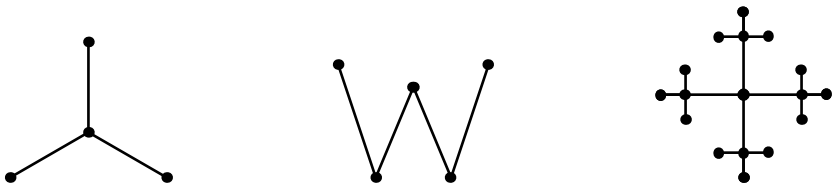
Kapitola 7

Stromy

Stromy jsou jednou z nejdůležitějších tříd grafů. O tom svědčí i množství vět, které je z různých pohledů charakterizují. Několik z nich dokážeme v této kapitole. Představíme také dvě praktické aplikace stromů: první se týká vyhledávání v lineárně uspořádané množině, druhá optimálního kódování symbolů z nějaké abecedy.

7.1 Definice

Definice 7.1 *Strom* je souvislý graf, který neobsahuje žádnou kružnici. *List* stromu T je libovolný vrchol, jehož stupeň v T je 1.



(a) (b) (c)

Obrázek 7.1: Stromy.

Několik příkladů stromů ukazuje obr. 7.1. Tyto stromy mají (zleva doprava) 3, 2 a 12 listů.

Tvrzení 7.2 *Má-li strom alespoň dva vrcholy, pak má alespoň dva listy.*

Důkaz. Vezměme cestu P maximální možné délky ve stromu T. Nechť x je koncový vrchol cesty P. Každý soused vrcholu x musí ležet na cestě P, jinak by bylo možné ji prodloužit. Dejme tomu, že x není list a má tedy alespoň dva

sousedy y_1, y_2. Sled, který dostaneme, pokud vyjdeme z vrcholu x do vrcholu y_1, dále po cestě P do y_2 a zpět do x, je kružnicí v grafu T. Ten je ale stromem a žádnou kružnici neobsahuje, takže x je skutečně list. Dalším listem je druhý koncový vrchol cesty P. □

V tomto oddílu si ukážeme tři důležité věty, z nichž každá charakterizuje stromy z jiného hlediska: věta 7.3 podle počtu cest mezi dvěma vrcholy, věta 7.4 podle počtu hran a konečně věta 7.6 podle nesouvislosti jistých podgrafů (tzv. faktorů).

Z definice je graf souvislý, obsahuje-li alespoň jednu cestu mezi každou dvojicí vrcholů. Pokud podmínku zesílíme a budeme požadovat existenci právě jedné cesty, pak podle následující věty budou této podmínce vyhovovat právě všechny stromy.

Věta 7.3 *Graf G je strom, právě když pro každé dva vrcholy $u, v \in V(G)$ existuje v grafu G právě jedna cesta z u do v.*

Důkaz. '⇒': Strom G je z definice souvislý, takže alespoň jedna cesta mezi každými dvěma vrcholy existuje. Nechť P, Q jsou dvě cesty z vrcholu u do vrcholu v, přičemž

$$P = (u = p_0, p_1, \ldots, p_k = v),$$
$$Q = (u = q_0, q_1, \ldots, q_\ell = v).$$

Nechť i je nejmenší index takový, že $p_i \neq q_i$. Vzhledem k tomu, že cesty P a Q jsou různé, ale začínají v témže vrcholu, platí $1 \leq i \leq k, \ell$. Zvolme j nejmenší takové, že $j > i$ a q_j leží na cestě P. Víme, že přinejmenším q_ℓ na P leží, proto takový index j jistě existuje. Definujme sled S následujícím způsobem: vyjdeme z vrcholu p_{i-1} po cestě P do p_j a odtud po cestě Q zpět do $q_{i-1} = p_{i-1}$. Je jasné, že S je kružnice v grafu G, což je spor s předpokladem, že G je strom. Mezi u a v je tedy jen jediná cesta.

'⇐': Graf G, ve kterém mezi každými dvěma vrcholy existuje nějaká cesta, je jistě souvislý. Pokud by obsahoval kružnici, vedly by mezi libovolnými dvěma vrcholy této kružnice alespoň dvě cesty. Graf G, který splňuje náš předpoklad, je tedy stromem. □

Podle věty 6.11 má souvislý graf na n vrcholech aspoň $n-1$ hran. Následující věta říká, že stromy lze charakterizovat jako grafy, u nichž v tomto dolním odhadu počtu hran platí rovnost.

