



# Úvod do Teorie grafů

Petr Kovář

Text byl vytvořen v rámci realizace projektu *Matematika pro inženýry 21. století* (reg. č. CZ.1.07/2.2.00/07.0332), na kterém se společně podílela Vysoká škola báňská – Technická univerzita Ostrava a Západočeská univerzita v Plzni




INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Petr Kovář
Úvod do Teorie grafů

# Předmluva

Text, který právě čtete, vznikal nejprve jako příprava přednášek i cvičení předmětu *Diskrétní matematika a úvod do Teorie grafů (DiM)* pro bakalářské studium na technické vysoké škole. Při výběru témat a přípravě textu jsem vycházel především z osnov předmětu, jak byly navrženy pro studenty oborů se zaměřením na informatiku a další technické obory. Čerpal jsem ze skript *Diskrétní matematika* Petra Hliněného [4], z knihy *Kapitoly z diskrétní matematiky* od Jiřího Matouška a Jaroslava Nešetřila [7], z knihy *Discrete Mathematics and Its Applications* od K. Rosena [8], z knihy *Introduction to Graph Theory* od Douglase B. Westa [9] a celé řady dalších učebnic a článků. Snažil jsem se, aby text byl jednak přehledný a současně přesný a přitom čtivý a především aby obsahoval dostatek motivačních problémů, které více či méně odpovídají reálným úlohám, s nimi se absolventi mohou setkat a k jejichž řešení lze použít teorii grafů.

Velký dík patří Bohumilu Krajcovi, který celý text pečlivě prošel a upozornil na řadu nepřesností a výrazně pomohl vylepšit kvalitu výsledného textu. Pokud máte pocit, že v textu je přesto nějaká nesrovnalost, dejte mi vědět. Budu rád, když mne upozorníte i na méně srozumitelné pasáže, abych je v dalších verzích textu mohl vylepšit.

Text byl vysázen pomocí sázecího systému TEX ve formátu pdfLATEX.

V Ostravě 31. 8. 2011 Petr Kovář

# Obsah

# Kapitola 1

# Pojem grafu

Teorie grafů je jednou z mladších disciplin matematiky. Zatímco počátky aritmetiky nebo geometrie se ztrácí hluboko před počátkem letopočtu, tak začátky teorie grafů klademe poměrně přesně do 30. let 18. století. V roce 1736 švýcarský a později pruský matematik Leonhard Euler vyřešil tak zvaný „Problém sedmi mostů města Královce“. Při řešení nevyužil metody žádné z tehdy známých disciplin matematiky, ale použil (jak sám napsal v dopise svému příteli) „geometrii pozic“, dnes bychom řekli teorii grafů. Za zakladatele teorie grafů proto považujeme Eulera a rok 1736 za rok vzniku teorie grafů. První monografii a současně učebnici teorie grafů („Theorie der endlichen und unendlichen Graphen“) vydal maďarský matematik Dénes Kőnig o dvě stě let později, v roce 1936.

Dnes našla teorie grafů jako významná část diskrétní matematiky své uplatnění při řešení mnoha praktických úloh. Její výhodou je snadná a intuitivní představa, která modeluje reálnou situaci jako množinu objektů (vrcholy grafu) a vztahy mezi nimi (hrany grafu). Převedeme-li konkrétní úlohu do řeči teorie grafů, často se jedná o již řešený obecný problém a můžeme použít známe algoritmy při implementaci řešení. Bezesporu zásadní výhodou je snadná implementace objektů a postupů teorie grafů v počítači.

## 1.1 Graf

### Průvodce studiem


*V této sekci zavedeme pojem grafu jako dvojici množin: množinu objektů, kterým říkáme vrcholy grafu, a množinu hran, jež reprezentují vztahy mezi objekty. Ukážeme několik způsobů, jak graf může modelovat reálnou situaci. Upozorníme na omezení zavedeného pojmu graf, na analogie a rozdíly mezi pojmy „graf“ a „relace“ a zmíníme také obecnější struktury než je „graf“.*

*Nakonec zavedeme několik běžných tříd grafů, se kterými se často setkáme v dalším textu.*

## Cíle

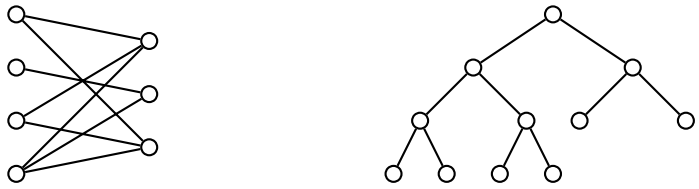
Po prostudování této sekce budete schopni:

- nadefinovat graf jako algebraickou strukturu,
- vysvětlit rozdíl mezi grafem a jeho nakreslením,
- popsat základní typy grafů.

Pojem grafu je jedním z ústředních pojmů současné diskrétní matematiky. Hned na úvod je nutno zdůraznit, že grafem *nebudeme rozumět graf funkce*! V diskrétní matematice grafem rozumíme algebraickou strukturu, která přehledně popisuje objekty a vztahy mezi nimi.

Neformálně si graf můžeme představit jako několik objektů (říkám jim vrcholy) a jejich spojnice (hrany). Nespornou výhodou grafů je jejich přehledné a intuitivní znázornění, kdy vrcholy nakreslíme jako body nebo puntíky v rovině a hrany zakreslíme jako křivky spojující příslušné vrcholy.

Současně je jasné, že při pečlivém rozboru komplikovaných úloh a zejména při implementaci algoritmů pro jejich řešení nevystačíme s neformálně zavedenou strukturou grafů jako puntíků a čar v rovině. Všimněte si, jak následující definice vystihne podstatu struktury grafu: objekty a jejich vazby. Jestliže mezi dvěma objekty je vazba, popíšeme ji jako dvojici (dvouprvkovou množinu) příslušných objektů (vrcholů). Naopak, jestliže mezi dvěma objekty vazba není, příslušnou dvojici objektů do množiny hran nezařadíme.



Obrázek 1.1 Ukázky grafů.

**Definice 1.1.** *Graf* $G$ (také *jednoduchý* graf nebo *obyčejný* graf) je uspořádaná dvojice $G = (V, E)$, kde $V$ je neprázdná množina *vrcholů* a $E$ je množina *hran* – množina (některých) dvouprvkových podmnožin množiny $V$.

V dalším textu budeme pracovat s konečnými grafy, tj. s takovými grafy, které mají konečnou a neprázdnou množinu vrcholů. Množinu vrcholů daného grafu $G$ budeme značit $V(G)$ a množinu jeho hran $E(G)$. Proto například $|V(G)|$ je číslo, které udává počet vrcholů grafu $G$ a $|E(G)|$ je počet hran grafu $G$.

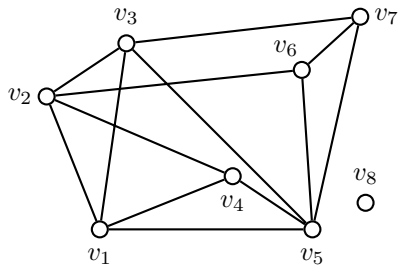
Vrcholy grafu značíme obvykle malými písmeny z konce abecedy (například $u, v, w, \dots$). Hrany jsou dvouprvkové podmnožiny $V$, například $\{u, v\}$ nebo $\{u, w\}$. Z praktických důvodů budeme hrany značit zkráceně $uv$ nebo $uw$, mějme však na paměti, že $uv$ je jen zkratkou zápisu $\{u, v\}$. Zejména má smysl říkat, že „vrcholy $u$ a

$v$ patří hraně“ $uv$, nebo že vrcholy $u$ a $v$ jsou „incidentní s hranou $uv$.“ *Incidence* je vztah mezi vrcholem a hranou, říkáme také, že „hrana je incidentní se svými koncovými vrcholy.“

Dále říkáme, že vrcholy $u$ a $v$ jsou *spojené hranou* $uv$ nebo že jsou *sousední* v grafu $G$, jestliže hrana $uv \in E(G)$. Pokud množina $E$ hranu $uv$ neobsahuje, tak vrcholy $u$, $v$ jsou *nesousední* nebo *nezávislé*. Naopak pracujeme-li s hranou $uv$, tak vrcholy $u$ a $v$ nazýváme *koncové* vrcholy hrany $uv$.

### Graf a jeho nakreslení

Graf můžeme znázornit nebo dokonce zadat vhodným obrázkem. Jedno takové *nakreslení* grafu je na Obrázku 1.2. Snadno ověříte (a to doporučujeme jako snadné cvičení), že graf $G$ na Obrázku 1.2 popisuje strukturu $G = (V, E)$, kde množina vrcholů je $V = \{v_1, v_2, \ldots, v_8\}$ a množina hran (dvouprvkových podmnožin množiny $V$) je $E = \{v_1v_2, v_1v_3, v_1v_4, v_1v_5, v_2v_3, v_2v_4, v_2v_6, v_3v_5, v_3v_7, v_4v_5, v_5v_6, v_5v_7, v_6v_7\}$.




Obrázek 1.2 Nakreslení grafu $G$.

Aby nakreslení grafu bylo opravdu přehledné, tak je praktické požadovat

- aby každý vrchol byl zakreslen do jiného bodu,
- aby každá hrana procházela jen jejími dvěma koncovými vrcholy (a žádným jiným vrcholem),
- aby se dvě různé hrany v nakreslení protínaly nejvýše jednou,
- aby žádná hrana neprotínala sama sebe,
- aby se v nakreslení křížilo co možná nejméně hran.

Zatímco první čtyři požadavky je možné splnit vždy, nakreslit graf bez křížení hran v obecném případě nemusí být možné. Více se o kreslení grafů dozvíme v Kapitole 6.

**Příklad 1.2.** Jsou vrcholy $v_1$ a $v_5$ v grafu $G$ na Obrázku 1.2 sousední? A jsou vrcholy $v_4$ a $v_7$ sousední?

*Řešení.* Protože množina hran $E(G)$ grafu $G$ obsahuje hranu $v_1v_5$, tak vrcholy $v_1$ a $v_5$ jsou sousední (v grafu $G$). Naproti tomu hrana $v_4v_7$ do množiny $E(G)$ nepatří a proto vrcholy $v_4$ a $v_7$ nejsou sousední, jsou nezávislé v grafu $G$. ▲

Hrany v grafu můžeme chápat jako spojnice, cesty, vodiče mezi nějakými objekty, městy nebo uzly. Ale hrany mohou reprezentovat i abstraktnější vazby: hrana $xy$ může znamenat „procesy $x$ a $y$ nemohou běžet současně“, protože například využívají stejné zdroje systému.

**Příklad 1.3.** Najděte co největší množinu nezávislých vrcholů (každé dva vrcholy v množině budou nezávislé) v grafu $G$ na Obrázku 1.2.

*Řešení.* Označme hledanou množinu $N$. Všimneme si, že ze tří vrcholů $v_1$, $v_2$ a $v_3$ jsou každé dva spojené hranou, proto do hledané množiny $N$ můžeme zahrnout nejvýše jeden z nich. Podobně do $N$ lze zařadit nejvýše jeden z vrcholů $v_5$, $v_6$ a $v_7$. Při sestavování největší nezávislé množiny $N$ proto musíme vyřadit vždy alespoň čtyři vrcholy z celkového počtu osmi vrcholů v $G$.

Dále snadno ověříme, že žádné dva vrcholy z množiny $N = \{v_3, v_4, v_7, v_8\}$ nejsou spojené hranou a proto množina $N$ je největší množinou nezávislých vrcholů. (Existují i jiné čtyřprvkové množiny nezávislých vrcholů, najdete nějakou?) ▲

Výsledek Příkladu 1.3 můžeme interpretovat takto: Je-li graf modelem osmi procesů, kdy hrany spojují procesy, které nemohou běžet současně, tak jsme ukázali, že v daném systému mohou běžet současně nejvýše čtyři procesy. Navíc jsme zjistili, které procesy to mohou být.

### Různé způsoby zadání grafu

Uvědomte si, že stejný graf můžeme popsat několika způsoby. Jednak algebraicky, kdy zadáme nebo popíšeme množinu vrcholů $V$ a množinu hran $E$ nebo „obrázkem“ (nakreslením grafu). Každý z nich má své výhody. Algebraický přístup je vhodný pro implementaci algoritmů, přičemž existuje několik možností, jak strukturu grafu popsat v rámci příslušného programovacího jazyka. Pro teoretický rozbor nebo dokonce jen první seznámení s problémem je naopak vhodný obrázek, často nakreslíme i jen neúplné schéma tak, abychom získali představu o grafu, který bude naším modelem.

V praxi je obvyklé, že struktura grafu není nijak přirozeně popsána a jsme to právě my, kdo bude příslušný model (graf) sestavovat. Je pak na nás, jaký popis zvolíme.

**Příklad 1.4.** Devět kamarádů si na Vánoce dalo dárky. Každý dal dárky třem svým kamarádům. Ukažte, že není možné, aby každý dostal dárky právě od těch tří kamarádů, kterým dárky sám dal.

*Řešení.* Situaci se pokusíme znázornit grafem. Vrcholy grafu budou kamarádi, máme tedy celkem 9 vrcholů. Hrana bude mezi těmi dvěma vrcholy (kamarády), kteří si navzájem dali dárky.

Pokud by řešení existovalo, tak bychom uměli sestavit graf s devíti vrcholy, ve kterém každý vrchol je sousední se třemi jinými vrcholy. Formulace zadání však naznačuje, že taková situace nemůže nastat a tedy že takový graf není možné sestrojit.

Vskutku, takový graf neexistuje a každý pokus o jeho sestrojení skončí nezdarem. Doporučujeme, abyste si zkusili graf sestavit a uvědomit si, proč se konstrukce nedaří. Podrobné řešení si ukážeme později v této kapitole. ▲

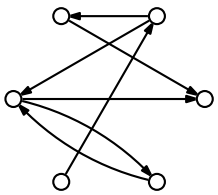
**Poznámka 1.5.** Připomeňme, že (binární) relace na množině $A$ je nějaká podmnožina kartézského součinu $A \times A$. Máme-li graf $G$ s množinou vrcholů $V$, tak hrany jsou tvořeny některými dvojicemi vrcholů z množiny $V$. Můžeme se proto na graf dívat jako na relaci $\varrho$ na množině $V$, kdy dva vrcholy $u$ a $v$ jsou v relaci, jestliže existuje hrana $uv$. Taková relace $\varrho$ je jistě symetrická (proč?) a není reflexivní. Dokonce žádný vrchol není v relaci sám se sebou a říkáme, že relace $\varrho$ je *ireflexivní*. Nakreslení grafu pak není nic jiného než nějaké znázornění relace $\varrho$.

Každé tvrzení o grafech můžeme zformulovat jako tvrzení o příslušné relaci. Upřednostňujeme však terminologii a pohled teorie grafů, neboť bývá přehlednější.

Tak, jako jsme zavedli jednoduchý (neorientovaný) graf, má smysl pracovat i s orientovanými grafy. Pokud graf reprezentuje dopravní síť, silniční síť nebo jiný produktovod, může být užitečné zavést *orientaci hrany*. Hrana $uv$ pak není totožná s hranou $vu$.

**Definice 1.6.** *Orientovaným grafem* rozumíme uspořádanou dvojici $G = (V, E)$, kde $V$ je množina *vrcholů* a množina *orientovaných hran* je $E \subseteq V \times V$.

Orientovaná hrana $uv$ pak není dvouprvková podmnožina $\{u, v\}$, ale uspořádaná dvojice $(u, v)$ dvou prvků $u, v \in V$. Vrchol $u$ je *počáteční* a vrchol $v$ *koncový* $uv$. V nakreslení znázorníme orientované hrany šipkami (Obrázek 1.3). Orientovanými grafy se budeme podrobně zabývat především v Kapitole 7.



Obrázek 1.3 Orientovaný graf.

## Pro zájemce:

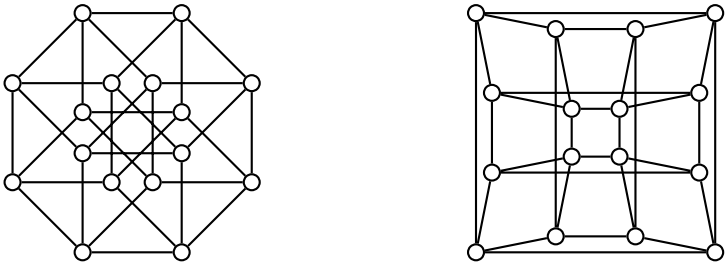
Ještě obecnější pojem než orientovaný graf, je *multigraf*. V multigrafech mohou existovat vícenásobné hrany a smyčky. V *pseudografech* mohou být současně orientované i neorientované a násobné hrany. Definici multigrafů a pseudografů najdete třeba v [5].

### Pro zájemce:

Terminologie „vrchol“ a „hrana“ grafu pochází z geometrie mnohostěnů. Například obyčejnou třírozměrnou krychli obvykle znázorňujeme jako na Obrázku 1.4. Osm vrcholů krychle přirozeně odpovídá vrcholům grafu na obrázku a stejně tak dvanáct hran krychle najdeme jako dvanáct hran v grafu na obrázku.

Obrázek 1.4 Různá nakreslení grafu krychle.

Na obrázku 1.5 pak vidíme znázornění čtyřrozměrné krychle, kterou samozřejmě není možné v třírozměrném prostoru sestrojit, ale její model můžeme celkem pěkně studovat na příslušném grafu.



Obrázek 1.5 Dvě nakreslení grafu čtyřrozměrné krychle (hyperkrychle čtvrtého řádu).

### Pojmy k zapamatování

— graf s množinou vrcholů a množinou hran
— orientovaný graf
— nakreslení grafu

## 1.2 Základní třídy grafů

V předchozí části jsme řekli, že grafy můžeme popsat algebraicky (zadat množinu vrcholů a množinu hran) nebo obrázkem. Graf však můžeme jednoznačně zadat popisem vlastností nebo jen jeho jménem. Řada struktur se vyskytuje natolik často, že se ustálilo jméno takového grafu nebo celé skupiny grafů.

Graf, který obsahuje jediný vrchol (a žádnou hranu), se nazývá *triviální* graf (Obrázek 1.6 vlevo).

### Kompletní graf

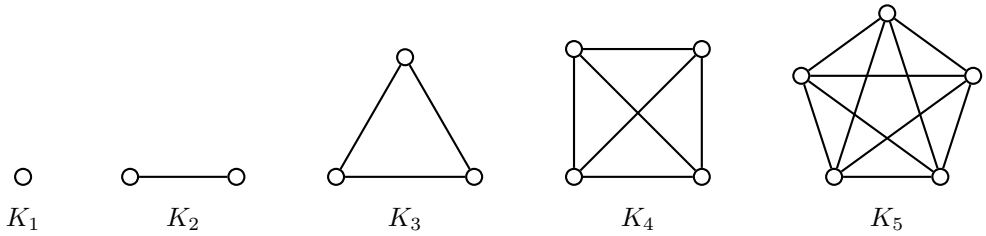
Jestliže graf na daném počtu vrcholů $n$ obsahuje všechny možné hrany, říkáme, že je *kompletní*.

**Definice 1.7.** Graf na $n$ vrcholech, kde $n \in \mathbb{N}$, který obsahuje všech $\binom{n}{2}$ hran se nazývá *úplný* nebo také *kompletní* graf a značí se $K_n$.

Algebraicky můžeme kompletní graf $K_n$ popsat následujícím způsobem.

$$K_n = (V, E): \quad V = \{1, 2, \ldots, n\}, \quad E = \{ij \colon i, j = 1, 2, \ldots, n \wedge i \neq j\}$$

Příklady nakreslení kompletních grafů pro malé hodnoty $n$ jsou na Obrázku 1.6.




Obrázek 1.6 Triviální graf a kompletní grafy.

Všimněte si, že v Definici 1.7 jsme za vrcholy grafu vzali množinu prvních $n$ přirozených čísel. Mohli jsme však stejně dobře zvolit libovolnou množinu s $n$ prvky.

### Cyklus (kružnice)

Graf, který má $n$ vrcholů spojených postupně $n$ hranami „dokola“, nazýváme *cyklus*.

**Definice 1.8.** Graf na $n$ vrcholech (kde $n \geqq 3$), které jsou spojeny po řadě $n$ hranami tak, že každý vrchol je spojen s následujícím vrcholem a poslední vrchol je navíc spojen s prvním vrcholem, se nazývá *cyklus* na $n$ vrcholech a značí se $C_n$. Číslo $n$ je *délka* cyklu $C_n$.

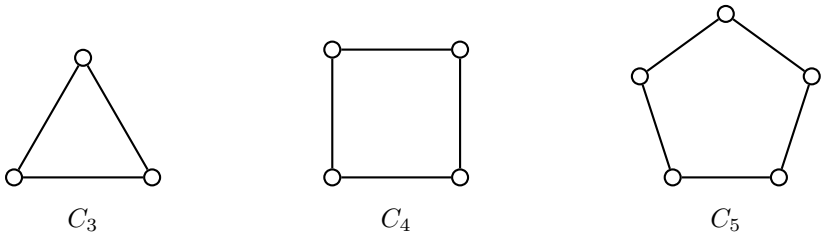
Algebraicky můžeme cyklus $C_n$ popsat takto:

$$C_n = (V, E): \quad V = \{1, 2, \ldots, n\}, \quad E = \{i(i+1) \colon i = 1, 2, \ldots n-1\} \cup \{1n\},$$

kde číslo $n$ je alespoň 3.

Připomeňme, že zápis $i(i+1)$ je zkrácený zápis množiny $\{i, i+1\}$. Některé učebnice místo „cyklus“ používají termín *kružnice*. Cyklu délky tři říkáme také *trojúhelník* a cyklu délky čtyři někdy říkáme *čtverec*. Všimněte si, že trojúhelník $C_3$ je současně kompletním grafem $K_3$.

Všimněte si, že podle definice cyklus nemůže mít méně než tři vrcholy. Pokud bychom připustili „cyklus se dvěma vrcholy“ nebo „cyklus s jediným vrcholem“, tak by taková struktura nebyla jednoduchým grafem.




Obrázek 1.7 Cykly $C_3$, $C_4$ a $C_5$.

### Cesta

Graf, jehož vrcholy je možno seřadit do řady tak, že každý vrchol (kromě prvního) je spojen s předchozím vrcholem a každý vrchol (kromě posledního) s následujícím vrcholem, nazýváme *cesta*.

**Definice 1.9.** Graf na $n$ vrcholech, které jsou spojeny po řadě $n-1$ hranami se nazývá *cesta* a značí se $P_n$.

Algebraicky můžeme cestu $P_n$ popsat

$$P_n = (V, E): \quad V = \{1, 2, \ldots, n\}, \quad E = \{i(i+1)\colon i = 1, 2, \ldots n-1\}.$$

Všimněte si, že kompletní graf $K_1$ je současně triviální graf i (triviální) cesta $P_1$ a kompletní graf $K_2$ je současně cestou $P_2$. První a poslední vrchol cesty nazýváme *koncové* vrcholy, ostatní vrcholy (pokud existují) jsou *vnitřní* vrcholy.


Obrázek 1.8 Cesty $P_3$ a $P_4$.

**Poznámka 1.10.** Pozor, v některých knihách nebo učebních textech (například [4]) se používá jiné značení, index v zápise $P_n$ odpovídá počtu hran, nikoliv počtu vrcholů.

### Kompletní bipartitní graf

Poslední třídou grafů, kterou si nyní popíšeme, jsou *kompletní bipartitní grafy*. Často se setkáme s problémem, kdy množinu vrcholů lze přirozeně rozdělit na dvě disjunktní podmnožiny, říkáme jim *partity*, přičemž je jasné že hranou jsou spojeny pouze vrcholy, které patří do různých partit. Typickým příkladem je přiřazovací úloha: máme skupinu zaměstnanců a několik pracovních úkolů. Hranou spojíme vždy nějakého pracovníka s nějakým úkolem. Hrana nikdy nespojuje dva pracovníky nebo dva úkoly. Následující definice popisuje jeden důležitý případ grafu se dvěma partitami.

**Definice 1.11.** Graf, který má vrcholovou množinu rozdělenu na dvě neprázdné disjunktní podmnožiny $M$ a $N$ ($|M| = m$, $|N| = n$) a který obsahuje všech $m \cdot n$ hran $uv$ takových, že $u \in M$ a $v \in N$, nazýváme *kompletní bipartitní graf* (nebo také *úplný bipartitní graf*) a značíme jej $K_{m,n}$.

Je dobré si uvědomit, že kompletní bipartitní graf $K_{m,n}$neobsahuje žádnou hranu $uv$, kde $u$, $v$ patří do stejné partity $M$ nebo $N$.

Algebraicky můžeme kompletní bipartitní graf $K_{m,n}$ popsat takto:

$$K_{m,n} = (M \cup N, E): \quad M = \{u_1, u_2, \ldots, u_m\}, \quad N = \{v_1, v_2, \ldots, v_n\},$$

$$M, N \neq \emptyset, \quad M \cap N = \emptyset, \quad E = \{u_i v_j : i = 1, 2, \ldots, m \wedge j = 1, 2, \ldots n\}.$$

Všimněte si, že kompletní bipartitní graf $K_{1,2}$ je současně cestou $P_3$ a kompletní bipartitní graf $K_{2,2}$ je současně cyklem $C_4$.

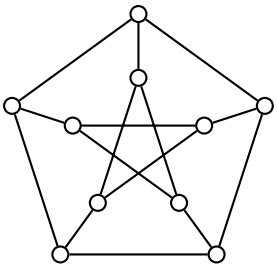
**Příklad 1.12.** Který graf má více vrcholů, $K_{10}$ nebo $K_{6,7}$? A který z nich má více hran?

*Řešení.* Podle definice má kompletní graf $K_{10}$ deset vrcholů a celkem $\binom{10}{2} = \frac{10 \cdot 9}{2} = 45$ hran. Naproti tomu kompletní bipartitní graf $K_{6,7}$ má $6 + 7 = 13$ vrcholů a celkem $6 \cdot 7 = 42$ hran.

Proto kompletní bipartitní graf $K_{6,7}$ má více vrcholů, ale kompletní graf $K_{10}$ má více hran. ▲

### Další grafy

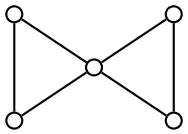
Některé další grafy se vyskytují často a mají svá jména. Mezi nejznámější patří Petersenův graf (čti „Petersenův“) na Obrázku 1.9 nebo motýlek na Obrázku 1.10.



Obrázek 1.9 Petersenův graf.

## Pojmy k zapamatování

— kompletní graf
— cyklus
— cesta
— kompletní bipartitní graf



Obrázek 1.10 Motýlek.

## 1.3 Stupeň vrcholu

### Průvodce studiem

*Nyní zavedeme „stupeň vrcholu“, což je číslo, které pro každý vrchol vyjadřuje počet sousedních vrcholů (souvisejících objektů). Zformulujeme a dokážeme tzv. „princip sudosti“, což je překvapivě jednoduché a důležité tvrzení, které ukazuje vazbu mezi stupni vrcholů a počtem hran grafu.*

*V druhé části sekce prozkoumáme stupně vrcholů v grafu detailněji. Ukážeme, které posloupnosti nezáporných celých čísel mohou tvořit posloupnost stupňů vrcholů v nějakém grafu a které ne.*

### Cíle

Po prostudování této sekce budete schopni:

- určit stupeň vrcholu v grafu,
- vysvětlit, jak spolu souvisí stupně vrcholů a počet hran v grafu,
- sestavit stupňovou posloupnost daného grafu,
- poznat, zda daná posloupnost nezáporných celých čísel je stupňovou posloupností nějakého grafu,
- sestavit graf s danou stupňovou posloupností.

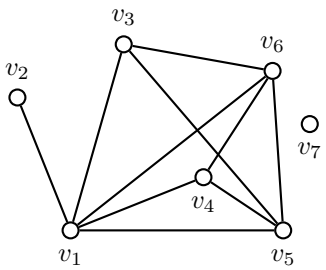
Mějme graf $G$ jako na Obrázku 1.11. Připomeňme, že je-li hrana $uv$ v grafu $G$, tak vrcholy $u$ a $v$ nazýváme koncové vrcholy této hrany a říkáme, že vrcholy $u$ a $v$ jsou s hranou $uv$ incidentní.

**Definice 1.13.** *Stupeň vrcholu* v grafu $G$ je definován jako počet hran, se kterými je vrchol incidentní.

Stupeň vrcholu $v$ v grafu $G$ značíme $\deg(v)$ nebo také $\deg_G(v)$, pokud chceme zdůraznit, v jakém grafu stupeň vrcholu $v$ zkoumáme. Často se používá symbol $\delta(G)$, který udává nejnižší stupeň vrcholu v daném grafu $G$, a symbol $\Delta(G)$, který udává nejvyšší stupeň vrcholu v grafu $G$.

Graf, ve kterém jsou všechny vrcholy stejného stupně, se nazývá *pravidelný graf*.

Určit stupně vrcholů v grafu na Obrázku 1.11 je snadné. Stačí pro každý vrchol spočítat počet incidentních hran. Stupně vrcholů však můžeme určit i v případě, kdy je graf popsán množinami vrcholů $V$ a hran $E$ nebo je-li zadán jménem.




Obrázek 1.11 Graf se stupni vrcholů 5, 1, 3, 3, 4, 4 a 0.

**Příklad 1.14.** Jaké jsou stupně vrcholů v grafu $K_n$? A jaké stupně mají vrcholy v grafu $K_{m,n}$ a $P_n$?

*Řešení.* Protože každý vrchol kompletního grafu $K_n$ je sousední se všemi ostatními vrcholy, tak stupeň každého vrcholu je $n-1$. Můžeme psát $\deg_{K_n}(v) = n-1$ pro každý vrchol $v \in V(K_n)$.

V kompletním bipartitním grafu $K_{m,n}$ je podle definice každý vrchol z jedné partity sousední pouze s vrcholy z druhé partity. Proto každý vrchol $u$ z partity s $m$ vrcholy je stupně $\deg(u) = n$ a každý vrchol $v$ z partity s $n$ vrcholy je stupně $\deg(v) = m$.

Pro cestu $P_n$ rozlišíme tři případy pečlivým rozborem Definice 1.9. Triviální cesta $P_1$ obsahuje jediný vrchol stupně 0. Cesta $P_2$ má oba vrcholy stupně 1 a konečně v cestě s více než dvěma vrcholy jsou vždy dva vrcholy stupně 1 (oba *koncové vrcholy* cesty $P_n$) a všechny zbývající vrcholy jsou stupně 2. ▲

Následující tvrzení ukazuje, že mezi stupni vrcholů a počtem hran grafu je jednoduchý vztah.

**Věta 1.15 (Princip sudosti).** *Součet stupňů všech vrcholů v grafu je vždy sudý a je roven dvojnásobku počtu hran.*

Máme-li graf $G$, tak tvrzení věty můžeme zapsat vztahem

$$\sum_{v \in V(G)} \deg_G(v) = 2|E(G)|. \tag{1.1}$$

*Důkaz.* Tvrzení ukážeme metodou dvojího počítání. Dvěma způsoby spočítáme koncové vrcholy hran. Pozor, každý vrchol započítáme tolikrát, kolikrát je koncovým vrcholem nějaké hrany – nejedná se jen o pouhý počet vrcholů.

První způsob: ve výrazu na levé straně rovnosti 1.1 stupeň každého vrcholu $v$ udává počet, kolikrát je daný vrchol $v$ koncovým vrcholem nějaké hrany. Proto součet všech stupňů vrcholů je počtem všech koncových vrcholů hran.

Druhý způsob: ve výrazu na pravé straně rovnosti 1.1 současně uvážíme, že každá hrana má dva koncové vrcholy, proto počet koncových vrcholů hran je roven současně dvojnásobku počtu hran. □

Uvědomte si, že známe-li stupně všech vrcholů v daném grafu, je počet hran grafu určen jednoznačně. Může sice existovat několik různých grafů s danými stupni, ale všechny tyto grafy mají stejný počet hran, který lze určit ze vztahu 1.1. Například, máme-li pravidelný graf $G$ na $n$ vrcholech, ve kterém jsou všechny vrcholy stupně $r$, tak graf $G$ má $nr/2$ hran.

**Příklad 1.16.** Kolik hran má graf na Obrázku 1.11?

*Řešení.* Protože stupně vrcholů v grafu na Obrázku 1.11 jsou 5, 1, 3, 3, 4, 4 a 0, tak součet stupňů jeho vrcholů je $5+1+3+3+4+4+0=20$. Podle vztahu 1.1 z Věty 1.15 má daný graf $20/2=10$ hran. ▲

Všimněte si, že součet stupňů všech vrcholů v daném grafu je podle Věty 1.15 vždy sudý. Proto se Větě 1.15 říká „princip sudosti“. Podle principu sudosti *žádný graf* nemůže mít lichý počet vrcholů lichého stupně. A proto, pokud je v grafu $G$ nějaký vrchol lichého stupně, musí v grafu $G$ takových vrcholů být sudý počet.

Nyní je čas vrátit se k Příkladu 1.4.

**Příklad 1.17 (Pokračování Příkladu 1.4).** Devět kamarádů si na Vánoce dalo dárky. Každý dal dárky třem svým kamarádům. Ukažte, že není možné, aby každý dostal dárky právě od těch tří kamarádů, kterým dárky sám dal. Návod: ukažte, že neexistuje graf, který by danou situaci modeloval.

*Řešení.* V řešení Příkladu 1.4 jsme ukázali, že pokud by existovalo řešení daného problému, tj. každý z devíti kamarádů by mohl dostat dárky od těch tří, kterým dárky dal, tak celou úlohu můžeme modelovat grafem s devíti vrcholy, přičemž každý vrchol bude stupně 3. Podle Věty 1.15 takový graf existovat nemůže, neboť by měl lichý počet (devět) vrcholů lichého stupně 3. ▲

## Pro zájemce:

V řešení Příkladu 1.17 jsme postupovali nepřímo. Uvažme implikaci $A \Rightarrow B$: „Jestliže má Příklad 1.4 řešení, pak existuje model řešení ve formě grafu.“ Vycházeli jsme z předpokladu $A$: „Existuje takové řešení úlohy, kdy si kamarádi navzájem vymění dárky“ a snažili jsme se odvodit tvrzení $B$: „sestavíme graf, který úlohu modeluje“. Místo, abychom ukázali implikaci $A \Rightarrow B$ jsme však ukázali její obměnu $\neg B \Rightarrow \neg A$ (připomínáme, že obměna implikace má vždy stejnou pravdivostní hodnotu jako implikace samotná). Protože neexistuje graf, který by danou situaci modeloval, tak Příklad 1.4 nemá řešení. To současně znamená, že neexistuje taková výměna dárků, aby každý z devíti kamarádů dostal dárky právě od těch tří kamarádů, kterým dárky sám dal.

**Příklad 1.18.** Kolik hran má graf, který má tři vrcholy stupně 5 a sedm vrcholů stupně 4?

*Řešení.* Takový graf neexistuje! Součet stupňů vrcholů takového grafu by byl $3 \cdot 5 + 7 \cdot 4 = 43$ a podle Principu sudosti (Věta 1.15) by měl 43/2 hran, což není možné, neboť počet hran nemůže být desetinné číslo.

Také si stačilo uvědomit, že v grafu by byl lichý počet vrcholů lichého stupně a proto takový graf neexistuje. ▲

**Definice 1.19.** Mějme graf $G$ s vrcholy $v_1, v_2, \ldots, v_n$. Posloupnost stupňů vrcholů $(\deg(v_1), \deg(v_2), \ldots, \deg(v_n))$ nazýváme *stupňovou posloupností* grafu $G$.

Pro přehlednost budeme pracovat se stupňovými posloupnostmi uspořádanými do nerostoucí posloupnosti, tj. $\deg(v_1) \geqq \deg(v_2) \geqq \ldots \geqq \deg(v_n)$).

Například stupňová posloupnost grafu na Obrázku 1.11 je $(5, 4, 4, 3, 3, 1, 0)$. Stupňová posloupnost kompletního grafu $K_6$ je $(5, 5, 5, 5, 5, 5)$ a stupňová posloupnost kompletního bipartitního grafu $K_{4,3}$ je $(4, 4, 4, 3, 3, 3, 3)$.

Pro každý graf umíme snadno sestavit stupňovou posloupnost a tato stupňová posloupnost je určena jednoznačně (jestliže uvažujeme neklesající posloupnost). Je přirozené se ptát, zda můžeme stupňovou posloupnost použít pro jednoznačné určení grafu. Odpověď je jednoduchá – ne. Stupňová posloupnost určuje (jednoduchý) graf jednoznačně jen ve speciálních případech. Například posloupnost $(3, 3, 3, 3)$ je stupňovou posloupností kompletního grafu $K_4$ a žádného jiného. Podobně stupňové posloupnosti $(0, 0, 0, 0, 0)$ nebo $(2, 2, 1, 1)$ určují graf jednoznačně. Naproti tomu na Obrázku 1.12 jsou dva různé grafy se stejnou stupňovou posloupností $(3, 2, 2, 1, 1, 1)$.

Obrázek 1.12 Dva grafy se stupňovou posloupností $(3, 2, 2, 1, 1, 1)$.

## Pro zájemce:

Zkuste najít nejmenší příklad (s co nejmenším počtem vrcholů a nejmenším počtem hran) dvou různých grafů se stejnou stupňovou posloupností. Napovíme, že příklad na Obrázku 1.12 není nejmenší.

Máme-li dánu posloupnost celých čísel, tato posloupnost může a nemusí být stupňovou posloupností žádného grafu. Například obsahuje-li posloupnost $P$ záporné číslo, evidentně se nejedná o stupňovou posloupnost. Podobně, jestliže posloupnost $n$ nezáporných celých čísel obsahuje číslo $n$ nebo číslo větší než $n$, tak se nejedená o stupňovou posloupnost žádného grafu, neboť žádný vrchol nemůže mít více než $n-1$ sousedních vrcholů. Podle principu sudosti ani posloupnost $(3, 3, 3, 3, 3, 3, 3, 3, 3, 3)$ není stupňovou posloupností žádného grafu (Příklad 1.4), neboť součet stupňů není sudý. Následující příklad ukazuje, že dokonce ani posloupnost $n$ nezáporných čísel,

z nichž žádné není větší než $n-1$ a součet prvků posloupnosti je sudý, nemusí být stupňovou posloupností nějakého grafu.

**Příklad 1.20.** Ukažte, že posloupnost $(3,3,3,1)$ není stupňovou posloupností žádného grafu.

*Řešení.* Nejprve si všimněte, že podle žádné z doposud zmíněných kriterií nepoznáme, že graf se stupňovou posloupností $(3,3,3,1)$ nemůže existovat: Podle principu sudosti by počet hran hledaného grafu byl $(3+3+3+1)/2 = 5$. Hledaný graf by musel mít čtyři vrcholy. Nejvyšší číslo 3 nepřesahuje nejvyšší možný stupeň grafu na čtyřech vrcholech.

Všimneme si, že graf na čtyřech vrcholech může obsahovat nejvýše $\binom{4}{2} = 6$ hran, s každým vrcholem budou incidentní vždy tři z nich. Víme, že hledaný graf by měl obsahovat 5 hran, avšak současně musí obsahovat vrchol stupně 1, a tedy z celkového počtu 6 hran musí alespoň dvě hrany chybět, což není možné. Proto graf se stupňovou posloupností $(3,3,3,1)$ neexistuje. ▲

V řešení Příkladu 1.20 se nám podařilo zdůvodnit, že uvedená posloupnost není stupňovou posloupností žádného grafu. Existuje nějaký jednoduchý a univerzální postup, jak pro každou posloupnost $P$ rozhodneme, zda existuje graf s takovou stupňovou posloupností $P$? Odpověď dává následující věta. Takový univerzální a poměrně jednoduchý postup existuje, jedná se však o rekurzivní postup, kdy odpověď na otázku, zda daná posloupnost je stupňovou posloupností nějakého grafu, převedeme na stejnou otázku avšak pro kratší posloupnost s menšími čísly.

**Věta 1.21 (Věta Havlova-Hakimiho).** *Nechť $(d_1 \geqq d_2 \geqq \ldots \geqq d_n)$ je posloupnost nezáporných celých čísel. Pak netriviální graf s $n$ vrcholy se stupňovou posloupností*

$$D = (d_1, d_2, \ldots, d_n)$$

*existuje právě tehdy, když existuje graf s $n-1$ vrcholy se stupňovou posloupností*

$$D' = (d_2 - 1, d_3 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2}, d_{d_1+3}, \ldots, d_n).$$

Větu 1.21 dokázali nezávisle na sobě český matematik V. J. Havel a americký matematik S. Louis Hakimi. Důkaz je poměrně technický a proto jej vynecháme. Na druhou stranu v dalším textu využijeme myšlenky důkazu, který je konstruktivní a umožní nám sestrojit alespoň jeden graf s danou stupňovou posloupností.

Posloupnost $D'$ vznikne z nerostoucí posloupnosti $D$ tak, že vynecháme první člen $d_1$ a právě $d_1$ následujících prvků (pokud existují) zmenšíme o jedničku. Dostaneme posloupnost $D'$. Nakonec její prvky přeuspořádáme tak, abychom dostali nerostoucí posloupnost. Posloupnost $D'$ je kratší a navíc téměř vždy obsahuje menší čísla než posloupnost $D$, a proto po nejvýše $n-1$ krocích dostaneme posloupnost, o které budeme umět ihned rozhodnout, zda je grafová. Stejná odpověď pak podle Věty 1.21 platí i pro posloupnost $D$.

Vrátíme se k Příkladu 1.20.

**Příklad 1.22.** Ukažte, že posloupnost $(3, 3, 3, 1)$ není stupňovou posloupností žádného grafu užitím Věty 1.21.

*Řešení.* Opakovaným užitím Věty 1.21 dostaneme následující posloupnosti

$$(3, 3, 3, 1) \overset{H-H}{\sim} (2, 2, 0) \overset{H-H}{\sim} (1, -1).$$

Poslední z nich jistě není stupňovou posloupností žádného grafu, proto ani $(3, 3, 3, 1)$ není stupňovou posloupností žádného grafu. ▲

Všimněte si, že mezi jednotlivými posloupnostmi ve výpočtu nemůžeme použít symbol „=“, neboť se jedná o různé posloupnosti. Místo toho používáme symbol $\sim$, kterým vyjadřujeme ekvivalenci jednotlivých posloupností, přesněji o ekvivalenci vzhledem k otázce, zda se jedná nebo nejedná o stupňovou posloupnost nějakého grafu.

**Poznámka 1.23.** K určení, zda daná posloupnost celých čísel je stupňovou posloupností nějakého grafu, můžeme použít následující postup. Jednotlivé kroky jsou seřazeny dle obtížnosti, začínáme nejsnadněji ověřitelnými možnostmi:

1) ověříme, zda posloupnost neobsahuje záporná čísla,
2) ověříme, zda posloupnost $n$ čísel neobsahuje číslo $n$ a větší,
3) ověříme, zda součet prvků posloupnosti je sudý (Věta 1.15),
4) opakovaným použitím Věty 1.21 ověříme, zda daná posloupnost je stupňovou posloupností nějakého grafu.

Samozřejmě můžeme ihned ověřovat podmínku Věty 1.21. Pokud by však odpověď na některou z předchozích otázek byla záporná, tak se můžeme vyhnout často zdlouhavému postupu.

**Příklad 1.24.** Užitím Věty 1.21 rozhodněte, zda posloupnost $(6, 1, 3, 2, 3, 2, 4, 1)$ je stupňovou posloupností nějakého grafu.

*Řešení.* Na první pohled je zřejmé, že posloupnost osmi čísel neobsahuje záporná čísla ani číslo větší než 7. Navíc posloupnost splňuje i princip sudosti (Věta 1.15), neboť počet lichých čísel v posloupnosti je sudý. Nemůžeme proto snadno vyloučit, že se jedná o stupňovou posloupnost grafu. Dále posloupnost seřadíme do nerostoucí posloupnosti $(6, 4, 3, 3, 2, 2, 1, 1)$ a užitím Věty Havla-Hakimiho (Věty 1.21) dostaneme

$$(6, 4, 3, 3, 2, 2, 1, 1) \overset{H-H}{\sim} (3, 2, 2, 1, 1, 0, 1) \sim (3, 2, 2, 1, 1, 1, 0).$$

To znamená, že posloupnost $(6, 4, 3, 3, 2, 2, 1, 1)$ je grafová právě tehdy, když je grafová posloupnost $(3, 2, 2, 1, 1, 1, 0)$, kterou jsme také seřadili do neklesající posloupnosti. Opakovaným použitím věty Havla-Hakimiho dostaneme

$$(3, 2, 2, 1, 1, 1, 0) \overset{H-H}{\sim} (1, 1, 0, 1, 1, 0) \sim (1, 1, 1, 1, 0, 0) \overset{H-H}{\sim} (0, 1, 1, 0, 0) \sim$$

$$\sim (1,1,0,0,0) \overset{H-H}{\sim} (0,0,0,0).$$

Protože posloupnost $(0,0,0,0)$ je bezpochyby stupňová posloupnost grafu sestávajícího ze čtyř izolovaných vrcholů (čtyř kopií grafu $K_1$), je také posloupnost $(6,4,3,3,2,2,1,1)$ stupňovou posloupností nějakého grafu.

Zkušený počtář mohl už v průběhu výpočtu snadno poznat, že již posloupnost $(1,1,1,1,0,0)$ je jistě stupňovou posloupností grafu, který obsahuje dvě kopie grafu $K_2$ a dvě kopie grafu $K_1$. ▲

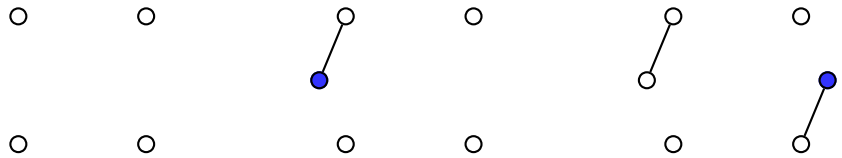
Rekurzivní výpočet, který slouží k ověření, zda daná stupňová posloupnost $D$ je nebo není stupňovou posloupností nějakého grafu, můžeme použít pro nalezení grafu s danou stupňovou posloupností. Při popisu použijeme označení z Věty 1.21. Nejprve sestavíme takový graf, jehož stupňová posloupnost $D'$ je ve výpočtu poslední, neboť víme, že takový graf existuje. V dalším kroku přidáme jeden vrchol stupně $d_1$ a spojíme jej hranami s vrcholy stupňů $d_2 - 1, d_3 - 1, \ldots, d_{d_1+1} - 1$ tak, aby vznikl graf se stupňovou posloupností $D$. Podobně přidáváme další vrcholy, až sestavíme graf se zadanou stupňovou posloupností.

**Příklad 1.25 (pokračování Příkladu 1.24).** Najděte graf se stupňovou posloupností $(6,1,3,2,3,2,4,1)$.

*Řešení.* Na straně 15 jsme ukázali, že posloupnost $(6,1,3,2,3,2,4,1)$ je stupňovou posloupností nějakého grafu. Nyní výpočet použijeme pro nalezení takového grafu. Nejprve nakreslíme graf se stupňovou posloupností $(0,0,0,0)$ (Obrázek 1.13 vlevo).

Potom přidáme vrchol stupně 1 (přidaný vrchol je vždy označen modře) a spojíme jej s některým z nakreslených vrcholů. Dostaneme graf se stupňovou posloupností $(1,1,0,0,0)$ (Obrázek 1.13 uprostřed).

Přidáme další vrchol stupně 1 a spojíme jej s některým z nakreslených vrcholů stupně 0. Dostaneme graf se stupňovou posloupností $(1,1,1,1,0,0)$ (Obrázek 1.13 vpravo).



Obrázek 1.13 Rekonstrukce grafu, první tři kroky.

V dalším kroku přidáme vrchol stupně 3 a spojíme jej se dvěma vrcholy stupně 1 a jedním vrcholem stupně 0. Všimněte si, že vrcholy stupně 1 můžeme vybrat několika způsoby, zvolili jsme náhodně dva z nich. Dostaneme graf se stupňovou posloupností $(3,2,2,1,1,1,0)$ (Obrázek 1.14 vlevo).

Nakonec přidáme vrchol stupně 6 a spojíme jej s jedním vrcholem stupně 3, se dvěma vrcholy stupně 2, se dvěma vrcholy stupně 1 a jedním vrcholem stupně 0. Znovu připomínáme, že hrany nepřidáváme úplně náhodně. Hranou spojíme vždy

přidaný vrchol a druhý koncový vrchol má předepsaný stupeň. Pokud máme více vrcholů stejného stupně, můžeme zvolit libovolný z nich. Dostaneme například graf se stupňovou posloupností $(6, 4, 3, 3, 2, 2, 1, 1)$ (Obrázek 1.14 vpravo).

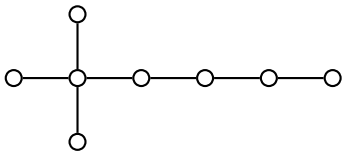
Obrázek 1.14 Rekonstrukce grafu, další dva kroky.

▲

## Pro zájemce:

Věta 1.21 spolu s postupem podrobně rozebraným v Příkladu 1.25 nám umožní pro každou posloupnost nezáporných čísel nejen rozhodnout, zda se jedná o stupňovou posloupnost nějakého grafu, ale také takový graf zkonstruovat. Jak jsme uvedli dříve, tak graf není svou stupňovou posloupností určen jednoznačně. Obvykle máme při rekonstrukci grafu možnost volby a mohli bychom zkonstruovat více různých grafů se stejnou stupňovou posloupností.

Není však těžké si rozmyslet, že existují grafy, které postupem uvedeným v Příkladu 1.25 nepůjde zkonstruovat. Pozor, neříkáme, že takový graf neexistuje, ale že konstrukce najde vždy *jiný* graf se stejnou stupňovou posloupností. Příklad takového grafu je na Obrázku 1.15. Rozmyslete si, proč pomocí výše popsaného algoritmu sestavíme jiný graf se stejnou stupňovou posloupností $(4, 2, 2, 2, 1, 1, 1, 1)$.



Obrázek 1.15 Graf, který nelze zkonstruovat postupem popsaným v Příkladu 1.25.

## Pojmy k zapamatování

— stupeň vrcholu
— princip sudosti
— stupňová posloupnost grafu
— Věta Havlova-Hakimiho

## 1.4 Podgrafy

Při řešení úloh často uvažujeme případy, kdy z daného grafu vynecháme některé vrcholy (a hrany s nimi incidentní), nebo vynecháme nějaké hrany, případně obojí. Přirozeně se tak dostáváme k pojmu „podgraf“.

**Definice 1.26.** Graf $H$ nazveme *podgrafem* grafu $G$, jestliže $V(H) \subseteq V(G)$ a $E(H) \subseteq E(G)$. Píšeme $H \subseteq G$.

Všimněte si, že vynecháme-li z $G$ některý vrchol, musíme současně vynechat i všechny hrany s ním incidentní. Definice je v tomto smyslu korektní, neboť pokud bychom vynechali jen vrcholy a nikoli hrany, tak $H$ by nemusel být grafem, ale jen nějakou dvojicí množin $V(H)$ a $E(H)$. Abychom $H = (V(H), E(H))$ nazvali grafem, tak $E(H)$ nemůže obsahovat jiné dvouprvkové množiny, než (některé) dvouprvkové podmnožiny $V(H)$.

Na Obrázku 1.16 máme graf $G$ a jeho podgraf $H$, který vznikl odebráním jednoho vrcholu (a hrany s ním incidentní) a dalších čtyř hran. Naopak, pokud bychom v grafu $G$ na Obrázku 1.2 odebrali vrchol $v_7$ a ponechali vrcholy $W = \{v_1, v_2, \ldots, v_6\}$ a ponechali hrany $F = \{v_1v_2, v_1v_3, v_1v_4, v_1v_5, v_2v_3, v_2v_4, v_2v_6, v_3v_5, v_3v_7, v_4v_5, v_5v_6\}$, tak sice $W \subseteq V$ a $F \subseteq F$ ale dvojice $(W, F)$ netvoří graf, protože hrana $v_3v_7$ nemá oba koncové vrcholy v množině $W$ (neplatí $\{v_3, v_7\} \subseteq W$).

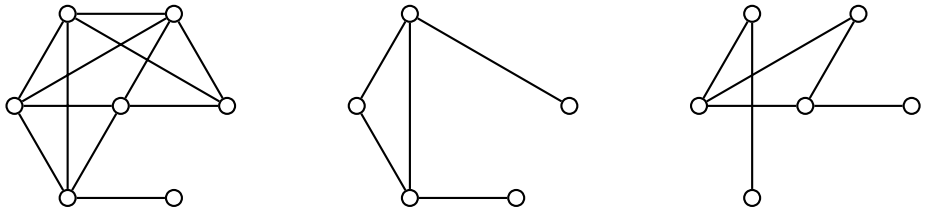
Obrázek 1.16 Graf $G$ a jeho podgraf $H$.

Nyní popíšeme dva důležité speciální případy podgrafů.

**Definice 1.27.** Podgraf $I$ grafu $G$ nazveme *indukovaným* podgrafem grafu $G$, jestliže $E(I)$ obsahuje všechny hrany grafu $G$, které jsou incidentní s vrcholy z $V(I)$. (Vynecháme pouze hrany, které byli incidentní s vynechanými vrcholy.)

Ekvivalentně můžeme říci, že indukovaný podgraf $I$ vznikne z grafu $G$ (případným) vynecháním některých vrcholů a vynecháním pouze těch hran, které jsou incidentní s některým vynechaným vrcholem. Zmíníme ještě třetí způsob, jak zavést pojem indukovaného podgrafu. Můžeme říci, že indukovaný podgraf $I$ grafu $G$, je takový podgraf grafu $G$, který má ze všech podgrafů $G$ s vrcholovou množinou $V(I)$ největší počet hran.

Graf $I$ na Obrázku 1.17 uprostřed je indukovaným podgrafem grafu $G$, avšak podgraf $H$ (vpravo) grafu $G$ není indukovaný, neboť v něm chybí i další hrany grafu $G$.

