Teorie Grafů (MA010 "Grafy")

Doc. RNDr. Petr Hliněný, Ph.D.

hlineny@fi.muni.cz

22. prosince 2006

Verze 0.5.

Obsah

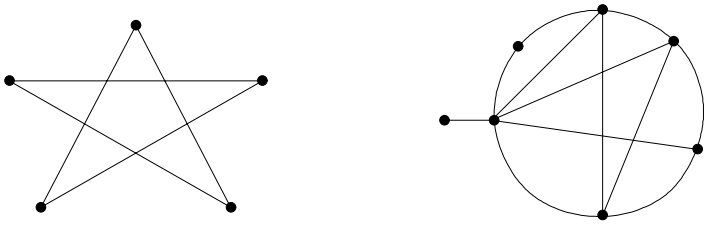
Část I
Základy Teorie Grafů

1 Pojem grafu

Úvod

Třebaže grafy jsou jen jednou z mnoha 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 asi nejvýznamnější součástí soudobé diskrétní matematiky. Znalost teorie grafů se jeví nezbytná ve většině oblastí moderní informatiky, včetně těch aplikovaných. Proto se náš základní kurz teorie grafů bude ve velké míře zaměřovat právě na jejich informatické aplikace (ale vcelku bez nutnosti informatiku ovládat, tj. naše látka bude přístupná i neinformatikům).

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



Takový 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 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). Také jsou zde uvedeny některé obvyklé typy grafů, jako cesty, kružnice, či úplné grafy. Především je však důležité, aby čtenář pochopil pojem isomorfismu grafů a dobře si jej procvičil v následujícím cvičení.

1.1 Definice grafu

Hned na úvod přistoupíme k formální definici grafu. Bude se jednat o základní definici tzv. obyčejného grafu; přičemž možné rozšiřující definice zmíníme krátce v závěru, ale stejně se budeme převážně zabývat obyčejnými grafy. V základní definici popisujeme hrany mezi dvojicemi vrcholů pomocí dvouprvkových podmnožin těchto vrcholů.

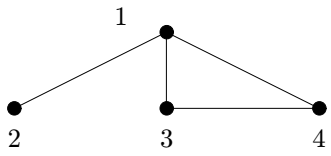
Definice 1.1. ***Graf*** (rozšířeně *obyčejný* či *jednoduchý neorientovaný* graf)
je uspořádaná dvojice $G = (V, E)$, kde V je množina *vrcholů* a E je množina *hran* – množina vybraných dvouprvkových podmnožin množiny vrcholů.

`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:

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

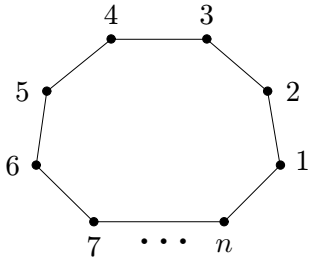


Běžné typy grafů

Pro snadnější vyjadřování je zvykem některé běžné typy grafů nazývat popisnými jmény.

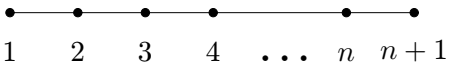
Kružnice délky n má $n \geq 3$ vrcholů spojených do jednoho cyklu n hranami:

C_n



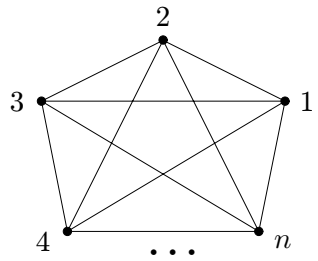
Cesta délky n má $n+1$ vrcholů spojených za sebou n hranami:

P_n



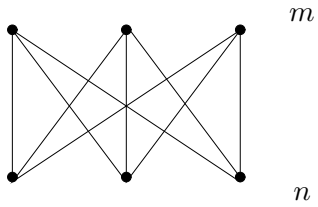
Úplný graf na $n \geq 1$ vrcholech má n vrcholů, všechny navzájem pospojované (tj. celkem $\binom{n}{2}$ hran):

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:

$K_{m,n}$



Úlohy k řešení

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

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

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

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

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

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

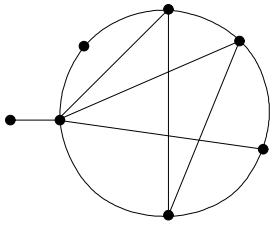
(1.1.7) *Má více hran úplný graf* K_9 *nebo úplný bipartitní graf* $K_{6,6}$*?*

1.2 Stupně vrcholů v grafu

Máme-li graf, často nás zajímá, kolik z kterého vrcholu vychází hran-spojnic. Proto je jedním z prvních definovaným pojmů stupeň vrcholu v grafu.

Definice 1.2. ***Stupněm vrcholu*** v v grafu G
rozumíme počet hran vycházejících z v. Stupeň 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 tento zakreslený graf má stupně vrcholů $1, 2, 3, 4, 4, 4, 6$.

`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 stupňů

Jelikož 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* $d_1 \leq d_2 \leq \ldots \leq d_n$ *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 použití 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.)

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.

□

Úlohy k řešení

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

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

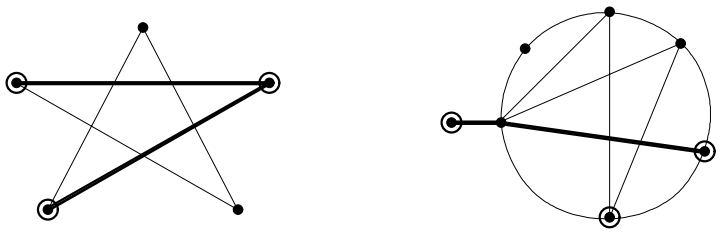
(1.2.3) *Existují dva různé grafy se* 6 *vrcholy stupňů* 2*?*

1.3 Podgrafy a Isomorfismus

Podgraf je, jednoduše řečeno, část grafu, ale toto musíme vyjádřit poněkud přesněji, abychom předešli situacím, kdy by nám zbyly hrany bez vrcholů. Navíc ještě definujeme indukovaný podgraf, který 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, co je (vlevo) a co není (vpravo) podgraf. (Zvýrazněny jsou vybrané podmnožiny vrcholů a hran.)



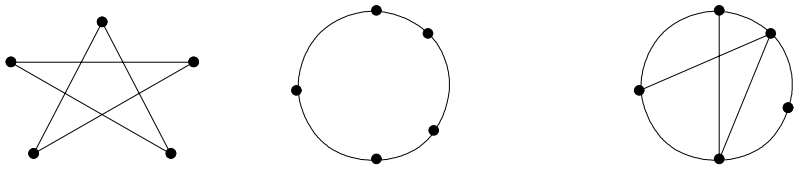
`Definice:` *Indukovaným podgrafem* je podgraf $H \subseteq G$ takový, který obsahuje všechny hrany grafu G mezi dvojicemi vrcholů z $V(H)$.

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*.

Pro správné pochopení a využití teorie grafů je tedy třeba nejprve dobře chápat pojem isomorfismu grafů.

Definice 1.6. ***Isomorfismus*** grafů G a H
je bijektivní (vzájemně jednoznačné) zobrazení $f : V(G) \to V(H)$, pro které platí, že každá 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).



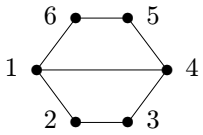
`Komentář:`
Z nakreslených tří grafů jsou první dva isomorfní – jsou to jen různá nakreslení "téhož" grafu, kružnice délky 5. (Určitě snadno najdete isomorfismus mezi nimi. Pro jednoduchost isomorfismus zakreslete do obrázku tak, že vrcholy prvního očíslujete čísly 1 až 5 a vrcholy druhého stejnými čísly v pořadí odpovídajícím jeho bijekci.) Třetí graf jim isomorfní není, neboť, například, má vrcholy jiných stupňů.

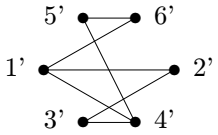
`Fakt`: Pokud je bijekce f isomorfismem, musí zobrazovat na sebe vrcholy stejných stupňů, tj. $d_G(v) = d_H(f(v))$. Naopak to však nestačí!

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 (taky stupně tři). vrchol označený 1 se zobrazí na $1'$. 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:





□

Abychom mohli s isomorfismem grafů přirozeně pracovat, je potřeba vědět následující fakt:

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

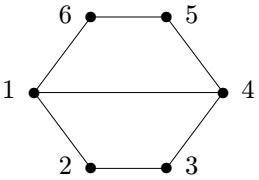
`Důkaz`. Relace $\simeq$ je reflexivní, protože graf je isomorfní sám sobě identickým zobrazením. Relace je také symetrická, neboť bijektivní zobrazení lze jednoznačně "obrátit". Tranzitivnost $\simeq$ se snadno dokáže skládáním zobrazení–isomorfismů. □

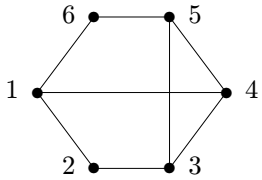
Důsledkem je, že všechny možné grafy se rozpadnou na *třídy isomorfismu*. V praxi pak, pokud mluvíme o *grafu*, myslíme tím obvykle jeho celou třídu isomorfismu, tj. nezáleží nám na konkrétní prezentaci grafu.

Další grafové pojmy

`Značení`: Mějme libovolný graf G.

- Podgrafu $H \subseteq G$, který je isomorfní nějaké kružnici, říkáme *kružnice v G*.
- Speciálně říkáme *trojúhelník* 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*.
- Podmnožině vrcholů $X \subseteq V(G)$, mezi kterými nevedou v G vůbec žádné hrany, říkáme *nezávislá množina X v G*.
- Indukovanému podgrafu $H \subseteq G$, který je isomorfní nějaké kružnici, říkáme *indukovaná kružnice v G*.



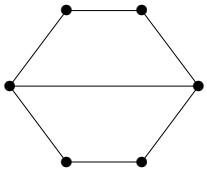


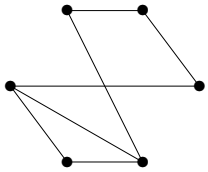
`Komentář:`

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 $2, 4, 6$.

Příklad 1.9. *Jsou následující dva grafy isomorfní?*



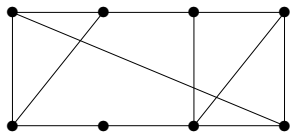


Postupovat budeme jako v Příkladě 1.7, nejprve ověříme, že oba grafy mají stejně mnoho vrcholů i stejnou posloupnost stupňů $2, 2, 2, 2, 3, 3$. Pokud se však budeme snažit najít mezi nimi isomorfismus, něco stále nebude sedět, že? Co nám tedy v nalezení isomorfismu brání? Podívejme se, že v druhém grafu 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. □

`Poznámka`: Výše uvedené příklady nám ukazují některé cesty, jak poznat (tj. najít nebo vyloučit) isomorfismus dvou grafů. Ty však ne vždy 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 má se za to, 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, $n!$)

Úlohy k řešení

(1.3.1) *Jaká je velikost největší nezávislé množiny v tomto grafu? (Tj. největší podmnožiny vrcholů, mezi kterými není žádná hrana.) Vyznačte tuto množinu.*



(1.3.2) *Jaká je délka nejkratší kružnice obsažené v grafu z Úlohy 1.3.1?*

(1.3.3) *Jaká je délka nejdelší cesty obsažené v grafu z Úlohy 1.3.1?*

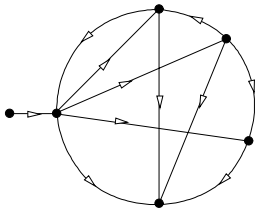
*(1.3.4)** *Jaká je délka nejdelší indukované cesty obsažené v grafu z Úlohy 1.3.1?*

(1.3.5) *Dokážete najít ke grafu z Úlohy 1.3.1 neisomorfní graf, který má stejnou posloupnost stupňů? Proč není isomorfní?*

(1.3.6) *Kolik existuje neisomorfních grafů s 5 vrcholy, všemi stupně 2?*

1.4 Orientované grafy a multigrafy

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



`Fakt`: Orientované grafy odpovídají relacím, které nemusí být symetrické.

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

`Poznámka`: Navíc někdy můžeme mluvit o tzv. *multigrafech*, ve kterých padají téměř všechna omezení na hrany...

V multigrafu může mezi dvojicí vrcholů vést libovolný počet hran – tzv. *násobné hrany*, a některé z nich mohou být i orientované. Také jsou povoleny hrany, které mají oba konce totožné – tzv. *smyčky*. Multigrafy zmiňujeme jen okrajově pro úplnost, nebudeme se jimi dále zabývat.

1.5 Implementace grafů

Mějme jednoduchý graf G na n vrcholech a značme vrcholy jednoduše čísly $V(G) = \{0, 1, \ldots, n-1\}$. Pro počítačovou implementaci grafu G se nabízejí dva základní způsoby:

- *Maticí sousednosti*, tj. dvourozměrným polem `g[][]`, ve kterém `g[i][j]=1` znamená hranu mezi vrcholy i a j.
- *Výčtem sousedů*, tj. použitím opět dvourozměrné pole `h[][]` a navíc pole `d[]` stupňů vrcholů. Zde prvky `h[i][0],h[i][1],...,h[i][d[i]-1]` udávají seznam sousedů vrcholu i.

`Poznámka`: Dávejte si pozor na symetrii hran v implementaci! To znamená, že pokud uložíte hranu `g[i][j]=1`, tak musíte zároveň uložit i hranu `g[j][i]=1`, jinak se dočkáte nepříjemných překvapení. Totéž se týká i seznamů sousedů.

`Komentář:` Implementace maticí sousednosti je hezká svou jednoduchostí. Druhá možnost se však mnohem lépe hodí pro grafy s malým počtem hran, což nastává ve většině praktických aplikací. (Navíc je implementace výčtem sousedů vhodná i pro multigrafy.)

Ke grafům lze do zvláštních polí přidat také *ohodnocení vrcholů a hran* libovolnými čísly či značkami...

Rozšiřující studium

Rozsáhlý úvod do teorie grafů je zahrnut v [5, Kapitola 3] a navazují na něj další části této učebnice.

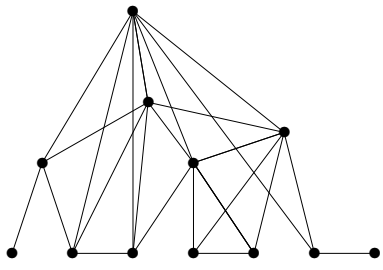
1.6 Cvičení: Příklady o grafech a isomorfismu

Příklad 1.10. *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).

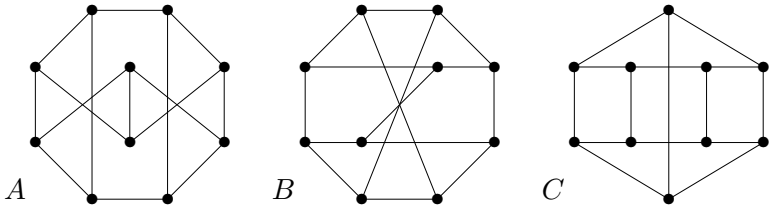
$1, 1, 3, 3, 3, 4, 4, 4, 6, 6, 6, 7 \quad \to \quad 1, 1, 3, 3, 2, 3, 3, 3, 5, 5, 5$
$1, 1, 2, 3, 3, 3, 3, 3, 5, 5, 5 \quad \to \quad 1, 1, 2, 3, 3, 2, 2, 2, 4, 4$
$1, 1, 2, 2, 2, 2, 3, 3, 4, 4 \quad \to \quad 1, 1, 2, 2, 2, 1, 2, 2, 3$
$1, 1, 1, 2, 2, 2, 2, 2, 3 \quad \to \quad 1, 1, 1, 2, 2, 1, 1, 1$
$1, 1, 1, 1, 1, 1, 2, 2 \quad \to \quad 1, 1, 1, 1, 1, 0, 1$
$0, 1, 1, 1, 1, 1, 1$ zřejmě tento graf existuje (3 samostatné hrany).

K poslední posloupnosti snadno sestrojíme graf, proto i graf s původní posloupností stupňů existuje. Zpětným přidáváním vrcholů sestrojíme i původní graf:



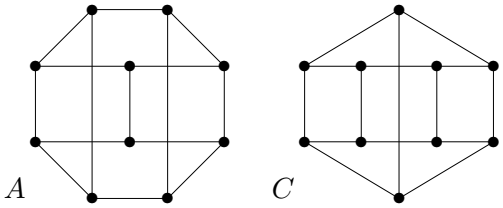
□

Příklad 1.11. *Najděte všechny isomorfní dvojice grafů v následujících obrázcích tří* 10*-vrcholových grafů. Isomorfní dvojice odpovídajícím způsobem očíslujte, u neisomorfních toto zdůvodněte.*



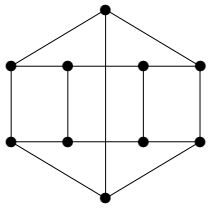
Postupujme systematicky: Všechny tři grafy mají po 10 vrcholech a všechny vrcholy stupňů 3. Takto jsme je tedy nijak nerozlišili. Podívejme se třeba na trojúhelníky v nich – opět si nepomůžeme, neboť žádný z nich trojúhelníky neobsahuje. Co se tedy podívat na obsažené kružnice C_4? Graf C jich jasně 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



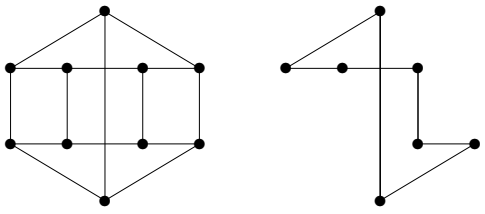
a odpovídající bijekce je na pohled zřejmá. (Doplňte si ji společným očíslováním vrcholů obou grafů.) □

Příklad 1.12. *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 být všech 5 vrcholů z jednoho pětiúhelníku, neboť to by uz byla kružnice délky jen 5.) Sami však snadným pokusem zjistíme, že v takové kružnici budou ještě hrany navíc, tudíž nebude indukovaná. □

Úlohy k řešení

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

Návod: Použijte Větu 1.4.

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

(1.6.3) *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 $5 - 3 = 2$.

(1.6.4) *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: $1, 1, 2, 2, 2, 3, 3, 4, 4, 4, 5, 6, 7$

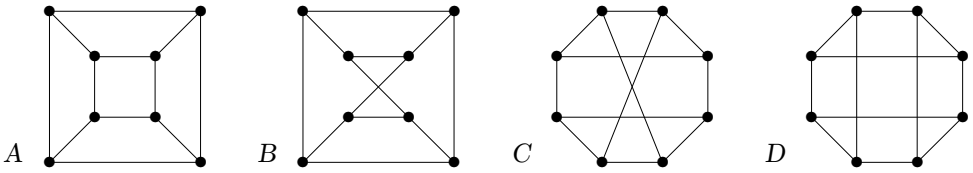
(1.6.5) *Najděte a vyznačte isomorfismus mezi následujícími dvěma grafy na* 7 *vrcholech.*

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

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

*(1.6.8)** *Existují dva neisomorfní grafy s posloupnostmi stupňů*
A: 3,3,3,3,3,3,
B: 2,2,3,3 ?
U možnosti, kde dva neisomorfní grafy existují, je nakreslete. Pokud neexistují, pokuste se to správně zdůvodnit. (Pamatujte, že uvažujeme jednoduché grafy, tj. grafy bez smyček a násobných hran. Grafy nemusí být souvislé.)

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



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

(1.6.10) *Jaká je nejdelší kružnice a nejdelší indukovaná kružnice obsažená v grafech A a B z Úlohy 1.6.9?*

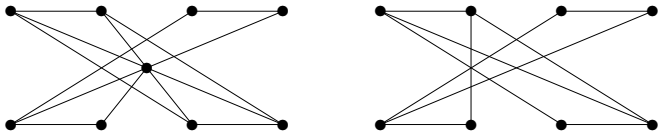
(1.6.11) *Kolik podgrafů úplného grafu K_{10} je isomorfních kružnici C_4?*

(1.6.12) *Najděte 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 zbudou 4 vrcholy stupňů 3, ale mezi nimi mohou být násobné hrany nebo i smyčky. Kreslete si obrázky.

(1.6.13) *Vraťte se zpět k příkladům a úlohám z Oddílu 1.3 a zkuste, zda je se znalostí nových metod jste schopni řešit lépe a elegantněji.*

(1.6.14) *Kolik nejvíce vrcholů obsahuje nezávislá množina v nakreslených grafech? Tuto největší nezávislou množinu si v grafech vyznačte.*

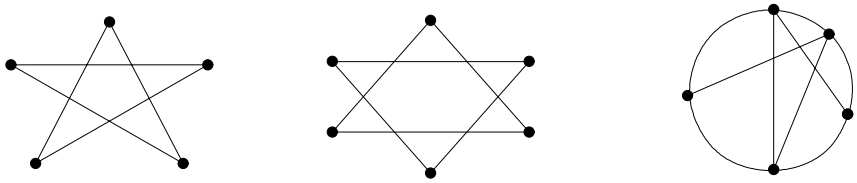


2 Souvislost grafů

Úvod

Pokud máme graf, který modeluje nějaká spojení či síť, přirozeně nás zajímá, jakou máme možnost se dostat odněkud někam v tomto grafu. To má množství praktických motivací – například počítačové, dopravní, telefonní či potrubní sítě. Je pochopitelné, že v takových sítích chceme mít možnost se dostat z každého místa do každého jiného.

Grafům s takovou vlastností říkáme souvislé. (Abychom si ujasnili terminologii, když mluvíme o souvislosti, nejedná se o žádné "hledání souvislostí grafů s něčím jiným", ale o možnost procházení mezi vrcholy jednoho grafu po jeho hranách.) Pro ukázku se podívejme na následující obrázky tří grafů:



Poznáte, který z nich je nesouvislý? Všechny tři vypadají "spojeně", ale podívejte se důkladně na ten prostřední – v něm není žádné propojení mezi vrchním a spodním vrcholem po hranách (křížení se nepočítají).

Další potřebnou znalostí je umět souvislý graf algoritmicky celý procházet. Zde si uvedeme obecnou kostru algoritmu pro procházení grafu a zmíníme některé konkrétní variace tohoto algoritmu. (Znáte algoritmus pro procházení bludiště? V grafech je to v obecnosti podobné.) Také se později zmíníme i o vyšších stupních souvislosti grafu – o zálohování spojení v grafech pro případy výpadku.

Cíle

Tato lekce definuje pojmy souvislosti a komponent grafu. Cílem je ukázat, jak se souvislostí grafu pracovat a jak algoritmicky graf procházet po jeho hranách. Zmíněny jsou i vyšší stupně souvislosti.

2.1 Spojení vrcholů, komponenty

Nejprve bychom si měli přesně ujasnit, jak se pohybujeme grafem, tedy co je vlastně procházkou v grafu. Tento pojem by měl postihnout základní věc, že v grafu procházíme hranami vždy z vrcholu do sousedního vrcholu, a přitom ponechat dostatek volnosti pro vracení se a zacyklení procházek.

`Definice`: *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 vždy hrana e_i má koncové vrcholy v_{i-1}, v_i.

`Komentář:` Sled je vlastně procházka po hranách grafu z u do v. Příkladem sledu může být průchod IP paketu internetem (včetně cyklení).

Lema 2.1. *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 můžeme na sebe navázat v jeden. □

`Definice`: Třídy ekvivalence výše popsané (Lema 2.1) relace $\sim$ na $V(G)$ se nazývají *komponenty souvislosti* grafu G.

Jinak se taky *komponentami souvislosti myslí podgrafy* indukované na těchto třídách ekvivalence.

Připomeňme si, že *cesta v grafu* je vlastně sledem bez opakování vrcholů.

Věta 2.2. *Pokud mezi dvěma vrcholy grafu G existuje sled, pak mezi nimi existuje cesta.*

`Důkaz`. Nechť $u = v_0, e_1, v_1, \ldots, e_n, v_n = v$ je sled délky n mezi vrcholy u a v v G. Začneme budovat nový sled W z vrcholu $w_0 = u$, který už bude cestou:

- Předpokládejme, že nový sled W už má počátek $w_0, e_1, w_1, \ldots, w_i$ (na začátku $i = 0$, tj. jen w_0 bez hran), kde $w_i = v_j$ pro některé $j \in \{0, 1, \ldots, n\}$.
- Najdeme největší index $k \geq j$ takový, že $v_k = v_j = w_i$, a sled W pokračujeme krokem $\ldots, w_i = v_j = v_k,\ e_{k+1},\ w_{i+1} = v_{k+1}, \ldots$.
- Zbývá dokázat, že nový vrchol $w_{i+1} = v_{k+1}$ se ve sledu W neopakuje. Pokud by tomu ale tak bylo $w_l = w_{i+1}$, $l \leq i$, pak bychom na vrchol w_{i+1} "přeskočili" už dříve z vrcholu w_l, spor.
- Nakonec skončíme, když $w_i = v$.

□

`Komentář:` Ačkoliv uvedený důkaz vypadá složitě, je to jen jeho formálním zápisem. Ve skutečnosti se v důkaze neděje nic jiného, než že se původní sled zkracuje o opakované vrcholy, až nakonec zákonitě vznikne cesta.

`Důkaz` kratší, ale nekonstruktivní, pro Větu 2.2.

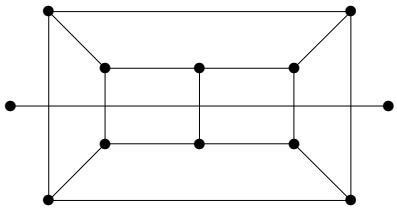
Ze všech sledů mezi vrcholy u a v v G vybereme sled W s nejmenší délkou. Je snadno vidět, že pokud W zopakuje některý vrchol grafu G, můžeme W ještě zkrátit, a to je spor s předpokladem. Proto je W cestou v G. □

Závěrem se dostáváme k nejdůležitější definici souvislého grafu:

Definice 2.3. ***Graf G je souvislý***
pokud je G tvořený jedinou komponentou souvislosti, tj. pokud každé dva vrcholy G jsou spojené cestou (dle Věty 2.2).

`Poznámka`: Prázdný graf je souvislý.

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

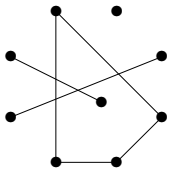


Vidíte obě dvě komponenty?

Úlohy k řešení

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

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



2.2 Prohledávání grafu

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), tento globální pohled nemáme k dispozici a graf musíme *prohledávat lokálně*. Z toho důvodu se potřebujeme seznámit, jako vůbec s prvním grafovým algoritmem, s obecným postupem *lokálního prohledávání grafu*. Tento algoritmus není složitý a víceméně zobecňuje dobře známý postup procházení všech cest v bludišti.

Pro vytvoření co nejobecnějšího schématu *algoritmu pro procházení grafu* vystačíme s následujícími datovými stavy a pomocnou strukturou:

- *Vrchol*: má stavy ...
  - iniciační – dostane na začátku,
  - nalezený – poté, co jsme jej přes některou hranu nalezli,

  - zpracovaný – poté, co jsme už probrali všechny hrany z něj vycházející.

- *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 nalezené a ještě nezpracované vrcholy.

`Poznámka`: Způsob, kterým se vybírají vrcholy z úschovny ke zpracování, určuje variantu algoritmu procházení grafu. V prohledávaných vrcholech a hranách se pak provádějí konkrétní programové akce pro prohledání a zpracování našeho grafu.

Algoritmus 2.4. ***Procházení souvislých komponent grafu***
Algoritmus projde a zpracuje každou hranu a vrchol grafu G. Průchod se děje po jednotlivých komponentách, po projití jedné komponenty se přejde na případnou další. Konkrétní programové akce potřebné ke zpracování grafu jsou uvedeny v pomocných funkcích `ZPRACUJ()`.

```
vstup < graf G;
stav(všechny vrcholy a hrany G) = iniciační;
uschovna U = {libovolný vrchol v0 grafu G};
stav(v0) = nalezený;
// zpracování vybrané komponenty G
while (U je neprázdná) {
    vybrat v ∈ U;   U = U\{v};
    ZPRACUJ(v);
    for (e hrany vycházející z v) {
        if (stav(e)==iniciační) ZPRACUJ(e);
        w = druhý vrchol e = vw;
        if (stav(w)==iniciační) {
            stav(w) = nalezený;
            U = U∪{w};
        }
        stav(e) = zpracovaná;
    }
    stav(v) = zpracovaný;
    // případný přechod na další komponentu G
    if (U je prázdná && G má další komponenty)
        U = {vrchol v1 další komponenty G};
}
```

Příklady aplikací schématu Algoritmu 2.4 jsou uvedeny dále například v Důsledku 3.4 a v Algoritmu 3.11. Konkrétní postup algoritmu procházení bude ilustrován ve Cvičení 2.4.

Způsoby implementace procházení grafu

- *Procházení "do hloubky"* – úschovna `U` je implementovaná jako zásobník, tj. dále prohledáváme od posledních nalezených vrcholů.
- *Procházení "do šířky"* – úschovna `U` je implementovaná jako fronta, tj. dále prohledáváme od prvních nalezených vrcholů.

- *Dijkstrův algoritmus* pro nejkratší cestu – z úschovny vybíráme vždy vrchol nejbližší k počátečnímu v_0. (Toto je dost podobné prohledávání do šířky, ale obecnější i pro případy, kdy hrany nejsou "stejně dlouhé".)
  Tento algoritmus bude popsán v příští lekci.

2.3 Vyšší stupně souvislosti

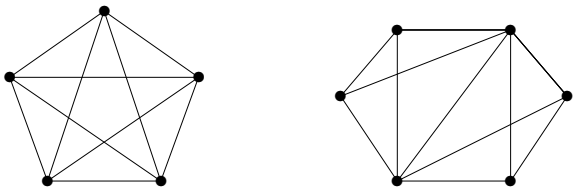
V *síťových aplikacích* nás často zajímá nejen, jestli se za normálních podmínek můžeme pohybovat mezi vrcholy/uzly, ale také, jaké spojení můžeme nalézt v případě lokálních výpadků (odolnost a redundance). Toto lze teoreticky podchytit zkoumáním "vyšších" stupňů souvislosti grafu.

`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 i po odebrání libovolných nejvýše $k-1$ vrcholů z G zůstane výsledný graf souvislý.
Speciálně úplný graf K_n je vrcholově $(n-1)$-souvislý.

Pokud mluvíme jen o k-souvislém grafu, máme (obvykle) na mysli vrcholově k-souvislý graf.

`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 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?)

Mengerova věta

Důkaz následujícího důležitého výsledku by nebyl jednoduchý při použití stávajících znalostí, proto jej ponecháme na pozdější Lekci **??**.

Věta 2.5. *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 prvního grafu 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 2.6. *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 2.5 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. □

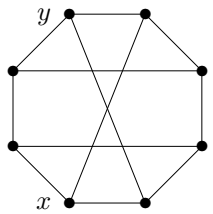
Rozšířením předchozí úvahy lze dokonce dokázat:

Věta 2.7. *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 zcela disjunktních cest z vrcholů U_1 do vrcholů U_2.*

Úlohy k řešení

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

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

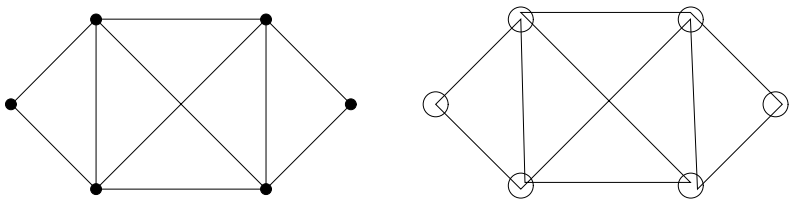


(2.3.3) *Kolik nejméně hran musíte přidat k cestě délky 7, aby vznikl vrcholově 2-souvislý graf?*

2.4 Jedním tahem: Eulerovské grafy

Především z historických důvodů zařazujeme na závěr tuto poznámku o kreslení grafů jedním tahem.

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



Nejstarší výsledek teorie grafů vůbec pochází od Leonarda Eulera: (Jedná se o problém slavných 7 mostů v Královci / Königsbergu / dnešním Kaliningradě.)

Věta 2.8. *Graf G lze nakreslit jedním uzavřeným tahem právě když G je souvislý a všechny vrcholy v G jsou sudého stupně.*

Důsledek 2.9. *Graf G lze nakreslit jedním otevřeným tahem právě když G je souvislý a všechny vrcholy v G až na dva jsou sudého stupně.*

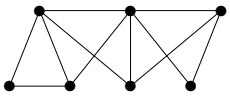
`Důkaz`: Dokazujeme oba směry ekvivalence.

