Teorie Grafů

pro (nejen) informatiky

Prof. RNDr. Petr Hliněný, Ph.D.

hlineny@fi.muni.cz

22. září 2016

Obsáhlý úvod do většiny základních oblastí teorie grafů na magisterské úrovni studia, s (přiměřeným) důrazem na algoritmické a informatické aplikace a doplněný procvičením látky a úlohami k samostatnému řešení.

Základní český výukový text pro předmět MA010 na FI MU od roku 2006.

Verze 2.00-alfa (2016)
© 2005–2016 Petr Hliněný, FI MU.

Předmluva

Vážení čtenáři,

dostává se vám do rukou výukový text Teorie grafů, který je primárně zaměřený pro potřeby studentů informatických oborů na vysokých školách (je ale dobře přístupný i ne-informatikům). Hloubkou teoretického záběru je tento text směřován do magisterské úrovně studia, ale při vynechání některých více teoretických pasáží dobře poslouží i jako základní přehled oblasti na bakalářské úrovni.

Diskrétní matematika a Teorie grafů jako její prominentní součást jsou moderními a rychle se rozvíjejícími oblastmi matematiky, které mají mnohé úzké vazby na teoretickou i aplikovanou informatiku. Kořeny diskrétní matematiky (kombinatoriky) sahají k velkým matematikům 18. a 19. století, z nichž si zaslouží jmenovat především Leonard Euler. Byl to však až nástup počítačů a rozvoj informatiky jako vědní disciplíny v druhé polovině 20. století, které přinesly nové teoretické pojmy a otázky a vtiskly tak diskrétní matematice její moderní tvář, spojenou dosti specificky s pojmy relací a grafů.

Náš text vás seznámí jak s přehlednými základy klasické teorie grafů, tak i s hlavními grafovými pojmy užitečnými pro aplikace, především ty informatické, a závěrem rozvede některé zajímavé nové směry vývoje. Je koncipován jako soběstačný celek, který od čtenáře nevyžaduje o mnoho více než běžné středoškolské matematické znalosti, k tomu základní formalismy matematického dokazování (matematickou indukci) a diskrétní matematiky (množiny, relace, kombinatorické počítaní) a chuť do studia. Části věnované algoritmickým aplikacím navíc předpokládají jistou znalost algoritmizace a zběhlost v programování. Svým rozsahem látka odpovídá jednomu semestru běžné VŠ výuky včetně cvičení.

Tento výukový text vychází ve svých základech z vynikající moderní učebnice [5] Kapitoly z Diskrétní Matematiky [5] autorů J. Matouška a J. Nešetřila z MFF UK. (Tato kniha se kromě českého vydání dočkala i úspěšných vydání ve světových jazycích, jako například [6]. Vřele ji doporučujeme všem zájemcům o rozšíření svých diskrétních matematických obzorů.) Záběr našeho textu není v oblasti diskrétní matematiky tak široký jako u zmíněné učebnice, ale zaměřujeme se hlouběji na ty různé oblasti grafů, které nacházejí nejčastější použití v informatické praxi. Na druhou stranu se zde více věnujeme abstraktním algoritmickým aplikacím uváděné látky a zmiňujeme také některé zajímavé pokročilé partie teorie grafů.

Ve stručnosti se zde zmíníme o struktuře naší učebnice. Přednášený materiál je dělený do jednotlivých lekcí (kapitol), které zhruba odpovídají obsahu týdenních přednášek v semestru. Lekce jsou dále děleny tematicky na oddíly. Výklad je strukturován obvyklým matematickým stylem na definice, tvrzení, případné důkazy, poznámky a neformální komentáře. Lekce obsahují někdy dodatky přinášející specializovaný pohled do hlubin vybraného tématu a ty jsou obecně volitelné při studiu. Každá lekce má v samém závěru uvedeny bohaté odkazy na případné rozšiřující (samo)studium.

Výukový text je dále proložen řadou vzorově řešených příkladů navazujících na vykládanou látku a doplněn dalšími otázkami a úlohami. (Otázky a úlohy označené hvězdičkou patří k obtížnějším a nepředpokládáme, že by na ně všichni studující dokázali správně odpovědět bez pomoci.) Každá lekce je zakončena zvláštním oddílem určeným k dodatečnému procvičení látky. Správné odpovědi k otázkám a úlohám jsou shrnuty na konci textu. Věříme, že příklady vám budou užitečným vodítkem při studiu a pro pochopení přednesené teorie. Mnoho zdaru.

O výuce Teorie grafů MA010 na FI MU

Teorie grafů MA010 je jedním ze stěžejních teoretických magisterských předmětů na Fakultě informatiky MU, ale jeho výuka je bez problémů přístupná i bakalářským studentům se zájmem o obor. Paralelně k MA010 je na FI MU vyučován předmět Grafové algoritmy MA015. Autor pak dále přednáší několik navazujících výběrových předmětů pokročilé teorie grafů – MA051, MA052 a MA053.

V rámci e-learningu Informačního systému (IS) MU je k předmětu vytvořena obsáhlá interaktivní osnova Teorie grafů (v anglickém jazyce), odpovídající struktuře výukového textu a s mnoha odkazy na přístupné webové zdroje k předmětu a procvičování. **Osnova** je veřejně přístupná na adrese

`http://is.muni.cz/el/1433/podzim2016/MA010/index.qwarp`.

O historii našeho učebního textu

Původní výukový text vznikal v průběhu let 2003 až 2005 na základě autorových přednášek a cvičení předmětu Diskrétní matematika na FEI VŠB – TUO. Tento text byl pak autorem upraven a druhotně (především v teoretické oblasti) výrazně rozšířen pro potřeby výuky magisterského předmětu MA010 Teorie grafů na FI MU Brno od roku 2006.

Vzhledem k tomu, že od roku 2009 je předmět MA010 na FI MU vyučován v angličtině, tento učební text poněkud ztrácí své výsadní postavení primárního studijního zdroje (kterým se stává výše odkázaná interaktivní osnova MA010 v IS MU), ale zůstává plnohodnotným českým studijním materiálem pro předmět MA010.

A slovo závěrem...

Přejeme vám mnoho úspěchů při studiu teorie grafů a budeme potěšeni, pokud se vám náš výukový text bude líbit. Jelikož nikdo nejsme neomylní, i v této publikaci zajisté jsou nejasnosti či chyby, a proto se za ně předem omlouváme. Pokud chyby objevíte, dejte nám prosím vědět e-mailem `mailto:hlineny@fi.muni.cz`.

Autor

Obsah

Část I
Základy Grafů a Jejich Využití

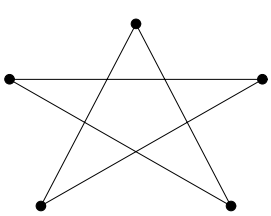
V úvodní části našeho učebního textu se nejprve seznámíme s grafy a naučíme se je pochopit a pracovat s nimi jako s matematickými objekty. V následujících třech lekcích se zaměříme na několik okruhů grafových pojmů a úloh, se kterými se asi nejčastěji setkáváme v jejich aplikacích souvisejících s informatikou.

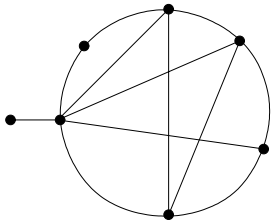
1 Jak Vzniká GRAF

Úvod

Třebaže grafy jsou jen jednou z mnoha používaných struktur v diskrétní matematice, vydobyly si svou užitečností a názorností tak důležité místo na slunci, až se dá bez nadsázky říci, že teorie grafů je nejvýznamnější součástí soudobé diskrétní matematiky. Primárně se dobrá znalost základů teorie grafů jeví nezbytná ve většině oblastí moderní informatiky, včetně těch aplikovaných. Proto se i náš všeobecný kurz teorie grafů bude v důležité míře zaměřovat na informatické aplikace grafů, avšak vcelku bez nutnosti informatiku ovládat (tj. naše látka je dobře přístupná i ne-informatikům).

Neformálně řečeno, graf se skládá z vrcholů (představme si je jako nakreslené puntíky) a z hran, které spojují dvojice vrcholů mezi sebou. Ale pozor, nepleťme si graf s grafem funkce!





Takto chápaný graf může vyjadřovat souvislosti mezi objekty, návaznosti, spojení nebo toky, atd. Své důležité místo v informatice si grafy získaly dobře vyváženou kombinací svých vlastností – snadným názorným nakreslením a zároveň jednoduchou zpracovatelností na počítačích. Díky těmto vlastnostem se grafy prosadily jako základní vhodný matematický model pro popis různých, i komplikovaných, vztahů mezi objekty.

Cíle

Prvním cílem této lekce je formálně definovat, co je to graf a jaké jsou nejzákladnější grafové pojmy (třeba hrany a stupně, podgrafy, apod). V tomto ohledu zaslouží specifickou zmínku pojem isomorfismu grafů, který je v úvodních partiích klíčový a je mu věnována značná pozornost. Také jsou zde shrnuty některé obvyklé typy grafů, jako cesty a kružnice, úplné grafy či stromy, a jejich orientovaná zobecnění. Především však je důležité, aby se čtenář naučil grafy "uchopit" a vykonávat s nimi základní operace a manipulace, což si také může ihned vyzkoušet v následujících oddílech cvičení.

1.1 Pojem grafu

Hned na úvod přistoupíme k formální definici grafu a přitom předesíláme, že soubor příkladů ukazujících praktické využití grafů je uveden v Oddíle 1.4 a dalších. Začneme základním pojmem tzv. obyčejného grafu, kterým se budeme zabývat v převážné většině tohoto textu a ve kterém zavádíme hrany mezi dvojicemi vrcholů pomocí dvouprvkových podmnožin těchto vrcholů. Možné rozšiřující definice grafů zmíníme krátce v závěru.

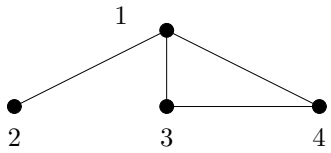
Definice 1.1. ***Graf*** (rozšířeně *obyčejný* či *jednoduchý neorientovaný* graf)
je uspořádaná dvojice $G = (V, E)$, kde V je množina *vrcholů* a E je množina *hran* – množina vybraných dvouprvkových podmnožin množiny vrcholů.

`Značení`: Hranu mezi vrcholy u a v píšeme jako $\{u, v\}$, nebo zkráceně uv. Vrcholy spojené hranou jsou *sousední*. Na množinu vrcholů známého grafu G odkazujeme jako na $V(G)$, na množinu hran $E(G)$.

`Fakt`: Na graf se lze dívat také jako na symetrickou ireflexivní relaci, kde hrany tvoří právě dvojice prvků z této relace.

`Poznámka`: Grafy se často zadávají přímo názorným obrázkem, jinak je lze také zadat výčtem vrcholů a výčtem hran. Například:

$$V = \{1, 2, 3, 4\}, \quad E = \Big\{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{3, 4\}\Big\}$$

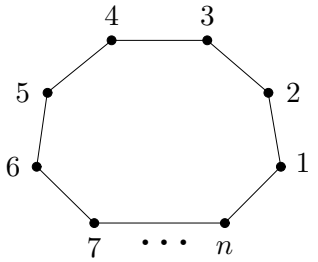


Běžné typy grafů

Pro snadnější vyjadřování je zvykem některé běžné typy grafů nazývat popisnými jmény. Jde čistě o věc konvence a autoři se mohou v některých názvech lišit (i přicházet s novými názvy), avšak následující názvy patří k všeobecně přijímaným základům grafového názvosloví.

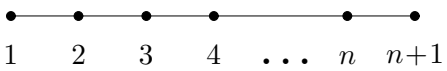
Kružnice délky n má $n \geq 3$ vrcholů spojených do jednoho cyklu n hranami:

C_n



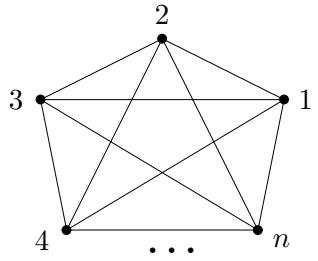
Cesta délky n má $n + 1$ vrcholů spojených za sebou n hranami:

P_n



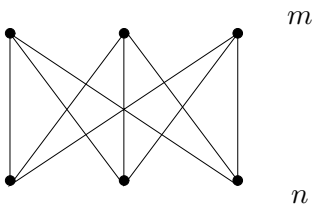
Úplný graf na $n \geq 1$ vrcholech má n vrcholů, všechny navzájem pospojované (tj. celkem $\binom{n}{2}$ hran):

K_n



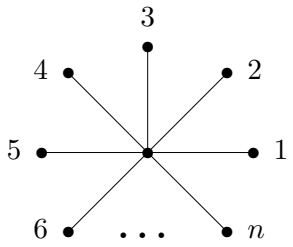
Úplný bipartitní graf na $m \geq 1$ a $n \geq 1$ vrcholech má $m+n$ vrcholů ve dvou skupinách (partitách), přičemž hranami jsou pospojované všechny $m \cdot n$ dvojice z různých skupin:

$K_{m,n}$



Hvězda s $n \geq 1$ rameny je speciálním případem úplného bipartitního grafu $K_{1,n}$:

S_n



Stupně vrcholů v grafu

Máme-li graf, často nás zajímá, kolik z kterého vrcholu vychází hran-spojnic, neboli kolik má vrchol „sousedů“. Proto jedním z prvních definovaným pojmů bude stupeň vrcholu v grafu.

Definice 1.2. ***Stupněm vrcholu*** v v grafu G rozumíme počet hran vycházejících z v. Stupeň vrcholu v v grafu G značíme $d_G(v)$.

`Komentář:` Slovo „vycházející“ zde nenaznačuje žádný směr; je totiž obecnou konvencí říkat, že hrana vychází z obou svých konců zároveň.

Například v nakreslené ukázce jsou stupně přímo zapsány u vrcholů.

`Definice`: Graf je *d-regulární*, pokud všechny jeho vrcholy mají stejný stupeň d.

`Značení`: *Nejvyšší* stupeň v grafu G značíme $\Delta(G)$ a *nejnižší* $\delta(G)$.

Věta 1.3. *Součet stupňů v grafu je vždy sudý, roven dvojnásobku počtu hran.*

`Důkaz`. Při sčítání stupňů vrcholů v grafu započítáme každou hranu dvakrát – jednou za každý její konec. Proto výsledek vyjde sudý. □

Vlastnosti posloupnosti stupňů

Soubor stupňů grafu je jeho důležitou kvantitativní charakteristikou. Jelikož však v obyčejném grafu zvlášť nerozlišujeme mezi jednotlivými vrcholy, nemáme dáno žádné

pořadí, ve kterém bychom stupně vrcholů měli psát (pokud je nepíšeme přímo do obrázku grafu). Proto si stupně obvykle seřazujeme podle velikosti.

Zajímavou otázkou je, které posloupnosti stupňů odpovídají vrcholům skutečných grafů? (Například posloupnost stupňů $1, 2, 3, 4, 5, 6, 7$ nemůže nastat nikdy.)

Věta 1.4. *Nechť $d_1 \leq d_2 \leq \cdots \leq d_n$ je posloupnost přirozených čísel. Pak existuje graf s n vrcholy stupňů*

$$d_1,\ d_2, \ldots,\ d_n$$

právě tehdy, když existuje graf s $n-1$ vrcholy stupňů

$$d_1,\ d_2, \ldots,\ d_{n-d_n-1},\quad d_{n-d_n}-1, \ldots,\ d_{n-2}-1,\ d_{n-1}-1\,.$$

`Komentář:` K pochopení Věty 1.4: Mírně nepřehledný formální zápis věty znamená, že ze seřazené posloupnosti odebereme poslední (největší) stupeň d_n a od tolika d_n bezprostředně předchozích stupňů odečteme jedničky. Zbylé stupně na začátku posloupnosti se nezmění. (Nezapomeňme, že posloupnost je třeba znovu seřadit dle velikosti.)

Důkaz tohoto nepříliš těžkého fundamentálního tvrzení pro tentokráte vynecháváme a odkazujeme čtenáře například na [5, Kapitola 4].

Příklad 1.5. *Existuje graf se stupni vrcholů:*

$(1, 1, 1, 2, 3, 4)$?

Dle Věty 1.4 upravíme posloupnost na $(1, 0, 0, 1, 2)$, uspořádáme ji $(0, 0, 1, 1, 2)$, pak znovu vše zopakujeme na $(0, 0, 0, 0)$ a takový graf už jasně existuje.

Na druhou stranu, jak takový graf sestrojíme? (Obvykle není jediný, ale nás zajímá aspoň jeden z nich.) Prostě obrátíme předchozí postup, začneme z grafu se 4 vrcholy bez hran, přidáme vrchol stupně 2 spojený se dvěma z nich a další vrchol stupně 4 takto:

$(1, 1, 1, 1, 2, 3, 4, 6, 7)$?

Podobně upravíme tuto posloupnost na $(1, 0, 0, 0, 1, 2, 3, 5)$ a uspořádáme ji $(0, 0, 0, 1, 1, 2, 3, 5)$, pak znovu upravíme na $(0, 0, -1, 0, 0, 1, 2)$, ale to už přece nelze – stupně v grafu nejsou záporné. Proto takový graf neexistuje.

□

Otázky a úlohy k řešení

(1.1.1) *Může být obyčejný graf bez hran?*

(1.1.2) *Kolik hran může mít obyčejný graf s jedním vrcholem?*

(1.1.3) *Graf zapíšeme výčtem vrcholů $\{a, b, c, d, e\}$ a hran $\{ac, ba, be, cd, de\}$. Nakreslete si jej! O jaký typ grafu se jedná?*

(1.1.4) *Pro jakou hodnotu n je úplný graf K_n zároveň cestou?*

(1.1.5) *Pro jakou hodnotu n je úplný graf K_n zároveň kružnicí?*

(1.1.6) *Pro jaké hodnoty $m, n > 0$ je úplný bipartitní graf $K_{m,n}$ zároveň kružnicí?*

(1.1.7) *Pro jaké hodnoty $m, n > 0$ úplný bipartitní graf $K_{m,n}$ neobsahuje žádnou kružnici?*

(1.1.8) *Kolik hran musíte přidat do kružnice délky 6, aby vznikl úplný graf na 6 vrcholech?*

(1.1.9) *Co se stane, pokud aplikujeme postup Věty 1.4 na posloupnost stupňů s lichým součtem?*

(1.1.10) *Existuje graf se stupni $4, 2, 3, 5, 3, 2, 3$? Nakreslete jej.*

(1.1.11) *Existují dva „různé“ grafy se 6 vrcholy stupňů 2?*

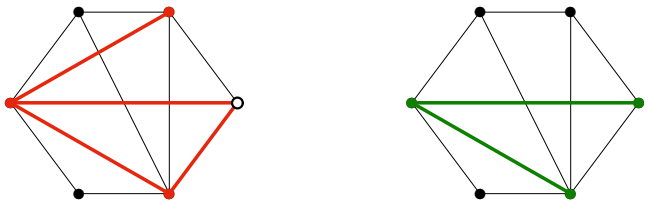
(1.1.12) *Proč má každý (jednoduchý) graf alespoň dva vrcholy stejného stupně?*

1.2 Podgrafy a Isomorfismus

Podgraf je, jednoduše řečeno, část grafu. Avšak tato slova musíme definovat poněkud přesněji, abychom předešli situacím, kdy by nám zbyly „hrany bez vrcholů“. Dále si definujeme tzv. indukovaný podgraf, který (na rozdíl od běžného podgrafu) z původního grafu přebírá *všechny* hrany mezi vybranými vrcholy.

`Definice`: *Podgrafem* grafu G rozumíme libovolný graf H na podmnožině vrcholů $V(H) \subseteq V(G)$, který má za hrany libovolnou podmnožinu hran grafu G majících oba vrcholy ve $V(H)$. Píšeme $H \subseteq G$, tj. stejně jako množinová inkluze (ale význam je trochu jiný).

> `Komentář`: Na následujícím obrázku vidíme zvýrazněné podmnožiny vrcholů hran. Proč se vlevo (červeně) nejedná o podgraf? Obrázek vpravo (zeleně) už podgrafem je.

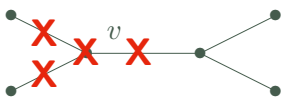


`Definice`: *Indukovaným podgrafem* je podgraf $H \subseteq G$ takový, který obsahuje všechny hrany grafu G mezi dvojicemi vrcholů z $V(H)$. *Překlenujícím podgrafem* je podgraf $H \subseteq G$ takový, že $V(G) = V(H)$.

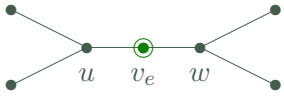
Elementární operace s grafy

Ke zjednodušení naší práce s grafy dobře poslouží také následující jednoduché operace, hojně v teorii grafů používané.

- *Vypuštění hrany e nebo vrcholu v* z grafu G je definováno:
  $G \setminus e :=$ indukovaný podgraf grafu G daný jako $E(G \setminus e) = E(G) \setminus \{e\}$,
  $G \setminus v :=$ indukovaný podgraf grafu G daný jako $V(G \setminus v) = V(G) \setminus \{v\}$.

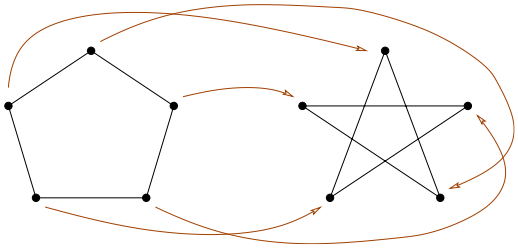


- Stejně tak lze definovat vypouštění množin vrcholů či hran z G: $G \setminus X$, $G \setminus F$.
- *Přidáním vrcholu či hrany* do grafu G, kde $v \notin V(G)$, $e = uw \notin E(G)$, vznikne:
  $G + v :=$ graf daný $V(G + v) = V(G) \cup \{v\}$ a $E(G + v) = E(G)$,
  $G + e :=$ graf daný $V(G + e) = V(G)$ a $E(G + e) = E(G) \cup \{e\}$,
  za předpokladu $u, w \in V(G)$.
- *Podrozdělením hrany* $e = uw \in E(G)$ v grafu G vzniká
  – graf $(G \setminus e) + v_e + \{uv_e, v_e w\}$.



„Stejnost“ grafů; Isomorfismus

Pozorný čtenář si možná již při čtení předchozího textu položil otázku: Co když vezmeme jeden graf (třeba kružnici délky 5) a nakreslíme jej jednou tak, podruhé zase jinak – je to stále tentýž graf nebo ne? Přísně formálně řečeno, každé nakreslení jistého grafu, třeba té kružnice C_5, je jiným grafem, ale přitom bychom rádi řekli, že různá nakreslení téhož grafu jsou *stále stejná*. Už jen proto, že grafy mají modelovat vztahy mezi dvojicemi objektů, ale tyto vztahy přece vůbec nezávisí na tom, jak si grafy nakreslíme. Pro tuto „*stejnost*“ grafů se vžil pojem *isomorfní grafy* a je ilustrována následujícím obrázkem.



Pro formálně správné pochopení a použití teorie grafů je potřeba nejprve dobře chápat tento pojem isomorfismu grafů:

Definice 1.6. ***Isomorfismem*** grafů G a H je každé bijektivní (*vzájemně jednoznačné*) zobrazení $f : V(G) \to V(H)$, pro které platí, že kterákoliv dvojice vrcholů $u, v \in V(G)$ je spojená hranou v G právě tehdy, když je dvojice $f(u), f(v)$ spojená hranou v H. Grafy G a H jsou *isomorfní* pokud mezi nimi existuje isomorfismus. Píšeme $G \simeq H$.

`Fakt`: Isomorfní grafy mají stejný počet vrcholů (i hran).

`Fakt`: Pokud je bijekce f isomorfismem, musí zobrazovat na sebe vrcholy stejných stupňů, tj. $d_G(v) = d_H(f(v))$. Naopak to však nestačí!

Příklad 1.7. *Jsou následující dva grafy isomorfní?*

Pokud mezi nakreslenými dvěma grafy hledáme isomorfismus, nejprve se podíváme, zda mají stejný počet vrcholů a hran. Mají. Pak se podíváme na stupně vrcholů a zjistíme, že oba mají stejnou posloupnost stupňů $2, 2, 2, 2, 3, 3$. Takže ani takto jsme mezi nimi nerozlišili a mohou (nemusejí!) být isomorfní. Dále tedy nezbývá, než zkoušet všechny přípustné možnosti zobrazení isomorfismu z levého grafu do pravého.

Na levém grafu si pro ulehčení všimněme, že oba vrcholy stupně tři jsou si symetrické, proto si bez újmy na obecnosti můžeme vybrat, že nejlevější vrchol prvního grafu, označme jej 1, se zobrazí na nejlevější vrchol $1'$ v druhém grafu (také stupně tři). Očíslujme zbylé vrcholy prvního grafu $2, \ldots, 6$ v kladném smyslu, jak je ukázáno na následujícím obrázku. Druhý vrchol stupně tři, označený 4, se musí zobrazit na analogický vrchol druhého grafu (pravý spodní). Pak je již jasně vidět, že další sousedé $2, 6$ vrcholu 1 se zobrazí na analogické sousedy $2', 6'$ vrcholu $1'$ v druhém grafu, a stejně je to i se zbylými vrcholy $3, 5$. Výsledný isomorfismus vypadá v odpovídajícím značení vrcholů takto:

□

Grafy jako třídy isomorfismu

S Definicí 1.6 a následujícím poznatkem pak již není problém matematicky správně abstrahovat od představy grafu jako jednoho konkrétního obrázku k představě zahrnující pod jedno jméno či označení všechny „stejně vypadající“ grafy.

Věta 1.8. *Relace „být isomorfní“ $\simeq$ na třídě všech grafů je ekvivalencí.*

`Důkaz`. Relace $\simeq$ je reflexivní, protože graf je isomorfní sám sobě identickým zobrazením. Relace je také symetrická, neboť bijektivní zobrazení lze jednoznačně obrátit. Tranzitivnost $\simeq$ se snadno dokáže skládáním zobrazení–isomorfismů. □

Důsledkem je, že všechny možné grafy se rozpadnou na *třídy isomorfismu*. V praxi pak, pokud mluvíme o *grafu*, myslíme tím obvykle jeho celou třídu isomorfismu, tj. nezáleží nám na konkrétní prezentaci grafu.

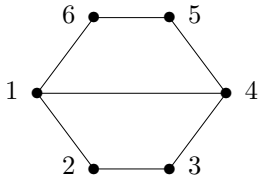
Další grafové pojmy

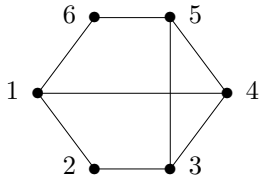
Následující terminologie nám pomáhá se srozumitelně vyjadřovat o vlastnostech grafů.

`Značení`: Mějme libovolný graf G.

- Podgrafu $H \subseteq G$, který je isomorfní nějaké kružnici, říkáme *kružnice v G*.
- Speciálně říkáme *trojúhelník* každému podgrafu isomorfnímu kružnici délky 3.
- Podgrafu $H \subseteq G$, který je isomorfní nějaké cestě, říkáme *cesta v G*.
- Podgrafu $H \subseteq G$, který je isomorfní nějakému úplnému grafu, říkáme *klika v G*. (Někdy se však za kliku považuje pouze takový úplný podgraf, který je maximální vzhledem k inkluzi.)
- Indukovanému podgrafu $H \subseteq G$, který je isomorfní nějaké kružnici, říkáme *indukovaná kružnice v G*. Podobně se definuje *indukovaná cesta*.

`Komentář:` Uvažujme následující ukázky grafů:





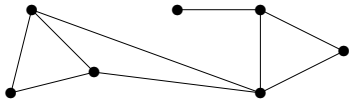
První z ukázaných grafů například neobsahuje žádný trojúhelník, ale obsahuje kružnici délky 4, dokonce indukovanou. Druhý graf trojúhelník obsahuje a kružnici délky 4 taktéž. První graf obsahuje cestu délky 4 na vrcholech $1, 2, 3, 4, 5$, ale ta není indukovaná. Indukovaná cesta délky 4 v něm je třeba $2, 3, 4, 5, 6$. Druhý graf tyto cesty také obsahuje, ale naopak žádná z nich není indukovaná.

První graf má největší kliku velikosti 2 – jedinou hranu, kdežto druhý graf má větší kliku na vrcholech $3, 4, 5$. Největší nezávislá množina u obou grafů má 3 vrcholy, třeba $2, 4, 6$.

Mimo to je zvykem hovořit o následujících specifických typech grafů, kterým se budeme blíže věnovat v nadcházejících lekcích.

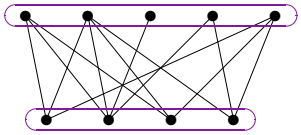
`Značení`: Mějme libovolný graf G.

- G je *souvislý graf* (ne „spojitý“!), pokud každé dva vrcholy $u, v \in V(G)$ jsou spojeny cestou v G mezi konci u a v.



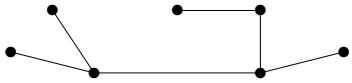
– detaily nalezneme v Lekci 2...

- G je *bipartitní graf*, pokud G je podgrafem úplného bipartitního grafu $K_{m,n}$ pro vhodná m, n.



– blíže rozebereme v Lekci **??**...

- G je *strom*, pokud je G souvislý a nemá žádnou kružnici jako podgraf.



O těžkosti problému isomorfismu

Mnohé příklady uváděné mimo jiné v této lekci nám ukazují (některé cesty), jak najít nebo vyloučit isomorfismus dvou grafů. Další, tentokrát negativní, příklad přidáváme.

Příklad 1.9. *Jsou následující dva grafy isomorfní?*

Postupovat budeme jako v Příkladě 1.7, nejprve ověříme, že oba grafy mají stejně mnoho vrcholů i stejnou posloupnost stupňů $2, 2, 2, 2, 3, 3$. Pokud se však budeme snažit najít mezi nimi isomorfismus, něco stále nebude vycházet, že? Co nám tedy v nalezení isomorfismu brání? Podívejme se, že v druhém grafu oba vrcholy stupně tři mají svého společného souseda, tvoří s ním trojúhelník. V prvním grafu tomu tak není, první graf dokonce nemá žádný trojúhelník. Proto zadané dva grafy nejsou isomorfní. □

Ne vždy však podobně jednoduchý přístup musí fungovat. Čtenář se může ptát, kde tedy najde nějaký univerzální postup pro nalezení isomorfismu? Bohužel vás musíme zklamat, žádný rozumný univerzální postup není znám a zatím platí, že jediná vždy fungující cesta pro nalezení či nenalezení isomorfismu mezi dvěma grafy je ve stylu *vyzkoušejte všechny možnosti* bijekcí mezi vrcholy těchto grafů. (Těch je, jak známo, až $n!$) Na druhou stranu existují dobře fungující heuristické přístupy řešení a v hierarchii výpočetní složitosti problém isomorfismu leží docela nízko, jen „kousek nad“ třídou polynomiálních problémů, takže je jistá naděje na nalezení univerzálního efektivního algoritmu v budoucnu.

Otázky a úlohy k řešení

(1.2.1) *Je nějaká kružnice podgrafem cesty?*

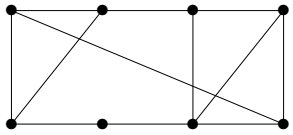
(1.2.2) *Musí mít isomorfní grafy stejný počet hran?*

(1.2.3) *Je nějaká cesta indukovaným podgrafem kružnice?*

(1.2.4) *Jaký má smysl mluvit o indukované klice? A o indukované hvězdě?*

(1.2.5) *Který z těchto dvou grafů je souvislý?*

(1.2.6) *Jaká je délka nejkratší kružnice obsažené v tomto grafu?*



(1.2.7) *Jaká je délka nejdelší cesty obsažené v grafu z Úlohy 1.2.6?*

*(1.2.8)** *Jaká je délka nejdelší indukované cesty obsažené v grafu z Úlohy 1.2.6?*

(1.2.9) *Dokážete najít ke grafu z Úlohy 1.2.6 neisomorfní graf, který má stejnou posloupnost stupňů? Proč tyto dva grafy nejsou isomorfní?*

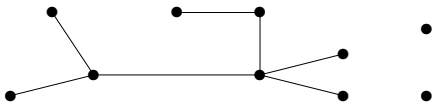
*(1.2.10)** *Kolik existuje jednoduchých neisomorfních grafů se všemi vrcholy stupně* 2 *na*
a) 5 *vrcholech;* *b)* 6 *vrcholech;* *c)* 9 *vrcholech?*

1.3 Stromy – grafy bez kružnic

V návaznosti na předchozí oddíl začněme definicí stromů a pokračujeme přehledem jejich základních vlastností.

Definice 1.10. ***Strom*** je jednoduchý souvislý graf T bez kružnic.
Jednoduchý graf bez kružnic (tj. bez požadavku souvislosti) nazveme *lesem*.

`Komentář:` Každý les je sjednocením jednoho či několika stromů. Jeden vrchol (bez hran) je také strom.



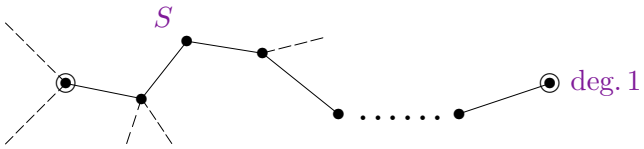
Jelikož smyčky a násobné hrany v multigrafech jsou považovány za kružnice délek 1 a 2, v lesech a stromech se nemohou nacházet. Stromy tudíž musí být jednoduchými grafy.

Základní vlastnosti stromů

Následující vlastnosti stromů, která uvádíme i s jednoduchými důkazy, plně popisují stromy jako specifický typ grafů a mohou tudíž být případně použity místo uvedené Definice 1.10. Jedná se sice na úvod o dosti formální látku, avšak alespoň zběžné porozumění uvedeným vlastnostem je potřebné pro další práci se stromy a s jinými strukturami založenými na stromech.

Lema 1.11. *Strom s více než jedním vrcholem obsahuje vrchol stupně* 1.

Důkaz: Souvislý graf s více než jedním vrcholem nemůže mít vrchol stupně 0. Proto vezmeme strom T a v něm libovolný vrchol v. Sestrojíme nyní co nejdelší sled S v T začínající ve v:



S začne libovolnou hranou vycházející z v. V každém dalším vrcholu u, do kterého se dostaneme a má stupeň větší než 1, lze pak pokračovat sled S další novou hranou. Uvědomme si, že pokud by se ve sledu S poprvé zopakoval některý vrchol, získali bychom kružnici, což ve stromě nelze. Proto sled S musí jednou skončit v nějakém vrcholu stupně 1 v T. □

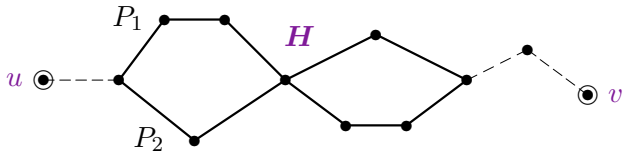
Komentář: Zamyslete se, proč v každém stromě s více než jedním vrcholem jsou alespoň dva vrcholy stupně 1 (odpověď je skrytá už v předchozím důkaze). Zároveň si odpovězte, jestli lze tvrdit, že každý strom s více než jedním vrcholem obsahuje tři vrcholy stupně 1.

Věta 1.12. *Strom na n vrcholech má přesně $n-1$ hran pro $n \geq 1$.*

Důkaz: Toto tvrzení dokážeme indukcí podle n. Strom s jedním vrcholem má $n-1=0$ hran. Nechť T je strom na $n>1$ vrcholech. Podle Lematu 1.11 má T vrchol v stupně 1. Označme $T' = T \setminus v$ graf vzniklý z T odebráním vrcholu v. Pak T' je také souvislý bez kružnic, tudíž strom na $n-1$ vrcholech. Dle indukčního předpokladu T' má $n-1-1$ hran, a proto T má $n-1-1+1 = n-1$ hran. □

Věta 1.13. *Mezi každými dvěma vrcholy stromu vede právě jediná cesta.*

Důkaz: Jelikož strom T je souvislý dle definice, mezi libovolnými dvěma vrcholy u, v vede nějaká cesta.