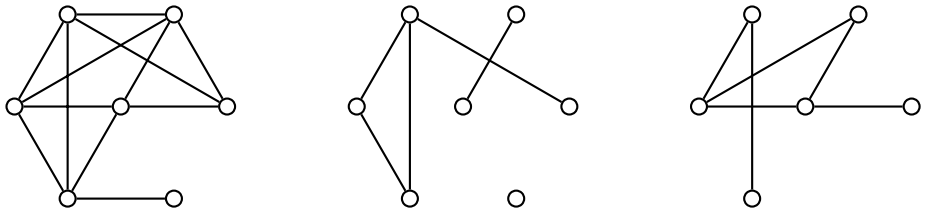


Obrázek 1.17 Graf $G$, jeho indukovaný podgraf $I$ a podgraf $H$, který není indukovaný.

Uvědomte si, že indukovaný podgraf obsahuje co možná největší počet hran původního grafu. V jistém smyslu analogicky můžeme požadovat, aby podgraf měl co nejvíce vrcholů původního grafu. Dostáváme následující definici.

**Definice 1.28.** Podgraf $F$ grafu $G$ nazveme *faktorem* grafu $G$, jestliže $V(F) = V(G)$.

Graf $F$ na Obrázku 1.18 uprostřed je faktorem grafu $G$, avšak podgraf $H$ (vpravo) není faktorem, neboť v něm chybí nějaké vrcholy grafu $G$ (stačí, aby chyběl jediný vrchol).



Obrázek 1.18 Graf $G$, jeho faktor $F$ a podgraf $H$, který není faktorem.

Jestliže o grafu $H$ říkáme, že je podgrafem grafu $G$, tak naopak graf $G$ se nazývá *nadgraf* grafu $H$. Pojmy podgrafu, indukovaného podgrafu, faktoru využijeme v argumentacích a při popisu algoritmů v dalším textu.

### Bipartitní grafy

Na straně 9 jsme nadefinovali kompletní bipartitní graf. Každému podgrafu kompletního bipartitního grafu říkáme *bipartitní graf*. O bipartitních grafech se dozvíme více v Sekcích 6.1 a 7.4.

## Pojmy k zapamatování

— podgraf
— indukovaný podgraf
— faktor

## 1.5 Isomorfismus grafů

Modelovat reálnou situaci grafem je praktické zejména proto, že při popisu abstrahujeme od konkrétní situace a zaměříme se na podstatné vlastnosti struktury: který vrchol je sousední s kterým vrcholem. Stejnou situaci však lze popsat na první pohled různými grafy (různé označení vrcholů, zcela odlišné nakreslení grafu, ...) Přitom se jedná o *stejnou strukturu*.

Je přirozené se ptát, jak můžeme poznat, že dva grafy mají stejnou strukturu a že se liší jen jejich označení nebo nakreslení. Právě proto se zavádí pojem isomorfismu.

**Definice 1.29.** *Isomorfismus grafů* $G$ *a* $H$ je bijektivní zobrazení $f : V(G) \to V(H)$, pro které platí, že každé dva vrcholy $u, v$ v grafu $G$ jsou sousední právě tehdy, když jsou sousední jejich obrazy $f(u), f(v)$ v grafu $H$.

Stručně můžeme zapsat

$$\forall u, v \in V(G) : uv \in E(G) \Leftrightarrow f(u)f(v) \in E(H). \tag{1.2}$$

Říkáme, že dva grafy jsou isomorfní, jestliže mají stejnou strukturu. Hlavním důvodem, proč se zavádí pojem isomorfismu grafů je, abychom měli nástroj, pomocí kterého se budeme snažit rozhodnout, zda nějaké dva grafy mají nebo nemají stejnou strukturu.

**Poznámka 1.30 (Isomorfní grafy si nejsou rovny).** Uvědomte si, že pokud dva grafy $G$ a $H$ jsou isomorfní, tak si *nejsou* rovny, protože každý má třeba úplně jinou množinu vrcholů. Samozřejmě $|V(G)| = |V(H)|$, ale obecně nemusí platit $V(G) = V(H)$. Důležité je, že struktura obou grafů je stejná. Bylo by formální chybou napsat $G = H$. Pro vyjádření informace, že grafy $G$ a $H$ jsou isomorfní používáme zápis $G \simeq H$.

**Příklad 1.31.** Jsou grafy $G$ a $H$ na Obrázku 1.19 isomorfní?


Obrázek 1.19 Grafy $G$ a $H$.

*Řešení.* Ukážeme, že grafy $G$ a $H$ jsou isomorfní, neboť zobrazení $f : V(G) \to V(H)$

dané seznamem vrcholů a jejich obrazů zachovává sousednost.

$$\begin{aligned} f(v_1) &= x_1 \\ f(v_2) &= x_4 \\ f(v_3) &= x_5 \\ f(v_4) &= x_3 \\ f(v_5) &= x_6 \\ f(v_6) &= x_2 \end{aligned}$$

Všimněte si, že vrchol $v_4$ stupně 4 se zobrazí na vrchol $x_3$, který je stupně 4. Oba vrcholy $v_1$, $v_2$ stupně 3 se zobrazí opět na vrcholy $x_1$, $x_4$ stupně 3. A konečně zbývající vrcholy $v_3$, $v_5$, $v_6$ stupně 2 se zobrazí na vrcholy $x_2$, $x_5$, $x_6$ stupně 2. Ale pozor! Samotná rovnost stupňů jako důkaz isomorfismu nestačí! Ověříme zachování sousednosti pro všech osm hran grafu $G$: hrana $v_1v_2$ grafu $G$ se zobrazí na hranu $f(v_1)f(v_2) = x_1x_4$ grafu $H$, hrana $v_1v_4$ grafu $G$ se zobrazí na hranu $f(v_1)f(v_4) = = x_1x_3$ grafu $H$ a konečně hrana $v_1v_6$ grafu $G$ se zobrazí na hranu $f(v_1)f(v_6) = x_1x_2$ grafu $H$. Zbývajících pět hran se ověří podobně. Navíc třeba vrcholy $v_1$, $v_3$ nejsou sousední a proto ani jejich obrazy $f(v_1) = x_1$, $f(v_3) = x_5$ nemohou být sousední.

Všimněte si, že pokud mají oba grafy $G$ a $H$ stejný počet hran, stačí ověřit zachovaní existujících hran. Chybějící hrany grafu $G$ pak musí chybět i v grafu $H$. ▲

Samotná rovnost stupňů vrcholů a stupňů jejich obrazů k zajištění isomorfismu nestačí, jak ukazuje následující příklad.

**Příklad 1.32.** Uvažujme opět grafy $G$ a $H$ na Obrázku 1.19. Ukažte, že zobrazení $h : V(G) \to V(H)$ dané seznamem vrcholů a jejich obrazů

$$\begin{aligned} h(v_1) &= x_1 \\ h(v_2) &= x_4 \\ h(v_3) &= x_5 \\ h(v_4) &= x_3 \\ h(v_5) &= x_2 \\ h(v_6) &= x_6 \end{aligned}$$

není isomorfismus grafů $G$ a $H$ i když stupeň obrazu je vždy roven stupni vzorového vrcholu (ověřte si to).

*Řešení.* Stačí si všimnou, že zatímco hrana $v_1v_6$ patří do grafu $G$ tak její obraz $h(v_1)h(v_6) = x_1x_6$ do grafu $H$ nepatří. Je porušena podmínka 1.2. To ale neznamená, že grafy nejsou isomorfní! To znamená, že zobrazení $h$ není isomorfismem grafů $G$ a $H$. ▲

Jak však poznáme, že dva grafy jsou isomorfní? Stručně řečeno „těžko“. Pokud jsou dva grafy isomorfní, musíme najít isomorfismus (bijektivní zobrazení mezi množinami vrcholů) a ověřit, že zachovává sousednost vrcholů (vztah 1.2). Hledat zobrazení „zkusmo“ není dobrý nápad, neboť různých bijektivních zobrazení mezi vrcholovými množinami dvou grafů na $n$ vrcholech existuje $n!$.

## Pro zájemce:

Mějme nějakou kladnou funkci $g(n) : \mathbb{N} \to \mathbb{N}$. Předpokládejme, že hodnota funkce $g(n)$ vyjadřuje odhad počtu kroků nějakého algoritmu v závislosti na velikosti vstupu, který je dán proměnnou $n$. Připomeňme, že symbolem $O(g(n))$ rozumíme takovou množinu nezáporných funkcí $f(n)$, pro které existují kladné konstanty $c$ a $n_0$ tak, že pro všechna přirozená čísla $n > n_0$ platí $0 \leqq f(n) \leqq c \cdot g(n)$. Jestliže řekneme, že algoritmus má složitost $O(g(n))$, tak tím chceme vyjádřit, že počet kroků (základních operací) algoritmu, a tedy i doba běhu algoritmu, poroste stejně jako roste funkce $g(n)$ v závislosti na velikosti vstupu $n$. Přesný počet kroků nebo doba běhu pak závisí (až na nějaký konstantní násobek) na konkrétní implementaci nebo architektuře, případně na taktu použitého procesoru.

### O složitosti ověření isomorfismu grafů

Při hledání isomorfismů nemusí už pro poměrně malé grafy stačit ani použití hrubé síly počítače, neboť počet prověřovaných možností roste v závislosti na počtu vrcholů velmi rychle. Složitost algoritmu, který prověřuje všechna možná bijektivní zobrazení množin vrcholů, není polynomiální, neboť počet kroků je úměrný $O(n!)$. Navíc pro každou možnost bychom měli ověřit zachování sousednosti pro *všechny* hrany.

Jestliže naopak dva grafy isomorfní nejsou, nečekejme, že se nám podaří vyloučit každý z $n!$ možných isomorfismů. Již pro malé grafy je takových bijekcí příliš mnoho. Pro vyloučení isomorfismu dvou grafů doporučujeme najít nějakou vlastnost, který jeden graf má a druhý nemá. Jakou vlastnost? Zde se uplatní důvtip či zkušenost řešitele.

**Příklad 1.33.** Jsou grafy $G$ a $H$ na Obrázku 1.20 isomorfní?

Obrázek 1.20 Grafy $G$ a $H$.

*Řešení.* Ukážeme, že grafy $G$ a $H$ nejsou isomorfní. Nebudeme však postupně zkoumat všechny možné bijekce $V(G) \to V(H)$, kterých existuje $6! = 720$. Dokonce nebudeme postupně procházet všech $4!$ bijekcí $V(G) \to V(H)$, které zobrazují na sebe

vrcholy odpovídajících stupňů. Všimneme si, že v grafu $G$ jsou dva vrcholy stupně 2 sousední (na Obrázku 1.21 jsou vyznačeny červeně), avšak v grafu $H$ žádné dva vrcholy stupně 2 nejsou sousední (na Obrázku 1.21 jsou všechny vrcholy stupně 2 vyznačeny modře), proto nemohou být grafy $G$ a $H$ zakreslením stejné struktury a nejsou isomorfní.

Obrázek 1.21 Grafy $G$ a $H$ s vyznačenými vrcholy.

▲

Další animovaný příklad najdete na adrese http://mi21.vsb.cz/sites/mi21.vsb.cz/files/unit/izomorfismus.pdf.

Při vyšetřování isomorfismu doporučujeme prozkoumat vlastnosti grafů, které shrnuje následující věta. Šikovný student ji jistě zvládne dokázat.

**Věta 1.34.** *Isomorfní grafy $G$ a $H$ mají*

- *stejný počet vrcholů,*
- *stejný počet hran,*
- *stejný nejvyšší stupeň $\Delta(G) = \Delta(H)$,*
- *stejný nejnižší stupeň $\delta(G) = \delta(H)$,*
- *stejnou stupňovou posloupnost,*
- *každý podgraf grafu $G$ musí být isomorfní s podgrafem grafu $H$ a naopak.*

Jestliže se pro dané dva grafy bude některá z uvedených vlastností lišit, pak jistě tyto dva grafy nejsou isomorfní.

Naopak, při konstrukci isomorfismu (bijekce $f : V(G) \to V(H)$) využijeme faktu, že vzor $x$ i obraz $f(x)$ musí být vrcholy stejných stupňů, tj. $\deg_G(v) = \deg_H(f(v))$. Opačná implikace však neplatí, jak jsme ukázali v Příkladu 1.20. Nestačí porovnat stupňové posloupnosti!

Na závěr upozorníme na jedno pozorování o isomorfních grafech.

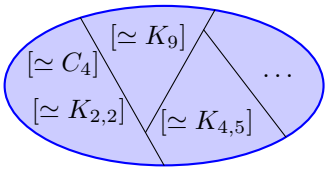
**Věta 1.35.** *Relace „být isomorfní“ $\simeq$ je relací ekvivalence na třídě všech grafů.*

*Důkaz.* Relace $\simeq$ je

- reflexivní, protože každý graf je isomorfní sám sobě (za isomorfismus zvolíme identické zobrazení),
- symetrická, neboť k isomorfismu (bijekci) $f : V(G) \to V(H)$ existuje (jednoznačně) isomorfismus $f^{-1} : V(H) \to V(G)$,
- tranzitivní, neboť skládáním isomorfismů $f : V(G) \to V(H)$ a $g : V(H) \to V(F)$ dostaneme isomorfismus (složené zobrazení) $g \circ f : V(G) \to V(F)$.

Ověřením reflexivity, symetrie a tranzitivity jsme ukázali, že $\simeq$ je relací ekvivalence na třídě všech grafů. □

Protože relace $\simeq$ z Věty 1.35 je relací ekvivalence, má smysl sestavit rozklad třídy (množiny) všech grafů podle této ekvivalence. Podrobnosti najdete v [6]. Pokud mluvíme o nějakém *grafu*, myslíme tím obvykle kterýkoliv prvek celé třídu takového rozkladu (Obrázek 1.22). Nezáleží na konkrétní reprezentaci, nakreslení či pojmenování grafu, všechny isomorfní grafy mají stejnou strukturu. Když řekneme například „v cyklu $C_5$“ najdeme dva nezávislé vrcholy, myslíme tím každý graf $C_5$, nikoliv nějaké jeho jediné nakreslení v sešitě.




Obrázek 1.22 Třídy grafů.

### Pojmy k zapamatování

— isomorfismus grafů

## 1.6 O implementaci grafů


### Průvodce studiem

*V závěru kapitoly zmíníme několik jednoduchých způsobů implementace grafů v počítači. Upozorníme na praktická omezení každé uvedené implementace.*

### Cíle

Po prostudování této sekce budete schopni:

- třemi způsoby implementovat strukturu grafu do programovacího jazyka,

- zvolit nejvhodnější způsob implementace v závislosti na řešeném problému,
- sestavit stupňovou posloupnost daného grafu,
- poznat, zda daná posloupnost nezáporných celých čísel je stupňovou posloupností nějakého grafu,
- sestavit graf s danou stupňovou posloupností.

Mějme nějaký graf $G$. Vrcholy grafu $G$ označíme čísly $0, 1, \ldots, n-1$, kde $n$ udává počet vrcholů grafu $G$. V algoritmech budeme obvykle pro uložení počtu vrcholů používat proměnnou `N`.

Většina algoritmů uvedených v tomto textu není psána pro nějaký konkrétní programovací jazyk, ale v jakémsi pseudokódu, který spíš než o formální přesnost se snaží o názornost a dobré porozumění. Při implementaci uvedených algoritmů do konkrétního programovacího jazyka by měly stačit základní znalosti a klasické programovací struktury jako jsou podmínky `if/else`, cykly `for` řízené proměnnou nebo cykly `while` řízené podmínkou. Dále předpokládáme, že můžeme pracovat s jedno- nebo dvourozměrnými poli, jejichž indexy budou začínat od 0. Také budeme používat funkce, zásobník a frontu.

Zmíníme tři nejběžnější implementace grafů v počítači.

### Matice sousednosti

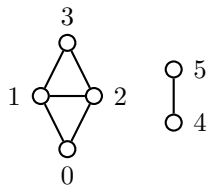
Pro grafy s velkým počtem hran a pro grafy, které se v průběhu algoritmu často mění je vhodné uložení pomocí matice sousednosti.

*Matice sousednosti* grafu $G$ je čtvercová matice $A = (a_{ij})$ řádu $n$, ve které je prvek $a_{ij} = 1$ právě tehdy, když jsou vrcholy $i$ a $j$ sousední. V opačném případě je $a_{ij} = 0$, přičemž $i, j \in \{1, 2, \ldots, n\}$. Můžeme tedy psát:

$$a_{ij} = \begin{cases} 1 & \text{je-li } v_i v_j \in E(G) \\ 0 & \text{jinak.} \end{cases}$$

Matici sousednosti můžeme uložit do dvourozměrného pole `g[][]`. Pozor: v matici $A$ stejně jako v poli `g[][]` indexujeme od 0.

Je zřejmé, že matice $A$ je pro jednoduché grafy symetrická a že součet čísel v $i$-tém řádku (v $i$-tém sloupci) matice $A$ je roven stupni vrcholu $i$. Snadno ověříme, zda se hrana $ij$ v grafu nachází nebo ne. Snadno také nějakou hranu $ij$ do grafu přidáme nebo ji z grafu odebereme. Obecně však je matice sousednosti rozsáhlá a zabírá mnoho paměti i pro řídké grafy.




Obrázek 1.23 Graf $G$ se šesti vrcholy $0, 1, \ldots, 5$.

Matice sousednosti grafu $G$ z Obrázku 1.23 je

$$A = \begin{array}{c} v\backslash v \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

### Matice incidence

Pro grafy s menším počtem hran a pro grafy, ve kterých se v průběhu algoritmu nemění počet hran, je vhodné uložení pomocí incidenční matice.

*Incidenční matice* $B$ grafu $G$ je obdélníková matice s $n$ řádky a $m = |E(G)|$ sloupci. Každému vrcholu grafu $G$ odpovídá jeden řádek matice $B$ a každé hraně grafu $G$ jeden sloupec matice $B$. Hrany označíme $e_0, e_1, \ldots, e_{m-1}$. Prvek $b_{ij}$ matice $B$ nabývá hodnoty 1 právě tehdy, když vrchol $i$ je incidentní s hranou $e_j$, v opačném případě je $b_{ij} = 0$. Je snadné si uvědomit, že součet čísel v každém sloupci incidenční matice je vždy roven 2 a součet čísel v $i$-tém řádku je roven stupni vrcholu $i$. Pro velké grafy by incidenční matice byla velmi rozsáhlá a poměrně řídká. Obsahuje jen $2m$ jedniček z celkového počtu $mn$ prvků.

Incidenční matice grafu $G$ je na Obrázku 1.23.

$$B = \begin{array}{c} v\backslash e \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 01 & 02 & 12 & 13 & 23 & 45 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

### Seznam sousedů

Graf můžeme reprezentovat i pomocí seznamů sousedních vrcholů. Pro každý vrchol $i = 0, 1, \ldots, n-1$ grafu $G$ vytvoříme seznam vrcholů, například užitím pole

`sous[i][]`, které jsou s vrcholem $i$ sousední. Každé pole bude mít $\deg(i)$ položek, kde $\deg(i)$ je stupeň vrcholu $i$, který je uložen v poli `deg[]`. Prvky `sous[i][0]`, `sous[i][1]`, ..., `sous[i][deg[i]-1]` obsahují vrcholy (nebo jejich indexy) sousední s vrcholem $i$. Seznam stupňů vrcholů a seznamy sousedů vrcholů grafu $G$ z Obrázku 1.23 jsou

```
deg[] = [2, 3, 3, 2, 1, 1],
```

```
sous[0][] = [1, 2],
sous[1][] = [0, 2, 3],
sous[2][] = [0, 1, 3],
sous[3][] = [1, 2],
sous[4][] = [5],
sous[5][] = [4].
```

Všimněte si, že má-li graf $m$ hran, budou seznamy (pole) obsahovat celkem $2m$ položek, neboť každá hrana $ij$ je v seznamech uložena dvakrát. Jednou jako vrchol $j$ v poli `a[i][]` a podruhé jako vrchol $i$ v poli `a[j][]`.

Nevýhodou polí je náročná úprava takové struktury v průběhu algoritmu. Pokud implementujeme algoritmus, který bude zpracovávat obecný řídký graf, tak pro seznamy sousedů může být vhodnější použít místo polí dynamické linkované seznamy. Výhodou je, že strukturu grafu můžeme v průběhu algoritmu snadno modifikovat. Položky seznamu můžeme vyřadit nebo do seznamu zařadit nové. Nevýhodou je časová náročnost vyhledání konkrétní hrany. Naproti tomu použijeme-li pole, můžeme (už při vytvoření) seřadit seznamy sousedů podle pevně zvoleného klíče a pro vyhledání použít binární dělení.

V Kapitole 4 se budeme věnovat ohodnoceným grafům. Pro uložení ohodnocených grafů lze použít podobná schémata a čísla přiřazená hranám nebo vrcholům grafu budou obvykle uložena do dalších polí nebo struktur. Při implementaci složitějších algoritmů můžeme použít i strukturované datové typy a pod.

Animovaný příklad, který ukazuje různé způsoby uložení grafu v počítači, je na adrese http://mi21.vsb.cz/sites/mi21.vsb.cz/files/unit/reprezentace_grafu_v_pc.pdf.

## Pojmy k zapamatování

— matice sousednosti

— matice incidence

— graf daný seznamem sousedů

# Kapitola 2

# Souvislost grafu

Jestliže máme počítačovou nebo dopravní síť, tak je přirozené zkoumat, zda je v síti možno vyslat signál, případně cestovat nebo dopravovat produkty z místa $X$ do místa $Y$. Bude-li tato síť reprezentována grafem, budeme zkoumat existenci cest mezi vrcholy. Zavedeme proto pojem sledu, tahu a cesty v grafu. Předpokládáme, že mezi vrcholy je dovoleno putovat pouze po hranách grafu.

Navíc je důležité umět rozpoznat, jak odolná je daná síť vůči poruchám, které mohou narušit komunikaci nebo transport v síti. Výpadky mohou být dvojího druhu: porucha může nastat jednak v rámci každého spojení, které odpovídá hraně grafu, nebo v křižovatkách/uzlech sítě, které odpovídají vrcholům grafu. V druhé části kapitoly si ukážeme, jak takovou odolnost vůči výpadkům měřit a že je možno ji popsat číselným parametrem.

## 2.1 Souvislost grafu, komponenty grafu


### Průvodce studiem

*Abychom mohli popsat, co to znamená, že graf je souvislý, zavedeme takzvaný* sled v grafu*. Zjednodušeně můžeme říci, že sled mezi dvěma vrcholy $u$, $v$ popisuje putování v grafu z vrcholu $u$ do vrcholu $v$ po hranách a vrcholech grafu. Vysvětlíme, co to znamená, že daná dopravní či komunikační síť je souvislá nebo nesouvislá.*

### Cíle

Po prostudování této sekce budete schopni:

- vysvětlit, co to znamená, že daná dopravní či komunikační síť je souvislá nebo nesouvislá,
- pro malé grafy ověřit, zda daná síť je souvislá nebo nesouvislá,
- najít komponenty souvislosti sítě.

Abychom uměli popsat, co to znamená, že graf je „souvislý“, zavedeme následující pojem.

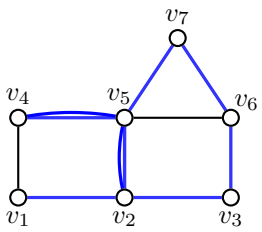
**Definice 2.1.** *Sledem* $v_0v_n$ v grafu $G$ rozumíme takovou posloupnost vrcholů a hran

$$(v_0, e_1, v_1, e_2, v_2, \ldots, e_n, v_n),$$

kde $v_i$ jsou vrcholy grafu $G$ a $e_i$ jsou hrany grafu $G$, přičemž každá hrana $e_i$ má koncové vrcholy $v_{i-1}$ a $v_i$. Počet hran nazveme *délkou sledu* $v_0v_n$.
Vrchol $v_0$ nazýváme *počáteční* a vrchol $v_n$ *koncový*.

Pro algoritmy, které mají prohledávat graf, je taková definice velmi šikovná! Všimněte si, že sled je jakýsi záznam o putování v grafu. Máme posloupnost vrcholů a hran, přičemž hrany a vrcholy se pravidelně střídají. Pokud graf reprezentuje silniční síť, tak sled v grafu je vlastně detailní popis cesty: „vyjedeme z Ostravy po cestě číslo 56, přijedeme do Frýdku-Místku, odkud dále pokračujeme po silnici číslo 48 do Českého Těšína.“

V jednoduchém grafu, tedy v takovém grafu, který neobsahuje násobné hrany, je mezi každou dvojicí vrcholů jen jedna hrana nebo žádná hrana. Můžeme se proto domluvit, že při popisu sledu *v jednoduchém grafu* nemusíme hrany vypisovat, neboť je můžeme vždy jednoznačně doplnit. Podle této úmluvy bude sled možno zapsat jako posloupnost vrcholů a obvykle nebudeme v zápise sledu používat závorky (jako v Příkladu 2.2). Je však potřeba mít na paměti, že v reálné situaci nemusí být předpoklad o jednoznačnosti hrany splněn. Například z Ostravy můžeme do Frýdku-Místku přijet jednak po dálnici (cesta číslo 56) nebo po silnici první třídy číslo 477. Proto „sled“ zapsaný stručně podle úmluvy „z Ostravy pojedeme do Frýdku-Místku a dále pokračujeme do Českého Těšína“ není obecně určen jednoznačně.




Obrázek 2.1 Příklad sledu v grafu.

**Příklad 2.2.** V grafu na Obrázku 2.1 je příklad sledu mezi vrcholy $v_1$ a $v_5$. Sled

$$v_1, v_1v_2, v_2, v_2v_5, v_5, v_5v_7, v_7, v_7v_6, v_6, v_6v_3, v_3, v_3v_2, v_2, v_2v_5, v_5, v_5v_4, v_4, v_4v_5, v_5$$

je vyznačen modrou křivkou. Podle úmluvy můžeme uvedený sled zapsat stručně

$$v_1, v_2, v_5, v_7, v_6, v_3, v_2, v_5, v_4, v_5.$$

Dvě křivky mezi vrcholy $v_4$ a $v_5$ a mezi vrcholy $v_2$ a $v_5$ nejsou násobné hrany, ale znázorňují opakování hrany v modře vyznačeném sledu. Protože sled obsahuje 9 hran (a tedy 10 vrcholů) je délka sledu 9. Připomeňme, že 9 hran nemusí být 9 *různých* hran a 10 vrcholů nemusí být 10 *různých* vrcholů, některé hrany nebo vrcholy se mohou i vícekrát opakovat.

Posloupnost, která obsahuje jediný prvek, třeba $v_1$ je také sled v daném grafu. Jedná se o *triviální sled* délky nula. ▲

**Příklad 2.3.** Všimněte si, že posloupnost vrcholů $v_1, v_2, v_3, v_4, v_5, v_6$ neurčuje žádný sled v grafu na Obrázku 2.1, protože hrana $v_3v_4$ v daném grafu není. ▲

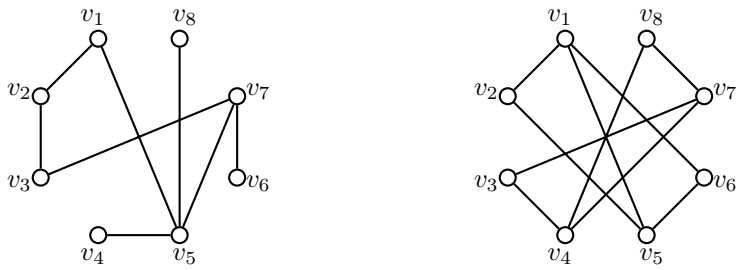
Pojem „souvislosti“ je vybudován na pojmu sled.

**Definice 2.4.** Řekneme, že vrchol $v$ je *dosažitelný* z vrcholu $u$, jestliže v grafu existuje sled z vrcholu $u$ do vrcholu $v$.
Graf nazveme *souvislý*, jestliže pro každé dva vrcholy $u$, $v$ je vrchol $v$ dosažitelný z vrcholu $u$. V opačném případě je graf *nesouvislý*.

Jestliže v neorientovaném grafu existuje sled z vrcholu $u$ do vrcholu $v$, tak obrácením pořadí vrcholů sledu dostaneme sled z vrcholu $v$ do vrcholu $u$. Pokud není nutné pečlivě rozlišovat počáteční a koncový vrchol, říkáme, že existuje sled mezi vrcholy $u$ a $v$.

Rozhodnout o tom, zda graf je nebo není souvislý patří mezi základní úlohy, které pro daný graf řešíme. Odpověď je obvykle zajímavá v kontextu praktické úlohy, kterou daný graf modeluje: „je silniční síť souvislá?“ Odpověď je za normální situace kladná. Avšak v extrémních případech musíme umět rychle rozhodnout, zda „je silniční síť souvislá v kalamitní situaci, kdy došlo k uzavření některých silnic?“ Později v této kapitole ukážeme algoritmus, který bude umět otázku zodpovědět pro libovolný graf.

**Příklad 2.5.** Jsou grafy na Obrázku 2.2 souvislé?




Obrázek 2.2 Grafy $G$ a $H$.

*Řešení.* Graf $G$ (na Obrázku 2.2 vlevo) je souvislý, neboť mezi každými dvěma vrcholy existuje sled. Například vrchol $v_8$ je z vrcholu $v_1$ dosažitelný, neboť $v_1, v_5, v_8$

je sled mezi vrcholy $v_1$ a $v_8$. Sled mezi libovolnou dvojicí vrcholů můžete získat vybráním vhodné části sledu $v_6, v_7, v_3, v_2, v_1, v_5, v_8, v_5, v_4$, který obsahuje všechny vrcholy grafu $G$.

Naproti tomu graf $H$ (na Obrázku 2.2 vpravo) není souvislý, neboť žádný sled mezi vrcholy $v_1$ a $v_8$ v grafu $H$ neexistuje. ▲

**Příklad 2.6.** Mějme pětiprvkovou množinu $A = [1, 5]$. Sestavíme graf $G$, jehož vrcholy budou všechny dvouprvkové podmnožiny množiny $A$. Dva vrcholy spojíme hranou, pokud jsou odpovídající podmnožiny disjunktní. Ukažte, že graf $G$ je souvislý.

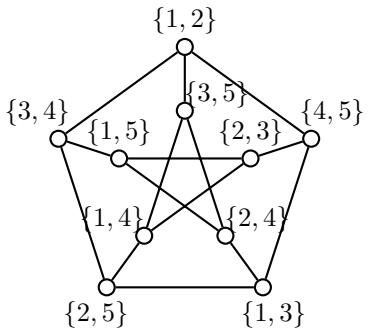
*Řešení.* Ukážeme, že graf $G$ splňuje definici souvislosti, tj. najdeme sled mezi libovolnými dvěma vrcholy. Označme $V_1$, $V_2$ nějaké dva vrcholy grafu $G$. (Uvědomte si, že vrcholy jsou dvouprvkové podmnožiny $A$ a že platí $V_1, V_2 \subset A$.) Rozlišíme tři možnosti:

1. $|V_1 \cap V_2| = 0$. Podle definice grafu $G$ je mezi vrcholy $V_1$ a $V_2$ hrana, neboť se jedná o disjunktní podmnožiny.

2. $|V_1 \cap V_2| = 1$. Množiny $V_1$ a $V_2$ mají jeden prvek společný a proto $|V_1 \cup V_2| = 3$. Označme $V_3 = A \setminus (V_1 \cup V_2)$, množina $V_3$ obsahuje zbývající dva prvky množiny $A$. Podle definice jsou v grafu $G$ hrany $V_1V_3$ i $V_2V_3$, neboť $V_1$ a $V_3$ a také $V_2$ a $V_3$ jsou disjunktní dvojice podmnožin. Proto $V_1, V_3, V_2$ je sled mezi vrcholy $V_1$ a $V_2$.

3. Zbývá prozkoumat případ $|V_1 \cap V_2| = 2$. Protože $V_1$ i $V_2$ jsou dvouprvkové podmnožiny, musí být $V_1 = V_2$ a vrchol $V_1$ je současně triviální sled mezi $V_1$ a $V_2$.

Našli jsme sled mezi každými dvěma vrcholy grafu $G$, proto je graf $G$ podle Definice 2.4 souvislý. Jedno možné nakreslení grafu $G$ je na Obrázku 2.3. ▲




Obrázek 2.3 Graf $G$ je Petersenův graf.

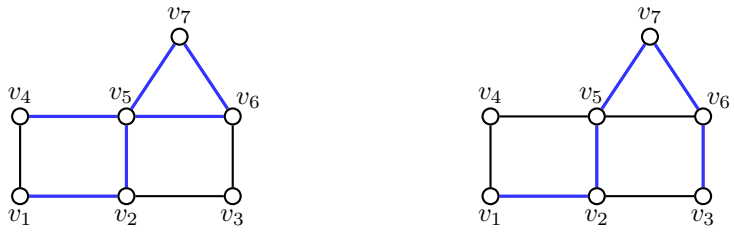
Někdy však není žádoucí, aby se při putování v grafu opakovaly hrany nebo vrcholy. Opakuje-li se vrchol ve sledu, který modeluje silniční síť, tak nejspíš bloudíme,

opakuje-li se hrana ve sledu, který modeluje dopravní síť, tak mrháme energií na dopravu. Zavedeme následující definici.

**Definice 2.7.** *Tah* je sled, ve kterém se neopakují žádné hrany a *cesta* je sled, ve kterém se neopakují žádné vrcholy (tedy ani hrany).

Název „tah“ vychází z analogie kreslení jedním tahem, neboť při kreslení grafu můžeme všechny hrany nějakého tahu nakreslit jedním tahem. Název „cesta“ odpovídá definici cesty v Kapitole 1.2, protože vrcholy a hrany takového sledu tvoří podgraf, který je cestou.

Všimněte si, že v daném grafu je každá cesta zároveň tahem a že každý tah je současně sledem. Opačné implikace obecně neplatí.




Obrázek 2.4 Příklad tahu v grafu z vrcholu $v_1$ do vrcholu $v_4$ a příklad cesty z vrcholu $v_1$ do vrcholu $v_3$.

**Příklad 2.8.** V grafu na Obrázku 2.4 vlevo je příklad tahu mezi vrcholy $v_1$ a $v_4$. Hrany tahu $v_1, v_2, v_5, v_7, v_6, v_5, v_4$ jsou vyznačeny modře.

Vpravo je pak příklad cesty mezi vrcholy $v_1$ a $v_3$. Hrany cesty $v_1, v_2, v_5, v_7, v_6, v_3$ jsou vyznačeny modře.

Všimněte si, že cesta na Obrázku 2.4 vpravo je současně tahem i cestou v daném grafu. Naproti tomu tah na Obrázku 2.4 vlevo je současně sledem, avšak není cestou, neboť vrchol $v_5$ se v něm vyskytuje dvakrát. ▲

Animovaný příklad sledu, tahu a cesty je na adrese `http://mi21.vsb.cz/sites/mi21.vsb.cz/files/unit/sled_tah_cesta.pdf`.

Tahům v grafu se budeme věnovat podrobněji v Kapitole 3. Následující věta ukazuje, že každý sled mezi dvěma vrcholy lze nahradit cestou.

**Věta 2.9.** *Pokud mezi dvěma vrcholy grafu $G$ existuje sled, pak mezi nimi existuje cesta.*

*Důkaz.* Mějme nějaký sled $S$ mezi vrcholy $u$, $v$ v grafu $G$. Vrcholy a hrany sledu $S$ označme $u = v_0, e_1, v_1, \ldots, e_n, v_n = v$, kde $n$ je délka sledu $S$. Ukážeme, jak ze sledu $S$ sestavit cestu $P$ mezi vrcholy $u$, $v$ (v cestě se žádný vrchol neopakuje).

Pokud se ve sledu $S$ žádný vrchol neopakuje, tak $P = S$ je hledanou cestou.

Pokud se některý vrchol opakuje, tak označme $v_i$ jeho první výskyt a $v_j$ jeho poslední výskyt. Celý úsek mezi $v_i$ a $v_j$ vynecháme (včetně vrcholu $v_j$) a ponecháme jen první výskyt vrcholu $v_i$. Dostaneme tak sled $S'$, ve kterém se daný vrchol již neopakuje.

Pokud se ve sledu $S'$ neopakuje už žádný jiný vrchol, tak $P = S'$ je hledaná cesta. Pokud se některý jiný vrchol opakuje, celý postup zopakujeme pro další takový vrchol.

Postup je jistě konečný, neboť graf $G$ obsahuje konečně mnoho vrcholů a po nejvýše $|V(G)|$ krocích dostaneme sled $P$, ve kterém se žádný vrchol neopakuje a který je hledanou cestou mezi vrcholy $u$ a $v$. □

Podle Věty 2.9 víme, že existuje-li mezi některými dvěma vrcholy grafu sled, tak mezi nimi existuje také cesta. Navíc důkaz Věty 2.9 je konstruktivní, neboť dává návod, jak takovou cestu z daného sledu zkonstruovat. Definici souvislosti bychom tedy mohli ekvivalentně vyslovit i takto: „Graf nazveme souvislý, jestliže mezi každými dvěma vrcholy existuje cesta.“ Na druhou stranu sled má jednu důležitou vlastnost, kterou tah ani cesty v grafu nemají. Spojením dvou sledů, kdy na posloupnost vrcholů jednoho sledu navážeme vrcholy druhého sledu (koncový vrchol prvního sledu splyne s počátečním vrcholem druhého sledu), dostaneme opět sled. Toto obecně *není* pravda pro cesty, ani pro tahy. Proto je pojem sledu šikovný pro argumentaci řady tvrzení týkajících se souvislosti.

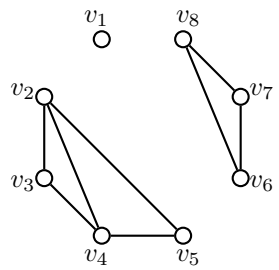
**Příklad 2.10 (Pokračování Příkladu 2.2).** V grafu na Obrázku 2.4 je vyznačen sled z vrcholu $v_1$ do vrcholu $v_5$. Podle postupu popsaného v důkazu Věty 2.9 sestavte cestu z vrcholu $v_1$ do vrcholu $v_5$.

*Řešení.* Ve sledu $v_1, v_2, v_5, v_7, v_6, v_3, v_2, v_5, v_4, v_5$ se vrchol $v_2$ opakuje dvakrát a vrchol $v_5$ dokonce třikrát. Nejprve vynecháme úsek mezi prvním a posledním (druhým) výskytem vrcholu $v_2$. Dostaneme sled $v_1, v_2, v_5, v_4, v_5$. Dále vynecháme úsek mezi prvním a posledním (nyní druhým) výskytem vrcholu $v_5$. Dostaneme sled $v_1, v_2, v_5$, který je současně cestou mezi vrcholy $v_1$ a $v_5$. ▲

Jestliže graf není souvislý, tak sestává z několika „částí“, které souvislé jsou. Tyto části se nazývají „komponenty“, jejich přesnou definici uvedeme později na straně 34. Pro jednoduchost můžeme říci, že komponentu tvoří každý takový podgraf, který je souvislý a současně obsahuje co nejvíce vrcholů a hran původního grafu. Například graf na Obrázku 2.5 má tři komponenty. Je důležité si uvědomit, že pokud bychom řekli pouze, že komponenta je souvislý podgraf, tak například indukovaný podgraf na vrcholech $v_3$, $v_4$ a $v_5$ je souvislý podgraf grafu na Obrázku 2.5, který však není tím, co čekáme pod pojmem „komponenta grafu“.

## Pro zájemce:

Pojem souvislosti je možno vybudovat s využitím sledů a relací. Mějme nějaký graf $G$ a na jeho vrcholové množině $V(G)$ zavedeme relaci $\sim$ tak, že dva vrcholy $u, v \in V(G)$ jsou




Obrázek 2.5 Graf se třemi komponentami.

v relaci $\sim$ (píšeme $u \sim v$) právě tehdy, když v grafu $G$ existuje sled $uv$. Relaci $\sim$ říkáme *relace dosažitelnosti*.

**Lemma 2.11.** *Relace dosažitelnosti $\sim$ je relací ekvivalence.*

*Důkaz.* Připomeňme, že binární relace je relací ekvivalence, pokud je reflexivní, symetrická a tranzitivní. Abychom ukázali, že relace $\sim$ je ekvivalencí, ověříme všechny tři vlastnosti

1. Reflexivita relace $\sim$ plyne snadno z existence triviálního sledu $uu$ délky 0. Pro každý vrchol $u \in V(G)$ proto platí $u \sim u$.

2. Symetrie relace $\sim$ je zřejmá, neboť ke každému sledu z vrcholu $u$ do vrcholu $v$ můžeme v grafu $G$ sestavit sled z vrcholu $v$ do vrcholu $u$ tak, že vezmeme posloupnost vrcholů a hran v opačném pořadí. Za předpokladu, že graf $G$ je neorientovaný, tak pro libovolnou dvojici vrcholů $u, v \in V(G)$ platí $u \sim v \Leftrightarrow v \sim u$.

3. A konečně abychom ověřili tranzitivitu stačí si uvědomit, že spojením sledu z vrcholu $u$ do vrcholu $v$ a sledu z vrcholu $v$ do vrcholu $w$ dostaneme opět sled, a sice sled z vrcholu $u$ do vrcholu $w$. Platí proto implikace $u \sim v \wedge v \sim w \Rightarrow y \sim w$ a relace $\sim$ je tranzitivní.

Ukázali jsme, že relace $\sim$ je reflexivní, symetrická a tranzitivní a jedná se proto o relaci ekvivalence. □

Nyní můžeme vyslovit jinou definici souvislosti grafu.

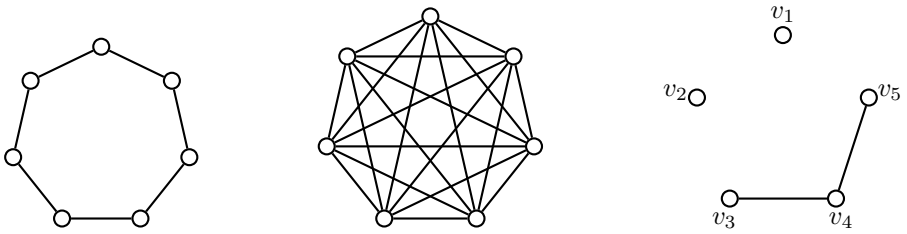
**Definice 2.12.** Řekneme, že graf $G$ je *souvislý*, jestliže relace $\sim$ na množině $V(G)$ je úplná.

Nyní můžeme vyslovit definici komponenty daného grafu $G$. Připomeňme, že v Lematu 2.11 jsme ukázali, že relace $\sim$ je relace ekvivalence. Můžeme proto sestavit rozklad vrcholové množiny grafu $G$ příslušný relaci $\sim$. V každé třídě rozkladu jsou vrcholy, které jsou navzájem dosažitelné.

**Definice 2.13.** Třídy rozkladu příslušného relaci ekvivalence $\sim$ jsou podmnožiny $V(G)$ a podgrafy grafu $G$ indukované na těchto podmnožinách se nazývají *komponenty souvislosti* grafu $G$.

Můžeme vyslovit třetí definici souvislého grafu, která vychází z počtu komponent daného grafu.

**Definice 2.14.** Řekneme, že graf $G$ je *souvislý*, pokud je graf $G$ tvořený jedinou komponentou souvislosti.




Obrázek 2.6 Příklad souvislých grafů $G_1$, $G_2$ a nesouvislého grafu $G_3$.

**Příklad 2.15.** Sestavte relaci $\sim$ pro grafy na Obrázku 2.6. Určete příslušné třídy rozkladu vrcholové množiny.

*Řešení.* Graf $G_1$ je současně cyklem $C_7$, a jedná se proto o souvislý graf. To znamená, že mezi každými dvěma vrcholy najdeme sled a relace $\sim_{G_1}$ obsahuje všechny dvojice: $\sim_{G_1} = V(G) \times V(G)$. Třída rozkladu této relace je jediná, obsahuje všechny vrcholy grafu $G_1$.

Zcela analogicky postupujeme při určování souvislosti grafu $G_2$. Protože graf $G_2$ je kompletní graf $K_7$, dostaneme $\sim_{G_2} = V(G) \times V(G)$. Třída rozkladu této relace je opět jediná, obsahuje všechny vrcholy grafu $G_2$.

Naproti tomu graf $G_3$ není souvislý, obsahuje tři komponenty. Relace $\sim$ obsahuje následující dvojice

$$\begin{aligned} \sim \quad = \quad & \{(v_1,v_1),(v_2,v_2),(v_3,v_3),(v_3,v_4),(v_3,v_5),\\ & (v_4,v_3),(v_4,v_4),(v_4,v_5),(v_5,v_3),(v_5,v_4),(v_5,v_5)\}\,. \end{aligned}$$

Třídy rozkladu této relace ekvivalence jsou tři: $[\sim_{G_3} v_1] = \{v_1\}$, $[\sim_{G_3} v_2] = \{v_2\}$ a $[\sim_{G_3} v_3] = \{v_3, v_4, v_5\}$. ▲

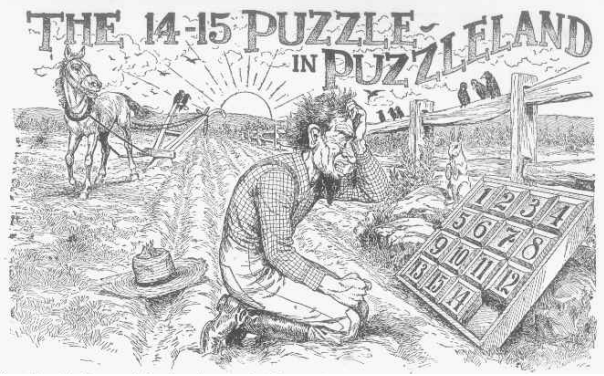
Ukázali jsme, že souvislost grafu je možno popsat několika způsoby. Samozřejmě všechny tři uvedené způsoby jsou ekvivalentní, to znamená, že je-li graf souvislý dle Definice 2.4, tak je také souvislý podle Definice 2.12 i podle Definice 2.14 a naopak. Pro praktické ověření souvislosti se mohou hodit všechny přístupy v závislosti na konkrétní implementaci.

V Kapitole 2.2 ukážeme algoritmus, který najde všechny komponenty daného grafu.

**Příklad 2.16.** Představme si graf $S$, který bude popisovat možné tahy střelce na klasické šachovnici. Připomeňme, že střelec se pohybuje diagonálně o libovolný počet polí v rámci šachovnice. Vrcholy grafu budou políčka šachovnice a hranou spojíme dvě políčka, pokud mezi nimi bude možno táhnout střelcem. Je graf $S$ souvislý?

*Řešení.* Graf $S$ má 64 vrcholů a poměrně mnoho hran, například vrcholy, které odpovídají rohovým políčkům, jsou stupně 7, zatímco vrcholy, které odpovídají čtyřem políčkům blízko středu šachovnice, jsou stupně 13. Není těžké si rozmyslet, že graf $S$ souvislý není, aniž bychom graf sestavovali. Protože střelec se pohybuje pouze diagonálně, nemůže vstoupit na políčko jiné barvy, než na kterém se nachází. Naproti tomu s využitím dostatečného počtu tahů se můžeme dostat na libovolné pole dané barvy. Proto graf $S$ není souvislý a má dvě komponenty souvislosti. Navíc obě komponenty jsou navzájem isomorfní podgrafy. ▲

**Příklad 2.17.** Loydova patnáctka je známý hlavolam, jehož úkolem je přesouváním patnácti dřevěných čtverečků s čísly 1 až 15 v krabici 4 krát 4 políčka zajistit, aby čísla čtena po řádcích tvořila aritmetickou posloupnost 1, 2 až 15. Výchozí pozice je na Obrázku 2.7. Jak by mohl vypadat graf popisující daný problém?




Obrázek 2.7 Loydova patnáctka, dobová ilustrace.

*Řešení.* Pokud bychom sestavili stavový graf dané úlohy (každý vrchol odpovídá jednomu možnému rozmístění čtverečků), bude takový graf $L$ mít 16! vrcholů. Každý vrchol bude spojen s nejvýše čtyřmi dalšími vrcholy (existují nejvýše čtyři čtverečky, které můžeme přesunout na volné políčko). Dá se však ukázat, že takový graf nebude souvislý, ale bude mít dvě komponenty. Výchozí pozice s prohozenými čísly 14 a 15, která je na Obrázku 2.7, se nachází v jiné komponentě, než požadovaný cílový stav. Proto daná úloha nemá řešení.

Existuje vysvětlení, které využívá vlastností permutací a které se obejde bez konstrukce obrovského grafu s 16! = 20922789888000 vrcholy, nicméně vysvětlení překračuje rámec našeho textu. ▲

## Pojmy k zapamatování

— sled v grafu
— souvislý a nesouvislý graf
— tah a cesta v grafu
— komponenty souvislosti

## 2.2 Prohledávání grafu

### Průvodce studiem


*Ukážeme algoritmus, který umožní rychle ověřit, zda daný graf (například graf, který odpovídá reálné úloze) je souvislý nebo není. Algoritmus bude obecný, drobnou modifikací dostaneme variantu algoritmu, která bude hledat komponenty souvislosti grafu (resp. dané sítě). Jinou jednoduchou modifikací dostaneme variantu algoritmu pro „prohledávání do hloubky“ nebo pro „prohledávání do šířky.“ Uvedený algoritmus půjde snadno modifikovat a použít pro* jakékoliv *zpracování struktury grafu, čímž rozumíme zpracování každého vrcholu grafu, případně každé hrany grafu. Nebudeme řešit implementaci algoritmu do konkrétního programovacího jazyka, ale zaměříme se na princip algoritmu.*

### Cíle

Po prostudování této sekce budete schopni:

- algoritmicky ověřit, zda daný graf (daná síť) je souvislý nebo nesouvislý,
- najít všechny komponenty souvislosti libovolného grafu,
- popsat a použít algoritmy, které zpracují daný graf (nebo síť) postupem do hloubky či do šířky.

Ve slíbeném obecném algoritmu pro „procházení“ grafu vystačíme s několika málo datovými typy. Pro každý vrchol budeme rozlišovat jeden ze tří možných stavů, ve kterých se vrchol může v průběhu algoritmu nacházet:

- iniciační – výchozí stav na začátku algoritmu,
- nalezený – jakmile vrchol najdeme jako koncový vrchol nějaké hrany,
- zpracovaný – jakmile prozkoumáme všechny hrany incidentní s tímto vrcholem.

Pro každou hranu grafu budeme rozlišovat dva stavy

- iniciační – výchozí stav na začátku algoritmu,
- zpracovaná – jakmile je hrana nalezena (prozkoumána) jako hrana incidentní s některým vrcholem.

Dále budeme udržovat pomocnou strukturu se seznamem všech vrcholů, které se nacházejí ve stavu „nalezený“ (a dosud „nezpracovaný“). Tomuto seznamu budeme říkat *úschovna*.

Způsob, jakým vybíráme vrcholy z úschovny, určuje různé varianty našeho algoritmu. Může se jednat o procházení grafu postupem „do hloubky“ nebo „do šířky“.

V algoritmu najdete volání funkcí `ZPRACUJ_VRCHOL()` a `ZPRACUJ_HRANU()`. Tyto funkce nemají pro samotné prohledaní grafu žádný význam, avšak využijeme je

později při sestavení algoritmu, který zpracuje graf, tj. pro sestavení algoritmu, který pro každý vrchol nebo pro každou hranu provede nějakou operaci. Příslušnou operaci budou realizovat zmíněné funkce `ZPRACUJ_VRCHOL()` a `ZPRACUJ_HRANU()`.

**Algoritmus 2.1 (Procházení souvislých komponent grafu).**

```
// na vstupu je graf G
vstup < graf G;
stav(všechny vrcholy a hrany G) = iniciační;
uschovna U = {libovolný vrchol u grafu G};
stav(u) = nalezený;

// zpracování vybrané komponenty G
while (U je neprázdná) {
   vyber vrchol v a odeber jej z úschovny "U := U - {v}";
   ZPRACUJ(v);
   for (hrany e vycházející z v) {    // pro všechny hrany
      if (stav(e) == iniciační)
         ZPRACUJ(e);
      w = druhý vrchol hrany e = vw;  // známe sousedy?
      if (stav(w) == iniciační) {
         stav(w) = nalezený;
         přidej vrchol w do úschovny "U := U + {w}";
      }
      stav(e) = zpracovaná;
   }
   stav(v) = zpracovaný;
   // případný přechod na další komponentu G
   if (U je prázdná && G má další vrcholy)
      uschovna U = {vrchol v_1 z další komponenty G};
}
```

Stručně popíšeme hlavní části algoritmu. Na začátku

- všem vrcholům i hranám přiřadíme iniciační stav,
- vybereme libovolný vrchol z úschovny.

V průběhu každého cyklu algoritmu zpracujeme nějaký jeden vrchol $v$. To znamená, že

- zpracovaný vrchol $v$ z úschovny odstraníme,
- prozkoumáme (a případně zpracujeme) všechny dosud nezpracované hrany incidentní s vrcholem $v$,

- pokud je některý vrchol $w$ sousední s vrcholem $v$ v iniciačním stavu, přidáme vrchol $w$ do úschovny $U$.

Prozkoumáme (a zpracujeme) tak celou komponentu grafu, která obsahuje výchozí vrchol `u`.

Jestliže se v grafu nachází další nezpracované vrcholy, víme že graf nebude souvislý a bude mít více komponent. Algoritmus umí najít (a zpracovat) všechny komponenty, pro rozlišení komponent můžeme zpracované vrcholy zařazovat do různých struktur (polí, množin a pod.).

Různým způsobem implementace úschovny dostaneme několik různých variant Algoritmu 2.1.

**Procházení „do hloubky“** Jestliže úschovnu `U` implementujeme jako zásobník, tak vrchol zpracovávaný v dalším cyklu bude poslední nalezený vrchol. Zpracováváme stále další a další, třeba i vzdálenější vrcholy, pokud existují.