Věta 7.4 *Graf G je strom, právě když je souvislý a má $n-1$ hran.*

Důkaz. '⇒': Strom je z definice souvislý. Ukážeme indukcí podle počtu vrcholů, že má $n-1$ hran. Pro strom na jednom vrcholu tvrzení jistě platí. Nechť je dán strom G s $n > 1$ vrcholy a nechť v je některý jeho list (existuje podle tvrzení 7.2).

Graf $G - v$ je strom (jak je vidět z definice) a má $n - 1$ vrcholů a $m - 1$ hran. Podle indukčního předpokladu je $m - 1 = (n - 1) - 1$ a tedy $m = n - 1$, což jsme chtěli dokázat.

'$\Leftarrow$': Potřebujeme dokázat, že souvislý graf s n vrcholy a $n - 1$ hranami neobsahuje kružnici. Opět použijeme indukci podle n, přičemž pro $n = 1$ je tvrzení triviální. Nechť souvislý graf G má $n - 1$ hran. Kdyby stupně všech vrcholů byly větší než 1, pak jejich součet je alespoň $2n$ a počet hran (který je přesně polovinou z tohoto čísla) by byl alespoň n. Proto G musí obsahovat nějaký vrchol stupně nejvýše 1. Stupeň 0 však díky souvislosti můžeme vyloučit. Nechť tedy v je vrchol stupně 1. Graf $G - v$ má $n - 1$ vrcholů, $n - 2$ hran a je souvislý. Podle indukčního předpokladu je to strom. Opětovným přidáním vrcholu v nemůže vzniknout kružnice, takže stromem je i celý graf G. □

Před uvedením poslední charakterizace stromů potřebujeme ještě jeden pojem. Obojí se nám bude hodit později, až budeme hovořit o souvislostech mezi grafy a maticemi.

Definice 7.5 *Faktor* grafu G je libovolný jeho podgraf, jehož množina vrcholů je $V(G)$. Faktor je *vlastní*, je-li různý od grafu G.

Věta 7.6 *Graf G je strom, právě když je souvislý a nemá žádný souvislý vlastní faktor.*

Důkaz. '$\Rightarrow$': Nechť strom G má souvislý vlastní faktor F. Protože $F \neq G$, existuje nějaká hrana $e \in E(G) - E(F)$. Podle věty 6.15 musí $G - e$ být nesouvislý graf, protože G neobsahuje žádnou kružnici. Graf F je ovšem faktorem grafu $G - e$ a musí tak být rovněž nesouvislý. To je spor.

'$\Leftarrow$': Nechť G je graf, který nemá souvislý vlastní faktor. Dejme tomu, že obsahuje kružnici C. Pro $e \in E(C)$ je opět podle věty 6.15 $G - e$ souvislý graf. Jedná se však o vlastní faktor grafu G, a to je spor. Vzhledem k tomu, že souvislost grafu G předpokládáme, je věta dokázána. □

Jak lze zjistit, zda je daný graf G stromem? Stejně jako u testování souvislosti je to patrné 'na první pohled' u malých grafů nakreslených přehledným obrázkem. Představíme-li si však třeba graf s desítkami tisíc vrcholů, navíc zadaný maticí, jak to uvidíme v kapitole 9, pak s prvním pohledem možná nevystačíme. Snadný postup však nabízí věta 7.4, podle níž stačí spočítat, zda má graf $n - 1$ hran, a otestovat, zda je souvislý (např. pomocí Dijkstrova algoritmu, se kterým se seznámíme v oddílu 12.2, nebo algoritmu ze cvičení 6.7). Není tedy potřeba zjišťovat, zda graf G obsahuje nějakou kružnici.

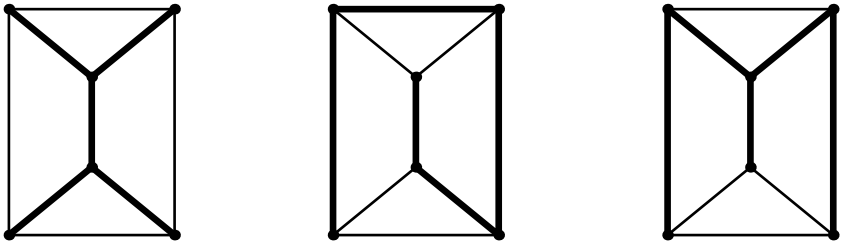
Cvičení

► **7.1** Kolik existuje různých stromů na množině $V = \{1, 2, 3, 4\}$? Kolik existuje navzájem neisomorfních stromů na množině V?

7.2 Kostry

Definice 7.7 *Kostrou* souvislého grafu G je každý jeho faktor, který je stromem. Jinak řečeno, kostra grafu G je souvislý podgraf bez kružnic, který obsahuje všechny vrcholy grafu G.

