Základy Teorie Grafů

pro (nejen) informatiky

Doc. RNDr. Petr Hliněný, Ph.D.

hlineny@fi.muni.cz

12. září 2012

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

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

Verze 1.991 (2012)

Předmluva

Vážení čtenáři,

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

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

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

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

Ve stručnosti se zde zmíníme o struktuře naší učebnice. Přednášený materiál je dělený do jednotlivých lekcí (kapitol), které zhruba odpovídají obsahu týdenních přednášek v semestru. Lekce jsou dále děleny tematicky na oddíly. Výklad je strukturován obvyklým matematickým stylem na definice, tvrzení, případné důkazy, poznámky a neformální komentáře. Každá lekce má v závěru uvedeny bohaté odkazy na případné rozšiřující (samo)studium.

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

O výuce Teorie grafů MA010 na FI MU

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

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

`http://is.muni.cz/el/1433/podzim2012/MA010/index.qwarp`.

O historii našeho učebního textu

Původní výukový text vznikal v průběhu let 2003 až 2005 na základě autorových přednášek a cvičení předmětu Diskrétní matematika na FEI VŠB – TUO. Tento text byl pak autorem upraven a druhotně (především v teoretické oblasti) výrazně rozšířen pro potřeby výuky magisterského předmětu MA010 Teorie grafů na FI MU Brno od roku 2006. Za přípravu některých příkladů a zvláště za pečlivou kontrolu celého textu patří díky také mnohým zúčastněným cvičícím – za FEI VŠB – TUO Martinu Kotovi, Ondřeji Kohutovi, Michalu Kolovratovi a Pavlu Moravcovi a za FI MU Brno Janu Boudovi, Robertu Ganianovi, Janu Obdržálkovi a Pavlíně Vařekové.

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

A slovo závěrem...

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

Autor

Obsah

I Jak vzniká GRAF **1**

1 Pojem grafu **1**
1.1 Definice grafu . . . 1
1.2 Stupně vrcholů v grafu . . . 3
1.3 Podgrafy a Isomorfismus . . . 5
1.4 Orientované grafy a multigrafy . . . 8
1.5 Implementace grafů . . . 10
1.6 Cvičení: Základní grafové otázky . . . 10
1.7 Apendix: Další úlohy k isomorfismu . . . 16

II Základní Poznatky a Využití Grafů **20**

2 Souvislost grafů **20**
2.1 Spojení vrcholů, komponenty . . . 20
2.2 Prohledávání grafu . . . 22
2.3 Vyšší úrovně souvislosti . . . 24
2.4 Souvislost v orientovaných grafech . . . 26
2.5 Jedním tahem: Eulerovské grafy . . . 27
2.6 Cvičení: Procházení grafů a souvislost . . . 29

3 Vzdálenost a nejkratší cesty v grafech **34**
3.1 Vzdálenost v grafu . . . 34
3.2 Výpočet metriky . . . 36
3.3 Vážená (ohodnocená) vzdálenost . . . 37
3.4 Hledání nejkratší cesty . . . 38
3.5 Pokročilé plánování cest . . . 40
3.6 Cvičení: O cestách a grafových vzdálenostech . . . 42

4 Toky v sítích **47**
4.1 Definice sítě a toku . . . 47
4.2 Hledání maximálního toku . . . 49
4.3 Zobecnění definice sítí . . . 52
4.4 Speciální aplikace sítí . . . 54
4.5 Cvičení: Příklady toků v sítích . . . 57

5 Stromy – grafy bez kružnic **61**
5.1 Základní vlastnosti stromů . . . 61
5.2 Kořenové stromy . . . 63
5.3 Isomorfismus stromů . . . 66
5.4 Kostry grafů . . . 68
5.5 Cvičení: Příklady o stromech a kostrách . . . 70

III Další Užitečné Oblasti a Problémy **73**

6 Minimální kostry, Hladový algoritmus **73**
6.1 Hledání minimální kostry . . . 73
6.2 Hladový algoritmus v obecnosti . . . 75
6.3 Pojem matroidu . . . 76
6.4 Kdy hladový algoritmus (ne)pracuje správně . . . 79
6.5 Cvičení: Ukázky použití hladových algoritmů . . . 80