- 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 $G' = G - E(T)$, o kterém předpokládejme, že je neprázdný. Jelikož G' má taktéž všechny stupně sudé, je (z indukčního předpokladu) libovolná jeho komponenta $C \subseteq G'$ nakreslená jedním uzavřeným tahem T_C.
- Vzhledem k souvislosti grafu G existuje vrchol w incidentní zároveň s hranami našeho tahu T i vhodné komponenty $C \subseteq G'$; tudíž lze oba tahy T a T_C „spojit přes w“. To je spor s 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. □

Úlohy k řešení



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

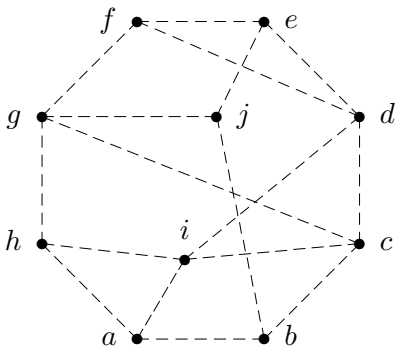
(2.4.2) *Kolik hran musíte přidat ke grafu z předchozí otázky, aby se dal nakreslit jedním uzavřeným tahem?*

Rozšiřující studium

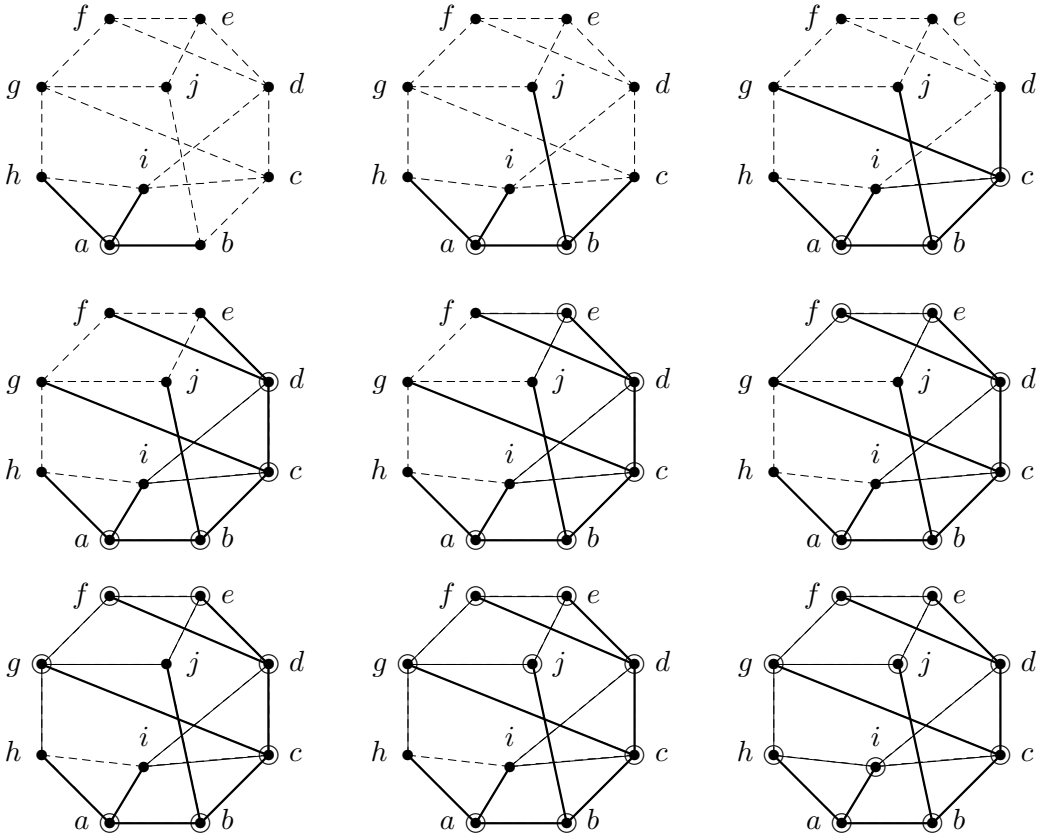
Souvislosti grafů se blíže teoreticky věnují [5, Oddíly 3.2,8]. Algoritmy pro procházení grafu jsou podrobně popsány (včetně netriviálních aplikací) v [3] a demonstrovány i v [4].

2.5 Cvičení: Příklady na souvislost grafů

Příklad 2.10. *Ukázka průchodu následujícím grafem do hloubky z vrcholu a.*

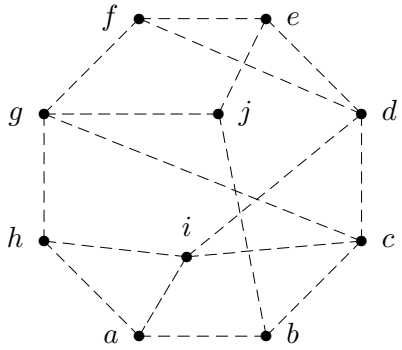


V této ukázce budeme obrázky znázorňovat stavy Algoritmu 2.4 v jednotlivých krocích takto: Neprohledané hrany jsou čárkované, prohledané hrany plnou čarou a hrany, které vedly k nalezení vrcholů, jsou tlustou čarou (tyto hrany často mívají speciální význam v aplikacích schématu algoritmu). Nalezené vrcholy se poznají podle příchozí tlusté hrany a zpracované vrcholy jsou značené dvojím kroužkem.

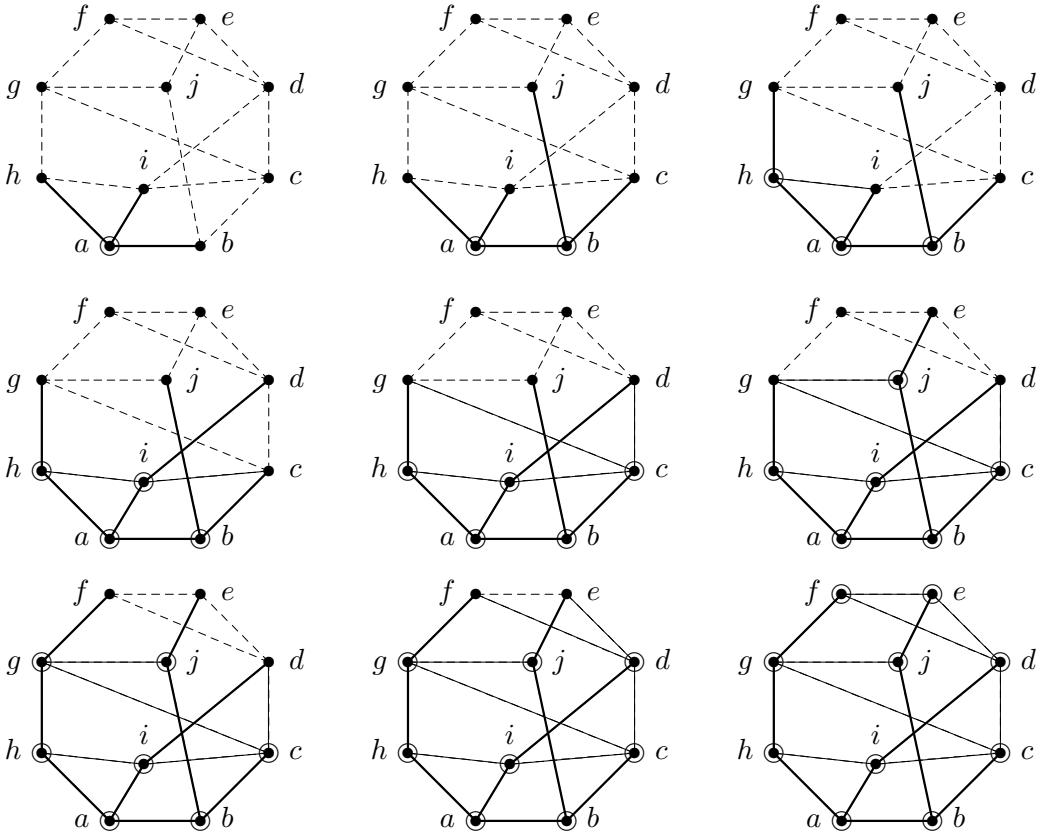


Tímto zpracování zadaného grafu skončilo. Mimo jiné jsme zjistili, že graf má jedinou komponentu souvislosti. □

Příklad 2.11. *Ukázka průchodu následujícím grafem do šířky z vrcholu a.*



V této ukázce budeme obrázky znázorňovat stavy Algoritmu 2.4 stejně jako v předchozím příkladě.



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

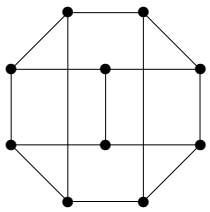
Příklad 2.12. *Mějme graf H_3, jehož vrcholy jsou všechny podmnožiny množiny $\{1,2,3\}$ a hrany spojují právě disjunktní dvojice podmnožin. (Tj. H_3 má 8 vrcholů.) Rozhodněte, zda je H_3 souvislý graf, a napište, kolik má H_3 hran.*

Vrchol $\emptyset$ odpovídající prázdné množině je spojený se všemi sedmi ostatními vrcholy, takže je graf 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 $\frac{1}{2}(7 + 3 \cdot 2^2 + 3 \cdot 2^1 + 2^0) = 13$ hran. □

Úlohy k řešení

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



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

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

(2.5.4) *Kolik existuje neisomorfních grafů s 9 vrcholy, všemi stupně 2? Nakreslete je všechny a nezapomeňte na ty nesouvislé.*

(2.5.5) *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.*

(2.5.6) *Kolik nejméně hran musí mít graf na 12 vrcholech, aby stupeň jeho souvislosti byl 3?*

(2.5.7) *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.

(2.5.8) *Existují dva neisomorfní grafy se stejnou posloupností stupňů*
A: 2,2,2,3,3,
B: 2,3,3,3,3,
C: 2,2,2,1,1 ?
U možnosti, kde dva neisomorfní grafy existují, je nakreslete. U zbylé možnosti pak správně zdůvodněte, proč dva takové neisomorfní grafy neexistují. (Pamatujte, že uvažujeme jednoduché grafy, tj. grafy bez smyček a násobných hran. Grafy nemusí být souvislé.)

(2.5.9) *Existují dva neisomorfní grafy se stejnou posloupností stupňů*
A: 2,2,2,2,2,
B: 1,1,3,3,3,3 ?

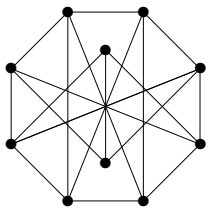
(2.5.10) *Kolik existuje neisomorfních grafů na 6 vrcholech s posloupností stupňů*

$$1,1,2,2,2,2\,?$$

Všechny tyto grafy nakreslete a zdůvodněte i, proč jich není více. (Pamatujte, že uvažujeme jednoduché grafy, tj. grafy bez smyček a násobných hran. Grafy nemusí být souvislé.)

(2.5.11) *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.*

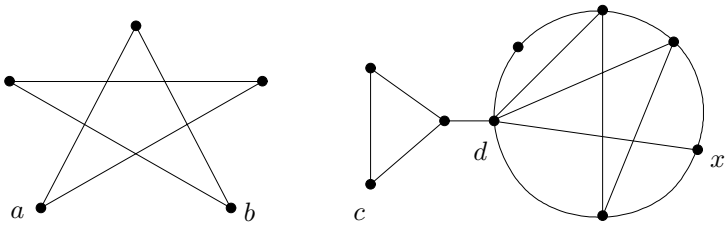
(2.5.12) *Lze tento graf nakreslit jedním tahem? A uzavřeným? (Uprostřed není vrchol, jen se tam kříží hrany.)*



3 Vzdálenost a metrika 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.

V obecném případě však při určování vzdálenosti bereme do úvahy délky jednotlivých hran podél cesty (tyto délky musí být nezáporné!). Jako hlavní náplň lekce si uvedeme známý Dijkstrův algoritmus pro určování nejkratší cesty v grafu. (Tento algoritmus je, mimo jiné aplikace, také základem programů vyhledávajících vlaková/autobusová spojení.) Většina tvrzení a algoritmy obsažené v této přednášce je beze změny aplikovatelná i na orientované grafy (tj. když nás zajímá i směr procházení hran).

Cíle

Prvním cílem této lekce je definovat vzdálenost v grafech a její základní vlastnosti. Dalším úkolem je ukázat a správně pochopit Dijkstrův algoritmus pro hledání nejkratší cesty v grafu.

3.1 Vzdálenost v grafu

Vzpomeňme 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 2.2.

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

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

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. □

Zjištění vzdálenosti

Věta 3.3. *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 $d_G(u,v') = d - 1$. Stejně tak značme w' souseda vrcholu w bližšího k u, tedy $d_G(u,w') > d_G(u,v')$. Potom byl vrchol v' nalezen v prohledávání do šířky dříve než vrchol w' podle indukčního předpokladu. To znamená, že v' se dostal do fronty úschovny dříve než w', a tudíž sousedé v' (mezi nima v) budou při prohledávání nalezeni dříve než sousedé w'. □

Důsledek 3.4. *Základní algoritmus procházení grafu do šířky lze použít pro výpočet vzdálenosti z vrcholu u:*

Toto je poměrně jednoduchá aplikace, kdy počátečnímu vrcholu přiřadíme vzdálenost 0, a pak vždy každému dalšímu nalezenému vrcholu přiřadíme vzdálenost o 1 větší než byla vzdálenost vrcholu, ze kterého jsme jej nalezli.

`Komentář:` My si ale dále ukážeme obecnější Dijkstrův algoritmus, který počítá nejkratší vzdálenost při libovolně kladně ohodnocených délkách hran.

Další pojmy a fakta

`Definice`. 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 $\mathrm{rad}(G)$.
- *Steinerova vzdálenost* mezi vrcholy libovolné podmnožiny $W \subseteq V(G)$ je rovna minimálnímu počtu hran souvislého podgrafu v G obsahujícího všechny vrcholy W.

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.

Jedna zajímavost

Pro zajímavost si uveďme následující vlastnost grafů.

`Definice`: Graf je *vzdálenostně dědičný* (distance-hereditary), pokud vzdálenost každé dvojice jeho vrcholů u, v je stejná ve všech indukovaných souvislých podgrafech obsahujících u, v.

`Fakt`. Grafy bez kružnic jsou vzdálenostně dědičné.

`Fakt`. Následující konstrukce vytváří vzdálenostně dědičné grafy:

- Začněme z jediného vrcholu.
- Mějme dva grafy G_1, G_2 získané touto konstrukcí (rekurzivně). Vytvoříme jejich disjunktní sjednocení $G_1 \dot\cup G_2$.
- Mějme dva grafy G_1, G_2 získané touto konstrukcí (rekurzivně). Vytvoříme jejich disjunktní sjednocení $G_1 \dot\cup G_2$ a přidáme všechny hrany mezi $V(G_1)$ a $V(G_2)$.

(Najděte vzdálenostně dědičný graf, který nelze takto sestrojit...)

Věta 3.5. *Graf G je vzdálenostně dědičný právě když každá kružnice délky alespoň 5 má v G dvě „zkřížené“ tětivy.*

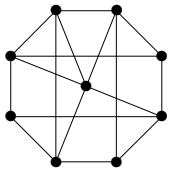
Úlohy k řešení

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

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

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

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



3.2 Výpočet metriky

Než se podíváme na samotný postup výpočtu vzdálenosti z jednoho vrcholu do druhého, uvedeme si ještě “globální” pohled na soubor všech vzdáleností v grafu, tedy na tzv. metriku grafu. Zajímavostí tohoto pohledu je především to, že výpočet celé metriky najednou má velice jednoduchý algoritmus, který není příliš známý a přitom je dobře použitelný i v jiných oblastech. (Jak třeba později uvidíte v teorii automatů.)

`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 3.6. ***Iterativní výpočet metriky skládáním cest***

- Na počátku nechť `d[i,j]` udává 1 (případně délku hrany $\{i,j\}$), nebo ∞ pokud hrana mezi i,j není.
- Po každém kroku $t \geq 0$ nechť `d[i,j]` udává délku nejkratší cesty mezi i,j, která jde pouze přes vnitřní vrcholy z množiny $\{0,1,2,\ldots,t-1\}$.
- Při přechodu z t na následující krok $t+1$ upravujeme vzdálenost pro každou dvojici vrcholů – jsou vždy pouhé dvě možnosti:
  - Buď je cesta délky `d[i,j]` z předchozího kroku stále nejlepší (tj. nově povolený vrchol t nám nepomůže),
  - nebo cestu vylepšíme spojením přes nově povolený vrchol t, čímž získáme menší vzdálenost `d[i,t]+d[t,j]`. (Nakreslete si obrázek!)

`Poznámka`: V praktické implementaci pro symbol ∞ použijeme velkou konstantu, třeba `MAX_INT/2`. (Nelze použít přímo `MAX_INT`, neboť by pak došlo k aritmetickému přetečení.)

Algoritmus 3.7. ***Výpočet metriky grafu***
Tento algoritmus, pro graf s N vrcholy daný maticí sousednosti G, vypočte celou jeho metriku d.

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

```
for (i=0; i<N; i++) for (j=0; j<N; j++)
    d[i,j] = (i==j?0: (G[i,j]? 1: MAX_INT/2));
for (t=0; t<N; t++) {
    for (i=0; i<N; i++) for (j=0; j<N; j++)
        d[i,j] = min(d[i,j], d[i,t]+d[t,j]);
}
```

`Poznámka`: Algoritmus 3.7 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...

Věta 3.8. *Algoritmus 3.7 v poli* `d[i,j]` *správně vypočte vzdálenost mezi vrcholy* i, j.

Úlohy k řešení

(3.2.1) *Vypočtěte dle Algoritmu 3.7 metriku kružnice* C_4.

3.3 Vážená (ohodnocená) vzdálenost

V dalším textu se již budeme věnovat cestám v grafech s obecně "dlouhými" hranami.

Definice 3.9. ***Vážený graf*** je graf 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`: Mějme (kladně) vážený graf G, w. Délkou váženého sledu $S = v_0, e_1, v_1, e_2, v_2, \ldots, e_n, v_n$ v G myslíme součet

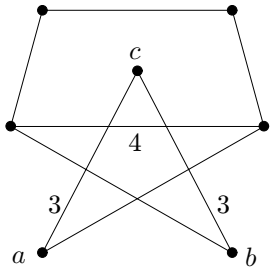
$$d^w_G(S) = w(e_1) + w(e_2) + \ldots + w(e_n)\,.$$

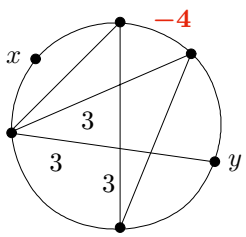
Váženou vzdáleností v G, w mezi dvěma vrcholy u, v pak myslíme

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

Lema 3.10. *Vážená vzdálenost v kladně vážených grafech také splňuje trojúhelníkovou nerovnost.*

`Komentář:` Podívejme se na následující graf vlevo. (Čísla u hran udávají jejich váhy, hrany bez čísel mají váhu 1.) Vážená vzdálenost mezi vrcholy a, c je 3, mezi b, c také 3. Jaká je vzdálenost mezi vrcholy a, b? Ne, není to 6, ale najdeme kratší cestu délky 5.





V druhém příkladě vpravo je uvedena i hrana se zápornou vahou -4. Nejkratší cesta mezi vrcholy x, y tak má délku -2, ale pokud vezmeme sled, který hranu váhy -4 vícekrát

zopakuje, dostaneme se na libovolně nízkou zápornou délku. To je samozřejmě nesmyslné, a proto se takovému problému radši vyhýbáme zákazem záporných vah hran.

Úlohy k řešení

(3.3.1) *Jaká je nejdelší vzdálenost mezi dvěma vrcholy v předchozím obrázku vlevo?*

(3.3.2) *O kterém vrcholu v předchozím obrázku vlevo 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?*

3.4 Hledání nejkratší cesty

Pro nalezení nejkratší (vážené) cesty mezi dvěma vrcholy kladně váženého grafu se používá známý tzv. *Dijkstrův algoritmus*.

`Poznámka`: Dijkstrův algoritmus je sice složitější než Algoritmus 3.7, ale na druhou stranu je *výrazně rychlejší*, pokud nás zajímá jen nejkratší vzdálenost z jednoho vrcholu místo všech dvojic vrcholů.
Zrovna tento algoritmus se například používá při vyhledávání vlakových či autobusových spojení. Pravděpodobně se i vy někdy dostanete do situace, kdy budete nejkratší cestu hledat, proto si tento algoritmus zapamatujte.

Dijkstrův 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) – do takového vrcholu se už lépe dostat nemůžeme, protože všechny jiné cesty by byly dle výběru delší.
- Na konci zpracování tyto proměnné vzdálenosti udávají správně nejkratší vzdálenosti z počátečního vrcholu do ostatních.

Algoritmus 3.11. ***Dijkstrův pro nejkratší cestu v grafu***
Tento algoritmus nalezne nejkratší cestu mezi vrcholy u a v kladně váženého grafu G, daného seznamem sousedů vrcholů.

`vstup:` *graf na* `N` *vrcholech daný seznamem sousedů* `sous[][]` *a* `del[][]`,
kde `sous[i][0],...,sous[i][st[i]-1]` *jsou sousedé vrcholu* `i` *stupně* `st[i]`
a hrana z `i` *do* `sous[i][k]` *má délku* `del[i][k]>0`;
`vstup:` `u,v`, *kde hledáme cestu z* `u` *do* `v`;

```
// stav[i] udává zpracovanost vrcholu, vzdal[i] zatím nalezenou vzdálenost
for (i=0; i<=N; i++) { vzdal[i] = MAX_INT; stav[i] = 0; }
vzdal[u] = 0;
while (stav[v]==0) {
    for (i=0, j=N; i<N; i++)
        if (stav[i]==0 && vzdal[i]<vzdal[j]) j = i;
    // zde jsme našli nejbližší nezpracovaný vrchol j, ten teď zpracujeme
    if (vzdal[j]==MAX_INT) return NENI_CESTA;
    stav[j] = 1;
    for (k=0; k<st[j]; k++)
        if (vzdal[j]+del[j][k]<vzdal[sous[j][k]]) {
```

```
            pri[sous[j][k]] = j;
            vzdal[sous[j][k]] = vzdal[j]+del[j][k];
        }
        // pole pri[.] uchovává, odkud jsme se do kterého vrcholu dostali
}
výstup 'Cesta délky vzdal[v], uložená "pozpátku" v poli pri[]';
```

`Poznámka`: Uvědomme si, že pokud necháme tento algoritmus proběhnout až do zpracování všech vrcholů, získáme v `vzdal[i]` nejkratší vzdálenosti z počátečního vrcholu do všech ostatních vrcholů. Všimněme si dále, že Algoritmus 3.11 počítá stejně dobře nejkratší cestu i v orientovaném grafu.

`Důkaz` správnosti Algoritmu 3.11.
Klíčovým faktem k důkazu správnosti algoritmu je to, že v každém kroku (počínaje stavem po prvním průchodu cyklem `while()`) proměnná `vzdal[i]` udává nejkratší vzdálenost z vrcholu `u` do vrcholu `i` při cestě pouze po vnitřních vrcholech x, jejichž `stav[x]==1` (po zpracovaných vrcholech). Stručně matematickou *indukcí*:

- V prvním kroku algoritmu je jako vrchol ke zpracování vybrán první `j=u` a potom jsou jeho sousedům upraveny vzdálenosti od `u` podle délek hran z `u`.
- V každém dalším kroku je vybrán jako vrchol `j` ke zpracování ten, který má ze všech nezpracovaných vrcholů nejkratší nalezenou vzdálenost od počátku `u`. To ale znamená, že žádná kratší cesta už do `j` nevede, neboť každá „oklika“ přes jiné nezpracované vrcholy musí být delší dle výběru `j`. (V tomto bodě potřebujeme nezápornost ohodnocení `del[][]`.) Naopak můžeme podle délek hran vycházejících z `j` „vylepšit“ cesty do jeho sousedů.

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

`Fakt`: Celkový počet kroků potřebný v Algoritmu 3.11 k nalezení nejkratší cesty z `u` do `v` je zhruba N^2, kde N je počet vrcholů grafu. Na druhou stranu, při lepší implementaci úschovny nezpracovaných vrcholů (třeba *haldou* s nalezenou vzdáleností jako klíčem) lze dosáhnout i mnohem rychlejšího běhu tohoto algoritmu na řídkých grafech – času zhruba úměrného počtu hran grafu.

`Komentář:` Vysvětleme si nyní podrobně srovnání obou našich algoritmů. Algoritmus 3.7 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.

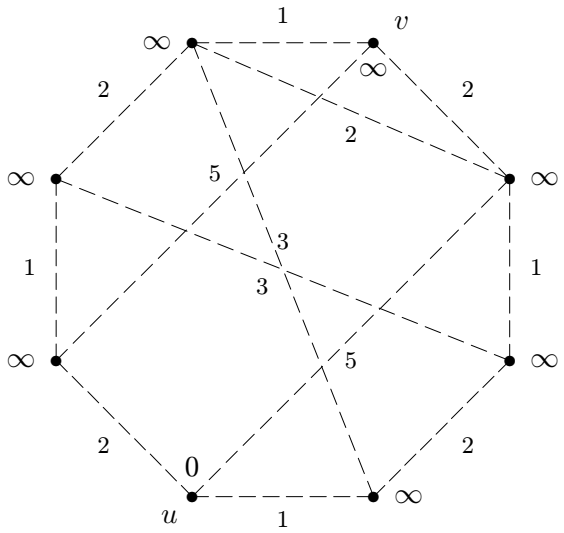
Dijkstrův algoritmus na druhou stranu počítá mnohem rychleji, pokud nás zajímá jen vzdálenost z jednoho vrcholu (zhruba N^2 nebo i méně kroků), a dokonce běží ještě rychleji, pokud nás zajímá jen vzdálenost dvou blízkých vrcholů – tehdy totiž můžeme prohledávání ukončit i jen na malé části celého grafu. To jsou přímo ideální vlastnosti, které bychom od takového algoritmu očekávali.

Rozšiřující studium

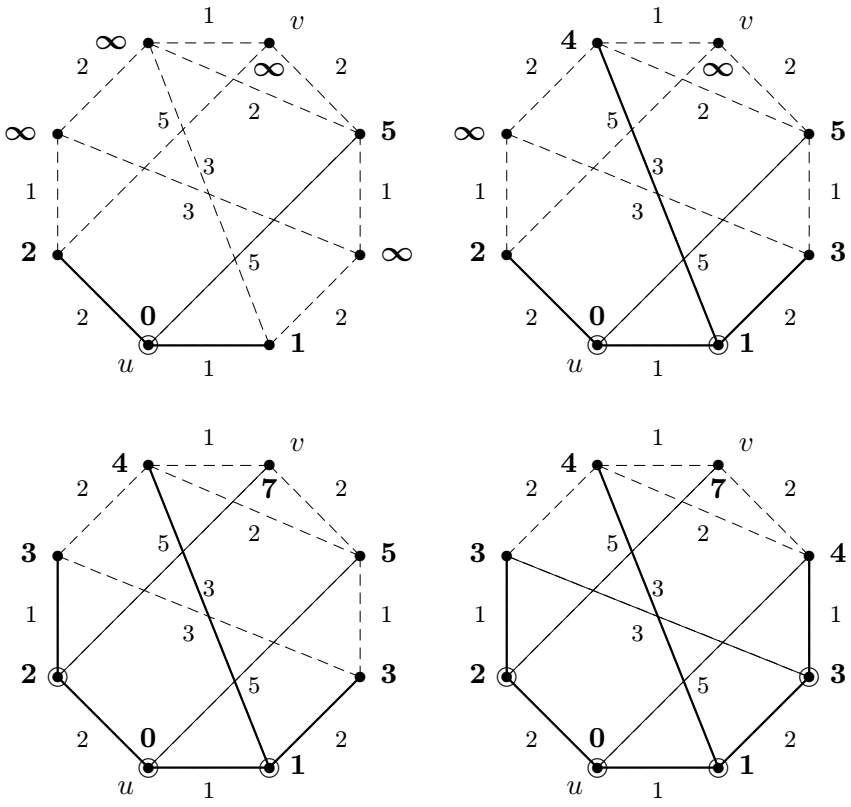
Vzdálenostem v grafech se více věnují [5, Oddíly 3.2,3]. Dijkstrův algoritmus je velmi oblíbeným tématem mnoha učebnic programování, a proto jej není třeba nijak zvlášť hledat.

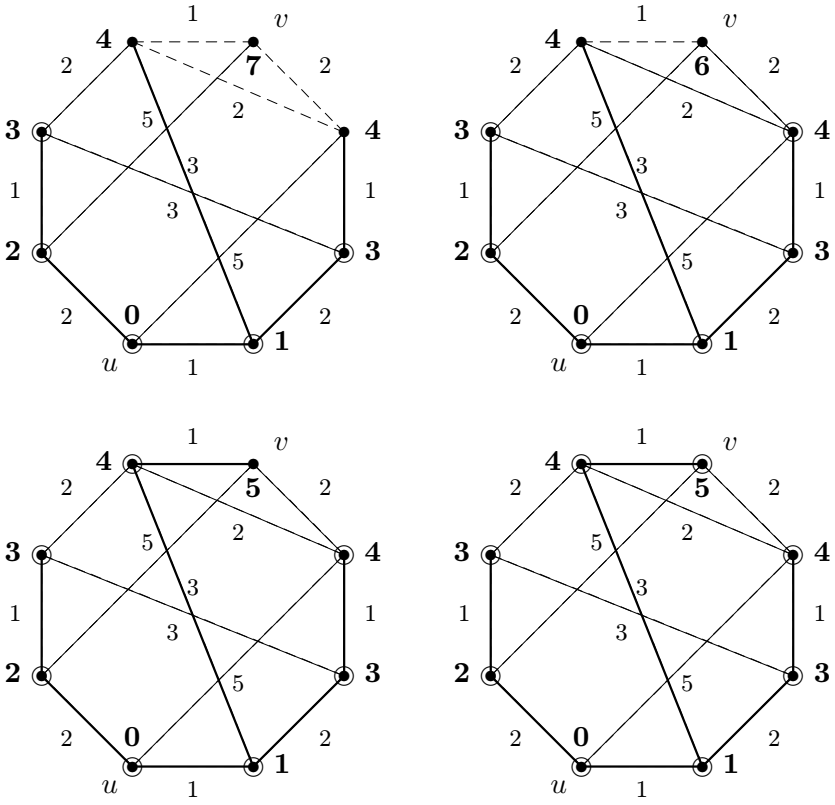
3.5 Cvičení: Příklady o grafových vzdálenostech

Příklad 3.12. *Ukázka běhu Dijkstrova Algoritmu 3.11 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 2.10 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:

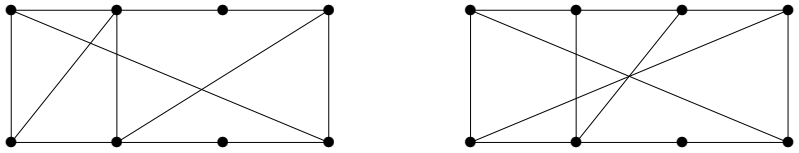




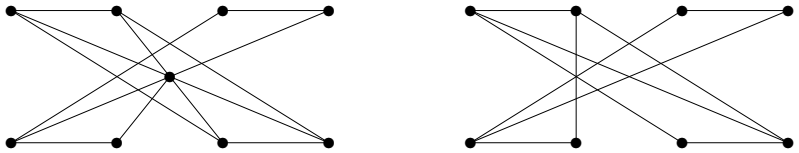
Z průběhu algoritmu vidíme, že již ve třetím kroku jsme určili dočasnou vzdálenost z U do v na 7, ale to nebyla ta nejkratší. Nakonec po proběhnutí všech kroků algoritmu vzdálenost u, v poklesla až na optimálních 5. Nejkratší cesta je naznačená tlustými čarami. Pamatujte proto, že Dijkstrův algoritmus musíte provádět vždy tak dlouho, dokud se konečně nezpracuje cílový vrchol. □

Úlohy k řešení

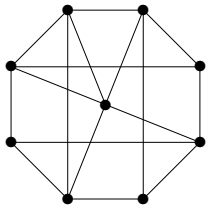
(3.5.1) *Jaká je největší vzdálenost mezi dvěma vrcholy v každém z následujících dvou grafů? Vyznačte tyto dva nejvzdálenější vrcholy.*



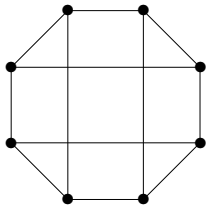
(3.5.2) *Jaká je největší vzdálenost mezi dvěma vrcholy v každém z následujících dvou grafů? Vyznačte tyto dva nejvzdálenější vrcholy.*



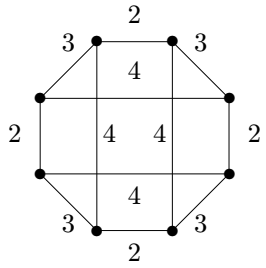
(3.5.3) *Kolik (neuspořádaných) dvojic vrcholů v následujícím grafu má vzdálenost právě* 3*?*



(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) *Jaká největší vzdálenost může být mezi dvěma vrcholy kružnice délky* 9*, jejíž hrany jsou ohodnoceny vzdálenostmi* $1, 2, \ldots, 8, 9$ *v nějakém (libovolném) pořadí?*

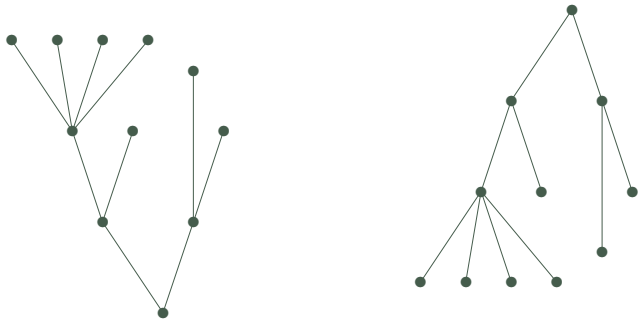
(3.5.8) *Představme si graf, jehož vrcholy jsou všechna přirozená čísla od* 2 *do* 15 *a hrany spojují právě ty dvojice vrcholů, které jsou soudělné jako čísla. Přitom délka hrany je vždy rovna největšímu společném děliteli. (Například* $4, 5$ *nejsou spojené a* $8, 12$ *jsou spojené hranou délky* 4*.)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

Část II
Užitečné Grafové Pojmy a Problémy

4 Stromy a les

Úvod

Jedním ze základních, a patrně nejjednodušším, typem grafů jsou takzvané stromy. Jedná se o souvislé grafy bez kružnic. Přes svou (zdánlivou) jednoduchost mají stromy bohatou strukturu a především množství vlastních aplikací. Patrně nejstarší motivací pojmu stromu jsou rodokmeny, jejichž původ sahá daleko před vznik teorie grafů. Proto také mnoho pojmů týkajících se stromů je motivováno rodokmeny. Co vidíme na následujících dvou obrázcích?