Obr. 7.2 ukazuje 3 různé kostry téhož grafu. Upozorněme, že dosti častou chybou je záměna pojmů kostra a strom. Rozdíl je ale podstatný: konkrétní graf může nebo nemusí být stromem, ale pojem kostra se vždy váže ještě k dalšímu grafu, např. v otázce 'Je graf G kostrou *grafu* H?'



Obrázek 7.2: Tři kostry téhož grafu (znázorněny tučně).

Má každý graf kostru? Z toho, že každý faktor nesouvislého grafu je nesouvislý, vidíme, že nesouvislý graf žádnou kostru mít nemůže. Na druhou stranu platí následující tvrzení.

Tvrzení 7.8 *Každý souvislý graf má alespoň jednu kostru.*

Důkaz. Nechť $G = (V, E)$ je souvislý graf. Uvažme množinu hran M, která má vlastnost, že

$$\text{faktor } (V, M) \text{ grafu } G \text{ neobsahuje kružnice} \tag{7.1}$$

a je s touto vlastností maximální vzhledem k inkluzi. (Taková množina existuje, protože například i prázdná množina splňuje podmínku (7.1).) Tvrdíme, že faktor (V, M) je souvislý.

Dejme tomu, že to tak není a uvažme komponenty A, B grafu (V, M). V souvislém grafu G mezi A a B jistě vede nějaká hrana e. Přidáním této hrany k množině M kružnice vzniknout nemůže, to by totiž vedlo ke sporu s větou 6.15. Proto množina $M \cup \{e\}$ má rovněž vlastnost (7.1) a dostáváme spor s maximalitou množiny M. □

Stromy a kostry mají řadu praktických aplikací. S některými z nich se v dalším textu seznámíme:

- *binární* a další stromy slouží jako datové struktury algoritmů (oddíly 7.3 a 7.4),

- *minimální kostra* se objevuje v úlohách, kde je cílem najít optimální propojení jistých objektů (oddíl 12.4),
- *rozhodovací strom* se používá k modelování rozhodovacích procesů (je popsán v oddílu 12.5).

Kostry grafů se rovněž uplatňují při analýze elektrických obvodů (tzv. metoda stavových proměnných).

Ve cvičeních k tomuto oddílu je zadáním spočítat kostry různých grafů. K této otázce se vrátíme v kapitole 9, kde odvodíme souvislost mezi počtem koster grafu a determinantem jisté matice.

Cvičení

▶ **7.2** Určete počet koster:

(a) stromu na n vrcholech,

(b) kružnice C_n,

(c) grafu, který vznikne, přidáme-li k C_{2n} hranu spojující dva vrcholy x, y s vlastností, že délka obou cest z x do y v grafu C_{2n} je n.

(Dvě kostry považujeme za různé, mají-li různé množiny hran.)

▶ **7.3** Nechť M_{2n} je graf tvořený n disjunktními hranami na $2n$ vrcholech. Určete počet koster grafu G, který vznikne přidáním nového vrcholu a jeho spojením s každým vrcholem z $V(M_{2n})$. (G je tedy n trojúhelníků slepených "za vrchol".)

▶▶ **7.4** Nechť G je graf na množině vrcholů $\{1, \ldots, n\} \cup \{v, w\}$, v němž je každý vrchol z $\{1, \ldots, n\}$ spojen hranou jak s v, tak s w, a jiné hrany G nemá. Určete

(a) počet koster grafu G,

(b) počet koster grafu $G + vw$, který vznikne z G přidáním hrany vw.

7.3 Binární stromy

Stromy mají široké uplatnění jako datové struktury pro různé algoritmy. Ukážeme si dva příklady použití tzv. binárních stromů. Uveďme nejprve jejich definici.

Kořenový strom je dvojice (T, r), kde T je strom a $r \in V(T)$ pevně zvolený vrchol, kterému budeme říkat *kořen*. *Hloubka* $h(w)$ vrcholu $w \in V(T)$ je délka cesty $P(w)$ z kořene r do w. Vrchol v je *rodičem* vrcholu w, pokud je jeho sousedem na cestě $P(w)$ (pak také řekneme, že w je *potomkem* vrcholu v). *Binární strom* je kořenový strom, v němž má každý vrchol nejvýše dva potomky a každý

potomek je označen jako *levý* nebo *pravý*. Žádný rodič samozřejmě nemá dva potomky stejného typu.

V obvyklém znázornění binárního stromu je kořen nahoře a levý resp. pravý potomek na příslušné straně pod svým rodičem. Příklady binárních stromů jsou na obr. 7.3.

(a) (b)

Obrázek 7.3: Binární stromy.