7 Barevnost a další těžké problémy **83**
7.1 Barevnost grafu . . . 83
7.2 Jak obarvit graf . . . 85
7.3 Variace na barevnost a jiné . . . 87
7.4 $\mathcal{NP}$-úplnost grafových problémů . . . 88
7.5 Příběh problému vrcholového pokrytí . . . 92
7.6 Cvičení: Příklady o barevnosti grafů . . . 93
7.7 Apendix: Další $\mathcal{NP}$-úplné grafové problémy . . . 95

8 Rovinnost a kreslení grafů **100**
8.1 Rovinné kreslení grafu . . . 100
8.2 Eulerův vztah o počtu stěn . . . 101
8.3 Rozpoznání rovinných grafů . . . 103
8.4 Barvení map a rovinných grafů . . . 105
8.5 Praktické „pružinové“ kreslení grafů . . . 106
8.6 Cvičení: Příklady na rovinnost grafů . . . 107

IV Vybrané Pokročilé Partie **109**

9 Povídání o průnikových grafech **109**
9.1 Průnikové a intervalové grafy . . . 109
9.2 Chordální grafy . . . 111
9.3 Další třídy průnikových grafů . . . 112
9.4 Průnikové grafy křivek a úseček . . . 114

10 Dekompozice grafů, algoritmy a minory **117**
10.1 Obtížné problémy na speciálních grafech . . . 117
10.2 Tree-width – čtyři definice . . . 118
10.3 Některé další parametry . . . 120
10.4 Efektivní algoritmy na dekompozicích . . . 122
10.5 Minory v grafech . . . 123

11 Pokročilé kreslení grafů **125**
11.1 Co jsou to plochy . . . 125
11.2 Kreslení grafů na plochy . . . 126
11.3 Překážky kreslení na plochy . . . 127
11.4 O průsečíkovém čísle grafů . . . 129
11.5 O problému rovinného pokrytí . . . 130

V **133**

Klíč k řešení úloh **133**
Literatura . . . 142

Část I
Jak vzniká GRAF

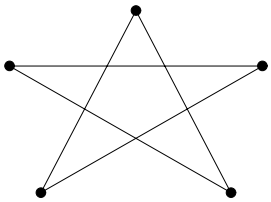
V úvodní části našeho učebního textu se nejprve seznámíme s grafy a naučíme se je pochopit a pracovat s nimi jako s matematickými objekty.

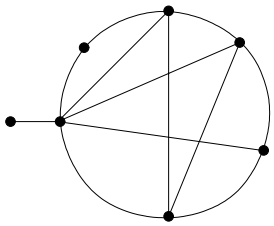
1 Pojem grafu

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, podgrafy). Také jsou zde shrnuty některé obvyklé typy grafů, jako cesty, kružnice, či úplné grafy. Především je však důležité, aby se čtenář naučil grafy "uchopit" a vykonávat s nimi základní operace a manipulace, což si dobře procvičí v následujících přiložených cvičeních. V tomto ohledu zaslouží specifickou zmínku pojem isomorfismu grafů, který je v úvodních partiích klíčový a je mu věnována značná pozornost.

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.

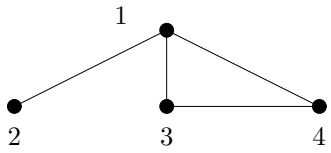
Definice 1.1. ***Graf*** (rozšířeně *obyčejný* či *jednoduchý neorientovaný* graf)
je uspořádaná dvojice $G = (V, E)$, kde V je množina *vrcholů* a E je množina *hran* – množina vybraných dvouprvkových podmnožin množiny vrcholů.

`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:

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

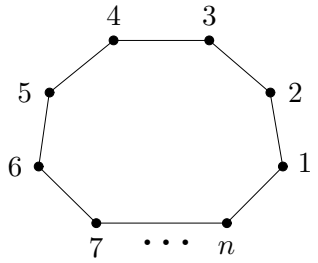


Běžné typy grafů

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

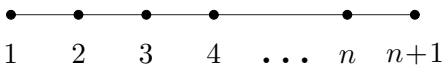
Kružnice délky n má $n \geq 3$ vrcholů spojených do jednoho cyklu n hranami:

C_n



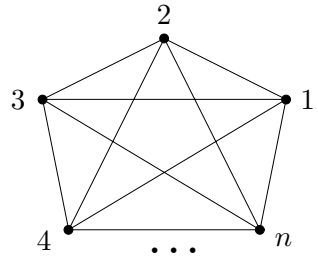
Cesta délky n má $n + 1$ vrcholů spojených za sebou n hranami:

P_n



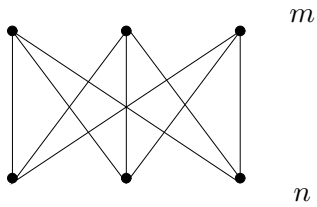
Úplný graf na $n \geq 1$ vrcholech má n vrcholů, všechny navzájem pospojované (tj. celkem $\binom{n}{2}$ hran):

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, neboli kolik má vrchol „sousedů“. Proto jedním z prvních definovaným pojmů bude stupeň vrcholu v grafu.

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

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

Například v nakreslené ukázce jsou stupně přímo zapsány u vrcholů.

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

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

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

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

Vlastnosti stupňů

Soubor stupňů grafu je jeho důležitou kvantitativní charakteristikou. Jelikož však v obyčejném grafu zvlášť nerozlišujeme mezi jednotlivými vrcholy, nemáme dáno žádné pořadí, ve kterém bychom stupně vrcholů měli psát (pokud je nepíšeme přímo do obrázku grafu). Proto si stupně obvykle seřazujeme podle velikosti.

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

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

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

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

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

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

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

Příklad 1.5. *Existuje graf se stupni vrcholů:*

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

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

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

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

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

□

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?*

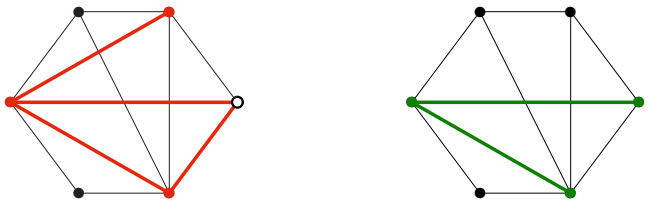
(1.2.4) *Proč má každý (jednoduchý) graf aspoň dva vrcholy stejného stupně?*

1.3 Podgrafy a Isomorfismus

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

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

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



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

„Různost“ grafů

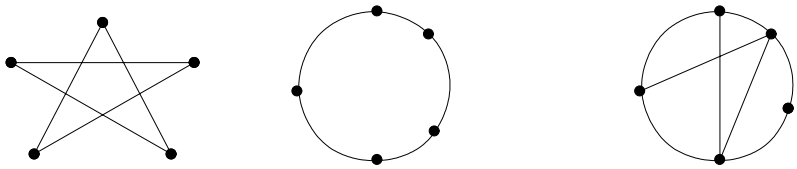
Pozorný čtenář si možná již při čtení předchozího textu položil otázku: Co když vezmeme jeden graf (třeba kružnici délky 5) a nakreslíme jej jednou tak, podruhé zase jinak – je to stále tentýž graf nebo ne? Přísně formálně řečeno, každé nakreslení jistého grafu, třeba té kružnice C_5, je jiným grafem, 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. □

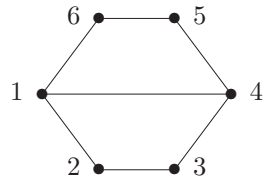
Důsledkem je, že všechny možné grafy se rozpadnou na *třídy isomorfismu*. V praxi pak, pokud mluvíme o *grafu*, myslíme tím obvykle jeho celou třídu isomorfismu, tj. nezáleží nám na konkrétní prezentaci grafu.

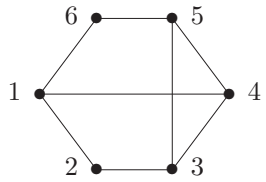
Další grafové pojmy

`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*. (Někdy se za kliku považuje pouze takový úplný podgraf, který je maximální vzhledem k inkluzi.)
- 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:` Uvažujme následující ukázky grafů:

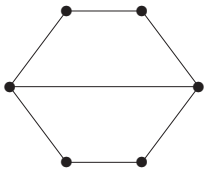


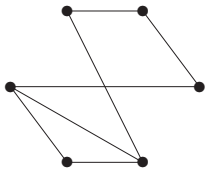


První z ukázaných grafů například neobsahuje žádný trojúhelník, ale obsahuje kružnici délky 4, dokonce indukovanou. Druhý graf trojúhelník obsahuje a kružnici délky 4 taktéž. První graf obsahuje cestu délky 4 na vrcholech $1, 2, 3, 4, 5$, ale ta není indukovaná. Indukovaná cesta délky 4 v něm je třeba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

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

Příklad 1.9. *Jsou následující dva grafy isomorfní?*





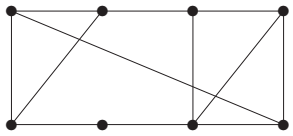
Postupovat budeme jako v Příkladě 1.7, nejprve ověříme, že oba grafy mají stejně mnoho vrcholů i stejnou posloupnost stupňů $2, 2, 2, 2, 3, 3$. Pokud se však budeme snažit najít mezi nimi isomorfismus, něco stále nebude vycházet, že? Co nám tedy v nalezení isomorfismu brání? Podívejme se, že v druhém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

`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 zatím platí, že jediná vždy fungující cesta pro nalezení či nenalezení isomorfismu mezi dvěma grafy je ve stylu *vyzkoušejte všechny možnosti* bijekcí mezi vrcholy těchto grafů. (Těch je, jak známo, až $n!$) V hierarchii výpočetní složitosti však problém isomorfismu leží docela nízko, jen „kousek nad“ třídou polynomiálních problémů, takže je jistá naděje na nalezení univerzálního efektivního algoritmu.

Úlohy k řešení

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

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



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

(1.3.4) *Jaká je délka nejdelší cesty obsažené v grafu z Úlohy 1.3.2?*

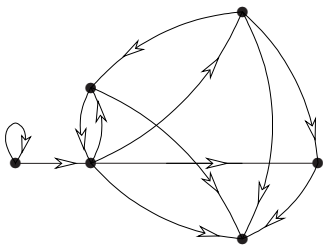
*(1.3.5)** *Jaká je délka nejdelší indukované cesty obsažené v grafu z Úlohy 1.3.2?*

(1.3.6) *Dokážete najít ke grafu z Úlohy 1.3.2 neisomorfní graf, který má stejnou posloupnost stupňů? Proč nejsou isomorfní?*

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

1.4 Orientované grafy a multigrafy

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



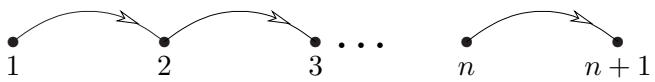
Definice 1.10. ***Orientovaný graf*** je uspořádaná dvojice $D = (V, E)$, kde $E \subseteq V \times V$. Pojmy podgrafu a isomorfismu se přirozeně přenášejí na orientované grafy.

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

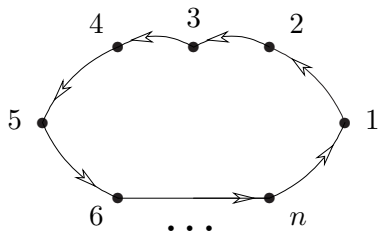
Speciálně hrana tvaru (u, u) se nazývá *orientovaná smyčka*.

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

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



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



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

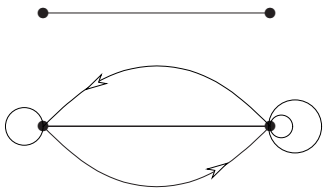
Různé modely grafu

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

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

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

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



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

Úlohy k řešení

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

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

1.5 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 matematický úvod do teorie grafů je zahrnut v [5, Kapitola 4] a volně na něj navazují i další části našeho učebního textu. Vřele doporučujeme [5] jako doplňkový studijní zdroj všem, kteří chtějí lépe pochopit grafy z jejich matematické stránky.

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

Studenti MU si také mohou vyzkoušet množství „online“ základních příkladů o grafech (na isomorfismus, stupně vrcholů, běžné vlastnosti grafů, atd.) ve formě odpovědníků v IS pod učebními materiály předmětu MA010 na FI MU od roku 2007. Někomu sice může připadat zbytečné trávit čas mechanickým řešením takových základních (a nudných?) příkladů, ale autor toto považuje za velmi přínosné zejména s ohledem na získání potřebného nadhledu nad grafy a „citu“ pro řešení složitějších grafových úloh v budoucnu.