Pokud by existovaly dvě různé cesty P_1, P_2 mezi u, v, tak bychom vzali jejich symetrický rozdíl, podgraf $H = P_1 \Delta P_2$ s neprázdnou množinou hran, kde H zřejmě má všechny stupně sudé. Na druhou stranu se však podgraf stromu musí opět skládat z komponent stromů (ne všechny jsou izolovanými vrcholy), a tudíž obsahovat vrchol stupně 1 podle Lematu 1.11, spor. Proto cesta mezi u a v existuje jen jedna. □

Důsledek 1.14. *Přidáním jedné nové hrany do stromu vznikne právě jedna kružnice.*

Důkaz: Nechť mezi vrcholy u, v ve stromu T není hrana. Přidáním hrany $e = uv$ vznikne právě jedna kružnice z e a jediné cesty mezi u, v v T podle Věty 1.13. □

Věta 1.15. *Strom je minimální souvislý graf (na daných vrcholech).*

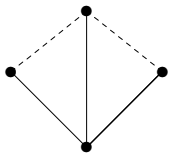
Důkaz: Strom je souvislý podle definice. Pokud by ale vypuštěním hrany $e = uv$ ze stromu T vznikl opět souvislý graf, pak by mezi u, v v T existovaly dvě cesty (dohromady kružnice) – hrana e a jiná cesta v $T \setminus e$. To je ve sporu s Větou 1.13. Naopak, pokud by souvislý graf měl kružnici, zůstal by souvislý i po vypuštění některé hrany té kružnice.

Proto každý minimální souvislý graf (na daných vrcholech) je stromem. Tudíž strom je právě minimálním souvislým grafem na daných vrcholech. □

Závěrem si pro správné pochopení základních vlastností stromů vyřešíme následující (ne zcela jednoduchý) příklad.

Příklad 1.16. *Kolik nejvýše kružnic vznikne v grafu, který vytvoříme ze stromu přidáním dvou hran?*

Přidáním jedné hrany do stromu T vznikne jedna kružnice dle Důsledku 1.14. Druhá hrana vytvoří nejméně ještě jednu kružnici ze stejných důvodů, ale může vytvořit i dvě další kružnice, jako třeba v následujícím grafu, kde strom T je vyznačen plnými čarami a dvě přidané hrany čárkovaně.



Každá z přidaných dvou hran vytvoří vlastní trojúhelník a navíc ještě vznikne kružnice délky 4 procházející oběma z přidaných hran.

Na druhou stranu chceme ukázat, že více než 3 kružnice po přidání dvou hran e, f do stromu T vzniknout nemohou: Podle Důsledku 1.14 vznikne jen jedna kružnice procházející hranou e a neobsahující f, stejně tak jedna kružnice procházející f a neobsahující e. Nakonec stačí nahlédnout, že je nejvýše jedna možná kružnice procházející oběma hranami e, f: Pokud by takové byly dvě různé C_1, C_2, podívali bychom se na jejich symetrický rozdíl, podgraf $H = C_1 \Delta C_2$, který má všechny stupně sudé, neprázdnou množinu hran a je navíc pografem stromu T. Takže stejně jako ve Větě 1.13 dostáváme spor s faktem, že podgrafy stromů s hranami musí obsahovat vrchol stupně 1. □

Otázky a úlohy k řešení

(1.3.1) *Je prázdný graf (bez vrcholů a hran) stromem?*

(1.3.2) *Které ze základních typů grafů z Oddílu 1.1 jsou stromy? (A nezapomněli jste na jeden podtyp?)*

(1.3.3) *Lze o některém ze základních typů grafů z Oddílu 1.1 říci, že to určitě nebude strom?*

(1.3.4) *Kolik je neisomorfních lesů na třech vrcholech? Nakreslete si je.*

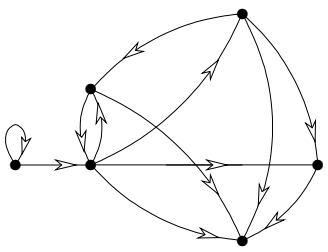
(1.3.5) *Kolik je neisomorfních stromů na čtyřech vrcholech? Nakreslete si je.*

*(1.3.6)** *Jak byste definovali strom s použitím poznatku Věty 1.12?*

*(1.3.7)** *Jak byste definovali strom s použitím poznatku Věty 1.13?*

1.4 Orientované grafy a multigrafy

V některých případech, jako například u toků v sítích (Lekce 4), potřebujeme u každé hrany grafu vyjádřit její směr. Tento požadavek přirozeně vede na následující definici *orientovaného grafu*, ve kterém hrany jsou uspořádané dvojice vrcholů. (V obrázcích kreslíme orientované hrany se šipkami.)



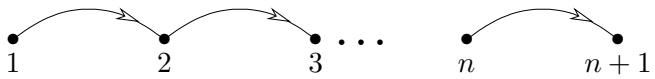
Definice 1.17. ***Orientovaný graf*** je uspořádaná dvojice $D = (V, E)$, kde $E \subseteq V \times V$. Pojmy podgrafu a isomorfismu se přirozeně přenášejí na orientované grafy.

`Značení`: Hrana (u, v) (zvaná také *šipka*) v orientovaném grafu D *začíná* ve vrcholu u a *končí* ve (míří do) vrcholu v. Opačná hrana (v, u) je různá od (u, v)!

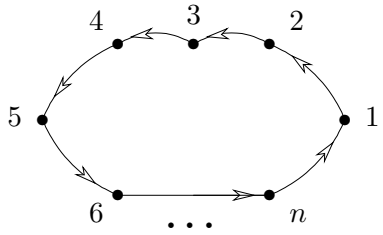
Speciálně hrana tvaru (u, u) se nazývá *orientovaná smyčka*.

`Fakt`: Orientované grafy odpovídají relacím, které nemusí být symetrické. Mezi stejnou dvojicí vrcholů mohou tudíž existovat dvě hrany v obou směrech, nebo jen kterákoliv jedna z nich nebo žádná.

Například *orientovaná cesta* délky $n \geq 0$ je následujícím grafem na $n + 1$ vrcholech



a *orientovaná kružnice* (také *cyklus*) délky $n \geq 1$ vypadá takto:



`Definice`: Počet hran začínajících ve vrcholu u orientovaného grafu D nazveme *výstupním stupněm* $d_D^+(u)$ a počet hran končících v u nazveme *vstupním stupněm* $d_D^-(u)$.

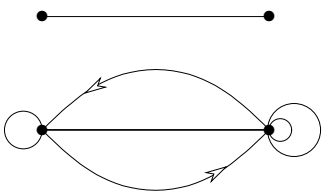
Různé modely grafu

Při nazírání na graf jako matematický objekt existují dva zásadně odlišné formální pohledy—tzv. *sousednostní* a *incidenční* model. Náš výklad dosud mlčky předpokládal první přístup a bude tomu tak i ve zbytku textu. Přesto by pozorný čtenář měl mít povědomí i o druhém incidenčním modelu grafu, který staví hrany jako samostatné elementy a místo relace sousednosti mezi dvojicemi vrcholů zavádí relaci *incidence* mezi dvojicí vrchol–hrana.

Předností sousednostního modelu je jeho jednoduchost a přímočarost, kdežto výhodou incidenčního modelu je možnost mluvit o tzv. *multigrafech*, ve kterých jsou povoleny i *násobné hrany*, mix neorientovaných a orientovaných hran i tzv. *smyčky* s oběma konci v jednom vrcholu.

`Definice`: *Multigraf* je uspořádaná trojice $M = (V, E, \varepsilon)$, kde $V \cap E = \emptyset$ a $\varepsilon : E \to \binom{V}{2} \cup V \cup (V \times V)$ je *incidenční zobrazení* hran.

> `Komentář`: Značení $\binom{V}{2}$ v definici reprezentuje neorientované hrany, V neorientované smyčky a $V \times V$ orientované hrany a smyčky. Smyčky a paralelní hrany (oproti běžné hraně) lehce ilustruje následující obrázek:



Multigrafy a incidenční model grafu zmiňujeme jen okrajově pro úplnost, nebudeme se jimi dále nijak zabývat. Faktem však je, že v případě jednoduchých grafů není mezi sousednostním a incidenčním modelem rozdíl a není třeba tuto formalitu vůbec brát do úvahy...

Otázky a úlohy k řešení

(1.4.1) *Nalezněte ve výše zakreslených obrázcích orientovaných grafů vrcholy, v nichž žádná hrana/šipka nezačíná, a ty, v nichž žádná hrana nekončí.*

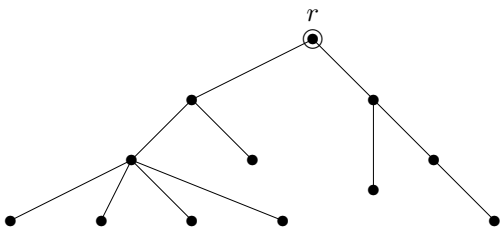
(1.4.2) *Proč u jednoduchých orientovaných grafů mluvíme i o cyklech délky 1 a 2, kdežto u jednoduchých neorientovaných grafů jen o kružnicích délky* ≥ 3

1.5 Dodatek: Kořenové stromy a isomorfismus

Při mnoha aplikacích stromových struktur se ke stromu jako grafu samotnému ještě váží dodatečné informace, jako třeba vyznačený jeden vrchol, tzv. kořen stromu, ze kterého celý strom „vyrůstá“. Typickým příkladem jsou různé (acyklické) datové struktury, ve kterých je vyznačený vrchol – kořen, referován jako „začátek“ uložených dat. Jinak třeba evoluční stromy druhů v biologii mají za kořen jejich společného (dávného) předchůdce, apod. Kořenové stromy mají také tradiční motivaci v rodokmenech a z toho vychází jejich běžná terminologie.

Definice 1.18. ***Kořenovým stromem*** je strom T spolu s vyznačeným *kořenem* $r \in V(T)$, zkráceně zapsaný dvojicí (T, r).

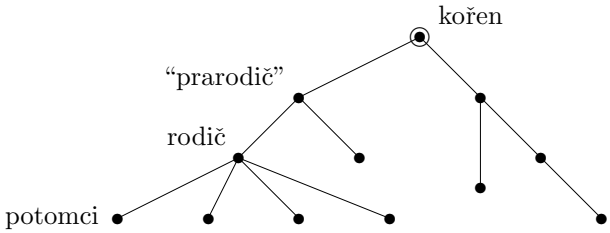
`Komentář:` Příklad kořenového stromu je na následujícím obrázku:



Zajímavostí je, že v informatice stromy většinou rostou od kořene směrem dolů. (Však také nejsme v biologii...)

`Definice`: Mějme kořenový strom T, r a v něm vrchol v. Označme u souseda v na cestě směrem ke kořeni r. Pak je u nazýván *rodičem* v a v je nazýván *potomkem* u.

`Komentář:` Kořen nemá žádného rodiče. V přirozeně přeneseném významu se u kořenových stromů používají pojmy prarodič, předchůdce, následovník, sourozenci, atd. Zběžná ilustrace použití těchto pojmů je na následujícím schématu stromu.



Často se také setkáte v kořenových stromech s označováním „otec–syn“ místo rodič–potomek. My jsme takové označení nepoužili proto, že by (hlavně v zemích na západ od nás a navzdory zdravému rozumu) mohlo být považováno za genderově nepatřičné.

`Definice`: Vrchol stupně 1 v libovolném stromu nazýváme *listem*.

`Komentář:` Pozor, i kořen stromu může být listem, pokud má stupeň 1, ale obvykle se to tak neříká. List kořenového stromu, který není kořenem, nemá potomky.

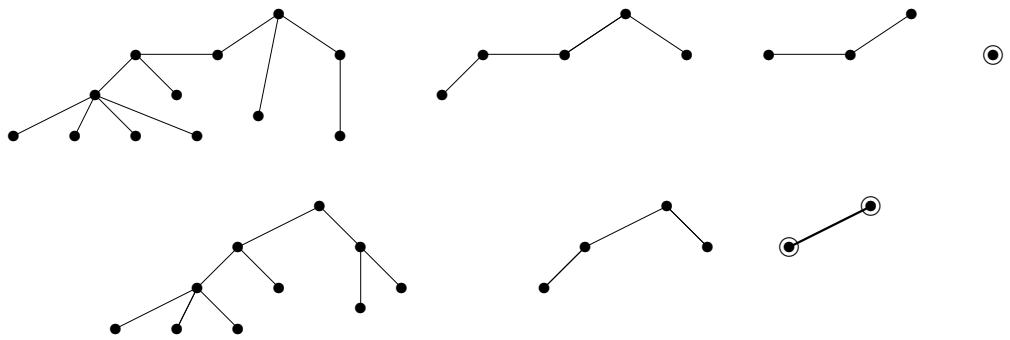
Centrum stromu

V některých situacích je třeba k danému stromu zvolit kořen tak, aby jeho volba byla jednoznačná, tj. aby bylo zaručeno, že pro dva isomorfní stromy nezávisle zvolíme tentýž kořen. K tomu nám dopomůže následující názorná definice.

`Definice`: *Centrem stromu* T rozumíme buď vrchol nebo hranu nalezenou v T následujícím postupem:

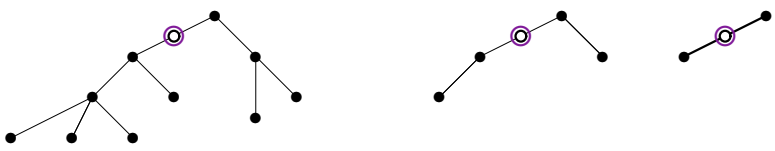
- Pokud má strom T jeden vrchol, je to jeho centrum. Pokud má strom T dva vrcholy, je jeho centrem hrana spojující tyto dva vrcholy.
- Jinak vytvoříme menší strom $T' \subset T$ vypuštěním všech listů T najednou. Je zřejmé, že T' je neprázdný, a vracíme se na předchozí bod. Rekurzivně získané centrum T' je zároveň centrem T.

Příklad 1.19. *Ilustrací definice centra jsou následující dvě ukázky jeho nalezení (čteme obrázky postupu zleva doprava):*



V prvním stromě získáme kořen jako jediný vrchol po třech rekurzivních krocích odebrání listů. V druhém stromě je kořenem hrana, kterou získáme po dvou rekurzivních krocích. □

`Komentář:` Pokud chceme danému (abstraktnímu) stromu přiřadit jednoznačně kořen, je nejlepší jej přiřadit centru stromu. Speciálně, pokud je centrem hrana, bude kořenem nový vrchol „rozdělující“ tuto hranu na dvě. Viz předchozí příklad:



Později si může čtenář také povšimnout, že předchozí definice centra stromu přesně odpovídá „vzdálenostnímu“ centru grafu–stromu danému Definicí 3.3.

Uspořádání kořenového stromu

Další doplňkovou informací často vázanou ke kořenovým stromům je uspořádání potomků každého vrcholu, jako třeba seřazení potomků v rodokmenech podle jejich data narození. To formalizujeme následující definicí.

`Definice`: Kořenový strom T, r je *uspořádaný*, pokud je pro každý jeho vrchol jednoznačně dáno pořadí jeho potomků (zleva doprava).
Uspořádaný kořenový strom se také nazývá *pěstovaný strom*.

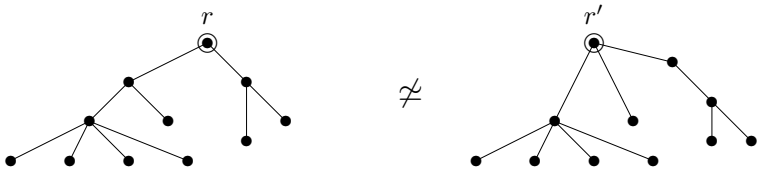
> `Komentář:` Uspořádaný kořenový strom si jinak také můžeme představit jako strom s vyznačeným kořenem a pevně zvoleným nakreslením v rovině bez křížení hran. Nakreslení hran potomků vzhledem k hraně rodiče pak udává (ve zvolené orientaci) pořadí potomků. Tento pohled vede k názvu pěstovaný strom.
>
> Uspořádání potomků vrcholu ve stromu je přirozeně požadováno v mnoha praktických situacích. Například ve stromových datových strukturách jsou často potomci explicitně seřazeni podle daného klíče, jako třeba ve vyhledávacích binárních stromech. I v případech, kdy uspořádání potomků ve stromě není dáno, je možné jej jednoznačně definovat, jak uvidíme v následující části.

Isomorfismus stromů

Jelikož stromy jsou speciálním případem grafů, je isomorfismus stromů totéž co isomorfismus grafů. Avšak na rozdíl od obecných grafů, kdy je určení isomorfismu algoritmicky (nakonec i ručně) těžký problém, pro isomorfismus stromů existuje efektivní postup, který si ukážeme dále. Nejprve si uvedeme restriktivnější (tj. vyžadující více shody) verze definice isomorfismu pro kořenové a uspořádané stromy.

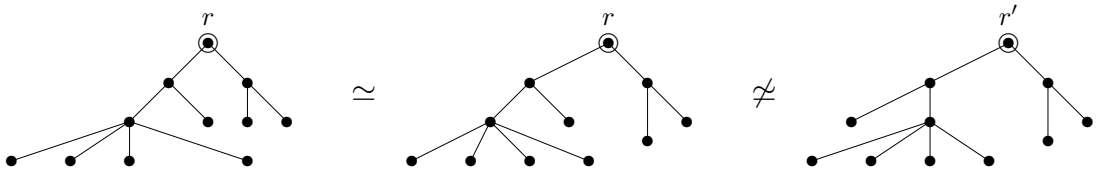
`Definice`: (Dva stromy jsou isomorfní pokud jsou isomorfní jako grafy.) Dva kořenové stromy T, r a T', r' jsou *isomorfní* pokud existuje isomorfismus mezi stromy T a T', který kořen r zobrazuje na kořen r'.

> `Komentář:` Například následující dva (isomorfní) stromy *nejsou isomorfní* coby kořenové stromy.



`Definice`: Dva uspořádané kořenové (pěstované) stromy jsou isomorfní pokud je mezi nimi isomorfismus kořenových stromů, který navíc zachovává pořadí potomků všech vrcholů.

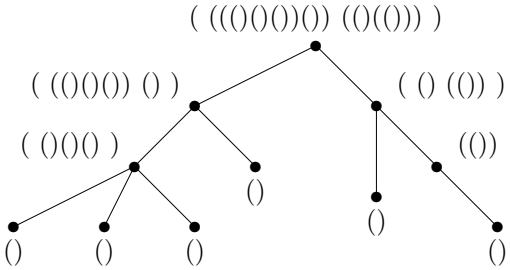
> `Komentář:` Například z následujících tří isomorfních kořenových stromů jsou první isomorfní i coby uspořádané kořenové stromy, avšak třetí s nimi takto *isomorfní není*, neboť pořadí potomků levého syna kořene se liší.



Kódování uspořádaných kořenových stromů

Uspořádanému kořenovému stromu lze snadným postupem přiřadit řetězec vnořených závorek, který jej plně popisuje. (Možná jste se již s touto jednoduchou korespondencí závorek a kořenových stromů setkali při sémantické analýze zanořených matematických výrazů.)

`Definice`:



Kód uspořádaného kořenového stromu se spočítá rekurzivně z kódů všech podstromů kořene, seřazených v daném pořadí a uzavřených do páru závorek.

`Poznámka`: Místo znaků '(' a ')' lze použít i jiné symboly, třeba '0' a '1'.

Klíčovým faktem o kódech pěstovaných stromů je toto tvrzení:

Lema 1.20. *Dva uspořádané kořenové (pěstované) stromy jsou isomorfní právě když jejich kódy získané podle předchozího popisu jsou shodné řetězce.*

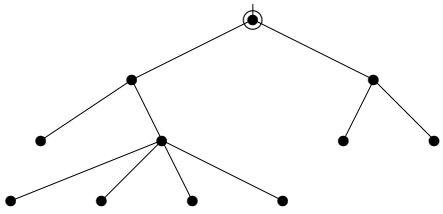
`Důkaz`: Postupujeme indukcí podle hloubky stromu (nejdelší vzdálenosti z kořene). Báze je zřejmá, strom hloubky 0 je jedinečný s kódem '()'. Dále podle definice jsou dva uspořádané kořenové stromy isomorfní, právě když v daném pořadí jsou po dvojicích isomorfní podstromy jejich kořenů, neboli podle indukčního předpokladu právě tehdy, když seřazené kódy podstromů (nižší hloubky) těchto kořenů vytvoří shodné řetězce. □

Příklad 1.21. *Nakreslete jako pěstovaný strom ten odpovídající závorkovému kódu*

$$(\,(()(()()()()))\,(()())\,)\,.$$

Je-li dán kód s uspořádaného kořenového stromu, snadno z definice nahlédneme, že příslušný strom nakreslíme následujícím postupem:

- Při přečtení znaku '(' na začátku spustíme pero na papír, do kořene.
- Při každém dalším přečtení znaku '(' nakreslíme hranu do následujícího potomka současného vrcholu.
- Při každém přečtení znaku ')' se perem vrátíme do rodiče současného vrcholu, případně zvedneme pero, pokud už jsme v kořeni.



□

Efektivní algoritmus isomorfismu pro stromy

Při určování isomorfismu obecných stromů použijeme Lema 1.20 a jejich jednoznačnou reprezentaci uspořádanými kořenovými stromy, ve které kořen volíme v centru a potomky seřadíme podle jejich kódů vzestupně abecedně. Jinak se dá také říci, že kód přiřazený stromu T je lexikograficky nejmenší ze všech kódů uspořádaných stromů vzniklých z T s kořenem v jeho centru. Takový kód je zřejmě určující vlastností stromu T jako takového, a proto správnost následujícího algoritmu můžeme snadno nahlédnout níže.

Algoritmus 1.22. ***Určení isomorfismu dvou stromů.***
Pro dané dva obecné stromy T a U implementujeme algoritmus zjišťující isomorfismus $T \simeq^? U$ následovně v symbolickém zápise:

```
vstup < stromy T a U;
if (|V(T)| ≠ |V(U)|) return 'Nejsou isomorfní.';
(T,r) ← korenove_centrum(T);    (U,s) ← korenove_centrum(U);
k ← minimalni_kod(T,r);          m ← minimalni_kod(U,s);
if (k=m coby řetězce)  výstup 'Jsou isomorfní.';
else  výstup 'Nejsou isomorfní.';

Funkce minimalni_kod(strom X, vrchol r) {
    if (|V(X)| = 1) return "()";
    d ← počet komponent grafu X\r, tj. podstromů kořene r;
    foreach (i ← 1,...,d) {
        Y[i] ← i-tá souvislá komponenta grafu X\r;
        s[i] ← i-tý soused r, tj. kořen podstromu Y[i];
        k[i] ← minimalni_kod(Y[i],s[i]);
    }
    sort k[1] ≤ k[2] ≤ ... ≤ k[d] lexikograficky (abecedně);
    return "("+k[1]+...+k[d]+")"(zřetězení);
}

Funkce korenove_centrum(strom X) {
    r = centrum(X);
    if (r je jeden vrchol ) return (X,r);
    else nový s, nahraď hranu r=uv v X hranami su, sv;
    return (X,s);
}
```

Důkaz správnosti Algoritmu 1.22 je podán v následujícím tvrzení.

Věta 1.23. *Mějme dva stromy T, U o stejném počtu vrcholů a nechť (T', r) a (U', s) jsou po řadě jejich kořenové stromy získané v první části Algoritmu 1.22 (kde r, s jsou centra T, U). Pak platí:*

a) T a U jsou isomorfní, právě když (T', r) je isomorfní (U', s).

b) (T', r) je isomorfní (U', s), právě když

$$\texttt{minimalni\_kod}(\texttt{T}', \texttt{r}) = \texttt{minimalni\_kod}(\texttt{U}', \texttt{s}).$$

Důkaz: Tvrzení (a) ihned plyne z jednoznačnosti definice centra stromu.

Za druhé (b) dokazujeme indukcí podle hloubky našich kořenových stromů T', r a U', s. (Zřejmě pokud mají různé hloubky, isomorfní nejsou.) Dva kořenové stromy hloubky 0 jsou vždy isomorfní a mají shodný kód „()“. Dále vezmeme T', r a U', s hloubky $\ell > 0$. Pokud jejich kódy vyjdou shodné, jsou isomorfní. Naopak pro isomorfní T', r a U', s existuje bijekce mezi vzájemně isomorfními podstromy jejich kořenů, tudíž podle indukčního předpokladu kódy těchto podstromů jsou po dvojicích shodné. Jelikož se v obou případech setřídí kódy podstromů stejně, vyjde $\texttt{minimalni\_kod}(\texttt{T}', \texttt{r}) = \texttt{minimalni\_kod}(\texttt{U}', \texttt{s})$. □

Otázky a úlohy k řešení

(1.5.1) *Který jediný strom nemá žádné centrum?*

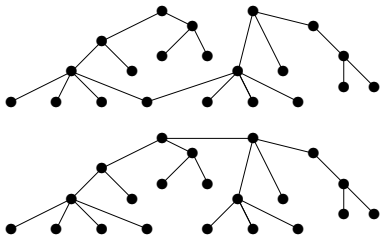
(1.5.2) *U jakého typu stromu trvá nalezení centra podle definice největší počet kroků (vzhledem k velikosti stromu)? A u jakého nejmenší počet kroků?*

*(1.5.3) *Jaká je souvislost nejdelší cesty obsažené ve stromu s jeho centrem?*

(1.5.4) *Binární vyhledávací strom je uspořádaný kořenový strom, který se pro daná data obvykle tvoří následovně: První prvek dat se uloží do kořene. Každý další příchozí prvek, pokud má menší klíč než kořen, uloží se rekurzivně do levého podstromu, pokud větší, tak do pravého podstromu. Vytvořte binární vyhledávací strom pro danou posloupnost dat s klíči*

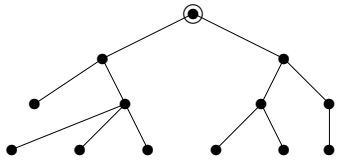
$$11, 15, 9, 5, 13, 10, 20, 21, 1, 6\,.$$

(1.5.5) *Určete centra následujících dvou stromů:*



(1.5.6) *Mějme libovolný strom s 33 vrcholy. Kolik jeho listů postupně odebereme, než určíme centrum? (Je to jednoznačné?)*

(1.5.7) *Odvoďte správný závorkový kód pro tento pěstovaný strom:*



(1.5.8) *Nakreslete pěstovaný strom odpovídající závorkovému kódu*

$$(\,(((())())())\,(((())(()))\,)\,.$$

Rozšiřující studium

Čtivý matematický úvod do teorie grafů je zahrnut v [5, Kapitola 4] a volně na něj navazují i některé další části našeho učebního textu. Vřele doporučujeme [5] jako doplňkový studijní zdroj všem, kteří s grafy začínají a chtějí lépe pochopit grafy z jejich matematické stránky. Pokročilé části našeho studijního textu jsou pak převážně inspirovány klasickou a obsáhlou učebnicí teorie grafů [1].

Co se týče detailů implementace grafů v programátorské praxi, najde čtenář asi nejlepší zdroje na internetu, třeba `http://en.wikipedia.org/wiki/Graph_(data_structure)` *nebo* `http://www.mozart-oz.org/mogul/doc/smiele/graph/` *a mnoho jiných stránek... Hezké, ale poněkud pokročilé pojednání o grafových algoritmech lze volně nalézt také v [4]. Speciálně co se týče praktického zjišťování isomorfismu grafů a dalších podobných grafových počítačových úloh, nejlépe je se podívat na klasiku* `http://cs.anu.edu.au/~bdm/nauty/`*. V našem textu se sice na okraj matematické teorie dotkneme mnoha zajímavých grafových algoritmů, ale zcela se abstrahujeme od otázek praktické implementace.*

Studenti MU si také mohou vyzkoušet množství „online“ základních příkladů o grafech (na isomorfismus, stupně vrcholů, běžné vlastnosti grafů, atd.) ve formě odpovědníků v IS pod učebními materiály předmětů IB000 a MA010. Někomu sice může připadat zbytečné trávit čas mechanickým řešením takových základních (a nudných?) příkladů, ale domníváme se, že pro čtenáře toto bude mít nezastupitelný přínos zejména s ohledem na získání potřebného nadhledu nad grafy a „citu“ pro řešení složitějších grafových úloh v budoucnu.

1.6 Cvičení: Základní grafové otázky

Než přikročíme k procvičení jednoduchých úloh vztahujících se k fundamentálním pojmům úvodní lekce, nastíníme základní motivaci pro zavedení a použití grafů při popisu a řešení problémů z běžného života.

Příklad 1.24. *Ukázky některých problémů popsatelných grafy. Podotýkáme, že tyto ukázky jsou často velmi zjednodušené (pro jejich lepší přístupnost širokému okruhu čtenářů), ale to neubírá jejich motivačnímu potenciálu.*

- Vyjádření mezilidských vztahů – „mají se rádi“, „kamarádí se“, „nesnesou jeden druhého“, apod:

  Zde jednotlivé osoby tvoří vrcholy grafu a vztahy jsou hranami (často neorientované, ale i orientace je přípustná). Všimněme si coby zajímavosti, jak tento model přirozeně preferuje „párový“ pohled na vztahy – hrany přece spojují jen dvojice vrcholů. Třebaže například vztah „kamarádí se“ může být obecně platný pro větší skupinky lidí než dvojice, stejně se obvykle vyjadřuje klikou v grafu (každí dva v našem družstvu jsou dobří kamarádi...). Tato obecně pojímaná tendence vyjadřovat i složitější vztahy těmi párovými je vodou na mlýn použití teorie grafů jako téměř univerzálního vyjadřovacího prostředku v podobných případech.

  Na druhou stranu i teorie grafů disponuje pojmem tzv. *hypergrafu* umožňujícího použití hran libovolné arity (počtu koncových vrcholů), ale rozsah výskytu hypergrafů v teorii i aplikacích je oproti grafům vskutku zanedbatelný.

- Vyjádření závislostí mezi objekty nebo procesy:

  Představme si situace, ve kterých jednotlivé entity (modelované jako vrcholy) závisí na výstupech jiných entit a naopak poskytují výstupy další entitám. Typickým příkladem mohou být závislosti jednotlivých kroků výrobního nebo rozhodovacího procesu. Ty pak vedou k definici *orientovaného grafu* na dané množině vrcholů/entit. Všimněme si, že závislosti často bývají časového charakteru (přičemž směr závislosti je implicitně jasný) a pak je nezbytnou doplňkovou podmínkou vyloučení výskytu orientovaných cyklů v modelovém grafu. Na druhou stranu existují i situace, kdy cyklické závislosti jsou dovoleny a mají svůj význam.

  Pro ještě jednu ukázku závislostí z běžného života informatika se podíváme na správu balíčků softwaru například v Linuxovýxh distribucích. V tom případě jsou jednotlivé balíčky vrcholy grafu, jejich vyžadované závislosti popisují odchozí hrany a jejich poskytované vlastnosti jsou příchozími hranami grafu závislostí. Korektní instalace zvoleného balíčku pak řeší problém zahrnutí všech dalších vrcholů „dosažitelných“ ze zvoleného. Vše je navíc komplikováno správou verzí balíčků, ale to už je mimo rámec našeho úvodního slova.

- Modelování technických či dopravních sítí grafy:

  V takových případech bývají vrcholy grafu jednotlivá technická zařízení jako třeba rozvodny, routery, křižovatky a podobně, kdežto hrany jsou tvořeny spojnicemi/vedením mezi vrcholy. Často se zde setkáváme s orientovanými grafy a obecně *multigrafy*. K této probematice se blíže vyjadřují následující Lekce 3 a 4.

- Vizualizace vztahů a závislostí pro lidského pozorovatele:

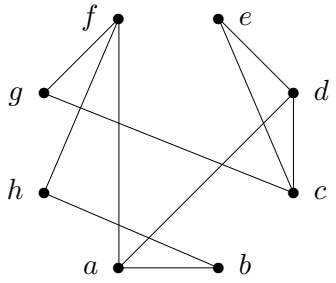
  Nejen při řešení cvičných příkladů v naší učebnici, ale i v mnoha reálných aplikacích využívajících grafy jako modely, je velmi potřebné tyto grafy *vizualizovat* (tj. hezky nakreslit) pro lidského pozorovatele. Jedná se obecně o poměrně obtížný úkol, který

přesahuje možnosti našeho textu, ale okrajově se o této obsáhlé problematice zmíníme také v pozdější Lekci **??**. □

Nyní nastává čas k procvičení jednotlivých pojmů zavedených v lekci. Začneme od toho skutečně jednoduchého.

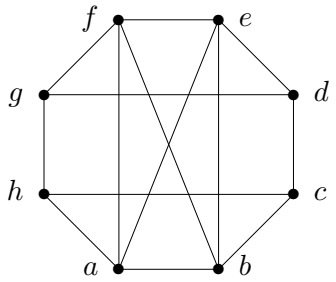
Příklad 1.25. *Zakreslete graf G na 8 vrcholech zadaný následujícím výčtem vrcholů a hran: $V(G) = \{a, b, c, d, e, f, g, h\}$ a $E(G) = \{ab, cd, ad, de, fh, fa, hb, ce, cg, gf\}$.*

V prvním kroku si zakreslíme a pojmenujeme vrcholy. Jak a kde je vrcholy vhodné nakreslit? Když nemáme dopředu žádnou představu o „tvaru“ našeho grafu G, asi nejjednodušší a přitom dostatečně přehledné je uspořádat nové vrcholy „do kruhu“, neboli do vrcholů pravidelného 8-úhelníku. Poté již jednoduše zakreslíme každou jednotlivou hranu:



□

Příklad 1.26. *Pro následující graf na 8 vrcholech zjistěte posloupnost stupňů jeho vrcholů a zjistěte, kolik graf obsahuje trojúhelníků.*



Posloupnost stupňů snadno nalezneme po řadě $4, 4, 3, 3, 4, 4, 3, 3$, neboli graf obsahuje po čtyřech vrcholech stupně 3 a čtyřech stupně 4. Počet hran je tudíž $(4 \cdot 3 + 4 \cdot 4)/2 = 14$. Ověřte si to pro kontrolu na obrázku.

Trojúhelníky snadno najdeme čtyři; abe, aef, abf, bef. Proč více trojúhelníků v obrázku nemáme? Všimněte si, že po odebrání ae, bf zůstane podgraf, který zjevně trojúhelník neobsahuje, a proto jediné trojúhelníky jsou ty obsahující ae nebo bf – tam už si snadno ověříme, že další takové trojúhelníky nejsou. □

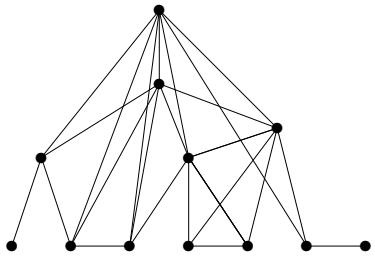
Příklad 1.27. *Existuje graf s posloupností stupňů $1, 1, 3, 3, 3, 4, 4, 4, 6, 6, 6, 7$?*

Daná posloupnost je již setříděná, jinak bychom ji nejprve vzestupně setřídili. Podle Věty 1.4 budeme posloupnost upravovat, přičemž vždy v řádku vypíšeme setříděnou posloupnost před odečtením a po odečtení (nesetříděnou).