**Procházení „do šířky“** Jestliže úschovnu `U` implementujeme jako frontu, tak nejprve zpracujeme všechny vrcholy sousední s vrcholem `u`. Označme si tyto vrcholy pro přehlednost $v_1, v_2, \ldots, v_d$. Dále postupně zpracujeme všechny nezpracované vrcholy, které jsou sousední s vrcholy $v_1, v_2, \ldots, v_d$ (jedná se o vrcholy ve vzdálenosti 2 od výchozího vrcholu `u`; pojem vzdálenosti zavedeme v Kapitole 4), atd.

**Dijkstrův algoritmus pro nejkratší cesty** Z úschovny vybíráme vždy ten vrchol, který je nejblíže k výchozímu vrcholu `u`. Podrobně se tomuto algoritmu budeme věnovat v Kapitole 4.

### Poznámky o složitosti Algoritmu 2.1

Algoritmus 2.1 je nejen přehledný, ale současně rychlý. Počet kroků algoritmu je úměrný součtu počtu vrcholů a hran daného grafu. Obecně můžeme říci, že složitost algoritmu je $O(n+m)$, kde $n$ udává počet vrcholů a $m$ počet hran daného grafu. To znamená, že například pro graf se 100 vrcholy a 400 hranami bude Algoritmus 2.1 potřebovat řádově $c \cdot (100 + 400)$ kroků, kde parametr $c$ závisí na konkrétní implementaci. Pokud bychom měli graf se 100 000 vrcholy a 400 000 hranami, tak počet kroků Algoritmu 2.1 bude řádově $c \cdot (100\ 000 + 400\ 000)$. Ve zmíněném algoritmu lze očekávat, že parametr $c$ je v řádu desítek.

### Úlohy související s prohledáváním grafu

Nyní zmíníme několik typických problémů, které můžeme řešit pomocí Algoritmu 2.1 přímo nebo s jeho drobnou modifikací.

**Příklad 2.18.** Jak pomocí Algoritmu 2.1 vypsat všechny hrany daného grafu?

*Řešení.* Stačí využít funkci `zpracuj(e)`. Jestliže $e$ je zpracovávaná hrana, tak ji v těle funkce `zpracuj(e)` vypíšeme. ▲

**Příklad 2.19.** Jak pomocí Algoritmu 2.1 zjistíme, zda je graf $G$ souvislý?

*Řešení.* Stačí upravit poslední řádek algoritmu. Jestliže je úschovna U prázdná a v grafu $G$ jsou další vrcholy, tak graf $G$ není souvislý. Protože jsme zpracovali všechny vrcholy a hrany komponenty, která obsahuje vrchol $u$, tak úschovna $U$ je prázdná. Ale graf obsahuje další vrcholy, které nejsou dosažitelné z vrcholu $u$ a není proto souvislý.

Naopak, jestliže je úschovna $U$ prázdná a žádné další vrcholy v grafu $G$ nejsou, jsou všechny vrcholy dosažitelné z vrcholu $u$ a graf $G$ je souvislý. ▲

**Příklad 2.20.** Jak pomocí Algoritmu 2.1 najít a označit všechny komponenty grafu $G$?

*Řešení.* Zavedeme pomocnou proměnnou označující číslo komponenty, například $k$. Na začátku algoritmu položíme $k = 1$ a při zpracování každého vrcholu mu přiřadíme číslo komponenty $k$.

Dále upravíme poslední řádek algoritmu. Jestliže je úschovna U prázdná a současně jsou v grafu $G$ další vrcholy, tak graf $G$ není souvislý. Do úschovny uložíme libovolný zbývající vrchol `v_1` a zvýšíme hodnotu proměnné $k$. Při zpracování vrcholů pak každému vrcholu přiřazujeme již nové číslo komponenty.

Podobně postupujeme i pro zbývající komponenty. ▲

## Pojmy k zapamatování

— prohledávání grafu

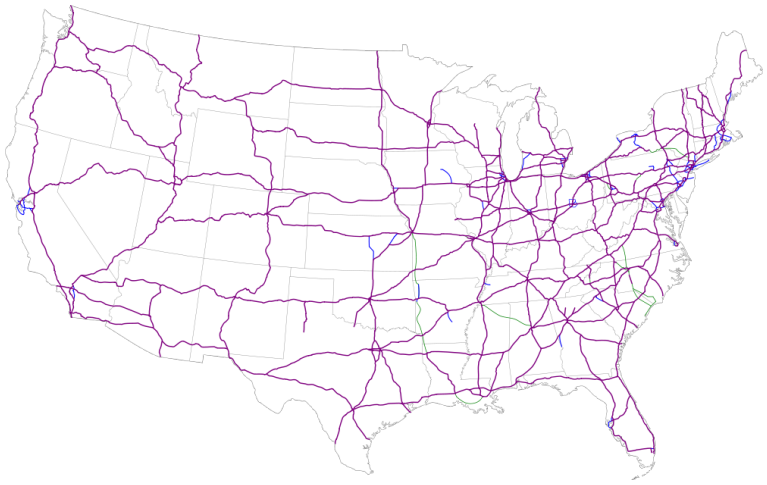
— postup do šířky a do hloubky

# 2.3 Vyšší stupně souvislosti


## Průvodce studiem

*Jestliže nějaký graf reprezentuje dopravní nebo komunikační síť, tak je důležité vědět, jak je taková síť odolná vůči lokálním výpadkům, které mohou narušit transport či komunikaci v síti. Výpadky mohou být dvojího druhu: jednak může porucha nastat u každého spojení, které odpovídá hraně grafu, ale také v křižovatkách/uzlech sítě, které odpovídají vrcholům grafu. V této kapitole si ukážeme, jak lze souvislost grafu měřit a jak popsat odolnost grafu vůči poruchám číselnými parametry.*

Na Obrázku 2.8 je dálniční síť v USA. Graf této sítě je evidentně souvislý a současně je zřejmé, že případná uzavírka některé jedné dálnice nejspíš příliš nenaruší dopravu mezi východním a západním pobřežím, neboť mezi atlantským a tichomořským existuje řada alternativních cest.



Obrázek 2.8 Eisenhowerův systém dálnic v USA.

## Cíle

Po prostudování této sekce budete schopni:

- popsat číselný parametr, který charakterizuje „stupeň“ souvislosti grafu,
- na základě znalosti stupně souvislosti rozhodnout, zda porušení (odebrání) jistého počtu vrcholů nebo hran naruší souvislost sítě,
- vysvětlit, jak disjunktní cesty mezi vrcholy v grafu zajistí vyšší souvislost, tj. vyšší odolnost příslušné sítě vůči poruchám.

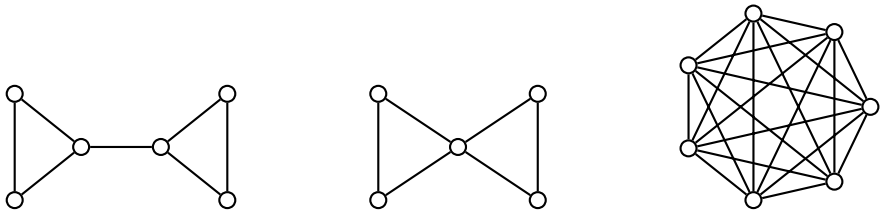
V praxi nás zajímá nejen, jestli v dané síti existuje spojení (cesta) mezi vrcholy grafu, ale také, jestli bude existovat nějaké spojení v případě lokálních výpadků. V řeči teorie grafů se ptáme, zda odebrání několika hran nebo vrcholů poruší souvislost grafu. Dostáváme se tak přirozeně k pojmům *stupeň vrcholové souvislosti a stupeň hranové souvislosti.*

**Definice 2.21.** Graf $G$ je *hranově $k$-souvislý*, pokud $k \geqq 1$ a po odebrání libovolných $k-1$ hran grafu $G$ zůstane výsledný faktor souvislý.
Stupeň hranové souvislosti grafu $G$ je takové největší číslo $k$, že graf $G$ je hranově $k$-souvislý.

Všimněte si, každý souvislý graf je podle uvedené definice automaticky hranově 1-souvislý, neboť odebráním 0 hran (neodebereme žádnou hranu) zůstane graf souvislý.

Dále je nutno upozornit, že v definici hranové souvislosti není řečeno, které hrany se odebírají. Má-li graf zůstat souvislý po odebrání *libovolných* $k-1$ *hran*, musíme uvážit všechny možnosti, jak $k-1$ hran odebrat. Jestliže existuje alespoň jedna množina nějakých $k-1$ hran, že jejich odebráním dostaneme nesouvislý graf, tak daný graf *není* hranově $k$-souvislý. Jinými slovy ještě jednou musíme zdůraznit, že

nestačí šikovně odstranit $k-1$ hran grafu $G$ tak, aby výsledný graf zůstal souvislý a pak můžeme graf $G$ prohlásit za hranově $k$-souvislý. Například z kompletního grafu $K_7$ na Obrázku 2.9 vpravo lze sice šikovně odebrat až 15 hran a výsledný podgraf zůstane souvislý, avšak nesouvislý podgraf můžeme dostat vhodným odebráním pouhých šesti hran a proto stupeň hranové souvislosti grafu $K_7$ není větší než 6 (později ukážeme, že stupeň hranové souvislosti grafu $K_7$ je roven 6).



Obrázek 2.9 Tři grafy s různými stupni souvislosti.

Není těžké si rozmyslet, že každý graf, který je hranově $k$-souvislý, je podle definice také hranově $(k-1)$-souvislý, hranově $(k-2)$-souvislý, atd., až hranově 1-souvislý, neboť nestačí-li odebrat $k-1$ hran, abychom dostali nesouvislý faktor, tak pro porušení souvislosti nemůže stačit odebrat *méně než* $k-1$ hran. Současně si uvědomme, že graf, který *není* hranově $k$-souvislý, nemůže být ani hranově $(k+1)$--souvislý, hranově $(k+2)$-souvislý, atd., protože stačí-li pro porušení souvislosti odebrat některých $k-1$ hran, můžeme odebrat i více hran (pochopitelně ne více, než je počet hran celého grafu) a souvislost porušit.

**Příklad 2.22.** Určete stupně hranové souvislosti grafů na Obrázku 2.9.

*Řešení.* Každý ze tří uvedených grafů je souvislý, proto podle definice hranové $k$--souvislosti jsou všechny grafy hranově 1-souvislé.

V grafu na obrázku vlevo stačí vhodně odebrat jednu hranu a dostaneme nesouvislý graf, proto graf vlevo *není* hranově 2-souvislý. Stupeň hranové souvislosti prvního grafu je proto 1.

Rozebráním šesti možností snadno nahlédneme, že odebráním jedné hrany grafu na obrázku uprostřed souvislost neporušíme. Proto uvedený graf je také hranově 2--souvislý. Není však již hranově 3-souvislý, neboť stačí odebrat dvě hrany incidentní s některým vrcholem stupně 2 a dostaneme nesouvislý podgraf (faktor). Stupeň hranové souvislosti druhého grafu je proto 2.

Konečně v grafu na obrázku vpravo neporušíme souvislost odebráním 1, 2, 3, 4 ani 5 hran, proto je kompletní graf $K_7$ hranově 2-, 3-, 4-, 5- a 6-souvislý. Podrobné zdůvodnění zahrnuje vyšetření mnoha případů, které zde nerozepisujeme, protože později ukážeme jednodušší argument. Avšak odebráním všech šesti hran incidentních s některým pevně zvoleným vrcholem, dostaneme nesouvislý podgraf (faktor), proto graf $K_7$ *není* hranově 7-souvislý. Stupeň hranové souvislosti grafu $K_7$ je proto 6. ▲

Zcela analogicky zavedeme stupně vrcholové souvislosti. Budeme zkoumat, zda odebráním vrcholů (a samozřejmě všech hran s nimi incidentních), dostaneme souvislý nebo nesouvislý graf.

**Definice 2.23.** Graf $G$ je *vrcholově $k$-souvislý*, pokud $|V(G)| > k \geqq 1$ a po odebrání libovolných $k-1$ vrcholů z grafu $G$ zůstane výsledný indukovaný podgraf souvislý.
Stupeň vrcholové souvislosti grafu $G$ je takové největší číslo $k$, že graf $G$ je vrcholově $k$-souvislý.

Všimněte si, že na rozdíl od definice hranové souvislosti požadujeme, aby číslo $k$ bylo ostře menší než počet vrcholů. Nemělo by smysl zjišťovat, zda po odstranění všech vrcholů (a příslušných hran) je výsledný graf souvislý, protože po odstranění všech vrcholů nezůstane žádný graf!

Podobně jako u hranové $k$-souvislosti můžeme vyslovit následující jednoduché pozorování: každý graf, který je vrcholově $k$-souvislý, je podle definice také vrcholově $(k-1)$-souvislý, vrcholově $(k-2)$-souvislý, atd. až vrcholově 1-souvislý. V důsledku graf, který *není* vrcholově $k$-souvislý, není ani vrcholově $(k+1)$-souvislý, vrcholově $(k+2)$-souvislý, atd., protože pro porušení souvislosti stačí odebrat některých $(k-1)$ vrcholů.

**Příklad 2.24.** Určete stupeň vrcholové souvislosti grafů na Obrázku 2.9.

*Řešení.* Všechny grafy na Obrázku 2.9 jsou souvislé, proto podle definice vrcholové $k$-souvislosti jsou vrcholově 1-souvislé.

V grafu na obrázku vlevo i v grafu na obrázku uprostřed stačí odebrat jediný vhodný vrchol a dostaneme nesouvislý graf, proto ani jeden z těchto grafů *není* vrcholově 2-souvislý. Stupeň vrcholové souvislosti je pro oba grafy roven 1.

V grafu na obrázku vpravo souvislost neporušíme odebráním 1, 2, 3, 4 ani 5 vrcholů, proto je kompletní graf $K_7$ také vrcholově 2-, 3-, 4-, 5- a 6-souvislý. Všimněte si, že po odebrání $t$ vrcholů, kde $1 \leqq t \leqq 6$ dostaneme *souvislý* podgraf $K_{7-t}$. Ale i když odebráním libovolných 6 vrcholů dostaneme souvislý podgraf $K_1$, tak graf $K_7$ *není* vrcholově 7-souvislý, neboť podle definice volíme $k < |V(G)|$. Stupeň vrcholové souvislosti grafu $K_7$ je proto 6. ▲

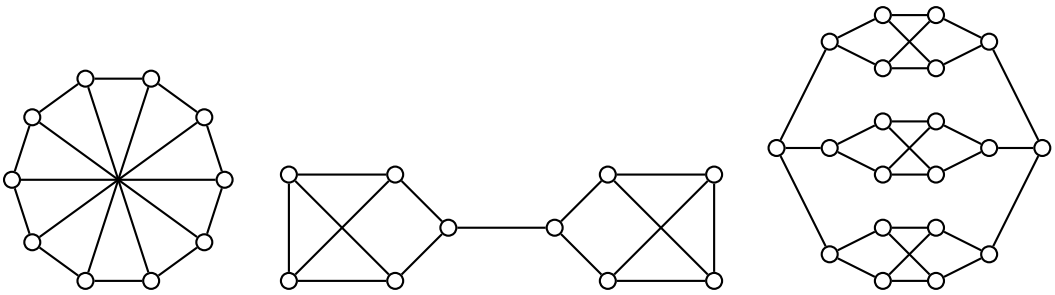
Všimněte si, že graf na Obrázku 2.9 uprostřed je vrcholově 1-souvislý a hranově 2-souvislý. Obecně je možno ukázat, že stupeň vrcholové souvislosti nemůže být vyšší než stupeň hranové souvislosti. Není proto možno například sestavit graf, který by byl hranově 2-souvislý a vrcholově 3-souvislý.

Současně je zřejmé, že stupeň hranové souvislosti nemůže být větší než nejmenší stupeň vrcholu v daném grafu, neboť odebráním všech hran incidentních s vrcholem nejmenšího stupně dostaneme nesouvislý graf. Následující větu uvedeme bez důkazu (důkaz lze najít například v [5]).

**Věta 2.25.** *Pro libovolný graf $G$ je vrcholový stupeň souvislosti nejvýše roven hranovému stupni souvislosti a ten je nejvýše roven nejmenšímu stupni $\delta(G)$.*

## Pro zájemce:

Pro pravidelné grafy, ve kterých jsou všechny vrcholy stupně 3, je možno ukázat ještě silnější tvrzení. V každém 3-pravidelném grafu je vrcholový i hranový stupeň souvislosti vždy roven stejnému číslu. Například graf krychle (Obrázek 1.4) i Petersenův graf (Obrázek 1.9) jsou hranově i vrcholově 3-souvislé, dále například graf na Obrázku 2.10 vlevo je hranově i vrcholově 3-souvislý, graf na Obrázku 2.10 uprostřed je hranově i vrcholově 1-souvislý a konečně graf na Obrázku 2.10 vpravo je hranově i vrcholově 2-souvislý.



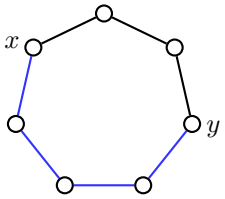
Obrázek 2.10 3-pravidelné grafy.

Mějme dán nějaký graf $G$ a v něm dvě cesty $P$ a $P'$ (cestou v grafu rozumíme podgraf, který je cestou). Řekneme, že cesty $P$ a $P'$ jsou *hranově disjunktní*, jestliže žádná hrana grafu nepatří současně do obou cest $P$ i $P'$. Následující věta pojednává o hranově disjunktních cestách a o cestách, které nemají společné vrcholy s výjimkou prvního a posledního vrcholu. Řekneme, že cesty $P$ a $P'$ jsou *interně disjunktní*, jestliže žádný vrchol grafu není současně vnitřním vrcholem obou cest $P$ i $P'$.

Následující věty dokázal německý matematik Karl Menger (čti „Menger“) v roce 1927.

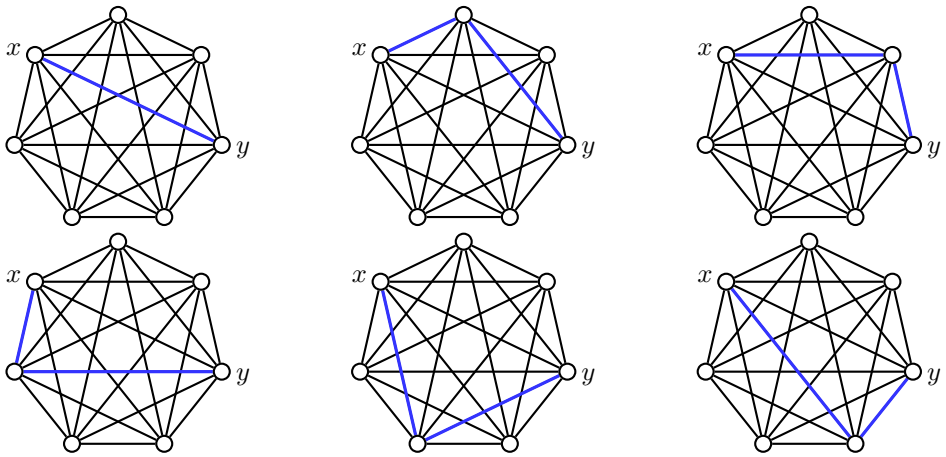
**Věta 2.26 (Mengerovy věty).** *Netriviální graf $G$ je hranově $k$-souvislý právě tehdy, když mezi libovolnými dvěma vrcholy existuje alespoň $k$ po dvou hranově disjunktních cest (vrcholy mohou být sdílené).*
*Netriviální graf $G$ je vrcholově $k$-souvislý právě tehdy, když mezi libovolnými dvěma vrcholy existuje alespoň $k$ po dvou interně disjunktních cest (koncové vrcholy jsou společné).*

Důkaz obou částí věty je možné postavit na existenci maximálního toku, kterému se budeme věnovat v Kapitole 7 na straně 149. Například v cyklu najdeme mezi každou dvojicí různých vrcholů dvě hranově disjunktní cesty i interně disjunktní cesty (Obrázek 2.11).



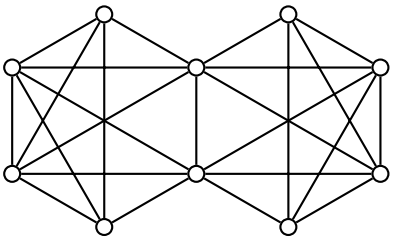

Obrázek 2.11 Dvě hranově i interně disjunktní cesty mezi $x$ a $y$ v cyklu $C_7$.

Netriviální kompletní graf $K_n$ je hranově i vrcholově $(n-1)$-souvislý. Podle Věty 2.26 to znamená, že mezi každou dvojicí vrcholů existuje $n-1$ interně disjunktních cest, které jsou současně hranově disjunktní. Například na Obrázku 2.12 je šest interně disjunktních cest mezi dvěma vybranými vrcholy $x$, $y$. Žádné dvě modře vyznačené cesty nemají společnou hranu ani společný vrchol (s výjimkou koncových vrcholů $x$ a $y$).



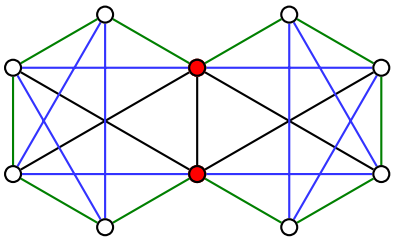

Obrázek 2.12 Šest různých interně disjunktních cest mezi vrcholy $x$, $y$ v grafu $K_7$.

**Příklad 2.27.** Určete stupeň hranové souvislosti i stupeň vrcholové souvislosti grafu $G$ na Obrázku 2.13.



Obrázek 2.13 Grafy $G$.

*Řešení.* Je zřejmé, že graf $G$ je souvislý a odebráním jediného (libovolného) vrcholu se souvislost neporuší. Naproti tomu odebráním obou vrcholů označených červeně na Obrázku 2.14 dostaneme nesouvislý graf, proto stupeň vrcholové souvislosti grafu $G$ je 2.



Obrázek 2.14 Hranově disjunktní cesty v grafu $G$.

Stupeň hranové souvislosti je nejvýše 4, neboť v grafu jsou vrcholy stupně 4 a odebráním všech hran incidentních s takovým vrcholem bychom dostali nesouvislý graf. Navíc mezi každou dvojicí vrcholů najdeme čtyři hranově disjunktní cesty (vždy dvě cesty jsou v Obrázku 2.14 vyznačeny zeleně a dvě modře), proto je podle Věty 2.26 graf $G$ hranově alespoň 4-souvislý. Dostáváme tak, že stupeň hranové souvislosti grafu $G$ je 4. ▲

## Pro zájemce:

Stupeň hranové souvislosti je možno definovat a určovat také pomocí řezů v grafu. Pojem řezu zavedeme v Kapitole 7, kde současně uvedeme algoritmus, který hledá minimální řez v síti. Tento algoritmus je možno mírně upravit a použít pro určení stupně hranové souvislosti libovolného grafu.

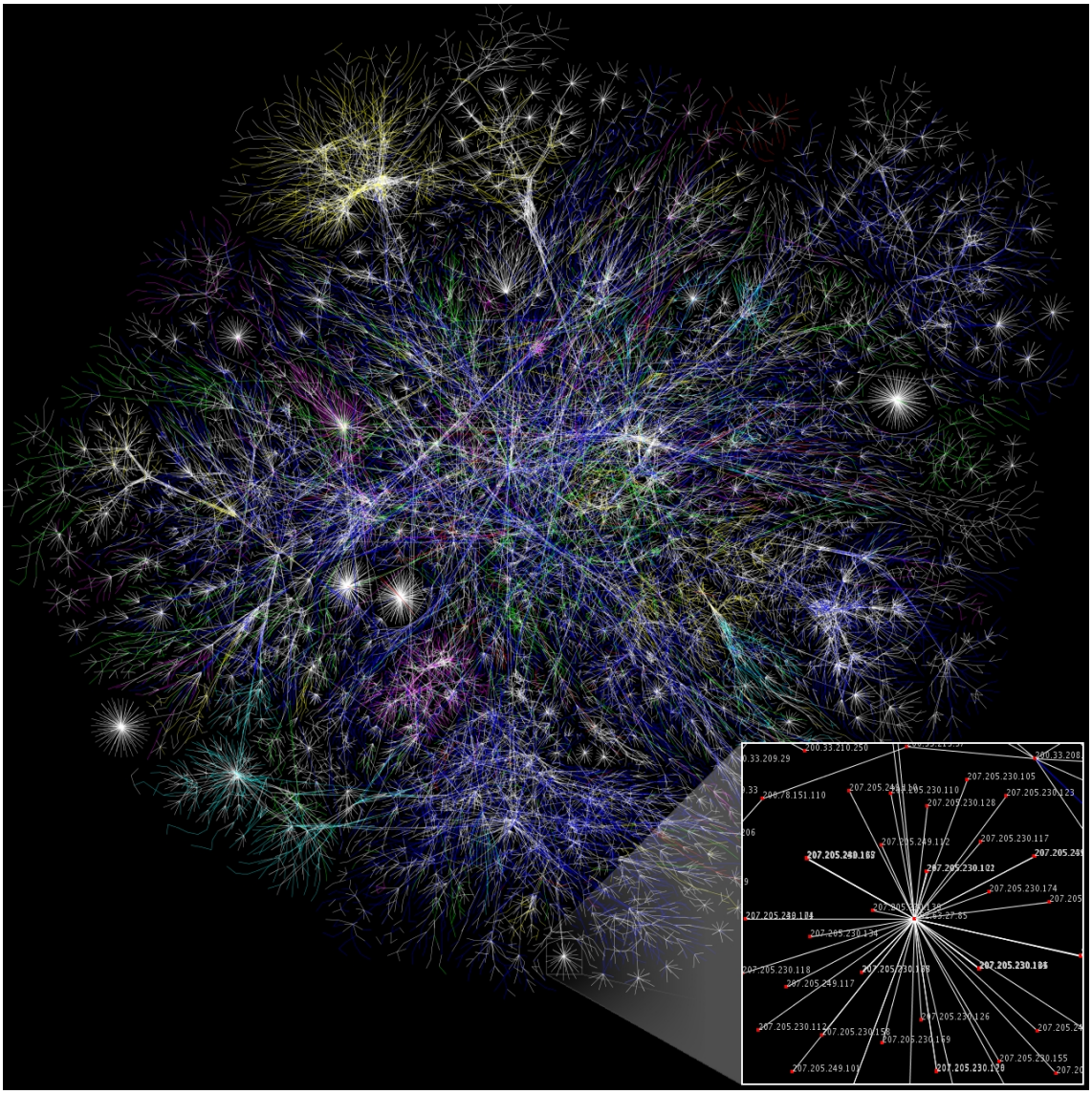
Internet (World Wide Web) bývá často znázorňován podobně jako na Obrázku 2.15. Takové zobrazení však nevystihuje jeden důležitý aspekt internetu – jeho poměrně vysokou hranovou i vrcholovou souvislost mezi páteřními uzly. Vždyť právě vysoká souvislost a schopnost doručení zprávy mezi libovolnými dvěma vrcholy byl jeden z klíčových požadavků, který stál u zrodu internetu.

## Pojmy k zapamatování

— stupeň vrcholové souvislosti

— stupeň hranové souvislosti

— Mengerovy věty




Obrázek 2.15 Znázornění části webu.

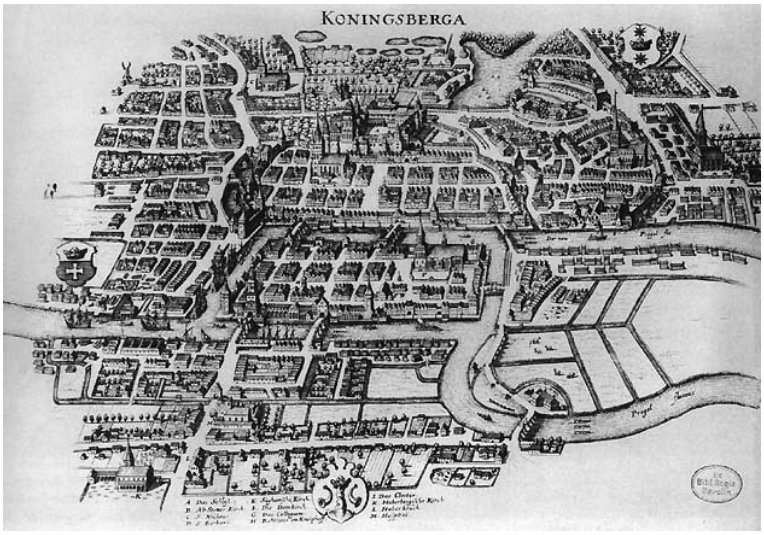
# Kapitola 3

# Eulerovské a hamiltonovské grafy


## Průvodce studiem

*Historicky první problém vyřešený pomocí teorie grafů v roce 1736 byl tak zvaný* Problém sedmi mostů města Královce.

*V osmnáctém století byl Královec bohatým městem. Na tehdejší dobu vyspělá ekonomika umožnila městu vystavět celkem sedm mostů přes řeku Pregolu (Obrázek 3.1). Podle tradice trávili měšťané nedělní odpoledne procházkami po městě. Vznikla tak otázka, zda je možno projít všech sedm královeckých mostů, každý z nich právě jednou. Žádnému z měšťanů se to nedařilo, avšak nikdo z nich ani neuměl zdůvodnit, proč by to nebylo možné.*




Obrázek 3.1 Dobová mapa města Královce.

*Oslovili Leonharda Eulera, který v té době žil v Petrohradu, zda by problém uměl vyřešit. Euler (čti „ojler“) úlohu snadno vyřešil. Uvědomil si, že při jejím řešení nevyužil ani geometrii, ani algebru ani známe početní metody, ale metodu, kterou ve své korespon-*

*denci s německým matematikem Leibnizem nazvali „geometrie pozic." Euler při řešení problému sedmi mostů města Královce tuto metodu formálně zavedl. Dnes bychom řekli, že použil teorii grafů při hledání tahu, který obsahuje každou hranu daného grafu právě jedenkrát.*

*Pošťák má za úkol roznést poštu do každé ulice ve svém okrsku. Je přirozené, že si naplánuje trasu tak, aby každou ulicí procházel pokud možno jen jednou – nachodí se tak co nejméně a navíc poštu doručí dříve. Představme si graf, který modeluje ulice okrsku tak, že hrany odpovídají ulicím a křižovatky vrcholům grafu. Pošťákova úloha tak přesně odpovídá hledání tahu, který obsahuje každou hranu právě jednou.*

*Podobně můžeme plánovat trasu sněžné frézy, která odklízí chodníky. Možná budeme chtít některé ulice projít dvakrát – jednou po každé straně kde se nachází chodník. Grafový model snadno upravíme tak, aby chodníky po obou stranách ulice odpovídaly násobným hranám nebo interně disjunktním cestám v grafu.*

*Podobně mohou trasy svých vozů plánovat popeláři, kropící vozy, vozidla technických služeb atd.*

## 3.1 Kreslení jedním tahem

### Cíle

Po prostudování této sekce budete schopni:

- zformulovat některé praktické úlohy jako úlohy nalezení eulerovského tahu,
- rozpoznat, zda daný graf je možno nakreslit (projít) jedním uzavřeným tahem,
- rozpoznat, zda daný graf je možno nakreslit jedním (ne nutně uzavřeným) tahem.

V úvodu jsme zmínili celou řadu úloh, které lze řešit tak, že v nějakém grafu hledáme tah, který obsahuje každou hranu grafu právě jedenkrát.

**Definice 3.1.** Tah $T$, který začíná a končí ve stejném vrcholu daného grafu se nazývá *uzavřený tah*. Uzavřený tah v souvislém grafu $G$, který navíc obsahuje všechny hrany grafu $G$, se nazývá *uzavřený eulerovský tah*. Tah v souvislém grafu $G$, který obsahuje všechny hrany grafu $G$ a výchozí vrchol se liší od koncového vrcholu, se nazývá *otevřený eulerovský tah*.
Graf, ve kterém existuje uzavřený eulerovský tah, se nazývá *eulerovský graf*.

Říkáme, že graf, ve kterém existuje otevřený nebo uzavřený tah a který obsahuje všechny hrany grafu, lze *nakreslit jedním (otevřeným nebo uzavřeným) tahem*. Při popisu úlohy hledání uzavřeného nebo otevřeného eulerovského tahu v daném grafu budeme v dalším textu obvykle hovořit jako o kreslení jedním tahem, ačkoliv hlavní motivací daného problému nemusí být právě kreslení jedním tahem, ale

třeba optimální řešení nějakého dopravního problému. Je to proto, že kreslení jedním tahem je názorné a velmi dobře popisuje důležité vlastnosti:

- hledaný tah má obsahovat každou hranu grafu,
- žádná hrana se neopakuje,
- uzavřený tah začíná a končí ve stejném vrcholu.

Budeme-li říkat „graf $G$ lze nakreslit jedním uzavřeným tahem“, tak tím současně říkáme „v grafu $G$ existuje uzavřený eulerovský tah“.

**Poznámka 3.2.** Všimněte si, že v definici eulerovského tahu požadujeme, aby daný graf $G$ byl souvislý. Uvědomte si, že například graf, který obsahuje dvě komponenty $C_5$ *není* podle definice eulerovský, ačkoliv má všechny vrcholy sudého stupně. Dokonce ani graf, jehož jedna komponenta je eulerovský graf a další komponenty jsou izolované vrcholy není eulerovský, ačkoliv v něm najdeme uzavřený tah, který obsahuje *všechny* hrany celého grafu.

**Příklad 3.3.** Které z grafů na Obrázku 3.2 lze nakreslit jedním tahem?


Obrázek 3.2 Které z uvedených grafů lze nakreslit jedním tahem?

*Řešení.* Budeme-li zkoušet nakreslit kompletní graf $K_4$ jedním tahem, nepodaří se nám to. Abychom to ukázali nade vší pochybnost, mohli bychom třeba systematicky prozkoumat všechny možnosti, jak seřadit hrany. Graf $K_4$ má 6 hran a ty lze seřadit celkem $P(6) = 6! = 720$ způsoby, avšak ani jeden z nich neodpovídá eulerovskému tahu v grafu. Ve Větě 3.4 ukážeme překvapivě jednoduchou podmínku, která pomůže snadno rozpoznat, zda v daném grafu (a tedy například i v grafu $K_4$) existuje eulerovský tah.

Naproti tomu v grafu na obrázku vpravo existuje otevřený eulerovský tah (dokonce několik různých), například $x_1, x_2, x_3, x_4, x_1, x_3, x_5, x_4, x_2$. Na druhou stranu, pokud bychom v grafu hledali uzavřený eulerovský tah, tak neuspějeme. V grafu neexistuje ani otevřený eulerovský tah, který by začínal nebo končil v některém z vrcholů $x_3$, $x_4$ nebo $x_5$. ▲

Následující věta dává jednoduché kriterium, kdy v daném grafu existuje uzavřený eulerovský tah. Právě toto tvrzení zformuloval a dokázal Leonhard Euler už v roce 1736.

**Věta 3.4 (Eulerova věta).** *Graf $G$ lze nakreslit jedním uzavřeným tahem právě tehdy, když je graf $G$ souvislý a všechny jeho vrcholy jsou sudého stupně.*

*Důkaz.* Ukážeme si myšlenku důkazu. Formální důkaz vyžaduje pečlivé značení, najdete jej na straně 54. Protože tvrzení věty má tvar ekvivalence (říkáme, že něco platí „právě tehdy, když“ platí něco jiného), budeme pro netriviální grafy dokazovat dvě implikace.

1. Nejprve ukážeme implikaci "$\Rightarrow$" : „jestliže lze graf nakreslit jedním uzavřeným tahem, tak je souvislý a všechny jeho vrcholy jsou sudého stupně.“

   V triviálním grafu je tvrzení splněno triviálně. Mějme netriviální graf $G$, který je možno nakreslit jedním uzavřeným tahem. Abychom zdůvodnili, že graf $G$ je souvislý, stačí si uvědomit, že mezi každými dvěma vrcholy najdeme sled tak, že z uzavřeného eulerovského tahu vybereme příslušný úsek.

   Dále, protože uzavřený eulerovský tah do každého vrcholu vždy nějakou hranou vstoupí a jinou hranou pokračuje, tak stupeň každého vrcholu je nějaký násobek čísla 2 a je proto sudý. Uvědomte si, že v uzavřeném tahu můžeme výchozí vrchol zvolit libovolně, proto sudého stupně je vskutku *každý* tedy i výchozí vrchol grafu $G$.

2. Nyní ukážeme opačnou implikaci "$\Leftarrow$" , která zní: „jestliže máme souvislý graf a všechny jeho vrcholy jsou sudého stupně, tak jej lze nakreslit jedním uzavřeným tahem.“

   Důkaz je konstruktivní, umožní nám uzavřený eulerovský tah najít. Postupujeme indukcí vzhledem k počtu hran.

   *Základ indukce:* Nejmenší graf, který je souvislý a má všechny vrcholy sudého stupně, je triviální graf. V triviálním grafu existuje triviální uzavřený sled a pro tento graf tvrzení platí.

   Nejmenší souvislý netriviální graf $G$ se všemi vrcholy sudého stupně je cyklus $C_n$, protože v netriviálním souvislém grafu nemohou být vrcholy stupně 0 a právě v cyklu $C_n$ jsou všechny vrcholy stupně 2. Cyklus $C_n$ je jistě možno nakreslit jedním uzavřeným tahem, neboť hrany cyklu tvoří hledaný uzavřený sled.

   *Indukční krok:* Mějme nějaký netriviální souvislý graf $G = (V, E)$ se všemi vrcholy sudého stupně. Předpokládejme, že každý souvislý graf s méně než $|E|$ hranami a se všemi vrcholy sudého stupně je možno nakreslit jedním uzavřeným tahem. V grafu $G$ jistě najdeme nějaký uzavřený tah $T$ (ne nutně eulerovský). Při hledání tahu $T$ můžeme začít v libovolném vrcholu a protože každý vrchol grafu $G$ je stupně alespoň 2, můžeme vždy pokračovat hranou, která se v tahu zatím nevyskytovala. Protože graf je konečný, dříve nebo později se v tahu zopakuje nějaký vrchol. Dostaneme tak uzavřený tah. Ale

pozor! Vrchol, který se zopakuje, *nemusí být první vrchol sestaveného tahu.* Úsek sestaveného tahu, který začíná prvním výskytem zopakovaného vrcholu, je hledaný uzavřený tah $T$.

Odebereme-li nyní všechny hrany uzavřeného tahu $T$, dostaneme graf označený $G - T$, ve kterém jsou opět všechny vrcholy sudého stupně (některé mohou být izolované vrcholy). Pokud graf $G - T$ není souvislý, lze každou jeho komponentu $L_i$ dle indukčního předpokladu nakreslit jedním uzavřeným tahem $T_i$. Nyní do uzavřeného tahu $T$ přidáme za každou komponentu uzavřený tah $T_i$ (tah $T_i$ je v komponentě eulerovský a obsahuje všechny hrany komponenty) a získáme uzavřený eulerovský tah původního grafu $G$.

Podle principu (silné) matematické indukce je důkaz hotov.

Ukázali jsme, že nutnou i dostatečnou podmínku pro existenci uzavřeného eulerovského tahu v grafu je, aby graf byl souvislý a měl všechny vrcholy sudého stupně. □

Na adrese http://mi21.vsb.cz/sites/mi21.vsb.cz/files/unit/eulerova_veta.pdf najdete animaci, která ilustruje jednotlivé kroky důkazu.

Eulerova věta řeší pouze existenci *uzavřeného* eulerovského tahu. Jak je to s existencí otevřeného eulerovského tahu ukazuje následující tvrzení.

**Věta 3.5.** *Graf $G$ lze nakreslit jedním otevřeným tahem právě tehdy, když je graf $G$ souvislý a právě dva jeho vrcholy jsou lichého stupně.*

*Důkaz.* Tvrzení Věty 3.5 má opět tvar ekvivalence a budeme dokazovat dvě implikace.

"⇒" Stejně jako v důkazu Věty 3.4 zdůvodníme, že můžeme-li graf $G$ nakreslit jedním otevřeným tahem, je graf $G$ souvislý a všechny vrcholy jsou sudého stupně s výjimkou prvního a posledního vrcholu otevřeného tahu. První i poslední vrchol jsou různé (tah je otevřený) a každý z nich je incidentní s lichým počtem hran, neboť první hrana tahu a poslední hrana tahu přispějí do stupně vrcholu jedničkou, zatímco vnitřní vrcholy tahu jsou incidentní vždy se sudým počtem různých hran tahu.

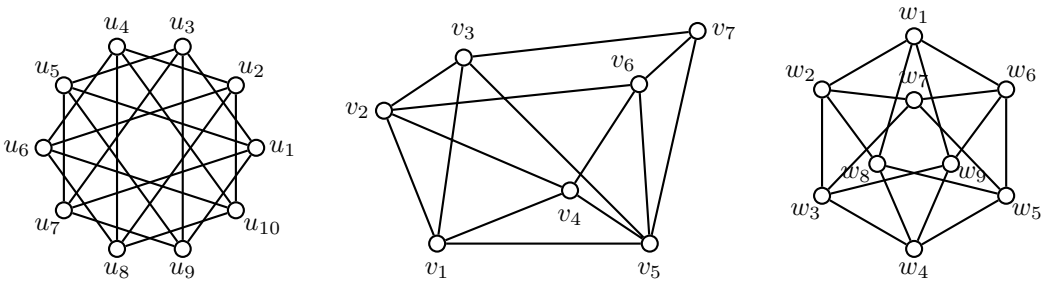
"⇐" Máme-li souvislý graf $G$ s právě dvěma vrcholy $u$, $v$ lichého stupně, můžeme do grafu $G$ přidat nový vrchol $x$, který spojíme hranami s vrcholy $u$, $v$ a dostaneme souvislý graf $G'$, ve kterém jsou všechny vrcholy (dokonce i vrcholy $u$, $v$, $x$) sudého stupně a podle Eulerovy věty 3.4 v grafu $G'$ existuje *uzavřený* eulerovský tah $T'$, který začíná a končí ve vrcholu $x$. Je zřejmé, že z uzavřeného tahu $T'$ stačí vynechat vrchol $x$ a obě přidané hrany a dostaneme otevřený eulerovský tah $T$ mezi vrcholy $u$ a $v$ v původním grafu $G$. □

Všimněte si, že důkaz Věty 3.5 jsme elegantním trikem převedli na tvrzení, pro které jsme využili už dokázanou Větu 3.4. Zcela analogicky bychom mohli ukázat už první implikaci "⇒" Věty 3.5. Proto je Věta 3.5 považována za důsledek

Eulerovy věty 3.4 a pro konstrukci otevřeného eulerovského tahu můžeme použít stejný postup jako pro konstrukci uzavřeného eulerovského tahu.

Je dobré zdůraznit, že existuje-li v daném grafu uzavřený eulerovský tah, tak v něm *neexistuje* otevřený eulerovský tah. Existuje-li totiž v grafu uzavřený eulerovský tah, tak všechny vrcholy grafu jsou sudého stupně a aby existoval otevřený eulerovský tah, tak musí v grafu být právě dva vrcholy lichého stupně.

**Příklad 3.6.** Které z grafů na Obrázku 3.3 je možno nakreslit jedním tahem? Najděte alespoň jeden takový tah, pokud existuje.




Obrázek 3.3 Které z uvedených grafů lze nakreslit jedním tahem?

*Řešení.* Graf na Obrázku 3.3 vlevo má sice všechny vrcholy sudého stupně, ale není souvislý. Proto v něm neexistuje uzavřený ani otevřený eulerovský tah.

Graf na Obrázku 3.3 uprostřed je souvislý a má právě dva vrcholy lichého stupně: $v_5$ a $v_7$. V grafu existuje otevřený eulerovský tah, například

$$v_7, v_6, v_4, v_1, v_2, v_3, v_5, v_1, v_3, v_7, v_5, v_6, v_2, v_4, v_5,$$

a tedy jej lze nakreslit jedním otevřeným tahem.

Graf na Obrázku 3.3 vpravo je souvislý a má všechny vrcholy sudého stupně, proto v něm existuje uzavřený eulerovský tah, například

$$w_1, w_2, w_7, w_5, w_8, w_2, w_3, w_4, w_8, w_1, w_9, w_4, w_5, w_6, w_7, w_3, w_9, w_6, w_1.$$

▲

Další animovaný příklad je k dispozici na adrese http://mi21.vsb.cz/sites/mi21.vsb.cz/files/unit/eulerovsky_tah.pdf.

**Poznámka 3.7.** Pokud bychom chtěli sestavit algoritmus, který bude v daném grafu hledat uzavřený eulerovský tah, tak nejprve můžeme snadno ověřit, zda takový tah vůbec existuje. Souvislost grafu ověříme užitím postupu popsaného v Příkladu 2.19 a ověřit, zda je každý vrchol sudého stupně je snadné. Nyní najdeme libovolný uzavřený tah $T$ postupem, který jsme popsali v indukčním kroku důkazu Věty 3.4.

Dále najdeme komponenty souvislosti grafu $G - T$ (podle Příkladu 2.20) a pro každou netriviální komponentu najdeme rekurzivně uzavřený eulerovský tah

(eulerovský vzhledem ke komponentě, nikoliv vzhledem k původnímu grafu $G$). Tyto uzavřené tahy vhodně vložíme do uzavřeného tahu $T$ a dostaneme uzavřený eulerovský tah původního grafu $G$.

Stejný trik jako v důkazu Věty 3.5 můžeme použít i pro důkaz ještě obecnějšího tvrzení: každý souvislý graf s $2k$ vrcholy lichého stupně lze nakreslit $k$ otevřenými tahy. Vzhledem k praktickým motivacím, které jsme zmiňovali v úvodu kapitoly se kreslení více tahy může hodit při plánování tras pro více pošťáků nebo pro více popelářských vozů.

## Pro zájemce:

Pro úplnost zde uvádíme i pečlivě zapsaný důkaz Eulerovy věty 3.4. Doporučujeme tento důkaz porovnat s důkazem uvedeným na straně 51.

*Důkaz.* Protože Eulerova věta 3.4 má tvar ekvivalence, ukážeme platnost dvou implikací.
"⇒" Uzavřený eulerovský tah v grafu $G$ při každém průchodu vrcholem $v$ obsahuje dvě hrany incidentní s $v$. Navíc první a poslední hrana uzavřeného tahu jsou incidentní se stejným vrcholem, proto stupeň každého vrcholu v grafu $G$ je násobek dvojky. Každý eulerovský graf $G$ má proto všechny vrcholy sudého stupně.
"⇐" Postupujeme indukcí vzhledem k počtu hran.
*Základ indukce:* Nejmenší graf splňující podmínky věty je triviální graf, který obsahuje triviální uzavřený sled s jediným vrcholem.
*Indukční krok:* Mějme nějaký netriviální souvislý graf $G$ se všemi vrcholy sudého stupně. Předpokládejme, že všechny souvislé sudé grafy s menším počtem hran obsahují eulerovský tah. Protože graf $G$ má všechny vrcholy stupně alespoň 2 (vrchol stupně menšího by byl triviální komponentou), tak podle Lemmatu 5.3 (větu ukážeme v Kapitole 5) obsahuje graf $G$ nějaký cyklus $C$. V grafu $G' = G - E(C)$ zůstane každý vrchol sudého stupně, neboť odebereme vždy 0 nebo 2 hrany incidentní s každým vrcholem. Proto každá komponenta $L_i$ grafu $G'$ je souvislý sudý graf s méně hranami než $G$ a podle indukčního předpokladu obsahuje komponenta $L_i$ uzavřený eulerovský tah. Navíc každá komponenta $L_i$ obsahuje nějaký vrchol cyklu $C$, jinak by původní graf nebyl souvislý. V každé komponentě $L_i$ označme $v_i$ jeden takový vrchol cyklu $C$. Nyní postupujeme podle cyklu $C$ a vždy, když narazíme na nějaký vrchol $v_i$, tak na stávající tah navážeme uzavřený eulerovský tah komponenty $L_i$ a dále pokračujeme po cyklu $C$. Sestavíme tak uzavřený tah, který bude obsahovat všechny hrany cyklu $C$ i všech komponent grafu $G - E(C)$, proto se jedná o eulerovský tah v grafu $G$. □

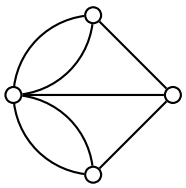
## Pro zájemce:

Využití eulerovských grafů se neomezují pouze na putování v grafu, který reprezentuje nějakou mapu. Další pěkné aplikace eulerovských grafů najdeme pro stavové grafy. Vrcholy stavového grafu nějakého systému odpovídají možným stavům, které mohou nastat, a hrany spojují stavy, mezi kterými existuje legální přechod. Například konečné automaty

jsou jedním typem stavových grafů. Při testu systému bychom rádi prověřili všechny možné stavy a přechody mezi nimi. Optimální test můžeme naplánovat podle eulerovského tahu grafem.

## Pro zájemce:

Není těžké ukázat, že platnost Eulerovy věty 3.4 je možno rozšířit i pro grafy s násobnými hranami (multigrafy), ve kterých je stupeň každého vrcholu určen počtem incidentních hran. Všimněte si, že dokonce původní úloha sedmi mostů města Královce vede na řešení úlohy v multigrafu, který je na Obrázku 3.4.



Obrázek 3.4 Graf úlohy sedmi mostů města Královce.

## Pro zájemce:

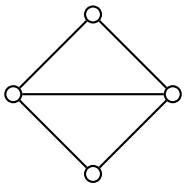
V důkazu Věty 3.4 je popsán algoritmus konstrukce eulerovského tahu, jehož složitost je polynomiální vzhledem k počtu vrcholů. Vhodnou implementací můžeme dostat algoritmus jehož složitost je dokonce menší než $O(n^2)$, kde $n$ udává počet vrchol daného grafu.

## Pro zájemce:

Na okraj uvedeme, že v dnešním Královci je možné projít každý z mostů právě jednou, procházka však není turisticky atraktivní, neboť musí začínat na jednom z ostrovů a končit na druhém. V centru dnešního Kaliningradu je nyní mostů pět. Dva ze sedmi mostů z původní úlohy byly zničeny během druhé světové války. Další dva byly později zbořeny a nahrazeny novými. Zbývající tři mosty zůstaly zachovány, přičemž jeden z nich byl v roce 1935 přestavěn. V terminologii teorie grafů by oběma břehům odpovídaly vrcholy stupně 2 a oběma ostrovům vrcholy stupně 3 (Obrázek 3.5).

## Pojmy k zapamatování

— kreslení jedním tahem
— Eulerova věta

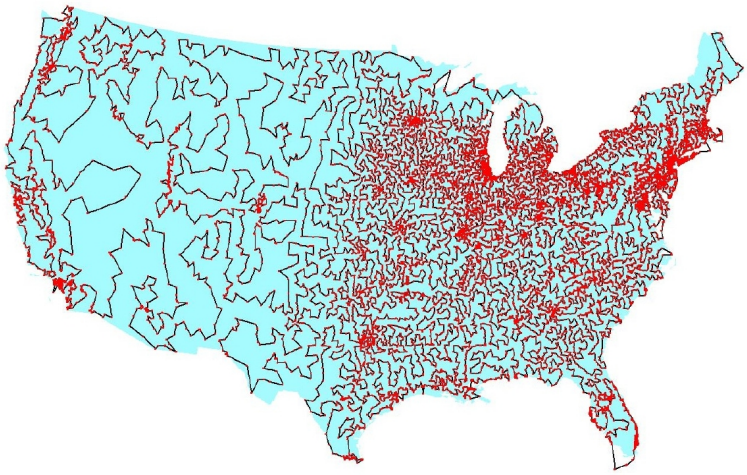


Obrázek 3.5 Graf úlohy sedmi mostů dnešního města Královce.

## 3.2 Hamiltonovské grafy


### Průvodce studiem

*Při hledání eulerovského tahu v grafu jsme chtěli navštívit každou hranu daného grafu právě jedenkrát, přičemž opakované navštívení vrcholu nehrálo roli. Přirozeně vzniká analogická úloha, kdy nemusíme projít každou hranu, ale chceme navštívit každý vrchol právě jednou. Jako klasická praktická motivace takové úlohy se uvádí problém obchodního cestujícího, který má navštívit všechna města svého regionu, vrátit se do výchozího města a přitom nacestovat co nejkratší vzdálenost. Ve zjednodušené verzi nás zajímá, zda můžeme navštívit každé město právě jednou a vrátit se zpět. V terminologii teorie grafů hledáme cyklus, který obsahuje všechny vrcholy daného grafu. Takový cyklus může, ale nemusí existovat.*



Obrázek 3.6 Optimální řešení úlohy obchodního cestujícího pro 13 509 měst v USA.

*Další podobné úlohy odpovídají plánování trasy pošťáka na vesnici, kde se roznáší jen málo poštovních zásilek. Spíš než projít každou ulici, musí pošťák navštívit všechny adresy (domy), do kterých má doručit nějakou zásilku. Podobně ve velkoskladu se při navážení nebo předávání zboží rozváží paleta, nebo třeba několik palet s různým zbožím na určitá místa ve skladu. Obsluha vozíku má opět za úkol navštívit všechna vybraná*

*místa a přitom urazit co nejkratší vzdálenost. Všechny uvedené úlohy odpovídají stejné úloze teorie grafů – nalezení tzv. hamiltonovského cyklu v grafu.*

## Cíle

Po prostudování této sekce budete schopni:

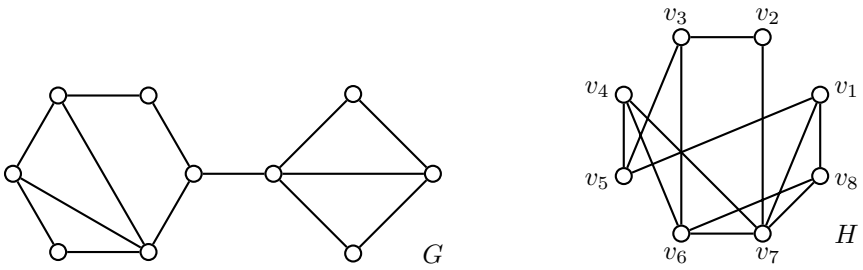
- popsat úlohu, která odpovídá hledání hamiltonovského cyklu v grafu,
- použít některé jednoduché dostatečné podmínky pro nalezení hamiltonovského cyklu.