1.6 Cvičení: Základní grafové otázky

Než přikročíme k jednoduchému procvičení úloh vztahujících se k fundamentálním pojmům úvodní lekce, nastíníme základní motivaci pro zavedení a použití grafů při popisu a řešení problémů v běžném životě.

Příklad 1.11. *Ukázky některých problémů ze života popsatelných grafy. Podotýkáme, že tyto ukázky jsou často velmi zjednodušené (pro jejich lepší přístupnost širokému okruhu čtenářů), ale to neubírá jejich motivačnímu potenciálu.*

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

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

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

- Vyjádření závislostí mezi objekty nebo procesy:

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

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

- Modelování technických či dopravních sítí grafy:

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

- Vizualizace vztahů a závislostí pro lidského pozorovatele:

  Nejen při řešení cvičných příkladů v naší učebnici, ale i v mnoha reálných aplikacích využívajících grafy jako modely, je velmi potřebné tyto grafy *vizualizovat* (tj. hezky nakreslit) pro lidského pozorovatele. Jedná se obecně o poměrně obtížný úkol, který přesahuje možnosti našeho textu, ale okrajově se o této obsáhlé problematice zmíníme také v pozdější Lekci 8.

□

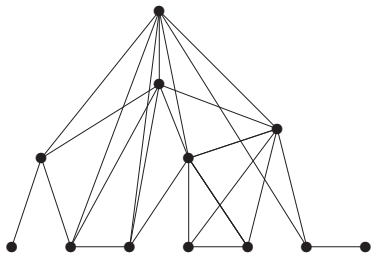
Nyní nastává čas k procvičení jednotlivých pojmů zavedených v lekci. Začneme od toho skutečně jednoduchého.

Příklad 1.12. *Existuje graf s posloupností stupňů* $1, 1, 3, 3, 3, 4, 4, 4, 6, 6, 6, 7$*?*

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

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

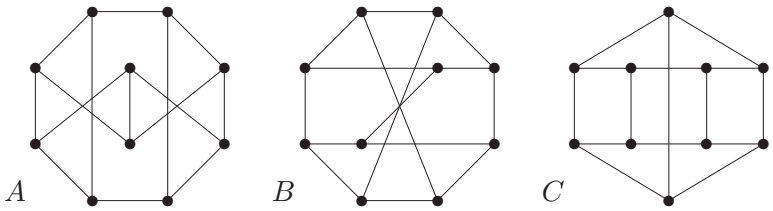
K poslední posloupnosti snadno nakreslíme graf, proto i graf s původní posloupností stupňů existuje. Zpětným přidáváním vrcholů pak můžeme sestrojit i jednu z možných podob původního grafu (který zajisté není jednoznačně určený!):



□

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

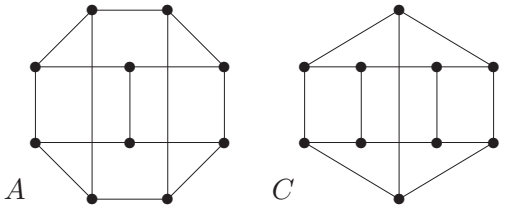
Příklad 1.13. *Najděte všechny isomorfní dvojice grafů v následujících obrázcích tří* 10*-vrcholových grafů. Isomorfní dvojice odpovídajícím způsobem očíslujte, u neisomorfních toto zdůvodněte.*



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.)

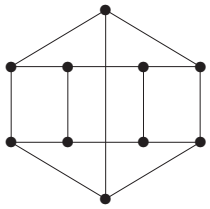
Takže co z dosavadního zkoumání plyne? Graf B nemůže být isomorfní žádnému z A, C. Nyní tedy zbývá najít (očekávaný) isomorfismus mezi grafy A a C. To se nám skutečně podaří poměrně snadno - stačí „prohozením“ prostředních dvou vrcholů u grafu

A získat lepší obrázek



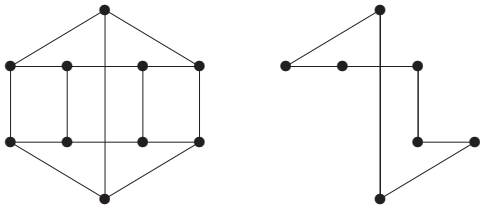
a odpovídající bijekce je na pohled zřejmá. (Doplňte si ji společným očíslováním vrcholů obou grafů.) □