$1, 1, 3, 3, 3, 4, 4, 4, 6, 6, 6, 7 \quad \to \quad 1, 1, 3, 3, 2, 3, 3, 3, 5, 5, 5$
$1, 1, 2, 3, 3, 3, 3, 3, 5, 5, 5 \quad \to \quad 1, 1, 2, 3, 3, 2, 2, 2, 4, 4$
$1, 1, 2, 2, 2, 2, 3, 3, 4, 4 \quad \to \quad 1, 1, 2, 2, 2, 1, 2, 2, 3$
$1, 1, 1, 2, 2, 2, 2, 2, 3 \quad \to \quad 1, 1, 1, 2, 2, 1, 1, 1$

$1,1,1,1,1,1,2,2 \quad \to \quad 1,1,1,1,1,0,1$
$0,1,1,1,1,1,1$ zřejmě tento graf existuje (3 samostatné hrany).

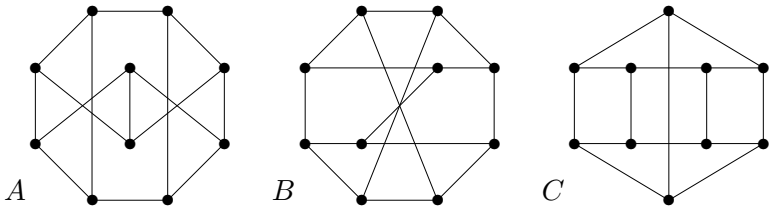
K poslední posloupnosti snadno nakreslíme graf, proto i graf s původní posloupností stupňů existuje. Zpětným přidáváním vrcholů pak můžeme sestrojit i jednu z možných podob původního grafu (který zajisté není jednoznačně určený!):



□

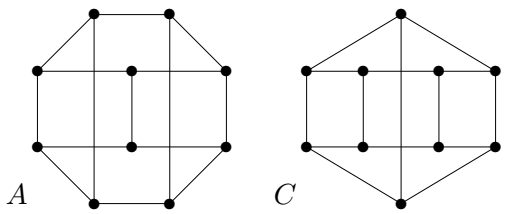
Velmi důležitou součástí učiva první lekce je zjišťování isomorfismu jednoduchých grafů. Jednoduché příklady tohoto typu se vyskytly už ve výkladu a další jsou mezi následujícími úlohami k samostatnému řešení. Nyní si ukážeme jeden z mírně složitějších příkladů (a další obtížnější příklady na grafový isomorfismus budou následovat v příštím speciálním oddíle).

Příklad 1.28. *Najděte všechny isomorfní dvojice grafů v následujících obrázcích tří* 10*-vrcholových grafů. Isomorfní dvojice odpovídajícím způsobem očíslujte, u neisomorfních toto zdůvodněte.*



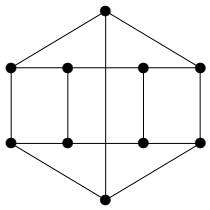
Postupujme systematicky: Všechny tři grafy mají po 10 vrcholech a všechny vrcholy stupňů 3. Takto jsme je tedy nijak nerozlišili. Podívejme se třeba na trojúhelníky v nich – opět si nepomůžeme, neboť žádný z nich trojúhelníky neobsahuje. Co se tedy podívat na obsažené kružnice C_4? Graf C jich jasně obsahuje pět, graf A po chvíli zkoumání také, ale v grafu B najdeme i při vší snaze jen tři kružnice délky 4. (Obdobného rozdílu si můžeme všimnout, pokud se zaměříme na kružnice C_5, zkuste si to sami.)

Takže co z dosavadního zkoumání plyne? Graf B nemůže být isomorfní žádnému z A, C. Nyní tedy zbývá najít (očekávaný) isomorfismus mezi grafy A a C. To se nám skutečně podaří poměrně snadno - stačí „prohozením“ prostředních dvou vrcholů u grafu A získat lepší obrázek



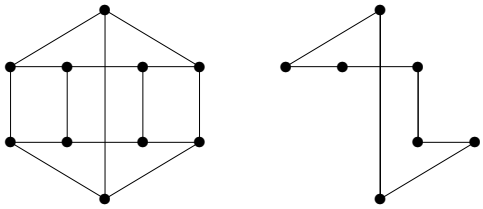
a odpovídající bijekce je na pohled zřejmá. (Doplňte si ji společným očíslováním vrcholů obou grafů.) □

Příklad 1.29. *Jaká je nejdelší kružnice a nejdelší indukovaná kružnice obsažená v následujícím grafu? Zdůvodněte.*



Co se týče nejdelší kružnice, ta nemůže být z principu delší než počet všech vrcholů, tedy 10. Přitom kružnici délky 10 v zadaném grafu snadno najdeme.

Co se týče indukované kružnice, ta musí s vybranými vrcholy podgrafu obsahovat i všechny hrany mezi nimi (a přesto to musí stále být jen kružnice). To zřejmě žádnou kružnicí délky 10 splněno není. Nenajdeme dokonce ani takové kružnice délky 9 a 8, nejdelší při troše snahy najdeme indukovanou kružnici délky 7, jako třeba na následujícím obrázku:



Závěrem se zamysleme, proč vlastně delší indukovanou kružnici (než 7) nelze v našem grafu nalézt. Pokud bychom, pro spor, nalezli indukovanou kružnici délky 8, tak by musela použít jak ze spodního, tak z horního pětiúhelníku po čtyřech vrcholech. (Nemůže totiž užít všech 5 vrcholů z jednoho pětiúhelníku, neboť to by už byla kružnicí délky jen 5.) Sami však snadným pokusem zjistíme, že v takové kružnici budou ještě hrany navíc, tudíž nebude indukovaná. □

Otázky a úlohy k řešení

(1.6.1) *Má více hran úplný graf K_9 nebo úplný bipartitní graf $K_{6,6}$?*

(1.6.2) *Kolik hran má graf se 17 vrcholy stupňů 4?*

(1.6.3) *Kolik vrcholů může mít obyčejný graf, který má všechny stupně rovny 3?*

(1.6.4) *Existuje graf s posloupností stupňů $1, 1, 1, 3, 3, 3, 4, 4, 4$?*

(1.6.5) *Existuje graf s posloupností stupňů $1, 1, 1, 1, 1, 1, 1, 1, 6, 6$?*

(1.6.6) *Pro která přirozená x existuje graf s posloupností stupňů $4, 4, 4, 8, 9, 10, 10, 10, 11, 12, 12, 12, 13, x$?*

(1.6.7) *Kolik existuje neisomorfních grafů s 6 vrcholy, všemi stupně 3? Nakreslete je.*

Návod: Podívejte se místo těchto grafů na jejich doplňky – dejte hrany právě tam, kde je původní graf nemá. Tím vzniknou grafy se všemi stupni $5 - 3 = 2$.

(1.6.8) *Kolik hran má graf s touto posloupností stupňů?*
A: $1, 1, 2, 2, 3, 3, 4, 4, 4, 4, 5, 6, 7$
B: $1, 1, 2, 2, 2, 2, 4, 4, 4, 5, 6, 7$
C: $1, 1, 2, 2, 2, 3, 3, 4, 4, 4, 5, 6, 7$

(1.6.9) *Najděte a vyznačte isomorfismus mezi následujícími dvěma grafy na* 7 *vrcholech.*

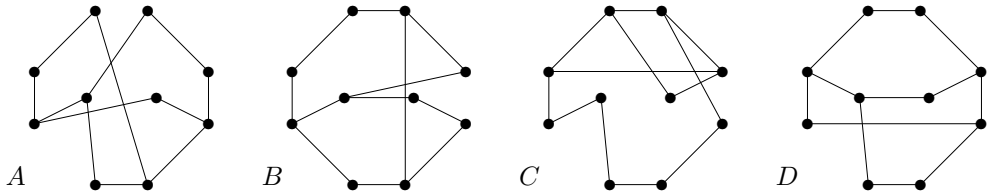
(1.6.10) *Vyznačte isomorfismus mezi následujícími dvěma grafy na* 8 *vrcholech:*

(1.6.11) *Vyznačte isomorfismus mezi následujícími dvěma grafy na* 8 *vrcholech:*

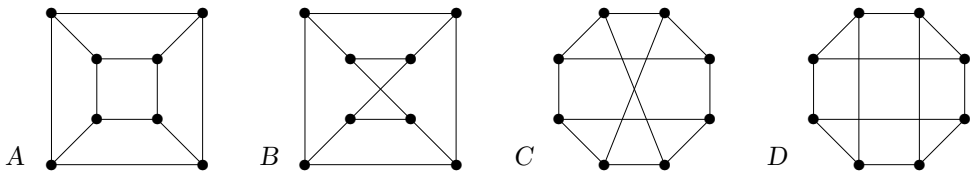
(1.6.12) *Jakou největší kliku obsahují grafy na obrázku?*

Návod: Překreslete si tyto (záměrně nepříliš přehledné) obrázky na „hezčí“.

(1.6.13) *Najděte mezi následujícími grafy všechny isomorfní dvojice a zdůvodněte svou odpověď.*



(1.6.14) *Najděte mezi následujícími grafy všechny isomorfní dvojice a zdůvodněte svou odpověď. (Isomorfní dvojice stejně očíslujte, u neisomorfních najděte odlišnosti.)*



Návod: Pokud vám třeba jedno očíslování pro isomorfismus nevyjde, musíte zkoušet další a další a probrat tak všechny možnosti.

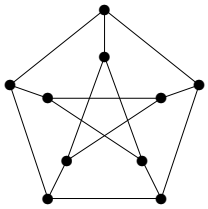
(1.6.15) *Jaká je nejdelší kružnice a nejdelší indukovaná kružnice obsažená v grafech A a B z Úlohy 1.6.14?*

(1.6.16) *Najdete v některém z výše uvedených obrázků kliku velikosti 4?*

(1.6.17) *Vraťte se zpět k příkladům a úlohám z Oddílu 1.2 a zkuste, zda je se znalostí nových metod jste schopni řešit lépe a elegantněji.*

(1.6.18) *Existují dva neisomorfní grafy s posloupnostmi stupňů*
A: 3,3,3,3,3,3,
B: 2,2,3,3,
C: 2,3,3,3,3 ?
U možnosti, kde dva neisomorfní grafy existují, je nakreslete. Pokud neexistují, pokuste se to správně zdůvodnit.

*(1.6.19) *Kolik podgrafů následujícího grafu je isomorfních kružnici C_9?*



*(1.6.20) *Kolik podgrafů úplného grafu K_{10} je isomorfních kružnici C_4?*

(1.6.21) *Kolik vrcholů má strom s 2004 hranami? Je to jednoznačné?*

(1.6.22) *Les má 2009 vrcholů a celkem 4 souvislé komponenty. Kolik má hran?*

(1.6.23) *Kolik komponent souvislosti má les s 2004 vrcholy a 1993 hranami?*

*(1.6.24) *Dokážete nalézt graf se dvěma kružnicemi, z něhož lze odebráním jedné hrany vytvořit strom? Zdůvodněte a případně nakreslete.*

(1.6.25) *Zorientujte hrany úplného grafu K_7 tak, aby z každého vrcholu vycházely nejvýše 3 šipky.*

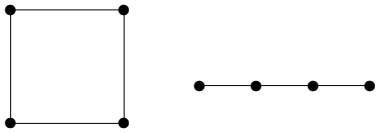
(1.6.26) *Proč v orientaci z úlohy 1.6.25 musí z každého vrcholu vycházet právě 3 šipky?*

1.7 Dodatek: Další úlohy k isomorfismu

Procvičování s (ne)isomorfismy grafů není samoúčelný dril, nýbrž pomáhá studujícím grafy a jejich vnitřní strukturu *lépe pochopit a „uchopit“*, také pomocí vhodně zvolených obrázků. Proto se tomuto specifickému typu úloh budeme podrobně věnovat v tomto dodatku ke cvičením. Dodatek lze také v prvním čtení textu vynechat (ale studenti předmětu MA010 na FI MU budou muset u zkoušek isomorfismus grafů velmi dobře ovládat!).

Příklad 1.30. *Kolik vzájemně neisomorfních jednoduchých grafů na 8 vrcholech existuje s posloupností stupňů* $1, 1, 2, 2, 2, 2, 2, 2$*?*

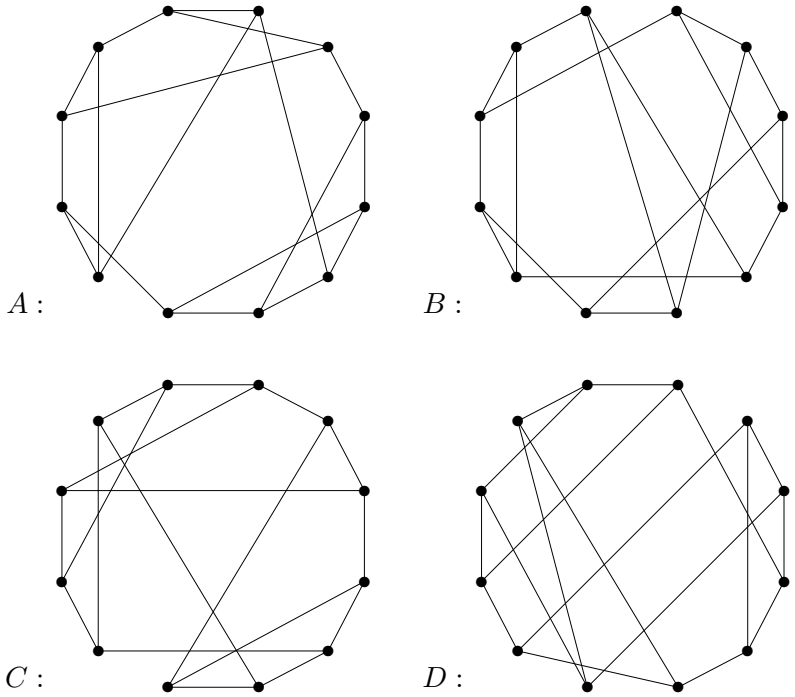
Nejprve si uvědomme, že každý graf musí obsahovat sudý počet vrcholů lichých stupňů, takže dva vrcholy stupně 1 musí být „u sebe“, neboli musí být součástí jedné cesty. Mimo to jedinými grafy se všemi stupni 2 jsou soubory kružnic, takže naše požadované grafy se skládají z jedné cesty a žádné až několika kružnic.



Použijeme-li k zápisu disjunktního sjednocení grafů symbol +, můžeme všechny možnosti systematicky vypsat: $P_1 + C_3 + C_3$, $P_1 + C_6$, $P_2 + C_5$, $P_3 + C_4$, $P_4 + C_3$, P_7. Existuje tedy právě 6 požadovaných grafů. □

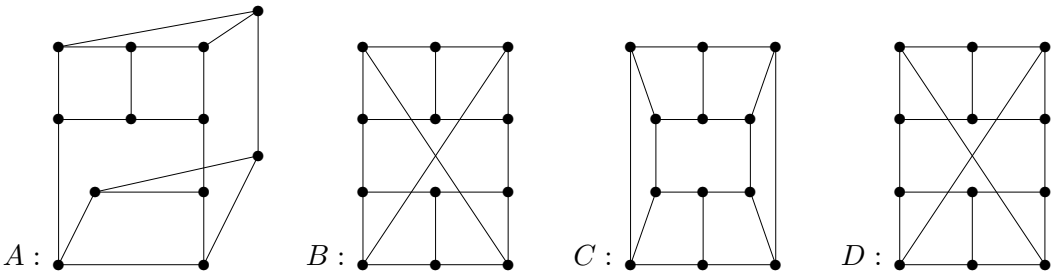
Všímejte si v dalších příkladech, jak významnou roli při řešení hraje nalezení vhodného nakreslení zkoumaného grafu!

Příklad 1.31. *Dány jsou následující čtyři jednoduché grafy na 12 vrcholech každý.*



Vašim úkolem je mezi nimi najít všechny isomorfní dvojice a matematicky zdůvodnit svou odpověď.

Zadané grafy si překreslíme například takto:

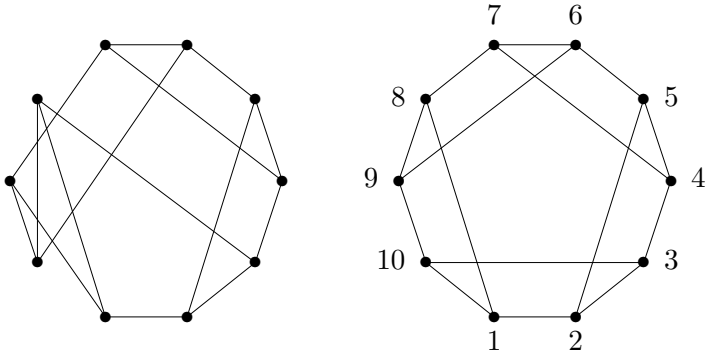


Jak jsme přišli zrovna na tato nakreslení? To nelze jednoduše vysvětlit, prostě si musíme několikrát každý graf zkoušet kreslit a představovat, až najdeme ten pravý pohled. (Ověřte si sami očíslováním, že nové obrázky jsou isomorfní zadaným... Chce to jen trochu cviku s kreslením grafů.)

Ihned tak vidíme, že $B \simeq D$. Pozor, nelze však nyní jen tak říci, že obrázky A a C vypadají jinak a tudíž nejsou s B isomorfní! Musíme najít jednoznačné zdůvodnění rozdílu mezi grafy. Grafy A i C například oba obsahují 6 kružnic délky 4 (vyznačte je v obrázku), ale v A existují vrcholy náležející zároveň do tří kružnic délky 4 (opět vyznačte), kdežto v C každým vrcholem procházejí právě dvě kružnice délky 4. Proto $A \not\simeq C$. Dále zdůvodníme v obrázku, že $B \simeq D$ obsahují celkem jen 4 kružnice délky 4, a tudíž $A \not\simeq B$ a $C \not\simeq B$ a stejně s D.

Tím jsme hotovi, existuje právě jedna isomorfní dvojice $B \simeq D$. □

Příklad 1.32. *Kolik navzájem neisomorfních jednoduchých grafů lze získat z následujícího grafu vlevo přidáním jedné hrany?*

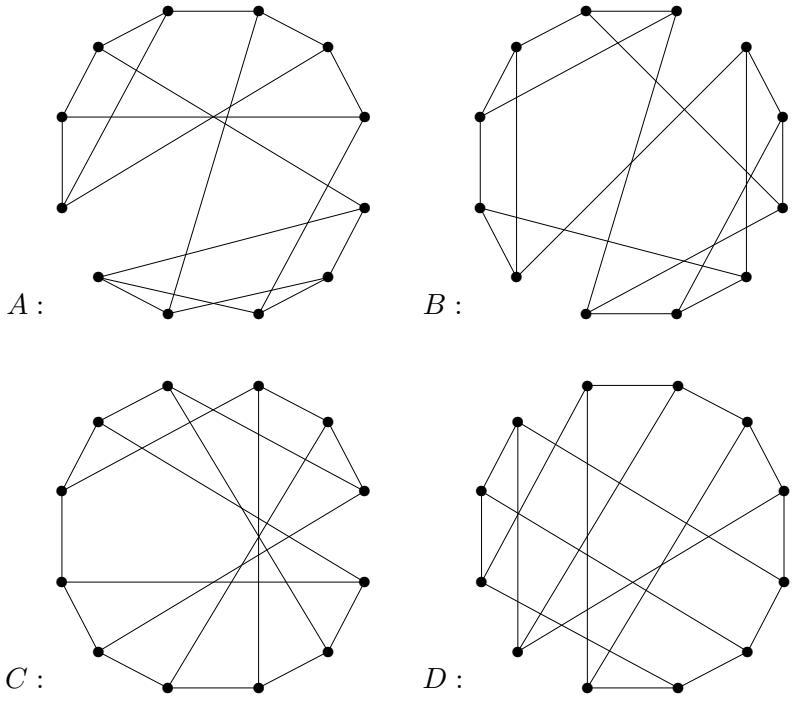


Prvním krokem k řešení příkladu jako tento je nalézt „co nejlepší“ obrázek našeho grafu. V tomto případě nalezneme isomorfní obrázek vpravo. (Jak, to lze jen těžko algoritmicky popsat, je k tomu potřeba zkušenost s grafy a představivost. . .)

Nyní vidíme, že všechny vrcholy našeho grafu jsou si navzájem symetrické, takže stačí uvažovat přidání hrany z jednoho z nich, třeba z 1. Nechť G_3 je graf vzniklý přidáním hrany $\{1, 3\}$, G_4 přidáním hrany $\{1, 4\}$ a G_5 přidáním hrany $\{1, 5\}$. Pak žádné dva z nich nejsou isomorfní, neboť G_3 obsahuje 2 trojúhelníky, G_5 obsahuje 1 trojúhelník a G_4 nemá žádný trojúhelník. Naopak přidáním hrany $\{1, 6\}$ vznikne graf isomorfní G_4, přidáním hrany $\{1, 7\}$ vznikne graf isomorfní G_5 a přidáním hrany $\{1, 9\}$ vznikne graf isomorfní G_3. (Kterému z našich tří grafů jsou tyto nové grafy isomorfní snadno odhadneme opět podle počtu trojúhelníku v nich, isomorfismus pak už najdeme standardním přístupem.)

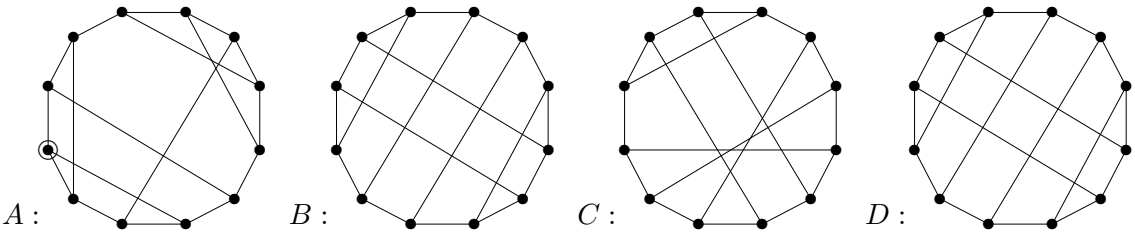
Takže odpověď je 3 vzájemně neisomorfní grafy. □

Příklad 1.33. *Dány jsou následující čtyři jednoduché grafy na* 12 *vrcholech každý.*



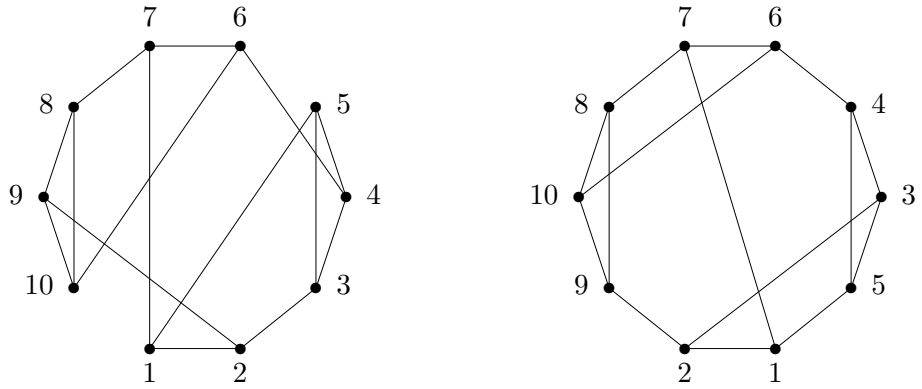
Vašim úkolem je mezi nimi najít všechny isomorfní dvojice a zdůvodnit svou odpověď.

Zadané grafy si opět (po pár pokusech) pěkně názorně překreslíme, například takto (procvičte si sami vyznačení isomorfismu mezi obrázky nahoře a dole):



Nyní je jasné, že $B \simeq D$. Zároveň jsou naše nové obrázky natolik „průzračné“, že v nich snadno můžeme spočítat počty kružnic délek 4: A, B i D jich obsahují 6, kdežto C je obsahuje jen 4 (vidíte proč?). Proto C není s žádným dalším isomorfní. Navíc A obsahuje vrchol incidentní se třema kružnicema délky 4 (vyznačený v obrázku vlevo), kdežto žádný takový vrchol v B ani D zřejmě neexistuje. Tudíž $A \not\simeq B$ a $C \not\simeq B$ a stejně s D. Tím jsme hotovi, existuje právě jedna isomorfní dvojice $B \simeq D$. □

Příklad 1.34. *Vašim úkolem je zjistit, kolik vzájemně neisomorfních grafů může vzniknout z následujícího grafu vlevo přidáním jednoho nového vrcholu x a (jediné) hrany z x do některého původního vrcholu. Svou odpověď aspoň stručně zdůvodněte.*

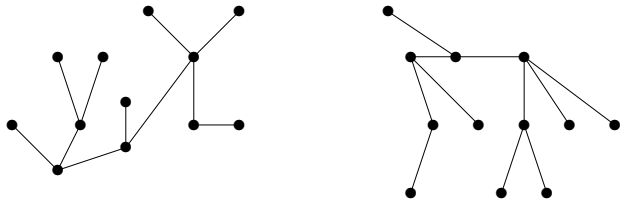


Nejprve si uvědomíme, že operace přidání nového vrcholu x má vlastně jen jediný účinek v našem příkladě – „označí“ sousední vrchol toho x. Po překladu tedy je úkolem nalézt, kolik vzájemně nesymetrických vrcholů náš graf má. V mírně překresleném obrázku téhož grafu vpravo vidíme *automorfismus*, tj. isomorfismus grafu sama na sebe, „středovou symetrií“. Z toho plynou symetrie mezi vrcholy $1 \sim 7$, $2 \sim 6$, $3 \sim 10$, $4 \sim 9$, $5 \sim 8$.

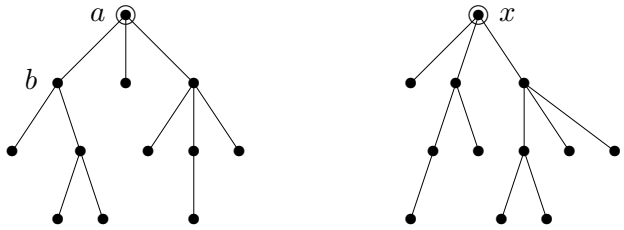
Nyní je třeba zdůvodnit, proč další dvojice symetrických vrcholů nejsou, neboli proč například neexistuje automorfismus našeho grafu mapující 1 na 2. To plyne třeba z faktu, že 1 sousedí s jedním trojúhelníkem grafu a 2 sousedí se dvěma. Navíc $3, 4, 5$ přímo v trojúhelníku jsou, takže s $1, 2$ nemohou být symetrické. 3 není symetrické s 4, neboť 3 leží ve 4-cyklu a 4 ne. Nakonec 5 není symetrické s 3 ani 4, protože 5 je sousedem 1, kdežto $3, 4$ jsou sousedé $2 \sim 6$, mezi kterými jsme již rozlišili.

Takže odpověď je 5 vzájemně neisomorfních grafů. □

Příklad 1.35. *Zjistěte, zda následující dva stromy jsou isomorfní pomocí Algoritmu 1.22.*



Nejprve si ověříme, že stromy mají stejný počet vrcholů (hran je pak také stejně). K daným dvěma stromům najdeme jejich centra a překreslíme si je jako kořenové stromy s kořeny v centrech.



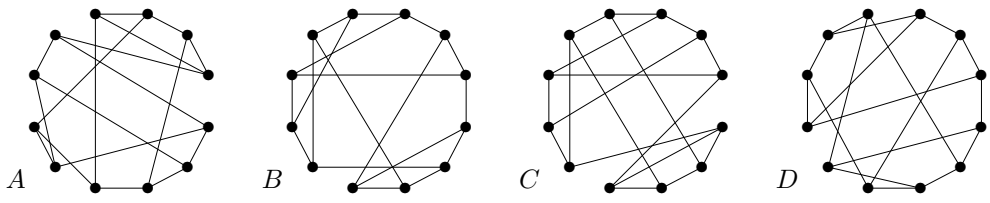
Nyní přejdeme k určení minimálního závorkového kódu pro levý strom (Algoritmus 1.22): Například při rekurzivním volání ve vrcholu b určíme kódy jeho podstromů jako '()' a '(()())'. Jelikož levou závorku považujeme za slovníkově menší, tyto dva podkódy seřadíme a spojíme takto '((()()) ())'. Obdobně postupujeme dále... Nakonec v kořeni a získáme rekurzivními voláními kódy jeho tří podstromů '((()()) ())', '()' a '((()) ()())'. Po seřazení a spojení nám celkový minimální kód levého stromu vyjde '((((()))()) (((()))()()) ())'.

Obdobně budeme postupovat při rekurzivním určování minimálního kódu pravého stromu. Zde nám podkódy jednotlivých tří podstromů kořene x vyjdou '()', '((()))()' a '(((()))()())'. Po seřazení a spojení nám celkový minimální kód pravého stromu vyjde '((((()))()()) (((()))()) ())'. Napíšeme si tedy získané dva kódy pod sebe a uvidíme, kde se liší ((((()))()) (((()))()) ())
((((()))()()) (((()))()) ()). Dané dva stromy tudíž nejsou isomorfní. □

Otázky a úlohy k řešení

(1.7.1) *Existují dva neisomorfní jednoduché grafy se stejnou posloupností stupňů*
A: 2,2,2,3,3,
B: 2,3,3,3,3,
C: 2,2,2,1,1,
D: 1,1,3,3,3,3 ?
U možnosti, kde dva neisomorfní grafy existují, je nakreslete. U zbylé možnosti pak správně zdůvodněte, proč dva takové neisomorfní grafy neexistují.

*(1.7.2) *Najděte mezi následujícími grafy všechny isomorfní dvojice a zdůvodněte svou odpověď.*



*(1.7.3) *Kolik navzájem neisomorfních jednoduchých grafů lze získat z grafu B v úloze 1.7.2 přidáním jedné hrany?*

*(**1.7.4**) *Najděte ručně všechny neisomorfní grafy se stupni* $3, 3, 3, 3, 2, 2, 2$.

Návod: Uvědomte si, že vrcholy stupně 2 lze „zanedbat“ – nahradit hranami. Ve výsledku pak zbydou 4 vrcholy stupňů 3, ale mezi nimi mohou být násobné hrany nebo i smyčky. Kreslete si obrázky.

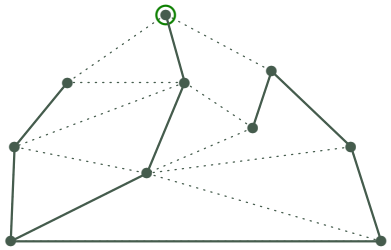
*(**1.7.5**) *Kolik navzájem neisomorfních jednoduchých grafů lze získat z grafu A v úloze 1.6.13 přidáním jedné hrany?*

*(**1.7.6**) *Určete, kolik neisomorfních stromů se všemi vrcholy stupňů* 1 *nebo* 3 *existuje na* 14 *vrcholech?*

2 Prohledávání a Souvislost Grafů

Úvod

Když se lidé dívají na grafy, obvykle je vnímají jako obrázky a jejich pohled je jakoby globální. Pokud je však graf zpracováván v počítači (a obzvláště pokud se jedná o skutečně velký graf), takový globální pohled nemáme je neudržitelný a graf musíme prohledávat lokálně. Z toho důvodu, než vůbec začneme uvažovat o využití a aplikacích grafů, je potřeba si ujasnit, jak by lokální prohledávání grafu mělo vypadat a co od něj můžeme očekávat.



S lokálním prohledáváním grafu jsou přímo svázány pojmy spojení v grafu a komponent souvislosti, neboli neformálně s možností se v určité oblasti grafu pohybovat (po hranách) odkudkoliv kamkoliv. To má jasné praktické motivace – například u počítačových, dopravních, telefonních či potrubních sítí. Při hlubším pohledu na tyto aplikace uvidíme také potřebu zachování spojení i v případech omezených lokálních výpadků, což je teoreticky podchyceno pojmy tzv. vyšší souvislosti grafů, nebo potřebu budovat svým způsobem minimální propojení v daném grafu, což vede na klasický problém minimální kostry. Poněkud složitější situaci pak najdeme v oblasti orientovaných grafů a jejich verzí souvislosti.

Cíle

Tato lekce ukazuje velmi obecné schéma pro lokální procházení celého grafu, které se dá využít v mnoha známých konkrétních prohledávacích algoritmech na grafech. Dále definujeme a vysvětlujeme pojmy souvislosti a komponent grafu. Zmíněny jsou i vyšší stupně souvislosti a silná souvislost orientovaných grafů. V nepodlední řadě je ukázán klasický problém minimální kostry grafu a jeho řešení (Jarníkův algoritmus) založené také na základním procházení grafu.

2.1 Algoritmické schéma procházení grafu

Výklad začneme obecným algoritmickým schématem *lokálního prohledávání grafu*. Tento algoritmus není v zásadě složitý a víceméně jen zobecňuje dobře známý postup procházení všech cest v bludišti. Pro vytvoření jeho co nejobecnějšího rámce vystačíme s následujícími datovými stavy a pomocnou datovou strukturou:

- *Vrchol:* má stavy ...
  - iniciační – dostane na začátku,
  - nalezený – poté, co jsme jej přes některou hranu nalezli (a odložili ke zpracování),
  - zpracovaný – poté, co jsme už probrali všechny hrany z něj vycházející.
  - (Případně ještě stav „post-zpracovaný“, po dokončení všech následníků.)
- *Hrana:* má stavy ...
  - iniciační – dostane na začátku,
  - zpracovaná – poté, co už byla probrána od jednoho ze svých vrcholů.
- *Úschovna:* je pomocná datová struktura (množina),
  - udržuje odložené, tj. nalezené a ještě nezpracované vrcholy.

Způsob, kterým se implementuje úschovna, určuje variantu algoritmu procházení grafu (viz dále). Zde pro ještě větší obecnost v úschovně udržujeme vrcholy spolu s příchozími hranami. V prohledávaných vrcholech a hranách se pak provádějí konkrétní programové akce pro zpracování našeho grafu.

Algoritmus 2.1. ***Generické procházení souvislého grafu*** G

Algoritmus projde a zpracuje každou hranu a vrchol souvislého grafu G. Konkrétní uživatelské programové akce potřebné ke zpracování grafu jsou provedeny v pomocných funkcích `ZPRACUJ()`.

```
vstup < graf G;
stav[ všechny vrcholy a hrany G ] ← iniciační;
úschovna U ← {(∅, v0)}, pro libovolný počáteční vrchol v0 grafu G;
strom prohledávání T ← ∅;
while (U ≠ ∅) {
    zvolit (e,v) ∈ U;
    U ← U \ {(e,v)};
    if (e ≠ ∅) ZPRACUJ(e);
    if (stav(v) ≠ zpracovaný) {
        foreach (f hrana vycházející z v) {
            w = opačný vrchol hrany f = vw;
            if (stav(w) ≠ zpracovaný) {                    (*)
                stav(w) ← objevený;
                U ← U ∪ {(f,w)};
            }
        }
        ZPRACUJ(v);
        stav(v) = zpracovaný;
        T = T ∪ {e,v};
    }
}
// (případný přechod na další nesouvislou část grafu G...)
G je zpracovaný;
```

K popsanému algoritmickému schématu doplňujeme několik poznámek:

- Algoritmus 2.1 je jen obecným schématem a ne úplným algoritmem. Mimo aplikačně závislého doplnění konkrétních akcí `ZPRACUJ()` je ještě velmi důležité specifikovat konkrétní implementaci úschovny `U` coby datové struktury a konkrétní provedení dosud nedeterministického kroku '`zvolit (e,v) ∈ U`' – právě toto rozlišuje různé varianty procházení (jako níže popsané BFS či DFS nebo jiné).
- Podmínka $(*)$ je dodatečná a zajišťuje zpracování každé hrany jen v jednom směru (což je přirozené pro neorientované grafy). Obecně je snadné v algoritmu nahlédnout, že každý vrchol či hrana grafu G budou zpracovány nejvýše jednou (tj. ne opakovaně). Pokud však $(*)$ vypustíme, bude každá hrana zpracována jednou v každém ze svých dvou směrů.
- V konkrétních implementacích procházení grafu lze často naše obecné schéma značně zjednodušit, jako například tím, že úschovna bude udržovat pouze vrcholy v místo dvojic (e, v) a nebude docházet k opakovanému ukládání. Některé aplikace procházení grafu však takto obecný přístup potřebují.
- Celkem Algoritmus 2.1 vždy vykoná jen lineárně mnoho kroků, kde za jeden krok považujeme i každé volání `ZPRACUJ()` a každou jednotlivou operaci s úschovnou `U`.