O důležitosti souvislosti grafu jsme pojednali dříve. Se stromy jako se souvislými (pod)grafy bez kružnic se pak setkáváme nejvíce v situacích, kde absence kružnic vyplývá z podstaty věci, nebo kde jsou kružnice zcela nežádoucí. To jsou různá schémata, datové struktury či již zmíněné rodokmeny. Stromy například přirozeně popisují různé hierarchické struktury. Mimo popisy struktur je ještě jedna důležitá oblast aplikací stromů, týkající se tzv. koster a problému hledání minimální kostry v grafu.

Cíle

Úkolem této lekce je hlavně definovat pojem stromu a ukázat základní vlastnosti stromů včetně jejich zdůvodnění. Podrobněji jsou probrány kořenové a uspořádané stromy a práce s nimi. Dále je zaveden pojem koster grafu a jejich zajímavé vlastnosti.

4.1 Základní vlastnosti stromů

Začneme definicemi lesa a stromů. Jelikož i smyčky a násobné hrany v multigrafech jsou považovány za kružnice délek 1 a 2, v lesech a stromech se nenacházejí. Bavíme se zde tudíž pouze o jednoduchých grafech.

`Definice`: *Les* je jednoduchý graf bez kružnic.

Definice 4.1. ***Strom*** je jednoduchý souvislý graf T bez kružnic.

`Fakt`. Komponenty souvislosti lesa jsou stromy. Jeden vrchol (bez hran) je také strom.

Dále si ukážeme přehled základních vlastností stromů, včetně jejich zdůvodnění, která jsou poměrně jednoduchá. Následující vlastnosti dokonce plně popisují stromy a mohou tudíž být použity místo uvedené Definice 4.1. Jedná se sice o dosti formální teoretický oddíl, ale alespoň zběžné porozumění uvedeným vlastnostem je potřebné pro další práci se stromy a s jinými strukturami založenými na stromech.

Lema 4.2. *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 vrchole 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 4.3. *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 4.2 má T vrchol v stupně 1. Označme $T'=T-v$ 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 4.4. *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 $H \subseteq P_1 \Delta P_2$ s neprázdnou množinou hran, kde H zřejmě má všechny stupně sudé. Na druhou stranu se však podgraf stromu musí opět skládat z komponent stromů, a tudíž obsahovat vrchol stupně 1 podle Lematu 4.2, spor. Proto cesta mezi u a v existuje jen jedna. □

Důsledek 4.5. *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 4.4. □

Věta 4.6. *Strom je minimální souvislý graf (na daných vrcholech).*

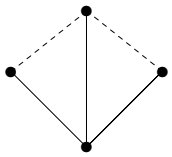
`Důkaz`. Strom je souvislý podle definice. Pokud by souvislý graf měl kružnici, zůstal by souvislý i po vypuštění některé hrany té kružnice. Proto minimální souvislý graf (na daných vrcholech) je stromem. Naopak, pokud by vypuštěním hrany $e=uv$ ze stromu T vznikl souvislý graf, pak by mezi u, v v T existovaly dvě cesty (dohromady kružnice) – hrana e a jiná cesta v $T-e$. To je ve sporu s Větou 4.4. Proto je strom minimálním souvislým grafem na daných vrcholech. □

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

Příklad 4.7. *Kolik nejvýše kružnic vznikne v grafu, který vytvoříme ze stromu přidáním dvou hran?*

Přidáním jedné hrany do stromu T vznikne jedna kružnice dle Důsledku 4.5. 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 vzniknout nemohou po přidání dvou hran e, f do stromu T: Podle Důsledku 4.5 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 4.4 dostáváme spor s faktem, že podgrafy stromů s hranami musí obsahovat vrchol stupně 1. □

Úlohy k řešení

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

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

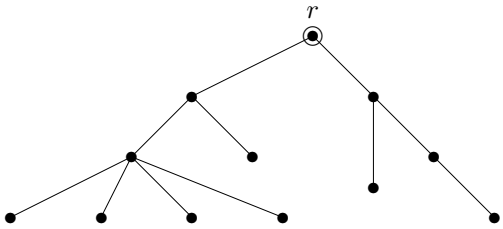
*(4.1.3)** *Najdete graf s dvěma kružnicemi, z něhož lze odebráním jedné hrany vytvořit strom? Zdůvodněte a případně nakreslete.*

4.2 Kořenové stromy

Při mnoha použitích stromů 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. Kořenové stromy mají také tradiční motivaci v rodokmenech a z toho vychází jejich běžná terminologie.

Definice 4.8. ***Kořenovým stromem*** je strom T spolu s vyznačeným *kořenem* $r \in V(T)$, zkráceně zapsaný dvojicí T, r.

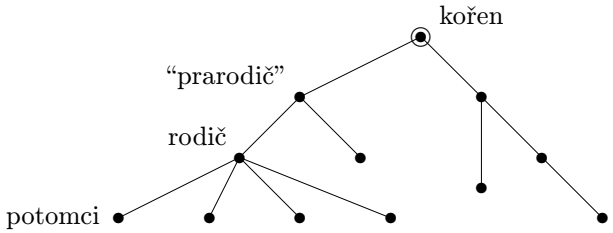
`Komentář:` Příklad kořenového stromu je na následujícím obrázku:



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) mohlo být považováno za “sexistické”.

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

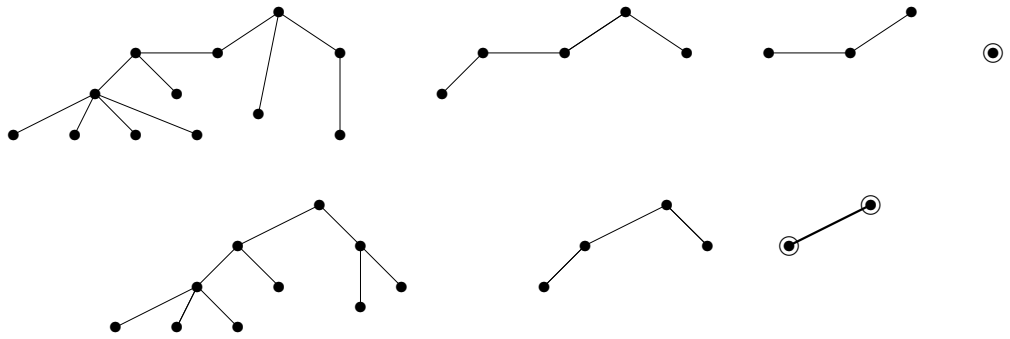
`Komentář:` Pozor, i kořen stromu může být listem, pokud má stupeň 1, ale obvykle se to tak neříká. List kořenového stromu, který není kořenem, nemá potomky.

Občas se můžeme dostat do situace, kdy k danému stromu potřebujeme zvolit kořen, 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 centra.

`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. Získané (rekurzivně) centrum T' je zároveň centrem T.

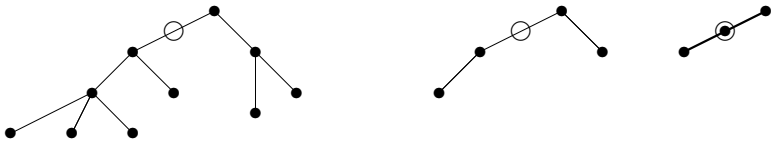
Příklad 4.9. *Ilustrací definice centra jsou následující dva postupy nalezení centra (zleva doprava):*



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. □

`Fakt`. 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:



Další dodatečnou informací často vázanou ke kořenovým stromům je nějaké 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.

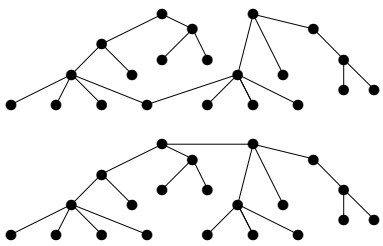
Uspořádání potomků vrcholu ve stromu je přirozeně požadováno v mnoha praktických situacích. Například ve stromových datových strukturách jsou často potomci explicitně seřazeni podle daného klíče, jako třeba ve vyhledávacích binárních stromech. I v případech, kdy uspořádání potomků ve stromě není dáno, je možné jej jednoznačně definovat, jak uvidíme v následující části.

Úlohy k řešení

(4.2.1) *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*

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

(4.2.2) *Určete centra následujících dvou stromů:*



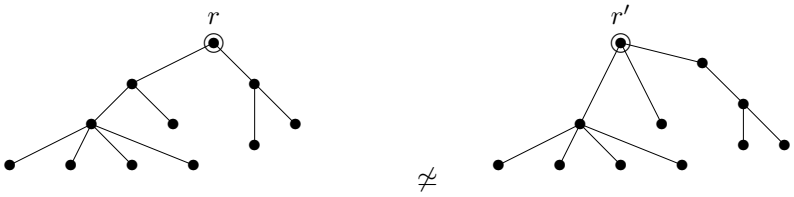
(4.2.3) *Mějme libovolný strom s* 33 *vrcholy. Kolik jeho listů postupně odebereme, než určíme centrum? (Je to jednoznačné?)*

4.3 Isomorfismus stromů

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

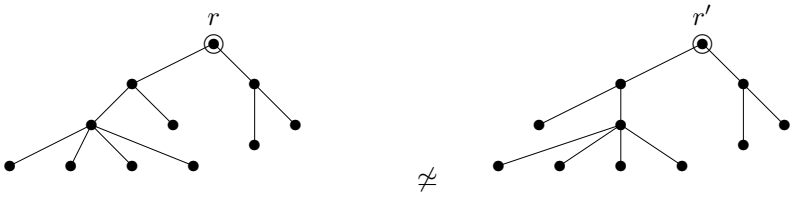
`Definice`: (Dva stromy jsou isomorfní pokud jsou isomorfní jako grafy.) Dva kořenové stromy T, r a T', r' jsou *isomorfní* pokud existuje isomorfismus mezi stromy T a T', který kořen r zobrazuje na kořen r'.

`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.

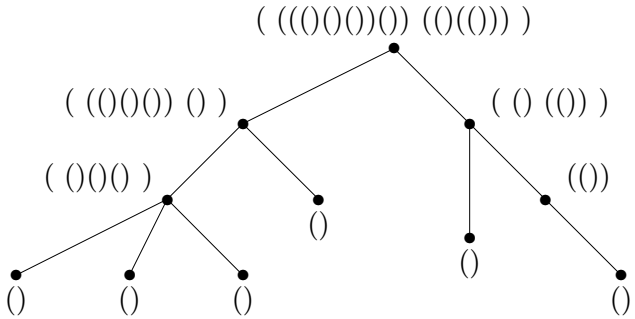
`Komentář:` Například následující dva (isomorfní) kořenové stromy *nejsou isomorfní* coby uspořádané kořenové stromy, neboť pořadí potomků levého syna kořene se liší.



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, třeba při sémantické analýze 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 a snadno zdůvodnitelným faktem o kódech pěstovaných stromů je toto tvrzení:

Lema 4.10. *Dva uspořádané kořenové (pěstované) stromy jsou isomorfní právě když jejich kódy získané podle předchozího popisu jsou shodné řetězce.*

`Fakt`. Je-li dán kód s uspořádaného kořenového stromu, příslušný strom nakreslíme následujícím postupem:

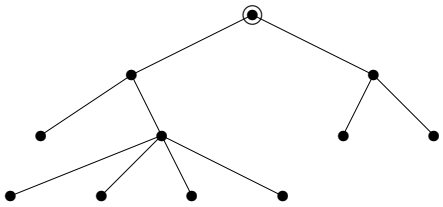
– Při přečtení znaku '(' na začátku spustíme pero na papír, do kořene.

– 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.

Příklad 4.11. *Nakreslete jako pěstovaný strom ten odpovídající závorkovému kódu*

$$((()(()()()())) (()())) .$$



□

Při určování isomorfismu obecných stromů použijeme Lema 4.10 pro jejich jednoznačnou reprezentaci uspořádanými kořenovými stromy, ve které kořen volíme v centru a potomky seřadíme podle jejich kódů vzestupně lexikograficky (tj. 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.

Algoritmus 4.12. ***Určení isomorfismu dvou stromů.***
Pro dané dva obecné stromy T a U implementujeme algoritmus zjišťující isomorfismus $T \simeq^{?} U$ následovně v symbolickém zápise:

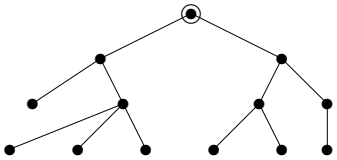
```
Vstup < stromy T a U;
for (X=T,U) { // určení center daných stromů pro kořeny
    x = centrum(X);
    if (x je jeden vrchol ) r = x;
    else nový r, nahraď hranu x=uv hranami ru, rv;
    k[X] = minimalni_kod(X,r);
}
if (k[T]==k[U] jako řetězce) printf("Jsou isomorfní.");
else printf("Nejsou isomorfní.");
exit;

Funkce minimalni_kod(strom X, vrchol r) {
    if (X má jeden vrchol) return "()";
    Y[1...d] = {souvislé komponenty X-r, tj. podstromy kořene r};
    s[1...d] = kořeny podstromů Y[] v odpovídajícím pořadí;
    for (i=1,...,d)
        k[i] = minimalni_kod(Y[i],s[i]);
    sort lexikograficky (abecedně) podle klíče k[1]<=k[2]<=...<=k[d];
    return "("+k[1]+...+k[d]+")";
}
```

Pro zdůvodnění správnosti Algoritmu 4.12 je klíčové následující tvrzení.

Věta 4.13. *Mějme dva stromy T, U a nechť T', r a U', s jsou po řadě jejich kořenové stromy získané v první části Algoritmu 4.12 (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 `minimalni_kod`$(T', r) =$ `minimalni_kod`(U', s).

Úlohy k řešení

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



(4.3.2) *Nakreslete pěstovaný strom odpovídající závorkovému kódu*

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

4.4 Kostry grafů

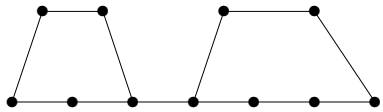
Kromě stromů samotných se zabýváme i stromy, které jsou obsaženy jako podgrafy ve větších grafech.

Definice 4.14. ***Kostrou*** souvislého grafu G
je podgraf v G, který je sám stromem a obsahuje všechny vrcholy grafu G.

`Komentář:` Kostrou stromu je strom sám. Na druhou stranu kostrou kružnice C_n je každá z n cest vzniklých z C_n vypuštěním jedné hrany. Složitější grafy mají pak ještě více možných koster.

`Poznámka`: Pojem kostry lze definovat i pro nesouvislé grafy, pak se kostrou myslí les, jehož stromy jsou kostrami jednotlivých komponent.

Příklad 4.15. *Kolik různých koster má tento graf?*



Podívejme se na kostru grafu takto – jaké hrany z grafu vymažeme, aby zbyl strom? Zajisté musíme vymazat některou hranu z první kružnice (5 možností) a některou hranu z druhé kružnice (6 možností). Na druhou stranu to v tomto jednoduchém příkladě už stačí, vždy pak zbude strom. Výběr vymazané hrany z první kružnice je nezávislý na druhé kružnici (jsou disjunktní), a proto dle principu nezávislých výběrů máme $5 \cdot 6 = 30$ možností vybrat dvě hrany k vymazání. Celkem tedy vyjde 30 koster. □

Následující výsledek patří ke „krásným drahokamům“ teorie grafů.

Věta 4.16. *Úplný graf K_n má přesně n^{n-2} koster.*

`Definice.` *Laplaceova matice* $\boldsymbol{Q} = (q_{ij})_{i,j=1}^{n}$ grafu G o n vrcholech je definována:

– $q_{ii} = d_G(i)$ (stupeň vrcholu),

– $q_{ij} = 0$ pro vrcholy $i \neq j$ nespojené hranou,

– $q_{ij} = -1$ pro vrcholy $i \neq j$ spojené hranou.

Věta 4.17. *Nechť $\boldsymbol{Q}$ je Laplaceova matice grafu G a matice $\boldsymbol{Q}'$ vznikne vyškrtnutím jejího prvního řádku a sloupce. Pak počet koster grafu G je roven determinantu $|\boldsymbol{Q}'|$.*

`Komentář:` Důkaz této překvapivé věty je mimo dosah naši přednášky (využívá silné nástroje lineární algebry). Uvědomte si, proč samotná matice $\boldsymbol{Q}$ je singulární (determinantu 0) – neboť součet prvků v každém řádku je 0. Je také možno vyškrtávat jiné řádky a sloupce, ale může se tím změnit znaménko determinantu.

Úlohy k řešení

(4.4.1) *Kolik koster má kružnice délky* 2004*?*

*(4.4.2) *Zkuste sami vlastním počítáním odvodit, že počet koster úplného grafu K_5 je $125 = 5^3$.*

Návod: Zkuste něco chytřejšího, než kreslení všech koster. Co třeba se podívat na jednotlivé "typy" koster?

*(4.4.3) *Odvoďte pomocí Věty 4.17, kolik koster má úplný bipartitní graf $K_{m,n}$.*

Rozšiřující studium

Stromům, jejich isomorfismu a kódování se věnují [5, Oddíly 4.1,2]. Zájemci o hlubší studium počítání koster v grafech si mohou přečíst (poměrně obtížnou) [5, Kapitola 7].

5 Minimální kostry, Hladový algoritmus

Úvod

Kromě teoretických "hrátek" mají kostry grafů (Oddíl 4.4) následující důležité praktické použití: Dříve jsme uvažovali spojení v grafech cestami jdoucími z jednoho místa do druhého, ale nyní se podíváme na jiný způsob "propojování" všech vrcholů grafu najednou. Vezměme si třeba požadavky propojení domů elektrickým rozvodem, propojení škol internetem, atd. Zde nás ani tak nezajímají délky jednotlivých cest mezi propojenými body, ale hlavně celková délka či cena vedení/spojení, které musíme postavit.

*Našim cílem je tedy najít minimální souvislý podgraf daného grafu, tedy minimální způsob propojení (v daných podmínkách), ve kterém existují cesty mezi každou dvojicí vrcholů. Podle Věty 4.6 je tímto minimálním propojením strom – kostra našeho grafu. Vstupní graf nám přitom udává všechny možné realizovatelné propojky s jejich cenami. Jak uvidíme, úloha se dá velmi dobře řešit tím asi nejjednodušším způsobem, tzv. hladovým postupem – **bereme vždy to nejlepší, co se zrovna nabízí**...*

Cíle

V lekci probereme významný problém minimální kostry grafu, včetně jednoduchého "hladového" algoritmu pro jeho řešení. Obecně je postup tzv. hladové optimalizace demonstrován na řešení několika jiných kombinatorických problémů, a v jeho souvislosti je také zmíněn pojem matroidu.

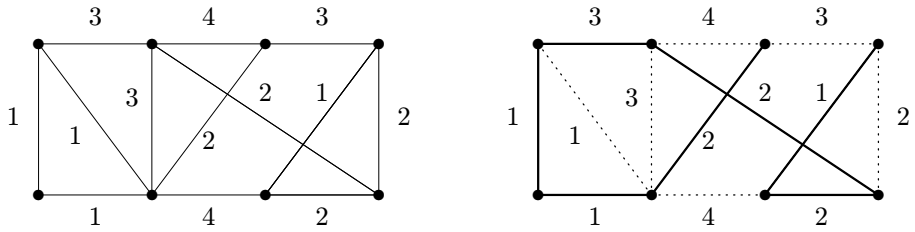
5.1 Hledání minimální kostry

Problém 5.1. ***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 5.2. ***„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. $w(e_1) \leq w(e_2) \leq \ldots \leq w(e_m)$.
- 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 5.17.

`Důkaz` správnosti Algoritmu 5.2:
Nechť T je množina hran získaná v Algoritmu 5.2 a nechť hrany jsou již seřazené $w(e_1) \leq w(e_2) \leq \ldots \leq w(e_m)$. 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 4.5 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 $(T_0 \setminus \{e_k\}) \cup \{e_j\}$ (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 důkaz toho, že ta poslední iterace záměn nemůže selhat. Nakreslete si tento důkaz obrázkem!

Základní hladový algoritmus pro hledání minimální kostry grafu byl poprvé explicitně popsán Kruskalem, ale už dříve byly objeveny jeho podobné varianty, které zde jen stručně zmíníme.

Algoritmus 5.3. ***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.

Avšak historicky vůbec první algoritmus pro problém minimální kostry (z roku 1928) byl nalezen jiným českým (brněnským) matematikem:

Algoritmus 5.4. ***Borůvkův pro minimální kostru.***
Toto je poněkud složitější algoritmus, chová se jako Jarníkův algoritmus spuštěný zároveň ze všech vrcholů grafu najednou. Detaily lze nalézt v literatuře [5, Oddíl 4.4].

Doplňkové otázky

(5.1.1) *Co se stane, pokud v Algoritmu 5.2 seřadíme hrany naopak, tedy sestupně?*

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

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

5.2 Hladový algoritmus v obecnosti

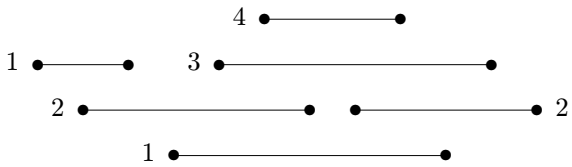
Asi nejprimitivnějším možným přístupem při řešení diskrétních optimalizačních úloh je postupovat stylem ***beru vždy to nejlepší, co se zrovna nabízí***... Tento postup obecně v češtině nazýváme *hladovým algoritmem*, i když lepší by bylo použít správnější překlad anglického "greedy", tedy nenasystný algoritmus. A ještě hezčí české spojení by bylo "algoritmus hamouna". Jednoduše bychom jej nastínili takto:

- Postupně v krocích vyber vždy to nejlepší, co se dá (nabízí).
- To vyžaduje zvolit *uspořádání na objektech*, ze kterých vybíráme.
- Průběh a úspěch algoritmu silně závisí na tomto zvoleném uspořádání (které již dále neměníme).

`Komentář:` Jak asi každý ví, nenasystnost či hamounství nebývá v životě tím nejlepším postupem, ale kupodivu tento princip perfektně funguje v mnoha kombinatorických úlohách! Jedním známým příkladem je výše uvedené hledání minimální kostry. Jiným příkladem je třeba jednoduchý problém přidělování (uniformních) pracovních úkolů, na němž si obecné schéma hladového algoritmu ilustrujeme.

Problém 5.5. ***Přidělení pracovních úkolů***
Uvažujeme zadané pracovní úkoly, které mají přesně určený čas začátku i délku trvání. (Jednotlivé úkoly jsou tedy reprezentovány uzavřenými intervaly na časové ose.) Cílem je takové přidělení úkolů pracovníkům, aby jich celkově bylo potřeba co nejméně. Všichni pracovníci jsou si navzájem rovnocenní – uniformní, tj. každý zvládne všechno.

`Komentář:` Pro příklad zadání takové problému si vezměme následující intervaly úkolů:



Kolik je k jejich splnění potřeba nejméně pracovníků? Asi sami snadno zjistíte, že 4 pracovníci stačí, viz zobrazené očíslování. Ale proč jich nemůže být méně?

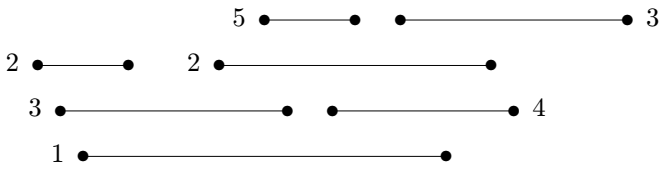
`Poznámka`: Uvedené zadání může být kombinatoricky popsáno také jako problém optimálního obarvení daného intervalového grafu (vrcholy jsou intervaly úkolů a hrany znázorňují překrývání intervalů).

Algoritmus 5.6. ***Hladový algoritmus rozdělení pracovních úkolů.***
Problém 5.5 je vyřešen následující aplikací hladového postupu:

1. *Úkoly nejprve seřadíme podle časů začátků.*
2. *Každému úkolu v pořadí přidělíme volného pracovníka s nejnižším číslem.*

`Důkaz`: Nechť náš algoritmus použije celkem k pracovníků. Dokážeme jednoduchou úvahou, že tento počet je optimální – nejlepší možný. V okamžiku, kdy začal pracovat pracovník číslo k, všichni $1, 2, \ldots, k-1$ také pracovali (jinak bychom vzali některého z nich). V tom okamžiku tedy máme k překrývajících se úkolů a každý z nich vyžaduje vlastního pracovníka. □

`Komentář:` Příklad neoptimálního přidělení pracovních úkolů dostaneme například tak, že na začátku úkoly seřadíme podle jejich časové délky. (Tj. čím delší úkol, tím dříve mu hladově přiřadíme pracovníka.)



Takový postup by se také mohl zdát rozumný, vždyť se v praxi často rozdělují nejprve ty velké úkoly a pak průběžně ty menší. Vidíme však na obrázku, že nalezené řešení není optimální – vyžaduje 5 místo 4 pracovníků.

Je tedy velmi důležité, podle jakého prinicipu seřadíme objekty (úkoly) na vstupu.

Doplňkové otázky

(5.2.1) *Proč tedy nestačí méně než 4 pracovníci pro splnění pracovních úkolů v zadání za Problémem 5.5?*

(5.2.2) *Co kdybychom v hladovém řešení Problému 5.5 seřadili úkoly podle časů jejich ukončení?*

5.3 Pojem matroidu

Pojem matroidu se často vyskytuje ve spojení s diskrétní optimalizací, viz třeba mnohé práce Edmondse. Jedná se o pojem velice "obtížný k uchopení", a proto si o něm zde uvedeme jen několik základních poznatků v souvislosti s hladovým algoritmem (Věta 5.12).

Definice 5.7. ***Matroid*** na množině X, značený $M = (X, \mathcal{N})$,
je takový systém $\mathcal{N}$ podmnožin nosné množiny X, ve kterém platí následující:

1. $\emptyset \in \mathcal{N}$
2. $A \in \mathcal{N}$ a $B \subset A \;\Rightarrow\; B \in \mathcal{N}$
3. $A, B \in \mathcal{N}$ a $|A| < |B| \;\Rightarrow\; \exists y \in B \setminus A : A \cup \{y\} \in \mathcal{N}$

Množinám ze systému $\mathcal{N}$ říkáme *nezávislé množiny*. Těm ostatním pak říkáme *závislé*. Nezávislým množinám, do kterých již nelze přidat žádný prvek tak, že zůstanou nezávislé, říkáme *báze matroidu*.

> `Komentář:` Nejdůležitější částí definice matroidu je zvýrazněný třetí bod. Přímo ukázkový příklad matroidu nám dává lineární algebra – všechny lineárně nezávislé podmnožiny vektorů tvoří matroid. Odtud také pocházejí pojmy nezávislosti a báze matroidu, které přímo odpovídají příslušným pojmům vektorového prostoru.

Lema 5.8. *Všechny báze matroidu obsahují stejně mnoho prvků.*

`Důkaz`: Toto přímo vyplývá z třetí vlastnosti definice matroidu: Pokud nezávislá množina A má méně prvků než báze B, tak do A lze vždy přidat další prvek x tak, že zůstane $A \cup \{x\}$ nezávislá. □

Nyní uvedeme několik poznatků o stromech, které jsou relevantní pro zavedení "grafových" matroidů.

Lema 5.9. *Les na n vrcholech s c komponentami souvislosti má přesně $n - c$ hran.*

`Důkaz`: Každý vrchol lesa L náleží právě jedné komponentě souvislosti z definice. Jak známo, každý strom, tj. komponenta lesa L, má o jednu hranu méně než vrcholů. Ve sjednocení c komponent tak bude právě o c méně hran než vrcholů. □

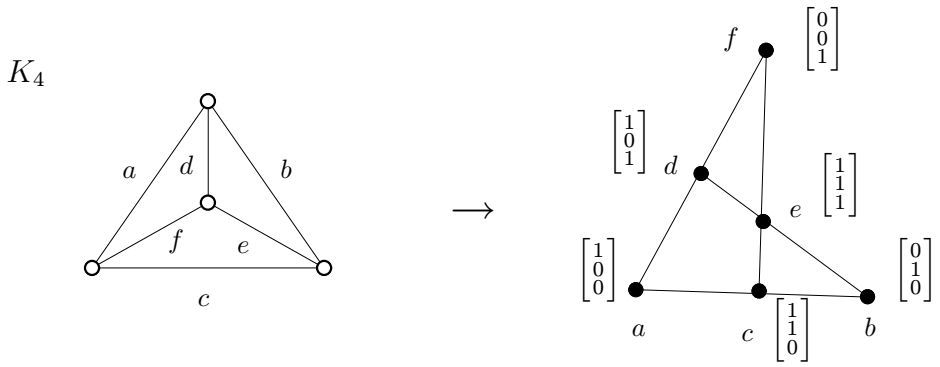
`Definice`: Řekneme, že podmnožina hran $F \subset E(G)$ je *acyklická*, pokud podgraf s vrcholy $V(G)$ a hranami z F nemá kružnici.

Lema 5.10. *Nechť F_1, F_2 jsou acyklické podmnožiny hran grafu G a $|F_1| < |F_2|$. Pak existuje hrana $f \in F_2 \setminus F_1$ taková, že $F_1 \cup \{f\}$ je také acyklická podmnožina.*

`Důkaz`: Jelikož $|F_1| < |F_2|$ a platí Lema 5.9, má podgraf G_1 tvořený hranami z F_1 více komponent než podgraf G_2 tvořený hranami z F_2. Potom však některá hrana $f \in F_2$ musí spojovat dvě různé komponenty podgrafu G_1, a tudíž přidáním f do F_1 nevznikne kružnice. □

`Definice`: Podle Lematu 5.10 tvoří systém všech acyklických podmnožin hran v (libovolném) grafu G matroid. Tento matroid nazýváme *matroidem kružnic* grafu G.
V analogii s grafy dále používáme název *kružnice* pro minimální závislé množiny matroidu.

`Komentář:` Dva příklady matroidu jsou hezky ilustrovány v následujícím obrázku, který ukazuje, jak hrany grafu K_4 vlevo odpovídají vektorům v matroidu kružnic vpravo. Čáry (zvané "přímky") v pravém schématu vyznačují lineární závislosti mezi vektory; tj. nezávislé jsou ty trojice bodů, které neleží na žádné společné "přímce".



Abstraktní hladový algoritmus

V praxi se matroid obvykle nezadává výčtem všech nezávislých množin, protože těch je příliš mnoho (až 2^n pro n-prvkovou množinu X). Místo toho bývá dána externí funkce pro testování nezávislosti dané podmnožiny.

Algoritmus 5.11. ***Nalezení minim. báze matroidu – hladový algoritmus.***
`vstup:` *množina X s váhovou funkcí $w : X \to \mathbb{R}$,*
matroid na X určený externí funkcí `nezavisla(Y);`

`setřídit X=(x[1],x[2],...,x[n])` *tak, aby* `w[x[1]]<=...<=w[x[n]];`
`B = ∅;`
`for (i=1; i<=n; i++)`
`  if (nezavisla(B∪{x[i]}))`
`    B = B∪{x[i]};`

`výstup:` *báze B daného matroidu s minimálním součtem ohodnocení vzhledem k w.*

`Poznámka`: Pokud X v tomto algoritmu je množina hran grafu, w váhová funkce na hranách a nezávislost znamená acyklické podmnožiny hran (matroid kružnic grafu), pak Algoritmus 5.2 je přesně instancí Algoritmu 5.11.