Příklad 1.14. *Jaká je nejdelší kružnice a nejdelší indukovaná kružnice obsažená v následujícím grafu? Zdůvodněte.*



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

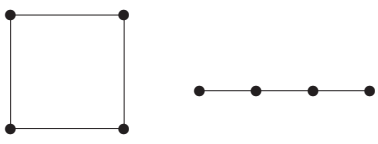
Co se týče indukované kružnice, ta musí s vybranými vrcholy podgrafu obsahovat i všechny hrany mezi nimi (a přesto to musí stále být jen kružnice). To zřejmě žádnou kružnicí délky 10 splněno není. Nenajdeme dokonce ani takové kružnice délky 9 a 8, nejdelší při troše snahy najdeme indukovanou kružnici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



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

Příklad 1.15. *Kolik vzájemně neisomorfních jednoduchých grafů na 8 vrcholech existuje s posloupností stupňů* $1, 1, 2, 2, 2, 2, 2, 2$*?*

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



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

Úlohy k řešení

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

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

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

(1.6.4) *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 $5 - 3 = 2$.

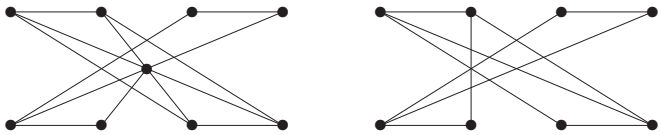
(1.6.5) *Kolik hran má graf s touto posloupností stupňů?*
A: $1, 1, 2, 2, 3, 3, 4, 4, 4, 4, 5, 6, 7$
B: $1, 1, 2, 2, 2, 2, 4, 4, 4, 5, 6, 7$
C: $1, 1, 2, 2, 2, 3, 3, 4, 4, 4, 5, 6, 7$

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

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

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

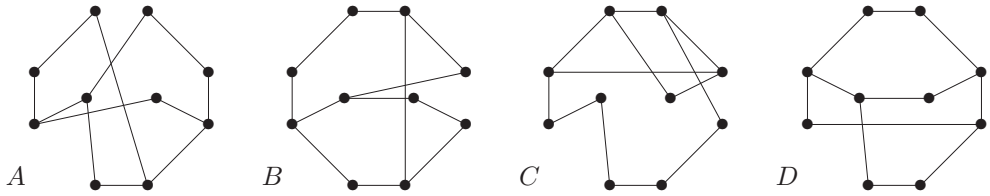
(1.6.9) *Kolik nejvíce vrcholů může obsahovat nezávislá množina v nakreslených grafech? Tuto největší nezávislou množinu si v grafech vyznačte.*



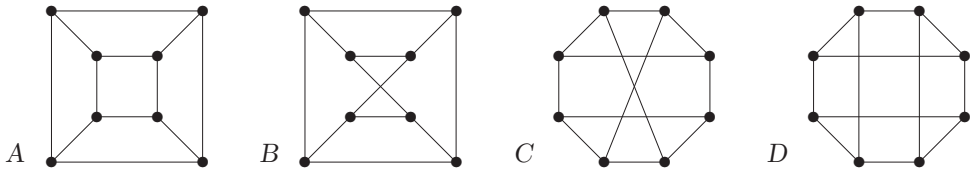
Návod: Překreslete si tyto (záměrně nepříliš přehledné) obrázky na „hezčí“.

(1.6.10) *Jakou největší kliku obsahují grafy z Úlohy 1.6.9?*

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



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



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

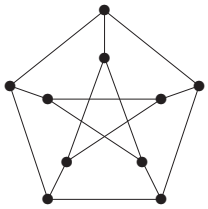
(1.6.13) *Jaká je nejdelší kružnice a nejdelší indukovaná kružnice obsažená v grafech A a B z Úlohy 1.6.12?*

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

(1.6.15) *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.*

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

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



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

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

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

*(1.6.20) *Kolik navzájem neisomorfních jednoduchých grafů lze získat z grafu A v úloze 1.6.11 přidáním jedné hrany?*