`Komentář:` Konkrétní postup algoritmu procházení bude ilustrován ve Cvičení 2.4.

Korektnost schématu procházení

Velkou výhodou *obecnosti*, kterou jsme si při představení způsobů procházení grafu zvolili v Algoritmu 2.1, je fakt, že správnost a dobré vlastnosti procházení nyní budeme umět dokázat najednou pro (téměř) všechny možné přístupy k procházení grafu. Jinými slovy, pokud nějaké konkrétní algoritmy popíšeme coby instance našeho obecného schématu procházení, nebudeme muset znovu a znovu pro každý dokazovat základní korektnost a složitost – ty budou přímým důsledkem následující Věty 2.2.

Věta 2.2. *Mějme souvislý graf G a jeho libovolný počáteční vrchol $v_0 \in V(G)$. Algoritmus 2.1 na vstupu G s počátkem v_0 zpracuje všechny vrcholy a hrany G, každý jednou.*

`Důkaz`: Tvrzení hned plyne ze tří jednoduchých faktů opírajících se o konkrétní pseudokód našeho algoritmu.

1. Každý element grafu G (vrchol nebo hrana), který je vložen do U, bude jednou zpracován. To proto, že dokud je $U \neq \emptyset$, opakuje se:

   ```
   zvolit (e,v) ∈ U;
   if (e ≠ ∅)  ZPRACUJ(e);
   if (stav[v] ≠ zpracovaný) { ...  ZPRACUJ(v); ...}
   ```

2. Pro každou hranu, která je kdy vložena do U, oba její konce jsou zpracovány (tj. ne jen ten konec, ze kterého je vložena). To je zjevné z pseudokódu souvisejícího se vkládáním hrany f do U:

   ```
   foreach (f hrana vycházející z v) {
       w = opačný vrchol hrany f = vw;
       if (stav(w) ≠ zpracovaný) { ... U ← U ∪{(f,w)}; }
   }
   ZPRACUJ(v);
   ```

   Při vkládání f do U je tudíž nutně už jeden konec f zpracován (v) a druhý konec (w) je vložen do U společně s f. Potom je w později zpracován podle předchozího bodu.

3. Zbývá potvrdit, že dříve nebo později se každá hrana grafu G dostane do U. Pro spor předpokládejme existenci hrany $f \in E(G)$, která do U nikdy nebude vložena. Z důvodu souvislosti G lze f zvolit jako takovou *nejbližší k* v_0. Nechť $h \in E(G)$ je některá hrana sdílející vrchol v s f a bližší k v_0; pak h musí v některém kroku být vložena do U jako součást dvojice $(h, v) \in U$. V každém případě pak jednou bude zpracováván vrchol v a všechny hrany vycházející z v, včetně f, budou vloženy do U, spor. □

Zajímavým a užitečným aspektem schématu Algoritmu 2.1 je konstruovaný graf T, zvaný *strom prohledávání*. Jak však víme, že se skutečně jedná o strom? To sice není okamžitě jasné, ale důkaz není složitý.

Tvrzení 2.3. *Podgraf $T \subseteq G$ sestrojený Algoritmem 2.1 je stromem a zároveň překlenujícím podgrafem grafu G (tzv. kostrou, viz Oddíl 2.3).*

`Důkaz`. Indukcí podle počtu iterací hlavního cyklu dokážeme,

```
while (U ≠ ∅) {  ... ...  T ← T∪{e,v}; } }
```

že T stromem s množinou vrcholů rovnou množině dosud zpracovaných vrcholů.

- V první iteraci je $T = \{\emptyset, v_0\}$, což reprezentuje jednovrcholový strom v_0. Platí.
- Nechť indukční předpoklad platí po i iteracích. V následující iteraci algoritmus volí $(e, v) \in U$. Pokud je v už zpracovaný, nic se ohledně T nezmění. Jinak

```
if (stav[v] ≠ zpracovaný) { ...  T ← T ∪ {e,v}; }
```

  a $T \leftarrow T \cup \{e, v\}$ znamená, že k T přidáváme nově zpracovaný vrchol v s hranou e spojující jej s některým stávajícím vrcholem T. Potom je $T \cup \{e, v\}$ opět souvislý a bez kružnic a tudíž strom.

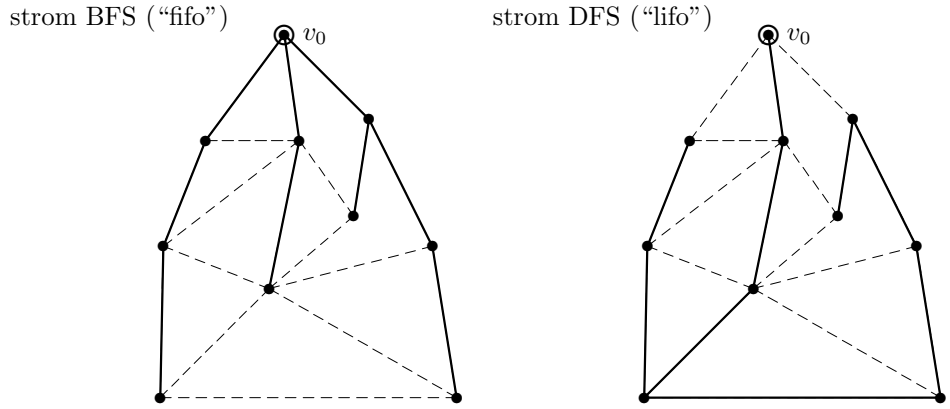
□

Základní varianty procházení

Pro odlišení různých způsobů procházení grafu jsou klíčové implementace úschovny `U` a pak způsob volby 'zvolit (e,v) ∈ U;'. Úplně základní příklady zahrnují:

- *Procházení „do šířky", BFS* – úschovna `U` je implementovaná jako fronta, neboli je voleno $(e, v) \in U$ od prvních vložených do úschovny.
- *Procházení „do hloubky", DFS* – úschovna `U` je implementovaná jako zásobník, neboli je voleno $(e, v) \in U$ od posledních vložených do úschovny. (Všimněte si, jakou důležitou roli konkrétně zde hraje opakované ukládání vrcholu do úschovny...)

`Komentář:` Rozdíl mezi BFS a DFS je dobře vidět na tvaru výsledného stromu procházení:



Pokročilejší aplikace schématu procházení grafu zahrnují například tyto algoritmy, se kterými se ještě v průběhu našeho kurzu setkáme.

- Rozpoznání *bipartitního grafu*: jednoduše použijeme BFS s dodatečnou rutinou ve `ZPRACUJ(e)`, která „střídá strany" mezi koncovými vrcholy hrany e – pokud toto střídání stran kdykoliv selže, nejedná se o bipartitní graf. Více se dozvíme v Lekci **??**.
- *Jarníkův algoritmus* pro minimální kostru (Oddíl 2.3) je implementován tak, že z úschovny `U` se vždy volí $(e, v) \in U$ takové, že e má nejmenší délku.
- *Dijkstrův algoritmus* pro nejkratší cesty v grafu (Lekce 3) pak volí $(e, v) \in U$ takové, že v je nejbližší vrchol k v_0 ze všech objevených vrcholů v U.

Na závěr si uvedeme jedno zřejmé zobecnění.

Algoritmus 2.4. ***Úprava Algoritmu 2.1 procházení pro orientovaný graf***
Místo řádku

```
foreach (f hrana vycházející z v) {
```

prostě použijeme

```
foreach (f šipka začínající ve v) {
```

a každou orientovanou hranu tak přirozeně projdeme právě jednou ve směru její šipky.

Otázky a úlohy k řešení

(2.1.1) *Jak je možné se v implementaci DFS podle schématu Algoritmu 2.1 vyhnout násobnému ukládání vrcholu do úschovny při zachování stejného výsledku?*

2.2 Spojení v grafu a Souvislost

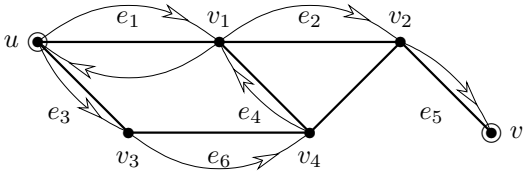
Souvislost grafu jsme v Lekci 1 definovali pomocí existence cest. Existuje však trochu obecnější a technicky vhodnější způsob, který místo ryzích cest v grafech popisuje obecné „procházky grafem“. Neformálně řečeno, při procházce grafem putujeme z vrcholu do sousedního vrcholu přes hrany grafu, ale nejsme (na rozdíl od cesty) vázáni podmínkou neopakovat stejné vrcholy ani hrany. Matematicky takové procházky podchytíme pojmem sledu v grafu:

`Definice`: *Sledem* W délky n v grafu G rozumíme posloupnost vrcholů a hran

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

ve které vždy hrana e_i má koncové vrcholy v_{i-1}, v_i. Konkrétně v tomto případě mluvíme o *sledu z vrcholu $u = v_0$ do $v_n = v$*, čímž pojmu sledu dodáváme implicitní orientaci i v neorientovaném grafu.

`Komentář`: V následujícím obrázku grafu je šipkami naznačen jeden sled z vrcholu u do vrcholi v, konkrétně se jedná o sled formálně zapsaný $(u, e_1, v_1, e_1, u, e_3, v_3, e_6, v_4, e_4, v_1, e_2, v_2, e_5, v)$.



Lema 2.5. *Mějme relaci $\sim$ na množině vrcholů $V(G)$ libovolného grafu G takovou, že pro dva vrcholy $u \sim v$ právě když existuje v G sled začínající v u a končící ve v. Pak $\sim$ je relací ekvivalence.*

`Důkaz`. Relace $\sim$ je reflexivní, neboť každý vrchol je spojený sám se sebou sledem délky 0. Symetrická je také, protože sled z u do v snadno obrátíme na sled z v do u. Stejně tak je $\sim$ tranzitivní, protože dva sledy se společným koncovým vrcholem můžeme na sebe navázat v jeden. □

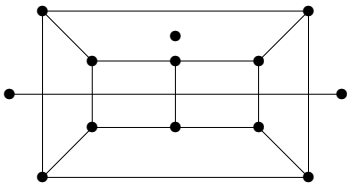
Dalším naším krokem je dokázat, že pokud definujeme souvislost grafu pomocí existence sledu, na rozdíl od existence cesty jako v Lekci 1, tak se vůbec nic nezmění. K tomu potřebujeme následující poznatek:

Věta 2.6. *Pokud mezi dvěma vrcholy grafu G existuje sled, pak mezi nimi existuje cesta.*

`Důkaz`. Ze všech sledů mezi vrcholy u a v v G vybereme sled W s nejmenší délkou. Je snadno vidět, že pokud W zopakuje některý vrchol grafu G, můžeme W ještě zkrátit, a to je spor s předpokladem. Proto je W cestou v G. □

`Definice`: Třídy ekvivalence výše popsané (Lema 2.5) relace $\sim$ na $V(G)$ se nazývají *komponenty souvislosti* grafu G. Jinak se taky komponentami souvislosti myslí podgrafy indukované na těchto třídách ekvivalence.

`Komentář:` Podívejte se, kolik komponent souvislosti má tento graf:



Vidíte v obrázku všechny tři komponenty? Jedna z nich je izolovaným vrcholem, druhá hranou (tj. grafem isomorfním K_2) a třetí je to zbývající.

Důsledek 2.7. *Graf je souvislý, právě když je tvořen nejvýše jednou komponentou souvislosti.*

`Poznámka`: Prázdný graf je sice souvislý, ale má 0 komponent souvislosti.

Vyšší úrovně souvislosti

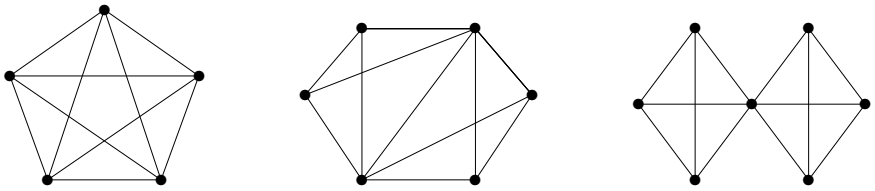
V aplikacích, kde grafy obvykle modelují různé druhy *sítí*, nás zajímá nejen, zdali se za normálních podmínek můžeme pohybovat mezi vrcholy/uzly (základní *souvislost*), ale také, jaké spojení můžeme garantovat v případě lokálních výpadků (odolnost a *redundance*). Toto lze teoreticky podchytit zkoumáním „vyšších“ stupňů souvislosti grafu.

`Definice`: Graf G je *hranově k-souvislý*, $k > 1$, pokud i po odebrání libovolných nejvýše $k-1$ hran z G zůstane výsledný graf souvislý.

`Definice`: Graf G je *vrcholově k-souvislý*, $k > 1$, pokud má G více než k vrcholů a i po odebrání libovolných nejvýše $k-1$ vrcholů z G zůstane výsledný graf souvislý.
Speciálně platí, že úplný graf K_n je vrcholově $(n-1)$-souvislý, ale ne n-souvislý, ale graf K_1 je 1-souvislý.

Pokud mluvíme jen o k-souvislém grafu, máme (obvykle) na mysli vrcholově k-souvislý graf. 1-souvislý graf je pouhé synonymum pro souvislý.

`Komentář:` Stručně řečeno, vysoká hranová souvislost znamená vysoký stupeň odolnosti sítě proti výpadkům spojení-hran, neboli síť zůstane stále dosažitelná, i když libovolných $k-1$ spojení bude přerušeno. Vysoká vrcholová souvislost je mnohem silnějším pojmem, znamená totiž, že celá síť zůstane dosažitelná i po výpadku libovolných $k-1$ uzlů-vrcholů (samozřejmě mimo těch vypadlých uzlů).



Na ilustračním obrázku má první graf vrcholovou souvislost 4 a snadno vidíme, že po odebrání tří vrcholů či hran zůstává souvislý. Z druhého grafu bychom museli odebrat

nejméně 3 hrany, aby se stal nesouvislým, a proto je jeho hranová souvislost 3. Na druhou stranu však stačí odebrat 2 vrcholy, aby mezi jeho levým a pravým krajním vrcholem žádné spojení nezůstalo. (Vidíte, které dva?) A jak je tomu u třetího grafu?

`Fakt`: Pokud je graf k-souvislý, pak je také $(k-1)$-souvislý.

Mengerova věta

Důkaz následujícího důležitého výsledku by nebyl jednoduchý při použití stávajících znalostí, proto jej ponecháme na pozdější Lekci 4, Důsledek 4.14.

Věta 2.8. *Graf G je hranově k-souvislý právě když mezi libovolnými dvěma vrcholy lze vést aspoň k hranově-disjunktních cest (vrcholy mohou být sdílené).*
Graf G je vrcholově k-souvislý právě když mezi libovolnými dvěma vrcholy lze vést aspoň k disjunktních cest (různých až na ty dva spojované vrcholy).

`Komentář:` Věta nám vlastně říká, že stupeň souvislosti grafu se přirozeně rovná stupni redundance spojení vrcholů. Na výše uvedeném obrázku mezi každými dvěma vrcholy grafu K_5 můžeme vést až 4 disjunktní cesty. U druhého grafu třeba mezi levým a pravým koncem lze vést jen 2 (vrcholově) disjunktní cesty, ale mezi každými dvěma vrcholy lze vést 3 hranově-disjunktní cesty.

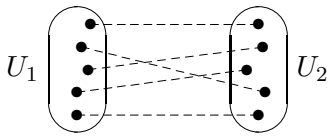
V duchu předchozí Mengerovy věty pokračujeme s následujícími poznatky.

Věta 2.9. *Nechť G je vrcholově 2-souvislý graf. Pak každé dvě hrany v G leží na společné kružnici.*

`Důkaz`: Nechť $e, f \in E(G)$. Sestrojíme graf G' podrozdělením obou hran e, f novými vrcholy v_e, v_f. Je zřejmé, že i G' je vrcholově 2-souvislý graf, takže podle Věty 2.8 existují v G' dvě disjunktní cesty spojující v_e s v_f, tvořící spolu kružnici C'. Nakonec C' indukuje v G kružnici C procházející e i f. □

Rozšířením předchozí úvahy lze dokonce dokázat:

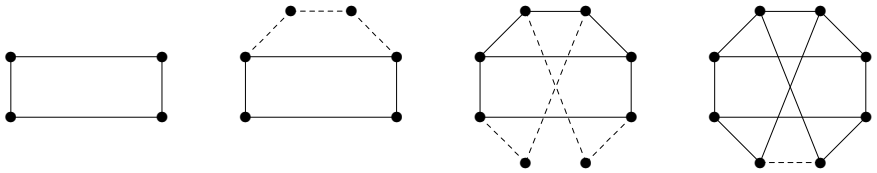
Věta 2.10. *Nechť G je vrcholově k-souvislý graf, $k \geq 1$. Pak pro každé dvě disjunktní množiny $U_1, U_2 \subset V(G)$, $|U_1| = |U_2| = k$ v G existuje k po dvou disjunktních cest z vrcholů U_1 do vrcholů U_2.*



Vrcholově 2-souvislé grafy mají následující jednoduchou konstruktivní chrakterizaci, která se vám může hodit například při pochopení a dokazování vlastností 2-souvislých grafů.

Věta 2.11. *Libovolný obyčejný graf je 2-souvislý, právě když jej lze vytvořit z kružnice „přidáváním uší“; tj. iterací operace, kdy libovolné dva stávající vrcholy grafu jsou spojeny novou cestou libovolné délky (ale ne paralelní hranou).*

`Komentář:` Ilustrací tohoto hezkého tvrzení jsou třeba následující obrázky...

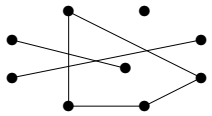


`Důkaz`: V jednom směru snadno nahlédneme, že graf G vzniklý z kružnice přidáváním uší je 2-souvislý.

V druhém směru zvolíme maximální 2-souvislý podgraf G' v G, který lze sestrojit z nějaké kružnice přidáváním uší. G' je neprázdný, neboť 2-souvislý G obsahuje aspoň kružnici. Pokud $V(G') = V(G)$, pak lze jako uši přidávat všechny zbylé hrany G, tudíž z maximality plyne $G' = G$. Takže existuje vrchol $x \in V(G) \setminus V(G')$. Pak lze dokázat, že z x vedou dvě vnitřně disjunktní cesty do různých dvou vrcholů G', které tvoří další ucho přidané k G'. (Existence těchto dvou cest plyne třeba z Věty 2.8, ale elementární krátký argument zatím v této lekci nemáme.) Opět máme spor s maximalitou G', tudíž opět $G' = G$. □

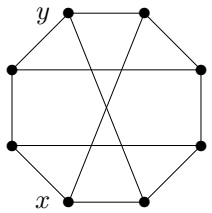
Otázky a úlohy k řešení

(2.2.1) *Kolik komponent souvislosti má tento graf?*



(2.2.2) *Jaký stupeň souvislosti má úplný bipartitní graf $K_{n,n}$?*

(2.2.3) *Kolik nejméně vrcholů mimo x, y musíme vypustit z nakresleného grafu, aby v něm nezbyla žádná cesta mezi vrcholy x a y? Zdůvodněte.*



(2.2.4) *Sestrojte graf K_5 „přidáváním uší“.*

(2.2.5) *Kolik nejméně hran musíte přidat k cestě délky 7, aby vznikl vrcholově 2-souvislý graf?*

(2.2.6) *Kolik nejvíce hran může mít nesouvislý graf na n vrcholech?*

*(2.2.7)** *Dokažte sami toto tvrzení: Každý 2-souvislý graf G na alespoň čtyřech vrcholech, který má všechny stupně ostře větší než 2, obsahuje hranu e takovou, že $G \setminus e$ je 2-souvislý.*

2.3 Hledání minimální kostry

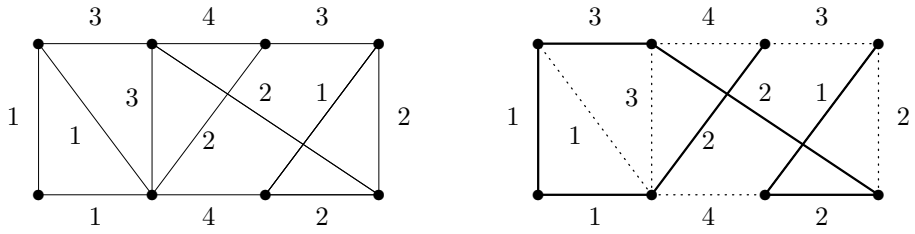
Pojmem *kostra* grafu G (angl. *spanning tree*) nazýváme strom, který je překlenujícím podgrafem G, tj. obsahuje všechny vrcholy G.

Problém 2.12. ***Problém minimální kostry (MST)***
Je dán (souvislý) vážený graf G, w s nezáporným ohodnocením hran w. Otázkou je najít kostru T v G, která má nejmenší možné celkové ohodnocení. Formálně

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

`Komentář`: Kostra daného grafu je minimální podgraf, který zachovává souvislost každé komponenty původního grafu. Proto nám vlastně ukazuje “minimální propojení” daných vrcholů, ve kterém ještě existují cesty mezi všemi dvojicemi, které byly propojeny i původně.

Praktickou formulací problému je třeba propojení domů elektrickým rozvodem, škol internetem, atd. Jedná se tady o zadání, ve kterých nás zajímá především celková délka či cena propojení, které je třeba vytvořit. Příklad je uveden na následujícím obrázku i s vyznačenou minimální kostrou vpravo.



Algoritmus 2.13. ***Jarníkův pro minimální kostru.***
Hrany na začátku neseřazujeme, ale začneme kostru vytvářet z jednoho vrcholu a v každém kroku přidáme nejmenší z hran, které vedou z již vytvořeného podstromu do zbytku grafu.

`Poznámka`: Tento algoritmus je velmi vhodný pro praktické výpočty a je dodnes široce používaný. Málokdo ve světě však ví, že pochází od Vojtěcha Jarníka, známého českého matematika — ve světové literatuře se obvykle připisuje Američanu Primovi, který jej objevil až skoro 30 let po Jarníkovi.

Algoritmus 2.14. ***„Hladový“ pro minimální kostru.***
Je dán souvislý vážený graf G, w s nezáporným ohodnocením hran w.

- Seřadíme hrany grafu G vzestupně podle jejich ohodnocení, tj. $w(e_1) \leq w(e_2) \leq \cdots \leq w(e_m)$.
- Začneme s prázdnou množinou hran $T = \emptyset$ pro kostru.
- Pro $i = 1, 2, \ldots, m$ vezmeme hranu e_i a pokud $T \cup \{e_i\}$ nevytváří kružnici, přidáme e_i do T. Jinak e_i “zahodíme”.
- Na konci množina T obsahuje hrany minimální kostry váženého grafu G, w.

`Komentář:` Podrobnou ukázku průběhu hladového algoritmu čtenář najde v Příkladě 2.27.

`Důkaz` správnosti Algoritmu 2.14:
Nechť T je množina hran získaná v Algoritmu 2.14 a nechť hrany jsou již seřazené $w(e_1) \leq w(e_2) \leq \cdots \leq w(e_m)$. Vezměme množinu hran T_0 té minimální kostry (může jich být více se stejnou hodnotou), která se s T shoduje na co nejvíce prvních hranách. Pokud $T_0 = T$, algoritmus pracoval správně.

Předpokládejme tedy, že $T_0 \neq T$, a ukážeme spor, tj. že toto nemůže ve skutečnosti nastat. Označme $j > 0$ takový index, že se množiny T_0 a T shodují na prvních $j-1$ hranách $e_1, \ldots, e_{j-1}$, ale neshodují se na e_j. To znamená, že $e_j \in T$, ale $e_j \notin T_0$. (Jistě nemůže nastat $e_j \notin T$, ale $e_j \in T_0$.) Podle Důsledku 1.14 obsahuje graf na hranách $T_0 \cup \{e_j\}$ právě jednu kružnici C. Kružnice C však nemůže být obsažena v nalezené kostře T, a proto existuje hrana e_k v C, která $e_k \notin T$ a zároveň $k > j$. Potom však je $w(e_k) \geq w(e_j)$ podle našeho seřazení hran, a tudíž kostra na hranách $\big(T_0 \setminus \{e_k\}\big) \cup \{e_j\}$ (vzniklá nahrazením hrany e_k hranou e_j) není horší než T_0 a měli jsme ji v naší úvaze zvolit místo T_0. To je hledaný spor. □

`Komentář:` Správný pohled na předchozí důkaz by měl být následovný: Předpokládali jsme, že nalezená kostra T se s některou optimální kostrou shoduje aspoň na prvních $j-1$ hranách.

Poté jsme ukázali, že některou další hranu e_k v (předpokládané) optimální kostře lze zaměnit hranou e_j, a tudíž dosáhnout shodu aspoň na prvních j hranách. Dalšími iteracemi záměn ukážeme úplnou shodu naší nalezené kostry T s některou optimální kostrou. V našem důkaze jsme se vlastně zaměřili na fakt, že ta poslední iterace záměn nemůže selhat. Nakreslete si tento důkaz obrázkem!

Doplňkové otázky

(2.3.1) *Co se stane, pokud v Algoritmu 2.14 seřadíme hrany naopak, tedy sestupně?*

(2.3.2) *Čím je Jarníkův algoritmus pro MST výhodnější než hladový postup?*

(2.3.3) *Promyslete si Jarníkův algoritmus, jaké datové struktury potřebujete pro jeho co nejrychlejší implementaci?*

2.4 Souvislost v orientovaných grafech

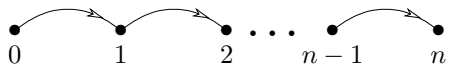
Třebaže orientované grafy jsou jen okrajovou součástí v našem výkladu látky, pojem orientované (silné) souvislosti je natolik fundamentálně odlišný (což je dané jeho „směrovostí“), že si zaslouží podrobný samostatný oddíl. Začneme analogicky Oddílu 2.2.

`Definice`: *Orientovaným sledem* délky n v orientovaném grafu D rozumíme střídavou posloupnost vrcholů a orientovaných hran

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

ve které vždy hrana e_i míří z vrcholu v_{i-1} do vrcholu v_i.

Věta 2.15. *Pokud mezi dvěma vrcholy grafu D existuje orientovaný sled, pak mezi nimi existuje orientovaná cesta.*

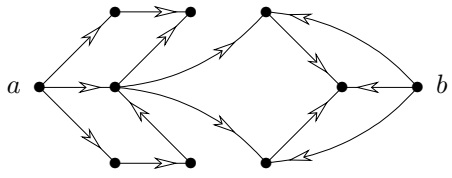


Pohledy na orientovanou souvislost

Prvním možným pohledem na souvislost orientovaných grafů je prostě požadovat tradiční grafovou souvislost po „zapomenutí“ směru šipek. Toto se nazývá *slabá souvislost*, avšak nemá valného významu, neboť proč bychom uvažovali orientované grafy a zároveň zapomínali směr jejich hran? Jiný přístup je následovný:

`Definice`: Orientovaný graf D je *dosažitelný směrem ven*, pokud v něm existuje vrchol $v \in V(D)$ takový, že každý vrchol $x \in V(D)$ je dosažitelný orientovaným sledem z v.

`Komentář`: Podrobným zkoumáním následujícího obrázku zjistíme, že tento graf není dosažitelný směrem ven, neboť chybí možnost dosáhnout vrchol b úplně vpravo. Na druhou stranu po vypuštění b je zbylý graf dosažitelný ven z vrcholu a vlevo.

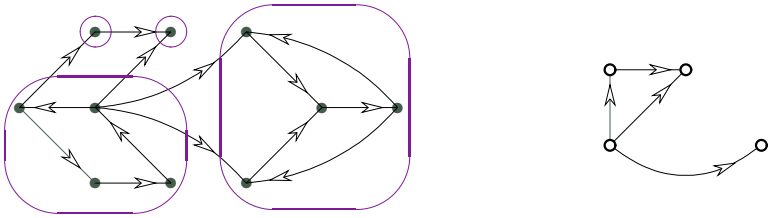


Nakonec „symetrizací“ přístupu dosažitelnosti se dobereme definici tzv. *silné souvislosti*, která je nejčastěji zmiňována u orientovaných grafů.

Lema 2.16. *Nechť $\approx$ je binární relace na vrcholové množině $V(D)$ orientovaného grafu G taková, že $u \approx v$ právě když existuje dvojice orientovaných sledů – jeden z u do v a druhý z v do u v grafu D. Pak $\approx$ je relace ekvivalence.*

Definice 2.17. ***Silné komponenty*** orientovaného grafu D jsou třídy ekvivalance relace $\approx$ z Lematu 2.16. Orientovaný graf D je *silně souvislý* pokud má nejvýše jednu silnou komponentu.

`Komentář:` Pro ilustraci si uvedem následující příklad orientovaného grafu vlevo, jenž má čtyři vyznačené silné komponenty.



Vpravo zároveň uvádíme pro ilustraci obrázek kondenzace silných komponent tohoto grafu, viz následující definice.

Kondenzace orientovaného grafu

`Definice`: Orientovaný graf, jehož vrcholy jsou tvořeny jednotlivými silnými komponentami orientovaného grafu D a šipky vedou právě mezi těmi dvojicemi různých komponent, mezi kterými vedou hrany v D, nazveme *kondenzací* grafu D.

`Definice`: Orientovaný graf D je *acyklický*, pokud neobsahuje jako podgraf orientovanou kružnici.

Tvrzení 2.18. *Kondenzace každého orientovaného grafu je acyklický orientovaný graf.*

`Důkaz` sporem: Nechť Z je kondenzace orientovaného grafu D. Pokud C je orientovaná kružnice v Z, tak podle definice silných komponent jí odpovídá orientovaná kružnice C' v původním D. To je však ve sporu, neboť C' by musela být součástí jedné silné komponenty a Z podle definice neobsahuje smyčky. □

Otázky a úlohy k řešení

(2.4.1) *V neorientovaném grafu platí, že každá hrana náleží právě jedné komponentě souvislosti. Lze totéž dokázat pro orientované grafy?*

(2.4.2) *Jak vypadá v orientovaném grafu silná komponenta o dvou vrcholech?*

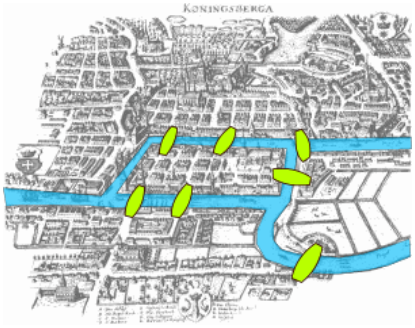
(2.4.3) *Kdy je v orientovaném grafu sám vrchol silnou komponentou?*

(2.4.4) *Dokažte následující: Orientovaný graf bez smyček je acyklický, právě když jeho silné komponenty jsou jednotlivé vrcholy.*

2.5 Dodatek: Eulerovské grafy

Pravděpodobně *nejstarší zaznamenaný výsledek teorie grafů* pochází od Leonarda Eulera – jedná se o problém slavných 7 mostů v Královci / Königsbergu / dnešním Kaliningradě. Tuto část, či spíše matematickou hříčku, o kreslení grafů „jedním tahem“ tak zařazujeme na závěr především z historických důvodů. Má však i některé zajímavé důsledky v jiných oblastech grafů, jak uvidíme ve cvičných příkladech.

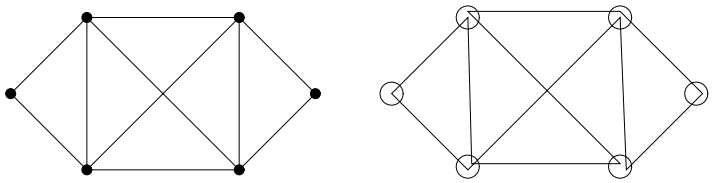
Komentář:



O jaký problém se tehdy jednalo? Městští radní chtěli vědět, zda mohou suchou nohou přejít po každém ze sedmi vyznačených mostů právě jednou. Rozbor tohoto problému vede k následující definici a odpovědi.

Definice: *Tah* je sled v grafu bez opakování hran.
Uzavřený tah je tahem, který končí ve vrcholu, ve kterém začal. *Otevřený tah* je tahem, který končí v jiném vrcholu, než ve kterém začal.

Komentář: Mějme následující příklad grafu vlevo, jehož *nakreslení jedním tahem* je vyznačeno vpravo.



Onen slavný výsledek teorie grafů od Leonharda Eulera poté zní:

Věta 2.19. *Graf G lze nakreslit jedním uzavřeným tahem právě když G je souvislý a všechny vrcholy v G jsou sudého stupně.*

Důsledek 2.20. *Graf G lze nakreslit jedním otevřeným tahem právě když G je souvislý a všechny vrcholy v G až na dva jsou sudého stupně.*

Důkaz: Dokazujeme oba směry ekvivalence. Pokud lze G nakreslit jedním uzavřeným tahem, tak je zřejmě souvislý a navíc má každý stupeň sudý, neboť uzavřený tah každým průchodem vrcholem „ubere“ dvě hrany.

Naopak zvolíme mezi všemi uzavřenými tahy T v G ten (jeden z) nejdelší. Tvrdíme, že T obsahuje všechny hrany grafu G.

- Pro spor vezměme graf $G' = G \setminus E(T)$, o kterém předpokládejme, že je neprázdný. Jelikož G' má taktéž všechny stupně sudé, je (z indukčního předpokladu) libovolná jeho hranově-neprázdná komponenta $C \subseteq G'$ nakreslená jedním uzavřeným tahem T_C.
- Vzhledem k souvislosti grafu G každá komponenta $C \subseteq G'$ protíná náš tah T v některém vrchole w, a tudíž lze oba tahy T_C a T „propojit přes w“. To je spor s naším předpokladem nejdelšího možného T.