Věta 5.12. *Algoritmus 5.11 (hladový algoritmus) pro danou nosnou množinu X s váhovou funkcí $w : X \to \mathbb{R}$ a pro daný matroid $\mathcal{N}$ na X správně najde bázi v $\mathcal{N}$ s nejmenším součtem vah.*

`Důkaz`: Z definice matroidu je jasné, že k výsledné množině B již nelze přidat další prvek, aby zůstala nezávislá, proto je B báze. Seřaďme si prvky X podle vah jako v algoritmu $w(x[1]) \leq \ldots \leq w(x[n])$. Nechť indexy $i_1, i_2, \ldots, i_k$ určují vybranou k-prvkovou bázi B v algoritmu a nechť indexy $j_1, j_2, \ldots, j_k$ vyznačují (třeba jinou?) bázi $\{x[j_1], \ldots, x[j_k]\}$ s nejmenším možným součtem vah.

Vezměme nejmenší $r \geq 1$ takové, že $w(x[i_r]) \neq w(x[j_r])$. Potom nutně $w(x[i_r]) < w(x[j_r])$, protože náš algoritmus je "hladový" a bral by menší $w(x[j_r])$ již dříve. Na druhou stranu, pokud by druhá báze $\{x[j_1], \ldots, x[j_k]\}$ dávala menší součet vah, muselo by existovat jiné $s \geq 1$ takové, že $w(x[i_s]) > w(x[j_s])$. Nyní vezměme nezávislé podmnožiny $A_1 = \{x[i_1], \ldots, x[i_{s-1}]\}$ a $A_2 = \{x[j_1], \ldots, x[j_s]\}$, kde A_2 má o jeden prvek více než A_1 a všechny prvky A_2 mají dle předpokladu menší váhu než $w(x[i_s])$.

Podle definice matroidu existuje $y \in A_2 \setminus A_1$ takové, že $A_1 \cup \{y\}$ je nezávislá. Přitom samozřejmě $y = x[\ell]$ pro nějaké ℓ. Ale to není možné, protože, jak je výše napsáno,

$w(y) < w(x[i_s])$, takže by náš hladový algoritmus musel $y = x[\ell]$, $\ell < i_s$ vzít dříve do vytvářené báze B než vzal $x[i_s]$. Proto jiná báze s menším součtem vah než nalezená B neexistuje. □

Tím jsme zároveň podali i jiný důkaz správnosti Algoritmu 5.2, který je jen specifickou instancí Algoritmu 5.11.

`Poznámka`: Požadavek, že hladový Algoritmus 5.11 hledá bázi s minimálním součtem vah je v zásadě jen naší konvencí. Je jasné, že obrácením znamének u hodnot w se z minimalizace stává maximalizace a naopak.
Na druhou stranu je obecně podstatný požadavek, že výsledná množina má být bází, ne jen nezávislou množinou, neboť při kladných hodnotách w by minimální nezávislou množinou byla vždy $\emptyset$. (Naopak při maximalizaci s kladnými hodnotami w vychází automaticky báze jako ta nezávislá množina s maximálním ohodnocením.)

5.4 Kdy hladový algoritmus (ne)pracuje správně

Čtenáře asi napadne, že hladový algoritmus nemůže fungovat vždy optimálně. My jsme dokonce schopni popsat všechny struktury, na kterých hladový postup funguje univerzálně – jsou to právě matroidy.

Věta 5.13. *Nechť X je nosná množina se systémem "nezávislých" podmnožin $\mathcal{N}$ splňující podmínky (1,2) Definice 5.7. Pokud pro jakoukoliv váhovou funkci $w : X \to \mathbb{R}$ najde Algoritmus 5.11 optimální nezávislou množinu z $\mathcal{N}$, tak $\mathcal{N}$ splňuje také podmínku (3), a tudíž tvoří matroid na X.*

`Důkaz`: Tvrzení dokazujeme sporem. Předpokládejme, že vlastnost (3) neplatí pro dvojici nezávislých množin A, B, tj. že $|A| < |B|$, ale pro žádný prvek $y \in B \setminus A$ není $A \cup \{y\}$ nezávislá. Nechť $|A| = a, |B| = b$, kde $2b > 2a+1$. Zvolíme následující ohodnocení

- $w(x) = -2b$ pro $x \in A$,
- $w(x) = -2a - 1$ pro $x \in B \setminus A$,
- $w(x) = 0$ jinak.

Hladový algoritmus přirozeně najde bázi B_1 obsahující A a disjunktní s $B \setminus A$ podle našeho předpokladu. Její ohodnocení je $w(B_1) = -2ab$. Avšak optimální bází je v tomto případě jiná B_2 obsahující celé B a mající ohodnocení nejvýše $w(B_2) \leq (-2a - 1)b = -2ab - b < w(B_1)$. To je ve sporu s dalším předpokladem, že i při námi zvoleném ohodnocení w nalezne hladový algoritmus optimální bázi. Proto je sporný náš předpoklad o množinách A, B a podmínka (3) je splněna. □

Příklad 5.14. *Nakonec uvádíme dvě ukázky dobře známých kombinatorických úloh, ve kterých hladový algoritmus výrazně selže:*

Obarvení grafu. Problém obarvení grafu žádá přiřazení co nejméně barev vrcholům tak, aby sousední dvojice dostaly různé barvy. Jak jsem již poznamenali, v Problému 5.5 bylo přidělování úkolů pracovníkům vlastně barvením grafu. Obecně hladově barvíme graf tak, že ve zvoleném pořadí vrcholů každému následujícímu přidělíme první volnou barvu.

Třeba v nakreslené cestě délky 3 můžeme barvit hladově v pořadí od vyznačených krajních vrcholů, a pak musíme použít 3 barvy místo optimálních dvou.

Vrcholové pokrytí. Problém vrcholového pokrytí se ptá na co nejmenší podmnožinu C vrcholů daného grafu takovou, že každá hrana má alespoň jeden konec v C. Přirozeným hladovým postupem by bylo vybírat od vrcholů nejvyšších stupňů ty, které sousedí s doposud nepokrytými hranami. Bohužel tento postup také obecně nefunguje.

□

`Poznámka`: Zmíněná selhání hladového algoritmu se obecně vážou k nevhodně zvolenému pořadí kroků. Nemysleme si však, že by se tato selhání dala nějak snadno napravit volbou jiného pořadí – platí, že nalezení optimálního pořadí kroků pro použití hladového algoritmu může být (a bývá) stejně obtížné jako vyřešení úlohy samotné.

Doplňkové otázky

(5.4.1) *Jak špatně může dopadnout hladové barvení bipartitního grafu? (Bipartitní grafy jsou ty, které lze optimálně obarvit 2 barvami.)*
Přesně se otázkou myslí, kolik barev se hladově použije pro nejhorší bipartitní graf při nejhorším uspořádání jeho vrcholů, když se v každém kroku pro nový vrchol vybírá první volná barva.

(5.4.2) *V jakém (jednoduše spočitatelném) pořadí barvit vrcholy bipartitního grafu, aby stačily 2 barvy?*

(5.4.3) *Jak lze (dobře) využít hladový algoritmus pro obarvení grafu se všemi vrcholy stupně k pomocí $k+1$ barev?*

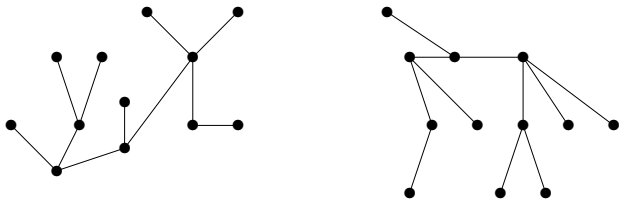
(5.4.4) *Kdy selže hladový postup pro vrcholové pokrytí?*

Rozšiřující studium

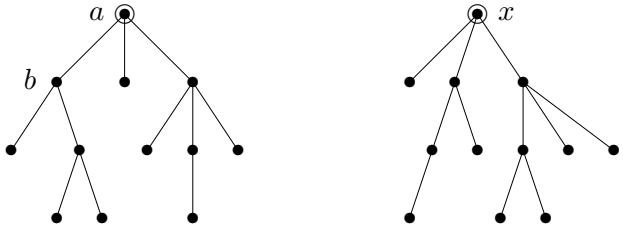
Problém minimální kostry a jeho historie jsou výborně shrnuty v [5, Oddíly 4.3,4,5]. Hladový postup je běžně zmiňován a používán v knihách zabývajících se algoritmy a optimalizací. Pro konkrétní algoritmické implementace odkazujeme i na [3]. V oblasti teorie matroidů je jen poskrovnu českých textů, ale obsáhlou a asi nejlepší anglickou monografií je [8]. Krátký čtivý úvod do matroidů je volně dostupný na [9].

5.5 Cvičení: Příklady o stromech, kostrách a hladovém postupu

Příklad 5.15. *Zjistěte, zda následující dva stromy jsou isomorfní.*



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.

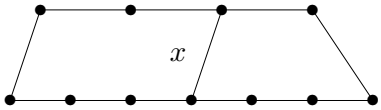


Nyní přejdeme k určení minimálního závorkového kódu pro levý strom (Algoritmus 4.12): Například při rekurzivním volání ve vrcholu b určíme kódy jeho podstromů jako '()' a '(()())'. Jelikož levou závorku považujeme za slovníkově menší, tyto dva podkódy

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. □

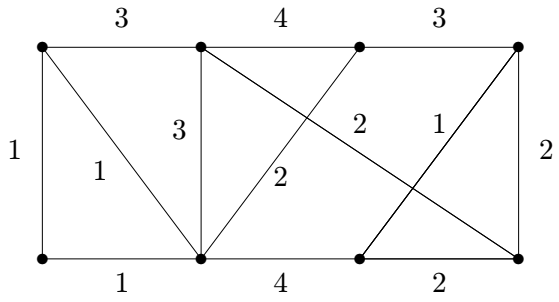
Příklad 5.16. *Kolik různých koster má tento graf?*



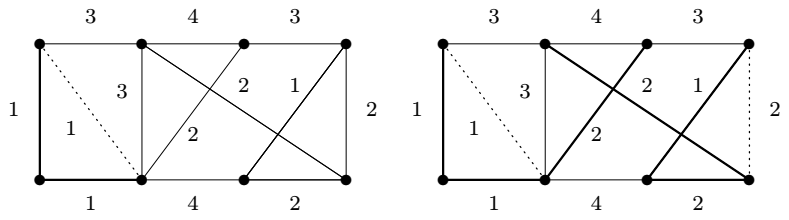
Jak vidíme, opět musíme vymazat dvě hrany z grafu, aby zbyla kostra. Nelze však vypouštět jednu hranu z levé kružnice a jednu z pravé, neboť by výběry nezyly nezávislé – hrana x je oběma sdílená. Podíváme se tedy na řešení jako na součet disjunktních možností, počtu koster obsahujících x a počtu koster bez hrany x.

Pokud hranu x vymažeme, zbude kružnice délky 11 a ta má 11 koster. Pokud naopak hranu x zachováme, musíme z levé kružnice od x vymazat jednu ze 6 hran a z pravé kružnice jednu z 5 hran. Tyto výběry již jsou nezávislé a počet možností je $6 \cdot 5 = 30$. Celkem má náš graf $11 + 30 = 41$ koster. □

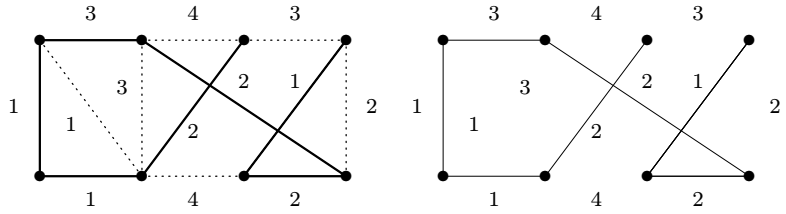
Příklad 5.17. *Najděme hladovým algoritmem minimální kostru v tomto váženém grafu (váhy hran jsou zapsány čísly v obrázku).*



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.

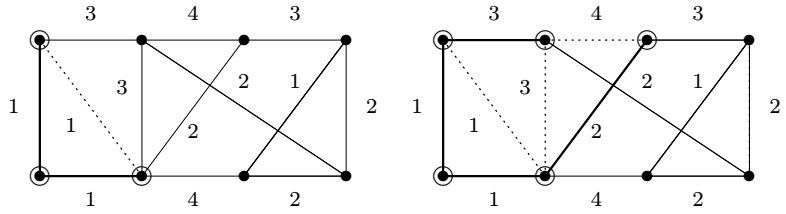


Získáme tak minimální kostru velikosti $1 + 2 + 2 + 3 + 1 + 1 + 2 = 12$, která je v tomto případě (náhodou) cestou, na posledním obrázku vpravo.

Poznamenáváme, že při jiném seřazení hran stejné váhy by kostra mohla vyjít jinak, ale vždy bude mít stejnou minimální velikost 12. (Například místo levé svislé hrany může obsahovat přilehlou úhlopříčku stejné váhy 1.) □

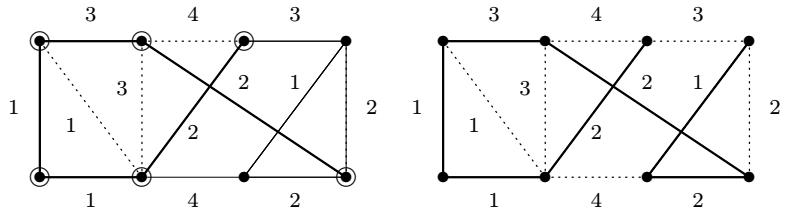
Příklad 5.18. *Najděme minimální kostru grafu z Příkladu 5.17 Jarníkovým algoritmem.*

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. (Všimněte si dobře tohoto rozdílu od základního hladového postupu – v tomto algoritmu se nemusí hrany probírat globálně monotónně podle svých vah!) Takto dosáhneme i pravého dolního vrcholu, na následujícím obrázku vlevo.



Nakonec ještě dosáhneme zbylé dva vrcholy hranami po řadě vah 2 a 1. Získáme tak stejnou minimální kostru jako v Příkladě 5.17. Opět je však možnost nalézt jinou z

minimálních koster stejné velikosti, pokud mezi hranami stejné váhy vedoucími ven v každém kroku vybíráme jinak. □

V dalších dvou příkladech si ukážeme použití hladového postupu na řešení jiných problémů než minimální kostry. Obě aplikace postupu jsou podobné a poměrně přirozené, přesto jen první z nich je matematicky korektní – dává vždy optimální výsledek, kdežto druhá ne.

Příklad 5.19. *Podívejme se na tento přirozený problém z univerzitního prostředí: Kroužky studentů mají během dne pevně naplánované časy cvičení. Našim úkolem je rozmístit cvičení do co nejméně učeben, aby se samozřejmě v jedné učebně cvičení nepřekrývala. K vyřešení použijeme hladový postup:*

- *Nejdříve začínající cvičení umístíme do učebny č. 1.*
- *Vybereme nejdříve začínající neumístěné cvičení (libovolné, pokud jich více začíná ve stejnou dobu) a umístíme jej do první volné učebny nejmenšího čísla.*
- *Opakujeme předchozí bod, dokud nejsou všechna cvičení rozmístěna.*

Jak dobrý výsledek (tj. maximální počet potřebných učeben) nám tento postup dá?

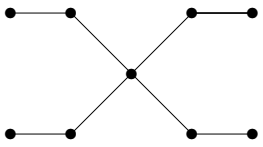
Je to možná s podivem, ale tento primitivní postup nám dá zcela optimální řešení úlohy, jak si nyní stručně zdůvodníme:
Předpokládejme, že učebny jsou číslovány $1, 2, 3, \ldots$. Pokud uvedený hladový postup potřebuje celkem k učeben, znamená to, že v některém svém kroku již měl učebny $1, 2, \ldots, k-1$ obsazené probíhajícími cvičeními a pro další cvičení musel použít učebnu číslo k. (Jinak by použil učebnu menšího čísla.) To však znamená, že v onom okamžiku se najednou koná k různých cvičení, a proto použití nejméně k učeben je nutné. □

Příklad 5.20. *Vrcholovým pokrytím v grafu G nazveme množinu vrcholů $X \subset V(G)$ takovou, že každá hrana grafu G má aspoň jeden konec v X. Zadaným úkolem je nalézt v grafu vrcholové pokrytí nejmenší možné velikosti. Použijeme hladový postup (kdy se snažíme každým krokem pokrýt co nejvíce ze zbylých hran v grafu):*

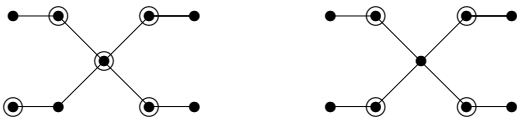
- *Začneme s prázdnou $X = \emptyset$.*
- *Vybereme vrchol v největšího stupně v G a přidáme jej do X.*
- *Vrchol v odebereme z grafu G (i s jeho hranami) a jdeme zpět na předchozí bod, dokud nezbude graf G prázdný.*

Proč je toto řešení obecně nekorektní?

Popsaný hladový postup v některých případech nenalezne nejmenší možnou množinu vrcholového pokrytí. Podívejme se na tento graf:



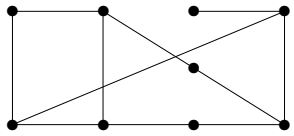
Hladový postup nejprve vybere prostřední vrchol, ale pak ještě musí v dalších čtyřech krocích vybrat 4 vrcholy, tedy celkem vyjde množina X velikosti 5:



Snadno však zjistíme, že optimální vrcholové pokrytí má velikost jen 4 a je vyznačené na obrázku vpravo. □

Úlohy k řešení

(5.5.1) *Kolik hran je třeba vypustit z následujícího grafu na 9 vrcholech, aby zbyla jeho kostra?*



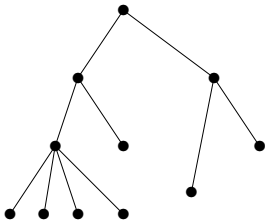
(5.5.2) *Kolik vrcholů má strom s 2004 hranami? Je to jednoznačné?*

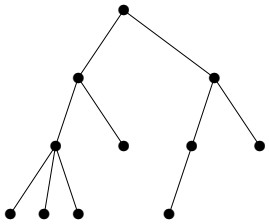
(5.5.3) *Les má 2009 vrcholů a celkem 4 souvislé komponenty. Kolik má hran?*

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

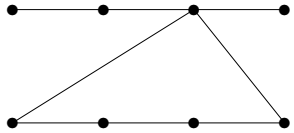
(5.5.5) *Mějme čísla $\{10, 11, 13, 14, 15, 21\}$ a spojme dvojice z nich hranami, pokud jsou soudělná. Je výsledný graf stromem? A aspoň lesem?*

(5.5.6) *Které z následujících výrazů jsou správnými závorkovými kódy pro nakreslené uspořádané kořenové stromy?*
A: `((((()()())())((())()))`
B: `((((()()()())())(()()))`
C: `((((()()())())(()(())))`





(5.5.7) *Kolik různých koster má tento graf?*

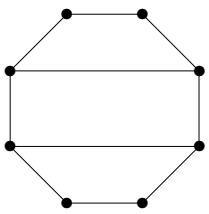


(5.5.8) *Kolik koster může mít graf, který vznikne ze stromu na 8 vrcholech přidáním libovolné nové hrany (mezi dvěma nespojenými vrcholy)? Uvědomte si, že pro různé stromy a hrany mohou vyjít různé výsledky, a vašim úkolem je najít všechny možné počty, včetně příslušných obrázků.*

(5.5.9) *Kolik hran je třeba vypustit z úplného grafu na n vrcholech, aby vznikla jeho kostra?*

*(5.5.10)** *Kolik nejméně vrcholů musí mít graf (bez násobných hran), ve kterém lze nalézt dvě kostry nesdílející žádnou hranu? Zdůvodněte.*

*(5.5.11)** *Kolik koster má graf na následujícím obrázku?*

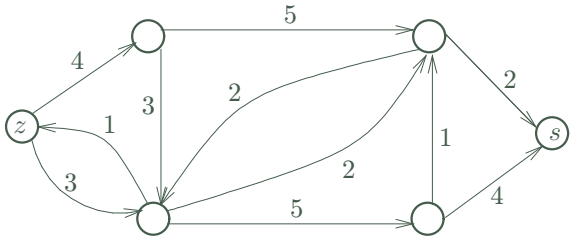


6 Toky v sítích

Úvod

Nyní se podíváme ještě na jednu oblast úloh, kde našla teorie grafů bohaté uplatnění (konkrétně orientované grafy). 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 řešení má (kupodivu) i mnoho teoretických důsledků, třeba pro souvislost či párování v grafech.

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).

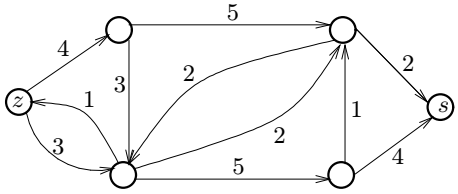
6.1 Definice sítě

Základní strukturou pro reprezentaci sítí je orientovaný graf. Vrcholy grafu modelují jednotlivé uzly sítě a hrany jejich spojnice. Vzpomeňme si (strana 7), 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 6.1. ***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, kolik nejvíce 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.

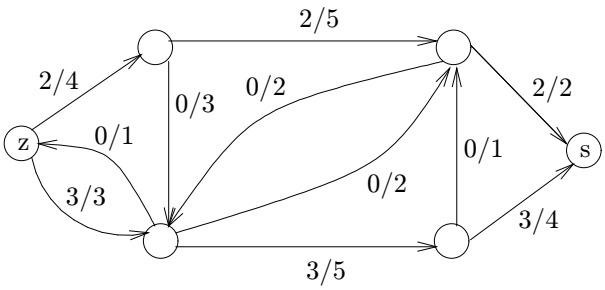
Definice 6.2. ***Tok*** v síti $S = (G, z, s, w)$
je funkce $f : E(G) \to \mathbb{R}_0^+$ splňující

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

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

`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.



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

$$0 = \sum_e (f(e) - f(e)) = \sum_v \sum_{e \leftarrow v} f(e) - \sum_v \sum_{e \to v} f(e) = \sum_{v=z,s} \left(\sum_{e \leftarrow v} f(e) - \sum_{e \to v} f(e) \right) .$$

(Dvojité sumy uprostřed předchozího vztahu nabývají stejných hodnot pro všechny vrcholy kromě z a s dle definice toku.) Proto velikost toku počítaná u zdroje je rovna opačné velikosti toku počítaného u stoku

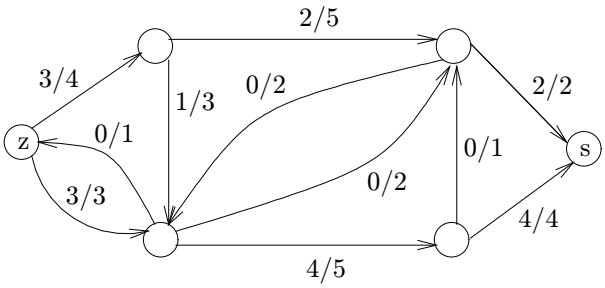
$$\left(\sum_{e \leftarrow z} f(e) - \sum_{e \to z} f(e) \right) = - \left(\sum_{e \leftarrow s} f(e) - \sum_{e \to s} f(e) \right) .$$

6.2 Hledání maximálního toku

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

Problém 6.3. ***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 6.2.*

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

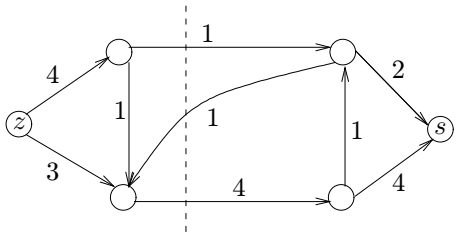


Jak však poznáme, že větší tok již v dané síti neexistuje? V této konkrétní ukázce to není obtížné, vidíme totiž, že obě dvě hrany přicházející do stoku mají součet kapacit $2 + 4 + 6$, takže více než 6 do stoku ani přitéct nemůže. V obecnosti lze použít obdobnou úvahu, kdy najdeme podmnožinu hran, které 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.

Definice 6.4. ***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 6.5. *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 $1 + 4 = 5$.



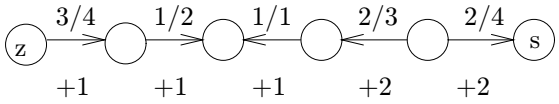
`Poznámka`: Tato věta poskytuje tzv. *dobrou charakterizaci* problému maximálního toku: Když už nalezneme maximální tok, tak je pro nás vždy snadné dokázat, že lepší tok není, nalezením příslušného řezu o stejné velikosti. Přitom toto zdůvodnění řezem můžeme směle ukázat i někomu, kdo se vůbec nevyzná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

Důkaz Věty 6.5 bude proveden následujícím algoritmem.

`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.



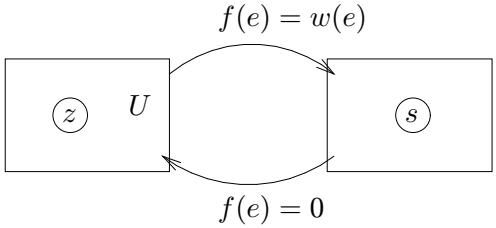
Všimněte si dobře, že cesta není orientovaná, takže hrany na ní jsou v obou směrech.

Algoritmus 6.6. ***Ford–Fulkersonův pro tok v síti.***

`vstup` *síť* $S = (G, z, s, w)$`;`
tok $f \equiv 0$`;`
`do {`
  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`;`
  `}`
`while (` $s \in U$ `);`
`výstup` *vypíšeme maximální tok* f`;`
`výstup` *vypíšeme min. řez jako množinu hran vedoucích z* U *do* $V(G) - U$`.`

`Důkaz` správnosti Algoritmu 6.6:
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. Zároveň tím dokážeme i platnost Věty 6.5.

Takže stačí dokázat, 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 $f(e) = 0$. Velikost toku f ze z do s se také dá psát jako

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

To je přesně, co jsme chtěli dokázat o výsledném toku. □

Z popisu Algoritmu 6.6 vyplývá ještě jeden důležitý důsledek:

Důsledek 6.7. *Pokud jsou kapacity hran sítě S celočíselné, optimální tok také vyjde celočíselně.*

`Poznámka`: Algoritmus pro celá čísla kapacit vždy skončí. Pro reálná čísla se ale dají najít extrémní případy, které nepovedou k řešení ani v limitě.

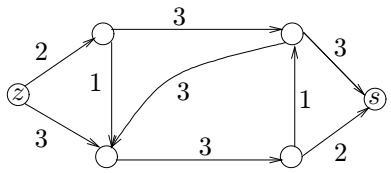
Pro rychlý běh algoritmu je vhodné hledat nejkratší nenasycenou cestu, tj. prohledáváním sítě do šířky. V takové implementaci algoritmus dobře a rychle funguje i s reálnými kapacitami hran.

Algoritmus 6.8. ***„Tří Indů" pro tok v síti***
V intencích Algoritmu 6.6 provádíme následující iterace:

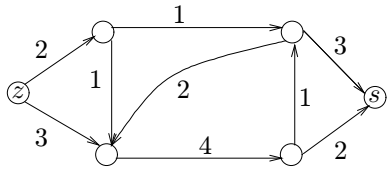
- *Prohledáváním sítě do šířky nalezneme všechny nenasycené cesty nejkratší délky.*
- *V předchozím nalezené nenasycené cesty souběžně nasytíme „dynamickým" algoritmem.*

Úlohy k řešení

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



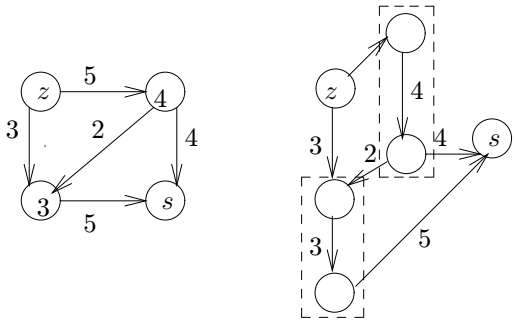
(6.2.2) *Co obsahuje výsledná množina U Algoritmu 6.6 v předchozím příkladě?*
(6.2.3) *Kde je minimální řez v této síti mezi z a s?*



6.3 Zobecnění sítí a další aplikace

Pojmy sítě a toků v ní lze zobecnit v několika směrech. My si zde stručně uvedeme tři možnosti:

1. U sítě můžeme zadat i *kapacity vrcholů*.
   To znamená, že žádným vrcholem nemůže celkem protéct více než povolené množství substance. 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:



2. Pro hrany sítě lze zadat také *minimální kapacity*, tedy dolní meze toku. (Například u potrubní sítě mohou minimální vyžadované průtoky vody garantovat, že nedojde k zanesení potrubí.) V této modifikaci úlohy již přípustný tok nemusí vůbec existovat. Takto zobecněná úloha je také snadno řešitelná, ale my se jí nebudeme zabývat.

3. V síti lze najednou přepravovat více substancí. To vede na problém tzv. *vícekomoditních toků* v síti. Tento problém je složitější a už není v obecnosti snadno řešitelný, a proto 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 dalších částech tohoto oddílu.

Bipartitní párování

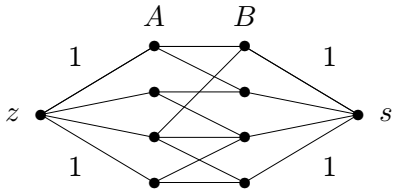
Biparitní grafy jsou grafy, jejichž vrcholy lze rozdělit do dvou množin tak, že všechny hrany vedou jen mezi těmito množinami.

`Definice`: *Párování* v (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 neuvažujeme bigamii ani jisté 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 partnerské vztahy představují 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 6.9. ***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 6.6. Do párování vložíme ty hrany grafu G, které mají nenulový tok.

`Důkaz` správnosti Algoritmu 6.9: Podle Důsledku 6.7 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. 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 6.7) 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 6.5 na tuto situaci přímo poskytne následující tvrzení.

Důsledek 6.10. *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 2.5 (Mengerovu). Ještě jiné použití si ukážeme na problému výběru reprezentantů množin.

Různí reprezentanti

`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 6.11. *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 $M_1 \cup M_2 \cup \ldots \cup M_k$. 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 6.9: 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 Lematu 6.10 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

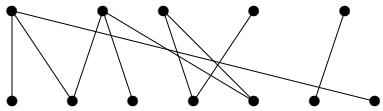
$$\left|\bigcup_{j\in J} M_j\right| = |X \cap \{x_1, \ldots, x_m\}| = |X| - |X \cap \{1, \ldots, k\}| = |X| - k + |J|\,.$$

(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. □

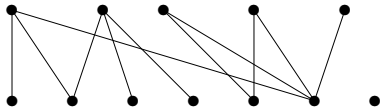
`Poznámka`: Předchozí důkaz nám také dává návod, jak systém různých reprezentantů pro dané množiny nalézt – stačí použít Algoritmus 6.6 na vhodně odvozenou síť.

Úlohy k řešení

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



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



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

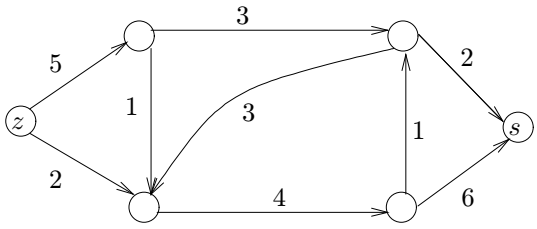
(6.3.4) *Mají všechny tříprvkové podmnožiny množiny* $\{1, 2, 3, 4, 5\}$ *systém různých reprezentantů?*

Rozšiřující studium

Problematika toků v sítích je běžnou součástí teorie grafů (i když není pokryta v [5]) a je možno najít její obsáhlejší popis třeba v [3] nebo v [?].