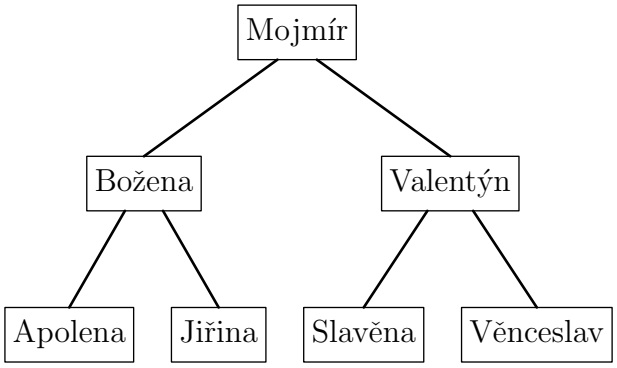
Binární stromy se dobře hodí k reprezentaci lineárních uspořádání. Dejme tomu, že potřebujeme implementovat algoritmus, který pracuje s abecedně seřazeným seznamem n osob, přičemž každé osobě odpovídá nějaký záznam. Seznam je obsáhlý a vzhledem k častým přístupům potřebujeme, aby čas nutný k nalezení konkrétního záznamu byl co nejmenší. Jako ilustraci použijme jména z jednoho únorového týdne v kalendáři. (Každý záznam obsahuje ještě další informace, kvůli kterým se vlastně vyhledávání provádí. Pro náš výklad však nejsou důležité.)

Apolena	Božena	Jiřina	Mojmír	Slavěna	Valentýn	Věnceslav
...	...	...	...	...	...	...

Jak efektivně najít záznam odpovídající určitému jménu? Jednou možností je procházet záznamy od začátku jeden po druhém a hledat ten správný. Pak ovšem v nejhorším případě musíme projít všech n záznamů (a v průměru $n/2$ záznamů).

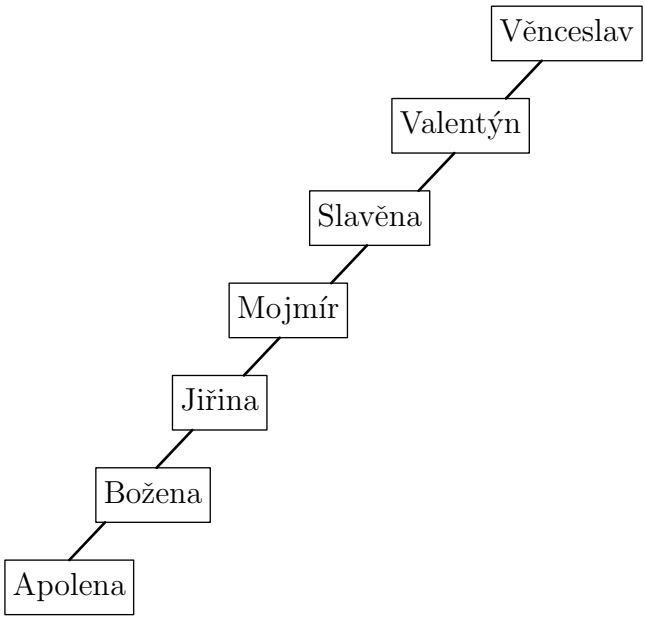
Tento čas však lze výrazně zkrátit s použitím vyhledávacího stromu. *Binární vyhledávací strom* pro lineárně uspořádanou množinu $(X, \leq)$ je binární strom, jehož vrcholy jsou prvky množiny X a v němž pro každý vrchol v platí, že je-li ℓ levý potomek vrcholu v, pak $\ell < v$, a podobně pro pravého potomka p vrcholu v je $p > v$. (Adjektivum *binární* zde budeme pro stručnost vynechávat.) V našem příkladu je množinou X množina záznamů, uspořádaná podle abecedy. Příklad vyhledávacího stromu pro množinu X je na obr. 7.4.

Označme jméno příslušné vrcholu v symbolem $a(v)$. Jak se v našem stromu vyhledá záznam pro konkrétní jméno x? Začneme od kořene r a porovnáme slova x a $a(r)$ (v abecedním uspořádání). Je-li $x = a(r)$, našli jsme hledaný záznam. Pokud $x < a(r)$, přejdeme k levému potomku vrcholu r, jinak k pravému potomku. Pokud příslušný potomek neexistuje, vyhledávání skončilo neúspěchem: hledaný prvek x není v množině X. Jinak se test opakuje.



Obrázek 7.4: Vyhledávací strom pro množinu záznamů.