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

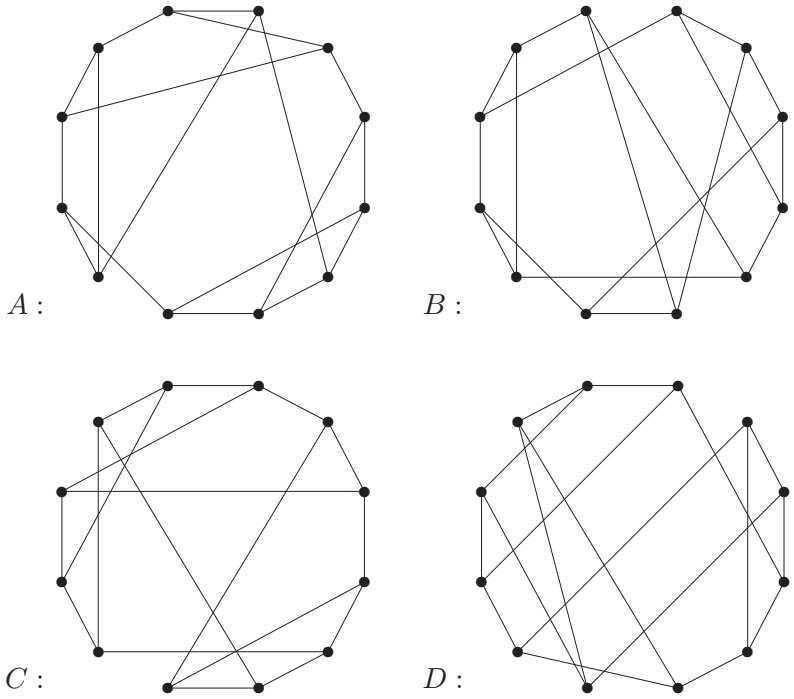
(1.6.22) *Proč v orientaci z úlohy 1.6.21 musí z každého vrcholu vycházet právě 3 šipky?*

1.7 Apendix: Další úlohy k isomorfismu

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

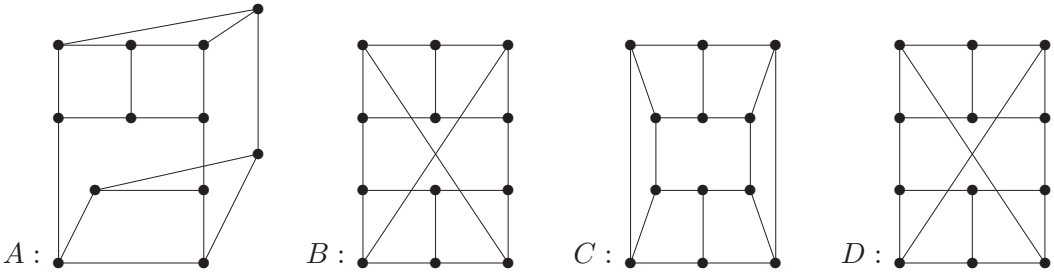
Všímejte si v dalších příkladech, jak významnou roli při řešení hraje nalezení vhodného nakreslení zkoumaného grafu!

Příklad 1.16. *Dány jsou následující čtyři jednoduché grafy na* 12 *vrcholech každý.*



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

Zadané grafy si překreslíme například takto:



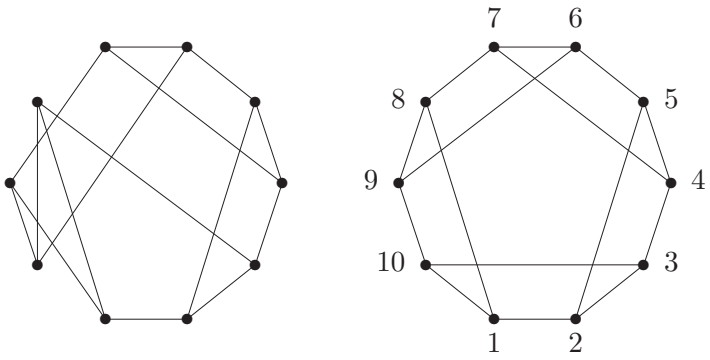
Jak jsme přišli zrovna na tato nakreslení? To nelze jednoduše vysvětlit, prostě si musíme několikrát každý graf zkoušet kreslit a představovat, až najdeme ten pravý pohled. (Ověřte si sami očíslováním, že nové obrázky jsou isomorfní zadaným... Chce to jen trochu cviku s kreslením grafů.)