□

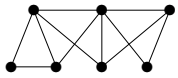
Důkaz důsledku: Nechť u, v jsou dva vrcholy grafu G mající lichý stupeň, neboli dva (předpokládané) konce otevřeného tahu pro G. Do G nyní přidáme nový vrchol w spojený hranami s u a v. Tím jsme náš případ převedli na předchozí případ grafu se všemi sudými stupni. □

Otázky a úlohy k řešení

(2.5.1) *Kterému grafu na 7 vrcholech odpovídá výše zakreslených 7 mostů v Královci?*

Doplňkové otázky

(2.5.2) *Kolik otevřených tahů budeme potřebovat na nakreslení grafu se čtyřmi vrcholy lichého stupně?*



(2.5.3) *Lze tento graf nakreslit jedním otevřeným tahem?*

(2.5.4) *Kolik hran musíte přidat ke grafu z předchozí otázky, aby se dal nakreslit jedním uzavřeným tahem?*

Rozšiřující studium

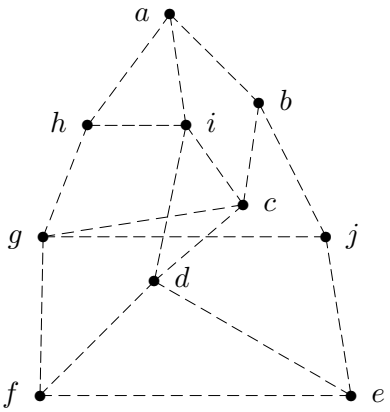
Základní souvislosti grafů se blíže teoreticky věnují [5, Oddíly 4.2, 4.7] a Eulerovským grafům obsáhleji [5, Oddíly 4.5, 4.6]. Algoritmy pro procházení grafu jsou podrobně popsány (včetně netriviálních aplikací) v [3] a demonstrovány třeba v `http://kam.mff.cuni.cz/~ludek/Algovision/Algovision.html`. *Za zmínku stojí i [4, Kapitola 3]. Pro hlubší teoretické poznatky a aplikace vyšší souvislosti grafů odkazujeme čtenáře na [1, Chapter 3].*

Mnohá spíše implementačně zaměřená pojednání o grafových algoritmech lze v dnešní době snadno nalézt ve Wikipedii. Konkrétní odkazy lze získat třeba ze stránky `http://en.wikipedia.org/wiki/Connected_component_(graph_theory)`. *Pro čtenáře by mohly být užitečné třeba algoritmy pro nalezení silných komponent orientovaného grafu* `http://en.wikipedia.org/wiki/Strongly_connected_components` *nebo 2-souvislých komponent (bloků), které oba vycházejí z prohledávání do hloubky. Jedná se o velmi zajímavé algoritmy chytře využívající zdánlivě primitivního postupu prohledávání, které studentům doporučujeme ke studiu.*

2.6 Cvičení: Souvislost a minimální kostry

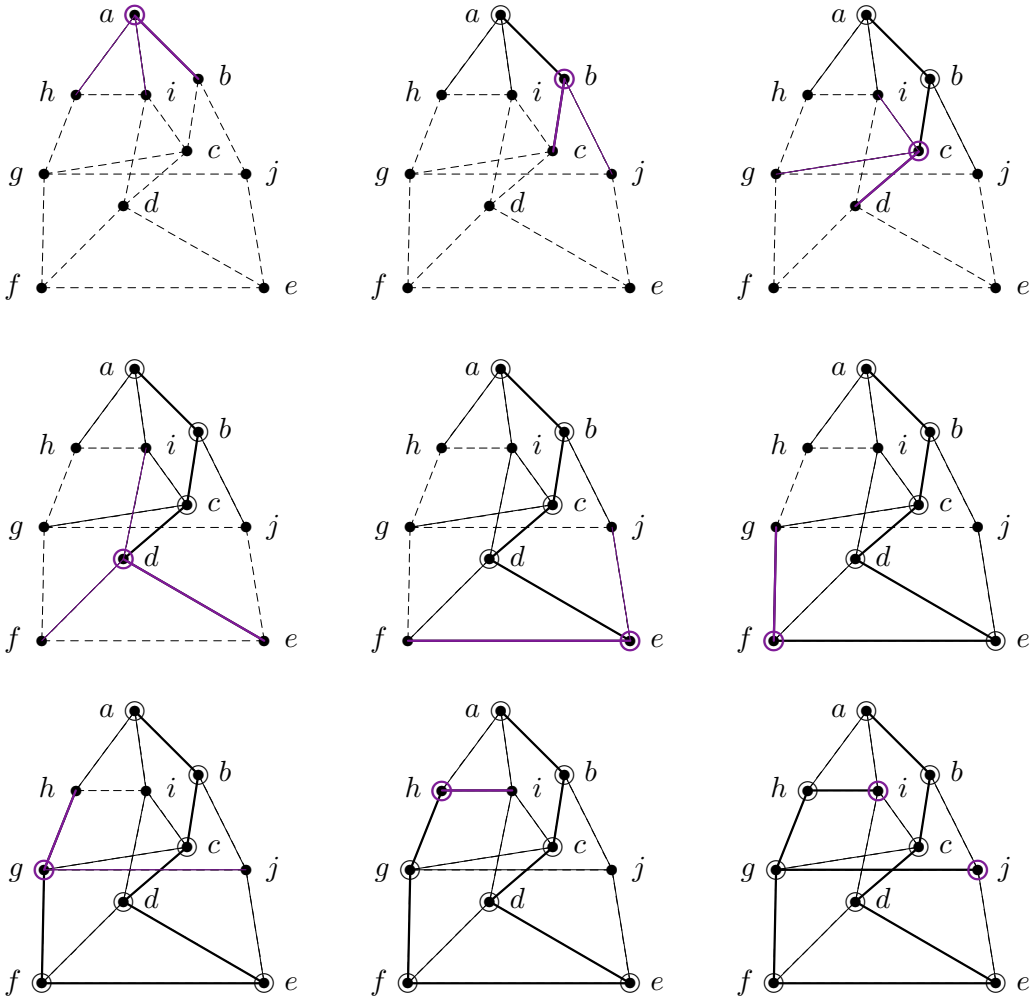
Následující cvičení je spíše demonstračního charakteru, jeho hlavním cílem je názornými ukázkami lépe přiblížit koncepty algoritmického procházení grafu a jeho souvislosti a problém minimální kostry.

Příklad 2.21. *Ukázka průchodu následujícím grafem do hloubky z vrcholu a.*



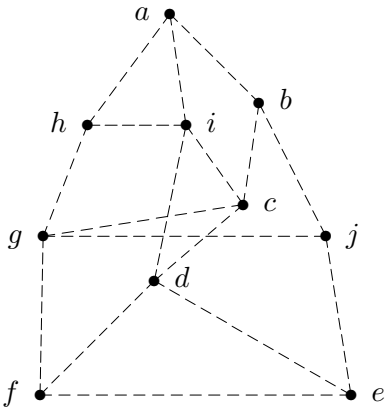
V této ukázce budeme obrázky znázorňovat stavy Algoritmu 2.1 v jednotlivých krocích takto: Neprohledané hrany jsou čárkované, prohledané hrany plnou čarou a

hrany, které vedly k nalezení vrcholů, jsou tlustou čarou barevně zvýrazněny (tyto hrany tvoří strom prohledávání T). Nalezené a zpracované vrcholy jsou značeny kroužkem, právě zpracovávaný vrchol opět barevně zvýrazněn.

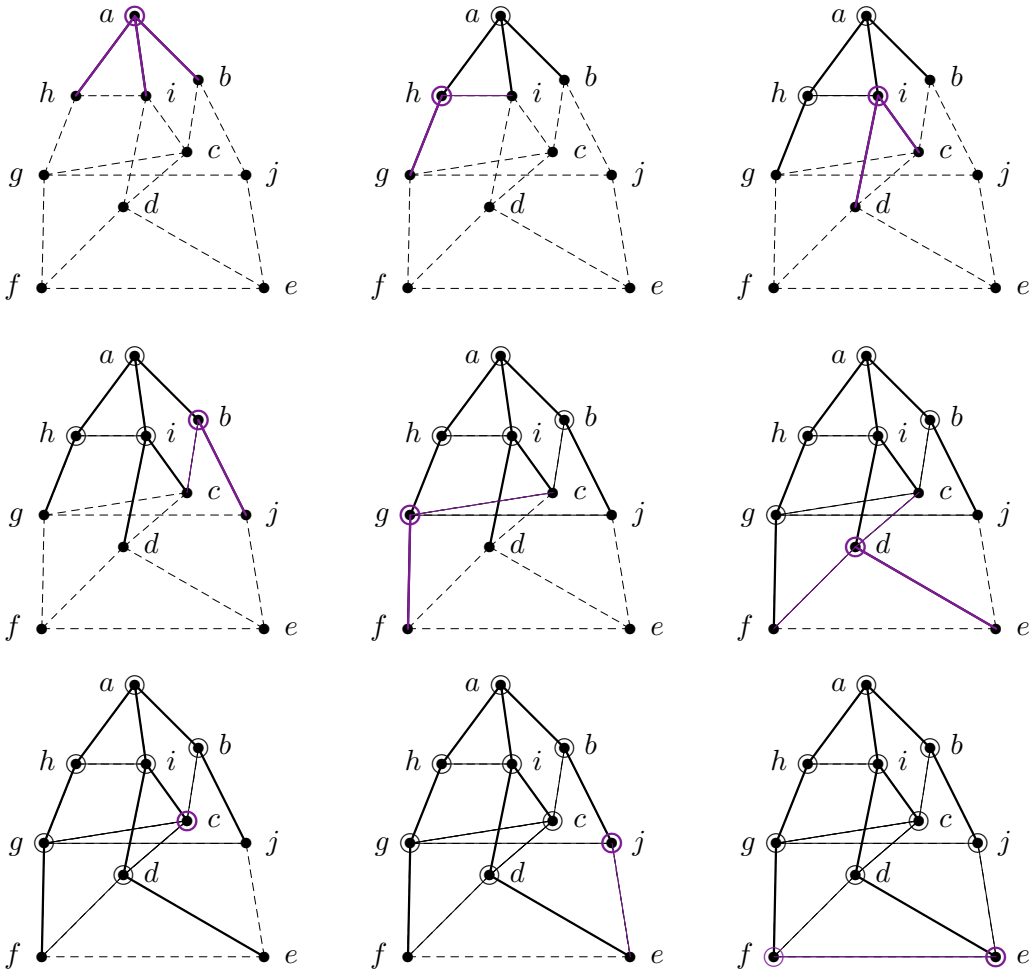


Tímto zpracování zadaného grafu skončilo. Mimo jiné jsme zjistili, že graf má jedinou komponentu souvislosti. □

Příklad 2.22. *Ukázka průchodu následujícím grafem do šířky z vrcholu a.*



V této ukázce budeme obrázky znázorňovat stavy Algoritmu 2.1 stejně jako v předchozím příkladě.



Tímto zpracování zadaného grafu skončilo. Vidíte rozdíly tohoto průchodu proti předchozímu příkladu? □

Souvislost grafu

Příklad 2.23. *Jak byste za pomocí terminologie grafové souvislosti vyjádřili míru odolnosti například počítačové sítě S proti výpadkům jednotlivých spojení či jejích uzlů?*

V souladu s Příkladem 1.24 budeme síť S modelovat (multi)grafem tak, že jednotlivé uzly budou vrcholy a spojení budou hrany. Předpokládáme pro jednoduchost duplexní síť, tj. neorientovaný graf G_S.

- Síť S bude odolná proti současnému výpadku k spojení, právě když graf G_S je hranově k-souvislý.
- Síť S bude odolná proti současnému výpadku k uzlů či spojení, právě když graf G_S je vrcholově k-souvislý.

Ja bychom však mohli modelovat situaci, kdy síť není všude duplexní? V takové případě je zapotřebí uvažovat orientovaný multigraf, ve kterém každému duplexnímu spojení odpovídá dvojice protiběžných šipek a ne-duplexnímu spojení příslušná jedna šipka. Poté bude zapotřebí přirozeným způsobem rozšířit definici k-souvislosti na orientovaný případ silné souvislosti. Zkuste si příslušnou definici zapsat sami... □

Příklad 2.24. *Mějme graf H_3, jehož vrcholy jsou všechny podmnožiny množiny $\{1,2,3\}$ a hrany spojují právě disjunktní dvojice podmnožin. (Tj. H_3 má 8 vrcholů.) Rozhodněte, zda je H_3 souvislý graf, a napište, kolik má H_3 hran.*

Vrchol $\emptyset$ odpovídající prázdné množině je spojený se všemi sedmi ostatními vrcholy, takže graf je souvislý. Nakreslete si jej! Každý vrchol i-prvkové podmnožiny je pak spojený s 2^{3-i} disjunktními podmnožinami doplňku. Celkem tedy máme součtem všech stupňů $\frac{1}{2}\big(7 + 3 \cdot 2^2 + 3 \cdot 2^1 + 2^0\big) = 13$ hran. □

Příklad 2.25. *Dokažte následující poznatek: Slabě souvislý orientovaný graf je silně souvislý právě tehdy, když každá jeho hrana leží v nějaké oritentované kružnici.*

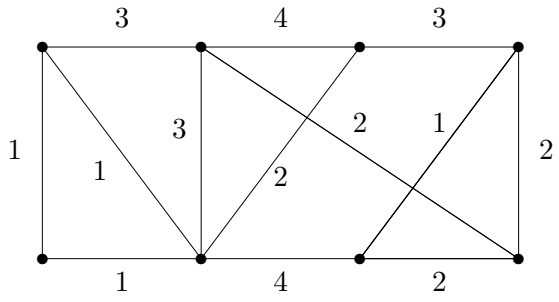
Nechť je orientovaný graf D silně souvislý a $(u,v) \in E(D)$ je libovolná jeho hrana. Podle definice tudíž existuje v D orientovaný sled z v do u, a tudíž i orientovaná cesta P_{vu} podle Věty 2.6. Pak $P_{vu} \cup \{(u,v)\}$ je orientovaná kružnice.

Naopak nechť je orientovaný graf D slabě souvislý a $u, v \in V(D)$ jsou jeho libovolné dva vrcholy. Podle definice slabé souvislosti existuje neorientovaná cesta Q z u do v (po „zapomenutí“ směru šipek D). Navíc pro každou hranu $e \in E(Q)$ existuje podle předpokladu oritentovaná kružnici obsahující $e = (x,y)$, neboli také orientovaná cesta P_e z y do x. Vhodným složením hran z Q a těchto cest P_e získáme orientovaný sled z u do v. Tudíž je D silně souvislý. □

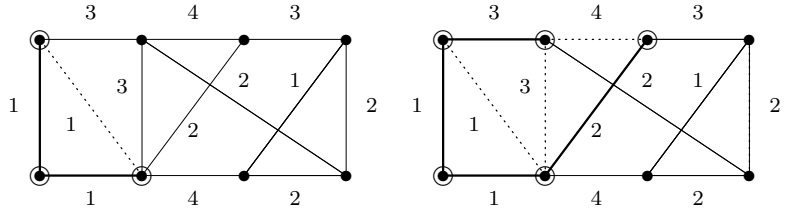
Minimální kostry

Problematiku minimální kostry taktéž ilustrujeme jednoduchými ukázkami postupů jejího hledání.

Příklad 2.26. *Najděme minimální kostru v zakresleném váženém grafu Jarníkovým Algoritmem 2.13.*

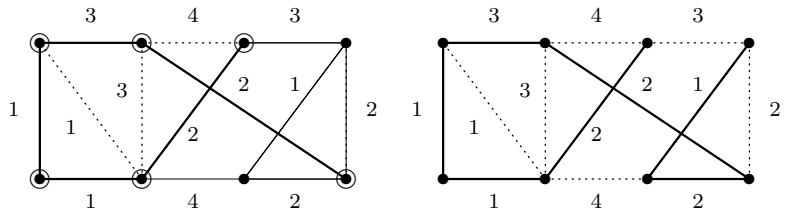


Kostru začneme budovat v levém dolním vrcholu. (Tj. naše částečná kostra zatím dosáhla jen na levý dolní vrchol.) Vidíme, že obě hrany z něj vycházející mají váhu 1, proto je obě do kostry přidáme. Takto naše budovaná kostra již dosáhla na tři vrcholy grafu, které si v následujícím obrázku vlevo vyznačíme. (Zbylé hrany mezi dosaženými vrcholy jsou již k ničemu, proto je zahodíme.)



V dalším kroku ze všech tří dosažených vrcholů vybíráme nejmenší hranu vedoucí mimo ně. Jak hned vidíme, nejmenší taková hrana má váhu 2, takže ji k naší kostře přidáme a dosáhneme čtvrtý vrchol grafu. Poté opět vybíráme nejmenší hranu z dosažených vrcholů ven, ale ty mají nyní váhu nejméně 3 (zvolíme tu nahoře vlevo). Také ji přidáme do kostry a dostaneme se do situace vyznačené na horním obrázku vpravo.

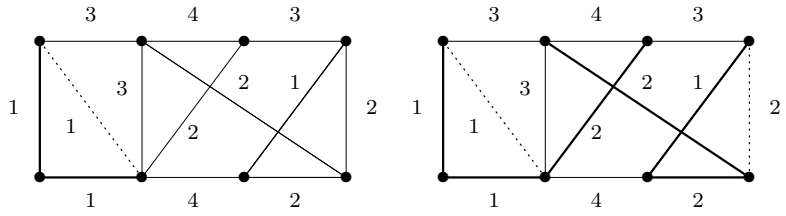
Nyní již je dosažených vrcholů 5 a nejmenší z hran vedoucích z dosažených ven má váhu 2. Takto dosáhneme i pravého dolního vrcholu, na následujícím obrázku vlevo.



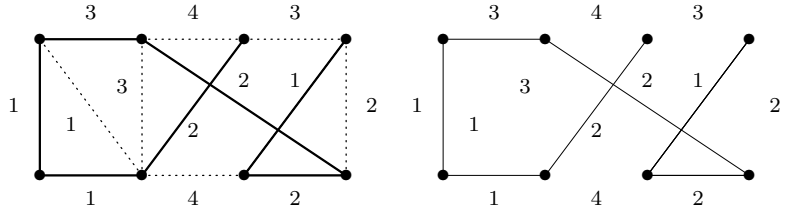
Nakonec ještě dosáhneme zbylé dva vrcholy hranami po řadě vah 2 a 1. Získali jsme tak minimální kostru velikosti $1+2+2+3+1+1+2=12$, která je v tomto případě (náhodou) cestou, na posledním obrázku vpravo. Snadno je vidět, že získaná minimální kostra není jedinečná a že jsme mohli v případech rovnosti vah volit jiné hrany a získat jinou kostru stejné velikosti. □

Příklad 2.27. *Najděme hladovým algoritmem minimální kostru ve váženém grafu z Příkladu 2.26.*

Hrany si nejprve seřadíme podle jejich vah $1,1,1,1,2,2,2,2,3,3,3,4,4$. (na pořadí mezi hranami stejné váhy nezáleží, proto jej zvolíme libovolně). Začneme s prázdnou množinou hran (budoucí) kostry. Pak hladovým postupem přidáme první dvě hrany váhy 1 vlevo dole, které nevytvoří kružnici. Třetí hrana váhy 1 vlevo s nimi už tvoří trojúhelník, a proto ji přidat nelze, je zahozena. Při znázornění průběhu používáme tlusté čáry pro vybrané hrany kostry a tečkované čáry pro zahozené hrany:



Poté přejdeme na hrany s vahou 2, z nichž lze tři postupně přidat bez vytvoření kružnice a čtvrtá (úplně vpravo) již kružnici vytvoří a je proto zahozena. Viz. obrázek vpravo. Nakonec ještě přidáme hranu nejmenší vyšší váhy 3 vlevo nahoře a zbylé hrany již zahodíme, protože všechny tvoří kružnice.



Získáme tak stejnou minimální kostru jako v Příkladě 2.26. Opět je však možno nalézt jinou z minimálních koster stejné velikosti 12, pokud mezi hranami stejné váhy v úvodním seřazení volíme jinak. vedoucími ven v každém kroku vybíráme jinak. (Například místo levé svislé hrany může kostra obsahovat přilehlou úhlopříčku stejné váhy 1.) □

Eulerovské grafy

Na závěr si ukážeme, kterak se dá zajímavě využít Eulerova věta.

Příklad 2.28. *Dokažte, že hrany každého* 6*-regulárního grafu lze zorientovat tak, aby z každého vrcholu vycházely právě* 3 *šipky.*

Myšlenka řešení je krátká, ale poměrně triková: Náš 6-regulární graf G je kreslitelný jedním uzavřeným tahem T podle Věty 2.19. Tento tah T si prostě zorientujeme – zvoleným směrem všechny jeho hrany. Jelikož T do každého vrcholu přichází třikrát a také odchází třikrát, dostaneme tři příchozí a tři odchozí šipky. □

Otázky a úlohy k řešení

(2.6.1) *Kolik nejvýše komponent může mít graf s* 15 *vrcholy, všemi stupně* 2*?*

(2.6.2) *Kolik nejvýše komponent může mít graf s* 30 *vrcholy, všemi stupně* 4*?*

(2.6.3) *Mějme graf H_5, jehož vrcholy jsou všechny dvouprvkové podmnožiny množiny $\{1, 2, 3, 4, 5\}$ a hrany spojují právě disjunktní dvojice podmnožin. (Tj. H_5 má* 10 *vrcholů.) Rozhodněte, zda je H_5 souvislý graf, a napište, kolik má H_5 hran.*

(2.6.4) *Kolik nejméně hran musí mít graf na* 12 *vrcholech, aby stupeň jeho souvislosti byl* 3 *(tj. aby se nestal nesouvislým odebráním dvou vrcholů)?*

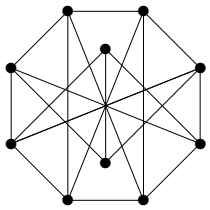
(2.6.5) *Kolik nejvíce hran může mít graf na* 10 *vrcholech,*
a) který se skládá ze tří komponent souvislosti,
b) jehož každá komponenta souvislosti má nejvíce tři vrcholy?
Tento graf také nakreslete.

*(2.6.6)** *Pokud vezmeme libovolný jednoduchý graf na* 6 *vrcholech, tak v něm najdeme trojúhelník nebo nezávislou množinu velikosti* 3*. Dokažte.*

(2.6.7) *Hamiltonovská kružnice v grafu G je takový podgraf, který je isomorfní kružnici a obsahuje všechny vrcholy G. (Neboli je to kružnice procházející všemi vrcholy G právě jednou.) Najděte a nakreslete jednoduchý neorientovaný graf, který zároveň je vrcholově* 3*-souvislý a přitom neobsahuje Hamiltonovskou kružnici.*

*(2.6.8)** *Dokažte, že pokud jsou hrany úplného grafu K_n jakkoliv zorientovány, tak buď lze jeho vrcholy uspořádat do řady tak, aby každá šipka mířila vpravo, nebo v této orientaci nalezneme orientovanou kružnici délky* 3 *jako podgraf.*

(2.6.9) *Lze tento graf nakreslit jedním tahem? A uzavřeným? (Uprostřed není vrchol, jen se tam kříží hrany.)*



(2.6.10) *Dokažte, že každý* 3*-regulární graf obsahuje* 2*-regulární podgraf.*

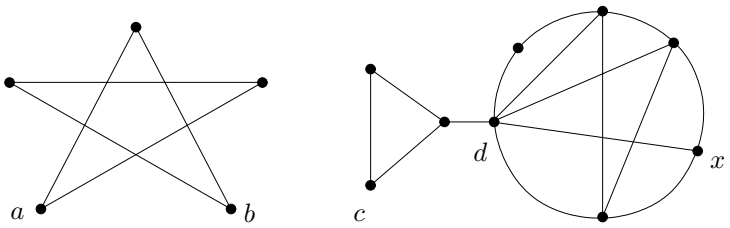
*(2.6.11)** *Dokažte, že každý* 4*-regulární graf se dá rozložit na dva* 2*-regulární grafy. (Rozklad je opakem operace množinového sjednocení, tj. sjednocením jejich množin hran i množin vrcholů dostaneme původní graf.)*

(2.6.12) *Vezměme kostky klasického domina, kde na políčcích jsou všechny polo-uspořádané dvojice čísel od* 0 *do* 6*. Pokud ze všech kostek poskládáme „hada“, dokažte, že pak první a poslední číslo jsou stejné.*

3 Vzdálenost a nejkratší cesty v grafech

Úvod

V minulé lekci jsme mluvili o souvislosti grafu, tj. o možnosti procházení z jednoho vrcholu do jiného. Někdy je prostá informace o souvislosti dostačující, ale většinou bychom rádi věděli i jak je to z jednoho vrcholu do druhého „daleko“. Proto se nyní podíváme, jak krátká či dlouhá taková procházka mezi dvěma vrcholy grafu je.



V jednodušším případě se při zjišťování grafové vzdálenosti díváme jen na minimální počet prošlých hran z vrcholu do vrcholu. Tak například v našem ilustračním obrázku je vzdálenost mezi a, b rovna 2, vzdálenost mezi a, c je rovna ∞ a vzdálenost mezi c, x je rovna 3. Navíc vidíme, že vrchol d má „centrální“ pozici v pravém grafu – každý další vrchol je od něj ve vzdálenosti nejvýše 2, kdežto vrchol c má „okrajovou“ pozici. Toto pojetí úzce souvisí s prohledáváním grafu do šířky.

Z obecného pohledu při určování grafové vzdálenosti bereme do úvahy délky jednotlivých hran v grafu, což mohou být libovolná reálná a obvykle nezáporná čísla. Problematika je taktéž téměř beze změn aplikovatelná na orientované grafy. Koncept grafové vzdálenosti a jeho základní algoritmy jsou pak základem například aplikací vyhledávajících vlaková/autobusová spojení a ve vylepšené podobě i dnes velmi populárních GPS navigací.

Cíle

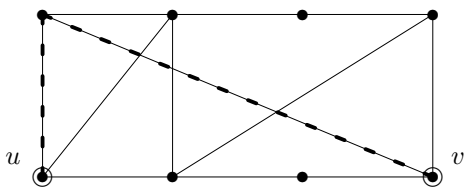
Prvním cílem této lekce je definovat vzdálenost v grafech a probrat její základní vlastnosti. Dalším je ukázat a správně pochopit (včetně důkazů) dva klasické postupy; Floyd–Warshallův a Dijkstrův algoritmus pro hledání nejkratších cest v grafu. Nakonec jsou předestřeny některé pokročilé myšlenky praktického plánování cest ve velkých grafech.

3.1 Vzdálenost v grafu

Zopakujme si, že sledem délky n v grafu G rozumíme posloupnost vrcholů a hran $v_0, e_1, v_1, e_2, v_2, \ldots, e_n, v_n$, ve které hrana e_i má koncové vrcholy v_{i-1}, v_i.

Definice 3.1. ***Vzdálenost*** $d_G(u,v)$ dvou vrcholů u, v v grafu G je dána délkou nejkratšího sledu mezi u a v v G. Pokud sled mezi u, v neexistuje, je vzdálenost $d_G(u,v) = \infty$.

`Komentář:` Neformálně řečeno, vzdálenost mezi u, v je rovna *nejmenšímu počtu hran*, které musíme projít, pokud se chceme dostat z u do v. Speciálně $d_G(u,u) = 0$. Uvědomme si, že nejkratší sled je vždy cestou (vrcholy se neopakují) – Věta 2.6.



Grafová vzdálenost se chová dosti podobně běžné vzdálenosti, jak ji známe z geometrie, což bude vidět z následujících tvrzení.

Lema 3.2. *Vzdálenost v grafech splňuje trojúhelníkovou nerovnost:*

$$\forall u, v, w \in V(G) \;:\; d_G(u, v) + d_G(v, w) \geq d_G(u, w)\,.$$

`Důkaz`. Nerovnost snadno plyne ze zřejmého pozorování, že na sled délky $d_G(u, v)$ mezi u, v lze navázat sled délky $d_G(v, w)$ mezi v, w, čímž vznikne sled délky $d_G(u, v) + d_G(v, w)$ mezi u, w. Skutečná vzdálenost mezi u, w pak už může být jen menší. □

`Fakt`: V neorientovaném grafu je vzdálenost symetrická, tj. $d_G(u, v) = d_G(v, u)$.

Některé další pojmy

Definice 3.3. Mějme graf G. Definujeme (vzhledem k G) následující pojmy a značení:

- *Excentricita* vrcholu $\mathrm{exc}(v)$ je nejdelší vzdálenost z v do jiného vrcholu grafu; $\mathrm{exc}(v) = \max_{x \in V(G)} d_G(v, x)$.
- *Průměr* $\mathrm{diam}(G)$ grafu G je největší excentricita jeho vrcholů, naopak *poloměr* $\mathrm{rad}(G)$ grafu G je nejmenší excentricita jeho vrcholů.
- *Centrem* grafu je množina vrcholů $U \subseteq V(G)$ takových, jejichž excentricita je rovna poloměru $\mathrm{rad}(G)$.
- *Steinerova vzdálenost* mezi vrcholy libovolné podmnožiny $W \subseteq V(G)$ je rovna minimálnímu počtu hran souvislého podgrafu v G obsahujícího všechny vrcholy W.

`Komentář:` Prostudujte si výše uvedené pojmy na různých příkladech (obrázcích) grafů. Zamyslete se třeba nad příklady grafů, ve kterých tvoří centrum všechny jejich vrcholy. Promyslete si také, jak by se naše pojmy související se vzdáleností v grafech zobecnily na Steinerovu vzdálenost.

Základní zjištění vzdálenosti

Věta 3.4. *Nechť u, v, w jsou vrcholy souvislého grafu G takové, že $d_G(u, v) < d_G(u, w)$. Pak při algoritmu procházení grafu G do šířky z vrcholu u je vrchol v nalezen dříve než vrchol w.*

`Důkaz`. Postupujeme indukcí podle vzdálenosti $d_G(u, v)$: Pro $d_G(u, v) = 0$, tj. $u = v$ je tvrzení jasné – vrchol u jako počátek prohledávání byl nalezen první. Proto nechť $d_G(u, v) = d > 0$ a označme v' souseda vrcholu v bližšího k u, tedy $d_G(u, v') = d - 1$. Obdobně uvažme libovolného souseda w' vrcholu w. Pak

$$d_G(u, w') \geq d_G(u, w) - 1 > d_G(u, v) - 1 = d_G(u, v')\,,$$

a tudíž vrchol v' byl nalezen v prohledávání do šířky dříve než vrchol w' podle indukčního předpokladu. To znamená, že v' se dostal do fronty úschovny dříve než w'. Proto sousedé v' (mezi nimiž je v, ale ne w neboť $v'w$ není hranou) jsou při pokračujícím prohledávání také nalezeni dříve než w coby soused w'. □

Důsledek 3.5. *Algoritmus procházení grafu do šířky lze použít pro výpočet grafové vzdálenosti z daného vrcholu u.*

Toto je poměrně jednoduchá aplikace, kdy počátečnímu vrcholu u přiřadíme vzdálenost 0, a pak vždy každému dalšímu nalezenému vrcholu v přiřadíme vzdálenost o 1 větší než byla vzdálenost vrcholu, ze kterého jsme jej právě nalezli. Podle Věty 3.4 se totiž nelze později dostat k v z vrcholu bližšího k počátku u.

`Komentář:` Důsledek 3.5 „funguje“ jen pro vzdálenosti s jednotkovou délkou všech hran. My si dále ukážeme obecnější Dijkstrův algoritmus, který obdobným postupem počítá nejkratší vzdálenost při libovolně kladně ohodnocených délkách hran.

Otázky a úlohy k řešení

(3.1.1) *Jaká je největší vzdálenost dvou různých vrcholů v úplném grafu?*

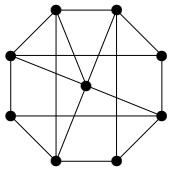
(3.1.2) *Jaká je největší vzdálenost dvou různých vrcholů v úplném bipartitním grafu $K_{33,44}$?*

(3.1.3) *Jaká je největší vzdálenost dvou různých vrcholů na kružnici C_{11}?*

(3.1.4) *Jaká je největší Steinerova vzdálenost tří různých vrcholů na kružnici C_{11}?*

(3.1.5) *Jaká je Steinerova vzdálenost všech vrcholů souvislého grafu s n vrcholy?*

(3.1.6) *Jaká je největší vzdálenost mezi dvěma vrcholy v následujícím grafu?*



(3.1.7) *Určete poloměr a centrum grafu z předchozího obrázku.*

(3.1.8) *Umíte nalézt graf na 8 vrcholech se všemi stupni 3 a průměrem 2?*

*(3.1.9)** *Zjistěte vztah mezi průměrem a poloměrem grafu a dokažte jej.*

3.2 Vážená (ohodnocená) vzdálenost

V dalším textu se již budeme věnovat cestám v grafech s obecně „dlouhými“ hranami, kde délka je dána ohodnocením hran reálnými čísly. Látku tohoto oddílu a dalších lze podat stejně dobře s orientovanými i s neorientovanými grafy (přičemž definice i výsledky obvykle platí v obou pohledech stejně). Pro větší obecnost proto podáme pohled implicitně založený na orientovaných grafech, avšak často bez explicitního zdůraznění.

`Definice`: Vážený graf je dvojice grafu G spolu s ohodnocením w hran reálnými čísly $w : E(G) \to \mathbb{R}$. *Kladně vážený graf* (G, w) je takový, že $w(e) > 0$ pro všechny hrany e.

Definice 3.6. ***(Vážená vzdálenost)*** Mějme vážený orientovaný graf (G, w). Váženou délkou (orientovaného) sledu $S = (v_0, e_1, v_1, e_2, v_2, \dots, e_n, v_n)$ v G myslíme součet

$$d^w_G(S) = w(e_1) + w(e_2) + \cdots + w(e_n)\,.$$

Váženou vzdáleností v (G, w) mezi dvěma vrcholy u, v pak myslíme

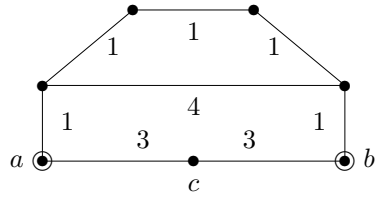
$$d^w_G(u, v) = \min\{d^w_G(S) : S \text{ je (orientovaný) sled od } u \text{ do } v\}\,.$$

Důležitým faktem je, že definice a výsledky vztahující se k vážené vzdálenosti na orientovaných grafech obvykle snadno převedeme na neorientované grafy pouhou korespondencí každé neorientované hrany na dvojici opačně zorientovaných hran stejné délky. Základní grafová vzdálenost z Oddílu 3.1 je pak speciálním případem se všemi délkami rovnými 1. Obdobně Větě 2.6 snadno dokážeme, že nejkratší sled v kladně váženém orientovaném grafu je vždy cestou (orientovanou), tj. nedochází k opakování vrcholů – viz Tvrzení 3.9. Dále platí:

Lema 3.7. *Vážená vzdálenost v orientovaných grafech, pokud je dobře definována, splňuje trojúhelníkovou nerovnost;*

$$\forall u, v, w \in V(G) \;:\; d_G(u, v) + d_G(v, w) \geq d_G(u, w)\,.$$

Příklad 3.8. *Podívejme se na následující ohodnocený graf (čísla u hran udávají jejich váhy–délky.)*

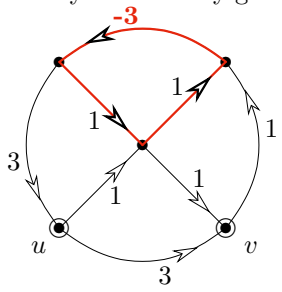


Vzdálenost mezi vrcholy a, c je 3, stejně tak mezi b, c. Co ale mezi a, b? Je jejich vzdálenost 6? Kdepak, vzdálenost a, b je 5, její cesta vede po „horních“ vrcholech. Povšimněte si, že tento příklad také zároveň ukazuje, že postup *prohledáváním do šířky není korektní* pro hledání vzdáleností ve váženém grafu. □

Záporné délky hran

Při vyslovení pojmu „délka“ si obvykle představíme pouze kladnou či nulovou hodnotu, ale ne záporné číslo. Přesto naše definice připouští záporné váhy hran grafu. Co to znamená v kontextu grafové vzdálenosti?

`Komentář:` Uvažme následující vážený orientovaný graf.



Jaká je v něm vzdálenost z vrcholu u do v? Není to ani 3, ani 2, ale přechodem přes horní hranu váhy -3 lze třeba získat sled délky 1. To ale není vše – můžeme si všimnout vyznačené orientované kružnice o třech hranách, jejíž vážená délka je rovna -1. Pochopitelně to není hezké a do důsledků to znamená, že Definice 3.6 není pro náš graf korektní (vzdálenost se blíží k $-\infty$ opakováním vyznačené kružnice).

Uvedený příklad je možno brát jako dobrý důvod k úplnému vyloučení záporných délek hran v našem výkladu, ale na druhou stranu jsou aplikace, ve kterých záporné délky mají své přirozené místo a lze s nimi pracovat. Klíčem ke zvládnutí záporných hran ve speciálních případech je následující poznatek.

`Definice`: Orientovanou kružnici záporné vážené délky nazveme *záporným cyklem*.

Tvrzení 3.9. *Mějme vážený orientovaný graf, který neobsahuje žádný záporný cyklus. Pak pro každou dvojici vrcholů u, v platí, že*

a) vážená vzdálenost z u do v je vždy korektně definována,

b) vážené vzdálenosti z u do v lze nabýt orientovanou cestou z u do v.

`Důkaz`: Oba závěry plynou zároveň z následujícího tvrzení (b): Pro každý orientovaný sled S v daném váženém grafu (G, w) existuje cesta P se stejnými konci taková, že $d^w_G(P) \leq d^w_G(S)$. Z toho (a) plyne proto, že možných orientovaných cest v grafu je konečně mnoho a tudíž lze určit nejkratší.