Ve stromu na obr. 7.4 lze libovolný záznam vyhledat nejvýše ve třech krocích. Bez použití vyhledávacího stromu bychom k vyhledání jména Věnceslav potřebovali 7 kroků. Ne každý vyhledávací strom pro danou množinu nám však nabízí takové zlepšení. Například u stromu na obr. 7.5 potřebujeme v nejhorším případě rovněž 7 kroků.



Obrázek 7.5: Jiný vyhledávací strom.

Kolik kroků tedy trvá vyhledání záznamu v obecném vyhledávacím stromu? Definujme *hloubku* vyhledávacího stromu T jako maximální hloubku $h(v)$ nějakého jeho vrcholu. K vyhledání záznamu ve stromu hloubky d je pak v nejhorším případě potřeba d kroků.

Pro danou uspořádanou množinu ale máme na výběr z mnoha stromů. Jaké nejmenší hloubky lze dosáhnout? Řekneme, že *vyhledávací* strom pro množinu X je *optimální*, má-li mezi všemi vyhledávacími stromy pro X minimální hloubku.

Věta 7.9 *Hloubka $h(n)$ optimálního vyhledávacího stromu pro n-prvkovou množinu X je*

$$h(n) = \lceil \log_2(n+1) \rceil - 1, \tag{7.2}$$

kde $\lceil \log_2(\dots) \rceil$ označuje horní celou část z dvojkového logaritmu.

Důkaz. Nejprve dokažme, že $h(n)$ je větší nebo rovno výrazu na pravé straně nerovnosti (7.2). Nechť T je libovolný binární strom na n vrcholech. Označme jeho hloubku d. Pro $0 \leq i \leq d$ je počet vrcholů stromu T s hloubkou rovnou i nejvýše 2^i. Odtud

$$n \leq 2^0 + 2^1 + \cdots + 2^d = 2^{d+1} - 1,$$

takže $d \geq log_2(n+1) - 1$. Protože d je celé číslo, platí dokonce $d \geq \lceil log_2(n+1) - 1 \rceil$. Splňuje-li tuto nerovnost hloubka každého stromu na n vrcholech, musí platit i (7.2).

V opačném směru potřebujeme zkontruovat vyhledávací strom pro množinu X, jehož hloubka bude rovna pravé straně nerovnosti (7.2). Nejprve najdeme strom S, ve kterém

$$\text{každý vrchol o hloubce nejvýše } h(S) - 2 \text{ má oba potomky,} \tag{7.3}$$

a potom ukážeme, že strom S má požadovanou vlastnost.

Označme $X = \{x_1, \dots, x_n\}$, kde $x_1 < \cdots < x_n$. Je-li $n \leq 2$, pak snadno najdeme strom pro X s hloubkou 0 nebo 1, který triviálně splňuje podmínku (7.3). Pro větší n za kořen stromu S zvolme *medián* posloupnosti $x_1, \dots, x_n$, tj. prvek x_m, kde $m = \lceil n/2 \rceil$ (medián je obecně 'prostřední' prvek monotónní posloupnosti). Pro uspořádané množiny $X_1 = \{x_1, \dots, x_{m-1}\}$ a $X_2 = \{x_{m+1}, \dots, x_n\}$ rekurzívním způsobem zkonstruujme binární vyhledávací stromy S_1 a S_2 s vlastností (7.3). Strom S získáme tak, že levým potomkem kořene x_m učiníme kořen stromu S_1 a pravým potomkem kořen stromu S_2. Vidíme, že podmínka (7.3) zůstane zachována.

Zbývá určit hloubku $d = h(S)$ stromu S. Z podmínky (7.3) plyne, že pro $0 \leq i \leq d-1$ obsahuje strom S přesně 2^i vrcholů o hloubce i. Má tedy přesně

$$1 + 2 + 2^2 + \cdots + 2^{d-1} = 2^d - 1$$

vrcholů o hloubce menší než d. Z definice navíc musí obsahovat alespoň jeden vrchol o hloubce d, takže

$$n \geq 2^d.$$

Po přičtení jedničky k oběma stranám a zlogaritmování snadno zjistíme, že platí

$$d \leq \lceil \log_2(n+1) \rceil - 1,$$

čímž je důkaz hotov. □

Důsledkem věty 7.9 je, že s použitím vhodného vyhledávacího stromu lze v n-prvkové uspořádané množině vyhledávat v počtu kroků, který je omezen *logaritmickou* funkcí n. (S použitím formalismu z oddílu 6.7 bychom řekli[1], že počet kroků je $O(\log n)$.) Oproti *lineárnímu* počtu kroků bez použití vyhledávacího stromu jde o značné zlepšení.