Nejprve zavedeme následující pojem.

**Definice 3.8.** *Hamiltonovský cyklus* v grafu je takový cyklus, který prochází všemi vrcholy daného grafu. Graf, ve kterém existuje hamiltonovský cyklus, se nazývá *hamiltonovský graf*.

Hamiltonovský cyklus v grafu prochází každým vrcholem právě jedenkrát. Takový cyklus nemusí pochopitelně obsahovat všechny hrany daného grafu.

**Příklad 3.9.** Které z grafů na Obrázku 3.7 jsou hamiltonovské?




Obrázek 3.7 Které z uvedených grafů obsahují hamiltonovský cyklus?

*Řešení.* Snadno odhadneme, že graf na Obrázku 3.7 vlevo není hamiltonovský. Zdůvodníme to nepřímo. Kdyby hamiltonovský byl, tak by obsahoval nějaký hamiltonovský cyklus $C$ a byl by současně vrcholově (i hranově) 2-souvislý, neboť mezi každou dvojicí vrcholů bychom našli na cyklu $C$ dvě interně disjunktní cesty. Ale protože vrcholová souvislost grafu na Obrázku 3.7 vlevo je rovna 1 (v grafu existuje hrana, jejímž odebráním dostaneme nesouvislý graf), tak tento graf neobsahuje hamiltonovský cyklus.

Graf na Obrázku 3.7 vpravo je hamiltonovský, neboť obsahuje hamiltonovský cyklus. Jeden takový cyklus je například

$$v_1, v_7, v_2, v_3, v_5, v_4, v_6, v_8.$$

▲

Hamiltonovský graf zavedli jako analogii eulerovského grafu. Na první pohled se zdá, že hledání hamiltonovského cyklu je velmi podobné hledání eulerovskému tahu. Není tomu tak! Zatímco pro rozhodnutí o existenci eulerovského tahu v souvislém grafu je nutnou i dostatečnou podmínkou sudost všech stupňů vrcholů, tak pro existenci hamiltonovského cyklu žádná taková snadno ověřitelná podmínka není známa. Řada matematiků se dokonce domnívá, že takovou podmínku ani nelze najít.

Není znám jednoduchý a současně rychlý algoritmus, pomocí kterého bychom v daném grafu našli hamiltonovský cyklus a nebo poznali, že takový cyklus neexistuje. Pochopitelně je možno prověřit všech $n!$ různých pořadí vrcholů, zda leží po řadě na nějakém cyklu, avšak takový postup vyžaduje pro obecný graf až $n!$ (řádově exponenciálně mnoho) kroků v závislosti na počtu vrcholů daného grafu. Algoritmus, který by existenci hamiltonovského cyklu ověřil obecně s využitím nejvýše polynomiálně mnoha kroků, znám není. Pro obecný graf proto není lehké rozhodnout, zda je hamiltonovský. Je známo alespoň několik jednoduchých dostatečných podmínek. Jestliže graf splňuje například některou z následujících podmínek, tak víme, že obsahuje hamiltonovský cyklus.

**Věta 3.10 (Diracova věta).** *Mějme graf $G$ na $n$ vrcholech, kde $n \geqq 3$. Je-li nejmenší stupeň vrcholů $v$ grafu alespoň $n/2$, tak je graf $G$ hamiltonovský*

Diracova věta 3.10 (čti „Dirakova“) je speciálním případem následujícího tvrzení, které je obecnější.

**Věta 3.11 (Oreho věta).** *Mějme graf $G$ na $n$ vrcholech, kde $n \geqq 3$. Jestliže pro každé dva nesousední vrcholy $u$ a $v$ v grafu $G$ platí $\deg(u) + \deg(v) \geqq n$, tak je graf $G$ hamiltonovský*

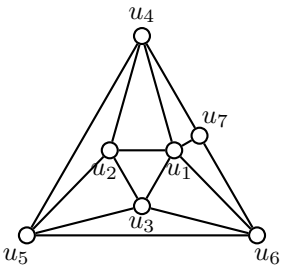
Důkaz můžete najít například v [5]. Všimněte si, že obě věty mají tvar implikace, nikoli ekvivalence. Proto graf, který splňuje podmínky je hamiltonovský, ale ne každý hamiltonovský graf musí podmínky věty splňovat. Například každý graf $C_n$ pro $n \geqq 3$ je jistě hamiltonovský, ale pro $n > 4$ nesplňuje podmínky ani Věty 3.10, ani Věty 3.11. Také graf na Obrázku 3.7 vpravo je hamiltonovský, ale uvedené dostatečné podmínky nesplňuje.

**Příklad 3.12.** Které kompletní bipartitní grafy $K_{m,m}$ jsou hamiltonovské?

*Řešení.* Každý bipartitní graf $K_{m,m}$ pro $m \geqq 2$ je hamiltonovský podle Diracovy věty, neboť má $2m$ vrcholů a stupeň každého vrcholu je (alespoň) polovina počtu vrcholů. Pro $m = 1$ Diracovu větu nemůžeme použít, protože graf $K_{1,1}$ má jen dva vrcholy. Na druhou stranu je zřejmé, že graf $K_{1,1}$ hamiltonovský není. ▲

**Příklad 3.13.** Rozhodněte zda je graf na Obrázku 3.8 hamiltonovský.

*Řešení.* Graf na Obrázku 3.8 je hamiltonovský podle Oreho věty 3.11, neboť má více než dva vrcholy a součet stupňů každých dvou nesousedních vrcholů je alespoň




Obrázek 3.8 Graf obsahuje pět vrcholů stupně 4, jeden vrchol stupně 5 a jeden vrchol stupně 3.

7. Všimněte si, že Diracovu větu 3.10 nemůžeme použít, neboť vrchol $u_7$ je stupně menšího než $|V(G)|/2$. Podle Oreho věty 3.11 určíme, že graf $G$ je hamiltonovský, protože součet stupňů každých dvou (nejen nesousedních) vrcholů $u$, $v$ je $\deg(u) + \deg(v) \geqq 7 = n$.

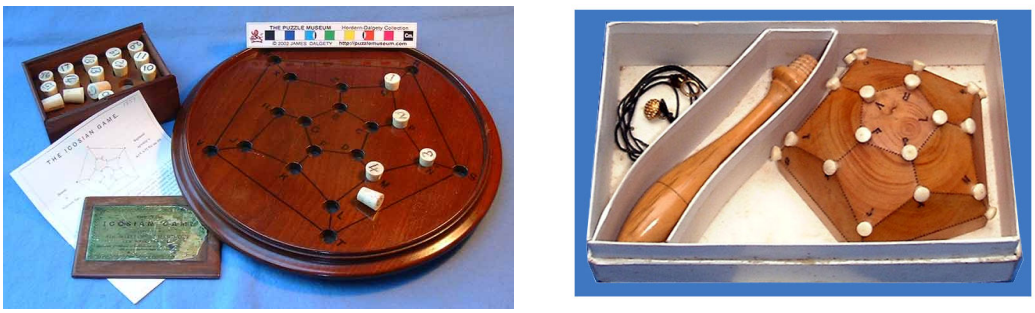
Hamiltonovský cyklus je například

$$u_1, u_2, u_3, u_6, u_5, u_4, u_7.$$

▲

## Pro zájemce:

Hamiltonovským grafům se říká „hamiltonovské“ podle irského matematika Williama Rowana Hamiltona, který v roce 1857 vymyslel a komerčně využil hru „Cesta kolem světa“. Jednalo se o nalezení takové cesty po hranách dvanáctistěnu, abychom každý vrchol navštívili jen jednou a vrátili se do původního vrcholu. Příklady hry jsou na Obrázku 3.9.



Obrázek 3.9 Různé varianty Hamiltonovy hry.

Další úlohou, kterou je možné přeformulovat na hledání hamiltonovského cyklu je klasická úloha jezdce na šachovnici. Máme za úkol koněm objet celou šachovnici tak, abychom každé políčko navštívili právě jednou a nakonec se vrátili na výchozí políčko. Sestavíme-li graf, jehož vrcholy odpovídají políčkům šachovnice a hranou spojíme každá dvě políčka, mezi kterými existuje regulérní tah jezdcem, dostaneme graf s 64 vrcholy, ve kterém máme za úkol najít hamiltonovský cyklus. Taková cesta jezdcem existuje. Dokážete ji najít?

## Pojmy k zapamatování

— hamiltonovský cyklus
— hamiltonovský graf

# Kapitola 4

# Vzdálenost a metrika v grafu

## Průvodce studiem


*V kapitole 2 jsme se zabývali otázkou, zda v grafu, který modeluje nějakou reálnou situaci, existuje cesta mezi dvěma vybranými vrcholy. V mnoha praktických aplikacích má smysl navíc zjistit, „jak daleko“ jsou vrcholy v grafu. Zde ukážeme, jak přirozeně „měřit“ vzdálenosti v grafu.*

*Jestliže například máme graf, který reprezentuje silniční síť, tak je velmi přirozené ptát se*

*„Jak daleko je z vrcholu (města) $u$ do vrcholu (města) $v$?“*

*nebo*

*„Jak dlouho trvá cesta z vrcholu (města) $u$ do vrcholu (města) $v$?“*

*Za délku cesty v grafu můžeme samozřejmě prohlásit počet hran této cesty. Při podrobnějším zkoumání nebudeme „vzdáleností“ rozumět pouhý* počet hran*, tedy například počet různých silnic, kterými pojedeme, nebo počet křižovatek, které projedeme. Pro určení vzdálenosti bude zásadní „délka hran“, neboli délka jednotlivých úseků mezi křižovatkami. Všimněte si, že informace o délkách úseků (hran) v grafu, tak jak jsme jej na straně 2 zavedli, zatím obsažena není.*

*Proto, abychom mohli v grafu měřit vzdálenosti, zavedeme pojem „ohodnocení hran grafu“, případně „ohodnocení grafu“. Ohodnocení bude zpravidla odpovídat nějaké fyzikální veličině, například délce v metrech, tloušťce, kapacitě, elektrickému odporu, barvě, a pod.*

*Budeme-li pracovat s ohodnoceným grafem, obvykle abstrahujeme od fyzikální podstaty příslušné veličiny a s ohodnocením budeme pracovat jako s číslem. Budeme například pracovat s hranami délek 4 a 12 a jejich navázáním dostaneme cestu nebo sled délky 16. Až budeme výsledky interpretovat v kontextu původní úlohy, neměli bychom zapomenout na příslušnou jednotku fyzikální veličiny. Délka 16 tak bude odpovídat například šestnácti kilometrům, v jiné interpretaci pak třeba 16 ohmům. V jiné úloze budou například hrany s ohodnocením 4 a 12, které jsou incidentní se stejným vrcholem, odpovídat*

*celkové kapacitě potrubí 16 kubických metrů za sekundu. A třeba v další úloze nebude mít smysl ohodnocení navazujících hran sčítat, neboť se bude jednat o barvu případně o šířku na sebe navazujících cest.*

*Poznamenejme ještě, že při ohodnocování hran obvykle vystačíme s celými čísly. Je-li například vzdálenost udávána v kilometrech včetně desetinné části odpovídající stovkám nebo desítkám metrů, tak je vhodné pracovat se vzdáleností vyjádřenou v metrech a všechny délky tak budou celá čísla.*

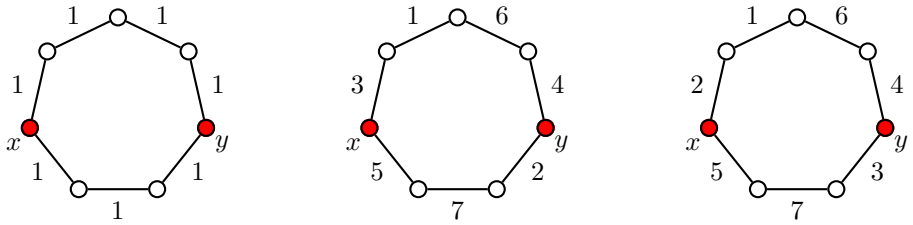
### Cíle

Po prostudování této kapitoly budete schopni:

- určit vzdálenost (délku nejkratší cesty) mezi dvěma vrcholy v ohodnoceném i neohodnoceném grafu,
- určit vzdálenosti z jednoho vrcholu do všech ostatních vrcholů,
- určit vzdálenost mezi každou dvojicí vrcholů daného grafu,
- správně určit meze použitelnosti jednotlivých algoritmů.

## 4.1 Vzdálenost v grafu

Dříve než podáme formální definici ohodnoceného grafu a dříve než nadefinujeme pojem *délka ohodnocené cesty v grafu*, tak vystačíme s intuitivní představou. Pro názornost srovnejme vzdálenost vrcholů $x$ a $y$ pro tři různé grafy na Obrázku 4.1. Předpokládejme, že čísla přiřazená hranám určují délku každého úseku. Zatímco v grafu na obrázku vlevo je vzdálenost mezi vrcholy $x$ a $y$ rovna 3 (cesta „spodem“ vede přes tři hrany délky 1), tak v prostředním grafu najdeme dvě cesty délky 14 (obě „horní“ i „spodní“ cesta mají stejnou délku). Konečně, v grafu na obrázku vpravo je nejkratší cesta z vrcholu $x$ do vrcholu $y$ délky 13 (cesta „horem“ přes celkem čtyři hrany délek 2, 1, 6 a 4).




Obrázek 4.1 Různé vzdálenosti v grafu v závislosti na délkách hran.

Jestliže jsou délky všech hran stejné, tak délka cesty je samozřejmě přímo úměrná počtu hran cesty. Z Obrázku 4.1 je jasné, že délka cesty nemusí nutně odpovídat pouze počtu hran cesty.

Nyní zavedeme pojem vzdálenosti v neohodnoceném grafu. Připomeňme, že na straně 29 jsme definovali délku sledu jako počet hran sledu.

**Definice 4.1.** *Vzdálenost* $\mathrm{dist}_G(u,v)$ mezi vrcholy $u$ a $v$ grafu $G$ je dána délkou nejkratšího sledu mezi $u$ a $v$ v $G$ (tj. sledu s nejmenším počtem hran). Pokud sled mezi $u$, $v$ neexistuje, položíme vzdálenost $\mathrm{dist}_G(u,v) = \infty$.

Všimněte si, že:

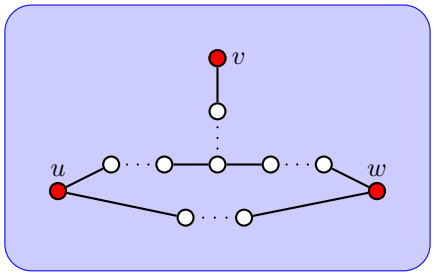
- nejkratší sled (ze všech sledů obsahuje nejméně hran) je *vždy cestou*,
- v neorientovaných grafech platí $\mathrm{dist}_G(u,v) = \mathrm{dist}_G(v,u)$,
- vzdálenost vrcholu od sebe samotného je 0, tj. platí $\mathrm{dist}_G(u,u) = 0$,
- pokud $\mathrm{dist}_G(u,v) = 1$, tak hrana $uv$ patří do grafu $G$.

Uvědomte si, že zavedeme-li v orientovaných grafech vzdálenost z vrcholu $u$ do vrcholu $v$ jako délku nejkratšího orientovaného $(u,v)$-sledu, tak v orientovaném grafu může být vzdálenost z $u$ do $v$ odlišná od vzdálenosti z $v$ do $u$.

**Lemma 4.2.** *Vzdálenost mezi vrcholy v daném grafu $G$ splňuje tzv.* trojúhelníkovou nerovnost*:*

$$\forall u,v,w \in V(G)\colon\ \mathrm{dist}_G(u,w) \leqq \mathrm{dist}_G(u,v) + \mathrm{dist}_G(v,w).$$

*Důkaz.* Nerovnost ihned plyne ze zřejmého pozorování, že na sled délky $\mathrm{dist}_G(u,v)$ mezi vrcholy $u$, $v$ lze navázat sled délky $\mathrm{dist}_G(v,w)$ mezi $v$, $w$, čímž dostaneme sled délky $\mathrm{dist}_G(u,v) + \mathrm{dist}_G(v,w)$ mezi $u$, $w$. Proto nemůže být $\mathrm{dist}_G(u,w) > \mathrm{dist}_G(u,v) + \mathrm{dist}_G(v,w)$. Může však existovat kratší sled mezi $u,w$, než je sled, který vznikne navázáním sledů délky mezi $u$ a $v$ a mezi $v$ a $w$, proto nemusí obecně nastat rovnost a platí $\mathrm{dist}_G(u,w) \leqq \mathrm{dist}_G(u,v) + \mathrm{dist}_G(v,w)$. Schématický příklad je na Obrázku 4.2. □




Obrázek 4.2 Sledy mezi $u$ a $v$, mezi $v$ a $w$; kratší sled mezi $u$ a $w$.

## Pro zájemce:

Metrika v matematice je zobecněním vzdálenosti, jak ji známe z běžného života. Uvědomte si, že vzdálenost je pro každou dvojici bodů (míst nebo měst) číslo, které umíme této dvojici jednoznačně přiřadit tak, aby byly splněny jisté „přirozené vlastnosti“. *Metrika* je taková

funkce $\rho$, která přiřazuje číslo každé dvojici prvků z nějaké dané množiny $A$ a která má tyto „přirozené vlastnosti“ popsané v následující definici.

**Definice 4.3.** *Metrika* $\rho$ na množině $A$ je zobrazení $\rho : A \times A \to \mathbb{R}$ takové, že $\forall x, y, z \in A$ platí

1. $\rho(x, y) \geqq 0$, přičemž $\rho(x, y) = 0$ právě tehdy, když $x = y$,
2. $\rho(x, y) = \rho(y, x)$,
3. $\rho(x, y) + \rho(y, z) \geqq \rho(x, z)$.

S různými metrikami se setkáváme i v jiných matematických disciplínách. V analýze se pracuje například s metrikou funkce sestavené pomocí absolutní hodnoty rozdílu dvou čísel, v geometrii může být metrikou eukleidovská vzdálenost dvou bodů v rovině nebo prostoru a v algebře lze získat různé metriky s využitím různých norem. Množina prvků spolu s metrikou $\rho$ se nazývá *metrický prostor*.

Vzdálenost v grafu zavedená v Definici 4.1 je metrika na množině vrcholů daného souvislého grafu. Všimněte si, že pro nesouvislé grafy musíme pojem metriky oproti Definici 4.3 ještě zobecnit, neboť se jedná o zobrazení $\rho : A \times A \to \mathbb{R}^*$ (vzdálenost v nesouvislých grafech může být i $\infty$). Přitom předpokládáme, že relace uspořádání i operace sčítání pracují se symbolem $\infty$ „přirozeně“, tedy například, že $x \leqq \infty$ pro každé $x \in \mathbb{R}^*$ a že $x + \infty = \infty$ pro každé $x \in \mathbb{R}^*$ a $x \geqq 0$. V dalším textu budeme pro nesouvislé grafy pracovat s metrikou, která připouští nevlastní hodnotou nekonečno.

Už dříve jsme zmínili, že budeme určovat vzdálenost mezi dvěma vrcholy grafu proto, abychom uměli určit odpovídající „vzdálenost“ v nějaké praktické úloze. Jak vzdálenost určíme? Uvědomte si, že kdybychom chtěli prozkoumat všechny možné cesty mezi dvěma vrcholy by bylo nejen časově náročné, ale také algoritmicky obtížné. Například v kompletním grafu umíme všechny cesty mezi danou dvojicí vrcholů najít snadno (Příklad 4.4). Na druhou stranu v obecném grafu je mezi danou dvojicí vrcholů cest méně, ale určit jejich počet může být náročnější než u kompletního grafu na stejném počtu vrcholů. Zkuste si rozmyslet, jak byste určili počet různých cest mezi vrcholy $u_1$ a $u_2$ na Obrázku 3.8.

**Příklad 4.4.** Určete počet všech cest mezi danou dvojicí vrcholů $u$ a $v$ v kompletním grafu $K_n$. Kolik je takových cest mezi dvojicí vrcholů v $K_7$?

*Řešení.* Mějme kompletní graf a v něm dva vrcholy $u$, $v$. Budeme určovat počet cest, které začínají ve vrcholu $u$ a končí ve vrcholu $v$. Rozlišíme dva případy:

1. Jestliže $u = v$, tak existuje jediná cesta z $u$ do $v$ – a sice triviální cesta.
2. Jestliže $u \neq v$, tak dále rozlišíme ještě možnosti podle toho, kolik dalších vrcholů se nachází na té které cestě mezi vrcholy $u$ a $v$.
   (a) V grafu existuje jediná taková cesta z $u$ do $v$, která neobsahuje žádný vnitřní vrchol, a sice cesta s jedinou hranou $uv$.

(b) Jestliže na cestě z vrcholu $u$ do $v$ je $k \geqq 1$ vnitřních vrcholů, tak takových různých cest existuje celkem $V(n-2,k) = \frac{(n-2)!}{(n-2-k)!}$, protože za vnitřní vrcholy cesty můžeme vybrat libovolných $k$ vrcholů a protože při sestavení cesty rozlišíme jejich pořadí. Počet cest s $k$ vnitřními vrcholy se proto rovná počtu variací bez opakování (podrobně o určování počtu výběrů se dočtete v [6]).

Počet vnitřních vrcholů může tedy nabývat libovolné hodnoty $k \in \{0, 1, \ldots, n-2\}$. Celkem je počet hledaných cest dán vztahem

$$1 + \sum_{k=1}^{n-2} \frac{(n-2)!}{(n-2-k)!} = \sum_{k=0}^{n-2} \frac{(n-2)!}{(n-2-k)!}. \tag{4.1}$$

Pro $n = 7$ stačí dosadit do obecného vztahu 4.1. Počet všech cest mezi dvěma vybranými vrcholy v grafu $K_7$ je roven $\sum_{k=0}^{5} \frac{5!}{(5-k)!} = 1 + 5 + 20 + 60 + 120 + 120 = 326$. ▲

Naštěstí se ukazuje, že pro určení vzdálenosti mezi dvěma vrcholy nemusíme prozkoumávat všechny cesty mezi danou dvojicí vrcholů. V další části této kapitoly se budeme věnovat algoritmům pro určení vzdálenosti v neohodnoceném grafu.

### Pojmy k zapamatování

— vzdálenost v grafu
— trojúhelníková nerovnost
— metrika na množině $A$ a metrika na množině vrcholů daného grafu

## 4.2 Určení vzdáleností neboli výpočet metriky

### Průvodce studiem


*V této sekci se budeme věnovat algoritmům pro určení vzdálenosti mezi vrcholy grafu. Ukážeme dva algoritmy, jeden pro určení vzdálenosti mezi každými dvěma vrcholy a druhý, rychlejší, pro určení vzdáleností všech vrcholů od jednoho pevně zvoleného vrcholu. Oba algoritmy mají polynomiální složitost vzhledem k počtu vrcholů daného grafu, budeme proto umět určit vzdálenost dvou vrcholů rychle i v poměrně velkém grafu.*

### Cíle

Po prostudování této sekce budete schopni:

- určit vzdálenost (délku nejkratší cesty) mezi dvěma vrcholy v neohodnoceném grafu,

- určit vzdálenost mezi každou dvojicí vrcholů daného grafu,
- rozhodnout, který z uvedených algoritmů je pro danou úlohu nejvhodnější.

Nejprve ukážeme, jak jednoduchá modifikace algoritmu pro prohledávání grafu (Algoritmus 2.1) umožní určit vzdálenosti z jednoho pevně zvoleného vrcholu do všech ostatních vrcholů. Předpokládejme, že máme implementaci Algoritmu 2.1 s postupem do šířky, tj. úschovna je implementována jako fronta. Místo označení úschovny „U“ jako v Algoritmu 2.1 budeme proto v tomto algoritmu úschovnu označovat „F“.

Nyní stačí, když v průběhu algoritmu každému nově objevenému vrcholu přiřadíme vzdálenost, která je o jedničku větší, než vzdálenost právě zpracovávaného vrcholu. Pro uchování informace o vzdálenosti každého vrcholu použijeme jednorozměrné pole `vzdal[]`.

Pseudokód algoritmu může vypadat například takto:

**Algoritmus 4.1 (Vzdálenosti z daného vrcholu).**

```
// na vstupu je graf G
vstup < graf G;
stav(všechny vrcholy a hrany G) = iniciační;
fronta F = libovolně zvolený vrchol u grafu G;
stav(u) = nalezený;
vzdal(u) = 0;                          // vzdálenost u

// zpracování vybrané komponenty G
while (F je neprázdná) {
   vyber vrchol v a odeber jej z fronty F: F = F - v;
   for (hrany e vycházející z v) {    // pro všechny hrany
      w = druhý vrchol hrany e = vw;  // známe sousedy?
      if (stav(w) == iniciační) {
         stav(w) = nalezený;
         přidej vrchol w do fronty: F = F + w;
         vzdal[w] = vzdal[v]+1;       // vzdálenost w
      }
   }
   stav(v) = zpracovaný;
}

// vrcholy z dalších komponent jsou nedosažitelné!
while (v grafu G je nezpracovaný vrchol w) {
   vzdal[w] = MAX_INT;              // nekonečno
   stav(w) = zpracovaný;
}
```

Po skončení běhu algoritmu bude každá položka pole `vzdal[]` obsahovat číselnou hodnotu, která udává vzdálenost příslušného vrcholu od výchozího vrcholu $u$. Dokonce i vrcholy nedosažitelné z výchozího vrcholu budou mít správně určenou vzdálenost, přičemž $\infty$ je v algoritmu implementováno pomocí systémové konstanty `MAX_INT`.

**Poznámka 4.5.** Všimněte si, že po skončení algoritmu sice bude známa vzdálenost každého vrcholu od výchozího vrcholu $u$, avšak nejkratší cestu nebudeme umět zrekonstruovat. Pokud bychom chtěli navíc zjistit, jak nejkratší cesty vypadají, bude potřeba v průběhu algoritmu získat další informace. Elegantním řešením je pro každý vrchol $w$ uložit informaci o *předchozím* vrcholu na nějaké nejkratší cestě z vrcholu $u$ do vrcholu $w$. K tomu stačí jednorozměrné pole `pre[]`. Do algoritmu stačí za řádek, kde je určena vzdálenost vrcholu $w$, přidat následující řádek.

```
pre[w] = v;                    // předchůdce w
```

Na konci běhu algoritmu bude pro každý vrchol, který se nachází v komponentě spolu s výchozím vrcholem $u$, uložena v poli `pre[]` informace, kudy nejkratší cesta z vrcholu $u$ do vrcholu $w$ vede. Tuto cestu můžeme zrekonstruovat:

- poslední vrchol takové cesty je vrchol $w$,
- předposlední vrchol cesty je vrchol $pre[w]$,
- předpředposlední vrchol cesty je vrchol $pre[pre[w]]$,
- ...
- první (tj. výchozí) vrchol cesty je vrchol $pre[\ldots pre[pre[w]]]$, což je vrchol $u$.

### Složitost Algoritmu 4.1

Uvedený algoritmus je poměrně rychlý. Počet kroků závisí na počtu vrcholů a hran daného grafu. Při vhodné implementaci je počet kroků algoritmu přímo úměrný počtu vrcholů (pro řídké grafy – s malým počtem hran) nebo počtu hran (pro husté grafy). Přesněji můžeme říci, že složitost algoritmu je $O(n+m)$, kde $n$ udává počet vrcholů a $m$ je počet hran daného grafu. I v kompletním grafu, který má $m = \binom{n}{2} = n(n-1)/2$ hran, je proto složitost algoritmu řádově $O(n^2)$.

### Důkaz správnosti Algoritmu 4.1

Ačkoliv to nebylo výslovně řečeno, při popisu Algoritmu 4.1 jsme využili předpoklad, že vzdálenější vrcholy budou zpracovány později, než vrcholy blíže k výchozímu vrcholu $u$. Toto pozorování pečlivě zformulujeme (Lemma 4.6), dokážeme jeho platnost, a pak jej použijeme v důkazu, že Algoritmus 4.1 funguje správně, tj. že po skončení běhu algoritmu dostaneme vzdálenosti z výchozího vrcholu do všech ostatních vrcholů (i nejkratší cesty, pokud existují).

**Lemma 4.6.** *Nechť $u, v, w$ jsou takové vrcholy souvislého grafu $G$, že* $\mathrm{dist}_G(u,v) < \mathrm{dist}_G(u,w)$. *Pak při procházení grafu $G$ postupem* do šířky *z vrcholu $u$ je vrchol $v$ nalezen dříve než vrchol $w$.*

*Důkaz.* Postupujeme silnou indukcí vzhledem ke vzdálenosti $\mathrm{dist}_G(u,v)$.
*Základ indukce:* Jestliže $\mathrm{dist}_G(u,v) = 0$, tak $u = v$ a tvrzení je zřejmé – vrchol $u$ dostane na začátku algoritmu stav „nalezen“ jako první.
*Indukční krok:* Předpokládejme, že pro vrcholy ve vzdálenosti *menší* než nějaká pevně zvolená hodnota $d > 0$, tvrzení platí. Mějme vrchol $v$, pro který platí $\mathrm{dist}_G(u,v) = d$ a označme $v'$ předposlední vrchol (sousední k vrcholu $v$) na nejkratší cestě z vrcholu $u$ do vrcholu $v$. Víme, že $\mathrm{dist}_G(u,v') = d - 1$. Stejně tak označme $w'$ předposlední vrchol na nejkratší cestě z vrcholu $u$ do vrcholu $w$. Protože $\mathrm{dist}_G(u,v) < \mathrm{dist}_G(u,w)$, je také $\mathrm{dist}_G(u,v) - 1 < \mathrm{dist}_G(u,w) - 1$ tedy $\mathrm{dist}_G(u,v') < \mathrm{dist}_G(u,w')$.

To podle indukčního předpokladu znamená, že vrchol $v'$ bude v prohledávání do šířky nalezen dříve než vrchol $w'$, protože vrchol $v'$ se nachází ve vzdálenosti menší než $d$ od výchozího vrcholu $u$. To ale současně znamená, že vrchol $v'$ se dostal do fronty $F$ dříve než vrchol $w'$, a proto sousedé vrcholu $v'$ (mezi nimi i vrchol $v$) budou při prohledávání nalezeni dříve než sousedé vrcholu $w'$. To je dokazované tvrzení a podle principu silnou matematické indukce platí pro vrcholy v libovolné konečné vzdálenosti.

Na závěr algoritmu je vzdálenost $\infty$ přiřazena všem zbývajícím neprozkoumaným vrcholům (vrcholům nedosažitelných z výchozího vrcholu $u$), což je v souladu s dokazovaným tvrzením a důkaz končí. □

Nyní můžeme ukázat, že Algoritmus 4.1 pracuje správně.

**Věta 4.7.** *Algoritmus 4.1 určí vzdálenosti z nějakého výchozího vrcholu $u$ do všech ostatních vrcholů.*

*Důkaz.* Tvrzení ukážeme opět indukcí vzhledem ke vzdálenosti vrcholu od výchozího vrcholu $u$..
*Základ indukce:* Podle Algoritmu 4.1 je na začátku přiřazena vrcholu $u$ vzdálenost `dist[u]=0` a dále se vzdálenost do výchozího vrcholu $u$ již nemění, neboť vzdálenosti se určují pouze pro vrcholy v iniciačním stavu. Proto i po skončení algoritmu bude `dist[u]=0`, což odpovídá správné vzdálenosti $\mathrm{dist}_G(u,u) = 0$.
*Indukční krok:* Mějme nějaký vrchol $w$ ve vzdálenosti $d > 0$ od vrcholu $u$ a předpokládejme, že všechny vrcholy ve vzdálenosti menší než $d$ mají vzdálenost určenu správně. Označme $w'$ předposlední vrchol na nejkratší cestě z vrcholu $u$ do vrcholu $w$. Podle Lemmatu 4.6 budou všechny vrcholy, které jsou blíže k výchozímu vrcholu $u$ zpracovány dříve než $w$, tedy i vrchol $w'$ bude zpracován dříve. Nejpozději při zpracování vrcholu $w'$ bude nastavena hodnota položky `dist[w]=dist[w']+1`, neboť

všechny vrcholy s vzdáleností od $u$ menší než vrchol $w$ mají podle indukčního předpokladu vzdálenost určenu správně a hodnota `dist[w]` se nastaví pouze při pracování nějakého vrcholu sousedního s $w$.

A konečně všem vrcholům nedosažitelným z výchozího vrcholu $u$ bude na závěr přiřazena vzdálenost $\infty$. Tím důkaz končí. □

**Poznámka 4.8.** Dobře si rozmyslete, že při použití jiného algoritmu než s postupem do šířky (například s postupem do hloubky) by tvrzení Věty 4.6 nebylo pravdivé. Při prohledávání grafu postupem do hloubky *není* pravda, že vrchol $v$, který je do úschovny vložen dříve než vrchol $w$, bude zpracován dříve než vrchol $w$. Naopak dokonce víme, že dříve přidaný vrchol bude zpracován později. Vzdálenosti od výchozího vrcholu $u$ by proto takový algoritmus neurčil správně.

### Výpočet metriky skládáním cest

Jak jsme uvedli na straně 64, je metrika funkce, která každé dvojici prvků dané množiny přiřadí číslo tak, aby byly splněny vlastnosti v Definici 4.3.

**Definice 4.9.** Mějme dán graf $G$. *Metrikou grafu* $G$ nazýváme funkci, která každé dvojici vrcholů $u, v \in V(G)$ přiřadí jejich vzdálenost $\mathrm{dist}_G(u, v)$.

V dalším budeme tuto metriku (přesněji funkční hodnoty metriky) ukládat do dvourozměrného pole `d[][]`, jehož každá položka `d[i][j]` bude obsahovat hodnotu vzdálenosti $\mathrm{dist}(i, j)$ mezi vrcholy $i$ a $j$ v daném grafu. (Pro jednoduchost předpokládáme, že vrcholy jsou čísla $0, 1, \ldots, |V(G)| - 1$.) Tomuto dvourozměrnému poli budeme říkat „metrika $d$“, místo abychom dlouze opisovali „metrika, jejíž funkční hodnoty jsou uloženy v poli `d[][]`.“

Pro sestavení metriky $d$ bychom mohli opakovaně použít Algoritmus 4.1. Stačí pro každý vrchol $u \in V(G)$ určit pomocí Algoritmu 4.1 vzdálenosti do všech ostatních vrcholů a určit tak všechny hodnoty v $u$-tém řádku metriky $d$. Všimněte si, že pro neorientované grafy tak současně určíme všechny hodnoty v $u$-tém sloupci metriky $d$.

Existuje však jednodušší algoritmus:

**Algoritmus 4.2 (Floydův algoritmus – sestavení metriky).**

```
vstup: matice sousednosti G[][] grafu na N vrcholech, kde
       G[i][j]=1 pro existující hranu mezi i,j a
       G[i][j]=0 jinak;

// inicializace (hodnotu MAX_INT/2 považujeme za "nekonečno")
for (i=0; i<N; i++)
   for (j=0; j<N; j++)
      d[i][j] = (i==j ? 0 : (G[i][j] ? 1 : MAX_INT/2));

// cyklus pro všechny vrcholy
for (t=0; t<N; t++)
   // probereme všechny dvojice vrcholů
   for (i=0; i<N; i++)
      for (j=0; j<N; j++)
         // vede přes vrchol t kratší cesta?
         d[i][j] = min(d[i][j], d[i][t]+d[t][j]);
```

Rozeberme podrobně, jak Algoritmus 4.2 funguje. Vrcholy daného grafu $G$ máme označené jako $0, 1, 2, \ldots, N-1$ (čísla můžeme chápat například jako indexy vrcholů).

Na začátku algoritmu položíme

- `d[i][j]=0` jestliže vrcholy $i = j$,
- `d[i][j]=1` jestliže vrcholy $i$ a $j$ jsou sousední (na straně 75 popíšeme modifikaci algoritmu, kde budeme prvku `d[i][j]` přiřazovat délku hrany $ij$),
- nebo `d[i][j]` položíme rovno nekonečno (implementováno jako `MAX_INT/2`), pokud hrana $ij$ v grafu není.

Nyní postupně určíme metriku grafu $G$. Po každém kroku $t \geqq 0$ algoritmu bude hodnota `d[i][j]` udávat délku takové nejkratší cesty mezi vrcholy $i$ a $j$, která vede pouze přes „povolené“ vrcholy z množiny $\{0, 1, 2, \ldots, t\}$. V průběhu každého kroku případně upravíme délku zatím nejkratší nalezené cesty pro každou dvojici vrcholů $i$, $j$. Rozlišíme dvě možnosti:

- najdeme kratší cestu přes nově povolený vrchol $t$; potom nahradíme hodnotu `d[i][j]` menší vzdáleností `d[i][t] + d[t][j]` nebo
- přidání vrcholu $t$ do seznamu „povolených“ vrcholů nepomůže najít cestu kratší než byla `d[i][j]` v předchozím kroku; potom hodnotu `d[i][j]` ponecháme beze změny.

Po nejvýše $N$ iteracích cyklu řízeného proměnnou $t$ budou prvky pole `d[][]` obsahovat vzdálenosti mezi odpovídajícími dvojicemi vrcholů. Všimněte si, že v každém průběhu vnějšího cyklu řízeného proměnnou $t$ můžeme obecně upravit vzdálenost

(vztaženou pouze na cesty přes vrcholy $0, 1, \ldots, t$) mezi každou dvojicí vrcholů. Proto každá hodnota pole `d[][]` se v průběhu algoritmu může mnohokrát změnit. Hodnoty v poli `d[][]` se v průběhu algoritmu nemohou zvyšovat.

Pokud je graf $G$ nesouvislý, tak i po skončení algoritmu bude v každém prvku pole `d[][]`, které odpovídá vzájemně nedosažitelným vrcholům, uložena hodnota `MAX_INT/2`. Neměli bychom však pracovat s hodnotou `MAX_INT`, aby na některých systémech nedošlo přičtením k „přetečení“ hodnoty odpovídající nekonečnu.

Rozmyslete si, že obecně neumíme v průběhu algoritmu poznat, zda algoritmus bude hodnoty v poli `d[][]` měnit, nebo zda už všechny hodnoty v poli `d[][]` odpovídají vzdálenostem. Není těžké vymyslet takový graf, aby se změnila hodnota jediného prvku v poli `d[][]` a to až v posledním kroku algoritmu.

### Složitost Algoritmu 4.2

Všimněte si, že v každém průběhu vnějšího cyklu řízeného proměnnou $t$ pracujeme s každým prvkem pole `d[][]`. Toto vyžaduje řádově $O(n^3)$ strojových operací, kde $n = N$ odpovídá počtu vrcholů daného grafu. Stejně tak pracujeme s každým prvkem pole `d[][]` během inicializace, což si vyžádá řádově $O(n^2)$ operací. Složitost celého algoritmu je proto $O(n^3)$.

### Porovnání Algoritmů 4.1 a 4.2

Srovnejme nyní oba algoritmy popsané v této sekci.

Zatímco Algoritmus 4.2 má velmi jednoduchou implementaci a najde vzdálenosti mezi každou dvojicí vrcholů, tak Algoritmus 4.1 má o trochu složitější implementaci a dává pouze vzdálenosti do všech vrcholů z jednoho pevně zvoleného vrcholu.

Na druhou stranu má Algoritmus 4.2 řádově vyšší složitost $O(n^3)$ oproti složitosti $O(n^2)$ Algoritmu 4.1. A pokud nás zajímá vzdálenost mezi jednou pevně zvolenou dvojicí vrcholů, tak Algoritmus 4.2 nemůžeme ukončit předčasně. Abychom měli jistotu, že máme určenu vzdálenost mezi danou dvojicí vrcholů, *musí* se totiž určit vzdálenost mezi každými dvěma vrcholy daného grafu.

## Pojmy k zapamatování

— algoritmy výpočtu metriky v grafu

## 4.3 Vzdálenost v ohodnocených grafech

## Průvodce studiem


*Už v úvodu Kapitoly 4 jsme zmínili význam grafu, ve kterém jsou hranám přiřazena čísla, přičemž tato čísla odpovídají délkám, tloušťkám, kapacitám, případně barvám v nějaké praktické úloze. V této sekci nejprve zavedeme pojem ohodnocení grafu formálně a pak*

*vyslovíme několik základních tvrzení. Dále ukážeme, proč se v dalším textu omezíme na grafy, které mají hrany ohodnocené pouze kladnými čísly.*

## Cíle

Po prostudování této sekce budete schopni:

- vysvětlit význam ohodnocení grafu,
- popsat některé praktické úlohy pomocí ohodnoceného grafu,
- vysvětlit, proč se omezíme na kladně ohodnocené grafy.

**Definice 4.10.** *Ohodnocení* grafu $G$ je funkce $w : E(G) \to \mathbb{R}$, která každé hraně $e \in E(G)$ přiřadí reálné číslo $w(e)$, kterému říkáme *váha hrany* (značení $w$ pochází z anglického „weight“). *Ohodnocený graf* je graf $G$ spolu s ohodnocením hran reálnými čísly.
*Kladně ohodnocený* (říkáme také *vážený*) *graf* $G$ má takové ohodnocení $w$, že pro každou hranu $e \in E(G)$ je její váha $w(e)$ kladná.

Podle definice mohou čísla přiřazená hranám být libovolná (konečná) reálná čísla. V praxi se však většinou setkáváme s grafy, v nichž váhy všech hran jsou kladné. Když budeme chtít zdůraznit, že hrany grafu jsou ohodnoceny kladnými čísly, budeme hovořit o „kladně váženém“ případně jen o „váženém“ grafu. Dále se z praktických důvodů často omezíme na ohodnocení přirozenými čísly (případně nezápornými celými čísly – včetně nuly), neboť váhy obvykle odpovídají fyzikálním veličinám a my můžeme zvolit dostatečně malou jednotku pro jejich vyjádření tak, aby rozdíly v desetinkách už neměly žádný praktický význam. Navíc se při početních operacích s přirozenými čísly vyhneme některým zaokrouhlovacím chybám, které mohou v různých implementacích způsobit, že by se výsledky algoritmů dosažené na různých systémech mohly lišit.

**Poznámka 4.11.** Striktně vzato, ohodnocený graf je *dvojice* $(G, w)$, kde $G$ je graf a $w$ je jeho ohodnocení. Dále v textu si však dovolíme psát jen o ohodnoceném grafu $G$, pokud bude z kontextu jasné, s jakým ohodnocením pracujeme.

Nyní zavedeme vzdálenost v ohodnoceném (tedy i ve váženém) grafu. Vážená vzdálenost nebude pouhý počet hran mezi danou dvojicí vrcholů, ale bude se jednat o součet vah jednotlivých hran.

**Definice 4.12.** Mějme ohodnocený graf $G$ s ohodnocením $w$. *Délkou ohodnoceného sledu* $S = v_0, e_1, v_1, e_2, v_2, \ldots, e_n, v_n$ v $G$ rozumíme součet ohodnocení všech jeho hran. Každá hrana se počítá tolikrát, kolikrát se ve sledu $S$ vyskytuje, a proto

$$d_G^w(S) = w(e_1) + w(e_2) + \cdots + w(e_n).$$

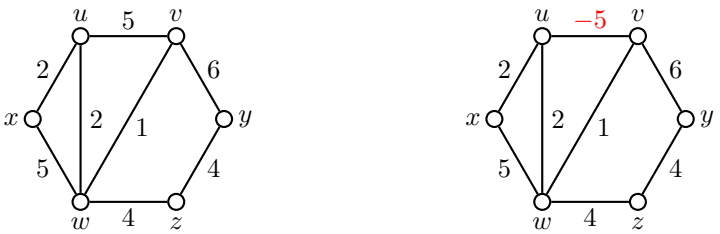
*(Vážená) vzdáleností* mezi dvěma navzájem dosažitelnými vrcholy $u$, $v$ ve váženém nebo ohodnoceném grafu $(G, w)$ rozumíme číslo

$$\mathrm{dist}_G^w(u, v) = \min\{d_G^w(S), \text{kde } S \text{ je } \textbf{cesta} \text{ s koncovými vrcholy } u, v\}.$$

Jestliže vrcholy $u$ a $v$ jsou nedosažitelné, klademe $\mathrm{dist}_G^w(u, v) = \infty$.

**Poznámka 4.13 (O existenci nejkratšího sledu a cesty).** Musíme upozornit na jeden podstatný rozdíl mezi určováním vzdáleností v neohodnocených a ohodnocených grafech. Intuitivně čekáme, že nejkratší sled mezi dvěma vrcholy je takový sled $S$, který má ze všech možných sledů mezi danými dvěma vrcholy nejmenší délku. Všimněte si, že mezi dvěma vrcholy v grafu se záporně ohodnocenými hranami nemusí nejkratší sled vůbec existovat! Například v grafech na Obrázku 4.3 budeme hledat nejkratší sled a cestu z vrcholu $x$ do vrcholu $y$. V kladně váženém grafu na obrázku 4.3 vlevo najdeme po drobném rozmyšlení nejkratší sled $S_1 = x, u, w, v, y$ délky 11. Tento sled je současně i nejkratší cestou z vrcholu $x$ do vrcholu $y$.

Jaký je však nejkratší sled z vrcholu $x$ do vrcholu $y$ v grafu na obrázku 4.3 vpravo? Sled $S_1 = x, u, w, v, y$ není nejkratší, neboť sled $S_2 = x, u, v, y$ má délku 3. Ale sled $S_3 = x, u, v, w, u, v, y$ má délku rovnu dokonce jen 1. Sled $S_3 = x, u, v, w, u, v, w, u, v, y$ má délku dokonce $-11$, atd. A například také sled $S_4 = x, u, v, u, v, y$ má délku $-7$. Je zřejmé, že ke každému sledu s libovolně malou délkou najdeme sled s ještě menší délkou. Proto nejkratší sled z vrcholu $x$ do vrcholu $y$ neexistuje.




Obrázek 4.3 Nejkratší sled a cesta mezi vrcholy $x$ a $y$.

Naproti tomu nejkratší cesta z vrcholu $x$ do vrcholu $y$ existuje, je to cesta $S_2 = x, u, v, y$ a má délku 3. Všimněte si, že vzdálenost vrcholů $x$ a $y$ podle Definice 4.12 je rovna 3, neboť je definována jako délka „nejkratší *cesty*“ a nikoliv délka „nejkratšího sledu“.

Problém s nejkratším sledem souvisí s tím, že ve sledu se mohou záporně ohodnocené hrany libovolně mnohokrát opakovat a mohou tak „pokazit“ existenci minimální délky sledu. V algoritmu, který bude hledat vzdálenosti nebo nejkratší cesty v obecném ohodnoceném grafu, bychom museli tento jev vzít v úvahu. To je důvod, proč se při popisu algoritmů omezíme na *kladně* vážené grafy. Určování vzdáleností vrcholů v grafu se záporně ohodnocenými hranami je možné, ale přesahuje rámec tohoto textu.

Dá se ukázat, že vážená vzdálenost je metrikou v grafu (podobně jako pro neohodnocené grafy). Platí následující pozorování.

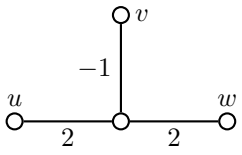
**Lemma 4.14.** *Vážená vzdálenost mezi vrcholy $v$ daném váženém grafu $G$ splňuje* trojúhelníkovou nerovnost*:*

$$\forall u, v, w \in V(G)\colon\ \mathrm{dist}^w_G(u, w) \leqq \mathrm{dist}^w_G(u, v) + \mathrm{dist}^w_G(v, w).$$

**Příklad 4.15.** Určete, zda Lemma 4.14 platí obecně i v ohodnocených grafech se záporně ohodnocenými hranami.

*Řešení.* Zkuste si řešení rozmyslet sami dříve než si přečtete následující zdůvodnění.

Snadno najdeme příklad grafu a trojice jeho vrcholů $u$, $v$, $w$, které trojúhelníkovou rovnost nesplňují. Jeden příklad je na Obrázku 4.4. Zatímco vzdálenost $\mathrm{dist}^w_G(u, v) = 1$, vzdálenost $\mathrm{dist}^w_G(v, w) = 1$, tak součet těchto vzdáleností *není větší nebo roven* vzdálenosti $\mathrm{dist}^w_G(u, w) = 4$. ▲




Obrázek 4.4 Příklad grafu se záporně ohodnocenou hranou, ve kterém neplatí trojúhelníková nerovnost.

### Algoritmus 4.1 pro ohodnocené grafy

Pokud bychom pro nalezení nejkratších vzdáleností z vrcholu $u$ do ostatních vrcholů daného grafu $G$ chtěli použít Algoritmus 4.1, tak je dobré si uvědomit, že by nestačilo v Algoritmu 4.1 upravit řádek, kde se nastavuje vzdálenost vrcholu. Kdybychom místo

```
vzdal[w] = vzdal[v]+1;        // vzdálenost w
```

použili

```
vzdal[w] = vzdal[v]+w(e);     // vzdálenost w
```

algoritmus by nedával správné výsledky. Vždyť nejkratší cesta z výchozího vrcholu nemusí nutně vést přes *dříve objevený*, ale spíše přes „*bližší vrchol*“ nebo po „*kratší hraně*“ (s menším ohodnocením).

Jestliže jsou váhy daného grafu $G$ tvořeny malými přirozenými čísly ($G$ je kladně vážený), tak místo abychom upravili algoritmus, budeme upravovat graf $G$. Každou hranu $e$ nahradíme cestou: místo jedné hrany přidáme tolik „nových“ hran a vrcholů, aby počet nových hran (a tedy délka cesty v neohodnoceném grafu) byl vždy roven váze $w(e)$ původní hrany $e$. Dostaneme sice obvykle mnohem větší graf, který však nebude ohodnocený a vzdálenosti mezi „původními“ vrcholy v něm budou odpovídat vzdálenostem v ohodnoceném grafu $G$.

### Algoritmus 4.2 pro ohodnocené grafy

Pro nalezení metriky $d$ (matice vzdáleností mezi každou dvojicí vrcholů) v kladně váženém grafu $G$ bychom mohli jednak použít modifikaci grafu $G$ jako v předchozím odstavci, ale mohli bychom poměrně snadno upravit samotný Algoritmus 4.2. Stačí, když během inicializace položíme

- `d[i][j]=0` jestliže vrcholy $i = j$,
- `d[i][j]=w(e)` jestliže vrcholy $i$ a $j$ jsou spojeny hranou $e$,
- nebo `d[i][j]` položíme rovno nekonečno (implementováno jako `MAX_INT/2`), pokud $i \neq j$ a hrana $ij$ v grafu není.

Změní se tedy jediný řádek algoritmu, inicializace. Místo řádku

```
d[i][j] = (i==j ? 0 : (G[i][j] ? 1 : MAX_INT/2));
```

použijeme řádek

```
d[i][j] = (i==j ? 0 : (G[i][j] ? w[i][j] : MAX_INT/2));
```

kde `w[i][j]` je dvourozměrné pole, které obsahuje v $i$-tém řádku a $j$-tém sloupci váhu hrany $ij$ nebo hodnotu $\infty$, pokud $i \neq j$ a hrana $ij$ neexistuje, případně hodnotu 0, pokud $i = j$. Algoritmus bude fungovat správně, dokonce se stejnou složitostí jako pro neohodnocené grafy.

V další sekci si ukážeme jiný algoritmus, který je vhodnější zejména v případě, kdy hledáme vzdálenosti mezi danou dvojicí vrcholů nebo když daný graf $G$ obsahuje málo hran. Při vhodné implementaci bude takový algoritmus navíc i rychlejší.

## Pojmy k zapamatování

— ohodnocený graf a vážený graf
— vzdálenost v ohodnocených grafech
— algoritmy pro určení vážené vzdálenosti

## 4.4 Nejkratší cesta v ohodnoceném grafu

### Průvodce studiem

*V této sekci ukážeme další modifikaci Algoritmu 2.1, tzv. Dijkstrův algoritmus (čti „Dajkstrův" podle nizozemského informatika Edsgera W. Dijkstry). Úschovna $U$ nebude implementována ani jako fronta, ani jako zásobník, ale jako množina, ze které budeme v každém kroku vybírat vrchol podle čísla, které mu bude v průběhu algoritmu přiřazené. Dijkstrův algoritmus lze použít pro nalezení nejkratší cesty z jednoho pevně zvoleného vrcholu do jiného zvoleného vrcholu, případně do všech vrcholů daného kladně ohodnoceného (váženého) grafu. Je to právě Dijkstrův algoritmus a jeho modifikace, které se používají v řadě praktických aplikací, například při vyhledávání vlakových a autobusových spojení.*

### Cíle

Po prostudování této sekce budete schopni:

- pomocí Dijkstrova algoritmu najít nejkratší cesty z pevně zvoleného vrcholu daného grafu,
- odhadnout složitost Dijkstrova algoritmu pro daný graf,
- vysvětlit, proč algoritmus funguje pouze pro kladně vážené grafy.