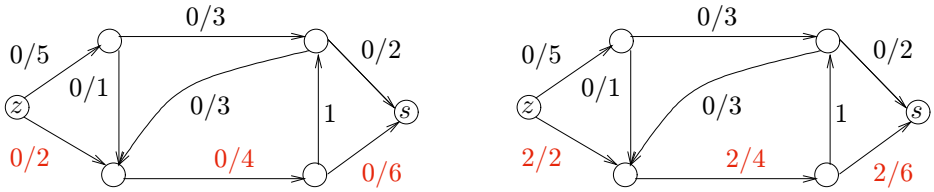
6.4 Cvičení: Příklady toků v sítích

Příklad 6.12. *Jaký je maximální tok touto sítí ze z do s?*

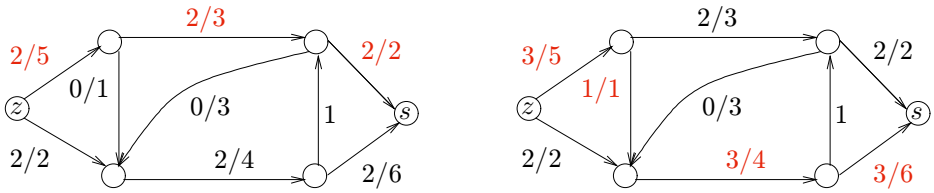


Na tomto příkladě si ukážeme průběh Algoritmu 6.6.

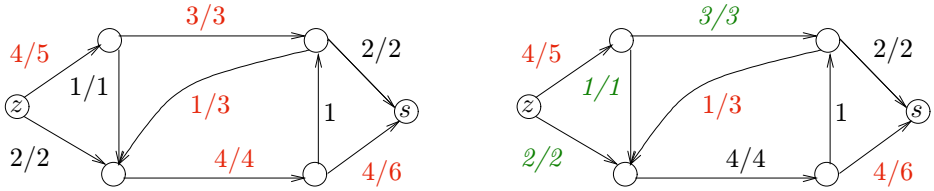
Začneme s nulovým tokem a najdeme nejkratší z nenasycených cest mezi z a s (vedoucí spodem grafu a vyznačenou na prvním obrázku). Minimální rezerva kapacity na této cestě je 2, takže tok nasytíme o 2 podél této cesty (viz obrázek vpravo).



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).



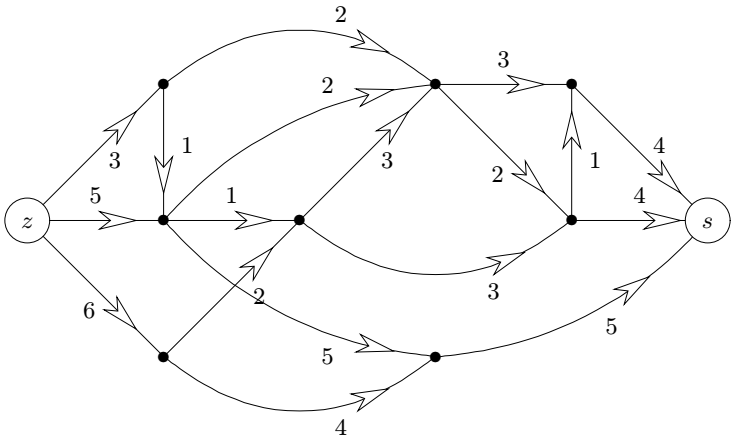
Obdobně ještě najdeme nenasycenou cestu délky 5, kterou také nasytíme o 1. Závěrečný obrázek nám ukazuje všechny nenasycené hrany, mezi kterými již nevede žádná nenasycená cesta ze z do s, takže máme maximální tok velikosti 6 v naší síti.



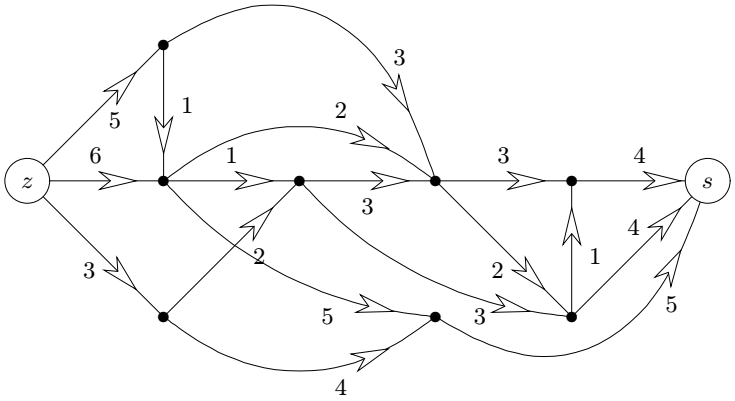
Zároveň je na posledním obrázku vyznačen i příslušný minimální řez velikosti 6. □

Úlohy k řešení

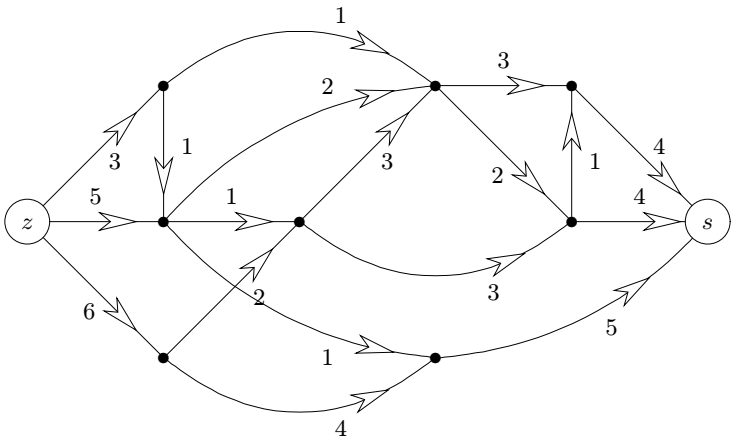
(6.4.1) *Najděte maximální tok v následující síti. (Kapacity hran jsou vyznačeny u svých šipek.)*
Tok do obrázku zapište, vyznačte hrany příslušného minimálního řezu a napište velikost toku.



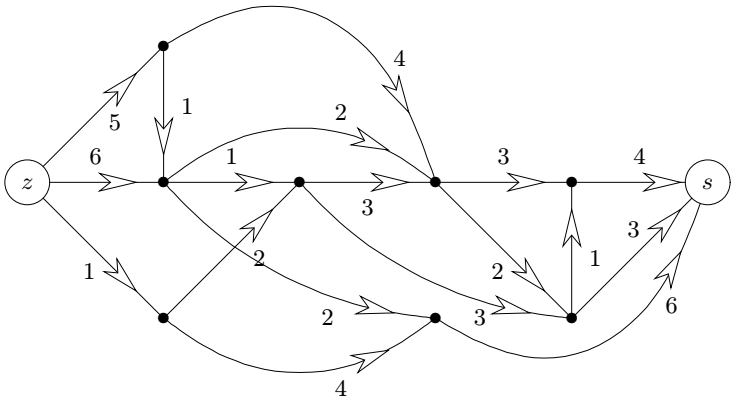
(6.4.2) *Najděte maximální tok v následující síti.*



(6.4.3) *Najděte maximální tok v následující síti.*



(6.4.4) *Najděte maximální tok v následující síti.*



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

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

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

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

7 Barevnost a další těžké problémy

Úvod

Již dříve jsme si připomínali jeden z historických kamenů teorie grafů – slavný problém sedmi mostů v Královci (dnešním Kaliningradě). Další neméně slavný problém pochází z poloviny 19. století a je obvykle zvaný problémem čtyř barev. Na rozdíl od sedmi mostů, problém čtyř barev zůstal nevyřešený po více než 100 let! A právě marné snahy o jeho vyřešení se zapříčinily o rozvoj mnoha oblastí teorie grafů a celé kombinatoriky.

Problém čtyř barev byl původně formulován pro politické mapy států: Výrobci map chtěli státy barevně odlišit, aby každé dva sousední měli různé barvy. Přitom chtěli mapy tisknout co nejlevněji, tedy s nejmenším možným počtem barev. Není problém nakreslit čtyři státy tak, že každý s každým sousedí, a proto 4 barvy jsou někdy potřebné. Praxe brzy ukázala, že 4 barvy vždy stačí, ale matematici si dlouho lámali hlavy s tím, proč tomu tak je...

Někteří matematici však kladou prapočátky teorie grafů daleko před Eulerovými sedmi mosty či problémem čtyř barev, až ke středověké otázce, zda lze šachovým koněm obejít celou šachovnici bez opakování. Tato otázka nakonec vede k dalšímu zajímavému a obtížnému problému tzv. Hamiltonovské kružnice v grafu, nazvané podle tvůrce příslušného hlavolamu z 19. století. I na tento specifický problém a jeho obtížnost se blíže podíváme.

Cíle

Prvním cílem této lekce je definovat barevnost grafů a naučit se s ní pracovat. Druhým cílem je ukázat několik dalších „obtížných" problémů definovaných přirozeně na grafech a přivést je do kontextu teorie výpočetní složitosti a $\mathcal{NP}$-úplnosti.

7.1 Barevnost grafu

Nejprve si uveďme pojem barevnosti – představme si, že hrany grafu nám říkají, že jejich koncové vrcholy musí být barevně odlišené (třeba proto, že reprezentují sousední státy, nebo proto, že jinak jsou si příliš podobné a je třeba je jinak rozlišit, atd). Samozřejmě bychom mohli každému vrcholu grafu dát jinou barvu, ale k čemu by pak takový problém byl? My bychom chtěli použít barev celkem co nejméně.

`Definice`: *Obarvením grafu* G pomocí k barev myslíme libovolné zobrazení

$$c : V(G) \to \{1, 2, \ldots, k\}$$

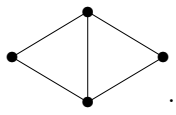
takové, že každé dva vrcholy spojené hranou dostanou různé barvy, tj. $c(u) \neq c(v)$ pro všechny $\{u, v\} \in E(G)$.

Definice 7.1. ***Barevnost*** grafu G
je nejmenší přirozené číslo $\chi(G)$ pro které existuje obarvení grafu G pomocí $\chi(G)$ barev.

Čísla $1, 2, \ldots, k$ z předchozí definice tak nazýváme barvami vrcholů (je to pohodlnější, než popisovat barvy běžnými jmény jako bílá, červená, atd).

`Poznámka`: Uvědomme si, že barevnost lze definovat pouze pro graf bez smyček, protože oba konce smyčky mají vždy stejnou barvu a nic s tím nenaděláme.

Příklad 7.2. *Určete barevnost grafu*



.

Na první pohled je vidět, že potřebujeme aspoň 3 barvy, neboť graf má trojici navzájem spojených vrcholů (trojúhelník). Na druhý pohled pak zjistíme, že levý a pravý vrchol spojené

hranou nejsou, a proto jim lze přiřadit stejnou barvu (a další dvě barvy vrchnímu a spodnímu vrcholu). Proto má graf barevnost 3. □

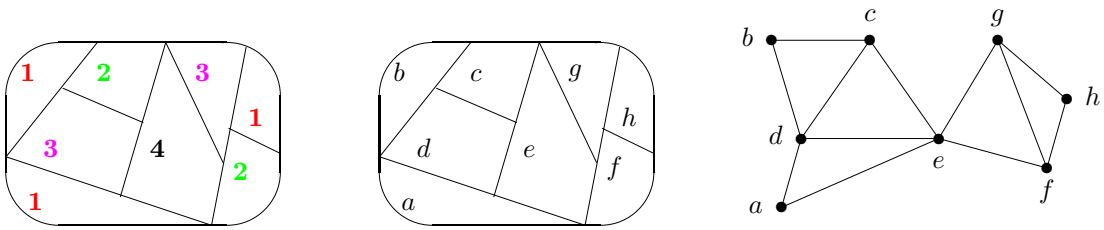
V obecnosti lze o barevnosti grafů říci následující.

Lema 7.3. *Nechť G je jednoduchý graf (bez smyček) na n vrcholech. Pak $\chi(G) \leq n$ a rovnost nastává právě když $G \simeq K_n$ je úplný graf.*

Důkaz: Stačí každý vrchol obarvit jinou barvou a máme skutečné obarvení n barvami dle definice. Navíc pokud některá dvojice u, v vrcholů není spojená hranou, můžeme volit lepší obarvení $c(u) = c(v) = 1$ a zbylé vrcholy různými barvami $2, 3, \ldots, n-1$, tj. pak $\chi(G) < n$. □

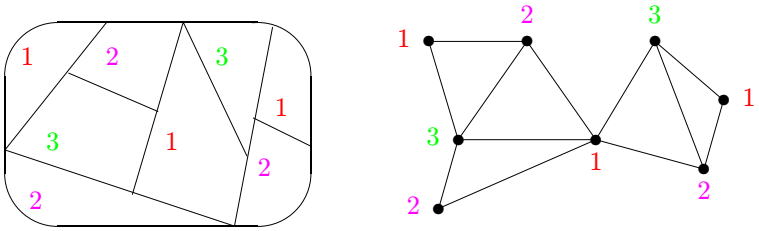
Příklad 7.4. *Vraťme se k příkladu „barvení“ mapy z úvodu lekce a ukažme si, jak mapy souvisejí s grafy a jejich barevností.*

Označme si státy na následující mapě písmeny $a, b, \ldots, h$.



Jednotlivé oblasti na mapě (předpokládáme, že každý stát má souvislé území, tj. státy = oblasti) prohlásíme za vrcholy našeho grafu a sousední dvojice států spojíme hranami. Nezapomeňme přitom, že “sousední” znamená sdílení celého úseku hranice, ne jen jednoho rohu. Výsledkem bude graf nakreslený vpravo. Podíváme-li se nyní zpátky na definici barevnosti grafu, vidíme, že se jedná přesně o ten samý problém jako u barvení původní mapy.

Při troše snahy také najdeme lepší obarvení uvedené mapy využívající pouhých tří barev:



□

Podívejme se, které grafy mají nízkou barevnost. Je jasné, že jedna barva stačí, jen pokud graf nemá hrany. Grafy obarvitelné dvěma barvami už tvoří zajímavější třídu a jsou obvykle nazývané jedním slovem *bipartitní*. Poměrně jednoduchý popis bipartitních grafů je uvedený v následujícím tvrzení.

Věta 7.5. *Graf G má barevnost 1 právě když nemá žádné hrany. Graf G má barevnost 2 právě když nemá žádnou kružnici liché délky jako podgraf.*

Důkaz: Pokud graf nemá hrany, můžeme všechny vrcholy obarvit stejnou barvou 1. Naopak pokud mají všechny vrcholy stejnou barvu, nemůže graf mít žádnou hranu.

Druhá část: Na jednu stranu, lichou kružnici nelze obarvit dvěma barvami, viz obrázek. Na druhou stranu si představme, že zvolíme libovolný vrchol v grafu G s barvou 1 a ostatní vrcholy obarvíme takto: Vrcholy, jejichž vzdálenost od v je lichá, obarvíme 2.

Vrcholy, jejichž vzdálenost od v je sudá, obarvíme 1. Stačí nyní dokázat, že jsme získali korektní obarvení.

Pokud bychom tak získali třeba dva vrcholy spojené hranou f v sudé vzdálenosti od v, získáme uzavřený sled S liché délky přes f a v. Stejně tak pro dva vrcholy v liché vzdálenosti. Vezmeme-li z S ty hrany, které se tam opakují lichý počet krát, dostaneme Eulerovský podgraf T lichého počtu hran. Jak již víme (Oddíl 4.1), T pak obsahuje kružnici a tudíž jej lze induktivně sestrojit jako hranově-disjunktní sjednocení kružnic. Avšak sjednocení kružnic sudé délky nevytvoří T liché velikosti, spor. Proto naše obarvení za daných předpokladů nemůže dát stejnou barvu sousedním vrcholům, a tudíž dvě barvy stačí. □

Poněkud nepříjemnou skutečností je, že o barvení a určování barevnosti grafů nemůžeme v obecnosti přesně říci o mnoho více, než jsme uvedli výše. Jedná se o problém algoritmicky ještě obtížnější než byl problém isomorfismu a nepředpokládá se, že by se barevnost grafu dala algoritmicky určit jinak, než hrubou silou probráním (téměř) všech možných obarvení.

Hladové obarvování

Přes všechnu svou obtížnost, za vhodných okolností má problém barvení grafů docela jednoduché přibližné hladové řešení (viz Oddíl 5.2).

`Definice`: Graf G je *k-degenerovaný*, pokud každý podgraf G obsahuje vrchol stupně nejvýše k.

> `Komentář:` Příkladem k-degenerovaného grafu je každý graf stupně nejvýše k, ale na druhou stranu k-degenerované grafy mohou mít vysoké stupně. (Nestačí však mít jen nízký nejmenší stupeň!)

Věta 7.6. *Každý k-degenerovaný graf lze hladově korektně obarvit $k+1$ barvami.*

`Důkaz`: Jelikož graf G je k-degenerovaný, vybereme libovolný jeho vrchol v_1 stupně nejvýše k a rekurzivní aplikací tohoto postupu obarvíme podgraf $G - v_1$, který je podle definice také k-degenerovaný. Nakonec si všimneme, že $\leq k$ sousedé vrcholu v_1 dostanou nejvýše k různých barev, takže v_1 dobarvíme zbylou barvou. □

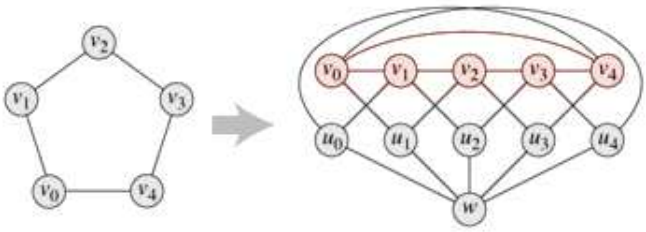
Důležité aplikace této věty uvidíme v příští lekci, avšak jedno zajímavé zesílení (*Brooksova věta*) si uvedeme nyní:

Věta 7.7. *Nechť G je souvislý jednoduchý graf maximálního stupně k. Pak $\chi(G) \leq k$ až na případy, kdy G je úplný graf nebo lichá kružnice.*

Grafy vysoké barevnosti

Ke správnému pochopení barevnosti grafu je nezbytné se zamyslet, které grafy mají vysokou barevnost. Jedná se například o grafy obsahující velké kliky (úplné podgrafy). Je to však vše? Není! Lze nalézt grafy s libovolně vysokou barevností neobsahující ani trojúhelníky.

Tvrzení 7.8. (Mycielski) *Graf získaný z grafu G následující konstrukcí (viz obrázek) má barevnost $\chi(G)+1$ a neobsahuje trojúhelníky, pokud je neobsahuje ani G.*

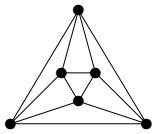


Nejobecněji lze říci následující překvapivé tvrzení:

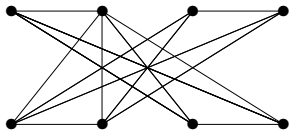
Věta 7.9. (Erdős) *Pro každá $c, r > 0$ existuje graf s barevností alespoň c a neobsahující kružnice kratší než r.*

Úlohy k řešení

(7.1.1) *Kolika nejméně barvami korektně obarvíte graf pravidelného osmistěnu?*



(7.1.2) *Kolik nejméně barev je třeba na korektní obarvení tohoto grafu? Příslušné obarvení vyznačte.*



(7.1.3) *Jaká je barevnost stromu?*

(7.1.4) *Kolik barev vždy stačí na korektní obarvení grafu vzniklého z kružnice délky* 2004 *přidáním jedné nové hrany?*

(7.1.5) *Představme si, že nějaký velký graf má všechny vrcholy stupně nejvýše* 101. *Kolika barvami takový graf jistě obarvíme?*

7.2 Variace na barevnost a jiné

Definice 7.10. ***Hranová barevnost*** grafu G.
Hledáme obarvení $c_e(E(G)) \to \{1, 2, \ldots, k\}$ takové, že žádné dvě hrany se společným vrcholem nedostanou stejnou barvu.

Nejmenší možný počet barev k, pro které hranové obarvení existuje, se nazývá *hranová barevnost* $\chi_e(G)$ grafu.

Na rozdíl od běžné barevnosti umíme hranovou barevnost docela ostře aproximovat.

Věta 7.11. (Vizing) *Pro každý jednoduchý graf platí $\Delta(G) \leq \chi_e(G) \leq \Delta(G)+1$.*

`Komentář:` Platí, že většina grafů splňuje $\Delta(G) = \chi_e(G)$. Umíte jednoduše sestrojit (a dokázat) příklady pro druhý případ?

Problém přesného určení hranové barevnosti grafu však stále zůstává algoritmicky velmi obtížný a také úzce souvisí s problémem čtyř barev.

Definice 7.12. ***Výběrová barevnost*** grafu G.
Je dán graf G spolu s přiřazenými „seznamy barev“ $L : V(G) \to \binom{\mathbb{N}}{k}$ (k-prvkové podmnožiny). Nyní hledáme obarvení $c_{ch}(V(G)) \to \mathbb{N}$ takové, že žádné dva sousední vrcholy nedostanou stejnou barvu a navíc $c_{ch}(v) \in L(v)$ pro každý vrchol v.

Nejmenší možná délka k seznamů barev, pro kterou výběrové obarvení vždy existuje (tj. pro každou možnou takovou volbu seznamů), se nazývá *výběrová barevnost* $ch(G)$ grafu.

Výběrová barevnost může (kupodivu!) být libovolně „vzdálena“ běžné barevnosti.

Tvrzení 7.13. *Pro každé k nalezneme bipartitní graf s výběrovou barevností větší než k.*

`Fakt`: Hranová výběrová barevnost úplných bipartitních grafů úzce souvisí se známým problémem *latinských obdélníků*.

Hamiltonovské grafy

`Definice`: Kružnice C obsažená v grafu G se nazývá *Hamiltonovská*, pokud C prochází všemi vrcholy G. Obdobně mluvíme o *Hamiltonovské cestě* P v G, pokud cesta $P \subset G$ prochází všemi vrcholy G.

Graf G je *Hamiltonovský*, pokud obsahuje Hamiltonovskou kružnici.

Možná to zní překvapivě, ale i problém Hamiltonovské kružnice úzce souvisí s řešením problému čtyř barev.

Věta 7.14. (Dirac) *Každý graf na $n \geq 3$ vrcholech s minimálním stupněm $\geq n/2$ je Hamiltonovský.*

`Důkaz` (náznak): Nechť P je nejdelší cesta v grafu G s vrcholy po řadě $u_0, u_1, \ldots, u_k$. Podle její maximality leží každý soused u_0 i u_k na P. Pak existuje $0 < i < k$ takové, že $u_0u_{i+1} \in E(G)$ a zároveň $u_ku_i \in E(G)$. Pak $u_0u_{i+1}\,P\,u_ku_i\,P$ tvoří kružnici v G a snadno plyne, že se jedná o Hamiltonovskou kružnici. □

7.3 $\mathcal{NP}$-úplnost grafových problémů

Zatím prakticky všechny grafové problémy probírané v minulých lekcích (s drobnou výjimkou isomorfismu) byly řešitelné rozumně rychlými algoritmy. V této lekci se situace radikálně obrací... Studující odkazujeme na formální definici $\mathcal{NP}$-úplnosti z oblasti algoritmické složitosti.

> `Komentář:` Definice třídy $\mathcal{NP}$ se týká výhradně rozhodovacích problémů (s „ANO/NE“ odpovědí). Dá se neformálně říci, že problém patří do třídy $\mathcal{NP}$, pokud jeho odpověď ANO lze prokázat (ve smyslu “uhodnout a ověřit”) výpočtem, který běží v polynomiálním čase. $\mathcal{NP}$-úplné problémy jsou zhruba řečeno ty, které ve třídě $\mathcal{NP}$ mají nejvyšší obtížnost řešení.

Nyní si ukážeme vhodnými převody, že oněch „nejobtížnějších“ ($\mathcal{NP}$-úplných) problémů je v teorii grafů mnoho, bohužel by se dalo říci většina. To ostatně ukazuje, proč jsme zatím v praxi tak málo úspěšní při počítačovém řešení mnohých praktických problémů – přesné a efektivní řešení $\mathcal{NP}$-úplných úloh se totiž všeobecně považuje za nemožné.

Problém 7.15. ***3-SAT (splnitelnost logických formulí ve spec. verzi)***
Následující problém je $\mathcal{NP}$-úplný:

Vstup: Logická formule Φ v konjunktivním normálním tvaru taková, že každá klauzule obsahuje nejvýše 3 literály.

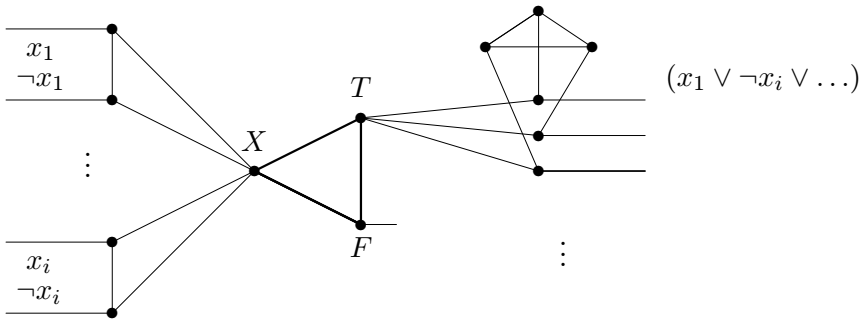
Výstup: Existuje logické ohodnocení proměnných tak, aby výsledná hodnota Φ byla 1 (pravda)?

Pomocí tohoto základního a pro převody velmi vhodného $\mathcal{NP}$-úplného problému snadno ukážeme $\mathcal{NP}$-úplnost mnoha běžných grafových problémů.

Problém 7.16. ***3-COL (3-obarvení grafu)***
Následující problém je $\mathcal{NP}$-úplný:

Vstup: Graf G.
Výstup: Lze vrcholy G korektně obarvit 3 barvami?

`Důkaz` (náznak): Ukážeme si polynomiální převod z problému 3-SAT. Sestrojíme graf G pro danou formuli Φ. Základem grafu je trojúhelník, jehož vrcholy označíme X, T, F (přitom barvy přiřazené vrcholům T, F budou reprezentovat logické hodnoty). Každé proměnné x_i ve Φ přiřadíme dvojici vrcholů spojených s X. Každé klauzuli ve Φ přiřadíme podgraf na 6 vrcholech (z nichž tři jsou spojené s T), jako na obrázku. Nakonec volné "půlhrany" z obrázku pospojujeme dle toho, jaké literály vystupují v klauzulích.



Například první půlhrana naznačené klauzule na obrázku je napojena na půlhranu značenou x_1 vlevo, druhá půlhrana na půlhranu značenou $\neg x_i$ vlevo, atd. Pokud v klauzuli chybí třetí literál, je jeho půlhrana napojena přímo na F vpravo.

Pak G má 3-obarvení právě když je Φ splnitelná, jak si lze ověřit na obrázku. Tím jsme sestrojili polynomiální převod formule Φ z problému 3-SAT na graf G v problému 3-obarvení. $\mathcal{NP}$-úplnost nyní vyplývá z Lemat **??** a 7.15. □

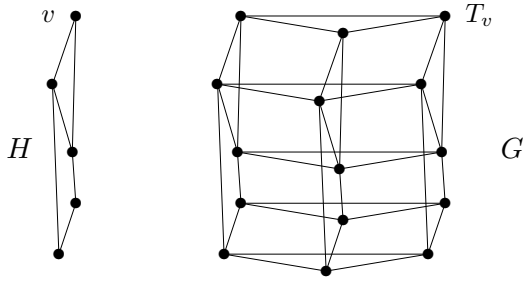
Kromě barevnosti grafu je $\mathcal{NP}$-úplných mnoho problémů ptajících se na výběry vrcholů v grafu s jistými vlastnostmi.

Problém 7.17. ***IS (nezávislá množina)***
Následující problém je $\mathcal{NP}$-úplný:

Vstup: Graf G a přirozené číslo k.
Výstup: Lze v G najít nezávislou podmnožinu velikosti (aspoň) k?

`Důkaz`: Ukážeme polynomiální převod z problému 3-COL. Nechť H je graf na n vrcholech, který je za úkol obarvit třemi barvami. Položíme $k = n$ a graf G sestrojíme ze tří disjunktních kopií grafu H tak, že vždy tři kopie každého jednoho vrcholu $v \in V(H)$

spojíme hranami do trojúhelníku T_v v G.



Pokud $c : V(H) \to \{1, 2, 3\}$ je obarvení H třemi barvami, v grafu G lze vybrat $k = n$ nezávislých vrcholů tak, že pro každý $v \in V(H)$ vezmeme $c(v)$-tou kopii vrcholu v v grafu G. (Nakreslete si v obrázku!) Naopak pokud I je nezávislá množina v grafu G o velikosti $k = n$, pak z každého trojúhelníku T_v, $v \in V(H)$ náleží do I právě jeden vrchol. Podle toho již určíme jednu ze tří barev pro vrchol v v H. □

Problém 7.18. ***VC (vrcholové pokrytí)***
Následující problém je $\mathcal{NP}$-úplný:

Vstup: Graf G a přirozené číslo k.
Výstup: Lze v G najít *vrcholové pokrytí*, tj. množinu $C \subseteq V(G)$ takovou, že každá hrana G má alespoň jeden konec v C, o velikosti nejvýše k?

`Důkaz`: Problém vrcholového pokrytí jasně patří do $\mathcal{NP}$ a jeho úplnost je dokázána polynomiálním převodem z Příkladu 7.26. □

Problém 7.19. ***DOM (dominující množina)***
Následující problém je $\mathcal{NP}$-úplný:

Vstup: Graf G a přirozené číslo k.
Výstup: Lze v G najít dominující množinu, tj. množinu $D \subseteq V(G)$ takovou, že každá vrchol G má některého souseda v D, o velikosti nejvýše k?

`Důkaz` (náznak): Problém dominující množiny jasně patří do $\mathcal{NP}$ a jeho úplnost je dokázána polynomiálním převodem z Příkladu 7.27. □

Další předvedené problému se týkají procházení grafem (pokud možno) bez opakování vrcholů.

Problém 7.20. ***HC (Hamiltonovský cyklus)***
Následující problém je $\mathcal{NP}$-úplný:

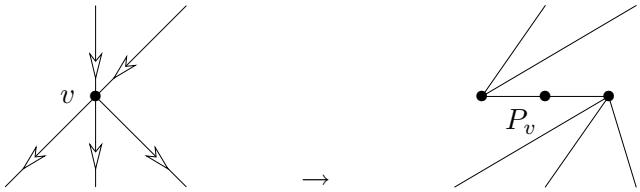
Vstup: Orientovaný graf G.
Výstup: Lze v G najít orientovanou kružnici (cyklus) procházející všemi vrcholy?

Tento převod pro jeho technickou obtížnost vynecháváme..

Problém 7.21. ***HK (Hamiltonovská kružnice)***
Následující problém je $\mathcal{NP}$-úplný:

Vstup: Graf G.
Výstup: Lze v G najít kružnici procházející všemi vrcholy?

Důkaz:



Použijeme snadný převod z předchozího problému HC. Každý vrchol v orientovaného grafu H nahradíme třemi vrcholy tvořícími cestu P_v délky 2 v grafu G. Orientované hrany grafu H přicházející do v pak přivedeme do prvního vrcholu cesty P_v, hrany odcházející z v naopak vedeme z posledního vrcholu cesty P_v. □

Problém 7.22. ***TSP (problém obchodního cestujícího)***
Následující problém je $\mathcal{NP}$-úplný:

Vstup: Souvislý graf G s nezáporným ohodnocením hran ("délkou") a číslo r.
Výstup: Lze v G najít uzavřený sled procházející všemi vrcholy a mající součet délek hran (včetně opakovaných) nejvýše roven r?

Důkaz: Použijeme snadný převod z předchozího problému HK. Každou hranu ohodnotíme délkou 1 a položíme $r = n$, kde n je počet vrcholů našeho grafu. Pak uzavřený sled délky n se nesmí opakovat v žádném vrcholu ani hraně, aby prošel všemi vrcholy, a proto musí být kružnicí. □

Úlohy k řešení

(7.3.1) *Zdůvodněte, proč je následující problém ("orientovaná dominující množina") $\mathcal{NP}$-úplný: Vstupem je orientovaný graf G a přirozené číslo k. Lze v grafu G najít podmnožinu $D \subseteq V(G)$ s nejvýše k vrcholy takovou, že každý vrchol w grafu G náleží do D nebo do w vede orientovaná hrana (šipka) z některého vrcholu v D?*

Návod: Použijte třeba polynomiální převod z problému vrcholového pokrytí.

7.4 Příběh problému vrcholového pokrytí

Komentář: Bylo nebylo, jednou se dva slovutní informatici při surfování na moři začali zabývat otázkou, proč dva tak „podobné" problémy jako vrcholové pokrytí a dominující množina mají (přestože oba $\mathcal{NP}$-úplné) tak rozdílné algoritmické chování... Ale dost pohádek, více konkrétních informací čtenář najde v [R. Downey, M. Fellows, Parametrized complexity, Springer 1999]. Mimo jiné se dozví, že tato zdánlivě okrajová otázka dala vzniknout zcela novému pohledu na výpočetní složitost problémů, který jde „jaksi mimo" klasickou polynomiální hierarchii a umožňuje docela rozumně řešit i některé problémy, které jsou jinak $\mathcal{NP}$-těžké.