Důkaz je pomocí jednoduché minimalizace; uvažujme sled S' nejmenší délky takový, že $d^w_G(S') \leq d^w_G(S)$. Pokud S neobsahuje opakované vrcholy, je hledanou cestou $P = S'$. V opačném případě zvolíme nejkratší podsled $S_1 \subseteq S'$ obsahující některý vrchol dvakrát – S_1 musí být cyklem (nebo lze vybrat kratší) nezáporné vážené délky $d^w_G(S_1) \geq 0$ podle předpokladu neexistence záporného cyklu. Nechť S'' je podsled získaný vypuštěním S_1 z S'. Pak $d^w_G(S') = d^w_G(S_1) + d^w_G(S'')$ a $d^w_G(S'') \leq d^w_G(S')$, což je spor s naší volbou S'. □

Bellman–Fordův algorithmus

V případě existence záporných hran musíme být při algoritmickém výpočtu vzdálenosti opatrní a schopni také detekovat nekorektní vstupy obsahující záporné cykly. Tradičním řešením je následující známý algoritmus.

Algoritmus 3.10. ***Výpočet vzdálenosti s detekcí záporného cyklu.***
V daném váženém orientovaném grafu (G, w) a s daným počátkem $u_0 \in V(G)$ je úkolem vypočítat vážené vzdálenosti $dist[u_0, v] = d^w_G(u_0, v)$ z u_0 do ostatních vrcholů $v \in V(G)$.

```
input 'Orientovaný vážený graf (G,w).'
inicializovat dist[u0,v] ← ∞,  pro všechny v ∈ V(G);
dist[u0,u0] ← 0;
repeat |V(G)| − 1 krát {
    foreach (e=uv ∈ E(G)) {
        dist[u0,v] ← min(dist[u0,v], dist[u0,u] + w(e));    (*)
    }
}
foreach (e=uv ∈ E(G)) {
    if (dist[u0,v]>dist[u0,u]+w(e))
        output 'Chyba; existuje (někde) záporný cyklus v (G,w).'
}
output 'Vzdálenosti z u0 jsou uloženy v dist[u0,·].'
```

(Snadno lze také na řádku (*) ukládat předchůdce vrcholů a získat tak na konci i příslušné nejkratší cesty...)

`Komentář`: Algoritmus 3.10 dokáže sice detekovat existenci záporného cyklu, který je dosažitelný z počátku u_0, nelze však pomocí něj snadno záporný cyklus najít (to je kupodivu výrazně obtížnější úloha a zde se jí nezabýváme).

`Důkaz`: Důkaz vedeme posloupností následujících tvrzení:

- V každém kroku Algoritmu 3.10 platí $dist[u_0, v] \geq d^w_G(u_0, v)$. Toto platí triválně po inicializaci a poté indukcí podle počtu kroků, neboť jedinou úpravou hodnoty této proměnné je '`dist[u0,v] ← min(dist[u0,v], dist[u0,u] + w(e))`', což odpovídá existujícímu sledu přicházejícímu do v přes e.
- Podle Tvrzení 3.9 se vzdálenost $d^w_G(u_0, v)$ nabývá nějakou orientovanou cestou v G. Nechť $V_i \subseteq V(G)$ je množinou těch vrcholů v, pro které se vzdálenosti $d^w_G(u_0, v)$ nabývá orientovanou cestou z u_0 do v používající $\leq i$ hran. Pak lze snadno indukcí dokázat, že pro každé k a $v \in V_k$ nastane `dist[u0,v]` $\leq d^w_G(u_0, v)$ nejpozději po k krocích algoritmu.
- Pokud v (G, w) existuje záporný cyklus, který je dosažitelný z počátku u_0, některá vzdálenost z u_0 se bude každou iterací zmenšovat a tudíž závěrečný test existenci záporného cyklu musí odhalit. Naopak je však nutno dokázat, že pokud závěrečný test existenci záporného cyklu neodhalí, všechny cykly v (G, w) jsou nezáporné. Teprve tímto poznatkem, Tvrzením 3.9 b) a předchozím bodem (dohromady) dokážeme, že `dist[u0,v]` udává korektní vzdálenosti v (G, w).
  Vezmeme tudíž jakýkoliv cyklus $C \subseteq G$ a tvrdíme, že pokud '`dist[u0,v]` $\not>$ `dist[u0,u]+w(e)`' pro všechny hrany $e = uv \in E(C)$, tak platí $d^w_G(C) \geq 0$. Předpoklad '`dist[u0,v]` $\not>$ `dist[u0,u]+w(e)`' si přepíšeme (matematickým zápisem) jako nerovnost $dist(u_0, v) - dist(u_0, u) \leq w(e)$ a tyto nerovnosti sečteme přes všechny $e = uv \in E(C)$. Po vyrušení členů vyjde $0 \leq \sum_{e \in E(C)} w(e)$. □

Otázky a úlohy k řešení

(3.2.1) *Jaká je nejdelší vzdálenost mezi dvěma vrcholy v grafu z Příkladu 3.8?*

(3.2.2) *O kterém vrcholu v grafu z Příkladu 3.8 se dá říci, že má „centrální pozici“, tj. že je z něj do všech ostatních vrcholů nejblíže? Jaká je z něj největší vzdálenost do ostatních vrcholů?*

(3.2.3) *Ve výše zakresleném orientovaném grafu se záporným cyklem rozhodněte, zda existuje dvojice vrcholů pro něž je jejich vzdálenost korektně definována.*

(3.2.4) *Dokážete najít jiný orientovaný vážený graf se záporným cyklem, ve kterém by vzdálenost pro některé dvojice vrcholů byla korektně definována?*

3.3 Nejkratší cesty při kladných délkách

Pro nalezení nejkratší (vážené) cesty mezi dvěma vrcholy kladně váženého grafu se, na rozdíl od poněkud upracovaného Algoritmu 3.10, používá neméně tradiční a přitom rychlý *Dijkstrův algoritmus*, či jeho vhodná další vylepšení (viz Oddílu 3.4).

Dijkstrův algoritmus

Tento algoritmus je variantou procházení grafu (skoro jako do šířky), kdy pro každý nalezený vrchol ještě máme *proměnnou udávající vzdálenost* – délku nejkratšího sledu (od počátku), kterým jsme se do tohoto vrcholu zatím dostali. Z úschovny nalezených vrcholů vždy vybíráme vrchol s nejmenší vzdáleností (mezi uschovanými vrcholy) – neformálně řečeno, do takového vrcholu se už lépe dostat nemůžeme, protože všechny jiné cesty by byly dle výběru delší. Na konci zpracování pak proměnné vzdálenosti udávají správně nejkratší vzdálenosti z počátečního vrcholu do ostatních.

Algoritmus 3.11. ***Výpočet vzdáleností z jednoho počátku.***
V daném kladně váženém orientovaném grafu (G, w) a s daným počátkem $u_0 \in V(G)$ je úkolem určit vážené vzdálenosti $dist[u_0, v] = d^w_G(u_0, v)$ z u_0 do vrcholů $v \in V(G)$.

```
input ‘Orientovaný kladně vážený graf (G,w).’
inicializovat dist[u0,v] ← ∞,  pro všechny v ∈ V(G);
dist[u0,u0] ← 0;
U ← {u0};
while (U ≠ ∅) {
    vybereme u∈U minimalizující dist[u0,u];
    foreach (edge f vycházející z u) {
        v ← opačný vrchol hrany ‘f = uv’;
        if (dist[u0,u]+w(uv)<dist[u0,v]) {
            U ← U∪{v};
            predec[v] ← u;
            dist[u0,v] ← dist[u0,u]+w(uv);
        }
    }
    U ← U\{u};
}
output ’vzdálenosti uloženy v dist[.], předchůdci pro nejkratší cesty v predec[]’;
```

Celkový počet kroků potřebný v Algoritmu 3.11 k nalezení nejkratší cesty z `u` do `v` může v základní „hloupé“ implementaci být zhruba n^2, kde n je počet vrcholů grafu. Na druhou stranu, při lepší implementaci grafu seznamem sousedů a použití vhodné úschovny nezpracovaných vrcholů (implementované *haldou* s nalezenou vzdáleností jako

klíčem) lze dosáhnout i mnohem rychlejšího běhu tohoto algoritmu na řídkých grafech – času téměř úměrného počtu hran grafu, přesněji $O\big(|E(G)| + n \log n\big)$.

Správnost Algoritmu 3.11 dokážeme snadno indukcí pomocí následujícího tvrzení.

Věta 3.12. *V každé iteraci Algoritmu 3.11 (počínaje stavem po prvním průchodu cyklem* `while()` *) proměnná* `dist[i]` *udává nejkratší vzdálenost z vrcholu* `u` *do vrcholu* `i` *při cestě pouze po vnitřních vrcholech x, které byly dosud odebrány z U (tj. zpracované).*

`Důkaz`: Stručně matematickou indukcí:

- V prvním kroku algoritmu je jako vrchol ke zpracování vybrán první `u`=u_0 a potom jsou jeho sousedům upraveny vzdálenosti od u_0 podle délek hran vycházejících z u_0.
- V každém dalším kroku je vybrán jako vrchol `u` ke zpracování ten, který má ze všech nezpracovaných vrcholů nejkratší nalezenou vzdálenost od počátku u_0. To znamená, že žádná kratší cesta z u_0 do `u` nevede, neboť každá „oklika“ přes jiné nezpracované vrcholy musí být delší dle výběru `u` a indukčního předpokladu. (V tomto bodě právě potřebujeme nezápornost ohodnocení w.)
  Naopak každá nová nejkratší cesta z u_0 do nezpracovaného vrcholu `v` procházející přes `u` musí mít `u` coby předposlední vrchol, tj. poslední hranu `uv`, a proto je upravená hodnota `dist[`u_0`,v]` správná i po přidání `u` mezi zpracované vrcholy.

□

Závěrem zbývá ověřit, že nalezená nejkratší cesta z `u` do `v` je pozpátku uložená v poli `predec[]`, tj. předposlední vrchol před `v` je `predec[v]`, předtím `predec[predec[v]]`, atd...

Všechny nejkratší cesty

Jiný přístup je možné volit, pokud je cílem spočítat vzdálenosti mezi všemi dvojicemi vrcholů. Tento přístup bude fungovat dobře i za přítomnosti záporných hran, pokud v orientovaném grafu nejsou záporné cykly.

`Definice`: Metrikou grafu myslíme soubor vzdáleností mezi všemi dvojicemi vrcholů grafu. Jinak řečeno, *metrikou grafu G* je matice (dvourozměrné pole) `d[,]`, ve kterém prvek `d[i,j]` udává vzdálenost mezi vrcholy i a j.

Metoda 3.13. ***Dynamický výpočet metriky skládáním cest (Floyd–Warshall)*** *ve váženém orientovaném grafu (G, w) bez záporných cyklů.*

- Uspořádejme si vrcholy grafu libovolně do posloupnosti $V(G) = \{v_0, v_1, \ldots, v_{n-1}\}$. Na počátku nechť `d[i,j]` udává délku hrany v_iv_j (která může být orientovaná), nebo ∞ pokud hrana z v_i do v_j není.
- Po kroku $t \geq 0$ nechť platí, že `d[i,j]` udává délku nejkratšího sledu z v_i do v_j, který užívá pouze vnitřní vrcholy z množiny $\{v_0, v_1, \ldots, v_{t-1}\}$ (prázdné v $t = 0$).
- Při přechodu z kroku t na následující krok $t + 1$ upravujeme vzdálenost pro každou dvojici vrcholů v_i, v_j – jsou vždy pouhé dvě možnosti:
  - Buď je sled délky `d[i,j]` z předchozího kroku t stále nejlepší (tj. nově povolený vrchol v_t nám nepomůže),
  - nebo sled vylepšíme spojením přes nově povolený vrchol v_t, čímž získáme menší vzdálenost `d[i,t]+d[t,j]` $\rightarrow$ `d[i,j]`. (Nakreslete si obrázek.)

Věta 3.14. *Metoda 3.13 po $n = |V(G)|$ iteracích v poli* `d[i,j]` *správně určí délku nejkratší (příp. orientované) cesty z v_i do v_j.*

`Důkaz` povedeme matematickou indukcí podle t. Báze je snadná – ve fázi $t = 0$ udává `d[i,j]` vzdálenost mezi v_i a v_j po cestách, které nemají vnitřní vrcholy (tj. pouze hranu).

Přejdeme-li na fázi $t + 1$, musíme určit nejkratší cestu P mezi v_i a v_j takovou, že P používá (mimo v_i, v_j) pouze vrcholy $\{v_0, v_1, \ldots, v_{t-1}\}$. Tuto nejkratší cestu P si hypoteticky představme: Pokud $v_t \notin V(P)$, pak `d[i,j]` již udává správnou vzdálenost. Jinak $v_t \in V(P)$ a označíme P_1 podcestu v P od počátku v_i do v_t a obdobně P_2 podcestu od v_t do konce v_j. Podle indukčního předpokladu je pak délka P rovna `d[i,t]+d[t,j]`.

Po provedení iterace (fáze) $t = n - 1$ jsme hotovi. □

Floyd–Warshallův algoritmus

`Poznámka`: V praktické implementaci pro symbol ∞ můžeme použít velkou konstantu, třeba `MAX_INT/2`. (Nelze použít přímo `MAX_INT`, neboť by pak došlo k aritmetickému přetečení.) Implementace je pak z dosud uvedených algoritmů pro nejkratší cesty úplně nejjednodušší.

Algoritmus 3.15. ***Výpočet vzdáleností mezi všemi dvojicemi vrcholů.***
V daném váženém orientovaném grafu (G, w) je úkolem vypočítat všechny vzdálenosti $dist[u, v] = d^w_G(u, v)$.

```
input 'Orientovaný vážený graf (G,w).'
inicializovat dist[u,v] ← ∞,  pro u,v ∈ V(G);
foreach (uv ∈ E(G))  dist[u,v] ← w(uv);
foreach (t ∈ V(G)) {
    foreach (u,v ∈ V(G)) {
        dist[u,v] ← min(dist[u,v],dist[u,t]+dist[t,v]);
    }
}
output 'Metrika grafu (G,w) je uložena v d[,]';
```

`Komentář:` Algoritmus 3.15 je implementačně velmi jednoduchý (vždyť se celý jeho kód vešel na 5 přehledných řádků) a provede zhruba N^3 kroků pro výpočet celé metriky. Jeho jedinou (ale velkou) nevýhodou je, že vzdálenosti mezi všemi dvojicemi vrcholů je třeba *počítat najednou*. V praktických situacích však obvykle požadujeme zjištění vzdálenosti mezi jedinou dvojicí vrcholů, a pak celý zbytek výpočtu je k ničemu...

`Komentář:` Vysvětleme si podrobněji srovnání obou předchozích algoritmů. Floyd–Warshallův Algoritmus 3.15 pro výpočet metriky je implementačně jednodušší a přímočařejší a hodí se tam, kde potřebujeme spočítat všechny dvojice vzdáleností v grafu najednou. Velkou nevýhodou však je, že vždy musíme provést N^3 kroků celého výpočtu, i když počítáme vzdálenost jen dvou blízkých vrcholů. Dijkstrův Algoritmus 3.11 na druhou stranu počítá mnohem rychleji, pokud nás zajímá jen vzdálenost z jednoho vrcholu (zhruba N^2 nebo i méně kroků).

Dokonce Dijkstrův algoritmus běží ještě rychleji, pokud nás zajímá jen vzdálenost dvou „blízkých“ vrcholů – tehdy totiž můžeme prohledávání dokončit jen na malé části celého grafu. Ještě lépe se z tohoto pohledu chová níže popsaný *algoritmus nazývaný* A^*, který použitím vhodného potenciálu „směřuje“ celé prohledávání grafu ke správnému cíli a je skvěle použitelný ve všech situacích, kdy pojem „směr k cíli“ má matematický význam. To je například při navigování v mapě.

Otázky a úlohy k řešení

(3.3.1) *Co konkrétně selže v Dijkstrově algoritmu při vstupu grafu se záporně ohodnocenou hranou?*

(3.3.2) *Bude Dijkstrův algoritmus pracovat správně, pokud sice graf obsahuje hrany záporné délky, ale každý jeho cyklus má kladnou délku?*

(3.3.3) *Vypočtěte si dle Algoritmu 3.13 metriku kružnice C_4.*

(3.3.4) *Zamyslete se nad implementací Algoritmu 3.13 v místě výpočtu* `d[i,t]+d[t,j]` *– není zde problém, že tyto hodnoty někdy jsou už nynější iterací změněné (kdežto jindy ještě nejsou)?*

(3.3.5) *Co v Algoritmu 3.13 přesně selže, pokud graf obsahuje záporné cykly?*

3.4 Pokročilé plánování cest

Třebaže je Dijkstrův algoritmus velmi rychlý a „téměř optimální“ mezi všemi způsoby hledání nejkratší cesty v nezáporně váženém grafu, stále zůstává příliš pomalým pro praktické nasazení třeba v přenosných navigačních zařízeních, jež obsahují mapová data v rozsahu desítek až stovek miliónů hran grafu. Co však může být uděláno lépe?

Odpovědí je vhodné předzpracování grafu: Lze dobře očekávat, že samotný graf je relativně stabilní a pro jeho předzpracování je k dispozici dostatek času na výkonných strojích. V tomto místě však vyvstává sekundární problém, kde uložit výsledky předzpracování? Určitě není reálné ukládat (např.) všechny nejkratší cesty mezi všemi dvojicemi vrcholů… Dva z nejjednodušších úsporných přístupů k předzpracování grafu pro rychlé plánováni cest si nyní zběžně ukážeme.

Algoritmus A^*

- Je pouhou reimplementací Dijkstrova algoritmu (hledajícího nejkratší cestu z vrcholu u do vrcholu v v orientovaném grafu) s „vhodně“ upravenými délkami hran.
- Nechť „*potenciál*“ $p_v(x)$ udává libovolný dolní odhad vzdálenosti z vrcholu x do cíle v. Každá (orientovaná!) hrana xy grafu (G, w) dostane nové délkové ohodnocení $w'(xy) = w(xy) + p_v(y) - p_v(x)$. Potenciál p_v je *přípustný*, pokud všechna upravená ohodnocení jsou nezáporná, neboli $w(xy) \geq p_v(x) - p_v(y)$.
- Upravená délka libovolného sledu S z u do v pak je $d_G^{w'}(S) = d_G^w(S) + p_v(v) - p_v(u)$, což je konstantní rozdíl oproti původní délce S. Takže S je optimální pro původní délkové ohodnocení w, právě když je optimální pro nové w'.

`Komentář:` Čtenář nechť si povšimne, že algoritmus A^* každý graf implicitně zorientuje – upravené délkové ohodnocení totiž nebude symetrické $w'(xy) \neq w'(yx)$.

Pro (časté) použití při navigaci v mapě může potenciál $p_v(x)$ udávat přímou (Euklidovskou) vzdálenost z bodu x do bodu v. Tento potenciál je vždy přípustný podle trojúhelníkové nerovnosti. Dijkstrův algoritmus pro délkové ohodnocení w' takto upravené potenciálem přímé vzdálenosti do v pak bude „silně preferovat“ hrany vedoucí ve směru k cíli v; délka takových hran bude téměř nulová, kdežto délka hran vedoucích od cíle se téměř zdvojnásobí. Výsledkem bude výrazně menší počet prohledávaných vrcholů grafu (do nalezení cesty do cíle v) a také menší potřebná velikost úschovny vrcholů.

Myšlenka „dosahu“ (reach)

- Tento přístup těží z přirozeného poznatku, že valná většina hran reálné cestní sítě je pro globální plánování cest vpodstatě „bezvýznamná“.

`Definice`: Nechť $S_{u,v}$ značí ve váženém grafu G optimální sled z u do v. Nechť dále pro $e \in E(S_{u,v})$ značí $prefix(S_{u,v}, e)$, $suffix(S_{u,v}, e)$ celý úsek sledu $S_{u,v}$ před (za) jeho

hranou e. *Dosah hrany* $e \in E(G)$ je dán vztahem

$$reach_G(e) = \max\big\{\, \min\big(\, d^w_G(prefix(S_{u,v}, e)), d^w_G(suffix(S_{u,v}, e))\big) : \\ \forall u, v \in V(G) \wedge e \in E(S_{u,v})\big\}.$$

`Komentář:` Dosah hrany e přesně matematicky kvantifikuje „(bez)významnost“ e pro plánování cest; čím menší je $reach_G(e)$, tím blíže počátku nebo cíle optimální cesty musí hrana e být. Neboli, prakticky, hrany s malým $reach_G(e)$ lze pro většinu dotazů na optimální cesty ignorovat...

Předpočítaného parametru dosahu hran lze přirozeně využít v *obousměrné variantě* Dijkstrova algoritmu (včetně A^*) ke vskutku radikální redukci počtu prohledávaných vrcholů při hledání nejkratší cesty:

- V řádku “`foreach (k` *soused* `m)`” (viz Algoritmus 3.11) prostě akceptujeme pouze ty sousedy `k` pro které platí $reach_G($`mk`$) \geq$ `dist[m]`.

Podle definice dosahu pak i s touto restrikcí v obousměrném prohledávání vždy najdeme optimální cestu z u do v.

Otázky a úlohy k řešení

(3.4.1) *Dokažte matematicky nezápornost $w'(xy)$ v algoritmu A^*, pokud $p_v(x)$ udává přímou vzdálenost z bodu x do v.*

(3.4.2) *Jaký problém nastane, pokud v algoritmu A^* bude hodnota $p_v(x)$ vyšší, než vzdálenost z vrcholu x do cíle v? (p_v nebude dolním odhadem vzdálenosti do v)*

Rozšiřující studium

Vzdálenostem v grafech se teoreticky věnuje [5, Oddíl 4.2]. Floyd–Warshallův algoritmus `http://en.wikipedia.org/wiki/Floyd-Warshall_algorithm` *je dobře známou ukázkou paradigmatu dynamického programování a jeho modifikace mohou počítat nejen nejkratší cesty či tranzitivní uzávěry, ale třeba i odvozovat regulární výraz (viz klasická učebnice Hopcroft–Ullman: Introduction to automata theory, languages, and computation, Addison-Wesley, 1979). Dijkstrův algoritmus je navíc velmi oblíbeným tématem mnoha učebnic programování, a proto jej není třeba nijak složitě hledat, na internetu například* `http://en.wikipedia.org/wiki/Dijkstra's_algorithm`.

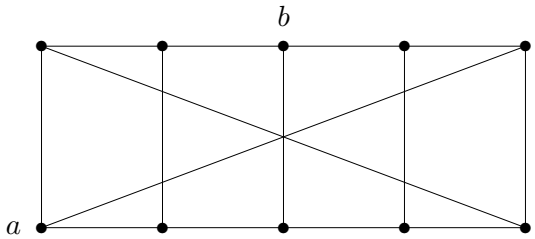
Problematika pokročilého plánování cest je v dnešní době mnohem bohatší, než jak jsme stačili naznačit v krátkém Oddíle 3.4, a čtenáře odkazujeme především na internetové zdroje. V prvé řadě je to `http://en.wikipedia.org/wiki/A*_search_algorithm` *a obousměrné varianty na* `http://en.wikipedia.org/wiki/Bidirectional_search`. *Z dalších přístupů lze kromě zmíněného klíčového slova „reach“ hledat i přístupy založené na „separators“, „highway / contraction hierarchies“, „landmarks“, „transit nodes“, či různé kombinované a heuristické přístupy.*

Naše krátké pojednání se z prostorových důvodů vůbec nevěnuje problematice výpočtu nejkratších cest za přítomnosti negativních hran. Tento problém lze řešit třeba Bellman–Fordovým algoritmem, `http://en.wikipedia.org/wiki/Bellman-Ford_algorithm`, *podobným Dijkstrovu, ale pomalejším. Sofistikovanější kombinované řešení nabízí Johnsonův algoritmus* `http://en.wikipedia.org/wiki/Johnson's_algorithm`. *Problematice výpočtu vzdálenosti za přítomnosti záporných hran se na MU do větší hloubky věnuje sesterský předmět MA015 Grafové algoritmy.*

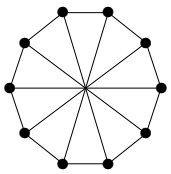
3.5 Cvičení: O cestách a grafových vzdálenostech

Ve cvičení se opět zaměříme hlavně na bližší demonstraci zavedených konceptů a poté navážeme několika cvičnými úlohami k samostatnému řešení.

Příklad 3.16. *Určete průměr níže zakresleného grafu na 10 vrcholech a odpovězte, kolik různých dvojic vrcholů v něm má vzdálenost rovnou průměru.*

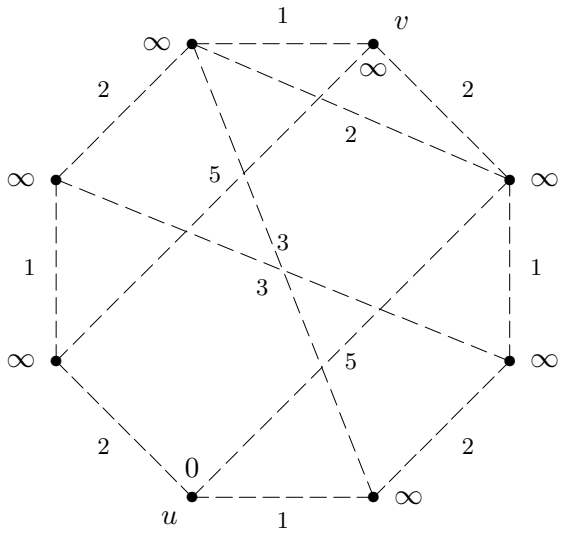


Snadným pohledem zjistíme, že například vrcholy a, b vyznačené v obrázku mají vzdálenost 3. Poté můžeme rozebráním pár možností argumentovat, že větší vzdálenost se v grafu nevyskutuje, tudíž jeho průměr je 3. Obdobně můžeme rozebráním možností spočítat, že vzdálenost 3 se nabývá právě mezi 10 dvojicemi vrcholů, ale také se můžeme snadno přepočítat.

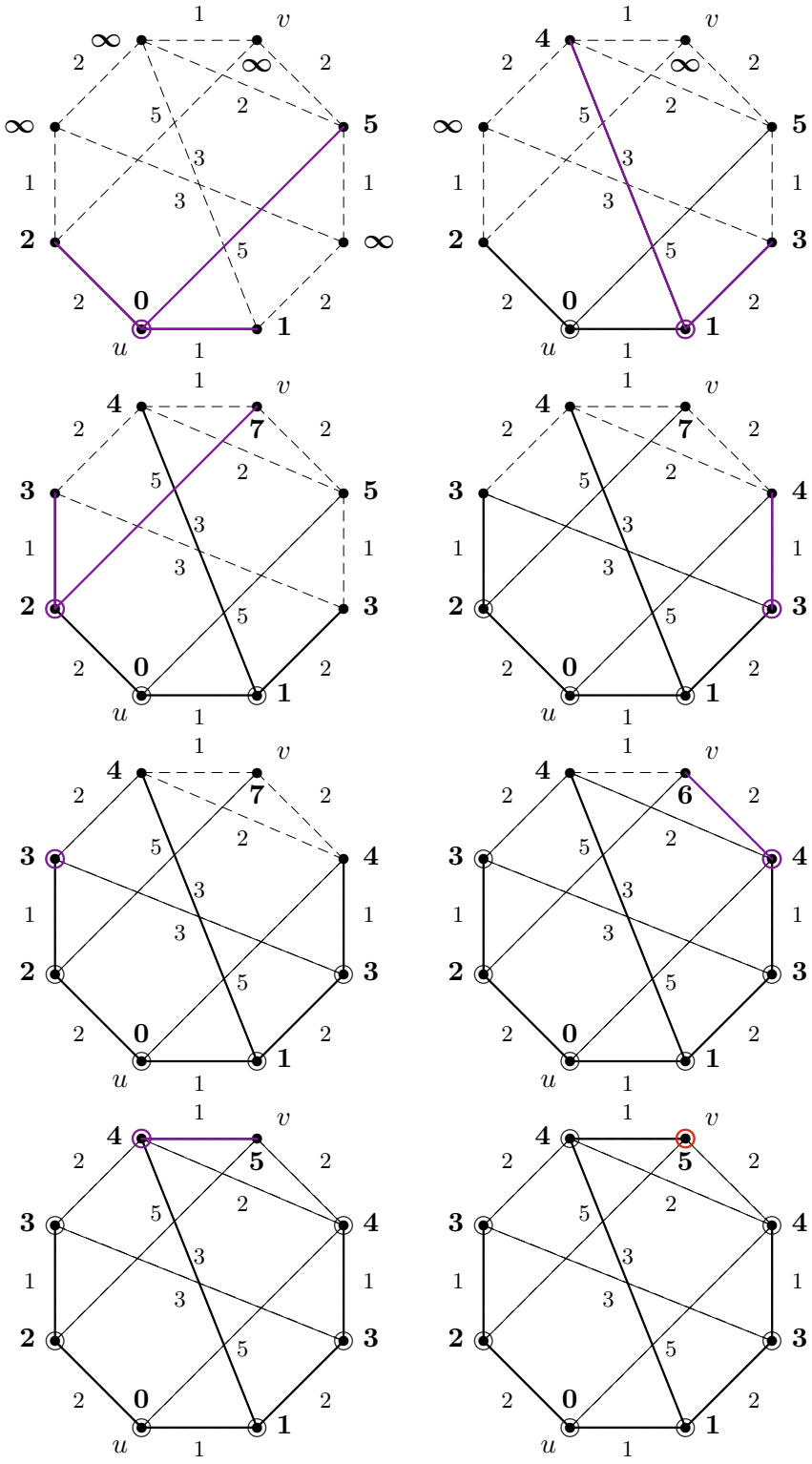


Správně doporučeným řešením je, jak už bylo zdůrazňováno několikrát, nalézt vhodnější obrázek našeho grafu. V tomto případě zde (výše). Nyní je okamžitě vidět, že všechny vrcholy jsou si navzájem symetrické, takže prostým jedním prohledáním do hloubky zjistíme nejvyšší vzdálenost 3 a právě dva vrcholy v této vzdálenosti z libovolného počátečního vrcholu. Matematicky rigorózně tak potvrdíme výše získané poznatky. □

Příklad 3.17. *Ukázka běhu Dijkstrova Algoritmu* **??** *pro nalezení nejkratší cesty mezi vrcholy u, v v následujícím grafu.*



U tohoto algoritmu se vlastně jedná a specifickou variantu procházení grafu. Proto použijeme stejné obrázkové značení jako v Příkladě 2.21 a navíc pro každý vrchol zakreslíme jeho okamžitou dočasnou vzdálenost `vzdal[x]`. (Tj. zpracované vrcholy budeme značit kroužkem a hrany plnou čarou.) Jednotlivé kroky následují:



Z průběhu algoritmu vidíme, že již ve třetím kroku jsme určili dočasnou vzdálenost z U do v na 7, ale ta nebyla nejkratší možná. Nakonec po proběhnutí všech kroků

algoritmu vzdálenost u, v poklesla až na optimálních 5. Nejkratší cesta je naznačená tlustými čarami. Pamatujte proto, že Dijkstrův algoritmus musíte provádět vždy tak dlouho, dokud se konečně nezpracuje cílový vrchol. □

Následují dvě doplňkové ukázky vztahující se k látce Oddílu 3.4. Vzhledem ke svému specifickému zaměření na úzkou (i když významnou a populární) aplikační oblast mohou být čtenáři přeskočeny.

Příklad 3.18. *Ukažme si podrobně, jak problém plánování nejkratší cesty v reálné cestní síti (viz GPS navigace) modelovat váženým grafem.*

V prvním přiblížení je problém velmi jednoduchý, prostě:

- Jednotlivé křižovatky cest (nebo také slepé konce cest) budou vrcholy grafu
- a úseky cest mezi křižovatkami budou reprezentovány hranami, jejichž váha udává délku úseku nebo jízdní dobu, apod.

To se zdá být vše podstatné, ale jen na první pohled. Na druhý pohled si zvídavý čtenář uvědomí další omezení reálných cestních sítí jako třeba zakázaná odbočení či přikázané směry jízdy. Samotný vážený graf cest (a Dijkstrův algoritmus na něm) tato omezení nemůže reflektovat.

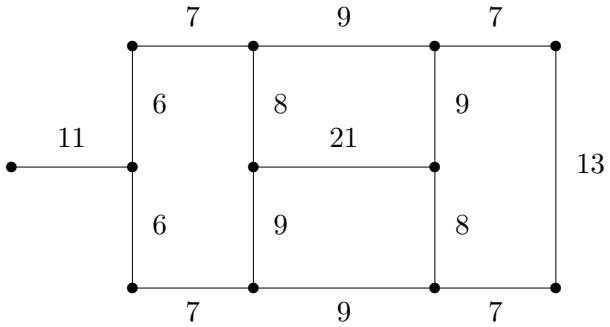
Ve druhé fázi tudíž věnujeme pozornost zmíněným dodatečným omezením přípustných cest. Ukazuje se, že jejich nejvhodnějším modelem je použití tzv. *manévrů*, které vyjadřují libovolné úseky vyžadující speciální pozornost. Formálně:

- Vedle samotného váženého grafu G, modelujícího naši cestní síť, je definována množina *manévrů* $\mathcal{M}$, kde každý prvek $P \in \mathcal{M}$ je sledem v G a má přiřazenu penalizaci nabývající libovolné (i záporné nebo ∞) hodnoty.
- Penalizovanou vahou sledu $S \subseteq G$ je pak součet délky S a penalizací všech manévrů, které jsou obsaženy jako podsledy v S. Dobře si povšimněte, že nejkratší sled vzhledem k penalizaci $\mathcal{M}$ již nemusí být cestou (může opakovat vrcholy).

Přitom Dijkstrův algoritmus lze vhodně zobecnit, aby v takto rozšířené cestní síti s manévry efektivně hledal nejkratší sled (za nutného předpokladu, že výsledné penalizace sledů nikdy nebudou záporné).

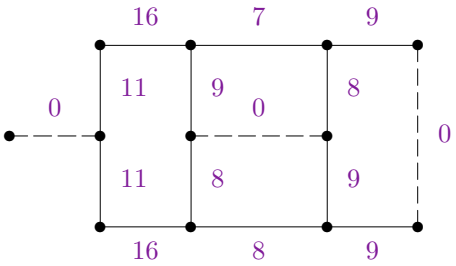
Za třetí je možno si všimnout ještě jednoho aspektu hledání najkratších cest, který nabývá na popularitě v navigačních zařízeních – váhy jednotlivých hran se v reálné cestní síti mohou v čase měnit. Jde o přirozený jev, kdy průjezdní doba na silnicích je výrazně delší v době dopravní špičky než mimo ni. V „lokálním“ pojetí není problém tento poznatek zahrnout do plánování cesty, prostě budeme vyhledávat vzhledem k aktuálním (ale statickým) vahám hran. Avšak z globálního pohledu plánování dlouhé trasy je nutno vzít do úvahy, že v jednotlivých oblastech sítě se potkáváme s různými fázemi dopravní špičky, a to je problém dosud nemající uspokojivá přesná algoritmická řešení. □

Příklad 3.19. *Určeme hodnotu dosahu (viz Oddíl 3.4) jednotlivých hran v následujícím váženém neorientovaném grafu:*



Které z hran se ukazují jako „bezvýznamné“?