Mějme dán *kladně* vážený graf $G$ s ohodnocením $w$. Zvolíme jeden vrchol $u$ grafu $G$ za výchozí, z něj budeme hledat vzdálenost do jiného pevně zvoleného vrcholu $v$ grafu $G$ (případně do všech ostatních vrcholů v grafu $G$). Podobně jako v Algoritmu 2.1 budeme prohledávat graf $G$ postupem „do šířky". Budeme pracovat s následujícími proměnnými:

- počet vrcholů grafu $G$ označíme `N`,
- předpokládejme, že graf $G$ bude uložen pomocí seznamu sousedů v dvourozměrném poli `sous[][]`, kde prvek `sous[j][k]` udává $(k+1)$. souseda (čti „$k$ plus prvního souseda") vrcholu $j$; pozor, protože indexujeme od 0, jedná se o $(k+1)$. souseda, nikoliv o $k$. souseda; o seznamu sousedů je psáno na straně 26,
- pro uložení stupňů vrcholů použijeme jednorozměrné pole `deg[]`,
- pracujeme s kladně váženým grafem, váhy budou uloženy v dvourozměrném poli `w[][]`; hodnota `MAX_INT` bude reprezentovat $\infty$,
- v průběhu algoritmu budeme v poli `vzdal[]` pro každý vrchol $i$ ukládat informaci o délce nejkratší doposud nalezené cesty do vrcholu $i$,

- abychom uměli nejkratší nalezenou cestu zrekonstruovat, budeme v poli `pre[]` uchovávat pro každý vrchol $i$ informaci o předposledním vrcholu na nejkratší cestě do vrcholu $i$ (podobně jako u Algoritmu 4.1 na straně 67),
- protože se jedná o modifikaci algoritmu prohledávání grafu, budeme pro každý vrchol ukládat informaci o jeho stavu do pole `stav[]`; rozlišíme, zda je vrchol v iniciačním nebo zpracovaném stavu.

**Algoritmus 4.3 (Dijkstrův algoritmus).**

```
vstup: graf na N vrcholech daný seznamem sousedů sous[][] a w[][],
       kde sous[i][0], ..., sous[i][st[i]-1]} jsou sousedé vrcholu i
       stupně st[i] a hrana z i do sous[i][k] má délku w[i][k] > 0;
vstup: u, v (hledáme cestu z u do v);
// stav[i] udává stav vrcholu i:
//    0 ... iniciační
//    1 ... zpracovaný
// vzdal[i] udává zatím nalezenou vzdálenost do vrcholu i
// pre[i] udává, ze kterého vrcholu jsme do i přišli

// inicializace
for (i=0; i<=N; i++)   // MAX_INT dáme navíc i do vzdal[N]!
   { vzdal[i] = MAX_INT; stav[i] = iniciační; }
vzdal[u] = 0;

while (stav[v] == iniciační) {
   for (i=0, j=N; i<N; i++)     // vzdal[N] = MAX_INT
      if (stav[i] == iniciační && vzdal[i] < vzdal[j])
         j = i;
   // našli jsme nejbližší nezpracovaný vrchol j
   // zpracujeme jej
   if (vzdal[j] == MAX_INT) return NENI_CESTA;
   stav[j] = zpracovaný;
   for (k=0; k<st[j]; k++)
      if (vzdal[j]+w[j][k] < vzdal[sous[j][k]]) {
         pre[sous[j][k]] = j;
         vzdal[sous[j][k]] = vzdal[j]+w[j][k];
      }
      // pole pre[] obsahuje informaci o tom,
      // odkud jsme se do každého vrcholu dostali
}
výstup: 'Cesta délky vzdal[v], uložená pozpátku v poli pre[]';
```

V každém kroku vybereme z úschovny nalezených vrcholů vždy ten vrchol $j$, který má ze všech vrcholů v úschovně *nejmenší (dosud nalezenou) vzdálenost* od vý-

chozího vrcholu $u$. Tato informace je uložena v proměnné `vzdal[j]`. Později ukážeme, že takto zvolený vrchol už má správně určenou vzdálenost a protože je graf $G$ kladně vážený, tak neexistuje kratší cesta do vrcholu $j$ než ta, kterou algoritmus předtím nalezl.

Na konci zpracování budou v poli `vzdal[]` uloženy vzdálenosti od výchozího vrcholu do cílového vrcholu $v$ (a do každého vrcholu grafu $G$, který je blíže než vrchol $v$). Tuto cestu budeme umět zrekonstruovat tak, jak bylo popsáno na straně 67.

Pozor, Dijstrův algoritmus pracuje správně jen pro kladně vážené grafy. Proč tomu tak je vysvětlíme na straně 81.

Všimněte si, že algoritmus pracuje vždy pouze s prvky polí, které odpovídají vrcholům v úschovně. To znamená, že jakmile je nějaký vrchol $v$ zpracovaný, nebude se měnit ani odpovídající položka pole `vzdal[v]`, ani jeho předchůdce `vzdal[v]` na nejkratší cestě z výchozího vrcholu $u$.

Animovaná ukázka Dijkstrova algoritmu je na adrese http://mi21.vsb.cz/sites/mi21.vsb.cz/files/unit/dijkstruv_algoritmus.pdf.

### Složitost Algoritmu 4.3

Dijkstrův algoritmus 4.3 je sice složitější než Algoritmus 4.2, je však obvykle *výrazně rychlejší*. Jestliže hledáme nejkratší cestu z výchozího vrcholu $u$ do nějakého vybraného vrcholu $v$, tak doba běhu závisí v podstatě pouze na volbě vrcholu $v$. Čím vzdálenější bude vrchol $v$ od vrcholu $u$ vzhledem k ostatním vrcholům, tím bude potřebný počet kroků větší. V nejhorším případě však bude počet kroků odpovídat počtu kroků pro nalezení nejkratší cesty do všech vrcholů. Celkový počet kroků Dijkstrova algoritmu potřebný k nalezení nejkratší cesty z výchozího vrcholu $u$ do všech ostatních vrcholů (v případě, že za $v$ zvolíme vrchol, který je od $u$ nejvzdálenější) je úměrný zhruba $n^2$, kde $n$ je počet vrcholů grafu. Proto složitost Dijkstrova algoritmu je $O(n^2)$.

Při vhodné implementaci úschovny nezpracovaných vrcholů (např. haldou, ve které používáme nalezenou vzdálenost jako vyhledávací klíč) lze dosáhnout i rychlejšího běhu tohoto algoritmu pro řídké grafy s malým počtem hran $m$. Potom je počet kroků algoritmu úměrný počtu hran grafu a složitost Dijkstrova algoritmu je $O(m)$. Pro husté grafy je $O(m) = O(n^2)$ a je tedy stejná jako pro Algoritmus 4.1, který hledá vzdálenosti z jednoho pevně zvoleného výchozího vrcholu v neohodnocených grafech.

### Modifikace Dijkstrova algoritmu

Jestliže vnější cyklus nebude řízen podmínkou `(stav[v] == iniciační)`, ale triviálně splněnou podmínkou, tak najde takto upravený algoritmus vzdálenost i nejkratší cesty z výchozího vrcholu $u$ do všech ostatních vrcholů. Triviálně splněnou podmínkou máme na mysli nekonečný cyklus řízený podmínkou `while (1)`. Uvědomte si, že po skončení takového upraveného Dijkstrova algoritmu obsahuje každá

položka pole `vzdal[]` vzdálenost z výchozího vrcholu $u$, zatímco po skončení Algoritmu 4.3 je zaručena správnost hodnot v poli `vzdal[]` pouze u zpracovaných vrcholů.

Pokud bychom chtěli Dijkstrův algoritmus použít pro nalezení nejkratších vzdáleností mezi všemi dvojicemi vrcholů, bude nutno jej spustit opakovaně – jednou pro každý vrchol zvolený jako výchozí. Složitost takového algoritmu pak bude $O(n^3)$.

Rozmyslete si, že Algoritmus 4.3 lze použít i pro hledání nejkratších cest v *orientovaném* grafu. Pouze bude jinak vypadat struktura vstupních dat, ale žádné další úpravy samotného algoritmu nejsou potřeba.

Dále je snadné Dijkstrův algoritmus modifikovat i pro hledání *nejširší* cesty (například pro přepravu nadrozměrného nákladu) z výchozího vrcholu $u$ do jiného vrcholu, případně do všech vrcholů daného grafu. Předpokládejme, že váha hran udává jejich šířku a je uložena v poli `w[][]` a že pole `vzdal[]` obsahuje šířky nejširších nalezených cest. Stačí nahradit část algoritmu

```
if (vzdal[j]+w[j][k] < vzdal[sous[j][k]]) {
     pre[sous[j][k]] = j;
     vzdal[sous[j][k]] = vzdal[j]+w[j][k];
}
```

následujícími řádky

```
if (min(vzdal[j], w[j][k]) > vzdal[sous[j][k]]) {
     pre[sous[j][k]] = j;
     vzdal[sous[j][k]] = min(vzdal[j], w[j][k]);
}
```

Současně je nutno upravit inicializaci na `vzdal[i] = MAX_INNT` na `vzdal[i] = 0`, dále pro výchozí vrchol položíme `vzdal[u] = MAX_INT` a při hledání nezpracovaného vrcholu obrátíme znaménko nerovnosti

```
if (stav[i] == iniciační && vzdal[i] > vzdal[j])
```

a konečně neexistenci dalších cest ověříme podmínkou

```
if (vzdal[j] == 0) return NENI_CESTA;
```

neboť algoritmus skončí, jakmile v úschovně nebude žádný nezpracovaný vrchol s nenulovou „vzdáleností“ od výchozího vrcholu (připomínáme, že vzdálenost v upraveném algoritmu má význam šířky).

### Důkaz správnosti Dijkstrova algoritmu

Nyní ukážeme, že v kladně vážených grafech pracuje Dijkstrův algoritmus správně.

**Věta 4.16.** *Mějme kladně vážený graf $G$ a v něm nějaké dva pevně zvolené vrcholy $u$ a $v$. Algoritmus 4.3 najde nejkratší cestu z vrcholu $u$ do vrcholu $v$.*

*Důkaz.* Klíčovým faktem je, že v každé iteraci obsahuje pole `vzdal[]` takové vzdálenosti (a nejkratší cesty) z vrcholu $u$ do ostatních vrcholů, které vedou pouze přes již zpracované vrcholy. Pro přehlednost označme $S$ množinu vrcholů, které jsou už zpracované do určité iterace Algoritmu 4.3. Ukážeme, že tento fakt se v průběhu algoritmu nezmění a proto po skončení algoritmu, kdy jsou zpracované všechny vrcholy dané komponenty, jsou všechny získané vzdálenosti (a cesty) nejkratší.

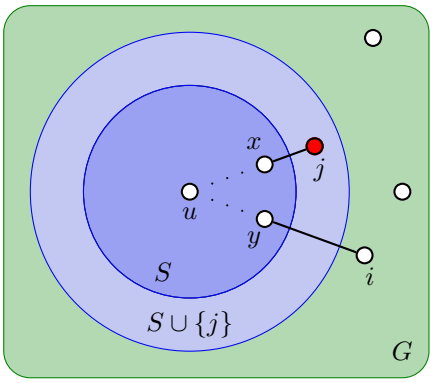
Důkaz provedeme matematickou indukcí vzhledem k počtu iterací.

*Základ indukce:* V první iteraci Algoritmu 4.3 vybereme vrchol $j$, který je v inicializačním stavu a má nejmenší hodnotu v poli `vzdal[j]`. Takový vrchol je jediný – a sice vrchol $j = u$. Tento vrchol zpracujeme a upravíme vzdálenosti všem jeho sousedům podle vah hran, které je spojují s vrcholem $j$. Předpokládaný fakt je splněn triviálně, neboť na konci iterace je $S = \{u\}$ a získané vzdálenosti do všech vrcholů jsou nejmenší, jestliže bereme v úvahu pouze cesty vedoucí přes vrcholy množiny $S$.

*Indukční krok:* V každé další iteraci vybereme ten vrchol $j$, který má ze všech nezpracovaných vrcholů nejkratší nalezenou vzdálenost od vrcholu $u$. To ale současně znamená, že vzdálenost (a cesta) do vrcholu $j$ je nejkratší možná v celém grafu a že žádná kratší cesta už do vrcholu $j$ nevede. Uvědomme si, že každá „oklika“ přes jiné nezpracované vrcholy $i$ (vrcholy mimo množinu $S$) musí být delší díky výběru vrcholu $j$ (Obrázek 4.5). Zde využíváme toho, že *žádná hrana v grafu nemá záporné ohodnocení*, protože jinak by přičtení takové záporné váhy k délce jiné cesty do vrcholu $j$ mohlo znamenat kratší cestu než přes zpracované vrcholy v množině $S$. V iteraci algoritmu zbývá projít (a případně „vylepšit“) cesty do sousedů vrcholu $j$.

Na konci algoritmu zbývá sestavit nejkratší cestu z vrcholu $u$ do vrcholu $v$. Je uložená v poli `pre[]`, můžeme ji „pozpátku“ od vrcholu $v$ zrekonstruovat: předposlední vrchol před $v$ je `pre[v]`, před předposlední vrchol je `pre[pre[v]]`, atd. . .

Podle principu matematické indukce je důkaz hotov. □




Obrázek 4.5 K důkazu správnosti Dijkstrova algoritmu.

### Poznámky k Dijkstrovu algoritmu

Všimněte si, že je-li nějaká hrana v daném grafu záporně ohodnocená, tak v důkazu nemůžeme použít tvrzení, že nejkratší cesty do vybraného vrcholu $j$ vedou pouze přes zpracované vrcholy v množině $S$. V některých velmi speciálních případech by takový algoritmus mohl fungovat, ale výhodnější je použít jiné (třebaže poněkud složitější algoritmy), které umí najít nejkratší cesty i v grafech se zápornými ohodnoceními. Takový je například Bellman-Fordův algoritmus, avšak jeho popis přesahuje rámec tohoto textu.

Pěkná analogie mezi Algoritmem 4.1 a Dijkstrovým algoritmem je popsána v knize [1]. Autor ukazuje, jak odvodit Dijkstrův algoritmus z Algoritmu 4.1 pomocí velkého množství paralelně prozkoumávaných hran.

## Pojmy k zapamatování

— Dijkstrův algoritmus

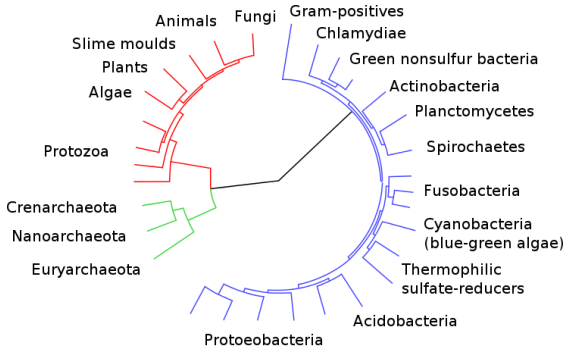
— modifikace Dijkstrova algoritmu

# Kapitola 5

# Stromy

## Průvodce studiem

*Mezi nejběžnější útvary v přírodě i v matematice patří „stromy“, tj. objekty se „stromovou strukturou“. Existuje celá řada situací, které můžeme popsat „stromem“: rodokmeny, evoluční strom, elektrické rozvodné sítě, hierarchické struktury se vztahem nadřazený a podřízený, popis větvení při vyhledávání a pod. Všechny tyto struktury mají jednu věc společnou – neobsahují „cykly“.*




Obrázek 5.1 Zjednodušený evoluční strom a rodokmen.

*V evolučním stromu cykly nenajdeme, cyklus v rozvodné síti může znamenat krátké spojení a cyklus v rodokmenu zavání incestem. Ukážeme několik základních tvrzení týkající se grafů, kterým budeme říkat stromy. Zavedeme pojem centra stromu a ukážeme, jak centrum vypadá a že centrum stromu je určeno jednoznačně.*

*V sekci 1.5 jsme zmínili, že je obtížné rozpoznat, zda dva obecné grafy jsou isomorfní. V sekci 5.3 ukážeme, že ve speciálním případě, kdy máme dva stromy, je poměrně snadné rozpoznat, zda jsou isomorfní.*

*V poslední části kapitoly zavedeme důležitý pojem kostry grafu a uvedeme několik algoritmů pro hledání kostry obecného grafu. Na závěr zmíníme jednu nečekanou a pěknou aplikaci algoritmů pro hledání minimální kostry.*

# 5.1 Základní vlastnosti stromů

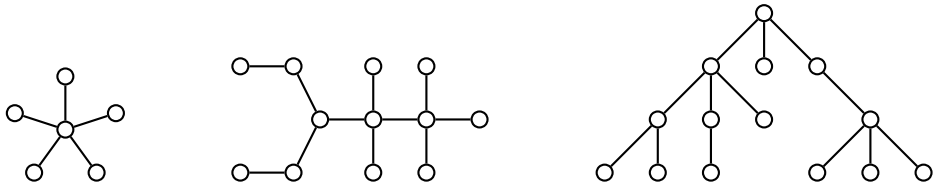
## Cíle

Po prostudování této sekce budete schopni:

- popsat základní vlastnosti stromu,
- rozpoznat strukturu stromu v praktické úloze,
- popsat kostru grafu a její význam.

Řekneme, že graf neobsahuje cyklus (kružnici), jestliže žádný jeho podgraf není cyklem. Jedním slovem se takovému grafu říká *acyklický* graf.

**Definice 5.1.** *Strom* je souvislý graf, který neobsahuje cyklus. *Les* je graf, jehož komponenty jsou stromy.



Obrázek 5.2 Příklady stromů.

Na Obrázku 5.2 jsou tři příklady grafů, kterým říkáme stromy. Tyto grafy tvoří dohromady jediný graf, který je lesem.

**Poznámka 5.2 (O názvosloví).** Obvykle je lepší definovat nejprve obecnější strukturu a potom specifikovat její další vlastnosti a popsat speciální případy. Bylo by tak přehlednější definovat *les* jako jednoduchý graf, který neobsahuje cyklus a *strom* jako souvislý les. Taková definice stromu je sice korektní, avšak říkat, že „strom je souvislý les“ zní v běžné řeči podivně.

V každém stromu, který obsahuje alespoň dva vrcholy, budeme rozlišovat, zda vrchol je stupně 1 nebo většího. Vrcholům stupně 1 budeme říkat *list* (tento název opět pěkně koresponduje s analogií k opravdovým stromům). Ostatním vrcholům budeme říkat *nelistové vrcholy*. Listy budou často hrát roli při různých konstrukcích, popisech algoritmů nebo v důkazech. Nejprve ukážeme, že list najdeme v každém stromu s výjimkou triviálního grafu (který je také stromem).

**Lemma 5.3.** *Každý strom s více než jedním vrcholem má alespoň jeden list.*

*Důkaz.* Mějme nějaký strom $T$ a v něm vybereme libovolný vrchol $v$. Protože $T$ je souvislý graf s více než jedním vrcholem, tak nemůže mít vrchol $v$ stupeň 0 – takový graf by nebyl souvislý.

Ve stromu $T$ budeme sestrojovat co nejdelší tah $S$, který začíná ve vrcholu $v$. Tah $S$ začneme libovolnou hranou vycházející z vrcholu $v$ (už víme, že taková hrana existuje). Nyní v každém dalším vrcholu $u$ sestavovaného tahu $S$ rozlišíme:

- buď má vrchol $u$ stupeň alespoň 2; potom prodloužíme tah $S$ o další hranu do nového vrcholu (který nyní označíme $u$) a postup opakujeme,
- nebo je vrchol $u$ stupně 1 a zároveň je hledaným listem.

Všimněte si, že pokud by se při konstrukci tahu $S$ mohl některý vrchol stromu $T$ zopakovat, tak by tah $S$ obsahoval nějaký cyklus. Tento cyklus by byl současně podgrafem stromu $T$, což odporuje definici stromu! Protože strom $T$ je konečný, tak popsaná konstrukce tahu $S$ musí skončit v hledaném listu. □

Uvedený důkaz je konstruktivní, ukazuje, jak v grafu list najít. Pokud známe nejen strukturu grafu, ale i stupně vrcholů, je samozřejmě nalezení listu ještě jednodušší.

Všimněte si, že strom s jediným vrcholem je z jistého úhlu pohledu výjimečný – neobsahuje list. Říká se mu *triviální strom*. Řada tvrzení (nejen Lemma 5.3) platí pro všechny stromy s výjimkou triviálního stromu, a proto někteří autoři triviální strom za strom nepovažují. Mějte to na paměti při studiu odborné literatury.

Není těžké ukázat, že v každém netriviálním stromu existují vždy alespoň dva vrcholy stupně 1. Navíc se dá ukázat, že je-li ve stromu $T$ nějaký vrchol stupně $k$, tak v $T$ najdeme alespoň $k$ listů. A dokonce se dá spočítat, kolik listů je v daném stromu, jestliže známe stupně všech nelistových vrcholů (Příklad 5.9). My však v dalším textu budeme pracovat s „alespoň jedním“ listem, proto vystačíme s tvrzením tak, jak bylo zformulováno (a dokázáno) v Lemmatu 5.3.

Nyní ukážeme, že počet hran stromu s daným počtem vrcholů je určen jednoznačně. Tvrzení můžeme interpretovat tak, že každý strom na daném počtu vrcholů má stejný počet hran.

**Věta 5.4.** *Strom s $n$ vrcholy má právě $n-1$ hran.*

*Důkaz.* Tvrzení ukážeme indukcí podle počtu vrcholů.
*Základ indukce:* Triviální strom s jedním vrcholem má $n-1=0$ hran a proto pro triviální strom tvrzení platí.
*Indukční krok:* Mějme netriviální strom $T$ s $n$ vrcholy (tj. $n>1$). Předpokládejme, že pro každý strom s méně než $n$ vrcholy tvrzení platí, tedy že takový strom má o jednu hranu méně než vrcholů. (Všimněte si, že indukční předpoklad stačí omezit na stromy s $n-1$ vrcholy.)

Nyní využijeme Lemma 5.3, podle kterého má strom $T$ alespoň jeden list $v$ (vrchol stupně 1).

Označme $T'=T-v$ graf, který vznikne ze stromu $T$ odebráním vrcholu $v$ a jediné(!) hrany s ním incidentní (tzv. „oholením“). Odebráním listu neporušíme souvislost grafu, neboť žádná cesta mezi dvěma vrcholy různými od $v$ nemůže vést přes vrchol $v$ stupně 1. Proto je graf $T'$ souvislý. Dále odebráním hrany nemůže vzniknout cyklus a proto je graf $T'$ také bez cyklů. To znamená, že $T'$ je strom s $n-1$ vrcholy a podle indukčního předpokladu má o jednu hranu méně než vrcholů.

Strom $T'$ má tedy $(n-1)-1$ hran a původní strom $T$ má o jeden vrchol a jednu hranu více než $T'$, tj. strom $T$ má $(n-1)-1+1 = n-1$ hran.

Podle principu matematické indukce je tvrzení dokázáno. □

**Příklad 5.5.** V databázi je 12 záznamů (objektů) a 34 vazeb mezi nimi. Chtěli bychom strukturu databáze přehledně znázornit grafem, ve kterém jsou objekty znázorněny jako vrcholy a vazby mezi nimi jsou zakresleny jako hrany. a) Bude takový graf stromem? b) Můžeme tvrdit, že takový výsledný graf bude souvislý?

*Řešení.* a) Výsledný graf jistě nebude stromem, ale musí obsahovat cykly, protože strom s 12 vrcholy by musel mít právě 11 hran (vazeb).

A dokonce ani kdyby vazeb bylo 11, nemohli bychom tvrdit, že výsledný graf bude strom, neboť záleží na struktuře dat. Výsledný graf by mohl obsahovat cykly; v takovém případě se však dá ukázat, že by nebyl souvislý.

b) Výsledný graf může, ale nemusí být souvislý. Výsledek velmi závisí na struktuře dat. Výsledný graf by mohl například obsahovat jednu komponentu s 9 vrcholy a 3 izolované vrcholy, neboť kompletní graf $K_9$ obsahuje $\binom{9}{2} = 36$ hran a my z něj můžeme vynechat 2 libovolné hrany.

Dokonce ani graf s 12 vrcholy a 55 hranami nemusí být souvislý, neboť graf se dvěma komponentami $K_1$ a $K_{11}$ má právě 12 vrcholů a 55 hran. Pokud by však daný graf měl 12 vrcholů a více než 55 hran, musí už být souvislý. Vskutku, graf $K_{12}$ má 66 hran a je hranově 11-souvislý a proto vynecháním libovolných méně než $66-55 = 11$ hran zůstane výsledný graf souvislý.

▲

Protože strom je souvislý graf, tak podle definice víme, že mezi každými dvěma vrcholy stromu najdeme alespoň jednu cestu. Nyní ukážeme silnější tvrzení, že mezi každou dvojicí vrcholů existuje ve stromu vždy *jediná* cesta. (Zde samozřejmě nerozlišujeme pořadí počátečního a koncového vrcholu, tj. pro pevně zvolené vrcholy $u$, $v$ považujeme cestu z $v$ do $u$, kterou dostaneme obrácením pořadí vrcholů cesty z $u$ do $v$, za stejnou cestu mezi vrcholy $u$ a $v$).

**Věta 5.6.** *Mezi každými dvěma vrcholy stromu existuje právě jedna cesta.*

*Důkaz.* Tvrzení dokážeme sporem. Budeme vycházet ze současné platnosti předpokladu věty ($T$ je strom) a negace tvrzení (mezi některými vrcholy stromu $T$ neexistuje žádná nebo existuje více než jedna cesta) a dojdeme ke sporu.

Podle definice stromu je $T$ souvislý graf, a proto víme, že mezi každými dvěma vrcholy $u$, $v$ stromu $T$ vede nějaká cesta. Pro spor tedy předpokládáme, že takových cest existuje více. Mějme nějaké dvě různé cesty mezi vrcholy $u$, $v$. Sjednocení těchto cest je uzavřený sled ve stromu $T$, ze kterého po vynechání případných opakujících se hran a vrcholů vznikne cyklus. Tento cyklus je podgrafem ve stromu $T$, což je spor s předpokladem, že $T$ je strom a neobsahuje cyklus.

Vidíme, že negace tvrzení vede ke sporu (sporem myslíme situaci, kdy jsme ukázali, že strom $T$ současně obsahuje i neobsahuje cyklus) a proto platí, že mezi vrcholy $u$ a $v$ existuje ve stromu právě jedna cesta. □

**Poznámka 5.7.** Tvrzení Věty 5.6 bychom mohli ukázat i nepřímo tak, že dokážeme obměnu věty. Vyjdeme z negace tvrzení (mezi některými vrcholy stromu $T$ neexistuje žádná nebo existuje více než jedna cesta) a odvodíme negaci předpokladu věty (daný graf není strom).

Podle Věty 5.4 víme, že strom s $n$ vrcholy má právě $n-1$ hran. Přidáním nějaké hrany do stromu souvislost neporušíme, ale výsledný graf už není stromem (má více než $n-1$ hran) a proto musí obsahovat nějaký cyklus. Z Věty 5.6 snadno ukážeme, že takový cyklus bude jediný.

**Důsledek 5.8.** *Přidáním jedné (nové) hrany do stromu (s alespoň třemi vrcholy) vznikne graf s právě jedním cyklem.*

*Důkaz.* Mějme strom $T$ a nějaké dva jeho vrcholy $u$, $v$, mezi kterými není hrana. Přidáním hrany $uv$ vznikne právě jeden cyklus spojením $uv$ a jediné cesty mezi vrcholy $v$, $u$ ve stromu $T$ (jediné podle Věty 5.6). □

Pozor, po přidání dvou hran do stromu počet cyklů výsledného grafu závisí na tom, kam hrany přidáme. Počet cyklů už není (na rozdíl od případu popsaného v Důsledku 5.8) určen jednoznačně. Výsledný graf může obsahovat dva nebo tři cykly.

## Kontrolní otázky

1. Jak se jmenuje strom, který má $n$ vrcholů a mezi nimi právě dva listy?
2. Kolik musíme z grafu $K_n$ vynechat hran, abychom dostali strom?

**Příklad 5.9.** Ukažte, že známe-li stupně $\deg(v_1), \deg(v_2), \ldots, \deg(v_k)$ všech nelistových vrcholů v nějakém stromu $T$, umíme také určit počet listů ve stromu $T$.

*Řešení.* Podle Věty 5.4 víme, že každý strom $T$ s $n$ vrcholy má $n-1$ hran. Nevíme sice zatím kolik je listů (neznáme počet $n-k$), ale víme, že každý list je stupně 1.

Dále víme podle Principu sudosti (Věta 1.15), že součet stupňů vrcholů je roven dvojnásobku počtu hran. Dostáváme tak následující rovnost.

$$\begin{aligned} 2(n-1) &= \sum_{i=1}^{k} \deg(v_i) + (n-k)\cdot 1 \\ 2n-2 &= \sum_{i=1}^{k} \deg(v_i) + n - k \\ n &= 2 - k + \sum_{i=1}^{k} \deg(v_i) \end{aligned}$$

Hledaný počet listů $n-k$ je proto

$$n-k = 2-2k+\sum_{i=1}^{k}\deg(v_i) = 2+\sum_{i=1}^{k}(\deg(v_i)-2)\,.$$

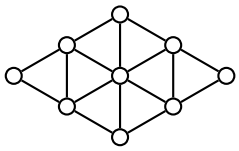
Například strom, který má čtyři nelistové vrcholy stupně 3 má $2+4(3-2)=6$ listů a celkem 10 vrcholů. ▲

Máme dán nějaký souvislý graf $G$. Při určování stupně hranové souvislosti jsme se ptali, kolik *nejméně* stačí z grafu odstranit hran, aby se graf $G$ stal nesouvislý.

Má však také smysl se ptát, kolik *nejvíce* hran můžeme z grafu $G$ odstranit, aby $G$ zůstal souvislý, nebo naopak kolik *nejméně* hran musí v grafu $G$ zůstat, aby byl ještě souvislý. Jsou to právě stromy, které jsou souvislé grafy, ale už z nich nemůžeme odebrat žádnou hranu, aniž by se porušila souvislost. To ukazuje následující věta.

**Věta 5.10.** *Strom je* minimální souvislý *graf (s daným počtem vrcholů).*

*Důkaz.* Strom je podle definice souvislý graf. Pokud souvislý graf obsahuje cyklus, zůstane souvislý i po vynechání některé hrany cyklu. Proto je minimální souvislý graf acyklický a je tedy stromem. □



Obrázek 5.3 Graf $G$.

**Příklad 5.11.** Kolik nejvíce hran můžeme odstranit z grafu $G$ na Obrázku 5.3, aby výsledný graf zůstal souvislý?

*Řešení.* Podle Věty 5.10 víme, že výsledný graf po odebírání hran bude stromem. Protože graf $G$ má 9 vrcholů a 16 hran, tak podle Věty 5.4 víme, že můžeme odstranit nejvíce 8 hran.

Z grafu na Obrázku 5.3 můžeme dokonce odstranit takových 8 hran, že odstraněné hrany tvoří spolu s vrcholy grafu $G$ souvislý faktor a současně ponechané hrany tvoří spolu s vrcholy grafu $G$ jiný souvislý faktor. Najdete takových 8 hran? ▲

## Pojmy k zapamatování

— strom a les

— list

— počet hran stromu

## 5.2 Kořenové stromy

### Průvodce studiem

*V této sekci zavedeme několik pojmů, které využijeme v další části při zkoumání isomorfismu stromů. Zavedeme pojem kořenového stromu a uspořádaného kořenového stromu. Nadefinujeme centrum stromu a na příkladech ukážeme, jak centrum stromu najít. Na závěr zmíníme obecnější pojem – centrum grafu.*

### Cíle

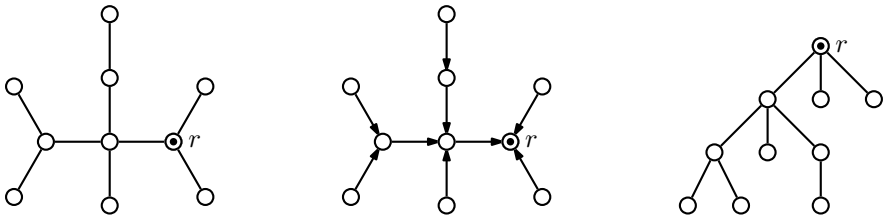
Po prostudování této sekce budete schopni:

- najít centrum stromu,
- vysvětlit rozdíl mezi stromem a kořenovým stromem,
- vysvětlit rozdíl mezi kořenovým stromem a uspořádaným kořenovým stromem.

Při práci se strukturou typu strom je někdy užitečné označit význačný vrchol, tzv. *kořen*. Kořen může představovat jakýsi „začátek“ uložených dat, nebo hlavní prvek v nějaké hierarchické struktuře objektů. Motivaci pro zavedení kořenových stromů najdeme mimo jiné v rodokmenech, ze kterých je převzata i terminologie.

**Definice 5.12.** *Kořenovým stromem* nazveme strom $T$ spolu s vyznačeným *kořenem* $r \in V(T)$ (značení $T$ pochází z anglického „tree“). Kořenový strom proto chápeme jako dvojici $(T, r)$, případně říkáme pouze strom $T$ s kořenem $r$.

Neměli bychom zaměňovat pojmy „strom“ a „kořenový strom“, protože kořenový strom obsahuje další informaci – víme, který vrchol je kořenem.




Obrázek 5.4 Různá nakreslení téhož kořenového stromu $T$ s vyznačeným kořenem $r$; v prostředním obrázku je navíc vyznačena orientace směrem ke kořenu.

Na Obrázku 5.4 máme tři různá nakreslení téhož kořenového stromu $(T, r)$. Vlevo na obrázku je nejjednodušší nakreslení – máme strom $T$ a jeho kořen $r$ je označený pro přehlednost puntíkem. Uvědomte si, že zvolením kořene v libovolném stromu umíme systematicky přiřadit orientaci každé hraně, například od listů ke kořeni. V některých knihách upřednostňují opačnou orientaci, obvykle závisí na interpretaci úlohy, která je grafem modelována. Orientace hran je znázorněna na Obrázku 5.4

uprostřed. V nakreslení však tuto orientaci nemusíme uvádět, protože orientaci snadno určíme.

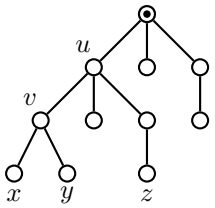
Pro přehlednost se kořenové stromy často zakreslují tak, aby kořen byl v nakreslení výš než ostatní vrcholy. Další vrcholy kreslíme do různých úrovní níž a níž podle jejich vzdálenosti od kořene. V tomto nakreslení se orientace hran určí ještě jednodušeji – jsou orientovány vždy „nahoru.“ Příklad takového nakreslení je na Obrázku 5.4 vpravo. V tomto nakreslení je navíc jednoznačně určen kořen i bez jeho označení – jedná se o vrchol, který je v obrázku nejvýše.

V dalším textu budeme kořenové stromy zakreslovat právě tak, aby kořen byl nahoře a další vrcholy budou zakresleny pod ním. Upozorníme ještě, že v některých knihách se kořenové stromy zakreslují „naopak“, s kořenem dole, což může někdy lépe odpovídat problému, který je stromem modelován.

### Rodiče a potomci

V kořenovém stromu můžeme pro dvojici sousedních vrcholů jednoznačně určit „příbuzenský vztah,“ což usnadňuje slovní formulace. Nejprve zavedeme následující pojmy.

**Definice 5.13.** Mějme dán kořenový strom $(T, r)$ a v něm dvojici vrcholů $u$, $v$, kde vrchol $u$ je sousední s vrcholem $v$ na cestě z vrcholu $v$ ke kořenu $r$. Pak $u$ nazýváme *rodičem* vrcholu $v$ a $v$ nazýváme *potomkem* vrcholu $u$.




Obrázek 5.5 Kořenový strom $T$ s rodiči a potomky.

Na Obrázku 5.5 máme vyznačený vrchol $u$, který je rodičem vrcholu $v$ a vrchol $v$ je potomkem rodiče $u$. Naproti tomu vrchol $v$ je rodičem vrcholů $x$ a $y$, přičemž vrcholy $x$ a $y$ jsou jeho potomci.

Někdy je šikovné popisovat jiné vztahy mezi vrcholy i když nemáme zavedenu formální definici. Je zřejmé, co máme na mysli, když například řekneme, že na Obrázku 5.5 „vrcholy $x$ a $y$ jsou sourozenci“, nebo „vrchol $u$ je prarodičem vrcholu $x$.

Všimněte si, že jinou volbou kořene se může vzájemný vztah vrcholů změnit. Pokud bychom ve stromu $T$ na Obrázku 5.5 zvolili za kořen vrchol $x$, tak najednou bude vrchol $v$ rodičem vrcholu $u$ a naopak $u$ bude potomkem $v$. Pokud bychom však za kořen zvolili vrchol $z$, tak vztah mezi vrcholem $u$ a $v$ zůstane stejný jako na Obrázku 5.5.

V kořenových stromech se místo „list“ často používá termín *koncový vrchol*. Kořen může být listem, avšak v tomto případě preferujeme název *kořen* a neříkáme mu koncový vrchol. V kořenovém stromu koncové vrcholy nemají potomky a nejsou kořenem. Například v grafu na Obrázku 5.5 je šest koncových vrcholů.

### Centrum stromu

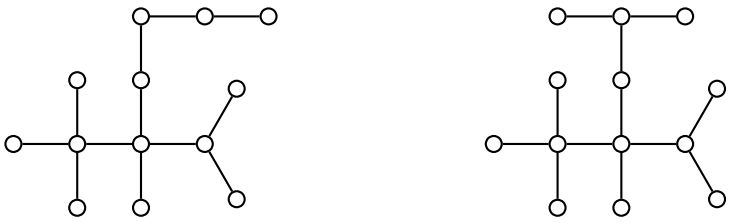
V další části bude pro kořen stromu hrát důležitou roli tzv. *centrum* stromu. Centrem stromu nazveme vrchol případně hranu spojující dvojici vrcholů, které získáme postupným odstraňováním listů. Procesu, kdy ve stromu označíme všechny jeho listy a potom je odstraníme (spolu s incidentními hranami), se říká *oholení stromu*.

**Definice 5.14.** *Centrem stromu* $T$ rozumíme vrchol nebo hranu v daném stromu $T$, které určíme následujícím postupem:

1. Jestliže má strom $T$ jediný vrchol $v$, je $v$ centrum stromu $T$. Pokud má strom $T$ dva vrcholy, je centrem stromu $T$ hrana spojující tyto dva vrcholy. Jinak přejdeme k bodu 2.
2. Vytvoříme (menší) strom $T' \subset T$ oholením všech listů stromu $T$ a vracíme se na předchozí bod 1.

Rekurzivním postupem získané centrum stromu $T'$ bude zároveň centrem stromu $T$.

Je zřejmé, že strom $T'$, který vznikne ze stromu $T$ je menší než strom $T$, protože v každém netriviálním stromu budeme podle Lemmatu 5.3 holit alespoň jeden list. Také si snadno rozmyslíme, že oholením v kroku 2) vznikne neprázdný strom, neboť netriviální strom, jehož každý vrchol je listem, je pouze strom na dvou vrcholech, jehož centrem je podle kroku 1) jediná jeho hrana.

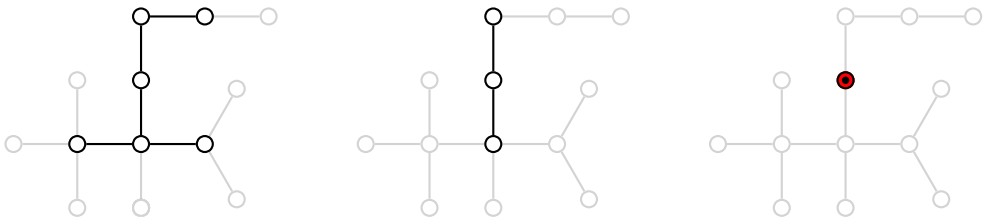


Obrázek 5.6 Stromy $T_1$ a $T_2$.

**Příklad 5.15.** Najděte centrum stromu $T_1$ na Obrázku 5.6 vlevo.

*Řešení.* Ve stromu $T_1$ najdeme všechny listy (je jich sedm) a odstraníme je (Obrázek 5.7 vlevo). Na obrázku jsou listy i hrany s nimi incidentní pro názornost ponechány, jsou však obarveny šedou barvou. Protože výsledný strom má více než dva vrcholy, celý postup zopakujeme. Na Obrázku 5.7 uprostřed jsme odebrali další

tři listy výsledného menšího stromu. Protože výsledný strom má opět více než dva vrcholy, celý postup znovu zopakujeme. Dostaneme strom s jediným vrcholem, který podle Definice 5.14 je centrem stromu $T_1$ (Obrázek 5.7 vpravo), označíme jej červeně.

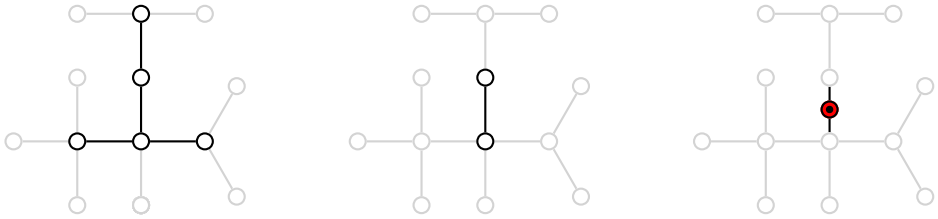


Obrázek 5.7 Hledání centra stromu $T_1$.

▲

**Příklad 5.16.** Najděte centrum stromu $T_2$ na Obrázku 5.6 vpravo.

*Řešení.* Ve stromu $T_2$ najdeme všechny listy (je jich osm) a odstraníme je (Obrázek 5.8 vlevo). Opět jsou listy i hrany s nimi incidentní obarveny šedou barvou. Protože výsledný strom má více než dva vrcholy, celý postup zopakujeme. Na Obrázku 5.8 uprostřed jsme odebrali další tři listy výsledného menšího stromu. Výsledný strom má jen dva vrcholy. Proto podle Definice 5.14 je hrana, která je spojuje, centrem stromu $T_2$. Centrum jsme na Obrázku 5.8 vpravo označili jako nový vrchol na této hraně.



Obrázek 5.8 Hledání centra stromu $T_2$.

▲

V Příkladu 5.16 jsme na hranu centra přidali nový vrchol. Výsledný graf má o jeden vrchol více, než měl původní strom $T$, přesto budeme pro jednoduchost nazývat tento vrchol „centrem stromu $T$“, ačkoliv se jedná o jiný strom než $T$.

Je zajímavé si rozmyslet, proč vrcholy centra stromu jsou „uprostřed stromu“, tj. žádný jiný vrchol stromu nemá tak malou maximální vzdálenost od ostatních vrcholů, jako má vrchol centra (je-li jediný) nebo koncové vrcholy centra (je-li centrem hrana).

Všimněte si, že kořen stromu obecně *nemusí* ležet v centru. My však často budeme za kořen volit právě centrum stromu. V následující sekci budou určení centra a umístění kořene do centra základní kroky algoritmu, který bude rozhodovat o isomorfismu daných stromů.

### Kořen a centrum stromu

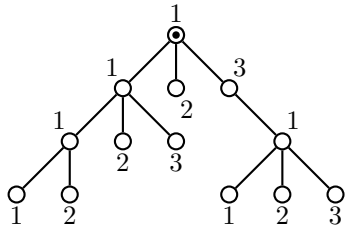
Budeme-li chtít nějakému stromu přiřadit kořen *jednoznačně*, tj. aby bylo jednoznačně určeno, který vrchol bude zvolen, tak je nejlepší zvolit za kořen *centrum*, protože centrum je určeno jednoznačně. V případě, že centrem stromu je hrana $uv$, „přidáme“ na tuto hranu nový vrchol a zvolíme jej kořenem. Přesněji: z daného stromu $T$ odstraníme hranu $uv$, přidáme kořen $r$ a přidáme dvě hrany $ur$, $vr$. Získáme tak strom $T'$. Snadno nahlédneme, že nově přidaný vrchol $r$ je centrem stromu $T'$. Strom $T'$ bude pak v dalších úvahách zastupovat strom $T$.

### Uspořádané kořenové (pěstované) stromy

Další informací vázanou ke kořenovým stromům je uspořádání potomků každého vrcholu (jako seřazení potomků jedné generace v rodokmenech podle data jejich narození).

**Definice 5.17.** Kořenový strom $(T, r)$ je navíc *uspořádaný*, jestliže pro každý jeho vrchol je jednoznačně dáno pořadí jeho potomků (například v nakreslení uspořádáním potomků „zleva doprava“). Uspořádaný kořenový strom se také nazývá *pěstovaný strom*.

Formálně můžeme pěstovaný strom popsat jako trojici $(T, r, f)$, kde $T$ je nějaký strom a $r$ jeho kořen. Funkce $f : V(T) \to \mathbb{N}$ přitom přiřadí každému vrcholu jeho pořadí mezi sourozenci. Má-li tedy vrchol $k$ potomků, budou těmto potomkům přiřazena čísla $1, 2, \ldots, k$. Jestliže nějaký vrchol nemá sourozence (například kořen, který není potomkem žádného vrcholu), můžeme mu přiřadit libovolné přirozené číslo, řekněme třeba číslo 1.




Obrázek 5.9 Uspořádaný (pěstovaný) strom.

### Centrum grafu

V Definici 5.14 jsme zavedli centrum stromu. Má smysl definovat také centrum souvislého grafu, avšak pro určení centra obecného grafu nevystačíme s Definicí 5.14. Obsahuje-li graf cyklus, tak oholením neodstraníme žádný vrchol cyklu, protože každý vrchol cyklu je stupně alespoň 2 a není listem.

### Pro zájemce:

Abychom mohli popsat centrum obecného grafu, musíme nejprve zavést několik pojmů. *Excentricitou vrcholu* nazveme číslo, které pro každý vrchol udává největší vzdálenost, kterou z daného vrcholu do dalších vrcholů najdeme. Předpokládejme, že máme sestavenou metriku (Definice 4.9), která je uložena do dvourozměrného pole `d[][]`. Excentricitu vrcholu $u$ určíme tak, že v řádku pole `d[][]`, který odpovídá vrcholu $u$, vybereme největší číslo. Všimněte si, že v nesouvislém grafu je excentricita každého vrcholu $\infty$.

Na straně 18 jsme definovali indukovaný podgraf na dané množině vrcholů.

Centrum obecného grafu je definováno jako podgraf indukovaný na množině vrcholů s nejmenší excentricitou. Dá se ukázat, že vezmeme-li strom, tak takto definované centrum grafu a centrum stromu, které získáme podle Definice 5.14, jsou totožné.

### Pojmy k zapamatování

— kořenový strom
— kořen, rodič a potomek
— centrum stromu
— uspořádaný kořenový (pěstovaný) strom

## 5.3 Isomorfismus stromů

### Průvodce studiem


*Centrum stromu i jednoznačné pořadí potomků kořenového stromu jsme zavedli, abychom mohli popsat algoritmus rozpoznávání isomorfních stromů. V této sekci ukážeme, jak každému stromu přiřadit jednoznačně kód a potom porovnáním kódů poznáme, které stromy jsou isomorfní a které ne.*

### Cíle

Po prostudování této sekce budete schopni:

- každému pěstovanému stromu (uspořádanému kořenovému stromu) přiřadit jeho kód,
- každému stromu přiřadit jednoznačně jeho kód,
- rozhodnout o isomorfismu daných stromů.

Pojem *isomorfismus stromů* je speciálním případem isomorfismu grafů. Dva stromy jsou isomorfní, pokud jsou isomorfní jako grafy.

Připomeňme, že pro úlohu rozhodnout, zda dva obecné grafy jsou isomorfní, není znám žádný rychlý algoritmus (s polynomiální složitostí). Stromy však tvoří natolik speciální třída grafů, že pro ně takový algoritmus existuje! Nyní si jej ukážeme. Nejprve zavedeme několik pojmů.

**Definice 5.18.** Dva kořenové stromy $(T, r)$ a $(T', r')$ jsou *isomorfní* pokud existuje isomorfismus mezi stromy $T$ a $T'$, který zobrazí kořen $r$ na kořen $r'$.

Na Obrázku 5.10 jsou dva stromy, které jsou isomorfní (zkuste isomorfismus najít), avšak jistě nejsou isomorfní jako kořenové stromy, neboť kořen stromu $T$ je stupně 2, zatímco kořen stromu $T'$ je stupně 3.


Obrázek 5.10 Isomorfní stromy, které nejsou isomorfní jako kořenové stromy.

Protože v algoritmu rozpoznávání stromů budeme pracovat s uspořádanými kořenovými stromy, bude důležité poznat, zda isomorfismus zachová také pořadí potomků. Proto zavedeme také pojem „isomorfismus pěstovaných stromů“.

**Definice 5.19.** Dva uspořádané kořenové stromy (pěstované stromy) jsou isomorfní, jestliže pro ně existuje isomorfismus kořenových stromů, který navíc zachová pořadí potomků každého vrcholu.

Stromy na Obrázku 5.10 nejsou isomorfní jako kořenové stromy a proto nejsou isomorfní ani jako uspořádané kořenové stromy. Na Obrázku 5.11 jsou dva stromy, které jsou isomorfní jako kořenové stromy, ale jistě nejsou isomorfní jako uspořádané kořenové stromy, neboť se liší pořadí potomků kořene.

Obrázek 5.11 Isomorfní kořenové stromy, které nejsou isomorfní jako uspořádané kořenové stromy.

### Kódování uspořádaných kořenových stromů

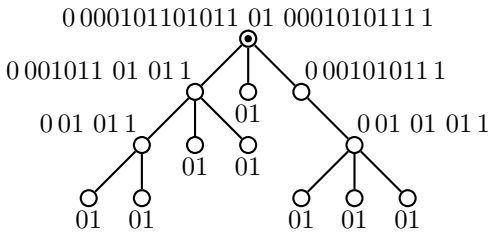
Ukážeme, že každému uspořádanému kořenovému stromu lze snadným postupem vhodně přiřadit řetězec, který jej jednoznačně popisuje (až na isomorfismus). Při popisu řetězce vystačíme se dvěma symboly, třeba 0 a 1 a proto se kódu někdy říká „binární kód stromu“.

Všimněte si, že odebráním nějakého vrcholu $u$ (a všech hran s ním incidentních) z netriviálního stromu $T$ vznikne les (graf bez cyklů). Tento les označíme $T - u$. Poznamenejme, že pokud odebraný vrchol byl listem stromu $T$, vznikne strom $T-u$. *Podstromem vrcholu* $u$ daného kořenového stromu $(T, r)$ rozumíme každou takovou komponentu grafu $T - u$, která obsahuje některého potomka $x$ vrcholu $u$. Snadno si rozmyslíme, že každý podstrom vrcholu $u$ je opět stromem, a můžeme jej považovat za kořenový strom s příslušným kořenem $x$.

**Definice 5.20.** *Kód* uspořádaného kořenového stromu se sestaví rekurzivně z kódů všech podstromů kořene, seřazených ve stejném pořadí jako jsou seřazeny kořeny podstromů (jeho potomci), a uzavřených do páru 0 a 1 (vizte Obrázek 5.12).

Všimněte si, že podle definice je kód koncového vrcholu roven „01“, neboť koncový vrchol nemá potomky (řetězec kódů podstromů je prázdný), pouze obsahuje pár symbolů 0 a 1.

Na Obrázku 5.12 jsme sestavili kód uspořádaného kořenového stromu. Nejprve jsme přiřadili kód „01“ koncovým vrcholům. Potom jsme přiřadili kódy jejich rodičům, potom prarodičům, atd. Jako poslední sestavíme kód kořene a jeho kód nazveme kódem celého uspořádaného kořenového stromu.




Obrázek 5.12 Kódování uspořádaného kořenového (pěstovaného) stromu.

**Poznámka 5.21.** Místo „0“ a „1“ lze použít i jiné symboly, třeba „(“ a „)“ nebo „A“ a „B“.

### Uspořádaný kořenový strom daný kódem

Ukázali jsme, jak každému stromu přiřadit kód. Nyní popíšeme opačný postup, jak sestavit nebo nakreslit uspořádaný kořenový strom, když máme dán jeho kód.

**Lemma 5.22.** *Máme dán kód $S$ uspořádaného kořenového stromu. Příslušný strom nakreslíme následujícím postupem:*

- *při přečtení prvního znaku „0“ na začátku položíme pero na papír a nakreslíme kořen,*
- *při každém dalším přečtení znaku „0“ nakreslíme hranu a nového následujícího potomka $x$ (kreslíme vždy napravo od případných sourozenců) současného vrcholu a přesuneme se do $x$,*

- *při každém přečtení znaku „1“ se vrátíme do rodiče současného vrcholu, případně ukončíme kreslení a zvedneme pero, pokud jsme v kořenu (a na konci kódu).*

Podobně, pokud bychom chtěli sestavit strom určený kódem $S$, snadno podle postupu v Lemmatu 5.22 najdeme množinu vrcholů i množinu hran takového stromu.

Všimněte si, že podle definice bude kód uspořádaného kořenového stromu obsahovat vždy stejný počet nul jako jedniček a sice tolik nul, kolik je vrcholů daného stromu. Máme-li nějaký strom (ne kořenový strom!) s alespoň třemi vrcholy, tak vždy můžeme zvolit kořen několika různými způsoby a často můžeme různě seřadit potomky vrcholů. To znamená, že dva isomorfní stromy mohou mít různé kódy. Dokonce dva stromy, které jsou isomorfní jako kořenové stromy, mohou mít různé kódy. Avšak dva stromy, které jsou isomorfní jako uspořádané kořenové stromy, budou mít vždy stejný kód. Toto pozorování shrneme do následující věty, jejíž důkaz nebudeme uvádět.

**Věta 5.23.** *Dva uspořádané kořenové (pěstované) stromy jsou isomorfní právě tehdy, když jejich kódy získané podle Definice 5.20 jsou shodné řetězce.*

Uvědomte si, že ne každá posloupnost 0 a 1 je kódem nějakého uspořádaného kořenového stromu. Je-li například počet jedniček a nul různý, nebo pokud posloupnost začíná jedničkou, tak se jistě nejedná o platný kód. Na druhou stranu každá neprázdná posloupnost stejného počtu jedniček a nul, kde každý počáteční úsek kódu obsahuje *méně* jedniček než nul, je platný kód nějakého uspořádaného kořenového stromu. Stejný počet jedniček a nul obsahuje teprve *celý* kód.