Takže v čem spočívá zásadní rozdíl v našich znalostech o řešení problémů dominující množiny a vrcholového pokrytí?

- Pokud se v analýze zaměříme na hodnotu parametru k vstupu, tak dominující množinu velikosti k stejně nedokážeme nalézt rychleji než probráním prakticky všech k-tic vrcholů grafu G. To je i pro malé fixní hodnoty k, třeba $k = 10, 20$ v praxi neproveditelné.

- Avšak vrcholové pokrytí velikosti k dokážeme nalézt jednoduchým algoritmem v čase $O(2^k \cdot n)$, což pro malé fixní hodnoty k dává skvěle použitelný lineární algoritmus!

Algoritmus 7.23. *k-VC (vrcholové pokrytí)*
Pro fixní k vyřešíme následující problém.

Vstup: Graf G.
Výstup: Lze v G najít vrcholové pokrytí o velikosti nejvýše k?

Pro inicializaci položíme $C = \emptyset$ a $F = E(G)$.

- *Pokud $F = \emptyset$, vracíme C jako vrcholové pokrytí.*
  Jinak pokud $|C| \geq k$, vracíme odpověď NE.
- *Vybereme libovolnou hranu $f = uv \in F$ a pro oba její konce $x = u, v$ uděláme:*
  - *$C' = C \cup \{x\}$ a F' vytvoříme z F odebráním všech hran vycházejících z vrcholu x v G.*
  - *Rekurzivně zavoláme tento algoritmus pro G, C' a F'.*

Kolik tento algoritmus provede rekurzivních volání celkem? Každý průchod generuje dvě další volání, ale jen do fixní hloubky k, takže ve výsledku bude čas výpočtu jen $O(2^k \cdot n)$.

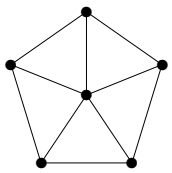
`Poznámka`: Dnes je již známo, že faktor 2^k lze promyšlenějším přístupem „vylepšit“ na mnohem menší základ mocniny. (2006: 1.2738^k)

Rozšiřující studium

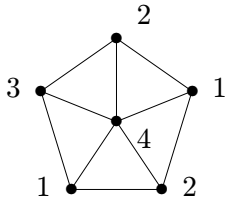
Problematika rovinného kreslení grafů a jejich barvení je pokryta v [5, Kapitola 5]. Zájemci o podrobný popis historie problému čtyř barev zase mohou mnoho zajímavých informací nalézt na internetu. Pro případné hlubší studium složitosti a $\mathcal{NP}$-úplnosti grafových problémů odkazujeme na [3].

7.5 Cvičení: Příklady o barevnosti grafů

Příklad 7.24. *Určete a zdůvodněte barevnost tohoto grafu:*



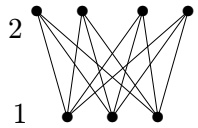
Vidíme, že obvodová kružnice tohoto grafu má délku 5, což je liché číslo, a proto je potřeba na její korektní obarvení použít aspoň tři barvy $1, 2, 3$. Pak je ale středový vrchol spojený s každou z těchto barev $1, 2, 3$, tudíž pro něj potřebujeme čtvrtou barvu 4.



To znamená, že daný graf má barevnost 4. □

Příklad 7.25. *Kolik nejméně hran musíte vypustit z úplného grafu na 7 vrcholech, aby se výsledný graf dal korektně obarvit dvěma barvama?*

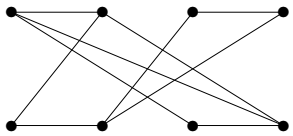
Musíme vypustit všechny hrany mezi vrcholy, kterým chceme dát stejnou barvu, aby výsledné obarvení bylo korektní. Jelikož všechny vrcholy úplného grafu jsou si ekvivalentní, stačí se rozhodnout, kolik z nich dostane barvu 1 a kolik barvu 2. Jak snadno zjistíme, optimální je barvy rozdělit napůl, tj. 3 a 4 v tomto případě. Je tedy potřeba vypustit nejméně $\binom{3}{2} + \binom{4}{2} = 3 + 6 = 9$ hran.



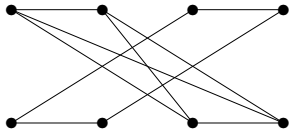
□

Úlohy k řešení

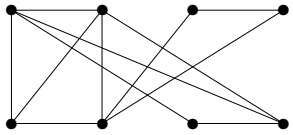
(7.5.1) *Kolik nejméně barev je třeba na korektní obarvení tohoto grafu? Příslušné obarvení vyznačte.*



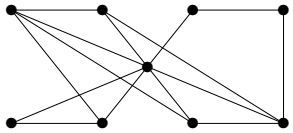
(7.5.2) *Kolik nejméně barev je třeba na korektní obarvení tohoto grafu? Příslušné obarvení vyznačte.*



(7.5.3) *Obarvěte následující graf na 8 vrcholech třemi barvami $1, 2, 3$ tak, aby se barva 3 použila co nejvíce krát.*



(7.5.4) *Obarvěte následující graf na 9 vrcholech třemi barvami $1, 2, 3$ tak, aby se barva 3 použila co nejméně krát.*



*(7.5.5)** *Jaká může být barevnost grafu, který vznikne z úplného grafu na 15 vrcholech vypuštěním tří hran? (Je to jednoznačné?)*

*(7.5.6)** *Zdůvodněte, proč na obarvení grafu vzniklého z libovolného stromu přidáním dvou libovolných nových hran vždy stačí 3 barvy.*

7.6 Apendix: Další $\mathcal{NP}$-úplné grafové problémy

V tomto (nepovinném) dodatku si jednak probereme příklady polynomiálních převodů mezi některými základními grafovými problémy. Za druhé si ukážeme, jak u různých problémů poznat, zda náleží do třídy $\mathcal{NP}$ nebo ne a zda případně jsou $\mathcal{NP}$-úplné.

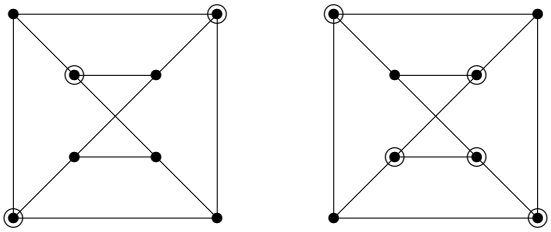
Polynomiální převody grafových problémů

`Komentář:` Neformálně řečeno, polynomiálním převodem problému P_1 na problém P_2 nazveme postup, který v polynomiálním čase ze známého způsobu řešení problému P_2 „získá“ řešení pro problém P_1.

Příklad 7.26. *Problém nezávislé množiny se ptá, zda v grafu existuje podmnožina k vrcholů nespojených žádnými hranami. Problém vrcholového pokrytí se ptá, zda v grafu existuje podmnožina m vrcholů dotýkajících se všech hran. (Vrchol se dotýká hrany, pokud je jejím koncem.) Ukažme si podrobně, jak se problém nezávislé množiny polynomiálně převede na problém vrcholového pokrytí.*

Nejprve si ujasněme, co je vstupem a výstupem kterého problému. Pro nezávislou množinu je vstupem dvojice G, k, kde G je graf a k přirozené číslo. Pro vrcholové pokrytí je obdobně vstupem dvojice G, m. (V případě nezávislé množiny se přirozeně snažíme dostat co největší vyhovující k, kdežto u vrcholového pokrytí co nejmenší m.) V obou případech se jedná o rozhodovací problémy, takže odpovědí je ANO/NE.

Všimněme si následujícího jednoduchého faktu: Pokud $I \subset V(G)$ je nezávislá množina v grafu G, pak žádná hrana G nemá oba konce v I. To ale znamená, že doplněk množiny $J = V(G) \setminus I$ se dotýká všech hran grafu G, a tudíž J je vrcholovým pokrytím. Pokud $|I| = k$, pak $|J| = |V(G)| - k = m$. Naopak doplňkem vrcholového pokrytí J je ze stejného důvodu nezávislá množina $I = V(G) \setminus J$. Příklad je na následujícím obrázku. (Zakreslete si to také do některého vlastního grafu.)



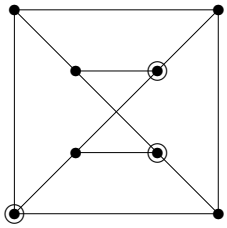
nezávislá množina vrcholové pokrytí

Takže stačí vstup G, k problému nezávislé množiny převést na vstup G, m, $m = |V(G)| - k$ problému vrcholového pokrytí, ze kterého už získáme správnou odpověď i na původní problém. Tento převod dokonce spočítáme v konstantním čase, jen provedeme jedno odečtení. (Všimněme si ještě jedné zajímavosti – při našem převodu se vůbec nezměnil graf G, jen číslo k na m, ale to je pouze specifickou vlastností tohoto jednoduchého převodu. Ve složitějších případech však dochází ke změně celého vstupu, včetně grafu.) □

Příklad 7.27. *Problém dominující množiny se ptá, zda v grafu existuje podmnožina m vrcholů taková, že každý jiný vrchol je s některým z nich spojený hranou. Ukažme si podrobně, jak se problém vrcholového pokrytí polynomiálně převede na problém dominující množiny na souvislých grafech.*

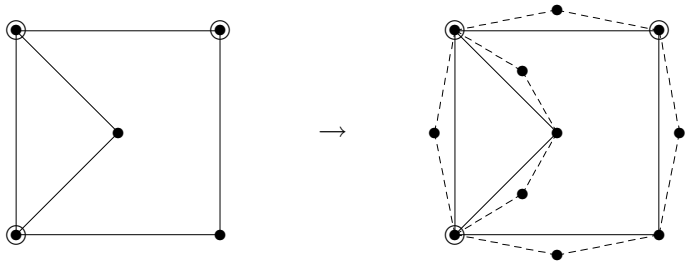
Definice dominující množiny je velmi podobná vrcholovému pokrytí, že? Vlastně by mělo stačit jaksi ke každé hraně daného grafu G (ve kterém hledáme vrcholové pokrytí) přiřadit nějaký nový vrchol, který bude třeba po převodu dominovat. Abychom se vyhnuli patologickým případům, předpokládáme souvislé grafy s více než jedním vrcholem.

Začneme jednoduchou ukázkou, jak třeba dominující množina může vypadat.



(Dokážete odpovědět, proč graf z obrázku nemá dominující množinu velikosti 2?)

Nyní přejdeme k popisu naznačeného převodu ze vstupu G, m problému vrcholového pokrytí na vstup H, m' problému dominující množiny.



Graf H vytvoříme z grafu G přidáním, pro každou hranu $e \in E(G)$, nového vrcholu v_e spojeného hranami do obou koncových vrcholů hrany e. (Tak se vlastně z každé hrany stane trojúhelník s třetím novým vrcholem, viz naznačený obrázek.) Číselný parametr $m' = m$ zůstane tentokrát nezměněn.

Abychom zdůvodnili, že se skutečně jedná o převod problémů, musíme dokázat implikace v obou směrech: Že vrcholové pokrytí $C \subseteq V(G)$ je zároveň dominující množinou v novém grafu H a že dominující množina $D \subseteq V(H)$ vytváří i stejně velké vrcholové pokrytí v původním grafu G. První část je snadná, podle definice vrcholového pokrytí každá hrana e grafu G má některý konec u v množině C, takže jak druhý konec hrany e, tak i nově přidaný vrchol v_e v grafu H jsou dominovány z vrcholu $u \in C$. Navíc z předpokladu souvislosti G plyne, že žádné další izolované vrcholy v H nejsou, takže C je zároveň dominující množinou v H.

Naopak vezměme dominující množinu $D \subseteq V(H)$ v novém grafu H. (Pozor, nelze hned říci, že by D byla vrcholovým pokrytím v G, neboť D může obsahovat přidané vrcholy, které v G nebyly.) Definujeme novou množinu $D' \subseteq V(G)$ takto: Pokud $w \in D \cap V(G)$, pak $w \in D'$. Jinak pro $w \in D$, kde $w = v_e$ byl přidán pro hranu $e \in E(G)$, dáme do D' libovolný z konců hrany e. Potom $|D'| \leq |D|$ a D' je vrcholovým pokrytím v původním grafu G, neboť pro každou hrany $e \in E(G)$ je přidaný vrchol $v_e \in V(H)$ dominován v grafu H množinou D, a tudíž hrana e bude mít některý konec v D'.

Dokázali jsme tedy, že se jedná o převod problému, a zbývá zdůvodnit, že tento převod je spočítán v polynomiálním čase. Pokud G má n vrcholů, má nejvýše $O(n^2)$ hran, a proto nový graf H má velikost $O(n^2)$ a v takovém čase jsme snadno schopni jej sestrojit. Je to polynom v n. □

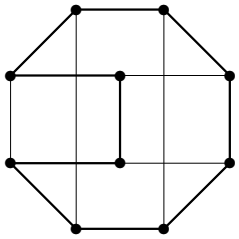
Problémy ve třídě $\mathcal{NP}$

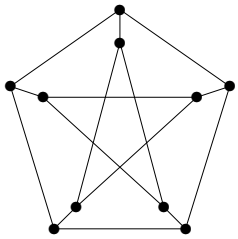
Nyní se budeme věnovat otázkám, proč některé grafové problémy náleží či nenáleží do třídy $\mathcal{NP}$.

`Komentář:` Definice třídy $\mathcal{NP}$se týká výhradně rozhodovacích problémů (s „ANO/NE“ odpovědí). Dá se neformálně říci, že problém patří do třídy $\mathcal{NP}$, pokud jeho odpověď ANO lze prokázat (ve smyslu “uhodnout a ověřit”) výpočtem, který běží v polynomiálním čase.

Příklad 7.28. *Hamiltonovská kružnice v grafu G je takový podgraf, který je isomorfní kružnici a přitom obsahuje všechny vrcholy G. (Jinak řečeno, kružnice procházející každým vrcholem jednou.) Proč patří do třídy $\mathcal{NP}$ problém poznat, zda daný graf G obsahuje Hamiltonovskou kružnici?*

Jak již bylo řečeno výše u definice, třída $\mathcal{NP}$ je vlastně třídou těch problémů, kde odpověď ANO lze ověřit efektivně s vhodnou nápovědou. Jestliže se ptáme na existenci Hamiltonovské kružnice v grafu G, přirozeně se jako nápověda nabízí právě ona kružnice. Pro ilustraci ukazujeme příklady dvou grafů, kde v prvním je Hamiltonovská kružnice vyznačena tlustě, kdežto ve druhém neexistuje.





Jak ale Hamiltonovskou kružnici popíšeme a jak ověříme, že se skutečně jedná o Hamiltonovskou kružnici? Obojí musíme zvládnout v polynomiálním čase!

Jako popis Hamiltonovské kružnice se přirozeně nabízí zadat tu permutaci vrcholů grafu G, v jejímž pořadí dotyčná kružnice vrcholy prochází. (Tím neříkáme, že by nebyly jiné způsoby popisu, jen že tento se nám hodí.) Takže nápovědu zadáme jednoduše polem $k[\,]$ délky n, kde n je počet vrcholů G. Pro ověření, že se jedná o Hamiltonovskou kružnici, stačí zkontrolovat, že $k[i] \neq k[j]$ pro různá i, j a že vždy $\{k[i], k[i+1]\}$ je hranou v grafu G pro $i = 1, 2, \ldots, n-1$ a také $\{k[1], k[n]\}$ je hranou. Při vhodné implementaci maticí sousednosti grafu G to zvládneme vše zkontrolovat v lineárním čase, ale i při jiných implementacích nám stačí čas $n \cdot O(n) = O(n^2)$, což je skutečně polynomiální. Proto problém existence Hamiltonovské kružnice patří do třídy $\mathcal{NP}$. □

Příklad 7.29. *Patří do třídy $\mathcal{NP}$ problém poznat, zda daný graf G obsahuje nejvýše čtyři Hamiltonovské kružnice?*

Čtenář opět může navrhnout, že vhodnou nápovědou pro příslušnost do třídy $\mathcal{NP}$ jsou ony čtyři Hamiltonovské kružnice v grafu. To lze přece snadno ověřit stejně jako v předchozím příkladě. Skutečně tomu tak je?
Není!!! My sice dokážeme ověřit, že napověděné čtyři kružnice v grafu jsou Hamiltonovské, ale nijak tím neprokážeme, že více Hamiltonovských kružnic v grafu není. Takové ověření by nakonec bylo stejně obtížné, jako nalezení Hamiltonovské kružnice samotné.

Proto na základě současných znalostí teoretické informatiky *nelze tvrdit*, že by popsaný problém náležel do třídy $\mathcal{NP}$. Avšak pokud bychom otázku negovali, tj. ptali se, zda graf G obsahuje více než čtyři Hamiltonovské kružnice, tak by už problém do třídy $\mathcal{NP}$ náležel. (Napověděli bychom některých pět Hamiltonovských kružnic.) Proto vidíte, jak je důležité správně se v zadání problému ptát. □

$\mathcal{NP}$-úplnost a její důkazy

Dále si zopakujme stručný neformální návod, jak by vlastně běžný polynomiální převod pro zdůvodnění $\mathcal{NP}$-úplnosti problému měl vypadat...

`Komentář:` Dle definice je $\mathcal{NP}$-úplný problém takový, že jeho efektivní vyřešení v polynomiálním čase by poskytlo podobná řešení i pro všechny ostatní problémy ve třídě $\mathcal{NP}$.

Předpokládejme, že o problému P již víme, že je $\mathcal{NP}$-úplný, a o problému Q to chceme dokázat. Neboli chceme ukázat, že každý vstup problému P bychom uměli rozřešit převodem na případ problému Q, a tudíž i problém Q musí být tak těžký jako P.

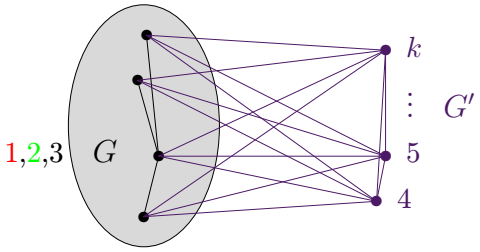
Takže v jednoduchých případech stačí vzít obecný (libovolný) vstup $\mathcal{V}$ známého těžkého problému P a "přeložit" jej na vstup $\mathcal{U}$ pro problém Q tak, že Q odpoví ANO na nový vstup $\mathcal{U}$ právě tehdy, když P odpověděl ANO na $\mathcal{V}$. (Všimněte si dobře, že musíte dokázat logickou ekvivalenci, tedy že z odpovědi ANO v Q vyplývá ANO v P i naopak!)

Příklad 7.30. *Problém k-obarvení grafu se ptá, zda daný graf lze obarvit k barvami tak, aby žádná hrana nespojovala dva vrcholy stejné barvy. (Skutečná barevnost našeho grafu může být i menší než k, o to v problému nejde.) Dokažme, že problém k-obarvení je $\mathcal{NP}$-úplný pro každé (i fixní) $k \geq 3$.*

Nejprve musíme zdůvodnit, že problém patří do třídy $\mathcal{NP}$. To je snadné, neboť nápovědou nám je ono obarvení grafu G pomocí k barev. Napověděné obarvení snadno zkontrolujeme v počtu kroků úměrném počtu hran grafu G, tedy polynomiálně v počtu vrcholů bez ohledu na hodnotu k.

Na druhou stranu už víme z Problému 7.16, že 3-obarvení grafu je $\mathcal{NP}$-úplné. Stačí nám tedy nalézt polynomiální převod z problému 3-obarvení na problém k-obarvení grafu. Předpokládejme tedy, že $k > 3$ a že je dán graf G, o kterém se ptáme, zda jej lze obarvit 3 barvami. My sestrojíme graf G' přidáním $k-3$ nových vrcholů ke grafu G spojených každý se všemi ostatními vrcholy. Proč to děláme?

To je zřejmé, v grafu G' všechny nově přidané vrcholy musí mít různých barvu od všech ostatních, takže pokud původní graf G šel obarvit 3 barvami, půjde tak i graf G' obarvit k barvami. Naopak pokud G' je obarven k barvami, všechny barvy nových $k-3$ vrcholů jsou jiné a různé od barev všech původních vrcholů, takže na původní vrcholy G nám zbudou jen $k-(k-3)=3$ různé barvy, což je přesně problém 3-obarvení na G. (Neboli 3-obarvení G jednoznačně odpovídají k-obarvením G'.) Schematickým obrázkem:



Vidíme tedy, že jsme našli polynomiální převod ze 3-obarvení grafu G na k-obarvení grafu G', a proto je problém k-obarvení $\mathcal{NP}$-těžký. Celkem dostáváme, že k-obarvení je $\mathcal{NP}$-úplné. □

Příklad 7.31. *Hamiltonovská cesta v grafu je takový podgraf, který je isomorfní cestě a prochází všemi vrcholy grafu. (Obdoba Hamiltonovské kružnice.) Dokažme, že problém zjištění existence Hamiltonovské cesty v daném grafu G je $\mathcal{NP}$-úplný.*

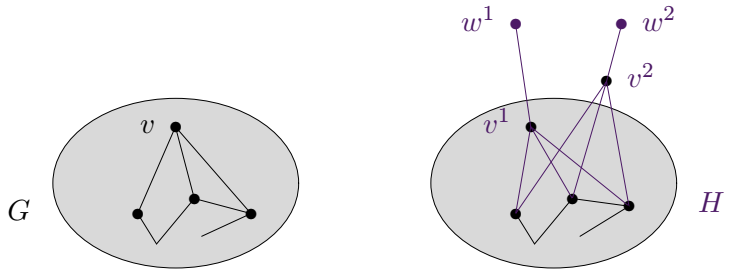
Již víme z Problému 7.21, zjištění existence Hamiltonovské kružnice je $\mathcal{NP}$-úplné. Problém Hamiltonovské cesty taktéž náleží do $\mathcal{NP}$, snadnou nápovědou existence je ukázat onu Hamiltonovskou cestu. Pro důkaz $\mathcal{NP}$-úplnosti využijeme polynomiální převod Hamiltonovské kružnice na Hamiltonovskou cestu.

Názorně řečeno, potřebujeme převést daný graf G na jiný graf H tak, že Hamiltonovská kružnice v G se stane Hamiltonovskou cestou v H a naopak. Na první pohled by se toto zdálo jako snadný cíl – přece z každé Hamiltonovské kružnice uděláme cestu odebráním hrany či vrcholu. Velký problém je však v onom slůvku "naopak", my také

musíme zajistit, že každá Hamiltonovská cesta v H vytvoří Hamiltonovskou kružnici v G! Proto nějakým způsobem musíme fixovat počátek a konec Hamiltonovské cesty v H tak, aby se daly spojit do kružnice v G.

Jednou z možností je vybrat jakýkoliv vrchol v v G, "zdvojit" jej (tj. přidat další vrchol se stejnými sousedy jako v) a navíc ke každé v^i ze zdvojených kopií v přidat novou hranu vedoucí do nového vrcholu w^i stupně 1. Tyto přidané vrcholy w^1, w^2 pak nutně musí být konci Hamiltonovské cesty, pokud ona existuje (jinak se do vrcholů stupně 1 přece dostat nedá).

Schematickým obrázkem:



□

Příklad 7.32. *Pro jaké k je $\mathcal{NP}$-úplný problém zjistit, zda v daném grafu existuje nezávislá množina velikosti k? (Nezávislá množina je taková podmnožina vrcholů grafu, z níž žádné dva její vrcholy nejsou spojené hranou.) Je tento problém $\mathcal{NP}$-úplný pro fixní k nebo pro proměnné hodnoty k?*

Tento příklad uvádíme proto, aby si čtenář dobře uvědomil roli *parametrů problému*, které jsou fixní, a těch, které jsou proměnlivé, neboli dané na vstupu.

Problém existence nezávislé množiny velikosti k totiž snadno rozřešíme v čase $O(n^{k+1})$ – prostě projdeme hrubou silou všechny k-tice vrcholů grafu a pokaždé se podíváme, zda náhodou netvoří nezávislou množinu. Čas $O(n^{k+1})$ je pochopitelně polynomiální pro každé fixní k, takže pak tento problém těžko může být $\mathcal{NP}$-úplný. Naopak pro k na vstupu problému se jedná o $\mathcal{NP}$-úplný problém podle našeho Tvrzení 7.17. □

Úlohy k řešení

(7.6.1) *Rozhodněte, které z následujících problémů patří do třídy $\mathcal{P}$ všech polynomiálně řešitelných problémů.*
A: Problém rozhodnout, zda daný graf obsahuje nezávislou množinu (tj. podmnožinu vrcholů nespojených hranami) velikosti 7.
B: Problém rozhodnout, zda daný graf obsahuje nezávislou množinu (tj. podmnožinu vrcholů nespojených hranami) velikosti nejméně 2005.
C: Problém rozhodnout, zda daný graf má barevnost nejméně tři.
D: Problém rozhodnout, zda daný graf má barevnost nejvýše tři.
E: Problém rozhodnout, zda daný graf má barevnost přesně tři.
F: Problém rozhodnout, zda daný graf má barevnost přesně dva.

(7.6.2) *Analogicky k Příkladu 7.26 ukažte převod problému vrcholového pokrytí na nezávislou množinu. (Tj. opačný směr.)*

(7.6.3) *Párováním v grafu rozumíme podmnožinu hran, které nesdílejí žádný svůj koncový vrchol. Jak byste polynomiálně převedli problém nalezení párování velikosti p v grafu G na problém nezávislé množiny?*

*(7.6.4)** *Dokážete najít převod problému dominující množiny na vrcholové pokrytí? (Tj. opačný směr k Příkladu 7.27.)*

(7.6.5) *Patří do třídy $\mathcal{NP}$ problém zjistit, zda graf G obsahuje dvě Hamiltonovské kružnice, které nesdílí žádnou hranu?*

(7.6.6) *Patří do třídy $\mathcal{NP}$ problém zjistit, jaká je barevnost grafu?*

(7.6.7) *Patří do třídy $\mathcal{NP}$ problém zjistit, zda graf G obsahuje právě jedinou Hamiltonovskou kružnici? A co třeba negace tohoto problému?*

(7.6.8) *Je známo, že do třídy $\mathcal{NP}$ patří problém k-obarvení (zda graf lze obarvit korektně k barvami, tj. zda barevnost je $\leq k$) pro všechna k. Patří ale do třídy $\mathcal{NP}$ problém zjistit, zda graf G má barevnost právě k? Pro která k?*

(7.6.9) *Rozhodněte, které z následujících problémů patří do třídy $\mathcal{NP}$.*
A: Problém rozhodnout, zda daný graf má barevnost nejvýše čtyři.
B: Problém rozhodnout, zda daný graf má barevnost přesně čtyři.
C: Problém rozhodnout, zda daný graf má barevnost nejméně čtyři.
D: Problém rozhodnout, zda daný graf obsahuje nejméně tři Hamiltonovské kružnice.
E: Problém rozhodnout, zda daný graf obsahuje přesně tři Hamiltonovské kružnice.

(7.6.10) *Ukažte, že také následující problém tzv. "kubické kostry" je $\mathcal{NP}$-úplný: Vstupem je jednoduchý souvislý neorientovaný graf G. Otázkou je, zde lze v grafu G najít takovou kostru, jejíž všechny vrcholy jsou stupně nejvýše 3? (Stupně se samozřejmě myslí v té kostře, ne v G.)*
Jedná se vlastně o obdobu Hamiltonovské cesty, která je kostrou, jejíž všechny vrcholy jsou stupně nejvýše 2.

Návod: Použijte (třeba) převod z problému Hamiltonovské cesty.

(7.6.11) *Ukažte, proč je následující problém $\mathcal{NP}$-úplný: Vstupem je jednoduchý neorientovaný graf G. Ptáme se, zda lze v grafu G najít uzavřený tah procházející všemi vrcholy takový, že nejvýše jednou projdeme znovu vrcholem, kterým jsme již prošli dříve?*
Pro osvětlení – u Hamiltonovské kružnice jde o uzavřený tah, který žádný vrchol nezopakuje, kdežto v našem případě je dovoleno tahem zopakovat nejvýše jednou jeden vrchol.

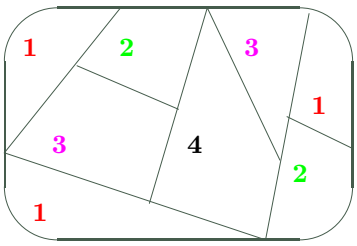
Návod: Použijte třeba převod z problému Hamiltonovské kružnice.

(7.6.12) *Sice už víme, že jak Hamiltonovská kružnice, tak i Hamiltonovská cesta jsou $\mathcal{NP}$-úplné, ale zkuste cvičně najít polynomiální převod Hamiltonovské cesty na Hamiltonovskou kružnici (naopak než v Příkladu 7.31).*

8 Rovinnost a kreslení grafů

Úvod

V přímé návaznosti na předchozí lekci se zaměříme na druhý důležitý aspekt slavného problému čtyř barev, který byl původně formulován pro barevné rozlišení států na politické mapě: (Praxe sice brzy ukázala, že 4 barvy vždy stačí, ale matematici si dlouho lámali hlavy s tím, proč tomu tak je…)



Jak tedy taková mapa souvisí s grafy? Jednoduše – souvislé státy můžeme reprezentovat jako vrcholy grafu a hranami pak zaznamenat „sousednost" mezi státy. Důležité je, že takto vzniklý graf můžeme zřejmě zakreslit v rovině bez křížení hran a nazýváme jej proto rovinným grafem. V našem výkladu si uvedeme se (a zčásti odvodíme) základní vlastnosti rovinných grafů a něco málo o kreslení grafů obecně.

Cíle

Prvním cílem je definovat rovinné grafy a přehledově ukázat jejich vlastnosti, především ve vztahu k jejich barevnosti a ke geometrii. Mimo jiné se naučíme rovinné grafy rozpoznávat podle Kuratowského věty. Za druhé si v závěru uvedeme poznatky o kreslení nerovinných grafů do roviny s co nejmenším počtem křížení (tzv. průsečíkové číslo).

8.1 Rovinné kreslení grafu

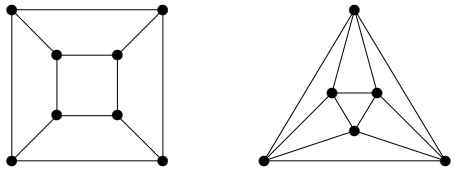
Dalším důležitým grafovým pojmem úzce svázaným s problémem čtyř barev je rovinné nakreslení, tj. nakreslení bez překřížených hran.

`Komentář:` Jen málokteré grafy rovinné nakreslení mají, ale důležitost grafů s rovinným nakreslením je motivována jak esteticky (nakreslení bez křížení hran vypadají hezky a jsou snadno "čitelná"), tak jejich teoretickým významem i praktickými aplikacemi (představme si jednostranný plošný spoj jako graf obvodu – hrany při jeho realizaci fyzicky nemůžeme křížit bez drátových propojek).

Definice 8.1. ***Rovinným nakreslení*** grafu G
myslíme zobrazení, ve kterém jsou vrcholy znázorněny jako různé body v rovině a hrany jako oblouky (čili jednoduché křivky) spojující body svých koncových vrcholů. Přitom hrany se nesmí nikde křížit ani procházet jinými vrcholy než svými koncovými body.

Graf je *rovinný* pokud má rovinné nakreslení.

`Komentář:` Důležitým příkladem rovinných grafů jsou grafy (třírozměrných Euklidovských) mnohostěnů, třeba graf čtyřstěnu, krychle, osmistěnu, dvanáctistěnu, atd.

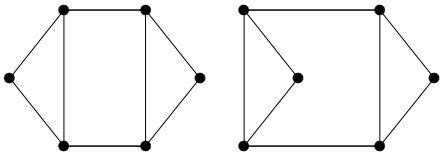


Platí, že grafy mnohostěnů jsou vždy rovinné a 3-souvislé. Naopak každý rovinný 3-souvislý jednoduchý graf je grafem nějakého mnohostěnu. (Důkaz tohoto tvrzení je obtížný.)

Geometrický příklad grafů mnohostěnů také přináší část terminologie rovinných kreslení a hlavně motivuje důležitý a slavný Eulerův vztah (Věta 8.2).

`Definice`: *Stěnami* rovinného nakreslení grafu nazýváme (topologicky) souvislé oblasti roviny ohraničené tímto nakreslením grafu.

`Komentář:`

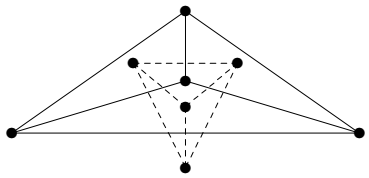


Rovinný graf může mít více *podstatně různých* nakreslení, jak vidíme na uvedeném ilustračním obrázku, ale platí, že 3-souvislý rovinný graf má ve všech svých rovinných nakresleních "stejné stěny" (Důsledek 8.11). To intuitivně znamená, že ze znalosti grafu mnohostěnu můžeme odvodit přibližný tvar původního tělesa.