Hodnoty dosahu jednotlivých hran spočítáme hrubou silou podle definice. Výsledkem jsou následující čísla:



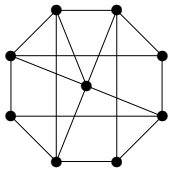
Velmi malých hodnot dosahu (konkrétně 0) nabývají vyznačené čárkované hrany, ty jsou tudíž z globálního pohledu plánování nevýznamné. Pochopitelně se jedná o příklad velmi malého grafu, takže výsledky jen lehce a dosti nepřesně naznačují skutečné chování parametru dosahu na velkých cestních sítích. □

Otázky a úlohy k řešení

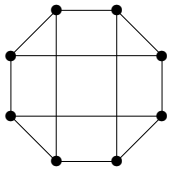
(3.5.1) *Jaká je největší vzdálenost mezi dvěma vrcholy v každém z následujících dvou grafů? Vyznačte tyto dva nejvzdálenější vrcholy.*

(3.5.2) *Jaká je největší vzdálenost mezi dvěma vrcholy v každém z následujících dvou grafů? Vyznačte tyto dva nejvzdálenější vrcholy.*

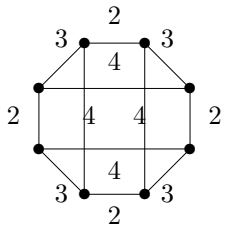
(3.5.3) *Kolik (neuspořádaných) dvojic vrcholů v následujícím grafu má vzdálenost právě* 3*?*



(3.5.4) *Kolik nejméně hran musíme přidat do následujícího grafu, aby největší vzdálenost mezi dvěma vrcholy byla* 2*? Zdůvodněte.*



(3.5.5) *Jaká je největší vzdálenost mezi dvojicí vrcholů v tomto váženém grafu?*



(3.5.6) *Kolik nejvíce vrcholů může mít graf, který má všechny vrcholy stupně* 3 *a největší vzdálenost mezi dvěma vrcholy je* 2*?*

(3.5.7) *Představme si graf, jehož vrcholy jsou všechna přirozená čísla od* 2 *do* 15 *a hrany spojují právě ty dvojice vrcholů, které jsou soudělné jako čísla. Přitom délka hrany je vždy rovna největšímu společném děliteli. (Například* $4, 5$ *nejsou spojené a* $8, 12$ *jsou spojené hranou délky* 4*.) Pomineme-li izolované vrcholy* 11 *a* 13*, jaká je největší vzdálenost mezi zbylými vrcholy tohoto grafu?*

*(3.5.8) *Proč každý* 3*-regulární graf s* 24 *vrcholy obsahuje dvojici vrcholů ve vzdálenosti* 4*?*

(3.5.9) *Nakreslete libovolný* 3*-regulární graf na* 12 *vrcholech s průměrem* 6*.*

*(3.5.10) *Pokud* 3*-regulární graf na* 12 *vrcholech obsahuje dva ve vzdálenosti* 5*, musí potom obsahovat i trojúhelník? Dokažte svou odpověď.*

(3.5.11) *Lze přidat dvě hrany ke kružnici délky* 10 *tak, aby výsledný graf měl průměr* 3*?*

*(3.5.12) *Kolika neisomorfními způsoby lze přidat dvě hrany ke kružnici délky* 12 *tak, aby výsledný graf měl maximální vzdálenost (průměr)* 4*?*

(3.5.13) *V návaznosti na Příklad 3.18 nalezněte ukázku cestní sítě s penalizovanými manévry, ve které existuje nejkratší sled (vzhledem k penalizované váze) opakující vrcholy.*

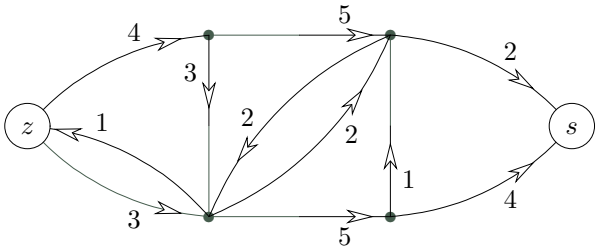
*(3.5.14) *Co by mělo být hlavní podstatnou změnou Dijkstrova algoritmu, aby jeho prostřednictvím bylo možno počítat nejkratší sledy vzhledem k manévrům (viz Příklad 3.18)?*

4 Toky v sítích

Úvod

Nyní se podíváme ještě na jednu oblast úloh, kde našla teorie grafů (konkrétně orientované grafy) bohaté uplatnění v praxi. Jde o oblast tzv. „síťových“ úloh: Pojem síť používáme jako souhrnné pojmenování pro matematické modely situací, ve kterých přepravujeme nějakou substanci (hmotnou či nehmotnou) po předem daných přepravních cestách, které navíc mají omezenou kapacitu.

Jedná se třeba o potrubní sítě přepravující vodu nebo plyn, o dopravní síť silnic s přepravou zboží, nebo třeba o internet přenášející data. Obvykle nás zajímá problém přenést z daného „zdroje“ do daného cíle čili „stoku“ co nejvíce této substance, za omezujících podmínek kapacit jednotlivých přepravních cest (případně i jejich uzlů). Obrázkem můžeme vyjádřit síť s danými kapacitami přepravy jako:



Problém maximálního toku v síti je snadno algoritmicky řešitelný, jak si také popíšeme. Jeho algoritmické řešení pak má (kupodivu) i mnoho teoretických důsledků, třeba pro souvislost či párování v grafech nebo výběry reprezentantů.

Cíle

Úkolem této lekce je teoreticky popsat problém toku v síti a vysvětlit základní algoritmus nenasycených cest pro jeho řešení. Dále jsou uvedeny některé důsledky vysvětlené látky (pro rozšířené sítě, pro bipartitní párování a výběr reprezentantů množin, pro vyšší souvislost grafů – Mengerovu větu, apod).

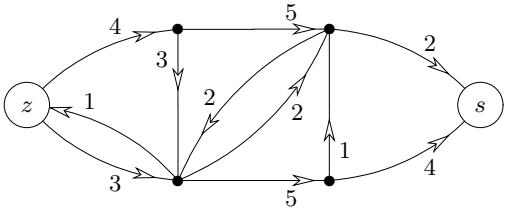
4.1 Definice sítě a toku

Základní strukturou pro reprezentaci sítí je orientovaný graf. Vrcholy grafu modelují jednotlivé uzly sítě a hrany jejich spojnice. Vzpomeňme si (Definice 1.17), že v orientovaných grafech je každá hrana tvořena uspořádanou dvojicí (u, v) vrcholů grafu, a tudíž taková hrana má *směr z vrcholu u do v*.

Definice 4.1. ***Síť*** je čtveřice $S = (G, z, s, w)$, kde

- G je orientovaný graf,
- vrcholy $z \in V(G)$, $s \in V(G)$ jsou *zdroj* a *stok*,
- $w : E(G) \to \mathbb{R}^+$ je kladné ohodnocení hran, zvané *kapacita hran*.

`Komentář:`



Na obrázku je zakreslena síť s vyznačeným zdrojem z a stokem s, jejíž kapacity hran jsou zapsány čísly u hran. Šipky udávají směr hran, tedy směr proudění uvažované substance po

spojnicích. Pokud směr proudění není důležitý, vedeme mezi vrcholy dvojici opačně orientovaných hran se stejnou kapacitou. Kapacity hran pak omezují maximální množství přenášené substance.

`Poznámka`: V praxi může být zdrojů a stoků více, ale v definici stačí pouze jeden zdroj a stok, z něhož / do nějž vedou hrany do ostatních zdrojů / stoků. (Dokonce pak různé zdroje a stoky mohou mít své kapacity.)

Obvykle nás na síti nejvíce zajímá, jak mnoho substance můžeme (různými cestami) přenést ze zdroje do stoku. Pro to musíme definovat pojem toku, což je formální popis okamžitého stavu přenášení v síti.

`Značení`: Pro jednoduchost píšeme ve výrazech znak $e \to v$ pro hranu e přicházející do vrcholu v a $e \leftarrow v$ pro hranu e vycházející z v.

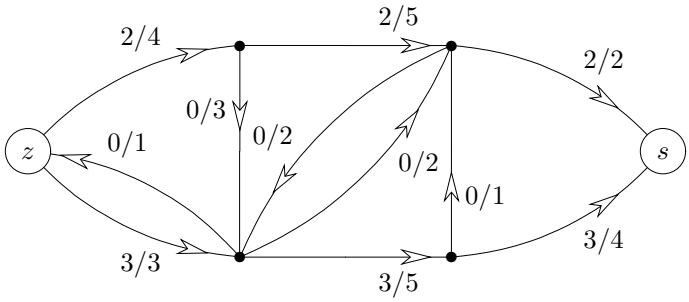
Definice 4.2. ***Tok*** v síti $S = (G, z, s, w)$
je funkce $f : E(G) \to \mathbb{R}_0^+$ splňující

- $\forall e \in E(G) : \ 0 \le f(e) \le w(e)$,
- $\forall v \in V(G), v \neq z, s : \ \sum_{e \to v} f(e) = \sum_{e \leftarrow v} f(e)$.

Velikost toku f je dána výrazem $\|f\| = \sum_{e \leftarrow z} f(e) - \sum_{e \to z} f(e)$.

`Značení`: Tok a kapacitu hran v obrázku sítě budeme zjednodušeně zapisovat ve formátu F/C, kde F je hodnota toku na hraně a C je její kapacita.

`Komentář:` Neformálně tok znamená, kolik substance je každou hranou zrovna přenášeno (ve směru této hrany, proto hrany musí být orientované). Tok je pochopitelně nezáporný a dosahuje nejvýše dané kapacity hrany. Navíc je nutno v každém vrcholu mimo z, s splnit podmínku „zachování substance“.



Ve vyobrazeném příkladě vede ze zdroje vlevo do stoku vpravo tok o celkové velikosti 5.

`Poznámka`: Obdobně se dá velikost toku definovat u stoku, neboť

$$0 = \sum_{e \in E} \big(f(e) - f(e)\big) = \sum_{v \in V} \left(\sum_{e \leftarrow v} f(e) - \sum_{e \to v} f(e) \right) = \sum_{v = z,s} \left(\sum_{e \leftarrow v} f(e) - \sum_{e \to v} f(e) \right) .$$

(Sčítance uprostřed předchozího vztahu nabývají nulové hodnoty pro všechny vrcholy v kromě z a s dle definice toku.) Proto velikost toku počítaná u zdroje je rovna opačné velikosti toku počítaného u stoku

$$\left(\sum_{e \leftarrow z} f(e) - \sum_{e \to z} f(e) \right) = \left(\sum_{e \to s} f(e) - \sum_{e \leftarrow s} f(e) \right) .$$

Otázky a úlohy k řešení

(4.1.1) *Jak ovlivní vlastnosti sítě přidání orientované smyčky?*

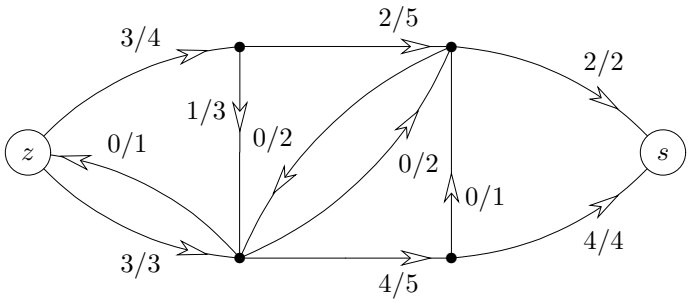
(4.1.2) *V dané síti máme dva různé toky stejné velikosti. Čím se od sebe „liší“?*

4.2 Hledání maximálního toku

Naším úkolem je najít co největší tok v dané síti. Pro jeho nalezení existují jednoduché a velmi rychlé algoritmy.

Problém 4.3. ***Maximálního toku v síti***
Je dána síť $S = (G, z, s, w)$ a našim úkolem je pro ni najít co největší tok ze zdroje z do stoku s vzhledem k ohodnocení w.
Formálně hledáme $\max \|f\|$ *dle Definice 4.2.*

`Komentář:` Tok velikosti 5 uvedený v ukázce v předchozí části nebyl optimální, neboť v této síti najdeme i tok velikosti 6:



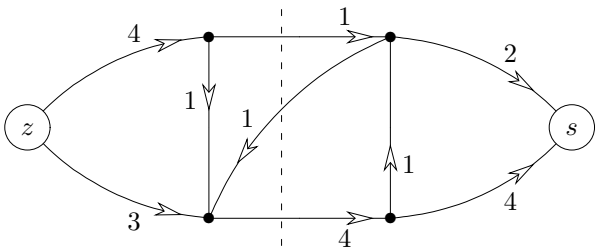
Jak však poznáme, že větší tok již v dané síti neexistuje? V této konkrétní ukázce to není obtížné, vidíme totiž, že obě dvě hrany přicházející do stoku mají součet kapacit $2 + 4 + 6$, takže více než 6 do stoku ani přitéct nemůže. V obecnosti lze použít obdobnou úvahu, kdy najdeme podmnožinu hran, které nelze tokem „obejít“ a které v součtu kapacit dají velikost našeho toku. Existuje však taková množina hran vždy? Odpověď nám dá následující definice a věta.

Pojem řezu v síti

Definice 4.4. ***Řez*** v síti $S = (G, z, s, w)$
je podmnožina hran $C \subset E(G)$ taková, že v podgrafu $G - C$ (tj. po odebrání hran C z G) nezbude žádná orientovaná cesta ze z do s.
Velikostí řezu C rozumíme součet kapacit hran z C, tj. $\|C\| = \sum_{e \in C} w(e)$.

Věta 4.5. *Maximální velikost toku v síti je rovna minimální velikosti řezu.*

`Komentář:` Na následujícím obrázku vidíme trochu jinou síť s ukázkou netriviálního minimálního řezu velikosti 5, naznačeného svislou čárkovanou čarou. Všimněte si dobře, že definice řezu mluví o přerušení *všech orientovaných cest ze z do s*, takže do řezu stačí započítat hrany jdoucí přes svislou čáru od z do s, ale ne hranu jdoucí zpět. Proto je velikost vyznačeného řezu $1 + 4 = 5$.



`Poznámka`: Tato věta poskytuje tzv. *dobrou charakterizaci* problému maximálního toku: Když už nalezneme maximální tok, tak je pro nás vždy snadné dokázat, že lepší tok není, nalezením příslušného řezu o stejné velikosti. Přitom toto zdůvodnění řezem můžeme směle ukázat i někomu, kdo se moc nevyzná v matematice.

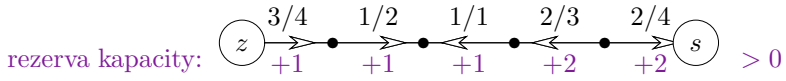
Důkaz Věty 4.5 bude proveden následujícím Algoritmem 4.7.

Nenasycené cesty v síti

`Definice`: Mějme síť S a v ní tok f. *Nenasycená cesta* (v S vzhledem k f) je neorientovaná cesta v G z vrcholu u do vrcholu v (obvykle ze z do s), tj. posloupnost navazujících hran $e_1, e_2, \ldots, e_m$, kde $f(e_i) < w(e_i)$ pro e_i ve směru z u do v a $f(e_i) > 0$ pro e_i v opačném směru.

Hodnotě $w(e_i) - f(e_i)$ pro hrany e_i ve směru z u do v a hodnotě $f(e_i)$ pro hrany e_i v opačném směru říkáme *rezerva kapacity* hran e_i. Nenasycená cesta je tudíž cesta s kladnými rezervami kapacit všech hran.

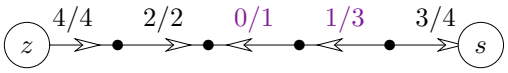
`Komentář:` Zde vidíme příklad nenasycené cesty ze zdroje do stoku s minimální rezervou kapacity +1.



Všimněte si dobře, že cesta není orientovaná, takže hrany na ní jsou v obou směrech.

Metoda 4.6. ***Maximální tok vylepšováním nenasycených cest.***
Základní myšlenkou této jednoduché metody hledání maximálního toku v dané síti je prostě opakovaně vylepšovat tok podél nalezených nenasycených cest.

`Komentář:` Ve výše zakresleném obrázku nenasycené cesty byla minimální rezerva kapacity ve výši +1, a tudíž nasycením této cesty o velikost toku 1 vzniká následující fragment přípustného toku v síti:



Zajímavé je se podívat, co se stalo s prostředními zpětně orientovanými hranami – fakticky byl jejich zpětný tok o 1 snížen/zastaven, přičemž toto množství nyní „teče doprava“ (velmi neformálně řečeno).

`Fakt`: Je-li minimální rezerva kapacity hran nenasycené cesty P ze z do s ve výši $r > 0$, pak tok ze z do s zvýšíme o hodnotu r následovně;

- pro hrany $e_i \in E(P)$ ve směru ze z do s zvýšíme tok na $f'(e_i) = f(e_i) + r$,
- pro hrany $e_i \in E(P)$ ve směru ze s do z snížíme tok na $f'(e_i) = f(e_i) - r$.

Výsledný f§ tok pak bude opět přípustný.

Základní algoritmus nenasycených cest

Implementační detaily Metody 4.6 následují popisem konkrétního algoritmu.

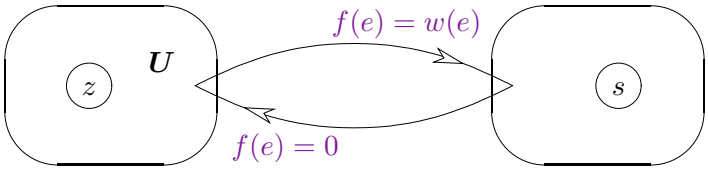
Algoritmus 4.7. ***Ford–Fulkersonův pro tok v síti.***

`vstup` *síť* $S = (G, z, s, w)$`;`
tok $f \equiv 0$`;`
`repeat` {
  Prohledáváním grafu najdeme množinu U *vrcholů* G*,*
  do kterých se dostaneme ze z *po nenasycených cestách*`;`
  `if` ($s \in U$) {
    `P =` *(výše nalezená) nenasycená cesta v* S *ze* z *do* s`;`
    Zvětšíme tok f *o minimální rezervu kapacity hran v* P`;`
  }

`until` ($s \notin U$);
`výstup` *vypíšeme maximální tok* f;
`výstup` *vypíšeme min. řez jako množinu hran vedoucích z* U *do* $V(G) - U$.

`Důkaz` správnosti Algoritmu 4.7:
Pro každý tok f a každý řez C v síti S platí $\|f\| \leq \|C\|$. Jestliže po zastavení algoritmu s tokem f nalezneme v síti S řez o stejné velikosti $\|C\| = \|f\|$, je jasné, že jsme našli maximální možný tok v síti S. (Pozor, zastavení algoritmu jsme zatím nezdůvodnili, to není tak jednoduché.)

Takže dokažme, že po zastavení algoritmu nastane rovnost $\|f\| = \|C\|$, kde C je vypsaný řez mezi U a zbytkem grafu G. Vezměme tok f v S bez nenasycené cesty ze z do s. Pak množina U z algoritmu neobsahuje s. Schematicky vypadá situace takto:



Jelikož z U žádné nenasycené cesty dále nevedou, má každá hrana $e \leftarrow U$ (odcházející z U) plný tok $f(e) = w(e)$ a každá hrana $e \rightarrow U$ (přicházející do U) tok $f(e) = 0$, takže

$$\sum_{e \leftarrow U} f(e) - \sum_{e \rightarrow U} f(e) = \sum_{e \leftarrow U} f(e) = \sum_{e \in C} w(e) = \|C\| \ .$$

Na druhou stranu, když v sumě uvažujeme všechny hrany grafu indukované na naší množině U, tj. $e \in E(G \restriction U)$, analogicky dostaneme požadované

$$0 = \sum_{e \in E(G \restriction U)} \big(f(e) - f(e)\big) = \sum_{v \in U} \left(\sum_{e \leftarrow v} f(e) - \sum_{e \rightarrow v} f(e) \right) - \left(\sum_{e \leftarrow U} f(e) - \sum_{e \rightarrow U} f(e) \right) =$$

$$= \left(\sum_{e \leftarrow z} f(e) - \sum_{e \rightarrow z} f(e) \right) - \left(\sum_{e \leftarrow U} f(e) - \sum_{e \rightarrow U} f(e) \right) = \|f\| - \|C\| \ , \quad \text{tj. } \|f\| = \|C\| \, .$$

□

Z důkazu Algoritmu 4.7 odvodíme několik zajímavých faktů:

`Fakt`: Pokud zajistíme, že Algoritmus 4.7 vždy skončí, zároveň tím dokážeme i platnost Věty 4.5. (Takže se teď podívejte na Algoritmus 4.9.)

`Fakt`: Pro celočíselné kapacity hran sítě S Algoritmus 4.7 vždy skončí.

Pro další využití v teorii grafů je asi nejdůležitější tento poznatek:

Důsledek 4.8. *Pokud jsou kapacity hran sítě S celočíselné, optimální tok také vyjde celočíselně.*

`Důkaz`: Postupujeme jednoduchou indukcí podle iterací Algoritmu 4.7:
Algoritmus začíná s celočíselným tokem 0. V každé další iteraci bude na počátku tok všemi hranami celočíselný, tudíž i rezervy kapacit hran vyjdou (rozdílem od celočíselné kapacity) celočíselně. Proto i hodnoty toku budou změněny jen celočíselně, což zakončuje indukční krok. □

Vylepšení algoritmu nenasycených cest

Bohužel pro reálná čísla kapacit hran není skončení Algoritmu 4.7 vůbec zaručeno, dokonce se dají najít extrémní případy, které nepovedou k řešení ani v limitě. Viz (4.5.7) ve cvičení. Proto je potřebné základní algoritmus (i pro potřeby teorie) poněkud vylepšit.

Algoritmus 4.9. ***Edmonds–Karpův pro tok v síti***
V Algoritmu 4.7 vždy sytíme nejkratší nenasycenou cestu, neboli prohledáváme naši síť do šířky (viz množina U). Tato implementace zaručeně skončí po $O(|V(G)| \cdot |E(G)|)$ iteracích cyklu, celkem tedy v čase $O(|V(G)| \cdot |E(G)|^2)$.

Ještě lepších výsledků dosahují následující "chytré" algoritmy.

Algoritmus 4.10. ***Dinicův pro tok v síti*** (náznak)
V intencích Algoritmu 4.7 provádíme následující iterace:

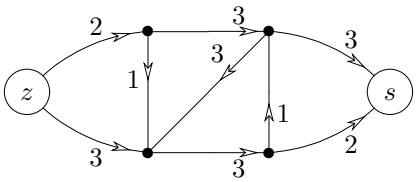
- Prohledáváním sítě do šířky nalezneme všechny nenasycené cesty nejkratší délky souběžně, které nám vytvoří "vrstvenou síť" (vrstvy odpovídají nenasycené vzdálenosti vrcholů od zdroje).
- Nalezené nenasycené cesty pak nasytíme novým prohledáním vrstvené sítě.

Tato implementace zaručeně skončí už po $O(|V(G)|)$ iteracích cyklu, ale jednotlivé iterace jsou poněkud náročnější, $O(|V(G)| \cdot |E(G)|)$.

Algoritmus 4.11. ***MPM („Tří Indů“) pro tok v síti*** (náznak)
Postupuje se stejně jako v Algoritmu 4.10, jen nesycení v druhém bodě proběhne rychlejším algoritmem v čase $O(|V(G)|^2)$ v každé iteraci, celkem tedy v $O(|V(G)|^3)$.

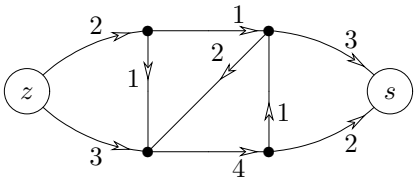
Otázky a úlohy k řešení

(4.2.1) *Jaký je maximální tok touto sítí ze z do s?*



(4.2.2) *Co obsahuje výsledná množina U Algoritmu 4.7 v předchozím příkladě?*

(4.2.3) *Kde je minimální řez v této síti mezi z a s?*



(4.2.4) *Proč Algoritmus 4.7 vždy skončí pro celočíselné kapacity hran?*

*(4.2.5)** *Dokažte časový odhad Algoritmu 4.9.*

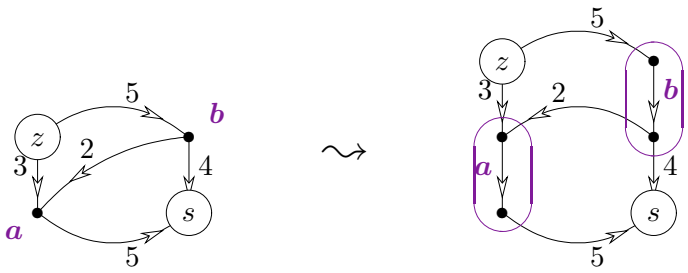
4.3 Zobecnění definice sítí

Pojmy sítě a toků v ní lze v kombinatorické optimalizaci zobecnit několika směry. My si zde stručně uvedeme tři možnosti.

Sítě s kapacitami vrcholů

U sítě můžeme zadat i *kapacity vrcholů*, neboli kapacitní váhová funkce je dána jako $w : E(G) \cup V(G) \to \mathbb{R}^+$. Význam pro přípustné toky v takové síti je, že žádným vrcholem nemůže celkem „protéct“ více než povolené množství substance.

`Fakt`. Takovou síť „zdvojením“ vrcholů snadno převedeme na běžnou síť, ve které kapacity původních vrcholů budou uvedeny u nových hran spojujících zdvojené vrcholy. Viz neformální schéma:



Sítě s dolními kapacitami

Pro hrany sítě lze zadat také jejich *minimální kapacity*, tedy dolní meze přípustného toku. Například u potrubní sítě mohou minimální vyžadované průtoky vody garantovat, že nedojde k zanesení potrubí, atd. To je modelováno druhou (vedle f) váhovou funkcí $\ell : E(G) \to \mathbb{R}_0^+$. Přípustný tok pak musí splňovat $\ell(e) \leq f(e) \leq w(e)$ pro všechny hrany e. V této modifikaci úlohy již přípustný tok nemusí vůbec existovat.

Algoritmus 4.12. ***Tok v síti s dolními kapacitami***
I tuto úlohu lze řešit dosud uvedenými nástroji, pokud postupujeme ve dvou fázích:

- *Nejprve nalezneme přípustnou cirkulaci v síti vzniklé přidáním „zpětné“ hrany sz. Toho lze dosáhnout hledáním toku ve speciální síti vyjadřující „přebytky“ (či nedostatky) dolních mezí toku v jednotlivých vrcholech.*
  - Z dané sítě $S = (G, z, s, w)$ utvoříme novou síť $S_0 = (G_0, z_0, s_0, w_0)$, kde G_0 vznikne přidáním hrany sz a nových vrcholů z_0, s_0 s hranami do a z $V(G)$ podle níže uvedeného pravidla. Pro $e \in E(G)$ je $w_0(e) = w(e) - \ell(e)$. Pro $x \in V(G)$ a „přebytek“ $p = \sum_{f \to x} \ell(f) - \sum_{f \leftarrow x} \ell(f) > 0$ je zavedena hrana z_0x s kapacitou $w(z_0x) = p$, naopak pro $p < 0$ je zavedena hrana xs_0 s kapacitou $w(xs_0) = -p$.
  - Pro S_0 je nalezen maximální tok, který musí nasytit všechny hrany vycházející ze z_0. Pokud takový neexistuje, neexistuje ani přípustné řešení původní úlohy. V opačné případě je tento tok na hranách G navýšen o příslušné dolní kapacity.
- *Poté z přípustné cirkulace jako výchozího stavu už získáme maximální tok kterýmkoliv algoritmem pro toky.*

Tzv. vícekomoditní toky

V síti lze najednou přepravovat více typů substancí. To vede na problém tzv. *vícekomoditních toků* v síti. Tento problém je složitější, už není v obecnosti snadno řešitelný a přesahuje rámec našeho textu, takže se jím nebudeme zabývat.

Kromě uvedených (a podobných) zobecnění toků v sítích jsou velmi zajímavé i některé speciální formulace problému toků, které se vyskytují v možná i nečekaných oblastech. Více o tom napíšeme v další části lekce.

Otázky a úlohy k řešení

(4.3.1) *Nakreslete si sami příklad sítě s dolními kapacitami bez přípustného toku.*

*(4.3.2)** *Dokažte si, že Algoritmus 4.12 funguje správně.*

4.4 Speciální aplikace sítí

Bipartitní párování

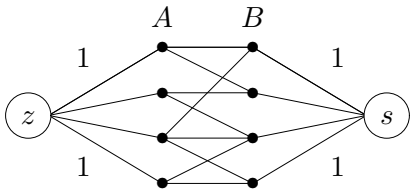
Biparitní grafy jsou grafy, jejichž vrcholy lze rozdělit do dvou množin tak, že všechny hrany vedou jen mezi těmito množinami, neboli podgrafy úplných bipartitních grafů.

`Definice`: *Párování* v (nyní biparitním) grafu G je podmnožina hran $M \subset E(G)$ taková, že žádné dvě hrany z M nesdílejí koncový vrchol.

> `Komentář:` Pojem (bipartitního) párování má přirozenou motivaci v mezilidských vztazích. Pokud pomineme bigamii a menšiny, můžeme si partnerské vztahy představit jako párování v bipartitním grafu. Jednu stranu grafu tvoří muži a druhou ženy. Hrana mezi mužem a ženou znamená vzájemné sympatie (přitom jedinec může mít vzájemné sympatie s několika jinými opačného pohlaví). Pak skutečné, tradiční partnerské vztahy by měly představovat párování v popsaném grafu.

Úlohu nalézt v daném bipartitním grafu co největší párování lze poměrně snadno vyřešit pomocí toků ve vhodně definované síti. Uvedená metoda použití toků v síti na řešení problému párování přitom hezky ilustruje obecný přístup, jakým toky v sítích pomohou řešit i úlohy, které na první pohled se sítěmi nemají nic společného.

Algoritmus 4.13. ***Nalezení bipartitního párování***
Pro daný bipartitní graf G s vrcholy rozdělenými do množin A, B sestrojíme síť S následovně:



Všechny hrany sítě S orientujeme od zdroje do stoku a přiřadíme jim kapacity 1. Nyní najdeme (celočíselný) maximální tok v S Algoritmem 4.7. Do párování vložíme ty hrany grafu G, které mají nenulový tok.

`Důkaz` správnosti Algoritmu 4.13: Podle Důsledku 4.8 bude maximální tok celočíselný, a proto každou hranou poteče buď 0 nebo 1. Jelikož však do každého vrcholu v A může ze zdroje přitéct jen tok 1, bude z každého vrcholu A vybrána do párování nejvýše jedna hrana. Stejně tak odvodíme, že z každého vrcholu B bude vybrána nejvýše jedna hrana, a proto vybraná množina skutečně bude párováním. Zároveň to bude největší možné párování, protože z každého párování lze naopak vytvořit tok příslušné velikosti a větší než nalezený tok v S neexistuje. □

`Poznámka`: Popsaná metoda je základem tzv. Maďarského algoritmu pro párování v bipartitních grafech (viz také Příklad 4.17). Úlohu nalezení maximálního párování lze definovat i pro obecné grafy a také ji efektivně algoritmicky vyřešit, ale příslušný algoritmus [Edmonds] není jednoduchý.

Vyšší grafová souvislost

Představme si, že na libovolném grafu G definujeme zobecněnou síť tak, že kapacity všech hran a všech vrcholů položíme rovny 1 v obou směrech. Pak celočíselný tok (viz Důsledek 4.8) velikosti k mezi dvěma vrcholy u, v se skládá ze soustavy k disjunktních cest (mimo společné koncové vrcholy u, v). Naopak řez odděluje u a v do různých souvislých komponent zbylého grafu. Aplikace Věty 4.5 na tuto situaci přímo poskytne následující tvrzení.

Důsledek 4.14. *Nechť u, v jsou dva vrcholy grafu G a $k > 0$ je přirozené číslo. Pak mezi vrcholy u a v existuje v G aspoň k disjunktních cest, právě když po odebrání libovolných $k-1$ vrcholů různých od u, v z G zůstanou u a v ve stejné komponentě souvislosti zbylého grafu.*

Použitím tohoto tvrzení pro všechny dvojice vrcholů grafu snadno dokážeme dříve uvedenou důležitou Větu 2.8 (Mengerovu). Ještě jiné použití si ukážeme na problému výběru reprezentantů množin.

Výběr různých reprezentantů

`Definice`: Nechť $M_1, M_2, \ldots, M_k$ jsou neprázdné množiny. *Systémem různých reprezentantů* množin $M_1, M_2, \ldots, M_k$ nazýváme takovou posloupnost různých prvků $(x_1, x_2, \ldots, x_k)$, že $x_i \in M_i$ pro $i = 1, 2, \ldots, k$.

Důležitým a dobře známým výsledkem v této oblasti je Hallova věta plně popisující, kdy lze systém různých reprezentantů daných množin nalézt.

Věta 4.15. *Nechť $M_1, M_2, \ldots, M_k$ jsou neprázdné množiny. Pro tyto množiny existuje systém různých reprezentantů, právě když platí*

$$\forall J \subset \{1, 2, \ldots, k\} : \left|\bigcup_{j \in J} M_j\right| \geq |J|\,,$$

neboli pokud sjednocení libovolné skupiny z těchto množin má alespoň tolik prvků, kolik množin je sjednoceno.

`Důkaz`: Označme $x_1, x_2, \ldots, x_m$ po řadě všechny prvky ve sjednocení $M_1 \cup M_2 \cup \cdots \cup M_k$. Definujeme si bipartitní graf G na množině vrcholů $\{1, 2, \ldots, k\} \cup \{x_1, x_2, \ldots, x_m\} \cup \{u, v\}$, ve kterém jsou hrany $\{u, i\}$ pro $i = 1, 2, \ldots, k$, hrany $\{v, x_j\}$ pro $j = 1, 2, \ldots, m$ a hrany $\{i, x_j\}$ pro všechny dvojice i, j, pro které $x_j \in M_i$.

> `Komentář`: Konstrukce našeho grafu G je obdobná konstrukci sítě v Algoritmu 4.13: Vrcholy u a v odpovídají zdroji a stoku, ostatní hrany přicházející do vrcholu x_j znázorňují všechny z daných množin, které obsahují prvek x_j.

Cesta mezi u a v má tvar u, i, x_j, v, a tudíž ukazuje na reprezentanta $x_j \in M_i$. Systém různých reprezentantů tak odpovídá k disjunktním cestám mezi u a v. Nechť X je nyní libovolná minimální množina vrcholů v G, neobsahující samotné u a v, po jejímž odebrání z grafu nezbude žádná cesta mezi u a v. Podle Důsledku 4.14 a této úvahy mají naše množiny systém různých reprezentantů, právě když každá taková oddělující množina X má aspoň k prvků.

Položme $J = \{1, 2, \ldots, k\} \setminus X$. Pak každá hrana z J (mimo u) vede do vrcholů z $X \cap \{x_1, \ldots, x_m\}$ (aby nevznikla cesta mezi u, v), a proto

$$\left|\bigcup_{j \in J} M_j\right| = |X \cap \{x_1, \ldots, x_m\}| = |X| - |X \cap \{1, \ldots, k\}| = |X| - k + |J|\,.$$

(První rovnost vyplývá z minimality množiny X!) Vidíme tedy, že $|X| \geq k$ pro všechny (minimální) volby oddělující X, právě když $\left|\bigcup_{j\in J} M_j\right| \geq |J|$ pro všechny volby J, což je dokazovaná podmínka naší věty. □

`Poznámka`: Tento důkaz nám také dává návod, jak systém různých reprezentantů pro dané množiny nalézt – stačí použít Algoritmus 4.7 na vhodně odvozenou síť.

