. Preparata, Raymond T. Yeh: **Úvod do téorie diskrétnych matematických štruktúr**, Alfa, Bratislava 1982.

[21] Edward M. Reingold, Jurg Nievergelt, Narsingh Deo: **Combinatorial Algorithms – Theory and Practice**, Prentice-Hall, New Jersey 1977.

[22] Beloslav Riečan a kol.: **Úlohy z matematiky pre 4. ročník gymnázia**, SPN, Bratislava 1976.

[23] Beloslav Riečan a kol.: **Matematika pre 4. ročník gymnázia**, SPN, Bratislava 1987.

[24] Beloslav Riečan a kol.: **Zbierka úloh z matematiky pre 4. ročník gymnázia**, SPN, Bratislava 1991.

[25] Karol Rovan a kol.: **Zbierka riešených úloh z algebry pre SVŠ a odborné školy**, SPN, Bratislava 1969.

[26] К. А. Рыбников: **Введение в Комбинаторный Анализ**, Издательство Московского Университета, Москва 1985.

[27] Raymond M. Smullyan: **A Beginner's Guide to Mathematical Logic**, Dover Publications, New York 2014.

[28] František Vejsada, František Talafous: **Zbierka úloh z matematiky pre SVŠ a gymnázia**, SPN, Bratislava 1973.

[29] Naum J. Vilenkin: **Rozhovory o množinách**, SPN, Bratislava 1972.

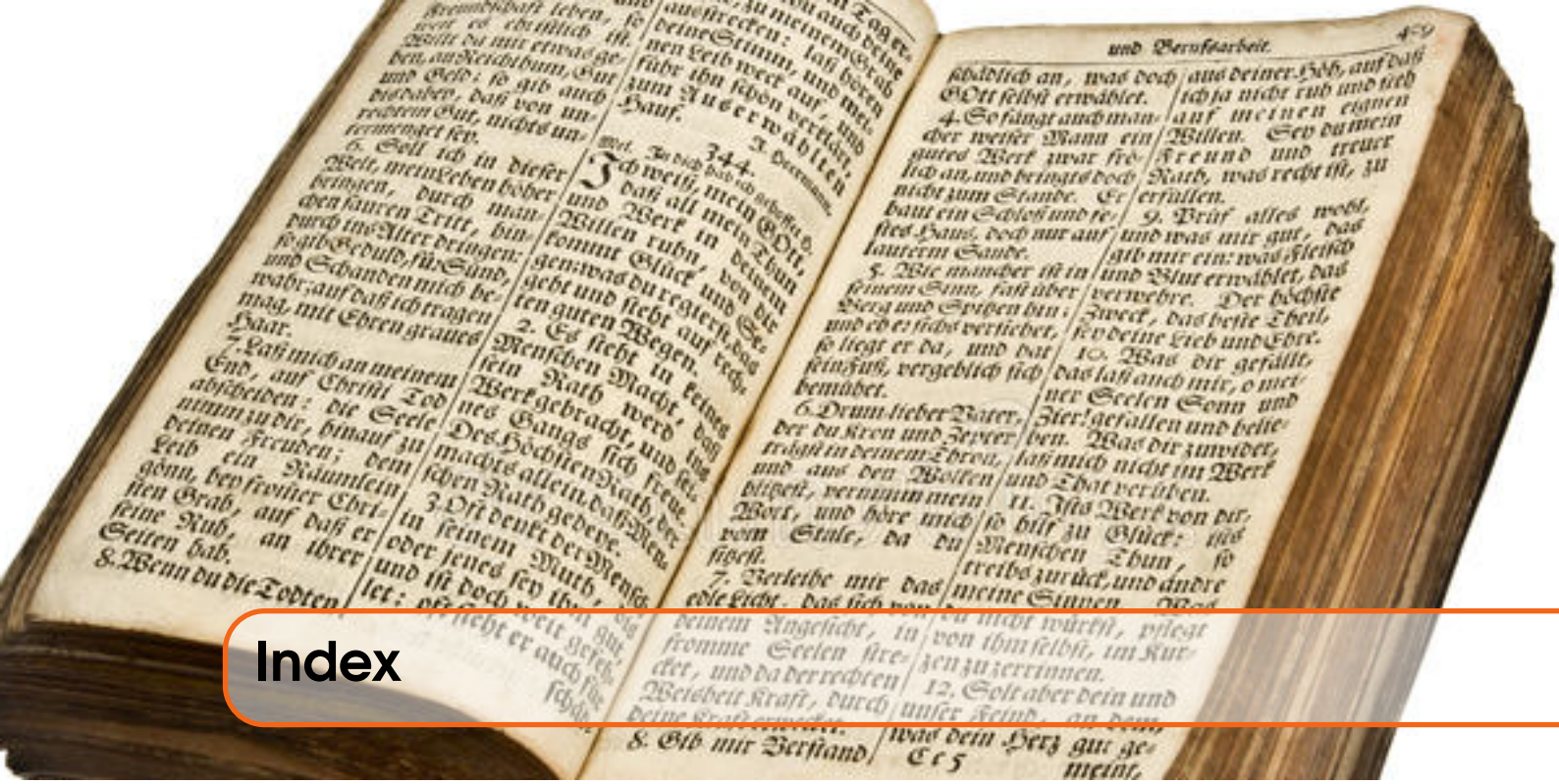
[30] Herbert S. Wilf: **generatingfunctionology**, Academic Press, Inc. 1994. `https://www.math.upenn.edu/~wilf/DownldGF.html`

[31] Niklaus Wirth: **Algoritmy a štruktúry údajov**, Alfa, Bratislava 1989.

[32] Štefan Znám: **Kombinatorika a teória grafov**, PF UK, Bratislava 1978.

IV

Index



Index

Symboly

A

B

C

Č

D

E

F

G

H

I

J

K

L

M

N

P

R

S

T

Ť

U

Ú

V

W

Z