Podívejte se zpět na konstrukci použitou v Příkladě 7.4 – oblasti, neboli stěny, mapy jsme nahrazovali vrcholy nového grafu a spojovali jsme hranami sousedící dvojice oblastí. Tuto konstrukci lze formálně zobecnit na stěny nakreslení libovolného rovinného grafu:

`Definice`. *Duální graf* rovinného nakreslení grafu G získáme tak, že stěny nahradíme vrcholy duálu a hranami spojíme sousedící dvojice stěn.

`Komentář:` Duální graf k rovinnému grafu je vždy rovinný, což je snadné dokázat. Pro příklad odvození duálního grafu se podívejme na obrázek:



Nyní si uvedeme zajímavý a vlastně "jediný rozumný" kvantitativní vztah o rovinných nakresleních grafů. Jedná se o slavný Eulerův vztah, který říká:

Věta 8.2. *Nechť rovinné nakreslení neprázdného souvislého grafu G má f stěn. Pak*

$$|V(G)| + f - |E(G)| = 2\,.$$

`Důkaz`: Nechť počet vrcholů v G je v a hran h. Důkaz této důležité věty pouze naznačíme, detaily lze najít v literatuře.

- Pokud je G strom, tj. nemá kružnice, má ve svém nakreslení jedinou stěnu a dle Věty 4.3 má přesně $h = v - 1$ hran. Potom platí $v + f - h = v + 1 - (v - 1) = 2$.
- Pokud G obsahuje kružnici C, pak vypustíme jednu její hranu e. Tím se počet hran sníží o 1, ale zároveň se sníží o 1 počet stěn, protože kružnice C původně oddělovala (viz Jordanova věta o kružnici) dvě stěny přilehlé k hraně e od sebe, ale nyní tyto dvě stěny „splynou" v jednu. Počet vrcholů se nezmění. Proto se nezmění hodnota $v + f - h = v + (f - 1) - (e - 1) = 2$.

Tvrzení tak plyne z principu matematické indukce. □

`Poznámka`: Všimněte si dobře, že Eulerův vztah vůbec nezávisí na tom, jak je graf G nakreslený, je to vlastnost grafu jako takového.

Tento jednoduše vypadající vztah má mnoho aplikací a důsledků, z nichž část si uvedeme i dokážeme.

Důsledek 8.3. *Jednoduchý rovinný graf na $v \geq 3$ vrcholech má nejvýše $3v - 6$ hran. Jednoduchý rovinný graf na $v \geq 3$ vrcholech a bez trojúhelníků má nejvýše $2v - 4$ hran.*

`Důkaz`: Můžeme předpokládat, že graf je souvislý, jinak bychom přidali další hrany. Nechť počet vrcholů v G je v, stěn je f a hran h. Jelikož nemáme smyčky ani násobné hrany, má každá stěna v nakreslení grafu na obvodu aspoň 3 hrany, přitom každou hranu započítáme ve dvou přilehlých stěnách. Pak tedy platí $e \geq \frac{1}{2} \cdot 3f$, neboli $\frac{2}{3}e \geq f$. Dosazením do vztahu Věty 8.2 získáme

$$2 = v + f - e \leq v + \frac{2}{3}e - e = v - \frac{1}{3}e$$

$$e \leq 3(v - 2) = 3v - 6\,.$$

Druhá část se dokazuje obdobně, ale nyní víme, že graf nemá ani trojúhelníky, a tudíž má každá stěna v nakreslení grafu na obvodu aspoň 4 hrany. Pak tedy platí $e \geq \frac{1}{2} \cdot 4f$, neboli $\frac{2}{4}e \geq f$. Dosazením do vztahu Věty 8.2 získáme

$$2 = v + f - e \leq v + \frac{2}{4}e - e = v - \frac{1}{2}e$$

$$e \leq 2(v - 2) = 2v - 4\,.$$

Tím jsme hotovi. □

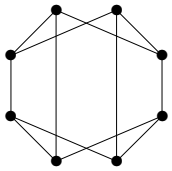
Důsledek 8.4. *Každý rovinný graf obsahuje vrchol stupně nejvýše* 5.
Každý rovinný graf bez trojúhelníků obsahuje vrchol stupně nejvýše 3.

Důkaz: Pokud by všechny vrcholy měly stupně alespoň 6, celý graf by měl aspoň $\frac{1}{2} \cdot 6v = 3v$ hran, což je ve sporu s Důsledkem 8.3. Některý vrchol musí tudíž mít menší stupeň než 6.

Obdobně postupujeme u druhého tvrzení. □

Úlohy k řešení

(8.1.1) *Nakreslete rovinně tento graf:*



(8.1.2) *Graf pravidelného dvacetistěnu má* 12 *vrcholů a* 20 *stěn. Kolik má hran?*

(8.1.3) *Co je duálním grafem k rovinnému nakreslení grafu krychle?*

(8.1.4) *K rovinnému nakreslení stromu přidáme dvě nekřížící se hrany. Kolik bude mít výsledný graf stěn?*

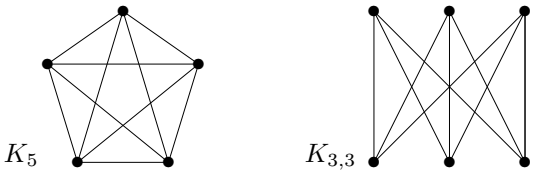
8.2 Rozpoznání rovinných grafů

Při praktickém využití rovinných grafů je potřeba umět abstraktně zadaný graf rovinně nakreslit bez křížení hran. Na rozdíl od problému určení barevnosti grafu se naštěstí jedná o efektivně algoritmicky řešitelný problém. První algoritmus běžící v lineárním čase byl podán Hopcroftem a Tarjanem 1974 a od té doby se objevilo několik jednodušších algoritmů, ale stále nejsou dostatečně přístupné, abychom je mohli předvést na přednášce.

Věta 8.5. *Rozhodnout rovinnost a nalézt příslušné nakreslení daného grafu lze v lineárním čase (vůči počtu vrcholů).*

Místo obecných algoritmů pro rovinné kreslení grafů se zde podíváme na otázku, jak odůvodnit nerovinnost (malého) grafu.

Příklad 8.6. *Ukažme, že následující dva grafy, K_5 a $K_{3,3}$ nejsou rovinné.*



Při zdůvodnění využijeme znalosti předchozího oddílu. Všimněme si, že graf K_5 má 5 vrcholů a $10 > 3 \cdot 5 - 6$ hran. Podobně graf $K_{3,3}$ má 6 vrcholů a $9 > 2 \cdot 6 - 4$ hran, přitom neobsahuje žádné trojúhelníky. Proto podle Důsledku 8.3 žádný z nich není rovinný. (Pokud by byl rovinný, počet jeho hran by musel být menší, než skutečně je.) □

Důsledek 8.7. *Grafy K_5 a $K_{3,3}$ nejsou rovinné.*

Význam grafů K_5 a $K_{3,3}$ uvedených v předchozím důsledku spočívá v tom, že jsou to "nejmenší" nerovinné grafy ve velmi silném významu slova nejmenší – každý další nerovinný graf jeden z nich "obsahuje". Pro uvedení takového popisu rovinných grafů ještě potřebujeme jednu definici.

`Definice`: *Podrozdělením* grafu G rozumíme graf, který vznikne z G rozdělením některých hran novými vrcholy stupně 2.

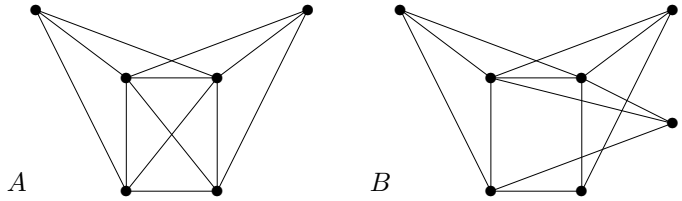
Důležitý abstraktní popis všech rovinných grafů nalezl K.Kuratowski:

Věta 8.8. *Graf G je rovinný právě když neobsahuje podrozdělení grafů K_5 nebo $K_{3,3}$ jako podgrafy.*

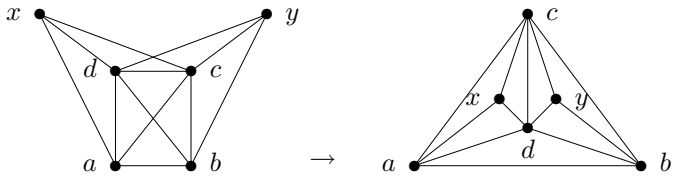
`Poznámka` o rozpoznání nerovinnosti grafu: Pokud chceme ukázat, že daný graf je rovinný, prostě jej nakreslíme v rovině bez křížení hran. Předchozí věta nám naopak dává spolehlivý způsob, jak ukázat, že daný graf není rovinný – prostě v něm najdeme podrozdělení grafů K_5 nebo $K_{3,3}$. (Ve skutečnosti tuto obtížnou větu ani nepotřebujeme ke zdůvodnění nerovinnosti, stačí nám Důsledek 8.7. Věta 8.8 nám jen říká, že příslušná podrozdělení vždy v nerovinných grafech najdeme.)

Pro praktické použití věty dodáme, že až na vzácné výjimky se lépe v nerovinných grafech najde podrozdělení grafu $K_{3,3}$ než grafu K_5.

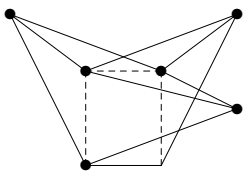
Příklad 8.9. *Které z následujících dvou grafů jsou rovinné? Najděte rovinné nakreslení (včetně očíslovaných vrcholů), nebo zdůvodněte nerovinnost grafu.*



Po chvíli zkoumání určitě přijdeme na to, že graf A se dá nakreslit rovinně takto:



Graf B na druhou stranu rovinný není podle Věty 8.8, protože je v něm obsaženo podrozdělení grafu $K_{3,3}$, které je ukázáno na tomto obrázku:



□

Jednoznačnost rovinného nakreslení

Již jsme neformálně zmiňovali, že rovinná nakreslení téhož grafu mohou být podstatně různá, ale pro 3-souvislé grafy je tomu jinak. Nyní si tento fakt zpřesníme.

`Fakt`: V 2-souvislém rovinném grafu je každá stěna ohraničená kružnicí.

Díky tomuto faktu lze snadno nadefinovat, že dvě rovinná nakreslení 2-souvislého grafu jsou *ekvivalentní*, pokud jejich stěny tvoří stejné soubory kružnic. Klíčovým výsledkem nyní je:

Lema 8.10. *Kružnice C v 3-souvislém rovinném grafu G je stěnou jeho nakreslení, právě když podgraf $G - V(C)$ je souvislý graf.*

Důsledek 8.11. *Každá dvě rovinná nakreslení 3-souvislého grafu jsou ekvivalentní.*

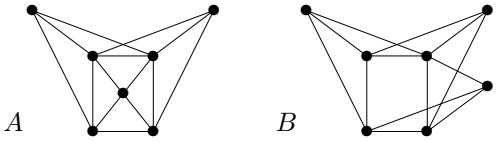
Závěrem si ještě bez důkazu uvedeme, že rovinné grafy vždy mají "pěkné" nakreslení v rovině.

Věta 8.12. *Každý jednoduchý rovinný graf lze nakreslit v rovině (bez křížení hran) tak, že hrany jsou úsečky.*

Úlohy k řešení

(8.2.1) *Pro která n je úplný graf K_n rovinný?*

(8.2.2) *Které z následujících dvou grafů jsou rovinné? Najděte rovinné nakreslení (včetně očíslovaných vrcholů), nebo zdůvodněte nerovinnost grafu.*



(8.2.3) *Kolik hran stačí přidat ke kružnici, aby vznikl nerovinný graf?*

8.3 Barvení map a rovinných grafů

`Komentář:` Vzpomeňme si na již zmiňovaný převod mapy na graf – jedná se vlastně o vytvoření duálního grafu k této mapě (strana 76). Aby v duálním grafu k mapě nevznikly smyčky, v mapě nesmí žádný stát sousedit sám se sebou, což je přirozený požadavek.

V roce 1976 Appel a Haken, a pak v roce 1993 znovu Robertson, Seymour, Sanders a Thomas, dokázali tuto větu, která rozřešila problém čtyř barev a která je jedním z nejslavnějších výsledků diskrétní matematiky vůbec:

Věta 8.13. *Každý rovinný graf bez smyček lze obarvit 4 barvami.*

Důkaz této věty je nesmírně složitý (však byl také hledán po více než 100 let a k jeho úplnému provedení je stále třeba výkonný počítač), a proto si uvedeme slabší a mnohem jednodušší tvrzení:

Tvrzení 8.14. *Každý rovinný graf bez smyček lze obarvit 6 barvami.*
Každý rovinný graf bez smyček a bez trojúhelníků lze obarvit 4 barvami.

`Důkaz`: Podle Důsledku 8.4 najdeme v každém podgrafu G vrchol v stupně nejvýše 5, a tudíž je G 5-degenerovaný a obarvíme jej podle Věty 7.6. Druhou část dokážeme obdobně, když nalezneme vrchol stupně ≤ 3. □

Už dávno je známo, že staré neúspěšné pokusy o důkaz problému čtyř barev ukazují alespoň, že barevnost rovinných grafů je nejvýše 5. Asi nejhezčí důkaz [Thomassen] tohoto faktu dokazuje indukcí mnohem více (a je optimální):

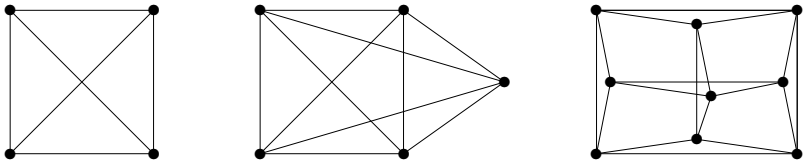
Věta 8.15. *Každý rovinný graf bez smyček má výběrovou barevnost nejvýše* 5.

8.4 O průsečíkovém čísle grafů

Jak víme, zdaleka ne všechny grafy jsou rovinné. Co tedy dělat s „kreslením“ nerovinných grafů do roviny?

`Definice` (rozšíření Definice 11.3):
Nakreslením grafu G do roviny rozumíme zobrazení, ve kterém jsou vrcholům G přiřazeny různé body roviny a hranám jednoduché křivky spojující koncové vrcholy. Přitom je požadováno, aby se žádné tři hrany neprotínaly v jednom bodě (jiném než koncový vrchol), aby žádná hrana neprocházela jiným vrcholem a aby se každá protínající dvojice hran skutečně „křížila“ (tj. ne jednostranný dotyk).

Příklad 8.16. *Podívejme se na následující tři (korektní) nakreslení do roviny. Jsou všechna „optimální“, tj. je počet jejich křížení nejmenší možný?*



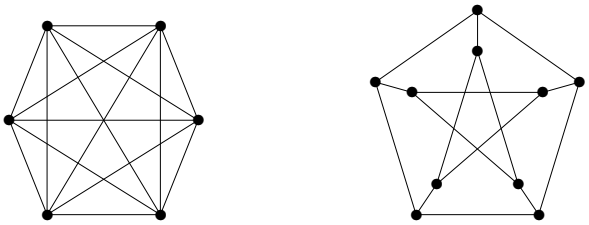
Snadno vidíme, že první graf lze nakreslit i bez křížení a druhý graf jen s jedním křížením. Naopak třetí graf už s méně kříženími nakreslit nelze. Uměli byste toto dokázat? □

Definice 8.17. ***Průsečíkové číslo*** grafu G v rovině
je definováno jako nejmenší možný počet křížení dvojic hran přes všechna korektní nakreslení G do roviny. Značíme $cr(G)$.

`Komentář`: Původ problému průsečíkového čísla spadá do doby druhé světové války, kdy P. Turán byl na nucených pracech v cihelně. Jejich úkolem bylo tlačit vozíky s cihlami po kolejnicích a na každé křižovatce s tím měli velké problémy. Proto Turán přemýšlel, jak navrhnout kolejiště lépe, aby minimalizoval počet křížení kolejnic.

V dnešní době je problém průsečíkového čísla velmi důležitý v praktických oblastech VLSI designu [Leighton] a „lidsky čitelné“ vizualizace grafů v různých schématech.

Příklad 8.18. *Určete průsečíková čísla následujících grafů:*



..... □

`Fakt`: Přesné obecné hodnoty průsečíkových čísel nejsou známy ani pro úplné či úplné bipartitní grafy.

Věta 8.19. *Problém určit, zda průsečíkové číslo $cr(G) \leq k$ pro G a k na vstupu je $\mathcal{NP}$-úplný. Toto platí dokonce i když G je kubický 3-souvislý graf.*

`Komentář:` Zamyslete se sami, proč by problém průsečíkového čísla měl vůbec náležet do třídy $\mathcal{NP}$, není to tak zřejmé...

V praxi se ukazuje, že určení průsečíkového čísla je přímo zoufale těžký problém, ještě mnohem beznadějnější než třeba barevnost. Snad jediným existujícím „pozitivním“ (i když zcela nepraktickým) algoritmickým výsledkem je následovné:

Věta 8.20. (Grohe) *Pro fixní k lze otestovat, zda $cr(G) \leq k$, v polynomiálním kvadratickém čase vzhledem k počtu vrcholů grafu.*
(2006: Dnes je známo i vylepšení v lineárním čase.)

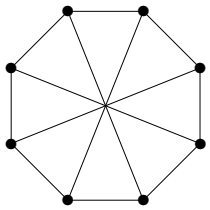
`Poznámka`: Dnes ani nevíme, jak v polynomiálním čase přesně určit průsečíkové číslo grafu, který vznikne z rovinného grafu přidáním jedné hrany! (Umíme jen aproximovat s faktorem daným největším stupněm grafu.)

Rozšiřující studium

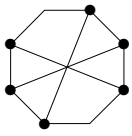
Problematika rovinného kreslení grafů a jejich barvení je pokryta v [5, Kapitola 5]. Zájemce o jednoduchý popis (poněkud pomalejšího) algoritmu na rovinné kreslení odkazujeme na [3]. Zájemci o podrobný popis historie problému čtyř barev zase mohou mnoho zajímavých informací nalézt na internetu, například `http://www.math.gatech.edu/~thomas/FC/fourcolor.html`. *O průsečíkovém čísle je také množství informací na internetu pod klíčovými slovy "crossing number".*

8.5 Cvičení: Příklady na rovinnost grafů

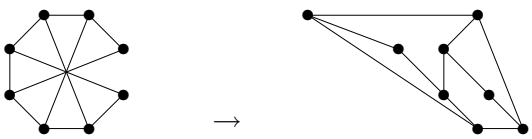
Příklad 8.21. *Kolik nejméně hran je třeba vypustit z následujícího 8-vrcholového grafu, aby vznikl rovinný graf? Zdůvodněte.*



Nejprve se podíváme, zda náš graf náhodou není rovinný. To ale není, protože zde vidíme v něm obsažené podrozdělení $K_{3,3}$:



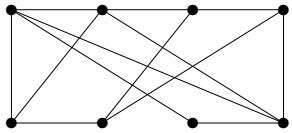
Tento obrázek zároveň ukazuje ještě jednu zajímavost – náš graf nebude rovinný, ani když vypustíme libovolnou z "tětiv" obvodové kružnice. Na druhou stranu není těžké objevit, že stačí vypustit třeba pravou svislou hranu a získáme rovinné nakreslení:



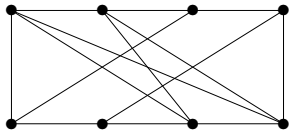
Stačí tedy vypustit jednu hranu a získáme rovinný graf. □

Úlohy k řešení

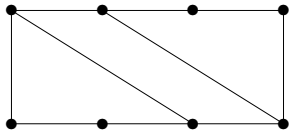
(8.5.1) *Je tento graf rovinný?*



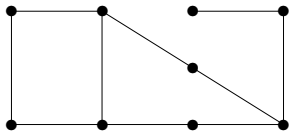
(8.5.2) *Je tento graf rovinný?*



(8.5.3) *Kolik nejvýše hran lze přidat do následujícího grafu na* 8 *vrcholech, aby ještě zůstal rovinný a jednoduchý?*



(8.5.4) *Kolik nejvýše hran lze přidat do následujícího grafu na* 9 *vrcholech, aby ještě zůstal rovinný a jednoduchý?*



(8.5.5) *Nakreslete libovolný rovinný graf s* 18 *hranami a* 10 *stěnami. Kolik má takový graf vrcholů?*

Část III
Vybrané Pokročilé i Aplikované Partie

9 Krátce o průnikových grafech

Úvod

Touto lekcí začínáme poslední část našeho studijního textu. Zatímco předchozí lekce pokrývaly obecné otázky teorie grafů, které by měly být v každém základním kurzu vysvětleny, nyní se zaměříme lehce na několik vybraných partií blízkých autorovu srdci.

Naším prvním výběrem jsou průnikové grafy, což jsou grafy, jejichž vrcholy jsou jisté množiny a hrany spojují pronikající se dvojice. Pochopitelně hlavní motivací studia těchto grafů je jejich geometrická názornost a aplikovatelnost v reálných situacích.

Cíle

Účelem této lekce je ukázat čtenáři pěkný svět tzv. průnikových grafů. Ty jsou definovány průniky jistých, povětšinou geometrických množin. Postupujeme od klasických intervalových a chordálních grafů, přes krátký neformální přehled jiných tříd až po křivkové a úsečkové reprezentace.

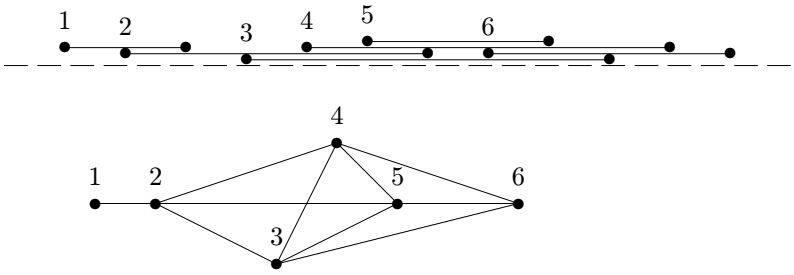
9.1 Intervalové a chordální grafy

`Definice`: Mějme množinový systém $\mathcal{M}$. Jeho *průnikovým grafem* nazveme graf $I_{\mathcal{M}}$ na vrcholech $V = \mathcal{M}$ s množinou hran $E = \{A, B \in \mathcal{M} : A \cap B \neq \emptyset\}$.

Intervalové grafy

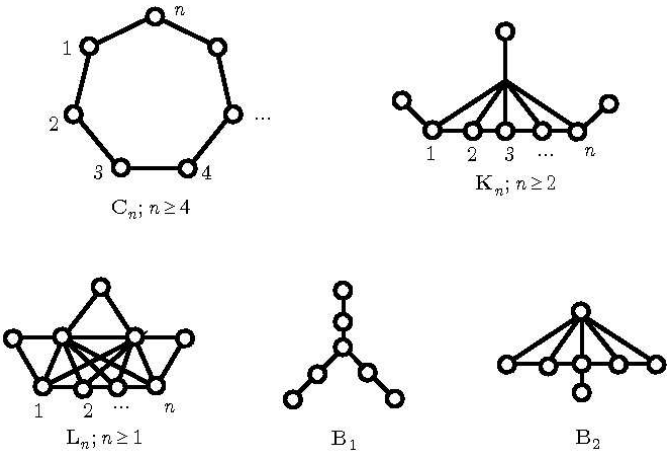
Jedná se o průnikové grafy intervalů na přímce (zkratka INT).

`Komentář:` Zde je jednoduchý příklad intervalového grafu:



`Fakt`: Každá kružnice délky větší než tři v intervalovém grafu má chordu.

- Intervalové grafy lze hezky popsat následujícími *zakázanými indukovanými podgrafy*:



- Graf je intervalový, právě když neobsahuje indukovanou C_4 a jeho doplněk má tranzitivní orientaci.
- *Jednotkové intervalové* grafy jsou ty mající intervalovou reprezentaci se všemi intervaly délky 1. Jsou to právě ty intervalové grafy bez indukovaného $K_{1,3}$.

Chordální grafy

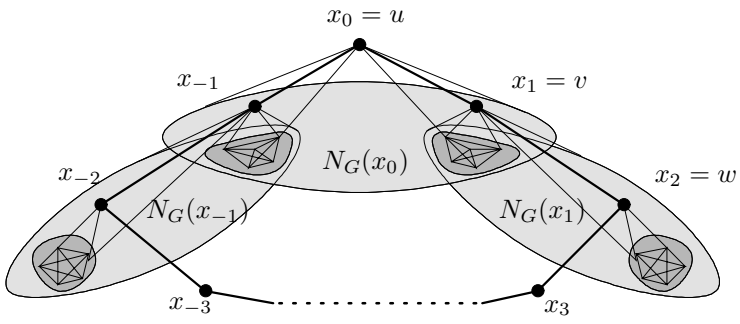
Tyto grafy představují užitečné zobecnění intervalových grafů.

`Definice`: Graf G je *chordální* pokud neobsahuje žádnou indukovanou kružnici delší než tři.

Věta 9.1. *Každý chordální graf G obsahuje simpliciální vrchol, tj. vrchol s takový, že všichni sousedé s tvoří kliku v G.*

`Důkaz`: Dokazujeme posloupnost tří snadných tvrzení o chordálním grafu G. Řekneme, že graf H je *bisimpliciální*, pokud H je úplný nebo H obsahuje dva nespojené simpliciální vrcholy.

1. Pro každou kružnici C a hranu e v G existuje hrana f taková, že $C \setminus \{e\} \cup \{f\}$ obsahuje trojúhelník.
2. Nechť uv je hranou G a sousedé v tvoří (indukují) bisimpliciální podgraf. Pokud v je simpliciální mezi sousedy u, ale ne v celém G, pak existuje vrchol w spojený s v a nespojený s u takový, že w je simpliciální mezi sousedy v.
3. Pokud G není bisimpliciální, ale sousedé každého jeho vrcholu indukují bisimpliciální podgraf, pak G obsahuje kružnici C odporující bodu 1.
4. Tudíž G je bisimpliciální. □



Simpliciální dekompozice... (postupně odebíráme simpliciální vrcholy)

`Fakt`: Simpliciální dekompozici lze využít k efektivnímu rozpoznávání intervalových i chordálních grafů.

Věta 9.2. *Graf G je chordální právě když G je průnikovým grafem podstromů v nějakém stromě.*

`Důkaz` indukcí podle počtu vrcholů – odebíráme simpliciální... □

9.2 Třídy průnikových grafů

Zde si uvedeme jen stručný neformální přehled některých typů průnikových grafů, které jsou studovány.

- *Hranový* graf je průnikovým grafem hran v běžném grafu.
- *Kruhově-intervalové* grafy (CA) jsou průnikovými grafy intervalů na kružnici.
- *Kružnicové* grafy (CIR) jsou průnikovými grafy tětiv v kružnici.
- *Diskové* grafy (DISC) jsou průnikovými grafy kruhů v rovině. Lze uvažovat také jen jednotkové kruhy (unit-DISC).
- *Kvádrové* grafy (BOX) jsou průnikovými grafy kvádrů ve dvou, třech či více dimenzích, se stěnami rovnoběžnými se souřadnicemi.
- *Dotykové*... grafy jsou variantou průnikových grafů geometrických objektů, ve které se požaduje, aby vnitřky objektů byly po dvou disjunktní.

Věta 9.3. *Graf je rovinný právě když je dotykovým grafem kruhů v rovině.*

- Jiné, tzv. *viditelnostní* grafy nejsou definovány průniky objektů, ale jejich vzájemnou viditelností v geometrickém světě.

9.3 Průnikové grafy křivek a úseček

Niťovými grafy krátce nazveme průnikové grafy křivek v rovině.

Tvrzení 9.4. *Existují grafy, které jsou niťové, ale každá jejich taková reprezentace obsahuje dvojici křivek majících exponenciálně mnoho (k počtu vrcholů) vzájemných průsečíků.*

Není tedy divu, že v následující větě bylo tou mnohem obtížnější částí dokázat příslušnost problému do třídy $\mathcal{NP}$ [Kratochvíl + Pelsmajer, Schaeffer, Štefankovič].

Věta 9.5. *Problém poznat, zda daný graf je niťový, je $\mathcal{NP}$-úplný.*

Velmi podobně je definována třída *segmentových grafů*, což jsou průnikové grafy úseček v rovině. Opět je dokázáno, že jejich rozpoznávání je $\mathcal{NP}$-těžké, ale příslušnost problému do třídy $\mathcal{NP}$ je tentokrát otevřená kvůli následujícímu:

Tvrzení 9.6. *Existují grafy, které jsou segmentové, ale každá jejich taková reprezentace obsahuje úsečku, k zápisu jejíž souřadnic je třeba exponenciálně mnoho (k počtu vrcholů) bitů.*

`Hypotéza 9.7.` Je každý rovinný graf segmentovým grafem?

"Zápalkové" grafy

Tímto pojmem nazýváme průnikové grafy úseček v rovině, přičemž dodatečnou podmínkou je, že žádné dvě úsečky se neprotínají ve svých vnitřních bodech. (Jedná se tedy o *dotykové grafy úseček*.)

Věta 9.8. *Graf je zápalkovým grafem s horizontálními a vertikálními „zápalkami", právě když se jedná o rovinný bipartitní graf.*

Zajímavá podotázka se týká toho, zda povolit „dvoustranné" dotyky zápalek nebo ne.

.....

Věta 9.9. *Problém poznat, zda daný graf je zápalkový, je $\mathcal{NP}$-úplný.*

Rozšiřující studium

.....

10 Dekompozice grafů, minory a algoritmy

Úvod

Další autorův výběr tématu nás zavede blíže ke strukturální teorii grafů, k různým jejich dekompozicím, grafovým minorům a navazujícím efektivním algoritmům pro jinak těžké problémy. Vrcholem této lekce je formulace hlavního výsledku takzvané „Graph Minors Theory" od Robertsona a Seymoura, který lze bez nadsázky prohlásit za asi největší výsledek, kterého dosud teorie grafů dosáhla (i v porovnání s Větou o čtyřech barvách).

Cíle

Cílem této lekce je uvést čtenáře do problematiky „šířkových" parametrů a dekompozic grafů a jejich aplikace na design efektivních algoritmů pro jinak velmi těžko řešitelné problémy. Dále je definován pojem minoru a formulována Robertsona–Seymourova věta o dobrém kvaziuspořádání konečných grafů vzhledem k minorům.

10.1 Tree-width – čtyři definice

Velikost největší kliky v grafu G označujeme $\omega(G)$.

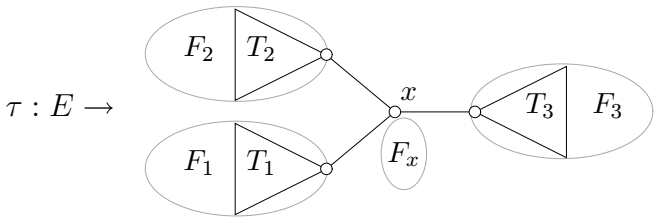
`Definice`: *Stromovou šířkou (tree-width)* grafu G nazveme nejmenší přirozené k takové, že exituje chordální graf H s $\omega(H) = k + 1$ obsahující G jako podgraf ($H \supseteq G$).

`Definice`: Vrcholy $V(G)$ grafu G uspořádáme do posloupnosti (permutace) $(v_1, v_2, \ldots, v_n)$. Pro $i = 1, 2, \ldots, n$ definujme $\ell(v_i)$ jako počet všech indexů $j \in \{1, \ldots, i-1\}$ takových, že vrcholy v_i a v_j jsou v G spojeny cestou používající pouze vrcholy z množiny $\{v_j, v_i, v_{i+1}, \ldots, v_n\}$. Druhou *stromovou šířkou* grafu G nazveme nejmenší hodnotu výrazu $\max_v \ell(v)$ přes všechny permutace vrcholů $V(G)$.

`Definice`: Pro libovolný strom T uvažujeme (libovolné) zobrazení $\tau : E(G) \to V(T)$. Pro vrchol $t \in V(T)$ označíme $T_1, \ldots, T_d$ jednotlivé komponenty lesa $T - t$ a $F_i = \tau^{-1}(V(T_i))$. Označme

$$\ell_\tau(t) = |V(G)| + (d-1) \cdot c(G) - \sum_{i=1}^{d} c(G - F_i),$$

kde $c(H)$ značí počet souvislých komponent grafu H.



Třetí *stromovou šířkou* grafu G pak definujeme jako nejmenší možnou, přes všechny dvojice T, τ, hodnotu výrazu $\max_{t \in V(T)} \ell_\tau(t)$.

Nakonec si uvedeme ještě jednu definici, která se většinou uvádí jako ta první a hlavní (a na ostatní možné definice málokdy přijde řeč).