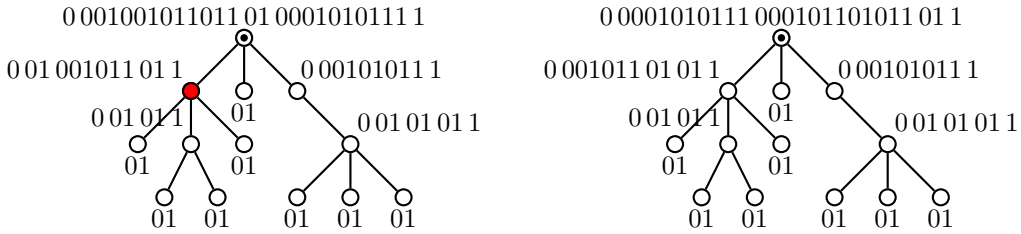
Z předchozího textu je zřejmé, že pokud chceme analogicky přiřadit nějaký kód jednoznačně každému stromu (nejen uspořádanému kořenovému stromu) musíme jednoznačně zvolit kořen a musíme umět jednoznačně seřadit potomky každého vrcholu. Už víme, že centrum stromu je určeno jednoznačně a proto budeme volit kořen v centru. Ještě si všimneme, že potomky vrcholu $u$ můžeme seřadit jednoznačně ve smyslu Definice 5.17 právě pomocí kódů podstromů vrcholu $u$. Kód pak sestavujeme opět rekurzivně dle Definice 5.20.

### Minimální kód stromu

Kódy můžeme chápat jako řetězce a ty umíme seřadit *jednoznačně*, například lexikograficky. Lexikografické uspořádání je „slovníkové uspořádání“. To znamená, že pro každou dvojici řetězců (v našem případě kódů) umíme rozhodnout, který kód by ve slovníku byl napsán dříve a který později. Předpokládáme, že znak 0 se ve slovníku nachází před znakem 1. Potom například řetězec 000111 by ve slovníku byl před řetězcem 001011, ale i před řetězcem 0011 a 01.

Jestliže při sestavení kódu pro každý vrchol nejprve seřadíme kódy jeho potomků lexikograficky, tak dostaneme jednoznačně určený kód kořenového (ne nutně uspořádaného) stromu, tzv. *minimální kód*.

Je dobré si uvědomit, že Definice 5.20 je formulována pouze pro *uspořádané* kořenové stromy, zatímco minimální kód můžeme sestavit pro kořenové stromy. Je proto nutno rozlišovat pojmy *kód uspořádaného kořenového stromu* $(T, r)$ a *minimální kód kořenového stromu* $(T, r)$. Když nakreslíme strom, který je určen kódem uspořádaného kořenového stromu $(T, r)$, dostaneme opět kořenový strom $(T, r)$. Pokud ale nakreslíme strom, který je určen minimálním kódem kořenového stromu, může se pořadí potomků oproti výchozímu kořenovému stromu $(T, r)$ lišit. Říkáme, že jsme kořenový strom $(T, r)$ „přepěstovali“ tak, aby jeho kód byl minimální a tím pádem jednoznačně určený.




Obrázek 5.13 Kód uspořádaného kořenového stromu a minimální kód kořenového stromu.

**Příklad 5.24.** Je některý z kódů na Obrázku 5.13 minimální?

*Řešení.* Při sestavení kódu červeně vybarveného vrcholu na Obrázku 5.13 vlevo bylo zvoleno takové pořadí kódů, jaké mají potomci červeně označeného vrcholu. Toto pořadí není lexikograficky nejmenší, neboť kód 0 0101 1 by musel předcházet kód 01. Dále ani kód kořene není lexikograficky nejmenší a proto kód na Obrázku 5.13 vlevo není minimální.

Avšak při sestavení minimálních kódů vrcholů na Obrázku 5.13 vpravo bylo zvoleno vždy takové pořadí kódů jeho potomků, že tyto kódy jsou při čtení zleva doprava seřazeny lexikograficky vzestupně. Proto je kód na Obrázku 5.13 vpravo minimální. ▲

Pro nalezení minimálního kódu použijeme následující funkci `minimalni_kod()`, která pro daný kořenový strom $X$ s kořenem $r$ rekurzivně sestaví (lexikograficky) minimální kód. Tuto funkci použijeme později v algoritmu rozpoznání isomorfismu stromů.

**Algoritmus 5.1 (Nalezení minimálního kódu kořenového stromu).**

```
// na vstupu je kořenový strom (případně podstrom)
vstup < kořenový strom (X,r);

funkce  minimalni_kod(strom X, vrchol r) {
   if (X má jeden vrchol)
      return "01";
   Y[1...d] = {podstromy po odebrání kořene r};
   s[1...d] = {kořeny podstromů Y[] v odpovídajícím pořadí};
               // kořeny jsou potomci kořene r
   for (i=1,...,d)
       k[i] = minimalni_kod(Y[i],s[i]);
   sort lexikograficky podle klíče k[1] <= k[2] <= ... <= k[d];
   return "0"+k[1]+...+k[d]+"1";
}
```

## Pro zájemce:

Počet různých uspořádaných kořenových stromů s daným počtem vrcholů je roven

$$C_n = \frac{1}{n+1}\binom{2n}{n}.$$

$C_n$ jsou tzv. *Catalanova čísla* (čti „Katalánova“). Catalanova čísla se objevují při řešení celé řady kombinatorických úloh, které obvykle popisují strukturu, která se skládá z menších struktur stejného typu – jako jsou například kořenový strom a jeho podstromy. Více se o Catalanových číslech můžete dozvědět například v [5].

## Pro zájemce:

Lexikografické uspořádání je relace uspořádání na množině všech řetězců, v našem případě kódů uspořádaných kořenových stromů. Jedná se o reflexivní, antisymetrickou a tranzitivní relaci, která je úplná, tj. pro každou dvojici řetězců $S_1$ a $S_2$ je buď $S_1 = S_2$, nebo $S_1 < S_2$, nebo $S_2 < S_1$. V řadě programovacích jazyků je tato relace implementována jako funkce pro porovnávání řetězců, pomocí které umíme rozhodnout, který řetězec nebo kód je „menší“.

Obecně bychom mohli pro nalezení jednoznačně určeného kódu zvolit jinou úplnou relaci uspořádání. Pro každé dva platné řetězce musíme umět rozhodnout, v jakém pořadí se v kódu objeví.

### Určování isomorfismu stromů

Jestliže budeme pro zadané dva stromy zjišťovat, zda jsou isomorfní, tak můžeme postupovat podle následujících kroků.

- Nejprve najdeme centrum každého stromu a v centru zvolíme kořen,
- pomocí Algoritmu 5.1 najdeme minimální kód každého kořenového stromu,
- porovnáním kódů rozhodneme podle Věty 5.23 o isomorfismu jednotlivých stromů.

Celý postup zformulujeme jako algoritmus, který zjistí, zda dané dva stromy $T$ a $U$ jsou isomorfní. Následující algoritmus využívá funkci `minimalni_kod(X,r)` z Algoritmu 5.1, která pro strom $X$ s kořenem $r$ najde (lexikograficky) minimální kód.

**Algoritmus 5.2 (Určení isomorfismu stromů).**

```
// Máme dva stromy U, T se stejným počtem vrcholů.
Vstup < stromy T a U;
for (X=T,U) {
   // určení center daných stromů U, T
   x = centrum(X);
   if (x je jeden vrchol)
      r = x;
   else
      do X přidej nový vrchol r, nahraď hranu x=uv hranami ru, rv;
   k[X] = minimalni_kod(X,r);
}
if ((|V(T)|==|V(U)|) && (k[T]==k[U] jako řetězce))
   vypiš("Stromy T, U jsou isomorfní.");
else
   vypiš("Stromy T, U nejsou isomorfní.");
exit;
```

Důkaz správnosti algoritmu vynecháme. Vyplývá z jednoznačnosti centra stromu a z jednoznačnosti minimálního kódu kořenového stromu.

Všimněte si, že v Algoritmu 5.2 kontrolujeme, zda oba stromy mají stejný počet vrcholů. Vskutku, pokud stromy mají různý počet vrcholů, jistě nejsou isomorfní. Na druhou stranu bez tohoto ověření by algoritmus mohl ve speciálních případech dát chybnou odpověď. Například cesty $P_{2n}$ a $P_{2n+1}$ jistě nejsou isomorfní, ale protože centrum cesty $P_{2n}$ je „prostřední“ hrana, tak v Algoritmu 5.2 bude na tuto hranu přidán nový vrchol a vznikne tak druhá cesta $P_{2n+1}$. Oba minimální kódy pak budou stejné a výsledkem algoritmu (bez testování rovnosti počtu vrcholů) by pak byl chybný závěr, že obě cesty $P_{2n}$ a $P_{2n+1}$ jsou isomorfní.

Pokud bychom měli stromů více, můžeme snadno upravit algoritmus tak, aby pro každý strom našel jeho minimální kód a za (po dvou) isomorfní označil ty skupiny stromů, které mají stejný minimální kód.

### Složitost Algoritmu 5.2

Složitost Algoritmu 5.2 závisí především na složitosti funkce `minimalni_kod()` v Algoritmu 5.1. Při detailnějším rozboru vidíme, že pro každý z $n$ vrcholů daného stromu $X$ bude na nějaké úrovni rekurze volána funkce `minimalni_kod()`. Můžeme pro jednoduchost předpokládat, že seřazení kódů vyžaduje polynomiální složitost $O(n^2)$ a toto seřazení bude prováděno pro každý vrchol stromu. Dostáváme, že horní odhad složitosti Algoritmu 5.1 pro nalezení minimálního kódu jednoho stromu je $O(n^3)$.

Doba běhu celého modifikovaného Algoritmu 5.2 pak pochopitelně závisí lineárně na počtu stromů, pro které minimální kód hledáme.

### Jiná kódování stromů

V praxi se setkáváme s celou řadou různých kódování stromů. Pro ukládání rozsáhlých dat se používá Huffmanův kód. Jedná se o metodu kódování znaků abecedy (podobně jako Morseova abeceda), kdy kódy přiřadíme znakům abecedy na základě binárního stromu (v binárním stromu má každý rodič nejvýše dva potomky), jehož koncové vrcholy odpovídají uloženým znakům a vzdálenost koncového vrcholu od kořene souvisí s relativní četností jednotlivých znaků.

Upozorňujeme, že Huffmanův kód je zcela odlišný kódu z Definice 5.20. Zatímco Hufmannův kód slouží k ukládání znaků rozsáhlého souboru dat, tak binární kódy zavedené v této kapitole slouží k ukládání uspořádaných kořenových stromů a porovnávání stromů. Neměli bychom různé kódy zaměňovat.

## Pojmy k zapamatování

— kód uspořádaného kořenového stromu
— minimální kód kořenového stromu
— isomorfismus stromů

## 5.4 Kostra grafu


## Průvodce studiem

*V této sekci se budeme věnovat jednomu ze základních problémů teorie grafů – hledání minimální kostry. Už víme, že strom je minimální souvislý graf na daném počtu vrcholů (Věta 5.10). V celé řadě praktických aplikací hraje takový souvislý podgraf (budeme mu říkat kostra grafu) důležitou roli při hledání optimálních řešení. Hledání kostry se tak objevuje jako část jiných algoritmů, například v Christofidově algoritmu pro heuristické řešení úlohy obchodního cestujícího se nejprve hledá minimální kostra daného grafu.*

*Při řešení elektrických obvodů (výpočet proudů v jednotlivých větvích a určení napětí mezi uzly) s využitím Kirchoffových zákonů je nutno sestavit dostatečný počet lineárně*

*nezávislých rovnic. Najdeme-li kostru grafu elektrické sítě, pak existuje poměrně jednoduchý postup jak tyto rovnice sestavit.*

## Cíle

Po prostudování této sekce budete schopni:

- vysvětlit pojem kostry grafu,
- najít kostru grafu pomocí hladového algoritmu,
- najít kostru grafu pomocí Jarníkova algoritmu.

Připomeňme, že faktor grafu $G$ je takový podgraf, který obsahuje všechny vrcholy grafu $G$ (Definice 1.28). Graf nazýváme ohodnocený, jestliže každé hraně přiřadíme reálné číslo (tzv. *ohodnocení hrany* nebo *váha hrany*), a (kladně) vážený, pokud je ohodnocení každé hrany kladné číslo.

**Definice 5.25.** *Kostrou* souvislého grafu $G$ rozumíme takový faktor grafu $G$, který je stromem. *Váhou kostry* ohodnoceného grafu $G$ rozumíme součet vah všech hran této kostry.

V této kapitole preferujeme termín „ohodnocení hrany“ před termínem „váha hrany“, abychom lépe odlišili váhu kostry a váhu hrany.

Význam „koster“ spočívá v jejich minimalitě vzhledem k počtu hran, přičemž současně je zachována souvislost grafu (Věta 5.10). Pokud je ohodnocení každé hrany stejné, bude mít každá kostra grafu stejnou váhu. Jestliže se ale ohodnocení jednotlivých hran liší, mohou mít různé kostry téhož grafu různou váhu. Naším úkolem bude najít takovou kostru, jejíž váha je nejmenší možná.

**Definice 5.26 (Problém minimální kostry (MST)).** Je dán souvislý ohodnocený graf $G$ s nezáporným ohodnocením hran $w : E(G) \to \mathbb{R}_0^+$. *Problém minimální kostry* znamená najít takovou kostru $T$ v grafu $G$, která má nejmenší možnou váhu. Formálně

$$MST = \min_{\text{kostra} T \subseteq G} \left( \sum_{e \in E(T)} w(e) \right).$$

MST (z anglického „Minimum spanning tree“) je číslo, které udává váhu kostry s nejmenší možnou vahou.

## Kontrolní otázky

1. Má smysl hledat minimální kostru v grafu se záporným ohodnocením hran? Je každý souvislý faktor (podgraf se všemi vrcholy grafu) s minimálním ohodnocením kostrou takového grafu?
2. Kolik hran má kostra souvislého grafu na $n$ vrcholech?

Je známa celá řada algoritmů pro nalezení minimální kostry daného nezáporně ohodnoceného grafu. My uvedeme několik z nich. Při řešení praktické úlohy můžeme vybrat nejvhodnější z nich v závislosti na způsobu uložení dat.

První z nich popíšeme neformálně (upozorňujeme, že zatímco obvykle symbolem $T$ označujeme strom, tak v následujícím algoritmu symbolem $T$ značíme množinu hran):

**Algoritmus 5.3 (Hladový pro minimální kostru).** *Mějme dán souvislý ohodnocený graf $G$ s nezáporným ohodnocením hran $w$. Počet hran grafu $G$ označíme $m$.*

- *Seřadíme hrany grafu $G$ vzestupně podle jejich ohodnocení:*

$$w(e_1) \leqq w(e_2) \leqq \cdots \leqq w(e_m).$$

- *Začneme s prázdnou množinou hran $T = \emptyset$ pro kostru.*
- *Pro $i = 1, 2, \ldots, m$ vezmeme hranu $e_i$ a pokud přidáním této hrany nevznikne cyklus (v grafu s množinou hran $T \cup \{e_i\}$), tak přidáme hranu $e_i$ do $T$.*

  *Jinak hranu $e_i$ „zahodíme“.*
- *Po zpracování všech hran obsahuje $T$ hrany minimální kostry váženého grafu $G$.*

Výstupem algoritmu je množina $T$, která obsahuje právě hrany minimální kostry. Kostra je pak tvořena všemi vrcholy daného grafu a množinou hran $T$.

**Poznámka 5.27.** Algoritmu se říká „hladový“, což je překlad anglického výrazu „greedy“. Výstižnější by možná byl termín „hamoun“, protože algoritmus se snaží získat v každém kroku co nejvýhodnější kus – v našem případě hranu s co nejmenší váhou. Hladový algoritmus neřeší strategii, zda by nebylo výhodnější spokojit se v nějakém kroku s horším prvkem (hranou s větší vahou) za cenu toho, že v pozdějších krocích budeme moci nevýhodu vyvážit mnohem výhodnějšími prvky, než které získá hladový algoritmus.

Uvědomte si, že algoritmus nikdy nevybírá z více možností a nezkouší různé varianty a výsledky mezi sebou porovnat. Úloha hledání minimální kostry spadá mezi několik málo problémů, při jejichž řešení hladový postup vždy vede k optimálnímu řešení. V dalších kapitolách ukážeme několik problémů, při jejichž řešení by podobný algoritmus selhal.

Nyní ukážeme, že Algoritmus 5.3 funguje správně.

**Věta 5.28.** *Algoritmus 5.3 najde minimální kostru souvislého grafu.*

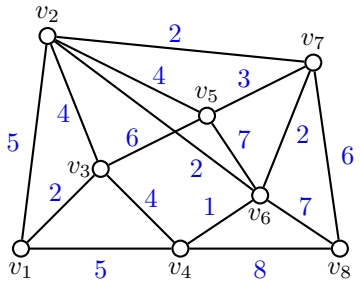
*Důkaz.* Správnost Algoritmu 5.3 ukážeme sporem.

Mějme souvislý graf $G$ s ohodnocením hran $w$. Nechť $T$ je množina hran získaná v průběhu Algoritmu 5.3. Předpokládejme, že hrany jsou již seřazené podle váhy $w(e_1) \leqq w(e_2) \leqq \cdots \leqq w(e_m)$. Minimálních koster daného grafu může existovat se stejnou vahou více. Označme $T_0$ množinu hran takové minimální kostry, která se s množinou $T$ shoduje na co nejvíce prvních hranách. Pokud $T_0 = T$, algoritmus pracoval správně.

Předpokládejme pro spor, že algoritmus nenašel minimální kostru a tedy že $T_0 \neq \neq T$. Označme $j > 0$ takový index, že se množiny $T_0$ a $T$ shodují na prvních $j-1$ hranách $e_1, e_2, \ldots, e_{j-1}$, ale neshodují se na hraně $e_j$. To znamená, že $e_j \in T$ a přitom $e_j \notin T_0$, neboť podle Algoritmu 5.3 vybíráme hrany postupně a hrana $e_j$ nemůže tvořit cyklus s předchozími hranami, neboť patří do množiny hran minimální kostry $T_0$.

Přidáme-li hranu $e_j$ do kostry s množinou hran $T_0$, vznikne podle Důsledku 5.8 graf s právě jedním cyklem $C$. Cyklus $C$ však nemůže být obsažen v nalezené kostře $T$, a proto existuje v cyklu $C$ hrana $e_k$, která do množiny $T$ nepatří, přičemž dle volby $j$ víme, že $k > j$.

Potom je však podle našeho seřazení hran je $w(e_k) \geqq w(e_j)$, a proto kostra na hranách $T' = \big(T_0 \setminus \{e_k\}\big) \cup \{e_j\}$ (kostra vznikne z minimální kostry nahrazením hrany $e_k$ hranou $e_j$) nemá vyšší ohodnocení (není horší) než kostra s hranami $T_0$. Množina hran $T'$ této minimální kostry se však shoduje s množinou hran $T$ na více hranách než $T_0$! To je spor s volbou $T_0$ a proto případ $T_0 \neq T$ nemůže nastat. To znamená, že $T_0 = T$ a algoritmus pracuje správně. □




Obrázek 5.14 Kladně ohodnocený graf $G$.

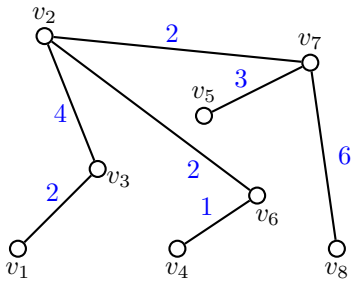
**Příklad 5.29.** Užitím hladového algoritmu (Algoritmus 5.3) najděte minimální kostru grafu $G$ na Obrázku 5.14.

*Řešení.* Podle Algoritmu 5.3 nejprve seřadíme hrany podle jejich ohodnocení do neklesající posloupnosti. Dostaneme následující posloupnost. Hrana s ohodnocením 1: $v_4v_6$, hrany s ohodnocením 2: $v_1v_3$, $v_2v_6$, $v_2v_7$, $v_6v_7$, hrana s ohodnocením 3: $v_5v_7$, hrany s ohodnocením 4: $v_2v_3$, $v_2v_5$, $v_3v_4$, hrany s ohodnocením 5: $v_1v_2$, $v_1v_4$, hrany s ohodnocením 6: $v_3v_5$, $v_7v_8$, hrany s ohodnocením 7: $v_5v_6$, $v_6v_8$, a hrana s ohodnocením 8: $v_4v_8$.

Nyní budeme postupně přidávat hrany do množiny $T$ tak, aby tyto hrany (spolu s koncovými vrcholy) netvořily žádný cyklus. Můžeme přidat první čtyři hrany $v_4v_8$, $v_1v_3$, $v_2v_6$, $v_2v_7$, ale pátou hranu $v_6v_7$ (délky 2) přidat nemůžeme, protože by spolu s již přidanými hranami $v_2v_6$ a $v_2v_7$ (a s příslušnými vrcholy) vytvořila cyklus $C_3$.

Potom přidáme další dvě hrany $v_5v_7$ a $v_2v_3$, ale osmou hranu $v_2v_5$ nemůžeme přidat, protože by vznikl cyklus na vrcholech $v_2, v_5, v_7$. Ani devátou hranu $v_3v_4$ nemůžeme přidat, protože by vznikl cyklus na vrcholech $v_2, v_3, v_4, v_6$.

Stejně tak přidáním hran $v_1v_2$, $v_1v_4$ nebo $v_3v_5$ by vznikl cyklus, proto je také zahodíme. Konečně přidáním hrany $v_7v_8$ vznikne kostra (podle Věty 5.4 víme, že vznikla kostra, neboť graf $G$ má osm vrcholů a v množině $T$ je sedm hran). Přidáním každé ze zbývajících hran by vznikl cyklus, proto algoritmus končí. Na Obrázku 5.15 je minimální kostra grafu $G$. Váha kostry je $1 + 2 + 2 + 2 + 3 + 4 + 6 = 20$.




Obrázek 5.15 Minimální kostra grafu $G$ s váhou 20.

▲

Zmíněný hladový algoritmus pro hledání minimální kostry grafu byl popsán poprvé užitím teorie grafů Kruskalem (1956). Je však známo, že Kruskal vycházel z práce českého matematika Otakara Borůvky. Už v roce 1926 řešil brněnský akademik Otakar Borůvka úlohu optimální stavby elektrické sítě na jižní Moravě a popsal velmi podobný algoritmus. Při formulaci ani v důkazu správnosti však nepoužil teorii grafů, ale maticovou algebru.

**Algoritmus 5.4 (Borůvkův algoritmus pro minimální kostru).** *Mějme souvislý (kladně) vážený graf $G$ s ohodnocením hran $w$ různými čísly. Na začátku seřadíme hrany vzestupně podle jejich ohodnocení $w(e_1) < w(e_2) < \ldots < w(e_m)$. Kostru začneme sestavovat tak, že přidáme hranu $e_i$ (pro $i = 1, 2, \ldots, n$), pokud přidáním nevznikne cyklus.*

Všimněte si, že v algoritmu se předpokládá, že žádné dvě hrany nemají stejnou váhu. Jako reakce na Borůvkovu práci vypracoval Vojtěch Jarník v roce 1929 podobný algoritmus. Jarníkův algoritmus je ve světě známý jako Primův algoritmus z roku 1957.

**Algoritmus 5.5 (Jarníkův algoritmus pro minimální kostru).** *Mějme souvislý graf $G$ s $n$ vrcholy a s nezáporným ohodnocením hran $w$. Kostru začneme sestavovat z jednoho (libovolného) vrcholu. V každém kroku algoritmu přidáme takovou hran, která má nejmenší ohodnocení mezi hranami, která vedou z již vytvořeného podstromu do některého ze zbývajících vrcholů grafu $G$. Po $n-1$ krocích algoritmus končí.*

Všimněte si, že v Jarníkově algoritmu 5.5 hrany na začátku neseřazujeme. Navíc není třeba testovat vznik cyklu, neboť v každém kroku přidáváme do rozrůstajícího se stromu vrchol, který je v tuto chvíli listem. To je výhodné, neboť testování existence cyklu v grafu je výpočetně náročné.

Ukázky běhu hladového (Kruskalova) a Jarníkova (Primova) algoritmu najdete na adrese http://mi21.vsb.cz/sites/mi21.vsb.cz/files/unit/minimalni_kostra.pdf.

### Pojmy k zapamatování

— kostra grafu
— hladový algoritmus

### Pro zájemce:

## 5.5 Bludiště

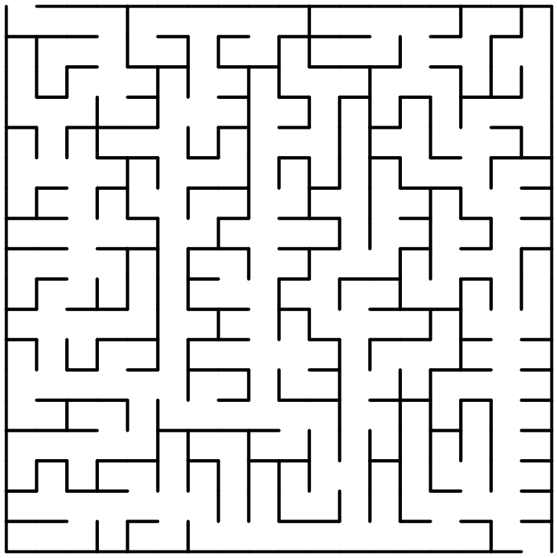
Máte rádi bludiště? A jak souvisí bludiště a kostry grafu? Ukážeme, jak můžeme pomocí algoritmů pro hledání minimální kostry snadno sestavit bludiště.

Bludiště na Obrázku 5.16 vzniklo při běhu Jarníkova (neboli Primova) algoritmu, který hledal minimální kostru v nějakém grafu. Jedná se vlastně o kostru grafu pravidelné čtvercové sítě: vrcholy odpovídají políčkům na čtverečkovaném papíře a náhodně ohodnocené hrany spojují sousední políčka. Na stránce http://en.wikipedia.org/wiki/File:MAZE_30x20_Prim.ogv najdete animaci, která zachycuje proces vzniku bludiště.

Uvědomte si, že bludiště, jehož chodby odpovídají hranám nějaké kostry, má pěkné vlastnosti, které od bludiště očekáváme: Je souvislé (protože kostra je souvislý podgraf) a proto v bludišti nejsou izolované nebo oddělené části, které nemohou přispět k „bloudění“. Dále bludiště neobsahuje cykly (protože kostra neobsahuje cykly) a proto mezi každými dvěma místy v bludišti vede právě jedna cesta, kterou je nutno objevit. Pokud bychom však chtěli vícenásobné cesty umožnit, můžeme vždy některé stěny v bludišti vymazat (neboli přidat náhodné hrany do nalezené kostry).

**Příklad 5.30.** Sestavíme jednoduché bludiště ve čtvercové síti na Obrázku 5.17.

*Řešení.* Bludiště zkonstruujeme podle postupu zmíněného dříve. Nejprve sestavíme graf vhodné struktury tak, že do každého čtverečku umístíme vrchol a dva vrcholy spojíme hranou, jestliže mezi odpovídajícími políčky sítě může být průchod. Tento graf náhodně ohodnotíme čísly z předem zvoleného intervalu, například celými čísly 1 až 9. Dostaneme



Obrázek 5.16 Bludiště sestavené pomocí Jarníkova algoritmu.

graf, který je na Obrázku 5.18 vlevo. V grafu najdeme minimální kostru (Obrázek 5.18 vpravo).

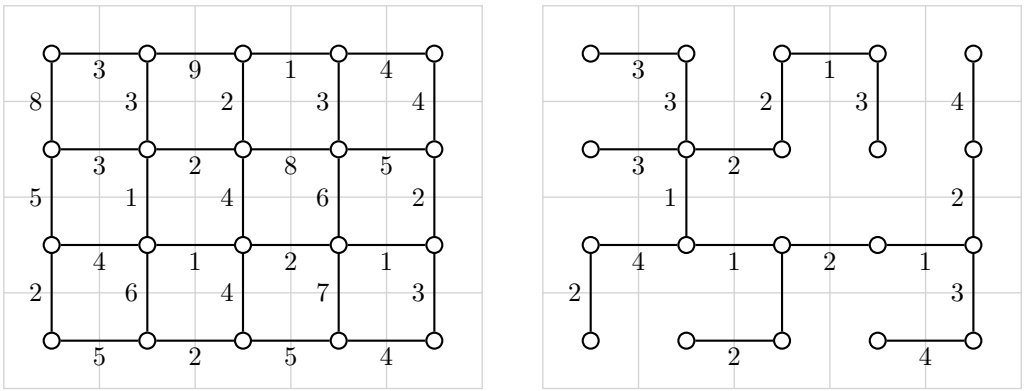
Hrany kostry znázorňují cesty v bludišti. Příslušné přepážky čtvercové sítě odstraníme a dostaneme bludiště, ve kterém je každé místo dosažitelné z libovolného výchozího místa po právě jedné cestě (Obrázek 5.19). ▲

## Kontrolní otázky

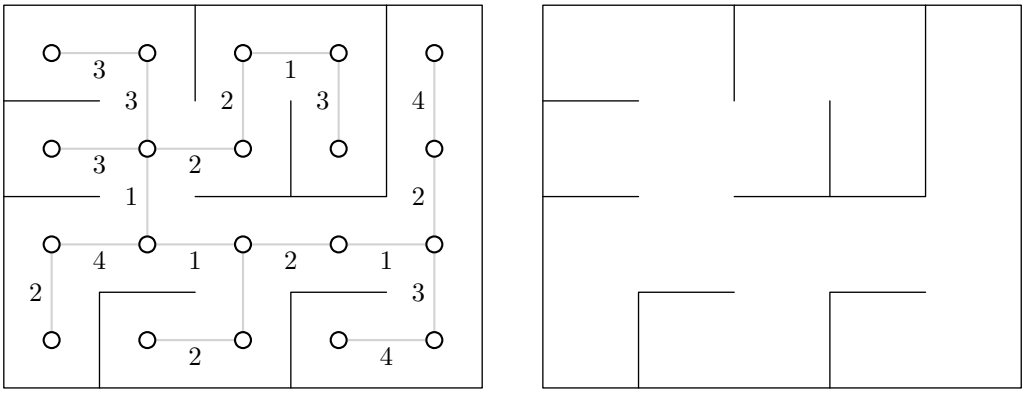
1. Bludiště na Obrázku 5.16 vzniklo při hledání kostry Jarníkovým (Primovým) algoritmem. Lze pro sestavení bludiště použít i hladový algoritmus? Jak se budou obě bludiště lišit?
2. Jak se liší bludiště získané jako minimální kostra nějakého ohodnoceného grafu a bludiště, jehož sousední políčka jsou spojena náhodně?

Poznamenejme ještě, že pro generování podobných bludišť můžeme použít i algoritmus pro prohledávání do hloubky. Buď budeme v každé iteraci volit pořadí sousedů k prozkoumání podle náhodného ohodnocení nebo vezmeme neohodnocený graf a v každé iteraci budeme vybírat náhodně další vrchol mezi dosud nezpracovanými sousedními vrcholy v úschovně. Animaci takového postupu najdete na adrese `http://commons.wikimedia.org/wiki/File:MAZE_30x20_DFS.ogv`. Pokud použijeme modifikaci algoritmu prohledávání do šířky, tak obvykle *nedostaneme* pěkné bludiště. Zkuste si rozmyslet proč.

Obrázek 5.17 Čtvercová síť, ve které ukážeme jednoduchou konstrukci bludiště.




Obrázek 5.18 Graf $G$ sestrojený podle struktury místností a jeho kostra.




Obrázek 5.19 Bludiště sestavené na základě kostry grafu $G$.

# Kapitola 6

# Barevnost a kreslení grafů

## Průvodce studiem

*V této kapitole se budeme věnovat dvěma tématům – vrcholovému barvení grafů a rovinnému kreslení grafů. Nejprve na příkladech vysvětlíme, jak barvení vrcholů grafu různými barvami slouží k modelování vztahů mezi objekty. Barvy reprezentují vlastnosti, přičemž samotné barvy nejsou důležité, slouží jen pro snadnou představu a výstižné formulace. Ukážeme, jak barvení grafů může pomoci optimalizovat některé úlohy plánování.*

*Poté zavedeme příslušné základní pojmy a uvedeme několik odhadů počtu nutného barev pro dobré grafové barvení.*

*Dále ukážeme, jak ověřit, zda je graf rovinný, tj. zda je možno jej nakreslit do roviny tak, aby se žádné hrany neprotínaly. Rovinné kreslení grafů najde své uplatnění například při výrobě tištěných spojů.*

*V poslední části kapitoly se budeme věnovat barvení vrcholů rovinných grafů a zmíníme pěkné aplikace, které s barvením rovinných grafů souvisí.*

## 6.1 Barevnost grafů

Zmíníme dvě úlohy, které lze snadno přeformulovat jako úlohu vrcholového barvení grafu.

### Skladovací problém

Ve skladu je uloženo mnoho druhů potravin. Podle předpisů některé druhy potravin musí být umístěny v oddělených prostorách. Například ovocné saláty nesmí být skladovány společně s čerstvými syrovými vejci nebo krájené salámy nesmí být skladovány společně se syrovým masem. Jaký je nejmenší počet oddělených místností, který ve skladu potřebujeme?

Pro řešení úlohy použijeme následující model. Sestavíme graf, jehož vrcholy budou odpovídat ukládaným potravinám (či jiným komoditám), přičemž hranou spojíme dva vrcholy, pokud odpovídající komodity nesmí být skladovány současně.

Jednotlivé místnosti budeme odlišovat barvami. Nyní můžeme skladovací problém přeformulovat takto: jaký je nejmenší počet různých barev potřebný k takovému obarvení vrcholů grafu, aby žádné dva sousední vrcholy nebyly obarveny stejně?

### Optimalizace křižovatek

Křižovatka má různé jízdní pruhy (koridory), jak pro auta, tak pro chodce. Doprava v koridorech, které se nekříží, může probíhat současně. Naopak koridory, které se kříží, musí „mít zelenou“ v jiných časových intervalech. Jaký je nejmenší počet časových intervalů v jednom „cyklu“ řízení semaforů, kdy každý koridor měl alespoň jedenkrát zelenou?

Úlohu budeme modelovat grafem, jehož vrcholy odpovídají dopravním koridorům a každé dva koridory, které spolu kolidují, spojíme hranou. Barvy vrcholů budou odpovídat různým časovým intervalům jednoho „cyklu“ semaforů. Opět se budeme ptát na nejmenší počet barev nutný k takovému obarvení vrcholů grafu, kdy žádné dva vrcholy spojené hranou nemají stejnou barvu.

## Cíle

Po prostudování této sekce budete schopni:

- vysvětlit význam barvení grafů,
- převést praktický problém na úlohu barvení grafu,
- poznat grafy s malou barevností.

Nejprve zavedeme pojem obarvení grafu a barevnost grafu. Potom uvedeme několik základních pozorování a tvrzení a ukážeme, jak řešit úlohu barvení grafu pro některé speciální případy. V dalším textu budeme často barvy vrcholů označovat přirozenými čísly, což je při formálním popisu přehlednější. V obrázcích a komentářích budeme naopak pracovat s barvami, jelikož to bude názornější.

**Definice 6.1.** *Obarvení grafu $G$ pomocí $k$ barev* je takové zobrazení

$$c : V(G) \to \{1, 2, \ldots, k\},$$

ve kterém každé dva vrcholy, které jsou spojené hranou, budou mít různou barvu, tj. $c(u) \neq c(v)$ pro každou hranu $uv \in E(G)$.
Uvedenému obarvení vrcholů grafu se říká také *dobré vrcholové barvení grafu*.

Samozřejmě můžeme každý graf obarvit pomocí tolika barev, kolik má graf vrcholů – každý vrchol jednoduše dostane jinou barvu. Nás však zajímá co možná *nejmenší* počet barev, pro které existuje dobré vrcholové barvení grafu $G$. V praktických úlohách odpovídá počet barev nákladům nebo požadavkům, které se snažíme minimalizovat.

**Definice 6.2.** *Barevnost* grafu $G$ je nejmenší přirozené číslo $\chi(G)$, pro které existuje obarvení grafu $G$ pomocí $\chi(G)$ barev.

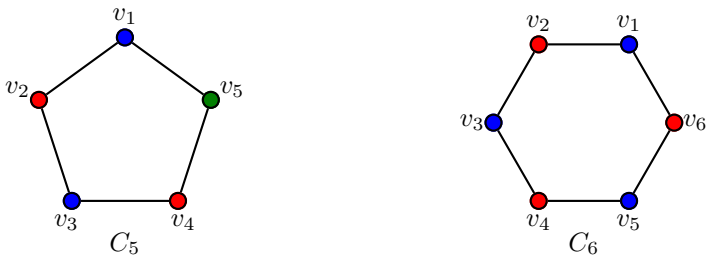
Protože nás zajímá nejmenší počet barev pro dobré obarvení daného grafu, tak zavedeme následující přirozenou úmluvu: v množině čísel, která reprezentují použité barvy, nebudou zbytečně vynechaná čísla. Nejvyšší číslo použité barvy bude proto odpovídat barevnosti grafu.

Barevnosti grafu $G$ se v některé literatuře říká také *chromatické číslo* grafu $G$.

**Příklad 6.3.** Určete barevnost cyklu $C_5$ a cyklu $C_6$.

*Řešení.* Nejprve určíme barevnost cyklu $C_5$. Vrcholy cyklu $C_5$ označíme po řadě $v_1, v_2, v_3, v_4, v_5$. Protože $v_1$ a $v_2$ jsou spojeny hranou, potřebujeme na obarvení grafu $C_5$ alespoň dvě barvy. Zkusíme celý graf obarvit dvěma barvami. Bez újmy na obecnosti obarvíme vrchol $v_1$ první barvou (například modře) a vrchol $v_2$ druhou barvou (například červeně). Potom však vrchol $v_3$ sousedí s červeně obarveným vrcholem $v_2$, proto obarvíme vrchol $v_3$ modrou barvou a to si zase vynutí obarvení vrcholu $v_4$ červenou barvou (Obrázek 6.1 vlevo).

Poslední neobarvený vrchol $v_5$ je sousední s modře obarveným vrcholem $v_1$ a červeně obarveným vrcholem $v_4$. Po obarvení prvního vrcholu $v_1$ jsme v žádném kroku už neměli možnost volby, proto dvě barvy na dobré vrcholové barvení cyklu $C_5$ nestačí. Na obarvení vrcholu $v_5$ musíme použít další barvu. Na Obrázku 6.1 je cyklus $C_5$ dobře obarven třemi barvami a proto barevnost cyklu $C_5$ je $\chi(C_5) = 3$. Analogicky bychom zdůvodnili, že barevnost každého lichého cyklu (cyklu liché délky) je 3.




Obrázek 6.1 Dobré vrcholové barvení cyklů $C_5$ a $C_6$.

Nyní určíme barevnost cyklu $C_6$. Označíme vrcholy podobně jako v cyklu $C_5$. Ihned je zřejmé, že na dobré vrcholové barvení potřebujeme alespoň dvě barvy. Na Obrázku 6.1 vpravo vidíme, že dvě barvy stačí, proto barevnost cyklu $C_6$ je $\chi(C_6) = 2$. Podobně můžeme zdůvodnit, že barevnost každého sudého cyklu (cyklu sudé délky) je 2. ▲

### Horní odhad počtu barev

Je zřejmé, že na obarvení daného grafu nemůžeme použít více barev, než je vrcholů v grafu. Ukážeme, že barevnost grafu $G$ se rovná počtu jeho vrcholů právě tehdy, když $G$ je kompletní graf.

**Lemma 6.4.** *Pro každý graf $G$ na $n$ vrcholech platí $\chi(G) \leqq n$. Rovnost nastává právě tehdy, když $G$ je úplný graf.*

*Důkaz.* Pro každý graf $G$ na $n$ vrcholech stačí každý vrchol obarvit jinou barvou a máme dobré vrcholové obarvení grafu $G$ $n$ barvami. To znamená, že $\chi(G) = \chi(K_n) \leqq \leqq n$.

Je-li $G \simeq K_n$, tak žádné dva vrcholy nemohou být obarveny stejnou barvou, protože každé dva vrcholy jsou sousední. Proto $\chi(K_n) = n$.

Pokud ale graf $G$ není kompletní a některá hrana $uv$ v grafu $G$ chybí, můžeme oba její koncové vrcholy obarvit stejnou barvou $c(u) = c(v) = 1$ a zbývající vrcholy obarvíme různými barvami $2, 3, \ldots, n-1$. Dostaneme tak obarvení grafu $G$ méně než $n$ barvami a proto platí $\chi(G) < n$. Tím je tvrzení dokázáno. □

Už jsme ukázali, že $\chi(C_5) = 3 = \Delta(C_5) + 1$ a $\chi(K_n) = n = \Delta(K_n) + 1$. Toto pozorování je možno upřesnit a dá se ukázat, že jsou to právě jen liché cykly a kompletní grafy, pro které platí $\chi(G) = \Delta(G) + 1$.

Dokonce je možno dokázat, že na dobré vrcholové obarvení každého grafu kromě kompletních grafů a lichých cyklů stačí nejvýše tolik barev, jaký je největší stupeň v grafu.

**Věta 6.5 (Brooksova věta).** *Pro každý graf $G$ na $n$ vrcholech různý od $K_n$ a lichých cyklů $C_n$ platí $\chi(G) \leqq \Delta(G)$.*

Důkaz tohoto tvrzení přesahuje rámec našeho kurzu, zájemci jej mohou najít třeba v [5].

Samozřejmě ne v každém grafu musíme použít tolik barev, jaký je nejvyšší stupeň vrcholů v grafu. Například na dobré vrcholové obarvení (kompletních) bipartitních grafů stačí dvě barvy (Obrázek 6.2).


Obrázek 6.2 Dobré vrcholové barvení grafu $K_{4,6}$ a stromu $T$.

Jak obecně pro daný graf $G$ určit chromatické číslo $\chi(G)$ a jak najít dobré vrcholové barvení pomocí $\chi(G)$ barev? Algoritmy pro nalezení takového barvení jsou komplikované a nejsou součástí tohoto textu. Jednoduchou heuristiku, která však nemusí dát vždy obarvení pomocí nejmenšího možného počtu barev najdete v [5]. Upozorněme ještě, že není znám algoritmus s polynomiální složitostí, který by uměl najít dobré vrcholové barvení s nejmenším počtem barev pro libovolný graf. Pro obecný graf existuje algoritmus složitosti $O(n \cdot 2^n)$, kde $n$ je počet vrcholů. V praxi se obvykle využívají heuristické algoritmy jako Brelazova heuristika nebo hladová heuristika.

### Dolní odhad počtu barev

Brooksova věta (čti „brůksova“) říká, kolik *nejvíce* barev může být potřeba na dobré vrcholové obarvení daného grafu. Nyní ukážeme několik jednoduchých odhadů, kolik barev je pro daný graf potřeba *nejméně*.

**Věta 6.6.** *Graf $G$ má barevnost* 1 *právě tehdy, když nemá žádné hrany.*

*Důkaz.* Důkaz tvrzení je snadný. Pokud graf nemá hrany, obarvíme všechny vrcholy barvou 1. A mají-li všechny vrcholy stejnou barvu, nemůže být v grafu žádná hrana. □

S jedinou barvou vystačíme na dobré vrcholové barvení pouze v grafu bez hran. Grafy, na jejichž dobré obarvení potřebujeme dvě barvy, už jsou zajímavější.

**Věta 6.7.** *Graf $G$ má barevnost* 2 *právě tehdy, když neobsahuje jako podgraf žádný cyklus liché délky.*

*Důkaz.* Pečlivý důkaz vyžaduje technicky komplikovaný rozbor grafu, proto zde naznačíme hlavní myšlenku důkazu. Omezíme se na souvislé grafy, pro grafy s více komponentami využijeme platnost tvrzení pro každou komponentu.

Už víme, že lichý cyklus nelze dobře obarvit dvěma barvami (Příklad 6.3). Tím jsme nepřímo zdůvodnili, že graf s barevností 2 nemůže obsahovat lichý cyklus.

A naopak, zvolíme libovolný vrchol $v$ v grafu $G$ a obarvíme jej barvou 1. Vrcholy, jejichž vzdálenost od $v$ je lichá, obarvíme barvou 2 a vrcholy, jejichž vzdálenost od $v$ je sudá, obarvíme barvou 1.

Označme $w$ poslední společný vrchol na nějakých nejkratších cestách z $v$ do $x$ a z $v$ do $y$. Pokud bychom získali dva vrcholy $x$, $y$ v sudé vzdálenosti od vrcholu $v$ (a tedy i navzájem stejné vzdálenosti od vrcholu $w$) spojené hranou $xy$, tak spojením cesty z $y$ do $w$, cesty z $w$ do $x$ a hrany $xy$ by vznikl uzavřený sled liché délky. Tento sled je současně cyklem (proč?) liché délky, což podle předpokladu není možné.

Pro dva vrcholy v liché vzdálenosti je zdůvodnění podobné. Proto navržené obarvení je dobré (neobarví sousední vrcholy stejnou barvou) a dvě barvy na obarvení grafu $G$ stačí. □

Netriviální grafy, které neobsahují cykly liché délky, jsou bipartitní (Obrázek 6.2). To znamená, že jejich vrcholy umíme rozdělit do dvou množin (partit) tak, že v rámci každé partity není mezi vrcholy žádná hrana. Potom stačí vrcholy v jedné partitě obarvit jednou barvou a vrcholy v druhé partitě obarvit druhou barvou.

Bez důkazu uvedeme ještě jeden snadný dolní odhad barevnosti daného grafu.

**Věta 6.8.** *Jestliže v grafu $G$ je kompletní podgraf na $k$ vrcholech, tak na dobré obarvení (celého) grafu $G$ je potřeba alespoň $k$ barev.*

### Kontrolní otázky

1. Existuje graf, jehož barevnost je menší než největší stupeň vrcholů v grafu?
2. Existuje graf, jehož barevnost je menší než nejmenší stupeň vrcholů v grafu?
3. Existuje graf, jehož barevnost je vyšší než největší stupeň vrcholů v grafu?

### Pojmy k zapamatování

— skladovací problém
— dobré vrcholové barvení
— barevnost (chromatické číslo) grafu
— Brooksova věta

## 6.2 Rovinná nakreslení grafů

### Průvodce studiem


*Už v Kapitole 1 jsme zavedli pojem nakreslení grafu. Dosud však nehrálo roli, jak daný graf nakreslíme. Pro některé aplikace je však zásadní, zda se nám podaří nakreslit graf tak, aby se hrany neprotínaly. Například schémata elektrických obvodů můžeme chápat jako grafy a při návrhu jednovrstevného tištěného spoje je žádoucí, aby se ve schématu nacházelo co nejméně křížení, protože křížení je nutno přemostit.*

*V této a příští sekci si zavedeme pojem tzv. „rovinného grafu" a ukážeme si, jak poznat grafy, které lze nakreslit v rovině bez křížení hran.*

### Cíle

Po prostudování této sekce budete schopni:

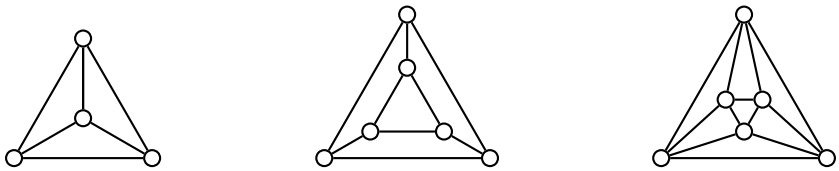
- popsat rovinný graf a vysvětlit jeho význam,
- vysvětlit rozdíl mezi rovinným a nerovinným grafem,
- určit počet oblastí grafu,
- použít Eulerův vzorec pro určení některých parametrů rovinného grafu,
- pro některé (husté) grafy poznat, že nejsou rovinné.

Nejprve vyslovíme definici rovinného grafu.

**Definice 6.9.** *Rovinným nakreslení* grafu $G$ je takové nakreslení grafu, ve kterém jsou vrcholy znázorněny jako různé body v rovině a hrany jako křivky spojující tyto body, které odpovídají koncovým vrcholům. Přitom žádná hrana se nesmí křížit ani procházet jinými body, než které odpovídají jejím koncovým vrcholům. Řekneme, že graf je *rovinný* pokud máme jeho rovinné nakreslení.

Grafům, pro které existuje rovinné nakreslení, se někdy říká *planární* grafy.

Na Obrázku 6.3 jsou příklady rovinných grafů. Pěknými příklady rovinných grafů jsou grafy mnohostěnů (graf čtyřstěnu, graf krychle, graf osmistěnu, graf dvanáctistěnu, hranoly, ...).



Obrázek 6.3 Rovinné grafy (graf čtyřstěnu, graf trojbokého hranolu a graf osmistěnu.

Bohužel, ne každý graf má rovinné nakreslení.

**Příklad 6.10.** Existuje rovinné nakreslení a) grafu $K_5$, b) grafu $K_5$ s jednou odebranou hranou?

*Řešení.* a) Na Obrázku 6.4 vlevo je obvyklé nakreslení grafu $K_5$, ve kterém je celkem pět křížení hran. Po několika pokusech se nám může podařit najít nakreslení s jediným křížením hran (Obrázek 6.4 vpravo). Dále v textu na straně 118 dokážeme, že žádné nakreslení grafu $K_5$ v rovině nemůže být bez křížení hran. Graf $K_5$ proto nemá rovinné nakreslení.

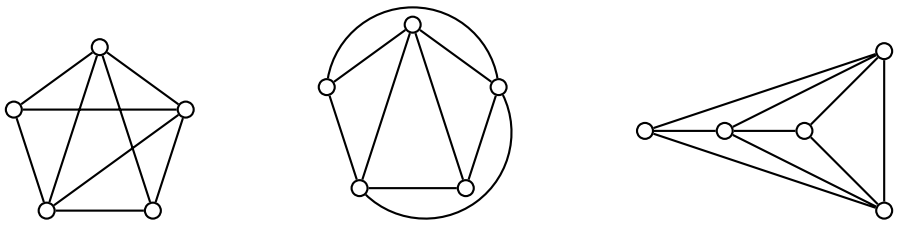
Obrázek 6.4 Graf $K_5$ a jeho nakreslení s jediným křížením hran.

b) Pro graf $K_5$ bez jedné hrany existuje jeho rovinné nakreslení. Na Obrázku 6.5 vlevo je nakreslení grafu $K_5$ bez jedné hrany. Na Obrázku 6.5 uprostřed je nakreslení stejného grafu, přičemž žádné dvě hrany se nekříží. Na Obrázku 6.5 vpravo je jiné nakreslení téhož grafu, přičemž hrany jsou nakresleny jako úsečky.

▲

Dá se ukázat, že grafy mnohostěnů jsou vždy rovinné a (alespoň) 3-souvislé. A naopak také platí, že každý rovinný 3-souvislý jednoduchý graf je grafem nějakého mnohostěnu.

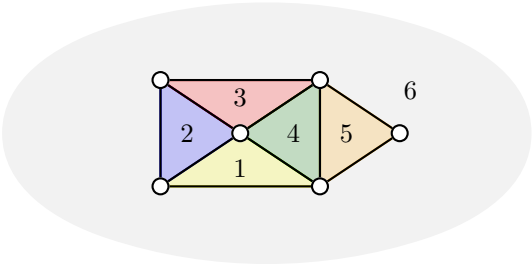
V rovinném nakreslení má smysl zkoumat nejen vrcholy a hrany, ale také oblasti, na které rovinu rozdělí dané nakreslení grafu.



Obrázek 6.5 Graf $K_5$ bez jedné hrany a dvě jeho rovinná nakreslení.

**Definice 6.11.** *Oblastmi* rovinného nakreslení grafu nazýváme souvislé oblasti roviny ohraničené nakreslením grafu.

Oblastem se někdy říká *stěny* a v anglické literatuře se nazývají „faces.“ Při počítání oblastí nezapomeňme na „vnější“ oblast, neboť i ta je součástí roviny (Obrázek 6.6). Například grafy na Obrázku 6.3 mají postupně 4, 5 a 8 oblastí. Pokud se na Obrázek 6.3 podíváme jako na nakreslení jediného grafu se třemi komponentami, má nakreslení celkem 16 oblastí. Uvědomte si, že oblast grafu je ohraničena cyklem, případně jedním nebo více uzavřenými sledy, pokud jsou v grafu mosty nebo pokud graf není souvislý (most je taková hrana souvislého grafu, jejímž odebráním se graf stane nesouvislý).




Obrázek 6.6 Oblasti rovinného grafu.

**Poznámka 6.12.** Pojem souvislé oblasti v rovině se pečlivě zavádí například v matematické analýze. Zde vystačíme s intuitivní představou takové části roviny, ve které každé dva body lze spojit křivkou, jejíž každý bod také leží v dané oblasti.

Už Leonhard Euler si kolem roku 1750 všiml, že mezi počtem vrcholů, počtem hran a počtem oblastí rovinného grafu je jednoznačný vztah.

**Věta 6.13 (Eulerův vzorec).** *Mějme jednoduchý souvislý rovinný graf s $f$ oblastmi, $v$ vrcholy a $h$ hranami. Potom platí*

$$v + f - h = 2.$$

*Důkaz.* Důkaz povedeme indukcí vzhledem k počtu hran $h$.
*Základ indukce:* Nejmenší souvislý graf pro pevně zvolený počet vrcholů $v$ je podle Věty 5.10 strom. Protože strom neobsahuje cyklus, tak nakreslení grafu má jedinou oblast, a sice vnější oblast. Podle Věty 5.4 víme, že takový graf má $h = v - 1$ hran a snadno ověříme, že platí $v + f - h = v + 1 - (v - 1) = 2$.
*Indukční krok:* Předpokládejme, že tvrzení platí pro všechny souvislé grafy, které mají $h - 1$ hran. Pokud souvislý graf není stromem, tak podle Důsledku 5.8 jistě obsahuje alespoň jeden cyklus $C$. Po vynechání jedné hrany $e$ cyklu $C$ zůstane výsledný graf souvislý a počet hran sníží o 1. Zároveň se změní počet oblastí, protože hrana $e$ cyklu $C$ oddělovala dvě oblasti (vnitřní a vnější oblast cyklu $C$, které sousedily společnou hranou $e$). Vynecháním hrany $e$ tyto oblasti splynou v jedinou oblast a proto se počet oblastí oproti původnímu nakreslení grafu sníží také o 1. Počet vrcholů se přitom nezmění.