V našem příkladu předpokládáme, že vyhledávací strom je *statický*: stačí jej jednou zkonstruovat a již se nemění. Existují také algoritmy, pomocí kterých lze do vyhledávacích stromů přidávat vrcholy nebo je odebírat, a to při zachování efektivity daného stromu. Více se o těchto tématech lze dozvědět např. v knize [3].

Cvičení

►► **7.5** (a) Určete součet

$$\sum_{k=1}^{n} k \cdot 2^k.$$

(b) Vyhledávací strom T o hloubce d je *úplný*, pokud všechny listy mají hloubku d a každý ostatní vrchol má 2 potomky. Kolik vrcholů má takový strom?

(c) S použitím části (a) spočítejte *průměrný* počet kroků potřebný k vyhledání záznamu v úplném vyhledávacím stromu o hloubce d. (Průměr se počítá přes všechny vrcholy stromu.)

7.4 Huffmanovo kódování

Dostáváme se k další aplikaci binárních stromů, k tzv. Huffmanovu[2] kódování. Nechť je dána konečná *abeceda* Σ. Její prvky budeme nazývat *symboly*. Naším cílem bude zakódovat každý symbol *binárním řetězcem* (posloupností nul a jedniček, tzv. *bitů*), tak, aby bylo splněno několik podmínek, k jejichž zformulování se dostaneme za chvíli. *Kódování* pro abecedu Σ je prostá funkce, která přiřazuje každému symbolu a konečnou posloupnost $c(a)$ symbolů 0 a 1 (*kód symbolu* a).

Nechť je pevně zvoleno kódování c. Pak lze definovat i kód posloupnosti symbolů $a_1 \ldots a_k$, a to jako zřetězení kódů

$$c(a_1) \ldots c(a_k).$$

[1] Všimněme si, že v tomto asymptotickém vyjádření nezáleží na základu logaritmu. Můžeme jej tedy vynechat.

[2] David A. Huffman (1905–1999).

Je z tohoto kódu možné rekonstruovat původní posloupnost $a_1 \dots a_k$? Ne vždy. Uvažme například abecedu $\Sigma = \{X, Y, Z\}$ s následujícím kódováním:

symbol	X	Y	Z
kód	0	1	01

Pak např. posloupnosti Z a XY mají stejný kód 01. Této situaci bychom se chtěli vyhnout. To se nám podaří například tehdy, když bude naše kódování *prefixové*, tj. když kód žádného symbolu nebude počátečním úsekem ('prefixem') kódu žádného jiného symbolu.

Pozorování 7.10 *Prefixové kódování přiřazuje různým posloupnostem symbolů různé kódy.*

Důkaz. Cvičení 7.6. □

Prefixové kódování se dá najít snadno: stačí symbolům přiřadit různé kódy stejné délky ℓ.

Nyní se dostáváme k dalšímu požadavku: chtěli bychom minimalizovat průměrnou délku kódu přiřazeného symbolu abecedy Σ. Předpokládáme přitom, že každý symbol a má určenou *váhu* $w(a)$, na kterou se můžeme dívat jako na poměrnou četnost symbolu a v 'typickém textu' nad abecedou Σ a která se bere v úvahu při výpočtu uvedeného průměru.

Jinak řečeno, označíme-li délku binárního řetězce r symbolem $|r|$, budeme chtít, aby *vážená délka* kódování c,

$$\mathrm{vd}(c) = \sum_{a \in \Sigma} w(a) \cdot |c(a)|, \tag{7.4}$$

byla co nejmenší.

Kódování c pro abecedu Σ je *optimální*, pokud je prefixové a má mezi všemi prefixovými kódováními pro Σ nejmenší váženou délku.

Uvažme jako jednoduchý příklad abecedu $\Sigma = \{A, B, C, D\}$. Jsou-li všechny váhy stejné, pak nelze počítat s lepším řešením, než je kódování s kódy 00, 01, 10 a 11 a s váženou délkou 2 (obecně pro n-prvkovou abecedu bychom potřebovali $\lceil \log_2 n \rceil$ bitů). Předpokládejme ovšem, že symboly mají následující váhy:

symbol	A	B	C	D
váha	0.6	0.3	0.05	0.05

(7.5)

Pak není vyloučeno, že se nám na vážené délce podaří ušetřit, pokud častému symbolu A přiřadíme nejkratší možný kód, i za ceny prodloužení kódů málo frekventovaných symbolů C a D. Skutečně, například kódování

symbol	A	B	C	D
kód	0	10	110	111

je prefixové a jeho vážená délka je pouze

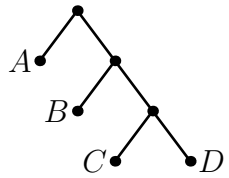
$$0.6 \times 1 + 0.3 \times 2 + 0.05 \times 3 + 0.05 \times 3 = 1.5.$$

S tímto kódováním tedy na jeden symbol potřebujeme průměrně pouze 1.5 bitu! Dá se ukázat, že je to optimální kódování.