Definice 10.1. ***Stromová dekompozice*** grafu G.
Stromovou dekompozicí grafu G nazveme strom T spolu se systémem množin $\mathcal{X}_t$ pro $t \in V(T)$, kde

- $\mathcal{X}_t \subseteq V(G)$ a $\bigcup_{t \in V(T)} \mathcal{X}_t = V(G)$,
- pro každou hranu $e = uv \in E(G)$ je $u, v \in \mathcal{X}_t$ pro nějaké $t \in V(T)$,
- (*interpolační* vlastnost) pro každý vrchol $v \in V(G)$ tvoří podmnožina všech $t \in V(T)$ s $v \in \mathcal{X}_t$ podstrom v T.

Šířkou dekompozice $T, \mathcal{X}$ rozumíme největší hodnotu $|\mathcal{X}_t| - 1$ pro $t \in V(T)$ a čtvrtou *stromovou šířkou* grafu G nazveme nejmenší možnou šířku stromové dekompozice G.

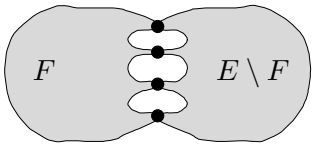
Věta 10.2. *Všechny čtyři výše uvedené definice stromové šířky definují přesně tutéž hodnotu, pokud graf G má neprázdnou množinu hran.*

10.2 Některé další parametry

`Definice`: *Cestní* dekompozici a šířku (path-width) grafu G definujeme stejně jako v Definici 10.1, jen požadujeme navíc, aby T byla cesta.

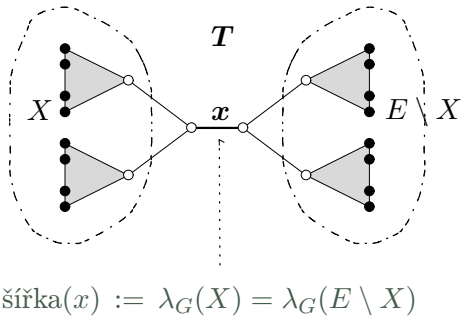
`Definice`: Vrcholy $V(G)$ grafu G uspořádáme do posloupnosti (permutace) $(v_1, v_2, \ldots, v_n)$. *Bandwidth* grafu G definujeme jako nejmenší hodnotu výrazu $\max_{v_i v_j \in E(G)} |i - j|$ přes všechny permutace vrcholů $V(G)$.

Graf je *kubický*, pokud má všechny vrcholy stupně 3. Strom je *podkubický*, pokud má všechny vrcholy stupně ≤ 3. Pro libovolný graf G a podmnožinu $F \subseteq E(G)$ definujeme *funkci souvislosti* $\lambda_G(F)$ jako počet vrcholů G, které jsou konci některých hran z F i hran z $E(G) \setminus F$.



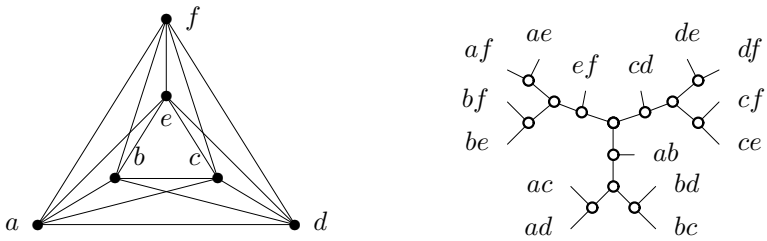
Definice 10.3. ***Větvená dekompozice*** grafu G
Nechť T je podkubický strom a $\tau : E(G) \to L(T)$ je bijekce hran grafu G do listů

$L(T)$ stromu T. Pro každou hranu x stromu T definujeme šířku x jako $\lambda_G(X)$, kde $X = \tau^{-1}(V(T_1))$ pro jednu z komponent T_1, T_2 lesa $T - x$.



Pak šířkou dekompozice T, τ je maximální šířka ze všech hran T a *větvenou šířkou* grafu G je nejmenší možná šířka větvené dekompozice G.

`Komentář:` Pro ilustraci si ukážeme větvenou dekompozici šířky 4 pro úplný graf K_6. (Například stromová šířka K_6 je rovna 6.)



Věta 10.4. (Robertson and Seymour) *Pokud graf G má stromovou šířku t a větvenou šířku $b > 1$, tak*

$$b \leq t + 1 \leq \left\lfloor \frac{3}{2} b \right\rfloor .$$

Nechť G je graf a $F \subseteq E(G)$. Jako funkci *hodnosti řezu* $\gamma_G(F)$ na množině F označíme hodnost $F \times (E \setminus F)$-matice $\boldsymbol{A} = (a_{i,j})$ nad binárním tělesem $GF(2)$, kde $a_{u,v} = 1$ pro $u \in F$ a $v \in E \setminus F$, právě když uv je hranou v G.

Definice 10.5. ***Ranková dekompozice*** grafu G
Nechť T je podkubický strom a $\tau : V(G) \to L(T)$ je bijekce vrcholů grafu G do listů $L(T)$ stromu T. Pro každou hranu x stromu T definujeme šířku x jako $\gamma_G(X)$, kde $X = \tau^{-1}(V(T_1))$ pro jednu z komponent T_1, T_2 lesa $T - x$. Pak šířkou dekompozice T, τ je maximální šířka ze všech hran T a *rankovou šířkou* grafu G je nejmenší možná šířka rankové dekompozice G.

`Fakt`: Rankovou šířku grafu lze omezit funkcí jeho větvené šířky, ale naopak toto neplatí – třeba úplné grafy mají rankovou šířku 1, kdežto jejich větvená i stromová šířka roste nade všechny meze.

10.3 Efektivní algoritmy na dekompozicích

Jak jistě víte, určení velikosti největší nezávislé množiny v grafu je $\mathcal{NP}$-úplný problém. Stromová dekompozice fixní šířky však tento problém umožňuje řešit velice snadno.

Algoritmus 10.6. ***Nezávislá množina na stromové dekompozici***

Problém hledání (maximálního) párování v grafu sice je polynomiálně řešitelný, ale zjišťování počtu všech párování už je #P-úplné, neboli stejně těžké (beznadějné) jako výpočet permanentu matice nebo spočítání všech řešení SAT problému.

Algoritmus 10.7. ***Počet párování na větvené dekompozici***

Poslední ukázka řeší problém 3-obarvení grafu s jeho rankovou dekompozicí fixní šířky a představuje netradiční nový přístup k takové problematice...

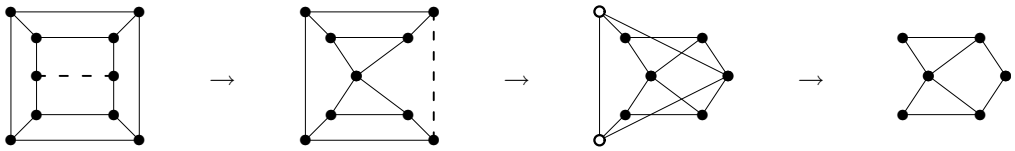
Algoritmus 10.8. ***3-obarvení na rankové dekompozici***

Pro zvídavé čtenáře na závěr zařazujeme následující těžkou otázku: Dokázali byste efektivně nalézt Hamiltonovskou kružnici v grafu na jeho rankové dekompozici fixní šířky?

10.4 Minory v grafech

`Definice`: Říkáme, že graf G je *minorem* grafu H, pokud lze G získat z H kontrakcemi hran a vypouštěním vrcholů a hran.

`Komentář:` Význam vypouštění vrcholů či hran je zřejmý. Pro pochopení operace kontrakce hrany („stažení do jednoho vrcholu“) je nejlepší se inspirovat následujícím obrázkem.



Robertson–Seymourova věta

Věta 10.9. *Mějme libovolnou grafovou vlastnost ϕ, která je uzavřená na minory (tj. pokud G má ϕ, pak každý minor G má také ϕ). Pak existuje konečně mnoho grafů $F_1, \ldots, F_\ell$ (zakázané minory) takových, že G má ϕ právě když G neobsahuje minor isomorfní žádnému z $F_1, \ldots, F_\ell$. Mimo jiné lze tudíž vlastnost ϕ rozhodnout v čase $O(n^3)$ pro každý n-vrcholový graf G.*

`Poznámka`: Tato věta je zcela nekonstruktivní, neboli nepodává žádný návod, jak zmíněný algoritmus sestrojit!

Malý námět k zamyšlení pro zvídavé: Dovedete si představit, co by znamenalo, pokud by se podařilo třeba dokázat $\mathcal{NP}$-úplnost nějakého problému uzavřeného na minory?

Rozšiřující studium

.....

11 Více o kreslení grafů

Úvod

Tato lekce se soustřeďuje na následující otázku: Jak vhodně nakreslíme nerovinný graf? Této otázce jsme se již implicitně věnovali u průsečíkového čísla grafu a nyní si ukážeme dva různé pohledy. Buď budeme grafy kreslit stále bez křížení, ale na tzv. “vyšší plochy” jako

torus nebo projektivní rovina. Nebo zůstaneme v rovině, ale povolíme křížení hran a budeme hledat "esteticky pěkná" nakreslení.

Cíle

Prvním cílem této lekce je definovat nakreslení grafu na vyšších plochách (kdy si stručně uvedeme i klasifikaci ploch). Druhou oblastí této lekce je vysvětlení praktického heuristického postupu na "estetické" kreslení libovolných grafů v rovině.

11.1 Kreslení grafů na plochy

Nejprve si stručně uvedeme důležitý výsledek klasické topologie – *klasifikaci ploch*.

Věta 11.1. *Každá plocha (tj. kompaktní 2-manifold) je homeomorfní jedné z*

- $\mathcal{S}$ *sféře.*
- $\mathcal{S}_h$ *sféře s* h *přidanýma "ušima" (handle).*
- $\mathcal{N}_k$ *sféře s* k *přidanými "křížícími místy" (crosscap).*

`Definice`: *Crosscap* na ploše je kružnice, jejíž protilehlé dvojice bodů jsou ztotožněny (kruhový vnitřek přitom ploše už nepatří).

`Poznámka`: Plocha $\mathcal{S}_1$ je známý *torus*, neboli povrch pneumatiky. Plocha $\mathcal{N}_1$ je *projektivní rovina* a vypuštěním kruhu z $\mathcal{N}_1$ vzniká známý *Möbiův proužek*. Plocha $\mathcal{N}_2$ je tzv. *Kleinova láhev*. Plochy $\mathcal{S}$ a $\mathcal{S}_h$ jsou orientovatelné, kdežto $\mathcal{N}_k$ jsou neorientovatelné.

Lema 11.2. *Máme-li plochu* Σ *vzniklou ze sféry přidáním* $k > 2$ *crosscapů a* h *uší, tak* Σ *je homeomorfní ploše vzniklé ze sféry přidáním* $k - 2$ *crosscapů a* $h + 1$ *uší.*

Definice 11.3. ***Nakreslení*** grafu G na plochu Σ
myslíme zobrazení, ve kterém jsou vrcholy znázorněny jako různé body na Σ a hrany jako oblouky (čili jednoduché křivky) spojující body svých koncových vrcholů. Přitom hrany se nesmí nikde křížit ani procházet jinými vrcholy než svými koncovými body.

Na vyšší plochy přenášíme i další pojmy rovinného kreslení, jako pojem stěny.

`Definice`: Nakreslení grafu G na plochu Σ je *buňkové*, pokud je každá stěna homeomorfní otevřenému disku.

`Fakt`: Buňkové nakreslení grafu G je jednoznačně určeno svými stěnovými kružnicemi a jako takové definuje i plochu Σ až na homeomorfismus. (Neboli plochu Σ lze "slepit" z jednotlivých disků stěn podél společných hran stěnových kružnic.)

Tvrzení 11.4. *Buňkové nakreslení grafu* G *na orientovatelnou plochu* Σ *je jednoznačně určeno rotačním schématem vycházejících hran u svých vrcholů. (V případě neorientovatelných ploch je třeba ještě přidat jistá "znaménka".)*

Kreslení na určenou plochu

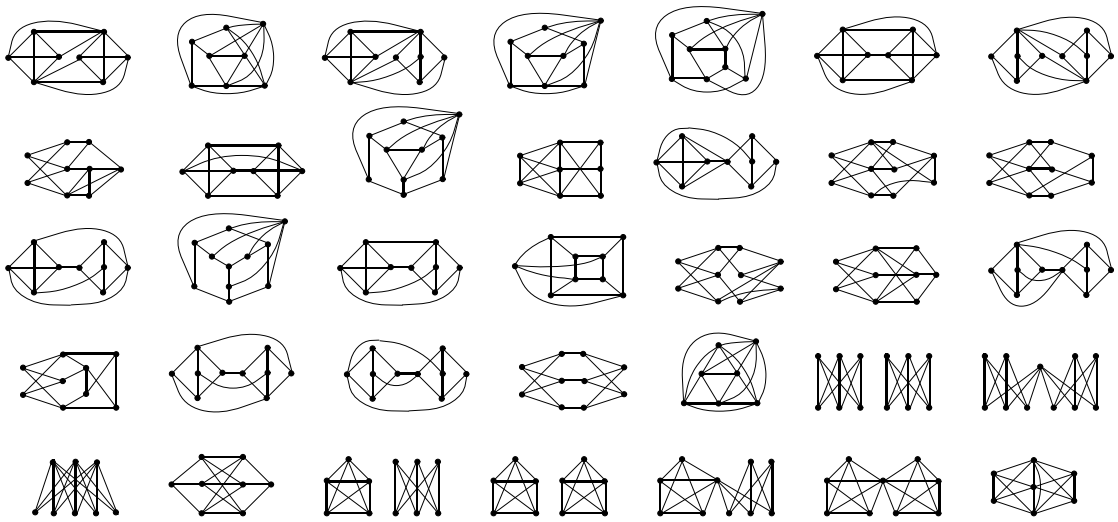
Mnohé poznatky o kreslitelnosti grafů lze zobecnit z rovinných na vyšší plochy.

Věta 11.5. (Mohar) *Pro každou pevnou plochu* Σ *existuje algoritmus, který v lineárním čase pro daný graf buď nalezne jeho nakreslení na* Σ*, nebo určí minimální překážku nakreslitelnosti na* Σ*.*

Za poznamenání stojí, že obecnější problém určit "nejjednodušší" plochu, na kterou lze daný graf nakreslit, je už $\mathcal{NP}$-těžký.

Z jiné strany lze zobecňovat Kuratowského větu na vyšší plochy. Třebaže je známo, že takové zobecnění s konečným počtem překážek je platné pro každou plochu, konkrétní seznam zakázaných minorů či podrozdělení je znám pouze u jediné vyšší plochy.

Věta 11.6. (Archdeacon) *Graf G je nakreslitelný do projektivní roviny právě když neobsahuje žádný z následujících* 35 *grafů isomorfní svému minoru.*



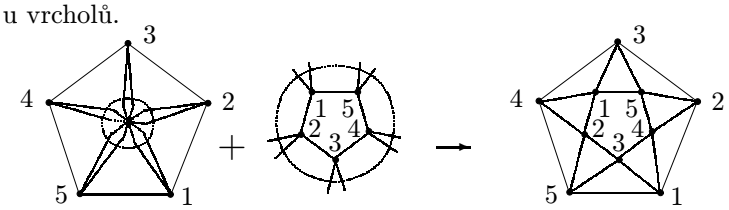
11.2 O problému rovinného pokrytí

Následující zajímavý partikulární problém je hezký především svou "chytlavostí"a jednoduchostí zadání. Je znám pod názvem *Negamiho hypotéza planárních pokrytí* nebo Negamiho $12\,\infty$ hypotéza.

`Definice`: Říkáme, že graf H *pokrývá* graf G, pokud existuje surjektivní zobrazení $\tau : V(H) \to V(G)$ takové, že sousedé každého vrcholu v grafu H jsou bijektivně zobrazeny na sousedy vrcholu $\tau(v)$ grafu G.

`Hypotéza 11.7.` Souvislý graf G má pokrytí (nějakým) konečným rovinným grafem, právě když G samotný je nakreslitelný do projektivní roviny.

> `Komentář:` Zde je příklad dvojitého pokrytí grafu K_5 rovinným grafem o 10 vrcholech. Pokrytí je získáno z projektivního nakreslení K_5 a zobrazení τ vrcholů z definice pokrytí je naznačeno čísly u vrcholů.



`Fakt`: Je-li G nakreslitelný do projektivní roviny, pak univerzální pokrytí projektivní roviny sférou okamžitě dá nakreslení dvojitého rovinného pokrytí grafu G.

Lema 11.8. *K důkazu Hypotézy 11.7 stačí ověřit, že žádný z* 32 *souvislých zakázaných minorů z Věty 11.6 nemá konečné rovinné pokrytí.*

Kupodivu pro většinu z oněch 32 grafů to lze dokázat velmi snadno, navíc lze využít následovný poznatek k další výrazné redukci počtu případů:

Lema 11.9. *Tzv. $Y\Delta$-operace (nahrazení vrcholy stupně 3 trojúhelníkem na jeho sousedech) zachovává konečné rovinné pokrytí.*

Přesto k této hypotéze stále není znám důkaz (a asi většina matematiků zabývajících se topologickou teorií grafů si s ní už zkoušela někdy lámat hlavu). Nejsilnější publikované poznatky o ní se dají shrnout následovně:

Věta 11.10. (Archdeacon, Fellows, Negami, PH) *Pokud graf $K_{1,2,2,2}$ nemá konečné rovinné pokrytí, tak je Hypotéza 11.7 pravdivá.*

Věta 11.11. (Thomas, PH) *Existuje jen 16 konkrétních grafů (až na triviální modifikace), pro které by Hypotéza 11.7 mohla být nepravdivá.*

11.3 Praktické "pružinové" kreslení grafů

Závěrem se podívejme na trochu jinou problematiku – jak prakticky nakreslit daný (nerovinný) graf, aby to "vypadalo hezky".

`Fakt`: Psychologické výzkumy ukázaly, že jedním z nejvýznamnějších kvantitativních parametrů nakreslení grafu pro jeho "lidskou čitelnost" je počet křížení hran.

Tento poznatek na jednu stranu ukazuje důležitost průsečíkového čísla grafu pro jeho vizualizaci, ale na druhou stranu nás nechává v poměrně beznadějné situaci, neboť určení průsečíkového čísla se ukazuje jako prakticky nemožné.

Jeden z nejstarších užitečných heuristických přístupů ke kreslení grafů se dá shrnout následovně:

Metoda 11.12. ***Pružinové kreslení grafu***

- Vytvoříme "fyzikální" model grafu, kde vrcholy budou kuličkami, které se vzájemně odpuzují, a hrany budou pružinami, které své koncové vrcholy vzájemně přitahují.
- Náš model budeme iterovat jako dynamický systém, až do konvergence pozic vrcholů. Zde je potřebné modelovat i "tlumení" pohybů vrcholů, aby nedošlo k rozkmitání systému.
- I když kreslíme graf do roviny, je užitečné začít modelovat systém s dimenzí navíc (aby měly vrcholy více "místa k pohybu") a teprve v průběhu času dodatečnou silou přidanou dimenzi "eliminovat", neboli zkonvergovat pozice vrcholů do zvolené roviny.

Rozšiřující studium

Při popisu kreslení grafů na plochy vycházíme ze skvělé monografie [7]. Asi v současnosti nejobsáhlejší popis Negamiho hypotézy a planárních pokrytích lze nalézt v autorově dizertaci (Georgia Tech, 1999) na webu `http://www.fi.muni.cz/~hlineny/Research/papers/gtthesis.pdf`.

Co se týče problematiky praktického kreslení grafů, je to velmi rozsáhlá oblast se spoustou materiálu na webu, hledejte pod klíčovými slovy "Graph drawing" a specificky "Spring embedder" pro pružinové kreslení.

Část IV

Klíč k řešení úloh

(1.1.1) Kružnice délky 5.

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

(1.1.3) Pro $n = 3$, trojúhelník.

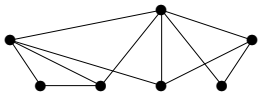
(1.1.4) Pro $m = n = 2$, délky čtyři.

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

(1.1.6) 9

(1.1.7) Oba stejně 36 hran.

(1.2.1) 34

(1.2.2) Ano: 

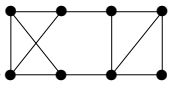
(1.2.3) Ano, kružnice délky 6 a dvě kružnice délky 3 vedle sebe.

(1.3.1) 3, více nelze, neboť na obvodě kružnice by se musely ob-jeden střídat.

(1.3.2) 3

(1.3.3) 7

(1.3.4)* 5, dokážete ji vůbec najít?

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

(1.3.6) Jeden, jen kružnice délky 5.

(1.6.1) Ano.

(1.6.2) Ne.

(1.6.3) Dva.

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

(1.6.5) Snadno...

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

(1.6.7) ...

(1.6.8)* 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.

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

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

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

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

(1.6.13) ...

(1.6.14) Vlevo 4, vpravo 3.

(2.1.1) B

(2.1.2) 4

(2.3.1) Vrcholově n-souvislý.

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

(2.3.3) 1 do kružnice.

(2.4.1) Ne, má čtyři vrcholy lichého stupně.

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

(2.5.1) Jednu.

(2.5.2) 5, samé trojúhelníky.

(2.5.3)* 6, samé úplné grafy K_5.

(2.5.4) Celkem 4.

(2.5.5) Souvislý s 15 hranami (ke každému vrcholu 3 sousedé).

(2.5.6) 18, stupně ≥ 3, třeba jako šestiboký hranol.

(2.5.7) a) 28: $K_8 + K_1 + K_1$, b) 9: $K_3 + K_3 + K_3 + K_1$

(2.5.8)* Jen v A: graf $K_{2,3}$ a graf C_5 s jednou diagonálou; C: cesta délky 4 a trojúhelník s hranou vedle. Pro B uvažte doplněk toho grafu – má stupně $2, 1, 1, 1, 1$, což dá jednoznačný graf.

(2.5.9)* Jen B: úplný K_4 s hranou vedle a dále K_4 s "rozpojenou" hranou. Pro A jde sestrojit jen kružnice C_5, protože na nesouvislý nemá dost hran.

(2.5.10)* 3

(2.5.11)* Třeba úplný bipartitní $K_{3,4}$.

(2.5.12) Ano, ale jen otevřeným.

(3.1.1) 1, ale jen 0 pro graf na jednom vrcholu.

(3.1.2) 2

(3.1.3) 5

(3.1.4) 3, najdete příslušnou dvojici vrcholů?

(3.2.1) Vyjde $\begin{pmatrix} 0 & 1 & 2 & 1 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 1 \\ 1 & 2 & 1 & 0 \end{pmatrix}$.

(3.3.1) 5

(3.3.2) Jsou to vrcholy ve zbylých dvou cípech hvězdy, do každého dalšího vrcholu je z nich vzdálenost nejvýše 4. Lépe to nejde.

(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)* 22, což je $\lfloor 45/2 \rfloor$, ohodnocení hran v pořadí $1, 2, 3, 4, 5, 7, 6, 8, 9$.

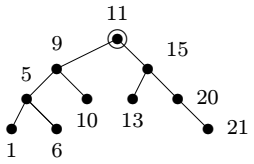
(3.5.8)* Vzdálenost 14 mezi 7 a 5.

(4.1.1) 3, jeden bez hran, jeden s jednou hranou a poslední strom s dvěma hranami.

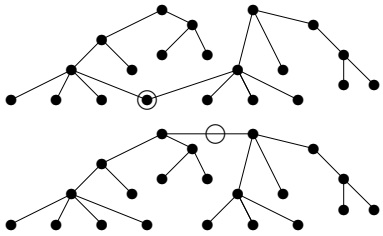
(4.1.2) 2, cesta P_3 a "hvězda" $K_{1,3}$.

(4.1.3)* Nenajdete, to by bylo v rozporu s Důsledkem 4.5.

(4.2.1)

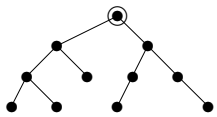


(4.2.2) Centra jsou vyznačená:



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

(4.3.1) ((()(()()())) (((())(())))

(4.3.2) 

(4.4.1) 2004

(4.4.2)* Je celkem $5!/2 = 60$ koster, co jsou cestou P_4, dále jen 5 koster, co jsou "hvězdou" $K_{1,4}$ a nakonec $5 \cdot 4 \cdot 3 = 60$ koster, které mají vrchol stupně 3.

(4.4.3)* ???

(5.1.1) Nalezneme kostru s největším celkovým ohodnocením.

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

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

(5.2.1) V jistém časovém okamžiku vidíme, že se zpracovávají 4 úkoly najednou, a proto méně než 4 pracovníci nestačí.

(5.2.2) ...

(5.4.1) Jakkoliv špatně, pro každý zvolený počet barev se dá nalézt špatný příklad, zkuste si to...

(5.4.2) V pořadí jakéhokoliv souvislého prohledávání našeho grafu.

(5.4.3) Jednoduše, prostě hladový algoritmus spustíme a on nikdy nepoužije barvu vyšší než $k + 1$, neboť každý vrchol má jen k (možná) obarvených sousedů.

(5.4.4)* Třeba pro graf, který je hvězdou s cestami délky 2 jako paprsky. Hladově se totiž nejprve vybere centrální vrchol velkého stupně, a pak také všichni jeho sousedé stupňů 2. Ten centrální vrchol tudíž bude přebývat, nebude optimální.

(5.5.1) 3

(5.5.2) Ano, 2005 vrcholů podle Věty 4.3.

(5.5.3) $2009 - 1 - 1 - 1 - 1 = 2005$

(5.5.4) 11, počítejte hrany v každé komponentě – stromu.

(5.5.5) Graf není souvislý, 13 není spojeno s ničím. Není ani lesem, neboť $10, 14, 21, 15$ tvoří kružnici.

(5.5.6) Vlevo B, vpravo A.

(5.5.7) 5

(5.5.8) 3 až 8 koster. Nejméně koster, 3, vznikne pokud nová hrana vytvoří trojúhelník. Nejvíce, 8, pokud strom byl cestou délky 7 a nová hrana ji uzavře do kružnice délky 8.

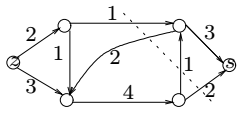
(5.5.9) $\binom{n}{2} - n + 1$

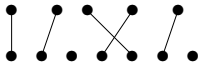
(5.5.10)* 4. Pro 3 vrcholy by každá kostra musela mít 2 hrany, ale $2 + 2 = 4$ hrany mezi třemi vrcholy nelze mít. Na druhou stranu úplný graf K_4 má takové dvě kostry.

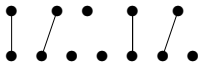
(5.5.11)* 56. Obvodová kružnice má 8 koster, dalších $3 \cdot 5 = 15$ koster obsahuje vrchní příčku a 15 spodní příčku, nakonec $3 \cdot 3 \cdot 2 = 18$ koster obsahuje obě příčky.

(6.2.1) Tok 5.

(6.2.2) Jen zdroj z.

(6.2.3) Řez velikosti 4 zde: 

(6.3.1) Velikosti 5: 

(6.3.2) Velikosti 4: 

(6.3.3) Ano, reprezentanti jsou zapsaní první: $(1,2,3)$, $(2,1,4)$, $(3,1,4)$, $(4,2,3)$.

(6.3.4) Ne, všech podmnožin je $\binom{5}{3} = 10$, ale prvků k reprezentaci jen 5.

(6.4.1) 12

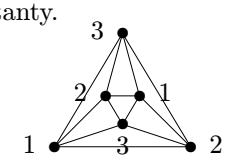
(6.4.2) 13

(6.4.3) 11

(6.4.4) 9

(6.4.5) Nemá – 6 z množin má sjednocení $\{3,4,5,6,7\}$, kde se najde jen 5 různých prvků pro reprezentanty.

(6.4.6) Nemá – 6 z množin má sjednocení $\{1,2,3,4,9\}$, kde se najde jen 5 různých prvků pro reprezentanty.

(7.1.1) 3 

(7.1.2) Méně než 4 barvy nestačí, protože najdeme úplný podgraf na 4 vrcholech. Obarvení 4 barvami je snadné.

(7.1.3) 1 pokud má jeden vrchol a jinak 2 podle Věty 7.5 – stromy totiž nemají žádné kružnice.

(7.1.4) 3

(7.1.5) 102 barvami, postup je následovný: Odebereme jeden vrchol v, zbytek grafu rekurzivně obarvíme 102 barvami a v nejhorším případě dostanou sousedé vrcholu v 101 různých barev. Proto i v můžeme korektně dobarvit.

(7.3.1) Je to kupodivu ještě jednodušší než v Tvrzení 7.19. Prostě v převodu z vrcholového pokrytí na grafu G vytvoříme orientovaný graf, jehož vrcholy jsou vrcholy G a také středy hran G. Šipky vedou vždy od vrcholů do středů přilehlých hran.

(7.5.1) 3

(7.5.2) 4, všimněte si, že graf obsahuje K_4.

(7.5.3) 3

(7.5.4) 1

(7.5.5)* 14, pokud vypustíme tři hrany z jednoho vrcholu, nebo 13, pokud vypustíme tři hrany jednoho trojúhelníku, nebo 12, pokud vypustíme tři disjunktní hrany.

(7.5.6)* ...

(7.6.1) A,B,C,F

(7.6.2) Je to stejný převod, neboť to funguje v obou směrech.

(7.6.3) Definujeme pro graf G nový graf H, jehož vrcholy budou odpovídat hranám grafu G, a hranami H budou spojeny dvojice hran z G sdílející vrchol. Potom prostě stačí v grafu H hledat nezávislou množinu velikosti p, což dá párování v G.

(7.6.4)* ...?

(7.6.5) Ano, patří, tyto kružnice napovíme a snadno ověříme.

(7.6.6) Ani nemůže patřit, neboť se nejedná o rozhodovací problém!

(7.6.7)* Nepatří, neboť nijak polynomiálně neověříme, že graf G neobsahuje více než jednu Hamiltonovskou kružnici. Ani v případě negace problému nejsme schopni ověřit případ, kdy G neobsahuje žádnou Hamiltonovskou kružnici.

(7.6.8)* Pro $k = 0, 1, 2$ lze barevnost efektivně určit, takže tam otázka patří přímo do třídy $\mathcal{P}$.

Také pro $k = 3$ patří problém barevnosti k do $\mathcal{NP}$, neboť napovědělé obarvení 3 barvami jsme schopni snadno ověřit, a zároveň dokážeme určit, že barevnost není 2 (kružnice liché délky). Ale pro $k > 3$ již problém (dle současných znalostí teoretické informatiky) do třídy $\mathcal{NP}$ nepatří, protože neumíme nijak efektivně prokázat, že graf G nelze obarvit méně než k barvami.

(7.6.9) A,D

(7.6.10) V převodu připojte ke každému vrcholu nový vrchol stupně 1.

(7.6.11)* Stačí G vytvořit tak, že k danému grafu připojíme jedním vrcholem w jeden nový trojúhelník. Právě vrchol w se pak v tahu bude muset zopakovat.

(7.6.12) Přidejte do grafu nový vrchol x spojený se vším – pak H. kružnice v novém musí procházet x a po odebrání x z ní zbude H. cesta.

(8.1.1) Je to graf krychle.

(8.1.2) $v = 12$, $f = 20$, $v + f - e = 2$, tedy $e = 12 + 20 - 2 = 30$.

(8.1.3) Graf pravidelného osmistěnu.

(8.1.4) 3

(8.2.1) Pro $n = 1, 2, 3, 4$.

(8.2.2) Rovinný jen B.

(8.2.3) 3 – takto z kružnice délky 6 vytvoříme graf $K_{3,3}$, nakreslete si to.

(8.5.1) Ano, stačí dolní levé dva vrcholy překreslit nahoru.

(8.5.2) Ne, obsahuje podrozdělení $K_{3,3}$, najdete jej?

(8.5.3) 8

(8.5.4) 11

(8.5.5) Má 10 vrcholů podle Eulerova vztahu.

Reference

[1] R. Diestel, Graph Theory (third edition), Springer, 2005.

[2] J.L. Gross, J. Yellen, eds. Handbook of Graph Theory, CRC Press, 2004.

[3] L. Kučera, Kombinatorické Algoritmy, SNTL, Praha, 1983.

[4] L. Kučera, Algovize,
`http://kam.mff.cuni.cz/~ludek/Algovision/Algovision.html`.

[5] J. Matoušek a J. Nešetřil, Kapitoly z Diskrétní Matematiky, Karolinum, UK Praha, 2000.

[6] J. Matoušek and J. Nešetřil, Invitation to Discrete Mathematics, Oxford University Press, 1998.

[7] B. Mohar, C. Thomassen, Graphs on Surfaces, Johns Hopkins, 2001.

[8] J. Oxley, Matroid Theory, Oxford University Press, 1992.

[9] J. Oxley, What is a Matroid?,
`http://www.math.lsu.edu/~preprint/2002/jgo2002e.pdf`, LSU, USA.