Podle indukčního předpokladu v menším grafu, který vznikne odebráním hrany $e$, platí $v + (f - 1) - (h - 1) = 2$. Odtud ihned vidíme, že platí také $v + f - h = 2$. □

Všimněte si, že Eulerův vzorec nezávisí na zvoleném rovinném nakreslení grafu, pouze na struktuře grafu. To znamená, že každé rovinné nakreslení grafu má stejný počet oblastí, tento počet je jednoznačně určen počtem vrcholů a hran rovinného grafu.

Ačkoli je vztah poměrně jednoduchý, má mnoho aplikací a důsledků. Některé důsledky zmíníme a dokonce dokážeme.

**Důsledek 6.14.** *Jednoduchý rovinný graf na $v$ vrcholech, kde $v \geqq 3$, má nejvýše $3v - 6$ hran.*

*Důkaz.* Ukážeme, že nerovnost platí v každé komponentě rovinného grafu, která má alespoň tři vrcholy. Není těžké si rozmyslet, že pro stromy a pro v grafy s menšími komponentami tvrzení bude platit také a proto můžeme předpokládat, že daný graf $G$ je souvislý a obsahuje nějaký cyklus. Označíme $v$ počet vrcholů v grafu $G$, $f$ počet oblastí a $h$ počet hran.

Protože jednoduchý graf neobsahuje smyčky ani násobné hrany, tak každá vnitřní oblast je (v libovolném nakreslení grafu $G$) ohraničena alespoň třemi hranami. Budeme-li počítat hrany na hranici každé oblasti, započítáme každou hranu nejvýše dvakrát (ve dvou přilehlých oblastech). Platí tedy $2h \geqq 3f$, neboli $\frac{2}{3}h \geqq f$. Dosazením do Eulerova vztahu dostaneme

$$2 = v + f - h \leqq v + \frac{2}{3}h - h = v - \frac{1}{3}h,$$

odkud snadno vyjádříme horní odhad počtu hran

$$h \leqq 3(v - 2) = 3v - 6,$$

což je dokazované tvrzení. □

Je zajímavé srovnat horní odhad počtu hran $3v - 6$ s počtem hran $v(v-1)/2$ kompletního grafu na $v$. Věta 6.14 říká, že rovinný graf nemůže obsahovat „mnoho hran“, ani ne trojnásobek počtu vrcholů. Velký kompletní graf proto má mnohem více hran než rovinný graf se stejným počtem vrcholů.

Pokud v rovinném grafu nejsou žádné cykly $C_3$ (říkáme jim trojúhelníky), tak graf musí obsahovat ještě méně hran, ani ne dvojnásobek počtu vrcholů.

**Důsledek 6.15.** *Jednoduchý rovinný graf bez trojúhelníků $C_3$ na $v$ vrcholech, kde $v \geqq 3$, má nejvýše $2v - 4$ hran.*

*Důkaz.* Tvrzení dokážeme obdobně jako předchozí důsledek. Označme $v$ počet vrcholů v grafu $G$, $f$ počet oblastí a $h$ počet hran. Tentokrát víme, že graf nemá ani trojúhelníky $C_3$, a proto je každá vnitřní oblast v libovolném nakreslení grafu ohraničena alespoň čtyřmi hranami. Opět, budeme-li počítat hrany na hranici každé oblasti, započítáme každou hranu nejvýše dvakrát. Platí tedy $2h \geqq 4f$, neboli $\frac{2}{4}h \geqq f$. Dosazením do Eulerova vztahu dostaneme

$$2 = v + f - h \leqq v + \frac{2}{4}h - h = v - \frac{1}{2}h,$$

odkud snadno dostaneme horní odhad počtu hran v grafech bez trojúhelníků

$$h \leqq 2(v-2) = 2v - 4,$$

což je dokazované tvrzení. □

Příkladem grafů bez trojúhelníků jsou bipartitní grafy. Rovinné bipartitní grafy jsou velmi řídké – mají málo hran.

Díky Eulerovu vzorci můžeme dokonce ohraničit shora nejnižší stupeň rovinného grafu!

**Důsledek 6.16.** *Každý rovinný graf obsahuje vrchol stupně nejvýše* 5. *Každý rovinný graf bez trojúhelníků obsahuje vrchol stupně nejvýše* 3.

*Důkaz.* Postupujme sporem. Pro grafy s méně než třemi vrcholy tvrzení jistě platí. Pokud by všechny vrcholy byly stupně alespoň 6, celý graf by měl podle Principu sudosti 1.15 alespoň $\frac{1}{2} \cdot 6v = 3v$ hran, což je ve sporu s Důsledkem 6.14. Proto musí mít některý vrchol stupeň menší než 6.

Podobně, pokud by všechny vrcholy v grafu bez trojúhelníků byly pro spor stupně 4 nebo většího, tak celý graf by měl podle Principu sudosti 1.15 alespoň $\frac{1}{2} \cdot 4v = 2v$ hran, což je ve sporu s Důsledkem 6.15. Proto musí mít některý vrchol stupeň menší než 4. □

Všimněte si, že rovinný graf sice *může* obsahovat vrcholy vysokých stupňů, avšak současně musí vždy obsahovat nějaký vrchol malého stupně. Je dokonce možno ukázat, že takových vrcholů s malým stupněm musí být v grafu několik.

### Pojmy k zapamatování

— rovinný graf
— oblast grafu
— Eulerův vzorec

## 6.3 Rozpoznání rovinných grafů


### Průvodce studiem

*„Být rovinný“, respektive „nebýt rovinný“ je důležitou vlastností grafu. Nejenže nakreslení grafu bez křížení hran je přehlednější, ale (jak jsme zmínili úvodu) často najdeme praktické uplatnění. Při výrobě tištěných spojů pro schémata, kterým odpovídá rovinný graf, vystačíme s jednovrstevnými tištěnými spoji.*

*Nyní ukážeme, jak pro daný graf rozhodnout, zda existuje jeho rovinné nakreslení nebo ne. Pochopitelně nemá smysl zkoušet náhodně nakreslit obecný graf bez křížení hran, protože různých nakreslení stejného grafu existuje nekonečně mnoho.*

*Ukážeme, že dobrým vodítkem při určování rovinnosti grafu je Eulerův vzorec, respektive jeho důsledky uvedené v předchozí sekci.*

### Cíle

Po prostudování této sekce budete schopni:

- ukázat, že dva významné grafy $K_5$ a $K_{3,3}$ nejsou rovinné,
- vysvětlit pojem rozdělení grafu,
- použít Kuratowského větu pro určení, zda daný graf je rovinný.

Na rozdíl od určení barevnosti grafu, je určení rovinnosti grafu relativně rychle algoritmicky řešitelné. My se zaměříme pouze na případy malých grafů, protože popis zmíněných algoritmů přesahuje rámec kurzu.

Nejprve rozebereme dva důležité grafy, které *nejsou* rovinné.

Obrázek 6.7 Grafy $K_5$ a $K_{3,3}$

**Příklad 6.17.** Ukažte, že grafy $K_5$ a $K_{3,3}$ nejsou rovinné.

*Řešení.* Všimněme si, že graf $K_5$ má 5 vrcholů a 10 hran. Ale podle Důsledku 6.14 má rovinný graf na pěti vrcholech nejvýše $3 \cdot 5 - 6 = 9$ hran. Proto není graf $K_5$ rovinný.

Podobně graf $K_{3,3}$ má 6 vrcholů a 9 hran a navíc neobsahuje žádné trojúhelníky. Ale podle Důsledku 6.15 má rovinný graf bez trojúhelníků na šesti vrcholech nejvýše $2 \cdot 6 - 4 = 8$ hran. Proto ani graf $K_{3,3}$ není rovinný. ▲

Výsledek příkladu zformulujeme jako další důsledek Eulerova vzorce 6.13.

**Důsledek 6.18.** *Grafy $K_5$ a $K_{3,3}$ nejsou rovinné.*

Ukazuje se, že oba grafy $K_5$ a $K_{3,3}$ mají klíčové postavení. Jejich struktura neumožňuje rovinné nakreslení, naproti tomu žádná další principiálně odlišná struktura neexistuje. Jak za chvíli uvedeme, jestliže graf není rovinný, obsahuje vždy jednu z uvedených struktur.

Abychom „nerovinnou strukturu“ popsali, zavedeme pojem *rozdělení* grafu. Rozdělení grafu je graf, který má oproti původnímu grafu přidané vrcholy stupně 2 „navlečené jako korálky“ na některé původní hrany.

**Definice 6.19.** *Rozdělením* grafu $G$ rozumíme graf, který vznikne z grafu $G$ (případným) nahrazením některých hran cestami (přidáním nových vrcholů stupně 2).

Rozdělení grafu se říká také *subdivize*. Samotný graf $G$ můžeme také považovat za rozdělení grafu $G$, protože tak žádná hrana nebyla nahrazena cestou.

Neformálně můžeme říci, že v nakreslení grafu smíme na hrany přidávat nové vrcholy, přitom ale nesmíme přidávat vrcholy v místě křížení hran, protože nově přidaný vrchol by nebyl stupně 2. Také pochopitelně nesmíme přidávat cesty tam, kde v původním grafu nebyla hrana.

Mějme graf $G$, přidáme nový vrchol $w$ a odebereme například hranu $uv$ grafu $G$ a nahradíme ji dvojicí hran $uw$ a $wv$. Dostaneme nový graf $G'$, který je *rozdělením* původního grafu $G$ (Obrázek 6.8). Symbolicky můžeme zapsat, že rozdělení $G'$ grafu $G$ je pro $w \notin V(G)$

$$G' = (V(G) \cup \{w\}, (E(G) \setminus \{uv\}) \cup \{uw, wv\}).$$


Obrázek 6.8 Graf $G$ s vyznačenou hranou $uv$ a rozdělení $G'$ grafu $G$.

Kazimierz Kuratowski v roce 1930 ukázal, že platí následující překvapivě jednoduché tvrzení.

**Věta 6.20.** *Graf $G$ je rovinný právě tehdy, když neobsahuje jako podgrafy rozdělení grafů $K_5$ nebo $K_{3,3}$.*

Důkaz nebudeme uvádět, protože je komplikovaný. Příklady rozdělení zakázaných podgrafů jsou na Obrázku 6.9. Takových zakázaných podgrafů sice existuje nekonečně mnoho, všechny však vycházejí ze struktury grafů $K_5$ a $K_{3,3}$.

Obrázek 6.9 Příklady rozdělení grafů $K_5$ a $K_{3,3}$.

**Příklad 6.21.** Je Petersenův graf na Obrázku 1.9 rovinný?

*Řešení.* Po chvíli zkoušení zjistíme, že se nám nedaří nakreslit Petersenův graf bez křížení hran. Zkusíme ukázat, že není rovinný.

Petersenův graf má 10 vrcholů a 15 hran. Podle Důsledku 6.14 víme, že rovinný graf na 10 vrcholech může mít nejvýše $3 \cdot 10 - 6 = 24$ hran. To Petersenův graf splňuje, proto podle Důsledku 6.14 nelze o rovinnosti nebo nerovinnosti rozhodnout.

Všimneme si, že Petersenův graf neobsahuje trojúhelníky. Podle Důsledku 6.15 víme, že rovinný graf bez trojúhelníků na 10 vrcholech může mít nejvýše $2 \cdot 10 - 4 = 16$ hran. To Petersenův graf splňuje, proto podle Důsledku 6.15 nelze o rovinnosti nebo nerovinnosti rozhodnout.

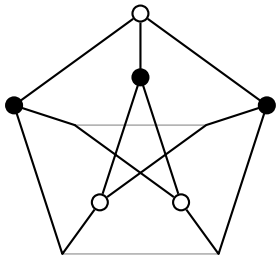
Dále Petersenův graf jistě neobsahuje rozdělení grafu $K_5$, protože nemá pět vrcholů stupně 4 nebo většího.

Při pečlivém zkoumání si všimneme, že nejkratší cyklus v Petersenově grafu má délku 5. Proto Petersenův graf neobsahuje $K_{3,3}$ jako podgraf, protože $K_{3,3}$ obsahuje cyklus $C_4$.

Petersenův graf však obsahuje rozdělení grafu $K_{3,3}$, proto není rovinný (Obrázek 6.10). ▲

**Příklad 6.22.** Existuje nějaký rovinný graf s 21 hranami a a 16 oblastmi?

*Řešení.* Ukážeme, že takový graf neexistuje. Podle Eulerova vzorce 6.13 by takový graf $G$ měl $v = 2 + h - f = 2 + 21 - 16 = 7$ vrcholů. Podle Principu sudosti 1.15 by graf $G$ měl součet stupňů roven 42 a protože žádný vrchol nemůže být stupně většího než 6, musel by graf $G$ být kompletním grafem $K_7$. Ten však není podle Kuratowského věty 6.20 rovinný, neboť obsahuje podgraf $K_5$. ▲



Obrázek 6.10 Petersenův graf obsahuje rozdělení grafu $K_{3,3}$ jako podgraf.

### Pro zájemce:

Víme, že rovinné grafy je možno nakreslit bez křížení hran. Dá je možno ukázat, že rovinné grafy je možno nakreslit dokonce tak „pěkně“, že všechny hrany můžeme nakreslit podle pravítka jako úsečky.

**Věta 6.23.** *Každý (jednoduchý) rovinný graf lze nakreslit v rovině bez křížení hran tak, že hrany jsou úsečky.*

### Pojmy k zapamatování

— rozdělení grafu
— Kuratowského věta

## 6.4 Barvení map a rovinných grafů

### Průvodce studiem


*Jeden z nejznámějších problémů teorie grafů je problém čtyř barev. Dnes bychom měli říkat „věta o čtyřech barvách,“ neboť zmíněný problém už byl vyřešen. Jeho formulace je sice jednoduchá, ale řešení si vyžádalo více než 100 let.*

#### Problém čtyř barev

*Stačí čtyři barvy na obarvení politické mapy tak, aby sousední státy nebyly obarveny stejnou barvou? Za sousední přitom považujeme ty státy, které mají společný úsek hranice, nikoli pouze bod.*

*Úplné řešení daného problému si vyžádalo, kromě důkazů celé řady teoretických tvrzení, také vyšetření velkého množství případů na počítači.*

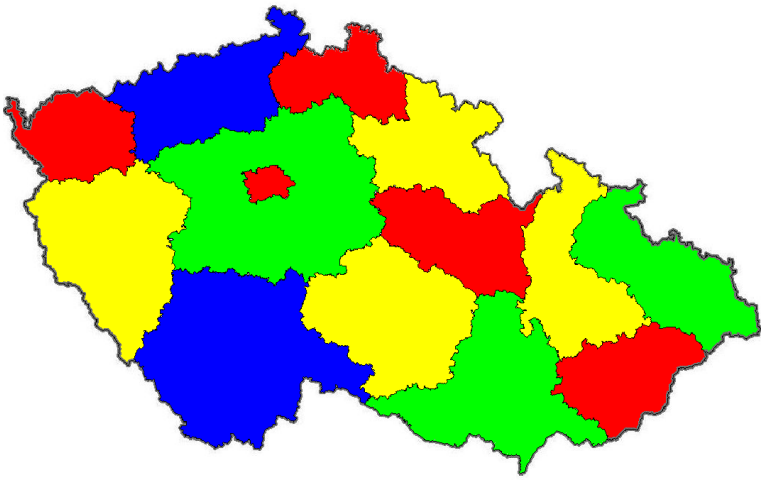
*V této sekci zavedeme několik pojmů a ukážeme některá jednodušší tvrzení a speciální případy.*

## Cíle

Po prostudování této sekce budete schopni:

- vysvětlit jak souvisí barvení politických map a barvení grafů,
- najít dobré vrcholové barvení rovinného grafu nejvýše šesti barvami,
- najít dobré vrcholové barvení rovinného grafu bez trojúhelníků nejvýše čtyřmi barvami.

Obarvení politické mapy snadno převedeme na barvení grafu. Vezměme například mapu krajů naší republiky (Obrázek 6.11). Každou oblast nahradíme vrcholem (hlavním nebo krajským městem) a dva vrcholy spojíme hranou, jestliže jsou odpovídající oblasti nebo státy sousední. Obarvení oblastí mapy tak převedeme na hledání dobrého vrcholového barvení odpovídajícího rovinného grafu.



Obrázek 6.11 Obarvení krajů republiky čtyřmi barvami.

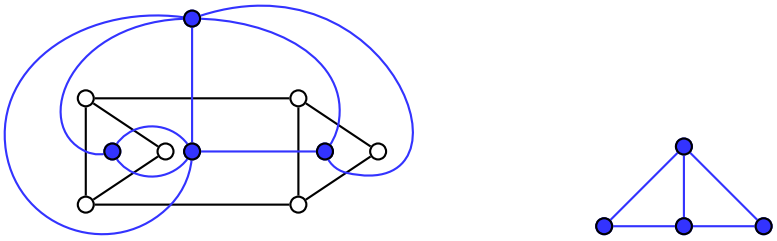
## Pro zájemce:

Druhou možností, jak sestavit graf $G$ dané úlohy, je označit hranice států za hrany a místa křížení za vrcholy. Dostaneme rovinný graf $G$. Potom budeme hledat barvení *oblastí* grafu $G$, ve kterém žádné dvě sousední oblasti nemají stejnou barvu. K tomuto grafu $G$ pak sestavíme tzv. „duální graf.“

**Definice 6.24.** *Duální graf* rovinného nakreslení grafu $G$ získáme, když každou oblast nahradíme vrcholem. Dva vrcholy nového grafu spojíme hranou, jestliže odpovídající dvojice oblastí sousedí hranou grafu $G$. *Duální multigraf* se definuje analogicky.

Na Obrázku 6.12 vlevo je rovinný graf $G$ a modře vyznačený duální *multigraf* $G'$. Duální multigraf obecně může obsahovat násobné hrany a smyčky. Pokud však v duálním multigrafu $G'$ odstraníme násobné hrany (násobné hrany odpovídají cestám v grafu $G$,

jejichž vnitřní vrcholy mají stupeň 2) a smyčky (smyčky odpovídají hranám incidentním s vrcholy stupně 1), tak dostaneme jednoduchý graf (Obrázek 6.12 vpravo). Dá se ukázat, že tento duální multigraf $G'$ i zjednodušený duální graf k rovinnému grafu jsou opět rovinné grafy (existuje jejich nakreslení do roviny bez křížení hran).



Obrázek 6.12 Graf $G$ s modře vyznačeným duálním multigrafem a překreslený duální graf.

Duální graf je pro dané nakreslení grafu určen (až na isomorfismus) jednoznačně, ale pro různá nakreslení téhož grafu můžeme dostat různé neisomorfní duální grafy.

### Věta o 4 barvách

Máme tedy rovinný graf $G$ a hledáme jeho dobré vrcholové barvení. Kolik nejméně barev potřebujeme? Tuto otázku si položil Francis Guthrie už v roce 1852. Všiml si, že při obarvování map Anglie vystačí se čtyřmi barvami. Během následujících let se o důkaz, že čtyři barvy stačí, pokoušela řada matematiků. Až v roce 1976 Appel a Haken (a později v roce 1993 jiným způsobem Robertson, Seymour, Sanders a Thomas), dokázali větu, která rozřešila problém čtyř barev. Jedná se o jeden z nejslavnějších výsledků diskrétní matematiky:

**Věta 6.25 (Věta o čtyřech barvách).** *Každý rovinný graf lze obarvit nejvýše čtyřmi barvami.*

Důkaz je velmi rozsáhlý, vydal by na samostatnou knihu, a proto jej vynecháme.

Není však těžké ukázat jednodušší tvrzení, že pro obarvení rovinného grafu stačí 6 barev. Také pro rovinné grafy bez trojúhelníků větu o čtyřech barvách dokážeme snadno.

**Věta 6.26.** *Každý rovinný graf lze obarvit nejvýše šesti barvami. Každý rovinný graf bez trojúhelníků lze obarvit nejvýše čtyřmi barvami.*

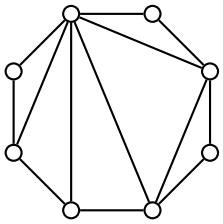
*Důkaz.* Tvrzení ukážeme indukcí vzhledem k počtu vrcholů.
*Základ indukce:* Graf s 1 vrcholem je jistě rovinný a na jeho dobré vrcholové barvení stačí jedna barva.
*Indukční krok:* Mějme graf s alespoň dvěma vrcholy. A předpokládejme, že pro všechny menší rovinné grafy tvrzení platí.

Podle Důsledku 6.16 najdeme v grafu $G$ vrchol $v$ stupně nejvýše 5. Graf $G'$, který vznikne z grafu $G$ odebráním vrcholu $v$ a všech hran s ním incidentních, je opět jednoduchý rovinný graf bez smyček. Podle indukčního předpokladu lze tento menší graf obarvit šesti barvami. Z nich jen nejvýše pět bude použito na obarvení sousedů vrcholu $v$ a tak šestou nepoužitou barvu můžeme *vždy* použít na obarvení vrcholu $v$. Tím dostaneme dobré vrcholové barvení grafu $G$.

Druhá část se dokáže analogicky s využitím druhé části Důsledku 6.16. □

Všimněte si, že důkaz Věty 6.26 je konstruktivní. Dává jednoduchý návod, jak najít dobré vrcholové obarvení rovinného grafu nejvýše šesti barvami. Podle Věty 6.25 však víme, že budou stačit čtyři barvy. Bohužel optimální algoritmus barvení je komplikovaný a přesahuje rámec našeho textu. Algoritmus naznačený v důkazu věty 6.26 nemusí dát barvení s nejmenším možným počtem barev.

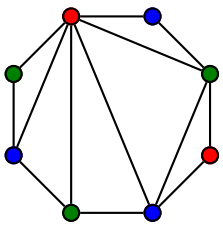
**Příklad 6.27.** Určete barevnost rovinného grafu na Obrázku 6.13.



Obrázek 6.13 Graf $G$

*Řešení.* Graf $G$ je rovinný, proto podle Věty 6.25 na jeho obarvení potřebujeme nejvýše čtyři barvy. Proto $\chi(G) \leqq 4$. Na druhou stranu snadno najdeme v grafu $G$ podgraf $K_3$, proto potřebujeme podle Věty 6.8 na obarvení grafu $G$ alespoň tři barvy: $\chi(G) \geqq 3$. Nyní víme $3 \leqq \chi(G) \leqq 4$. Protože na Obrázku 6.14 se nám podařilo najít obarvení grafu $G$ třemi barvami, tak platí $\chi(G) = 3$. ▲

Podobně jako v řešení Příkladu 6.27 můžeme postupovat i při určování barevnosti jiných grafů.



Obrázek 6.14 Obarvení grafu $G$ třemi barvami.

$\sum$

## Pojmy k zapamatování

— věta o čtyřech barvách

— barvení oblastí, případně duální graf

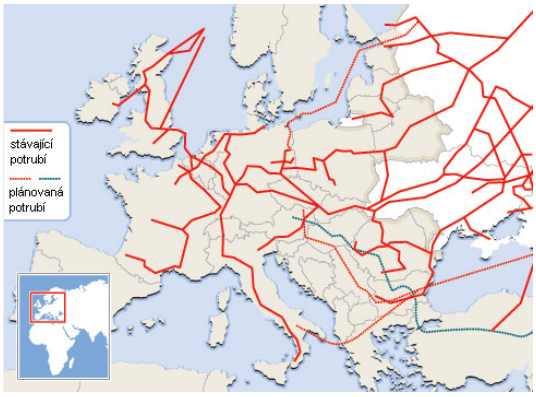
# Kapitola 7

# Toky v sítích

## Průvodce studiem

*Teorie grafů hraje klíčovou roli při řešení řady síťových úloh. Máme dánu síť (počítačovou síť, produktovod, a podobně), kde hrany reprezentují spojení a vrcholy jsou například křižovatky nebo switche.*

*Je přirozené, že hrany a vrcholy mají omezenou kapacitu, která je dána nějakým číslem (a často fyzikální jednotkou). Hlavní otázka zní: Jaký největší objem látky nebo dat můžeme přepravit po síti s danými omezeními z výchozího vrcholu $z$ do cílového vrcholu $s$. Této úloze se říká hledání největšího toku v síti.*




Obrázek 7.1 Trasy evropských ropovodů a plynovodů.

*Na Obrázku 7.1 je mapa ropovodů a plynovodů v Evropě, stav v roce 2009. V posledních letech se často řešila otázka, zda je možno dopravit dostatečný objem plynu z míst těžby na východě a na severu do míst spotřeby ve střední Evropě.*

*V této kapitole zavedeme nutné pojmy a zavedeme teoretické nástroje k tomu, abychom uměli největší tok v dané síti najít.*

# 7.1 Definice sítě

## Cíle

Po prostudování této kapitoly budete schopni:

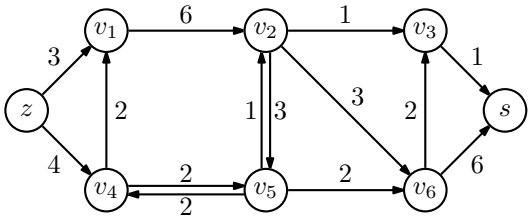
- popsat následující pojmy: síť, kapacita hrany a tok v síti,
- vysvětlit a pro konkrétní síť ověřit platnost zákona kontinuity,
- vysvětlit rozdíl mezi běžným uzlem sítě, zdrojem a stokem v síti,
- sestavit graf sítě odpovídající reálné úloze.

Na straně 5 jsme definovali orientovaný graf. Připomeňme, že množina (orientovaných) hran $E$ v orientovaném grafu obsahuje *uspořádané* dvojice vrcholů. Pro orientovanou hranu $e = (u, v)$ rozlišujeme počáteční vrchol $u$ a koncový vrchol $v$.

Pod pojmem síť budeme rozumět orientovaný graf, ve kterém máme dva význačné vrcholy (zdroj a stok) a ohodnocené hrany (stanovíme kapacity hran).

**Definice 7.1.** *Síť* je čtveřice $S = (G, z, s, w)$, kde $G$ je orientovaný graf. Vrcholy $z \in V(G)$, $s \in V(G)$ budeme nazývat *zdroj* a *stok* v síti $S$. Funkce $w : E(G) \to \mathbb{R}^+$ je kladné ohodnocení hran, které každé hraně přiřadí tzv. *kapacitu* hrany.

Někdy budeme pro zjednodušení říkat síť $G$, přičemž zdroj stok i kapacity hran budou známy z kontextu. Příklad sítě je na Obrázku 7.2.




Obrázek 7.2 Síť $(G, z, s, w)$.

Uvědomte si, že kapacita hrany nemusí nutně odpovídat fyzikální kapacitě. Číslo přiřazené hraně může reprezentovat šířku vozovky, maximální počet automobilů, který cestou projede za jednu minutu, tloušťku potrubí, počet přenesených megabytů za vteřinu, odpor vodiče a podobně. Přesto se při popisu se budeme držet terminologie, která vychází z dopravy tekutin. Umožní to výstižné formulace a názornou představu. Budeme mluvit o množství, které „teče“ hranou, budeme zkoumat, kolik jednotek „přitéká“ do vrcholu a kolik z vrcholu „odtéká“.

Budeme řešit úlohu nalezení největšího toku: jaký je největší objem, který můžeme v dané síti dopravit ze zdroje do místa spotřeby (stoku)? Je samozřejmé, že musíme dodržet maximální stanovené kapacity hran, tj. hranou nikdy nelze přepravovat větší množství, než udává kapacita hrany.

Na druhou stranu je nutné si uvědomit, že největší tok neznamená prosté naplnění všech hran na maximální kapacitu. Takový postup by ani nemusel odpovídat

fyzikální realitě! Například do vrcholu $v_1$ v síti na Obrázku 7.2 může přitékat maximálně pět jednotek, zatímco kapacita odchozí hrany by pojala šest jednotek. Tato kapacita šesti jednotek nemůže být nikdy naplněna. Proto kromě pojmu „kapacita hrany“ zavedeme ještě pojem „tok hranou.“

Pro zjednodušení zápisu zavedeme následující značení: Symbolem $e \to v$ označíme všechny *příchozí* hrany do vrcholu $v$ a symbolem $e \leftarrow v$ označíme všechny *odchozí* hrany z vrcholu $v$. Oba symboly reprezentují množiny orientovaných hran grafu $G$:

$$e \to v = \{uv : u \in V(G) \wedge (u,v) \in E(G)\},$$
$$e \leftarrow v = \{vu : u \in V(G) \wedge (v,u) \in E(G)\}.$$

Nyní zavedeme tok v síti jako ohodnocení hran, přičemž budeme požadovat, aby ohodnocení splňovalo určité vlastnosti.

**Definice 7.2.** *Tok* v síti $S = (G, z, s, w)$ je funkce $f : E(G) \to \mathbb{R}_0^+$, která má následující vlastnosti:

i) ohodnocení hrany $f(e)$ se nazývá *tok hranou* a nesmí překročit kapacitu hrany:

$$\forall e \in E(G) : 0 \leqq f(e) \leqq w(e),$$

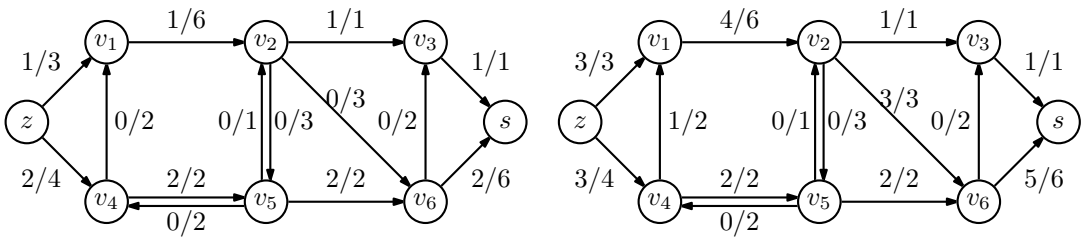
ii) pro všechny vrcholy s výjimkou zdroje a stoku platí tzv. *zákon kontinuity*:

$$\forall v \in V(G), v \neq z, v \neq s : \sum_{e \to v} f(e) = \sum_{e \leftarrow v} f(e).$$

Pro zdroj obecně platí $\sum_{e \to v} f(e) \leqq \sum_{e \leftarrow v} f(e)$, zatímco pro stok platí opačná nerovnost $\sum_{e \to v} f(e) \geqq \sum_{e \leftarrow v} f(e)$.

*Velikost toku* $f$ označíme $\|f\|$ a je dána vztahem

$$\|f\| = \sum_{e \leftarrow z} f(e) - \sum_{e \to z} f(e).$$



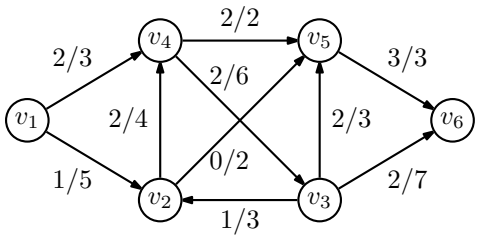

Obrázek 7.3 Tok a největší tok v síti $(G, z, s, w)$.

Na Obrázku 7.3 vidíme příklady toků v nějaké síti. Čísla $a/b$ přiřazená hranám mají význam toku a kapacity příslušné hrany. Například hranou $zv_1$ v síti vlevo teče

jedna jednotka, zatímco kapacita hrany je 3, hrana $v_2v_3$ má nasycenou kapacitu a tok hranou $v_2v_6$ je nulový, třebaže její kapacita je 3.

Uvědomte si, že zákon kontinuity říká: „co do vrcholu přiteče, to z vrcholu odteče“. Tato rovnost však obecně neplatí pro zdroj ani stok, protože ze zdroje (místa vzniku nebo výroby) zpravidla odtéká do sítě více jednotek než do zdroje přitéká a naopak do stoku (místa spotřeby) obvykle přitéká ze sítě více jednotek než ze stoku odtéká.

Složitější síť může obsahovat více zdrojů a stoků. Definici 7.1, která předpokládá existenci jediného zdroje a jediného stoku, není obtížné zobecnit. Místo jediného zdroje $z$ a jediného stoku $s$ můžeme pracovat s množinou zdrojů $Z = \{z_1, z_2, \ldots, z_p\}$ a množinou stoků $S = \{s_1, s_2, \ldots, s_q\}$. Avšak místo abychom zavedli nové definice, tak v další sekci ukážeme, jak některé obecnější úlohy převést na základní úlohu s jediným zdrojem a jediným stokem popsanou sítí z Definice 7.2.




Obrázek 7.4 Síť s vyznačeným tokem i kapacitami hran.

**Příklad 7.3.** Které vrcholy v síti na Obrázku 7.4 jsou podle uvedených toku zdroje? A které vrcholy jsou stoky?

*Řešení.* Ověříme platnost zákona kontinuity pro každý vrchol. Vidíme, že z vrcholu $v_1$ pouze odtékají $1 + 2 = 3$ jednotky a jedná se proto o zdroj. Do vrcholu $v_2$ přitékají i odtékají 2 jednotky, nejedná se o zdroj ani o stok. Do vrcholu $v_3$ přitékají 2 jednotky, ale odtéká celkem $1 + 2 + 2 = 5$ jednotek, proto je vrchol $v_3$ druhým zdrojem v dané síti. Do vrcholu $v_4$ přitékají i odtékají 4 jednotky, nejedná se o zdroj ani o stok. Do vrcholu $v_5$ přitékají $2 + 2 = 4$ jednotky, ale odtékají jen 3 jednotek, proto je vrchol $v_5$ stokem v dané síti. A konečně do vrcholu $v_6$ přitéká $2 + 3 = 5$ jednotek, proto je vrchol $v_6$ druhým stokem v dané síti. ▲

Všimněte si, že definice toku v síti nevylučuje možnost, že i do zdroje mohou přicházet hrany s nenulovým tokem a ze stoku mohou odcházet hrany s nenulovým tokem. Proto je velikost toku rovna rozdílu všeho, co ze zdroje odteče a toho co do zdroje přiteče.

Snadno ověříte, že pro každý vrchol sítí na Obrázku 7.3 (kromě zdrojů a stoků) platí zákon kontinuity a také že velikost toku v síti na Obrázku 7.3 vlevo je 3, zatímco vpravo je tok o velikosti 6, který je největší. Později v této kapitole ukážeme, jak poznat, že tok je největší možný a jak takový tok najít.

Už jsme řekli, že pro zdroj a stok obecně *nemusí platit* zákony kontinuity! Ze zdroje odtéká více jednotek než do zdroje přitéká a do stoku přitéká více než ze stoku odtéká avšak tento rozdíl musí být pro oba vrcholy (až na znaménko) stejný. Toto pozorování shrneme jako tvrzení.

**Lemma 7.4.** *Označíme-li $f_z$ rozdíl toků na odchozích a příchozích hranách zdroje a dále $f_s$ rozdíl toků na odchozích a příchozích hranách stoku, tak platí $f_z = -f_s$.*

*Důkaz.* Tvrzení ukážeme přímo. Mějme síť $G$, $G = (V, E)$, se zdrojem $z$, stokem $s$ a tokem $f$. Využijeme následující trik: napíšeme nulu jako rozdíl dvou stejných výrazů $f(e) - f(e)$ pro každou hranu $e \in E(G)$ v síti.

$$0 = \sum_{e \in E(G)} (f(e) - f(e))$$

Nyní přeorganizujeme sčítance sum tak, aby jsme pro každý vrchol sečetli ohodnocení všech odchozích hran a odečteme pro každý vrchol součet ohodnocení všech příchozích hran.

$$\begin{aligned} 0 &= \sum_{v \in V(G)} \sum_{e \leftarrow v} f(e) - \sum_{v \in V(G)} \sum_{e \rightarrow v} f(e) \\ &= \sum_{v \in V(G)} \left( \sum_{e \leftarrow v} f(e) - \sum_{e \rightarrow v} f(e) \right) \end{aligned}$$

Podle zákona kontinuity je pro každý vrchol (s výjimkou zdroje $z$ a stoku $s$) rozdíl $\sum_{e \leftarrow v} f(e) - \sum_{e \rightarrow v} f(e)$ roven nule. Dostaneme součet s pouze dvěma sčítanci.

$$0 = \sum_{v \in \{z,s\}} \left( \sum_{e \leftarrow v} f(e) - \sum_{e \rightarrow v} f(e) \right)$$

Odtud ihned dostaneme hledané tvrzení

$$f(z) = \left( \sum_{e \leftarrow z} f(e) - \sum_{e \rightarrow z} f(e) \right) = -\left( \sum_{e \leftarrow s} f(e) - \sum_{e \rightarrow s} f(e) \right) = f(s).$$

Všimněte si, že díky zákonu kontinuity můžeme vlastně celou síť chápat jako zdroj spojený jedinou hranou se stokem, přičemž tok touto hranou je roven toku v dané síti. □

## Kontrolní otázky

1. Může existovat síť, kde pro zdroj i stok platí zákon kontinuity?
2. Může existovat síť s nenulovým tokem, kde pro zdroj i stok platí zákon kontinuity?
3. Může existovat síť s nenulovým tokem na odchozích hranách ze stoku, kde však pro zdroj i stok platí zákon kontinuity?

**Poznámka 7.5.** Zdroj se v anglické literatuře nazývá „source“ a stok se nazývá „sink“. V české v literatuře se stoku říká také „nor“, podobně jako se nazývá místo, kde řeka vtéká pod zem.

Obrázek 7.5 Řeka Punkva v Moravském krasu.

### Pojmy k zapamatování

— síť $(G, z, s, w)$
— kapacita hrany a tok hranou
— zdroj a stok v síti
— tok a velikost toku v síti
— zákon kontinuity

## 7.2 Hledání největšího toku

### Průvodce studiem


*V předchozí sekci jsme ukázali, jak popsat reálnou dopravní nebo komunikační úlohu sítí, tj. grafem s předepsanými kapacitami hran a vyznačeným výchozím i cílovým místem dopravy nebo komunikace. Nyní ukážeme, jak určit největší možný objem dopravované látky nebo dat v dané síti s předepsanými omezujícími kapacitami hran.*

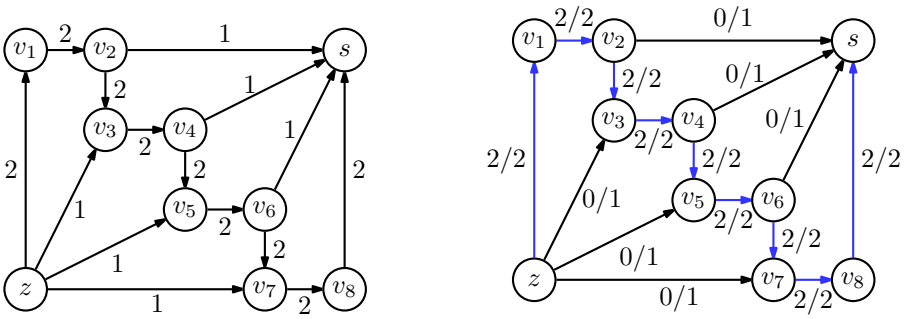
### Cíle

Po prostudování této kapitoly budete schopni:

- popsat, co je to řez v síti a vysvětlit, jak souvisí s tokem v síti,

- vysvětlit pojem „nenasycená cesta“ v síti a nenasycené cesty v dané síti najít,
- najít největší tok v dané síti.

Hlavním úkolem, kterému se věnuje tato kapitola, je nalezení co *největšího* toku (toku s největší možnou velikostí) ze zdroje do stoku v síti dané orientovaným grafem $G$ s předepsanými kapacitami hran. V sekci 5.4 jsme při hledání minimální kostry s výhodou využili hladový algoritmus. Bohužel, při hledání největšího toku v síti hladový algoritmus obecně *nevede* k nejlepšímu řešení! Na Obrázku 7.6 vlevo je příklad sítě s kapacitami hran, pro který hladový algoritmus hledání největšího toku selže.




Obrázek 7.6 Příklad sítě, kdy hladový algoritmus nedá optimální řešení.

Bude-li hladový algoritmus hledat v síti co nejvýhodnější cestu ze zdroje do stoku (nejvýhodnější cestou rozumíme takovou cestu, po které můžeme navýšit tok o nejvíce jednotek), tak využije cestu $z$, $v_1$, $v_2$, $v_3$, $v_4$, $v_5$, $v_6$, $v_7$, $v_8$, $s$, po které může zvýšit tok o dvě jednotky (Obrázek 7.6 vpravo). Užitím hladového algoritmu najdeme tok o velikosti 2. Potom skutečně ještě sice vedou nenasycené hrany ze zdroje do vrcholů $v_3$, $v_5$ a $v_7$, ale z ani jednoho z těchto vrcholů již nevede hrana, která by měla volnou kapacitu. Tok už nemůžeme zvýšit přidáním nějaké orientované cesty s nenulovým tokem, avšak nalezený tok není největší, neboť v dané síti existuje tok velikosti 5 (Obrázek 7.7).

Abychom popsali algoritmus, který bude umět největší tok najít, budeme potřebovat několik nových pojmů. Uvědomte si, že vůbec nemusí být na první pohled jasné, zda nalezený tok je největší. Abychom uměli poznat, zda nějaký tok má nebo nemá největší možnou velikost, kterou daná síť připouští, tak zavedeme tzv. „řez“ v grafu. Jeli $X \subseteq E(G)$ množina některých hran grafu, tak symbolem $G-X$ budeme rozumět graf, který vznikne z grafu $G$ odebráním hran množiny $X$.

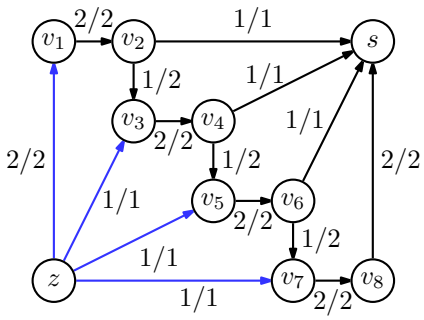
**Definice 7.6.** *Řez* $C$ v síti $S = (G, z, s, w)$ je taková podmnožina hran $C \subseteq E(G)$, že v grafu $G - C$ nezůstane žádná orientovaná cesta ze zdroje $z$ do stoku $s$.
*Kapacitu* (nebo velikost) řezu $C$ značíme $\|C\|$ a definujeme ji jako součet kapacit všech hran řezu $C$, tj. $\|C\| = \sum_{e \in C} w(e)$.

Pro označení řezu používáme písmeno „$C$“ podle anglického termínu „cut“. Význam řezu je jasný – odebrání hran řezu rozdělí, „rozřízne“, orientovaný graf sítě na části tak, že stok se nachází v jiné části než zdroj, přičemž neexistuje orientovaná cesta ze zdroje do stoku. Pozor, nemusí se jednat o komponenty ve smyslu jednoduchého grafu, neboť odhlédneme-li od orientace hran, tak neorientované cesty ze zdroje do stoku ve výsledném grafu existovat mohou. Každá však bude obsahovat alespoň jednu hranu, která je v původním grafu orientována „proti směru“ cesty. Následující věta ukáže, že řez je klíčovým nástrojem pro rozpoznání, zda nějaký tok je největší. Řez v síti označíme za *nejmenší*, jestliže má ze všech možných řezů dané v síti nejmenší kapacitu.

**Věta 7.7.** *Velikost největšího toku v síti je rovna kapacitě nejmenšího řezu.*

Důkaz Věty 7.7 uvedeme později, na straně 135.

Na Obrázku 7.7 je příklad řezu, který je vyznačen modře. Zdůrazněme, že řez je množina modrých hran.




Obrázek 7.7 Příklad sítě, s největším tokem o velikosti 5 a řezem s kapacitou 5.

Věta 7.7 je dobrou charakterizací největšího toku. Podle ní snadno poznáme, že tok o velikosti $\|f\|$ v dané síti je největší, jestliže najdeme řez o velikosti $\|C\| = \|f\|$. Zatím ještě neumíme největší tok najít.

Doposud jsme mlčky předpokládali, že při zvětšování toku po nějaké cestě ze zdroje do stoku musíme vždy postupovat „ve směru šipek“. Ukazuje se, že v algoritmu hledání největšího toku s takovými cestami nevystačíme a budeme pracovat i s cestami, jejichž některé hrany budou orientovány „proti“ postupu ze zdroje do stoku, ovšem za předpokladu, že taková hrana už má nenulový (opačný) tok, který budeme moci „přetlačit“ a snížit.

**Definice 7.8.** *Polocestou* v orientovaném grafu rozumíme takovou posloupnost vrcholů orientovaných hran $(v_0, e_1, v_1, e_2, v_2 \ldots, e_m, v_n)$, kde orientovaná hrana $e_i$ má buď počáteční vrchol $v_{i-1}$ a koncový vrchol $v_i$ (hrana „ve směru polocesty“) nebo počáteční vrchol $v_i$ a koncový vrchol $v_{i-1}i$ (hrana „proti směru polocesty“).

Polocesta v orientovaném grafu je vlastně cestou v neorientovaném grafu, kterou dostaneme, odhlédneme-li od orientace hran. My však nechceme orientaci hran

zanedbat zcela, protože v síti s daným tokem budeme jinak počítat rezervy kapacit pro hrany „ve směru“ a jinak pro hrany „proti směru“ polocesty.

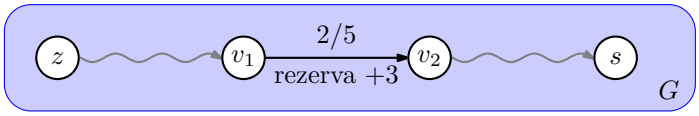
**Definice 7.9.** Máme dánu síť $S = (G, z, s, w)$ a v ní tok $f$. *Nenasycená cesta* v síti $S$ s tokem $f$ je polocesta $e_1, e_2, \ldots, e_m$ v orientovaném grafu $G$ z vrcholu $u$ do vrcholu $v$ (obvykle ze zdroje $z$ do stoku $s$), na které kde

- pro všechny hrany $e_i$ „ve směru“ z $u$ do $v$ nedosahuje tok kapacity ($f(e_i) < w(e_i)$) ,
- pro všechny hrany $e_i$ v opačném směru je tok kladný ($f(e_i) > 0$).

Hodnotě $w(e_i) - f(e_i)$ pro hranu $e_i$ ve směru z vrcholu $u$ do vrcholu $v$ říkáme *rezerva kapacity hrany* $e_i$ a stejně říkáme hodnotě $f(e_i)$ pro hrany $e_i$ v opačném směru.

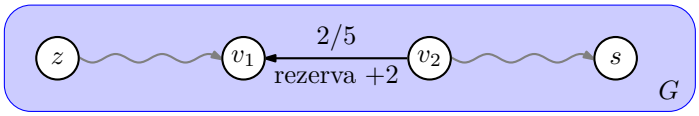
Každá hrana může být orientována libovolně: buď „ve směru“ nebo „proti směru“ cesty. Důležité je pouze, aby každá hrana nenasycené cesty měla kladnou rezervu kapacity hrany.

Na Obrázku 7.8 je příklad hrany na cestě ze zdroje do stoku orientované „ve směru“ cesty, přičemž tok hranou nedosahuje kapacity a rezerva kapacity hrany je určena rozdílem kapacity a aktuálního toku hranou.




Obrázek 7.8 Hrana s rezervou kapacit +3 na hraně „ve směru“ cesty.

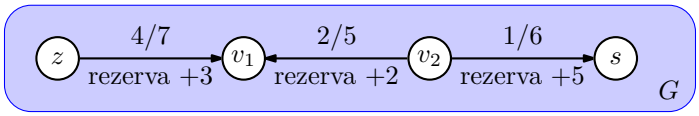
Naproti tomu na Obrázku 7.9 je příklad hrany na cestě ze zdroje do stoku orientované „proti směru“ cesty, přičemž tok není nulový a rezerva kapacity hrany je určena aktuálním tokem. Abychom zvýšili tok podél celé nenasycené cesty, tak můžeme *snížit* tok touto opačně orientovanou hranou (a pochopitelně upravíme toky i na ostatních hranách této nenasycené cesty).




Obrázek 7.9 Hrana s rezervou kapacit +2 na hraně „proti směru“ cesty.

Nenasycená cesta je podle definice cesta s kladnými (tj. nenulovými) rezervami kapacit všech hran. Na Obrázku 7.10 je příklad nenasycené cesty s různě orientovanými hranami, přičemž každá má kladnou rezervu kapacity hrany. Nejmenší rezervu všech kapacit hran nenasycené cesty nazveme *rezervou nenasycené cesty* a označíme

ji třeba $\delta$. To znamená, že podél této nenasycené cesty můžeme zvýšit tok až o $\delta$ jednotek.




Obrázek 7.10 Cesta s rezervou kapacit $\delta = 2$.

Analogicky bychom mohli zavést pojem *nasycené cesty* ze zdroje, pro kterou alespoň jedna hrana má nulovou rezervu, avšak taková cesta není důležitá při hledání největšího toku v síti. Klíčové je najít nenasycené cesty, neboť právě nenasycené cesty umožní postupně zvyšovat aktuální tok v síti.

Nyní můžeme popsat algoritmus, který v dané síti najde největší tok.

**Algoritmus 7.1 (Ford–Fulkersonův algoritmus).**

```
vstup < síť S = (G,z,s,w);
výchozí tok f je pro každou hranu nulový;

do {
    do U zařadíme zdroj z a prohledáním grafu G přidáme do U,
    všechny vrcholy dosažitelné ze z po nenasycených cestách;
    if (s náleží U) {
       P = nějaká (výše nalezená) nenasycená cesta v S ze z do s;
       zvětšíme tok f o nejmenší rezervu kapacit hran podél cesty P;
} while (s náleží U);
výstup:  vypíšeme největší tok f;
výstup:  nejmenší řez je množina hran vedoucích z U do V(G)-U.
```

Nyní ukážeme, že Algoritmus 7.1 pracuje správně. Dokonce ukážeme, že na konci běhu algoritmu dostaneme tok, jehož velikost se rovná kapacitě (některého) nejmenšího řezu a tím současně ukážeme platnost Věty 7.7.

**Věta 7.10.** *Mějme síť $S = (G, z, s, w)$ s přirozenými (případně nezápornými celočíselnými) kapacitami hran. Algoritmus 7.1 najde největší tok v síti $S$.*

*Důkaz.* Důkaz správnosti algoritmu provedeme přímo.

Podle definice toku a řezu je zřejmé, že pro libovolný tok $f$ a libovolný řez $c$ musí platit $\|f\| \leqq \|C\|$. Jestliže po skončení algoritmu budeme mít tok $f$ a najdeme v síti $S$ řez $C$ o stejné velikosti $\|C\| = \|f\|$, tak to znamená, že jsme našli největší možný tok v síti $S$, neboť $\|f\|$ nemůže přesáhnout $\|C\|$. Stačí tedy dokázat, že po skončení algoritmu dostaneme rovnost $\|f\| = \|C\|$, kde $C$ je řez obsahující hrany, které mají počáteční vrchol v množině $U$ a koncový vrchol v množině $V(G) \setminus U$.

Předpokládejme, že máme tok $f$ v síti $S$ a že algoritmus skončil, protože už v síti není nenasycená cesta ze zdroje $z$ do stoku $s$. To znamená, že množina $U$ (po skončení algoritmu) neobsahuje vrchol $s$ (už není dosažitelný po nenasycené cestě).

Z žádného vrcholu množiny $U$ nevede nenasycená hrana (ani cesta) do vrcholu v množině $V(G) \setminus U$, protože bychom koncový vrchol takové hrany mohli přidat do množiny $U$. A podobně do žádného vrcholu množiny $U$ nevede hrana s nenulovým tokem z nějakého vrcholu v množině $V(G) \setminus U$, protože bychom počáteční vrchol takové hrany mohli přidat do množiny $U$. Označme množinu všech hran odcházejících z množiny $U$ symbolem $e \leftarrow U$ a množinu všech hran přicházející do množiny $U$ symbolem $e \rightarrow U$. Každá z hran v $e \leftarrow U$ má plný tok $f(e) = w(e)$ a každá hrana v $e \rightarrow U$ má nulový tok $f(e) = 0$, jinak by množina $U$ neodpovídala algoritmu.

Podle definice toku je

$$\|f\| = \sum_{e \leftarrow z} f(e) - \sum_{e \rightarrow z} f(e).$$

S využitím zákona kontinuity snadno nahlédneme, že pokud nějaká množina vrcholů $X \subseteq V(G)$ obsahuje zdroj $z$ a neobsahuje stok $s$, tak

$$\|f\| = \sum_{v \in X} \left( \sum_{e \leftarrow v} f(e) - \sum_{e \rightarrow v} f(e) \right).$$

Odstraněním všech hran množiny $e \leftarrow U$ z grafu přerušíme všechny orientované cesty ze zdroje do stoku, proto je množina $e \leftarrow U$ současně řezem $C$ v dané síti. Jestliže za $X$ vezmeme množinu $U$, tak pro velikost toku ze zdroje $z$ do stoku $s$ můžeme počítat:

$$\|f\| = \sum_{e \leftarrow U} f(e) - \sum_{e \rightarrow U} f(e) = \sum_{e \leftarrow U} f(e) - 0 = \sum_{e \in C} w(e) = \|C\|.$$

Vidíme, že velikost toku $f$, je rovna kapacitě řezu $C$. To znamená, algoritmus pracuje správně a najde největší tok i nejmenší řez. To je dokazované tvrzení. Současně byla dokázána Věta 7.7. □

Uvedený algoritmus nejen najde největší tok s velikostí $\|f\|$, ale snadno sestavíme i nejmenší řez s kapacitou $\|C\|$.