Prefixová kódování mají úzký vztah k binárním stromům. Nechť T je binární strom, jehož listům jsou vzájemně jednoznačně přiřazeny symboly z abecedy Σ. Označme list odpovídající symbolu $a \in \Sigma$ jako ℓ_a. Strom T určuje kódování pro abecedu Σ, a to následujícím způsobem. Určíme kód $c(a)$ symbolu $a \in \Sigma$. Označme vrcholy na cestě $P(\ell_a)$ v pořadí od kořene jako $r = v_0, v_1, \ldots, v_d = \ell_a$. Hledaný kód pak bude posloupnost $c(a) = (b_1, \ldots, b_d)$, kde d je hloubka listu ℓ_a, všechny b_i jsou z množiny $\{0, 1\}$ a

$$b_i = 0 \iff \text{vrchol } v_i \text{ je levým potomkem vrcholu } v_{i-1}.$$

V opačném směru lze pro každé prefixové kódování sestrojit odpovídající binární strom (viz cvičení 7.11). Například našemu optimálnímu kódování pro abecedu $\{A, B, C, D\}$ odpovídá strom na obr. 7.6. Pokusme se přeformulovat úlohu hledání optimálního kódování v řeči binárních stromů.



Obrázek 7.6: Binární strom odpovídající optimálnímu kódování pro abecedu $\{A, B, C, D\}$ s váhami 7.5.

Optimální strom pro abecedu $\Sigma = \{a_1, \ldots, a_n\}$ s váhami $w(a_i)$ je binární strom T s listy $\ell_1, \ldots, \ell_n$ a s nejmenší možnou *průměrnou váženou hloubkou* $\mathrm{vh}(T)$. Ta je definována předpisem

$$\mathrm{vh}(T) = \sum_{i=1}^{n} w(a_i) \cdot h(\ell_i),$$

kde symbol $h(\ell_i)$ označuje hloubku listu ℓ_i. Je jasné, že optimálnímu stromu odpovídá optimální kódování pro abecedu Σ s danými váhami. (Pozor na rozdíl mezi pojmem 'optimální strom' a termínem 'optimální vyhledávací strom' z oddílu 7.3.)

Než popíšeme Huffmanův algoritmus pro nalezení optimálního stromu, zaveďme ještě dvě definice. Vrchol v binárního stromu T je *následovníkem* vrcholu u, pokud u leží na cestě $P(v)$. Speciálně je tedy každý vrchol svým vlastním

následovníkem. *Levý (pravý) podstrom* pod vrcholem v je indukovaný podgraf stromu T na množině všech následovníků levého (pravého) potomka vrcholu v.

Vstupem *Huffmanova algoritmu* je abeceda $\Sigma = \{a_1, \ldots, a_n\}$ spolu s přiřazením vah $w(a_i)$ symbolům $a_i \in \Sigma$. Jeho výstupem je (některý) optimální strom H pro abecedu Σ, tzv. *Huffmanův strom*. Příslušné kódování se označuje jako *Huffmanovo kódování*.

Algoritmus pracuje s posloupností kořenových stromů s váženými listy, která se v každém kroku mění. Výchozí posloupnost $P_0 = (T_1, \ldots, T_n)$ je tvořena triviálními stromy T_i o jediném vrcholu a_i s vahou $w(a_i)$.

Algoritmus skončí, až bude naše pracovní posloupnost obsahovat jediný strom, a tím bude hledaný Huffmanův strom H. Popišme k-tý krok algoritmu. Váha $w(T)$ libovolného stromu T je definována jako součet vah jeho listů.

Najdeme v pracovní posloupnosti P_{k-1} dvojici stromů s minimálním součtem vah. Takových dvojic může být více; abychom se vyhnuli nejednoznačnosti, definujme množinu uspořádaných dvojic

$$M_{k-1} = \{(s,t) : s < t \text{ a součet } w(T_s) + w(T_t) \text{ je minimální možný}\}$$

a najděme v ní nejmenší prvek (i,j) v lexikografickém uspořádání (viz cvičení 3.2). Posloupnost P_k vznikne z posloupnosti P_{k-1} následovně:

(1) vypustíme strom T_j,

(2) nahradíme strom T_i stromem, jehož kořenem je nově přidaný vrchol v_k, levým podstromem pod kořenem je T_i a pravým podstromem pod kořenem je T_j (váhy na listech zůstávají beze změny).