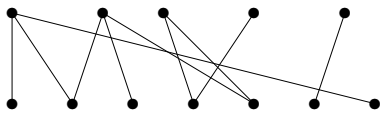
Segmentace obrazu

Teorii i algoritmy toků a řezů v sítích lze výhodně aplikovat i v následující vyloženě praktické oblasti: Jedním ze základních problémů počítačového vidění je segmentace obrazu na popředí a pozadí. K jeho řešení lze (mezi mnoha jinými přístupy) zvolit i následující postup, který velmi neformálně nyní naznačíme:

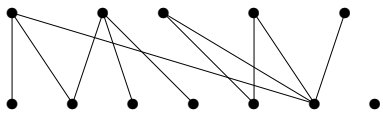
- Vstupem je matice obrazových bodů (pixelů), z nichž každý má přiřazeny hodnoty pravděpodobnosti bytí v popředí a bytí v pozadí. Dále jsou určeny „separační penalizace“ pro dvojice sousedních pixelů.
- Síť zkonstruhujeme přidáním univerzálního zdroje z a stoku s, přičemž hrany ze z do pixelů mají kapacitu rovnou pravděpodobnosti bytí v popředí a hrany do s obdobně pro pozadí. Hrany mezi sousedními pixely jsou obousměrné s kapacitami rovnými zmíněným penalizacím.
- Minimální řez v této síti poté určuje „přechody“ mezi popředím a pozadím daného obrazu.

Otázky a úlohy k řešení

(4.4.1) *Najděte největší párování v tomto bipartitním grafu:*



(4.4.2) *Najděte největší párování v tomto bipartitním grafu:*



(4.4.3) *Mají všechny tříprvkové podmnožiny množiny* $\{1, 2, 3, 4\}$ *systém různých reprezentantů?*

(4.4.4) *Mají všechny tříprvkové podmnožiny množiny* $\{1, 2, 3, 4, 5\}$ *systém různých reprezentantů?*

Rozšiřující studium

Problematika toků v sítích je běžnou součástí teorie grafů (i když není pokryta v [5]) a je možno najít její popis třeba v [3], případně na volně dostupných webových zdrojích jako `http://en.wikipedia.org/wiki/Flow_network`, `http://en.wikipedia.org/wiki/Ford-Fulkerson_algorithm`, *nebo*

http://mathworld.wolfram.com/NetworkFlow.html. *Jiný podrobný popis variant algoritmů pro toky v sítích se nachází v prvních dvou kapitolách [4]. Jedná se také o klasické téma kombinatorické optimalizace. Z dalších internetových zdrojů je možno zmínit třeba implementace na* http://www.cs.sunysb.edu/~algorith/files/network-flow.shtml.

Vícekomoditní toky přímo zobecňují toky jedné komodity (substance), které byly předneseny v této lekci. Jedná se však o problém výpočetně mnohem složitější (NP-úplný už pro 2 komodity a celočíselné kapacity) a neexistují pro něj podobně snadné algoritmy. Jako takovým jsme se proto tímto problémem nezabývali a čtenáře pouze odkazujeme na „rozcestník“ http://en.wikipedia.org/wiki/Multi-commodity_flow_problem.

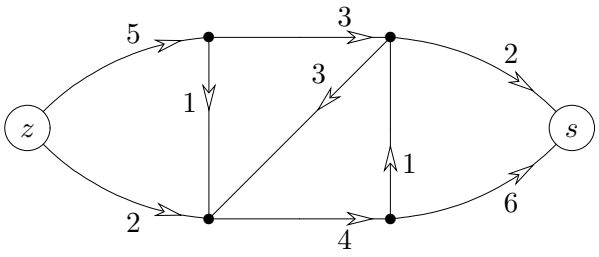
Teorie párování (Oddíl 4.4) tvoří samostatnou rozvinutou oblast grafů, mající zajímavé aplikace v jiných oborech jako statistické fyzice. Zájemce o obecnou teorii odkazujeme třeba na [1, Chapter 2] nebo internetově na http://en.wikipedia.org/wiki/Matching.

Problematika segmentování obrazu leží mimo teorii grafů a použití toků je v ní pouze okrajové, přesto uvádíme jeden obecný odkaz pro možné zájemce http://en.wikipedia.org/wiki/Segmentation_(image_processing).

4.5 Cvičení: Příklady toků v sítích

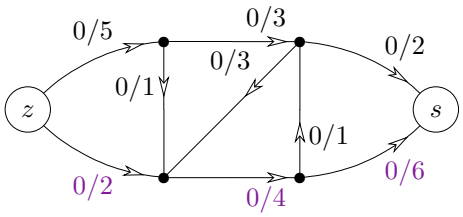
Mnohé příklady definic sítí a toků v nich byly již zmíněny v předchozím textu lekce, a proto ve cvičení přejdeme rovnou k ukázkovému řešení úloh souvisejících s toky v sítích. Cílem je nejen se naučit používat poměrně jednoduchý algoritmus hledání maximálního toku v síti, ale především se seznámit se širokým spektrem situací a problémů, pro jejichž vyřešení jsou toky v sítích užitečné a mnohdy klíčové.

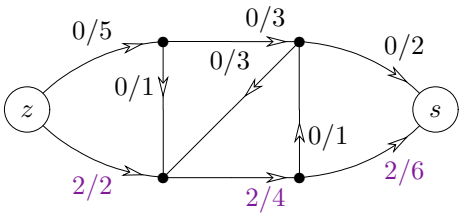
Příklad 4.16. *Jaký je maximální tok znázorněnou sítí ze z do s?*



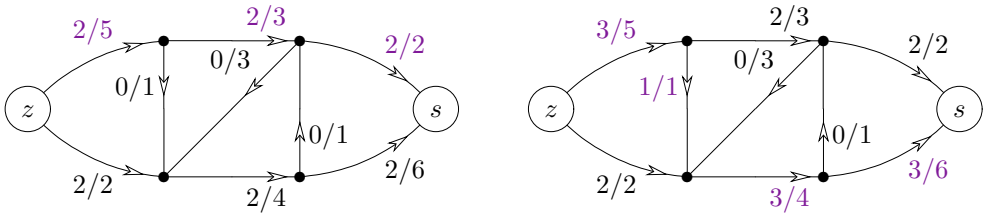
Na tomto příkladě si ukážeme průběh Algoritmu 4.7.

Začneme s nulovým tokem a najdeme nejkratší z nenasycených cest mezi z a s (vedoucí spodem grafu a vyznačenou na prvním obrázku). Minimální rezerva kapacity na této cestě je 2, takže tok nasytíme o 2 podél této cesty (viz obrázek vpravo).

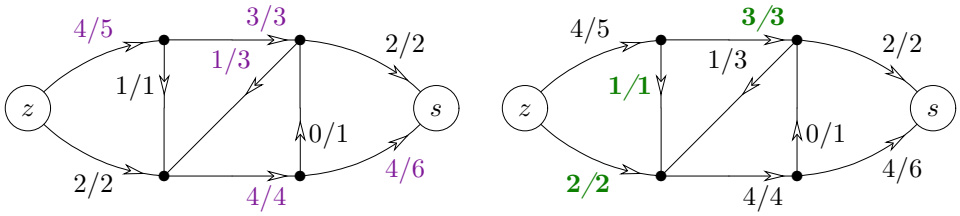




V dalším kroku najdeme nenasycenou cestu délky 3 vedoucí vrchem grafu. Její minimální rezerva kapacity je opět 2, takže ji o tolik nasytíme (viz další obrázek vlevo). Nyní již nenasycená cesta délky 3 neexistuje, takže najdeme nenasycenou cestu délky 4 s rezervou kapacity 1, kterou hned nasytíme (viz obrázek vpravo).

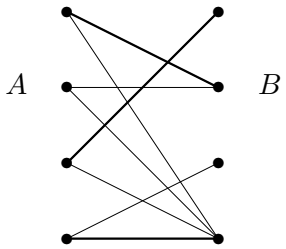


Obdobně ještě najdeme nenasycenou cestu délky 5, kterou také nasytíme o 1. Závěrečný obrázek nám ukazuje všechny nenasycené hrany, mezi kterými již nevede žádná nenasycená cesta ze z do s, takže máme maximální tok velikosti 6 v naší síti.



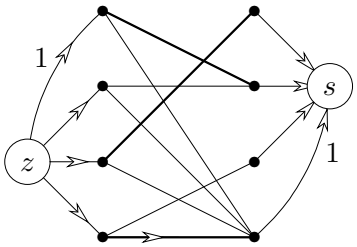
Zároveň je na posledním obrázku vyznačen i příslušný minimální řez velikosti 6. □

Příklad 4.17. *Uvažujme vyznačené párování velikosti* 3 *v následujícím bipartitním grafu* G.



Odvoďte, jak vypadají v tomto grafu nenasycené cesty sítě použité v Algoritmu 4.13 pro výpočet bipartitního párování. Vylepšete stávající párování pomocí zvolené nenasycené cesty.

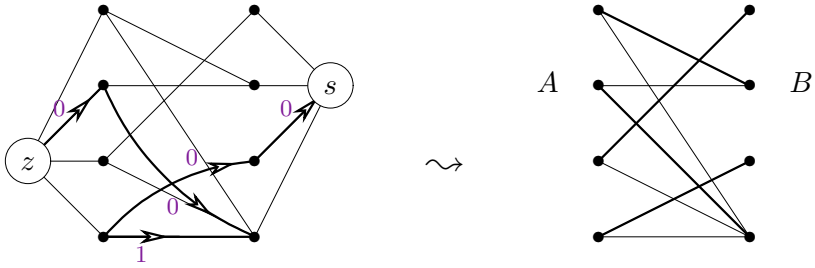
Síť konstruhovaná Algoritmem 4.13 vypadá (podle definice) následovně:



Jelikož všechny kapacity hran jsou 1 a tok je volen celočíselně hranami náležejícími do párování, každá nenasycená cesta má rezervu kapacity $+1$ a začíná šipkou z s do A toku 0, pokračuje šipkou z A do B toku 0, případně se vrací proti šipce z B do A při toku 1, a tak dále…, až poslední je šipka z B do s toku 0. Nejjednodušší by byla taková nenasycená cesta délky 3, která odpovídá hraně v G nemající ani jeden konec incidentní se stávajícím párováním (takovou hranu lze přidat přímo do párování).

Ve zobrazeném případě však žádnou další hranu přímo do párování nelze přidat. Za nenasycenou cestu mezi z, s lze tak například zvolit tu níže zobrazenou, jejímž nasycením

vznikne nové (již perfektní) párování vpravo:



Obecně nenasycené cestě při řešení úlohy maximálního párování odpovídá tzv. *volná střídavá cesta*, což je cesta v grafu mající právě každou druhou svou hranu v párování, avšak první a poslední hrana v párování není. Takovou volnou střídavou cestu lze „přepnout“ v párování na doplněk a získat tak párování o jednu hranu větší. □

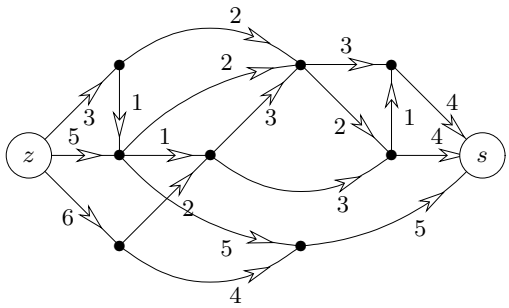
Příklad 4.18. *Mějme neorientovaný graf G. Pak hrany G lze zorientovat tak, aby z každého vrcholu vycházela nejvýše jedna šipka, právě když žádný podgraf v G neobsahuje více hran než vrcholů. Dokažte.*

Ač to tak na první pohled nevypadá, jedná se o příklad na systém různých reprezentantů (SRR): Máme systém dvouprvkových množin—hran G, přičemž reprezentantem každé hrany bude počáteční vrchol její orientace coby šipky. Zřejmě požadovaná orientace G existuje, právě když existuje tento SRR.

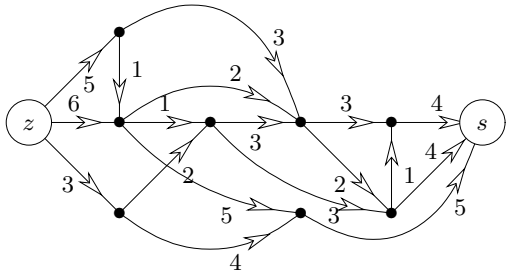
Podle Hallovy věty má systém hran G různé reprezentanty, právě když pro každé $F \subseteq E(G)$ platí, že $|\bigcup F| \geq |F|$, kde $\bigcup F$ zkráceně zapisuje podmnožinu těch vrcholů incidentních s některou hranou F. Potom pokud některý podgraf $H \subseteq G$ má více hran než vrcholů, je $|\bigcup E(H)| = |V(H)| < |E(H)|$ a SRR neexistuje. Naopak pokud SRR neexistuje, je $|\bigcup F| < |F|$ pro vhodnou F a stačí volit H (s více hranami než vrcholy) indukovaný touto množinou hran F. □

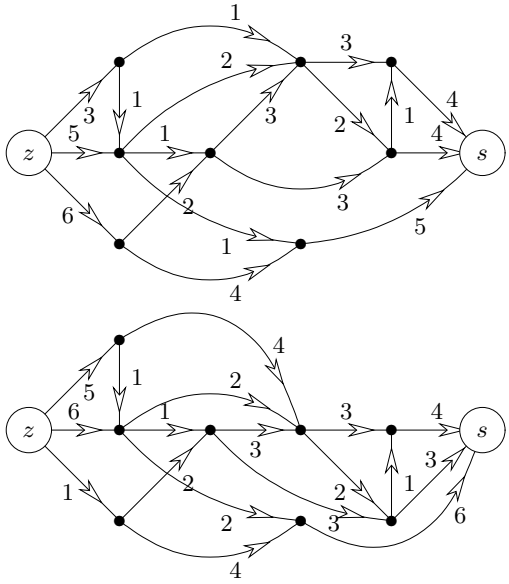
Otázky a úlohy k řešení

(4.5.1) *Najděte maximální tok v této síti. (Kapacity hran jsou vyznačeny u svých šipek.) Tok do obrázku zapište, vyznačte hrany příslušného minimálního řezu a napište velikost toku.*



(4.5.2) *Obdobně najděte maximální toky v následujících sítích.*





(4.5.3) *Má tento seznam množin systém různých reprezentantů? Pokud ano, vyznačte je v množinách, jinak správně zdůvodněte, proč systém různých reprezentantů nemají.*

$\{1,2,3,4\}$, $\{6,7,8,9\}$, $\{3,5,7\}$,
$\{3,4,5,6\}$, $\{1,2,8,9\}$, $\{3,6,7\}$,
$\{4,5,6,7\}$, $\{4,5,6\}$, $\{4,6,7\}$.

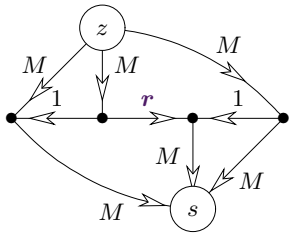
(4.5.4) *Má tento seznam množin systém různých reprezentantů? Pokud ano, vyznačte je v množinách, jinak správně zdůvodněte, proč systém různých reprezentantů nemají.*

$\{1,2,3,4\}$, $\{6,7,8,9\}$, $\{1,3,9\}$,
$\{3,4,5,6\}$, $\{1,2,3,9\}$, $\{2,4,9\}$,
$\{4,5,6,7\}$, $\{2,3,4\}$, $\{3,4,9\}$.

*(4.5.5) *Dokažte, že hrany neorientovaného grafu G lze zorientovat tak, aby z každého vrcholu vycházely nejvýše dvě šipky, právě když žádný podgraf v G neobsahuje více jak dvojnásobek hran než vrcholů.*

*(4.5.6) *Představme si graf G nakreslený do roviny bez křížení hran (viz Lekce **??** pro bližší definice). Nechť u, v jsou dva různé vrcholy v nakreslení. Pak z u do v lze dojít „procházkou" po stěnách grafu s překročením nejvýše k jiných vrcholů, právě když neexistuje $k+1$ vzájemně disjunktních vnořených kružnic oddělujících u od v.*

(4.5.7) *Ukažte, že v následujícím příkladu sítě s reálnými kapacitami Algoritmus 4.7 při (ne)vhodném výběru nenasycených cest nikdy neskončí, dokonce ani nekonverguje k maximálnímu toku.*



$r = \frac{\sqrt{5}-1}{2}$, $M \geq 2$

Část II
Více Teorie Grafů

.....

Mimo témat přímo vázaných k nejčastějím (obvykle) informatickým aplikacím nám teorie grafů nabízí i mnohá jiná zajímavá zákoutí, která si zaslouží své místo v obecném kurzu a jeho učebním textu. Následující tři lekce tak nabízejí reprezentativní výběr dalších tradičních grafových oblastí a témat, která se oproti předchozí části vyznačují poněkud vyšší matematickou náročností, avšak stále si zachovávají užitečné vazby také na informatiku.

Část III
Vybrané Pokročilé Partie

.....

Poslední část našeho studijního textu se od dosavadních lekcí liší podstatněji (a nejen absencí tradičních cvičení na závěr). Zatímco předchozí lekce pokrývaly základní otázky teorie grafů, které by měly být v každém obecném kurzu vysvětleny, nyní se zběžně a tak trochu i neformálně zaměříme na několik vybraných partií pokročilé teorie grafů, které sice přesahují běžnou náplň základních kurzů, ale jsou tak nějak „blízké autorovu srdci“. Dalším společným jmenovatelem je fakt, že příští lekce nepodávají ucelený výklad, nýbrž hlavně naznačují zajímavé směry vývoje a náměty pro další pokročilé studium.

Tato část učebnice bude v permanentním vývoji (také neboť reaguje na nejnovější publikované poznatky) a výběr látky z ní k výuce na FI MU bude na aktuálním rozhodnutí učitele...

Část IV

Klíč k řešení úloh

(1.1.1) Může, je $E(G) = \emptyset$, kreslíme z něj jen vrcholy.

(1.1.2) Žádnou.

(1.1.3) Kružnice délky 5.

(1.1.4) Pro $n = 1$ a 2, cesty délky 0 a 1.

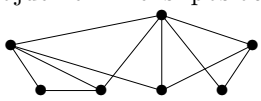
(1.1.5) Pro $n = 3$, trojúhelník.

(1.1.6) Pro $m = n = 2$, délky čtyři.

(1.1.7) Pro $m = 1$ nebo $n = 1$ a druhé libovolné.

(1.1.8) 9

(1.1.9) Zákonitě dojdeme k menší posloupnosti se záporným stupněm (-1).

(1.1.10) Ano: 

(1.1.11) Ano, kružnice délky 6 a dvě kružnice délky 3 vedle sebe.

(1.1.12) Po odebrání případného izolovaného vrcholu zbývá n vrcholů, ze kterých každý má stupeň mezi 1 a $n - 1$, takže některé musí mít stejné stupně.

(1.2.1) Není.

(1.2.2) Ano, vždyť se zachovají všechny stupně.

(1.2.3) Ano, ale musí mít délku aspoň o 2 kratší než délka kružnice. Vidíte proč?

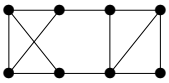
(1.2.4) Každá klika je z definice indukovaná, takže tam to moc smysl nemá. U hvězdy naopak můžeme mít indukované i neindukované případy.

(1.2.5) B

(1.2.6) 3

(1.2.7) 7

(1.2.8)* 5, dokážete ji vůbec najít?

(1.2.9) Ano, například  . Tento graf obsahuje 4 trojúhelníky, kdežto původní graf jen 3. Mohli jste však sestrojit úplně jiný graf...

(1.2.10)* a) Jeden, jen kružnice délky 5. b) Dva, kružnice délky 6 a dva trojúhelníky. c) Čtyři...

(1.3.1) Ano, neobsahuje nic, takže neobsahuje ani kružnici, a zároveň je souvislý.

(1.3.2) Jsou to cesty P_n a úplné bipartitní grafy $K_{1,n}$.

(1.3.3) Lze, ale jenom o kružnici. Úplné grafy K_1 a K_2 totiž stromy jsou.

(1.3.4) 3, jeden bez hran, jeden s jednou hranou a poslední strom s dvěma hranami.

(1.3.5) 2, cesta P_3 a hvězda $K_{1,3}$.

(1.3.6)* Strom je právě takový souvislý graf, který má pro n vrcholů přesně $n - 1$ hran.

(1.3.7)* Strom je takový graf, ve kterém je každá dvojice vrcholů spojena právě jedinou cestou.

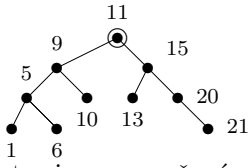
(1.4.1) Tyto zajímavé typy vrcholů orientovaných grafů se nazývají také zdroj a stok.

(1.4.2) Neboť definice hran z $V \times V$ umožňuje hrany typu (u, u) i $(u, v), (v, u)$ zároveň. Totéž není možné u hran coby neuspořádaných dvojich vrcholů.

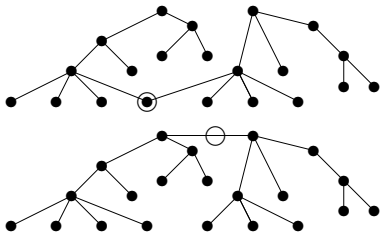
(1.5.1) Prázdný graf – strom.

(1.5.2) Nejdelší u cesty P_n – $\lfloor n/2 \rfloor$ kroků, nejkratší u hvězdy $K_{1,n}$ – 1 krok.

(1.5.3)* Centrum vždy bude ležet na té nejdelší cestě. Bude tvořeno jedním vrcholem, pokud je nejdelší cesta sudé délky, a hranou, pokud je liché délky.

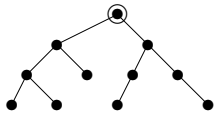
(1.5.4) 

(1.5.5) Centra jsou vyznačená:



(1.5.6) Pokud centrum vyjde jeden vrchol, odebereme celkem 32 listů – za každou hranu jeden. Jinak odebereme jen 31 listů a jedna hrana zbude jako centrum.

(1.5.7) ((()(()()())) ((()())(())))

(1.5.8) 

(1.6.1) Oba stejně 36 hran.

(1.6.2) 34

(1.6.3) Sudý počet, aby součet stupňů vyšel sudý, a ≥ 4.

(1.6.4) Ano.

(1.6.5) Ne.

(1.6.6) Pro $x = 7, 9, 11$.

(1.6.7) Dva.

(1.6.8) A: 23, B: 20, C: 22.

(1.6.9) Snadno, začněte mapováním mezi jedinečnými vrcholy stupně 4.

(1.6.10) Isom. z pravého do levého grafu: spodní dva nahoru vpravo, pravé dva dolů vpravo, atd.

(1.6.11) Začněte mapování od jedinečného (izolovaného) trojúhelníku v grafech.

(1.6.12) Vlevo 3, vpravo také 3.

(1.6.13) $A \simeq B \simeq D$, ale C má kružnici délky 4. Nakreslili jste si hezké obrázky, aby toto bylo vidět?

(1.6.14) $B \simeq C$, $A \simeq D$, u neisomorf. dvojic vždy jedna obsahuje kružnice délky 5 a druhá ne.

(1.6.15) Nejdelší C_8 v obou, nejdelší indukovaná C_6 v A a jen C_5 v B.

(1.6.16) Ano, třeba v úloze 1.6.10 vlevo.

(1.6.17) ...

(1.6.18) A ano, $K_{3,3}$ a graf trojbokého hranolu. B ne, protože dva vrcholy stupně 3 musí být spojené s ostatníma a tím již získáme celý graf jednoznačně. C ne, neboť lze vrchol stupně 2 zanedbat a zbytek musí být K_4.

(1.6.19)* 20, po vynechání jednoho vrcholu jsou vždy dvě možnosti na vyplnění zbytku kružnicí.

(1.6.20)* $\binom{10}{4} \cdot 3$, neboť $\binom{10}{4}$ způsoby vybereme čtveřici vrcholů pro ten podgraf a každé 4 vrcholy lze třemi způsoby propojit do kružnice.

(1.6.21) Ano, 2005 vrcholů podle Věty 1.12.

(1.6.22) $2009 - 1 - 1 - 1 - 1 = 2005$

(1.6.23) 11, počítejte hrany v každé komponentě – stromu.

(1.6.24)* Nenajdete, to by bylo v rozporu s Důsledkem 1.14.

(1.6.25) Je mnoho možností, třeba tato: Rozložte hrany K_7 do tří kružnic C_7, každou zorientujte jedním směrem. Jedna z těch tří kružnic prochází vrcholy po řadě, druhá ob dva, třetí ob tři vrcholy.

(1.6.26) Protože každá šipka jednou vychází a jednou přichází, takže výchozích musí být celkem stejně jako příchozích.

(1.7.1) Ano v A: graf $K_{2,3}$ a graf C_5 s jednou diagonálou; C: cesta délky 4 a trojúhelník s hranou vedle; D: úplný K_4 s hranou vedle a dále K_4 s "rozpojenou" hranou. Nelze pro B: uvažte doplněk toho grafu – má stupně $2, 1, 1, 1, 1$, což dá jednoznačný graf.

(1.7.2)* $A \simeq C \simeq D$

(1.7.3)* 5

(1.7.4)* Je jich 10, 6 z nich vznikne různým přidáváním vrcholů stupně 2 na hrany úplného grafu K_4 a 4 další jsou jiné.

(1.7.5)* 11

(1.7.6)* Jen 4 (odtrhejte 8 vrcholů stupně 1, co zbyde?).

(2.1.1) Stačí místo každého opakovaného uložení na zásobník jen "zvednout" již uložený odkaz na vrchol.

(2.2.1) 4

(2.2.2) Vrcholově n-souvislý.

(2.2.3) Najdeme 3 disjunktní cesty mezi x, y. Proto musíme vypustit 3 vrcholy.

(2.2.4) Prostě začněte z C_5.

(2.2.5) 1 do kružnice.

(2.2.6) $\binom{n-1}{2}$

(2.2.7)* Třeba podle přidávání uší – poslední přidaná cesta v konstrukci našeho grafu G má délku 1, neboli je naší hranou e. Proč?

(2.3.1) Nalezneme kostru s největším celkovým ohodnocením.

(2.3.2) Není potřeba na začátku všechny (mnoho) hrany seřadit, seřazují se jen některé hrany potřebné v průběhu algoritmu.

(2.3.3) Hlavně nějaký rychlý typ haldy pro ukládání seznamu hran vycházejících ze současné komponenty ven a vybírání nejmenší z nich.

(2.4.1) Ne, je to vidět v ilustracích k textu – hrany, co nejsou v cyklu, neleží v žádné silné komponentě.

(2.4.2) Cyklus délky 2 (dvojice protiběžných hran).

(2.4.3) Právě když neleží v žádné orientované kružnici.

(2.4.4) Cyklus (délky větší než 1) by dle definice byl součástí nějaké silné komponenty.

(2.5.1) Žádnému!!! Jde o multigraf na 4 vrcholech se 7 hranami.

(2.5.2) Dva.

(2.5.3) Ne, má čtyři vrcholy lichého stupně.

(2.5.4) Tři, dvě by teoreticky stačily na změnu všech stupňů na sudé, ale v tomto grafu jen dvě hrany prostě přidat nejdou.

(2.6.1) 5, samé trojúhelníky.

(2.6.2) 6, samé úplné grafy K_5.

(2.6.3) Souvislý s 15 hranami (ke každému vrcholu 3 sousedé).

(2.6.4) 18, stupně ≥ 3, třeba jako šestiboký hranol.

(2.6.5) a) 28: $K_8 + K_1 + K_1$, b) 9: $K_3 + K_3 + K_3 + K_1$

(2.6.6)* Některý vrchol má stupeň aspoň 3, jinak vezmeme doplněk. Pak ti tři sousedi jsou nezávislí...

(2.6.7) Třeba úplný bipartitní $K_{3,4}$.

(2.6.8)* Pokud máme vrchol, ve kterém žádná šipka nekončí, položíme jej první v řadě a pokračujeme indukcí. Jinak v grafu nalezneme orientovanou kružnici (neznámé délky) a z ní lze minimalizací získat kružnici délky 3.

(2.6.9) Ano, ale jen otevřeným.

(2.6.10) Není to strom, proto obsahuje kružnici.

(2.6.11)* Pomůže vám nakreslení uzavřeným tahem; co tak vybrat z něj každou druhou hranu?

(2.6.12) Jde o Eulerovský tah, který musí být uzavřený, neboť všechny stupně jsou sudé 6 (plus

2 za smyčku).

(3.1.1) 1, ale jen 0 pro graf na jednom vrcholu.

(3.1.2) 2

(3.1.3) 5

(3.1.4) 7

(3.1.5) $n-1$ – tj. počet hran kostry.

(3.1.6) 3, najdete příslušnou dvojici vrcholů?

(3.1.7) Poloměr 2, centrum obsahuje právě 3 vrcholy (najdete je?).

(3.1.8) Třeba: 

(3.1.9)* $\mathrm{rad}(G) \leq \mathrm{diam}(G) \leq 2 \cdot \mathrm{rad}(G)$, rovnost v prvním pro K_n, v druhém pro cestu.

(3.2.1) 5

(3.2.2) Jsou to oba vrcholy „nad“ a, b, do každého dalšího vrcholu je z nich vzdálenost nejvýše 4. Lépe to nejde.

(3.2.3) Ne, pro každou dvojici lze sled zkracovat neustálým opakováním záporného cyklu.

(3.2.4) Ano, ale graf nesmí být silně souvislý. Pro dvojici vrcholů z jiné silné komponenty než té se záporným cyklem je vzdálenost korektní.

(3.3.1) Krok výběru dalšího vrcholu ke zpracování – i když bereme nejbližší nezpracovaný vrchol, z jiného, vzdálenějšího, vrcholu se lze později přes hranu záporné délky dostat do vybraného vrcholu “kratší” cestou.

(3.3.2) Bohužel nebude, důvod je stejný jako v předchozí otázce. Je nutno použít jiné algoritmy.

(3.3.3) Vyjde $\begin{pmatrix} 0 & 1 & 2 & 1 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 1 \\ 1 & 2 & 1 & 0 \end{pmatrix}$.

(3.3.4) Není, hodnoty s jedním indexem t se přece v iteraci t nezmění nikdy.

(3.3.5) Implicitní předpoklad, že nejkratší sled mezi v_i, v_j procházející přes v_t tak činí jen jednou.

(3.4.1) $w(xy) + p_v(y) \geq ||x, y|| + p_v(y) \geq p_v(x)$

(3.4.2) Pak (za samozřejmého předpokladu $p_v(v) = 0$) nutně bude upravená délka některé hrany na optimální cestě z x do v záporná. Neboli býti dolním odhadem je nutná podmínka pro přípustný potenciál, bohužel ale obecně nedostačující.

(3.5.1) Vlevo 3 a vpravo 2.

(3.5.2) Vlevo 3 a vpravo 5.

(3.5.3) 3. Ten prostřední vrchol má vzdálenost ≤ 2 do každého dalšího. Zbytek grafu je grafem krychle, kde největší vzdálenost 3 mají protilehlé dvojice vrcholů (ve smyslu geometrické krychle). Z těchto 4 dvojic má kratší spojení přes prostřední vrchol, takže zbývají 3 dvojice.

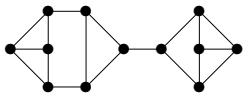
(3.5.4) 2 – jedná se o graf krychle, tak přidáme dvě úhlopříčky na jednu stěnu–čtverec. Pak budou vzdálenosti ≤ 2. Naopak jedna hrana nestačí, protože po přidání libovolné hrany e stále najdeme dva protilehlé vrcholy krychle, které hraně e nepatří, a tyto dva vrcholy zůstanou ve vzdálenosti 3.

(3.5.5) $2 + 3 + 2 = 7$

(3.5.6) 10. Vezmeme jeden vrchol v, ten má 3 sousedy a každý ze sousedů v má 2 další sousedy mimo v. To je dohromady 10 vrcholů a více jich už nemůže být, protože vzdálenosti jsou ≤ 2 od v. Nakreslete si takový graf!

(3.5.7) Vzdálenost 14 mezi 7 a 5. Dokážete zdůvodnit, že větší vzdálenost v grafu není?

(3.5.8)* Spočítejte si, kolik vrcholů může být do vzdálenosti 3...

(3.5.9) 

(3.5.10)* Ano, musí.

(3.5.11) Ano.

(3.5.12)* Osmi.

(3.5.13) Například zakázané odbočení na křižovatce může vynutit u optimální trasy prijetí křižovatky přímo a pak třeba návrat z opačného směru, ze kterého již odbočit lze. Je i spousta jiných příkladů.

(3.5.14)* Klíčovou myšlenkou je, že do úschovny nalezených vrcholů ke zpracování se nebudou ukládat jednotlivé vrcholy samotné, nýbrž spolu s vhodným sufixem sledu, kterým byly dosaženy. Jeden vrchol se tak v úschovně může vyskytnou mnohokrát v závislosti na tom, v jakých se nachází manévrech.

(4.1.1) V podstatě nijak, pouze touto smyčkou může cirkulovat jisté množství substance bez vlivu na velikost toku.

(4.1.2) Rozdílem těchto dvou toků je tzv. cirkulace, splňující zachování substance ve všech vrcholech včetně z, s.

(4.2.1) Tok 5.

(4.2.2) Jen zdroj z.

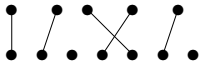
(4.2.3) Řez velikosti 4 oddělující dva vrcholy vpravo nahoře.

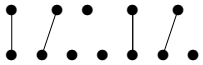
(4.2.4) Protože rezervy kapacit jsou celočíselné, neboli aspoň +1 v každé iteraci, takže v konečném počtu kroků se dostaneme k maximálnímu toku.

(4.2.5)* Myšlenka: Pokud se jednou nasycená hrana vrátí mezi nenasycené, bude to určitě na delší nenasycené cestě.

(4.3.1) Snadné...

(4.3.2)* Klíčové je dokázat, že získaný tok v první fázi po navýšení vytváří cirkulaci na původním grafu G (součty ve všech vrcholech 0).

(4.4.1) Velikosti 5: 

(4.4.2) Velikosti 4: 

(4.4.3) Ano, reprezentanti jsou zapsaní první: $(1, 2, 3)$, $(2, 1, 4)$, $(3, 1, 4)$, $(4, 2, 3)$.

(4.4.4) Ne, všech podmnožin je $\binom{5}{3} = 10$, ale prvků k reprezentaci jen 5.

(4.5.1) 12

(4.5.2) 13, 11 a 9, po řadě.

(4.5.3) Nemá – 6 z množin má sjednocení $\{3, 4, 5, 6, 7\}$, kde se najde jen 5 různých prvků pro reprezentanty.

(4.5.4) Nemá – 6 z množin má sjednocení $\{1, 2, 3, 4, 9\}$, kde se najde jen 5 různých prvků pro reprezentanty.

(4.5.5)* Stejně jako v Příkladě 4.18, jen je třeba zdvojit každý vrchol grafu (aby mohl reprezentovat dvě vycházející šipky).

(4.5.6)* Jedná se o další zajímavé použití toků v síti, kde hledaná procházka odpovídá vlastně minimálnímu řezu ve vhodně odvozeném grafu z G a oddělující kružnice jsou naopak získány z maximálního celočíselného toku.

(4.5.7) Viz http://en.wikipedia.org/wiki/Ford-Fulkerson_algorithm#Non-terminating_example

Reference

[1] R. Diestel, Graph Theory (third edition), Springer, 2005.
Online `http://diestel-graph-theory.com/`

[2] J.L. Gross, J. Yellen, eds. Handbook of Graph Theory, CRC Press, 2004.

[3] L. Kučera, Kombinatorické Algoritmy, SNTL, Praha, 1983.

[4] M. Mareš, Krajinou grafových algoritmů, ITI MFF UK, 2007.
`http://iti.mff.cuni.cz/series/files/iti330.pdf`.

[5] J. Matoušek a J. Nešetřil, Kapitoly z Diskrétní Matematiky. čtvrté vydání, Karolinum, UK Praha, 2009.

[6] J. Matoušek and J. Nešetřil, Invitation to Discrete Mathematics, second edition. Oxford University Press, 2008.

[7] B. Mohar, C. Thomassen, Graphs on Surfaces. Johns Hopkins, 2001.

[8] J. Oxley, Matroid Theory, Oxford University Press, 1992.