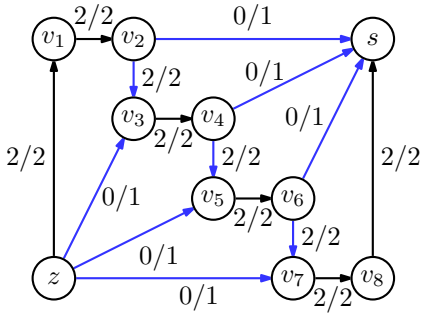
Poslední tvrzení této sekce ukazuje jedno pěkné pozorování: Největší tok v síti vždy plně využije kapacity hran nějakého nejmenšího řezu a proto je velikost toku součtem kapacit hran řezu. To znamená, že budou-li všechny kapacity nezáporná celá čísla, bude i velikost toku celočíselná.

**Důsledek 7.11.** *Jsou-li kapacity hran sítě $S$ přirozená čísla (resp. nezáporná celá čísla), tak největší tok v síti bude také přirozené číslo (resp. nezáporné celé číslo).*

**Příklad 7.12.** Najděte nenasycené cesty v síti na Obrázku 7.6 vpravo.

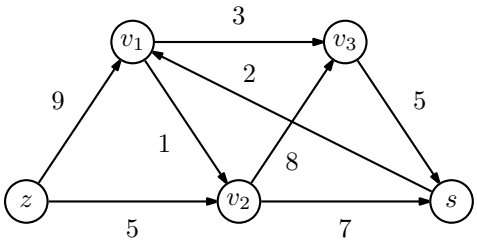
*Řešení.* Nenasycených cest ze zdroje do stoku existuje v dané síti šest různých (najdete je?). My vyznačíme tři z nich, které jsou interně disjunktní a každá z nich má rezervu kapacit rovnu 1.

Na Obrázku 7.11 jsou nenasycené cesty vyznačeny modře. Zvětšením toku pomocí těchto nenasycených cest dostaneme tok o velikosti 5, který je největší a který je uveden na Obrázku 7.7. Také řez uvedený na Obrázku 7.7 se shoduje s nejmenším řezem, který sestavíme pomocí Algoritmu 7.1. ▲




Obrázek 7.11 Příklad tří nenasycených cest v síti.

**Příklad 7.13.** Najděte největší tok v síti na Obrázku 7.12.

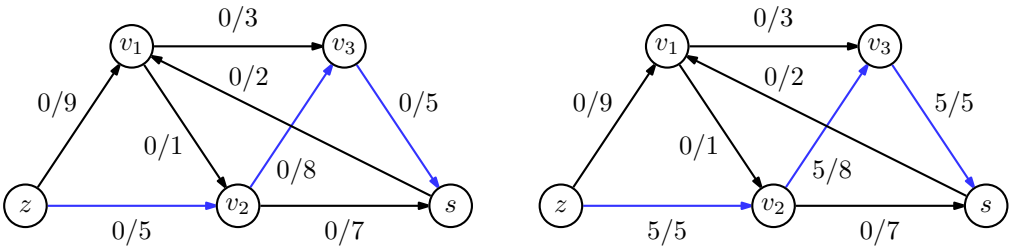



Obrázek 7.12 Síť s danými kapacitami hran.

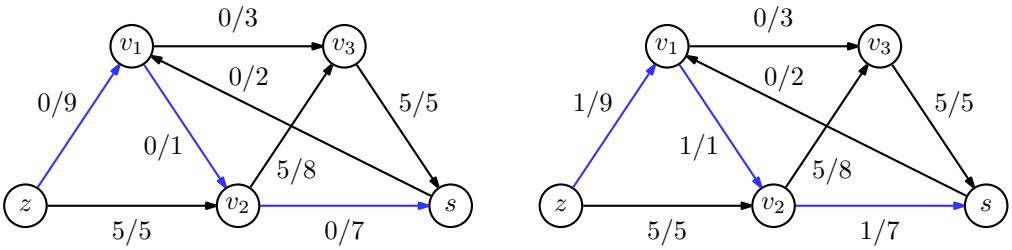
*Řešení.* K nalezení největšího toku $f$ použije Algoritmus 7.1. Začínáme hledat v síti s nulovým tokem. Najdeme nenasycenou cestu $z, v_2, v_3, s$, jejíž hrana $zv_2$ má rezervu kapacity rovnu 5, hrana $v_2v_3$ má s rezervu kapacity 8 a hrana $v_3s$ má rezervu kapacity 5. Na Obrázku 7.13 je nenasycená cesta $z, v_2, v_3, s$ zvýrazněna modře. Minimální rezerva na této cestě je 5, proto můžeme podél cesty $z, v_2, v_3, s$ zvýšit tok o 5 jednotek. Dostaneme tok o velikosti $\|f\| = 5$ v síti na Obrázku 7.13 vpravo.

Opět hledáme nenasycenou cestu a zvolíme například cestu $z, v_1, v_2, s$ (na Obrázku 7.14 je zvýrazněna modře). Rezervy kapacit jsou po řadě 9, 1 a7, proto rezerva nenasycené cesty má hodnotu 1. Zvýšíme tok podél nenasycené cesty $z, v_1, v_2, s$ o 1 jednotku a dostaneme celkový tok o velikosti $\|f\| = 5+1 = 6$ (Obrázek 7.14 vpravo).

Všimněte si, že nyní už nenajdeme *orientovanou* nenasycenou cesty ze zdroje do stoku (Obrázek 7.14), protože každá taková cesta by musela vést buď přes vrchol $v_2$ (avšak všechny hrany přicházející do vrcholu $v_2$ jsou nasyceny), nebo přes

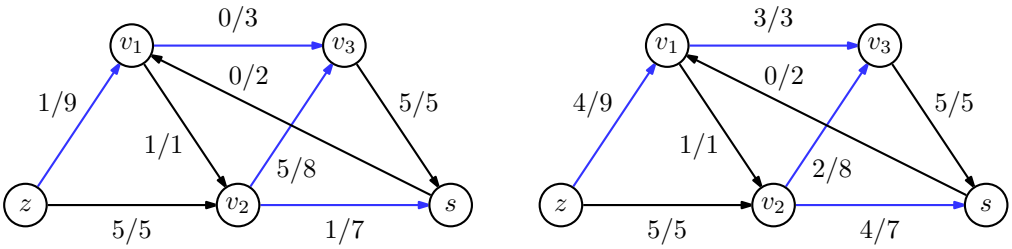



Obrázek 7.13 Zvětšení toku podél nenasycené cesty $z, v_2, v_3, s$.




Obrázek 7.14 Zvětšení toku podél nenasycené cesty $z, v_1, v_2, s$.

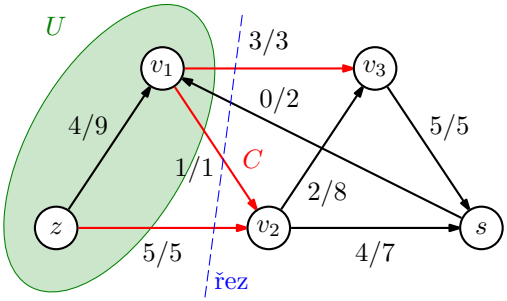
vrchol $v_3$ (avšak všechny hrany odcházející z vrcholu $v_3$ jsou také nasyceny). To ale *neznamená*, že nalezený tok o velikosti $\|f\| = 6$ je největší! Jestliže pečlivě prozkoumáme, do kterých vrcholů najdeme ze zdroje nenasycenou cestu, tak zjistíme, že do vrcholu $v_1$ vede nenasycená hrana s rezervou kapacity 8, z vrcholu $v_1$ do vrcholu $v_3$ hrana s rezervou kapacity 3, z vrcholu $v_3$ do vrcholu $v_2$ hrana s rezervou kapacity 5 (tato hrana je opačně orientována!) a konečně z vrcholu $v_2$ do stoku $s$ hran s rezervou kapacity 6. Do množiny $U$, která obsahuje vrcholy, do nichž vede nenasycená cesta, patří všechny vrcholy grafu. Na Obrázku 7.15 je nenasycená cesta $z, v_1, v_3, v_2, s$ vyznačena modře. Podél této cesty můžeme tok zvětšit o další 3 jednotky, což je nejmenší rezerva jednotlivých hran, a dostaneme tok o velikosti $\|f\| = 6 + 3 = 9$ (Obrázek 7.15 vpravo).




Obrázek 7.15 Zvětšení toku podél nenasycené cesty $z, v_1, v_3, v_2, s$.

Nyní ukážeme, že nalezený tok $f$ je největší. Opět sestavíme množinu $U$, zařadíme do ní všechny vrcholy, které jsou dosažitelné ze zdroje po nenasycených cestách. Nenasycená cesta však vede pouze do vrcholu $v_1$. Na Obrázku 7.16 je množina $U$ vyznačena zeleně, přičemž všechny hrany odcházející z vrcholů množiny $U$

jsou nasyceny a všechny hrany přicházející do množiny $U$ mají nulový tok. Odcházející hrany jsou vyznačeny červeně a tvoří řez $C = \{zv_2, v_1v_2, v_1v_3\}$ s velikostí $\|C\| = 5 + 1 + 3 = 9$. Protože jsme našli řez $C$ o kapacitě $\|C\| = 9$ stejné jako je největší tok o velikosti $\|f\| = 9$, tak podle Věty 7.7 víme, že nalezený tok $f$ je největší (Obrázek 7.16).




Obrázek 7.16 Největší tok $f$ a nejmenší řez $C$ v síti.

Všimněte si, že po odstranění hran řezu nebude v grafu existovat žádná orientovaná cesta ze zdroje do stoku. Neorientovanou cestu ze zdroje do stoku však najdeme, protože v grafu zůstane hrana $sv_1$. Nebude se však jednat o nenasycenou cestu!

Řez v grafu bývá někdy naznačen čarou. Například řez $C$ na Obrázku 7.16 je vyznačen modrou čárkovanou čarou. Při interpretaci takového řezu musíme být velmi opatrní, neboť řez je tvořen červeně zvýrazněnými hranami, avšak hrana $sv_1$ do řezu *nepatří*! ▲

## Pro zájemce:

Uvědomte si, že část pseudokódu Algoritmu 7.1 je velmi volně zformulována. Neříká, jakým způsobem najdeme nenasycené cesty, protože postup závisí na konkrétní implementaci sítě. Můžeme postupovat například modifikovaným Dijkstrovým algoritmem 4.3 a pro každý vrchol ukládat největší rezervu spolu s předchozím vrcholem pro snadnou rekonstrukci nenasycených cest. Tak můžeme dostat modifikaci Ford-Fulkersonova algoritmu, kde se v každém kroku použije nenasycená cesta s největší kapacitou, kterou najdeme například užitím algoritmu pro hledání nejširších cest ze strany 79. Můžeme však sestavit množinu $U$ prohledáváním do šířky a dostaneme modifikaci Ford-Fulkersonova algoritmu, kde se pracuje s nejkratšími nenasycenými cestami. Dokonce můžeme v každém cyklu zvolit libovolnou z nenasycených cest a vždy nakonec dostaneme tok se stejnou největší velikostí.

**Poznámka 7.14.** Je nutno připomenout, že v diskrétní matematice rozlišujeme pojmy *maximální* a *největší tok*. Zatímco „největší“ tok nabývá v porovnání s ostatními variantami největší možné hodnoty, tak „maximální“ tok obvykle chápeme jako stav, kdy daný parametr (tok v síti) použitým postupem (přidáváním toků podél

orientovaných cest) již nejde zvětšit. Proto se důsledně bavíme o „největším toku“ a ne o maximálním toku. Vždyť bez znalosti nenasycených cest můžeme chápat tok na Obrázku 7.6 jako maximální, který ale jistě není největším tokem v dané síti (Obrázek 7.7). Tok na Obrázku 7.6 nemůžeme zvětšit, aniž bychom *zmenšili* tok na některých hranách. Rozdíl mezi pojmy *maximální* a *největší* v širším kontextu je podrobně popsán v textu [6] v kapitole o relacích.

Analogicky chápeme rozdíl mezi pojmy *nejmenší* a *minimální řez*. V síti hledáme nejmenší řez, neboť „minimální řez“ lze také chápat jako takovou množinu hran, že žádná její vlastní podmnožina již řez netvoří. Nás však zajímá řez s nejmenší kapacitou. Víme, že nejmenší řez bude současně minimální, ale ne každý minimální řez musí být nejmenší. Nejmenší řez dokonce nemusí mít ani nejmenší počet hran mezi všemi řezy v dané síti.

### Pro zájemce:

Je dobré upozornit, že pokud vynecháme požadavek, aby kapacity hran byly nezáporná celá čísla (případně racionální čísla), tak je možno sestavit příklady jednoduchých sítí s iracionálními kapacitami, pro které Algoritmus 7.1 neskončí po *konečném* počtu kroků. V důkazu jsme vycházeli z předpokladu celočíselnosti kapacit hran a předpokládali jsme, že algoritmus skončil. Rezerva nenasycené cesty byla celočíselná a proto se v každé iteraci zvýšil tok alespoň o 1.

V síti s iracionálním ohodnocením hran může rezerva nenasycené cesty být libovolně malá a proto počet kroků algoritmu nemusí být konečný. Dokonce postupné iterace získané v průběhu algoritmu nemusí konvergovat k největšímu (ani maximálnímu) toku v dané síti. Pokud jsou však všechny kapacity hran nezáporná celá čísla, tak uvedený algoritmus dává správné výsledky.

### Pojmy k zapamatování

— řez v síti, minimální řez
— nenasycená cesta v síti
— rezerva hrany a rezerva nenasycené cesty
— Ford-Fulkersonův algoritmus

## 7.3 Zobecnění sítí


### Průvodce studiem

*V předchozích sekcích jsme pečlivě popsali úlohu hledání největšího toku v síti s jediným zdrojem a jediným stokem. V praxi je běžné, že v dopravní nebo komunikační síti je však zdrojů i míst spotřeby mnoho. Pro řešení úlohy hledání největšího toku v síti s více zdroji a stoky nebudeme modifikovat algoritmus, ale ukážeme, jak jednoduchým trikem převést*

*síť s více zdroji na síť se zdrojem jediným (a podobně s jediným stokem). Dále ukážeme, jak upravit síť s neorientovanými hranami tak, abychom mohli použít Algoritmus 7.1, který pracuje s orientovanými hranami.*

*V této sekci navíc ukážeme, jak úlohu hledání největšího toku v síti s danými kapacitami hran zobecnit na úlohu v takové síti, kde jsou navíc předepsané kapacity vrcholů a jak tuto úlohu řešit. Také se zmíníme o komplikacích, které přináší obecnější problém, kdy zkoumáme síť, ve které se přepravuje více různých komodit a upozorníme na chyby, kterých se musíme vyvarovat. Na závěr zmíníme zobecnění úlohy (které souvisí například s přepravou pitné vody), kdy máme pro každou hranu kromě maximální kapacity stanoven požadavek minimální kapacity, kdy požadujeme, že danou hranou musí téci nějaké minimální předepsané množství.*

## Cíle

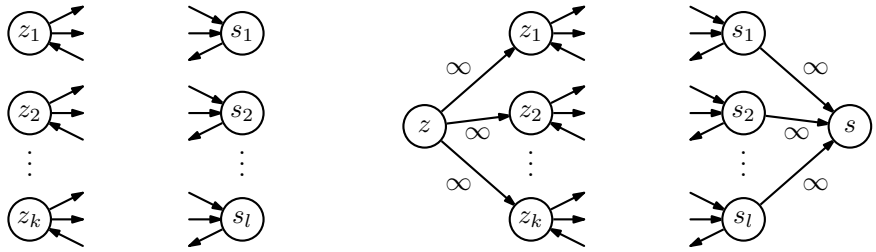
Po prostudování této kapitoly budete schopni:

- převést úlohu hledání největšího toku v síti s více zdroji a stoky na úlohu s jediným zdrojem a jediným stokem,
- převést úlohu hledání největšího toku v síti s danými kapacitami vrcholů na úlohu v síti s kapacitami hran,
- vysvětlit rozdíl v přístupu k řešení mezi sítěmi s jediným a s více dopravovanými produkty.

Jak jsme zmínili v úvodu, dopravní nebo komunikační síť může mít předepsané kapacity vrcholů, minimální kapacity hran, v síti se může vyskytovat více zdrojů a stoků a může se dopravovat více různých produktů. Ukážeme, že některé z těchto úloh můžeme převézt na základní úlohu nalezení největšího toku v síti řešitelnou Algoritmem 7.1. Místo abychom sestavovali nové algoritmy, tak ukážeme, jak modifikovat samotnou síť a využít už popsaný (a dokázaný) Algoritmus 7.1.

### Více zdrojů a stoků v síti

Vyskytuje-li se v grafu sítě více zdrojů nebo stoků, můžeme úlohu jednoduše převést na úlohu s jediným zdrojem a jediným stokem. Stačí, když do sítě přidáme jeden nový „hlavní“ zdroj a spojíme jej orientovanou hranou s každým ze zdrojů v síti (Obrázek 7.17). Kapacity nových hran stanovíme dostatečně vysoké. Kapacitu hrany z nového zdroje do původního zdroje $z_i$ by měl být alespoň tak vysoká jako je nejvyšší možný tok ze zdroje $z_i$. Pro názornost jsme na Obrázku 7.17 zvolili kapacitu $\infty$ a v závislosti na konkrétní implementaci použijeme například konstantu reprezentující nejvyšší možné číslo. Podobně přidáme nový „hlavní“ stok. Tak dostaneme síť s jediným zdrojem a jediným stokem a není těžké si rozmyslet, že nalezení největšího toku v nové síti je ekvivalentní úloze nalezení největšího toku v původní síti.




Obrázek 7.17 Převedení sítě s více zdroji a stoky (vlevo) na síť s jediným zdrojem a stokem (vpravo).

## Pro zájemce:

Novému zdroji se někdy říká „superzdroj“ a stoku „superstok“ z anglického „supersource“ a „supersink“. Slovíčko „super“ má v tomto kontextu význam „nadřazený“, ne „vynikající“.

Můžeme dokonce stanovit kapacity jednotlivých zdrojů a stoků, protože v praxi je přirozené požadovat, že místa výroby nebo spotřeby mají také omezenou kapacitu. Abychom stanovili kapacitu $c_i$ zdroje $z_i$, tak hraně $zz_i$ místo kapacity $\infty$ přiřadíme kapacitu $c_i$. Analogicky můžeme omezit kapacity stoků.

### Neorientované hrany

Pokud máme například kanalizační potrubí, tak orientace hran je jasná, protože voda teče vždy dolů. U silniční sítě je orientace většinou také daná, protože jízdní směry jsou oddělené a každý směr má svůj jízdní pruh. Avšak u produktovodů mohou být hrany orientované (komoditu můžeme dopravovat jen jedním směrem), ale i neorientované, kdy směr dopravy může být řízen podle potřeby (například některé plynovody a ropovody). A konečně informační nebo počítačové síti odpovídá většinou neorientovaný graf, protože komunikace probíhá oběma směry.

Uvědomte si, že Algoritmus 7.1 funguje správně jen pro orientované grafy, tedy takové grafy, kde každá hrana je orientovaná. Při hledání nenasycených cest pečlivě rozlišujeme orientaci hran a pro správné ukončení algoritmu je rozlišení souhlasně a nesouhlasně orientovaných hran klíčové.

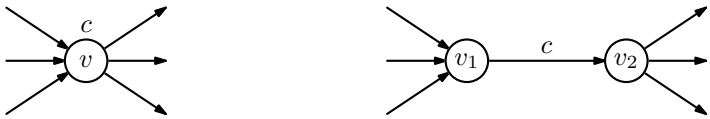

Obrázek 7.18 Nahrazení neorientované hrany dvojicí opačně orientovaných hran.

Naštěstí můžeme snadno neorientovanou hranu v síti eliminovat, když ji nahradíme dvojicí opačně orientovaných hran se stejnou kapacitou (Obrázek 7.18). Ale pozor, po skončení algoritmu se může stát, že existuje nenulový tok oběma opačně

orientovanými hranami. To neodpovídá obvykle realitě, neboť ve skutečnosti je mezi koncovými uzly spojnice jediná. Tok původní (neorientovanou) hranou je dán rozdílem hodnot toků na obou orientovaných hranách a směr, kterým výsledný tok neorientovanou hranou teče, odpovídá směru orientované hrany s větší hodnotou nalezeného toku.

### Kapacity vrcholů

V praktické úloze, kterou reprezentujeme nějakou síti, mohou mít omezenou kapacitu nejen hrany, ale také vrcholy (uzly nebo místa křížení). V dopravní síti jsou nejvíce omezujícím místem křižovatky, v internetové síti je zpravidla rozhodující zatížení aktivních prvků (uzlů sítě). Ukážeme, že síť s předepsanými kapacitami hran i vrcholů můžeme poměrně snadno převést na síť, ve které jsou ohodnocené pouze hrany.




Obrázek 7.19 Vrchol s předepsanou kapacitou $c$ (vlevo) a jeho zdvojení s kapacitou $c$ na hraně (vpravo).

Stačí, když každý vrchol s nějakou stanovenou kapacitou $c$ *zdvojíme*, tj. nahradíme jej dvojicí vrcholů spojených hranou s kapacitou $c$. Přitom všechny hrany přicházející do vrcholu $v$ budou přicházet do vrcholu $v_1$ a všechny hrany odcházející z vrcholu $v$ budou odcházet z vrcholu $v_2$. Jestliže takto zdvojíme každý vrchol s omezenou kapacitou, tak dostaneme (větší) síť, ve které jsou ohodnocené pouze hrany, přičemž neohodnoceným hranám z původního grafu přiřadíme dostatečně velkou kapacitu (například $\infty$ nebo nejvyšší kapacitu ze všech kapacit hran a vrcholů původního grafu). Pro hledání největšího toku v novém grafu tak vystačíme opět s Algoritmem 7.1.

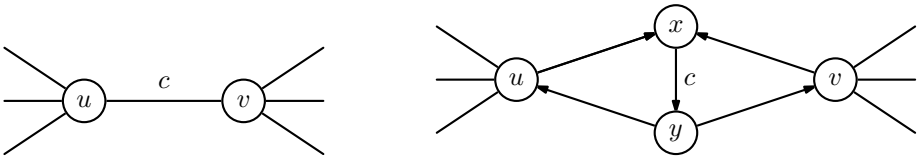
## Pro zájemce:

### Víceproduktové toky

Jestliže máme dopravní síť, ve které se dopravuje více různých produktů (typickým příkladem je silniční síť, může se však jednat i o schéma výrobní haly, kde se pohybují dopravní vozíky, tak se jedná o složitější úlohu. Mohlo by se zdát, že stačí najít největší tok pro každý dopravovaný produkt zvlášť, ale není tomu tak, pokud se produkty dopravují ve *společné síti*. Budeme-li pro každý dopravovaný produkt počítat s plnou kapacitou hrany, tak se snadno může stát, že součet toků různých komodit jednou hranou přesáhne kapacitu hrany. Na druhou stranu, pokud po nalezení toku pro první produkt snížíme kapacitu hrany o hodnotu toku prvního produktu, tak při následném hledání největšího toku dalších komodit nemusíme najít největší možný tok pro další komodity, jaký by byl možný

v původním grafu. Nenulový tok pro další komoditu dokonce ani nemusí existovat. Pro víceproduktové toky proto není jednoznačné určit, který tok je „lepší“. Pro různá pořadí zpracovaných produktů můžeme dostat různá řešení. Algoritmus pro hledání největšího víceproduktového (vícekomoditního) toku je nad rámec tohoto textu.

Ukážeme alespoň, jak převedeme neorientovaný graf na orientovaný s předepsanou kapacitou hrany. Všimněte si, že nemůžeme použít trik z Obrázku 7.18, protože by každou z dvojice opačně orientovaných hran mohl být nenulový tok pro *jiný* produkt a celkem by pak byla překročena kapacita neorientované hrany $uv$. Není totiž možné jednoduše vzít rozdíl toků, neboť se jedná o tok jiné komodity.




Obrázek 7.20 Nahrazení neorientované hrany pro víceproduktové toky.

Lze však použít jiné, složitější, nahrazení (Obrázek 7.20). Původní neorientovanou hranu $uv$ nahradíme dvojicí nových vrcholů $x$, $y$ a pěticí nových orientovaných hran. Přitom klíčové je omezení kapacity $c$ pro hranu $xy$, ostatní čtyři orientované hrany mohou mít kapacitu $c$ nebo vyšší. Symbol $\infty$ může v konkrétní implementaci reprezentovat fakt, že kapacita hrany není omezena. Nyní tok pro každou komoditu bude dopravován orientovanou hranou $xy$ *v tomto směru* a zabráníme výše popsanému překročení kapacity.

### Minimální kapacity hran

Ačkoliv to nebylo výslovně řečeno, tak existuje-li nějaká orientovaná cesta ze zdroje do stoku s kladnými kapacitami hran, tak úloha nalezení největšího toku má vždy řešení. Pokud však kromě maximálních kapacit hran stanovíme i minimální kapacity hran, tj. budeme požadovat, aby tok hranou měl nějakou nenulovou hodnotu, tak taková úloha obecně *nemusí* mít řešení. Minimální tok požadujeme například ve vodovodním řádu nebo v bezdrátové síti pro udržení spojení. Hledání největšího toku v grafu s omezenými maximálními i minimálními kapacitami hran opět přesahuje rámec textu, zájemcům doporučíme například [2].

## Pojmy k zapamatování

— superzdroj a superstok
— kapacity vrcholů
— víceproduktové toky

## 7.4 Další aplikace algoritmu pro hledání největšího toku

### Průvodce studiem


*Rozmanitost aplikací algoritmu pro hledání největšího toku je překvapivá. V této sekci ukážeme několik odlišných problémů, které budou mít jednu věc společnou: pro jejich řešení využijeme úlohu hledání největšího toku.*

*Ukážeme, jak najít pomocí největšího toku najít největší párování v bipartitním grafu, dokážeme Mengerovy větu, které jsme bez důkazu vyslovili na straně 44, budeme hledat systém různých reprezentantů v systému množin a dokážeme Hallovu větu.*

### Cíle

Po prostudování této sekce budete schopni:

- najít největší párování v bipartitním grafu,
- vyřešit přiřazovací úlohu,
- zjistit zda existuje a najít systém různých reprezentantů daného množinového systému.

#### Největší párování v bipartitním grafu

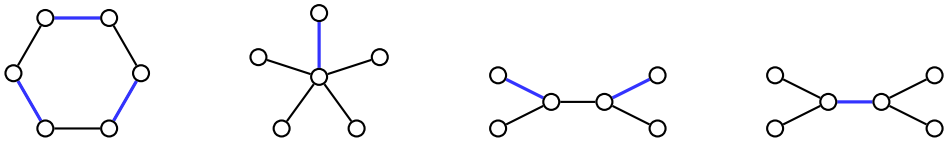
Bipartitní grafy patří mezi základní třídy grafů (definice je na straně 9 v Sekci 1.2). Připomeňme, že bipartitní grafy mají vrcholy rozdělené do dvou množin (partit) a hrany bipartitního grafu se vyskytují pouze mezi vrcholy z různých partit. Žádná hrana se nevyskytuje mezi dvěma vrcholy stejné partity.

V praxi se bipartitní grafy přirozeně objeví například při popisu přiřazovací úlohy (nazývá se také přiřazovací problém): máme několik pracovníků nebo strojů a máme množinu úkolů, které je nutno vykonat. Každý pracovník může v dané chvíli pracovat jen na jednom úkolu a navíc mohou mít různí pracovníci různé kvalifikace nebo možnosti a každý tak může provádět jen některé požadované úkoly. Cílem je zjistit, komu který úkol pro danou chvíli přiřadíme tak, aby bylo co nejvíce úkolů přiděleno a řešeno.

V teorii grafů popíšeme přiřazovací úlohu pomocí bipartitního grafu: jednu partitu budou tvořit pracovníci a druhou úkoly. Hranou spojíme každého pracovníka s těmi úkoly, které může vykonat. Samotné přidělení úkolů je pak popsáno množinou nezávislých hran, tedy takovou množinou hran $M$, že žádné dvě hrany v $M$ nemají společný vrchol. Této množině se říká „párování“, což pěkně vystihuje podstatu úlohy: sestavujeme dvojice typu „zaměstnanec, úkol“.

**Definice 7.15.** *Párování* $M$ v (bipartitním) grafu $G$ je podmnožina nezávislých hran $M \subseteq E(G)$ (žádné dvě hrany z $M$ nemají společný koncový vrchol).
Říkáme, že párování $M$ *pokrývá* vrchol $v$, jestliže $v$ je koncovým vrcholem nějaké hrany v párování $M$.

Cílem je najít největší párování, což je párování s největším počtem hran. Největší párování je řešením přiřazovací úlohy, tj. víme jak přidělit úkoly tak, aby co nejvíce úkolů bylo v řešení.



Obrázek 7.21 Párování v bipartitních grafech je vyznačen modře.

Na Obrázku 7.21 vidíme příklady párování v různých bipartitních grafech. Zatímco v prvním grafu existuje párování, jehož hrany pokrývají všechny vrcholy, tak pro druhý graf (hvězdu) má párování nejvýše jednu hranu. Ve třetím grafu je vyznačeno největší párování, protože žádné párování v daném grafu nemůže mít více než dvě hrany. Párování vyznačené ve čtvrtém grafu není největší, ale je *maximální*, protože do tohoto párování už nemůžeme přidat žádnou nezávislou hranu.

Ukážeme, jak převést úlohu hledání největšího párování na úlohu hledání největšího toku.

**Algoritmus 7.2 (Nalezení největšího bipartitního párování).** *Mějme bipartitní graf $G$ s vrcholovou množinou rozdělenou do partit $U$, $W$.*

1. *Sestrojíme síť $S$: do bipartitního grafu přidáme zdroj $z$ a stok $s$, všechny vrcholy z $U$ spojíme se zdrojem $z$ a všechny vrcholy z $W$ spojíme s stokem $s$, dále všechny hrany sítě $S$ orientujeme od zdroje do stoku a přiřadíme jim kapacity* 1,
2. *najdeme (celočíselný) největší tok v $S$ Algoritmem 7.1,*
3. *největší párování $M$ v grafu $G$ obsahuje všechny ty hrany původního grafu $G$, které mají nenulový tok v síti $S$.*

Když uvedený postup najde tok o velikosti $k = \|f\|$, tak to současně znamená, že největší párování má $k$ hran.

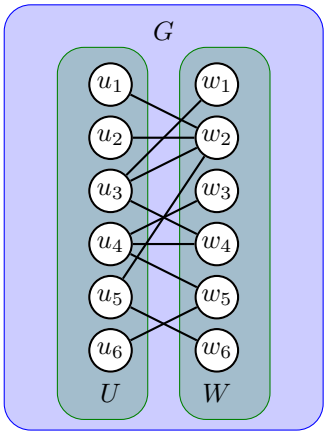
**Věta 7.16.** *Algoritmus 7.2 najde největší párování daného bipartitního grafu.*

*Důkaz.* Podle Důsledku 7.11 bude největší tok celočíselný, tj. každou hranou „poteče“ buď tok 0 nebo tok 1.

Označme $M$ množinu hran mezi partitami $U$ a $W$, které mají nenulový tok. Do každého vrcholu v partitě $U$ vede pouze jedna hrana s kapacitou 1, proto bude každý vrchol partity $U$ patřit nejvýše jedné hraně v množině $M$. Podobně každý vrchol množiny $W$ bude patřit nejvýše jedné hraně v nalezené množině $M$. To znamená, že hrany v množině $M$ jsou nezávislé (žádné dvě nesdílí koncové vrcholy) a odpovídající hrany původního bipartitního grafu tvoří párování.

Zbývá ukázat, že toto párování je největší. To ukážeme sporem: kdyby nalezené párování nebylo největší, tak tok nalezený Algoritmem 7.1 nemohl být největší, neboť z každého párování s $k$ hranami v síti získané konstrukcí v Algoritmu 7.2 lze najít odpovídající tok o velikosti $k$. □

Celý postup konstruktivního důkazu si ukážeme na příkladu.




Obrázek 7.22 Bipartitní graf $G$.

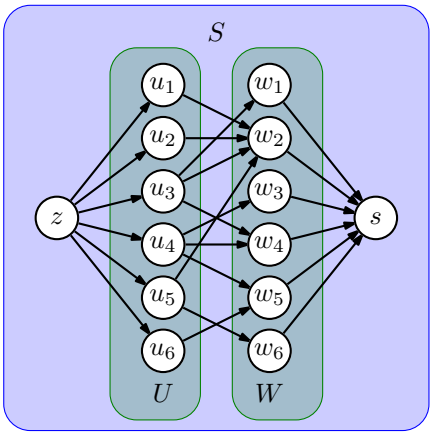
**Příklad 7.17.** Nejděte největší párování v bipartitním grafu $G$ na Obrázku 7.22.

*Řešení.* Nejprve sestrojíme příslušnou síť $S$:

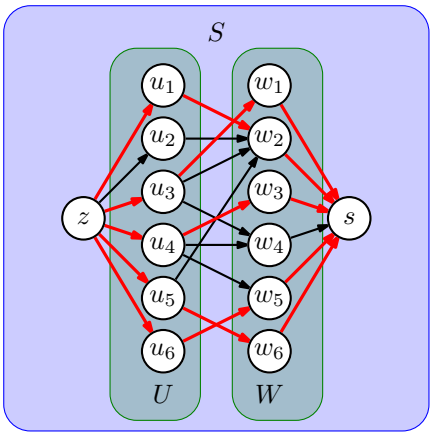
- do bipartitního grafu $G$ přidáme zdroj $z$ a stok $s$,
- zdroj $z$ spojíme hranou se všemi vrcholy v partitě $U$ a všechny vrcholy z partity $W$ spojíme se stokem $s$,
- všechny hrany sítě $S$ orientujeme od zdroje do stoku a přiřadíme jim kapacity 1. (Protože všechny hrany mají stejnou kapacitu 1, nemusíme kapacity do grafu značit.)

Dostaneme síť na Obrázku 7.23.

V nově sestavené síti $S$ najdeme největší tok pomocí Algoritmu 7.1. Podle Důsledku 7.11 víme, že tento tok bude celočíselný (tok každou hranou bude 0 nebo 1). Získaný tok je na Obrázku 7.24. Hrany s nenulovým tokem jsme pro přehlednost zvýraznili červeně.




Obrázek 7.23 Síť $S$, která vznikne přidáním zdroje, stoku, a orientovaných hran s kapacitou 1.



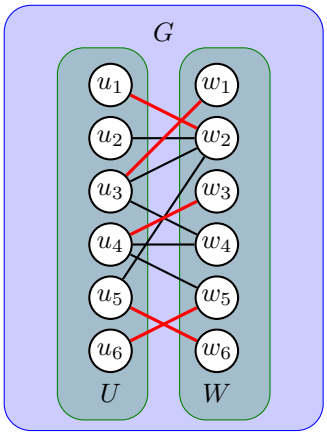

Obrázek 7.24 Největší tok v síti $S$ je vyznačen červeně.

Hledané párování obsahuje jen ty hrany původního grafu $G$, které odpovídají orientovaným hranám s nenulovým tokem v síti $S$. Největší párování v grafu $G$ je vyznačeno na Obrázku 7.25.

▲

## Pro zájemce:

Párování má smysl hledat i v obecných grafech, nikoliv jen v bipartitních grafech. Jedná se o složitější úlohu, která přesahuje rámec tohoto textu. Zájemci se mohou dočíst více v [5].




Obrázek 7.25 Největší párování v grafu $G$.

### Vyšší stupně souvislosti

V kapitole 2.3 jsme popsali, jak lze měřit vyšší stupně souvislosti. Bez důkazu jsme uvedli Mengerovy věty:

**Věta 2.26 (Mengerovy věty).** Netriviální graf $G$ je hranově $k$-souvislý právě tehdy, když mezi libovolnými dvěma vrcholy existuje alespoň $k$ po dvou hranově disjunktních cest (vrcholy mohou být sdílené).
Netriviální graf $G$ je vrcholově $k$-souvislý právě tehdy, když mezi libovolnými dvěma vrcholy existuje alespoň $k$ po dvou interně disjunktních cest (koncové vrcholy jsou společné).

Nyní pomocí algoritmu hledání největšího toku druhou větu dokážeme. Nejprve si všimneme, že podle definice vrcholové souvislosti platí:

**Lemma 7.18.** *Nechť $u$, $v$ jsou dva různé vrcholy grafu $G$ a $k > 0$ je přirozené číslo. Mezi vrcholy $u$ a $v$ v grafu $G$ existuje alespoň $k$ interně disjunktních cest právě tehdy, když po odebrání libovolných $k-1$ vrcholů různých od $u$, $v$ z grafu $G$ zůstanou vrcholy $u$ a $v$ ve společné komponentě souvislosti.*

*Důkaz.* Jedná se o ekvivalenci, proto dokážeme obě implikace.
"⇒" Jestliže mezi vrcholy $u$ a $v$ v grafu $G$ existuje alespoň $k$ interně disjunktních cest, tak odebráním $k-1$ vrcholů odstraníme nejvýše $k-1$ různých cest z $u$ do $v$ a vrchol $v$ zůstane dosažitelný po alespoň jedné cestě z vrcholu $u$. To znamená, že $u$ i $v$ patří do stejné komponenty souvislosti.
"⇐" Mějme graf $G$ a v něm zvolené dva vrcholy $u$ a $v$. Označíme například vrchol $u$ za zdroj a vrchol $v$ za stok a každé hraně přiřadíme kapacitu 1. Pomocí Algoritmu 7.1 najdeme největší tok z $u$ do $v$.

Dále sporem ukážeme, že velikost toku je alespoň $k$. Vskutku, pokud by největší tok měl hodnotu menší než $k$, tak kapacita nejmenšího řezu by byla podle

Věty 7.7 také menší než $k$. Odebráním libovolného koncového vrcholu každé hrany řezu dostaneme nesouvislý graf, přičemž odebraných vrcholů je méně než $k$, což podle předpokladu není možné.

To znamená, že největší tok má hodnotu alespoň $k$ a proto všechny hrany s tokem 1 tvoří různé (interně disjunktní) cesty z $u$ do $v$ v grafu $G$. □

Z Lemmatu 7.18 nyní snadno dokážeme druhou Mengerovu větu. Vrcholy $u$ a $v$ můžeme zvolit v grafu libovolně a proto graf $G$ je $k$-souvislý podle definice $k$-souvislosti, neboť odebráním libovolných $k-1$ vrcholů zůstane výsledný graf souvislý.

První Mengerova věta se ukáže podobně.

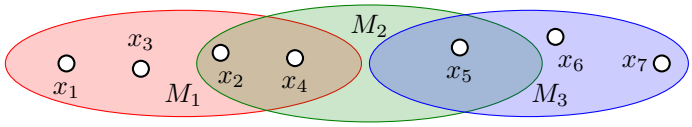
### Systém různých reprezentantů

Sestavení systému různých reprezentantů je úloha zobecněním přiřazovacího problému. Mějme nějakou množinu $X$ a systém jejích podmnožin $M_1, M_2, \ldots, M_k \subseteq X$. Ptáme se, zda je možné v každé podmnožině $M_i$ vybrat jeden prvek, určitého reprezentanta tak, aby každá množina měla jiného reprezentanta.

Například pokud $X$ představuje množinu dárců pro transplantaci ledviny a $M_i$ je taková podmnožina množiny dárců $X$, kteří jsou vhodným dárcem pro pacienta $i$ ($i = 1, 2, , \ldots, , k$), tak nalezení systému reprezentantů odpovídá nalezení vhodného dárce pro každého pacienta.

Systém různých reprezentantů nadefinujeme pečlivě:

**Definice 7.19.** Nechť $M_1, M_2, \ldots, M_k$ jsou neprázdné množiny. *Systémem různých reprezentantů* systému množin $\{M_1, M_2, \ldots, M_k\}$ rozumíme takovou posloupnost $(m_1, m_2, \ldots, m_k)$ *po dvou navzájem různých* prvků, že $m_i \in M_i$ pro každé $i = = 1, 2, \ldots, k$. Říkáme, že prvek $m_i$ je reprezentant množiny $M_i$.

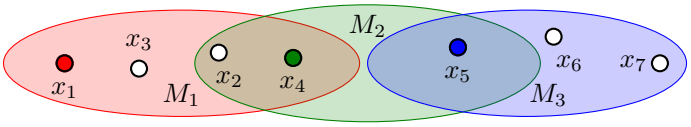



Obrázek 7.26 Množiny $M_1$, $M_2$, $M_3$.

**Příklad 7.20.** Najděte systém různých reprezentantů pro množiny na Obrázku 7.26.

*Řešení.* Stačí zvolit například prvek $x_1$ jako reprezentanta množiny $M_1$, prvek $x_4$ jako reprezentanta množiny $M_2$ a prvek $x_5$ jako reprezentanta množiny $M_3$ (Obrázek 7.27). Protože $(m_1, m_2, m_3) = (x_1, x_4, x_5)$ obsahuje navzájem různé prvky a platí $x_1 \in M_1$, $x_4 \in M_2$, $x_5 \in M_3$, tak se jedná o platný systém různých reprezentantů daného systému množin.

▲




Obrázek 7.27 Systém různých reprezentantů množin $M_1$, $M_2$, $M_3$.

Ale ne každý systém množin má svůj systém různých reprezentantů, jak ukazuje následující příklad.

**Příklad 7.21.** Najděte systém různých reprezentantů systému množin $M_1 = \{1,5,9\}$, $M_2 = \{2,6,7\}$, $M_3 = \{3,4,8\}$, $M_4 = \{1,6,8\}$, $M_5 = \{2,4,9\}$, $M_6 = \{3,5,7\}$, $M_7 = \{1,4,7\}$, $M_8 = \{2,5,8\}$, $M_9 = \{3,6,9\}$, $M_{10} = \{1,2,3\}$, $M_{11} = \{4,5,6\}$ a $M_{12} = \{7,8,9\}$.

*Řešení.* Systém různých reprezentantů pro daných dvanáct množin neexistuje. Stačí si všimnout, že všechny množiny obsahují některé z devíti čísel $1, 2, , \ldots, 9$, a proto mezi nimi nemůžeme najít dvanáct různých reprezentantů. Zadaná úloha nemá řešení. ▲

Ale pozor, ani pokud je různých prvků více než množin, tak příslušný systém různých reprezentantů nemusí existovat. Jak poznáme, zda systém různých reprezentantů existuje? Stručně řečeno: těžko – výpočet je náročný, pro obecný případ není znám polynomiální algoritmus pro nalezení nebo ověření systému různých reprezentantů.

Důležitým výsledkem je Hallova věta (čti „Hólova“), která udává nutnou a dostatečnou podmínku pro to, aby systém různých reprezentantů existoval:

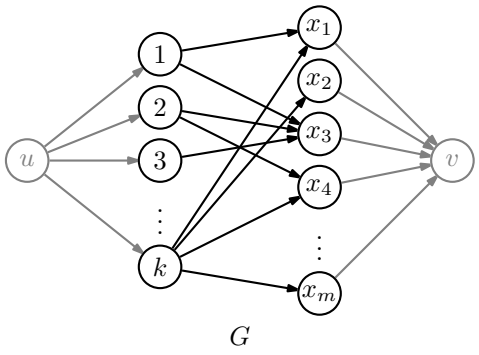
**Věta 7.22 (Hallova věta).** *Nechť $M_1, M_2, \ldots, M_k$ pro $k > 0$ jsou neprázdné množiny. Pro systém množin $\{M_1, M_2, \ldots, M_k\}$ existuje systém různých reprezentantů právě tehdy, když platí*

$$\forall J \subset \{1,2,\ldots,k\} : \left|\bigcup_{j\in J} M_j\right| \geqq |J|,$$

*neboli pokud sjednocení libovolné skupiny $j$ množin má alespoň $j$ prvků (tedy tolik prvků, kolik množin je sjednoceno).*

*Důkaz.* K důkazu opět využijeme Algoritmus 7.1.

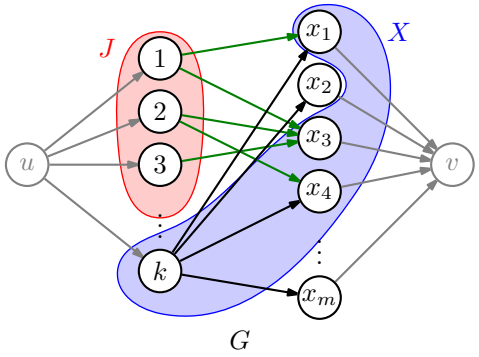
Označme $x_1, x_2, \ldots, x_m$ po řadě všechny prvky ve sjednocení $M_1 \cup M_2 \cup \ldots \cup M_k$. Definujeme bipartitní graf $G$ na množině vrcholů s partitami $\{1,2,\ldots,k\}$ a $\{x_1, x_2, \ldots, x_m\}$ (Obrázek 7.28). Dále přidáme nové vrcholy $u$ a $v$. Navíc přidáme hrany $\{u,i\}$ pro $i = 1,2,\ldots,k$, hrany $\{x_j, v\}$ pro $j = 1,2,\ldots,m$ a hrany $\{i, x_j\}$ jestliže $x_j \in M_i$. Všechny hrany zorientujeme „zleva doprava“. Dostaneme síť $S$ na Obrázku 7.28 se zdrojem $u$ a stokem $v$. Konstrukce sítě $S$ je obdobná konstrukci sítě v Algoritmu 7.2.




Obrázek 7.28 Graf pro hledání systému různých reprezentantů.

Každá cesta mezi $u$ a $v$ má tvar $u, i, x_j, v$ a každá taková cesta ukazuje reprezentanta $x_j$ množiny $M_i$. Systém různých reprezentantů odpovídá $k$ interně disjunktním cestám mezi $u$ a $v$.

Nechť $X$ je nyní libovolná minimální množina vrcholů v $G$, po jejímž odebrání z grafu nezůstane žádná cesta mezi $u$ a $v$. Podle Lemmatu 7.18 existuje pro systém množin $\{M_1, M_2, \ldots, M_k\}$ systém různých reprezentantů právě tehdy, když každá taková oddělující množina $X$ má alespoň $k$ prvků. Nyní položme $J = \{1, 2, \ldots, k\} \setminus X$. Na Obrázku 7.29 je množina $J$ znázorněna červeně a množina $X$ modře.




Obrázek 7.29 Rozbor grafu pro hledání systému různých reprezentantů.

Je-li $J = \emptyset$, tak tvrzení je splněno triviálně. Pro $J \neq \emptyset$ vede každá hrana z vrcholů v množině $J$ (hrany mezi $u$ a vrcholy v $J$ neuvažujeme) do vrcholů z $X \cap \{x_1, \ldots, x_m\}$, jinak by existovala cesta mezi $u$ a $v$ a $X$ by nebyla oddělující množinou vrcholů. Dále je zřejmé, že do každého vrcholu v $X \cap \{x_1, x_2, , \ldots, x_m\}$ vede hrana z nějakého vrcholu v množině $J$. Proto platí

$$\left|\bigcup_{j\in J} M_j\right| = |X \cap \{x_1, \ldots, x_m\}| = |X| - |X \cap \{1, \ldots, k\}| = |X| - k + |J|.$$

Vidíme, že $|X| \geqq k$ pro všechny (minimální) volby oddělující množiny $X$ právě tehdy, když $\left|\bigcup_{j\in J} M_j\right| \geqq |J|$ pro jakoukoli množinu $J$, což je dokazované tvrzení. □

## Pojmy k zapamatování

— párování v bipartitním grafu
— systém různých reprezentantů
— Hallova věta

# Autorská práva

Při sestavování textu jsem se snažil důsledně dodržovat autorská práva. Většinu obrázků grafů jsem připravil sám v METAFONTU nebo v METAPOSTU. Oba programy jsou volně šiřitelné součásti řady distribucí operačního systému Linux.

Všechny fotografie, které jsem stáhnul z internetu (většinou z Wikipedie), buď mají na Wikipedii výslovně uvedeno, že jsou jejich autory volně dány k dispozici, nebo se jedná o reprodukce děl staších než 70 let. Novější fotografie budov, plaket a míst je možno použít ve smyslu § 33 odst. 1 autorského zákona, který říká „*Do práva autorského nezasahuje ten, kdo kresbou, malbou nebo grafikou, fotografií nebo filmem nebo jinak zaznamená nebo vyjádří dílo, které je trvale umístěno na náměstí, ulici, v parku, na veřejných cestách nebo na jiném veřejném prostranství; do autorského práva nezasahuje ani ten, kdo takto vyjádřené, zachycené nebo zaznamenané dílo dále užije. Je-li to možné, je nutno uvést jméno autora, nejde-li o dílo anonymní, nebo jméno osoby, pod jejímž jménem se dílo uvádí na veřejnost, a dále název díla a umístění.*“

Následuje seznam fotografií a obrázků i s uvedením zdrojů, ze kterých jsem čerpal.

- Eisenhoverův systém dálnic, str. 41; http://en.wikipedia.org/wiki/File:Map_of_current_Interstates.svg
- optimální řešení problému obchodního cestujícího pro 13 509 měst, str. 56; http://www.cs.princeton.edu/courses/archive/spring11/cos126/checklist/tsp.html
- Hamiltonova hra, str. 59; http://puzzlemuseum.com/month/picm02/200207icosian.htm, způsob použití je v souladu s podmínkami na stránce http://puzzlemuseum.com/faqs/status&obj.htm
- evoluční strom, str. 82; http://en.wikipedia.org/wiki/File:MyosinUnrootedTree.jpg
- rodokmen, str. 82; http://cs.wikipedia.org/wiki/Soubor:Cesky_Sternberk_Castle_CZ_family_tree_116.jpg
- bludiště, str. 106; http://en.wikipedia.org/wiki/Maze_generation_algorithm
- mapa ropovodů a plynovodů v Evropě, str. 126; http://aktualne.centrum.cz/zahranici/evropa/clanek.phtml?id=626363
- řeka Punkva, str. 131; http://cs.wikipedia.org/wiki/Punkva

# Rejstřík

# Literatura


[1] J. Černý, *Základní grafové algoritmy*, MFF UK, [online] (2010) [cit. 22.7.2012]. http://kam.mff.cuni.cz/~kuba/ka/

[2] J. Demel, Grafy, SNTL, Praha, (1989).

[3] D. Fronček, *Úvod do teorie grafů*, Slezská univerzita Opava, (1999), ISBN 80-7248-044-8.

[4] P. Hliněný, *Diskrétní matematika*, VŠB–TU Ostrava, skriptum (2006).

[5] P. Kovář, *Teorie grafů*, VŠB–TU Ostrava, skriptum (2012).

[6] M. Kubesa, *Základy diskrétní matematiky*, VŠB–TU Ostrava, skriptum (2012).

[7] J. Matoušek, J. Nešetřil, *Kapitoly z diskrétní matematiky*, Karolinum Praha, (2000), ISBN 80-246-0084-6.

[8] K.H. Rosen, *Discrete Mathematics and Its Applications – 6th ed.*, McGraw-Hill, New York NY, (2007), ISBN-10 0-07-288008-2.

[9] D.B. West, *Introduction to graph theory – 2nd ed.*, Prentice-Hall, Upper Saddle River NJ, (2001), ISBN 0-13-014400-2.

# Přehled použitých symbolů

| | |
|---|---|
| $A(G)$ | matice sousednosti grafu $G$ (str. 25) |
| $B(G)$ | incidenční matice grafu $G$ (str. 26) |
| $\|C\|$ | kapacita řezu $C$ (str. 132) |
| $\deg(v)$ | stupeň vrcholu $v$ v grafu (str. 10) |
| $\mathrm{dist}(u, v)$ | vzdálenost vrcholů $u$ a $v$ v grafu (str. 63) |
| $E(G)$ | množina hran grafu $G$ (str. 2) |
| $\|f\|$ | velikost toku $f$ (str. 128) |
| $h(G)$ | počet hran grafu $G$ (str. 115) |
| $f(G)$ | počet oblastí grafu $G$ (str. 115) |
| $V(G)$ | množina vrcholů grafu $G$ (str. 2) |
| $v(G)$ | počet vrcholů grafu $G$ (str. 115) |
| $\delta(G)$ | nejmenší stupeň v grafu $G$ (str. 10) |
| $\Delta(G)$ | největší stupeň v grafu $G$ (str. 10) |
| $\chi(G)$ | chromatické číslo (barevnost) grafu $G$ (str. 110) |
| $O(n)$ | horní odhad složitosti algoritmu (str. 22) |
| $G \simeq H$ | isomorfní grafy $G$ a $H$ (str. 20) |
| $G \cup H$ | sjednocení grafů $G$ a $H$ |
| | |
| $\prod_{i=1}^{n} a_i$ | součin prvků posloupnosti $(a_i)_{i=1}^{n}$ |
| $\sum_{i=1}^{n} a_i$ | součet prvků posloupnosti $(a_i)_{i=1}^{n}$ |
| | |
| $\emptyset$ | prázdná množina |
| $x \in A$ | prvek $x$ množiny $A$ |
| $A \subseteq A$ | podmnožina $A$ množiny $B$ |
| $A \subset A$ | vlastní podmnožina $A$ množiny $B$ |
| $A \cap B$ | průnik množin $A$ a $B$ |
| $A \cup B$ | sjednocení množin $A$ a $B$ |
| $A \setminus B$ | rozdíl množin $A$ a $B$ |
| $A \triangle B$ | symetrická diference množin $A$ a $B$ |
| $A \times B$ | kartézský součin množin $A$ a $B$ |
| $\simeq$ | relace ekvivalence |
| $A \wedge B$ | logická konjunkce („$A$ a současně $B$“) |
| $A \vee B$ | logická disjunkce („$A$ nebo $B$“) |
| $A \Rightarrow B$ | implikace („jestliže platí $A$, tak platí $B$“) |
| $A \Leftrightarrow B$ | ekvivalence („$A$ platí právě, když $B$“) |
| | |
| $C_n$ | cyklus na $n$ vrcholech (str. 7) |
| $P_n$ | cesta na $n$ vrcholech (str. 8) |
| $K_n$ | kompletní graf na $n$ vrcholech (str. 7) |
| $K_{m,n}$ | kompletní bipartitní graf na $m + n$ vrcholech (str. 9) |