Vidíme, že počet prvků pracovní posloupnosti stromů se provedením k-tého kroku snížil o 1. Algoritmus tedy skončí po $p-1$ krocích.

Ilustrujme jeho průběh na příkladu. Nechť jsou pro abecedu

$$\Sigma = \{A, E, G, I, L, R\}$$

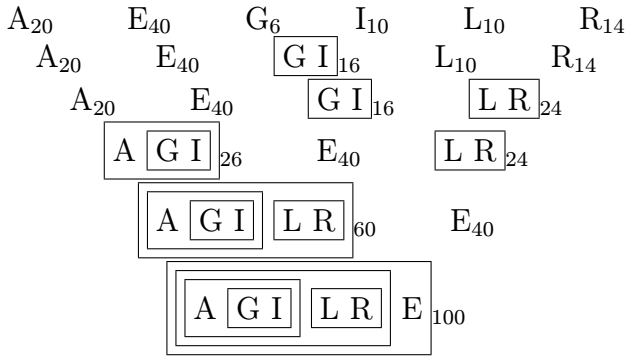
určeny tyto váhy:

symbol	A	E	G	I	L	R
váha	0.2	0.4	0.06	0.1	0.1	0.14

(7.6)

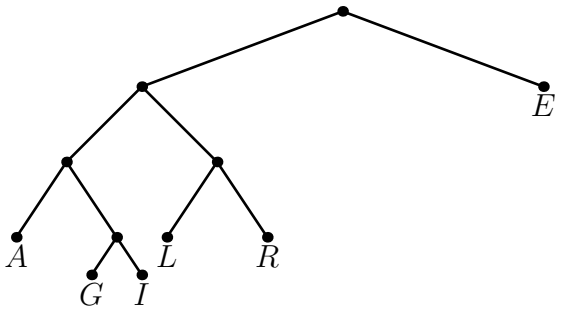
Najdeme Huffmanův strom pro tuto abecedu.

V následujícím schématickém znázornění průběhu Huffmanova algoritmu každý řádek zachycuje stav pracovní posloupnosti stromů v příslušném kroku. Písmena odpovídají stromům na jednom vrcholu. Strom S na více vrcholech je znázorněn symbolem $\boxed{S_1 S_2}$, kde S_1 (resp. S_2) je levý (resp. pravý) podstrom pod kořenem stromu S. Prvky posloupnosti jsou odděleny mezerami, čísla (indexy) udávají váhu daného stromu, pro přehlednost násobenou stem.



Výsledný Huffmanův strom (který odpovídá poslednímu řádku v tomto zápisu) je na obr. 7.7. Našli jsme tedy následující optimální kód:

symbol	A	E	G	I	L	R
kód	000	1	0010	0011	010	011



Obrázek 7.7: Huffmanův strom pro abecedu s váhami (7.6).

Pomocí této tabulky není problém zakódovat libovolné slovo v abecedě Σ. Například kód slova ALERGIE je 0000101011001000111.

Korektnost Huffmanova algoritmu zaručuje následující věta, jejíž důkaz ponecháváme jako (těžší) problém na cvičení 7.12.

Věta 7.11 *Nechť $\Sigma = \{a_1, \ldots, a_n\}$ je abeceda s váhami $w(a_i)$. Strom zkonstruovaný Huffmanovým algoritmem je optimálním stromem pro tuto abecedu.* □

Cvičení

▶ **7.6** Dokažte pozorování 7.10.

▶ **7.7** Určete průměrnou váženou hloubku $\mathrm{vh}(T)$ stromu na obr. 7.7.

▶ **7.8** Zakódujte pomocí Huffmanova kódování z našeho příkladu slova ELEGIE a LILIE. Vyplatí se pro tato slova Huffmanovo kódování použít? Proč?

► **7.9** Určete slovo, jehož Huffmanův kód v kódování z našeho příkladu je

$$01000110010000.$$

► **7.10** Najděte Huffmanův strom pro abecedu $\{A, B, C, D, E\}$ s váhami

symbol	A	B	C	D	E
váha	0.1	0.15	0.5	0.1	0.15

a určete jeho průměrnou váženou hloubku.

► **7.11** Ukažte, že prefixová kódování pro abecedu Σ jsou ve vzájemně jednoznačném vztahu s binárními stromy, jejichž listy jsou označeny symboly abecedy Σ.

►► **7.12** Dokažte větu 7.11.
[*Nápověda.* Nechť a_i, a_j jsou dva symboly s minimální vahou. Ukažte, že pro abecedu Σ existuje optimální strom, ve kterém symboly a_i a a_j mají stejného předchůdce. Z tohoto lemmatu plyne věta 7.11 indukcí přes n.]