Ihned tak vidíme, že $B \simeq D$. Pozor, nelze však nyní jen tak říci, že obrázky A a C vypadají jinak a tudíž nejsou s B isomorfní! Musíme najít jednoznačné zdůvodnění

rozdílu mezi grafy. Grafy A i C například oba obsahují 6 kružnic délky 4 (vyznačte je v obrázku), ale v A existují vrcholy náležející zároveň do tří kružnic délky 4 (opět vyznačte), kdežto v C každým vrcholem procházejí právě dvě kružnice délky 4. Proto $A \not\simeq C$. Dále zdůvodníme v obrázku, že $B \simeq D$ obsahují celkem jen 4 kružnice délky 4, a tudíž $A \not\simeq B$ a $C \not\simeq B$ a stejně s D.

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

Příklad 1.17. *Kolik navzájem neisomorfních jednoduchých grafů lze získat z následujícího grafu vlevo přidáním jedné hrany?*

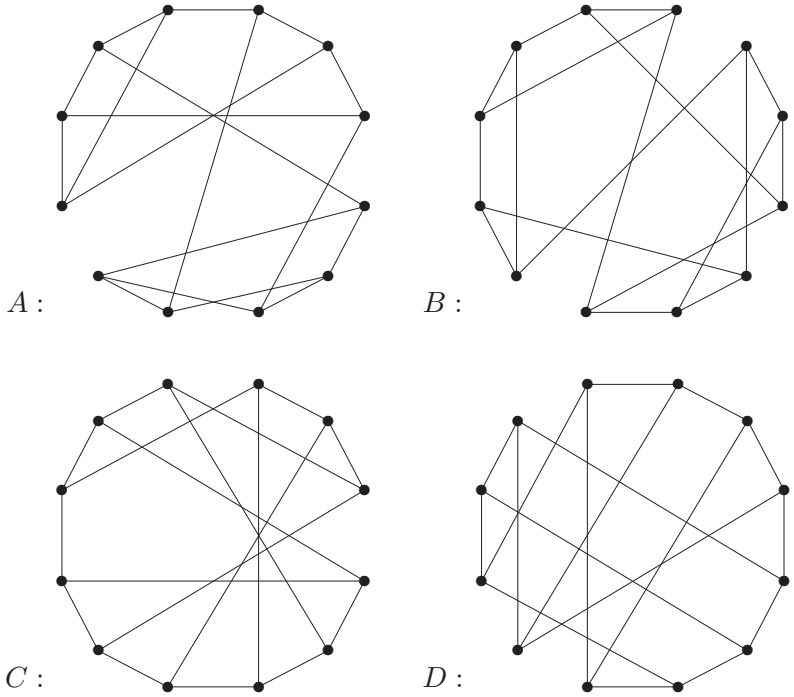


Prvním krokem k řešení příkladu jako tento je nalézt „co nejlepší“ obrázek našeho grafu. V tomto případě nalezneme isomorfní obrázek vpravo. (Jak, to lze jen těžko algoritmicky popsat, je k tomu potřeba zkušenost s grafy a představivost. . .)

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

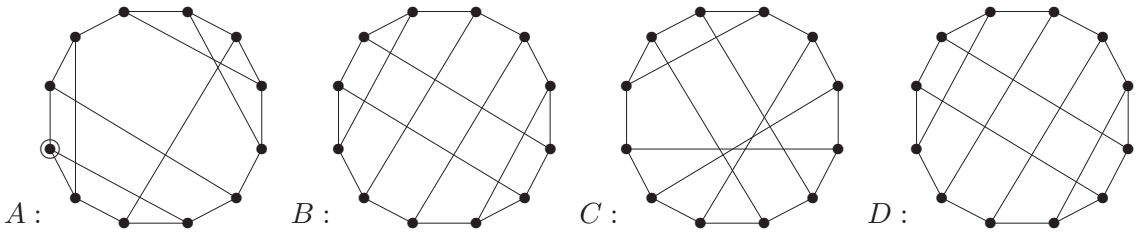
Takže odpověď je 3 vzájemně neisomorfní grafy. □

Příklad 1.18. *Dány jsou následující čtyři jednoduché grafy na* 12 *vrcholech každý.*



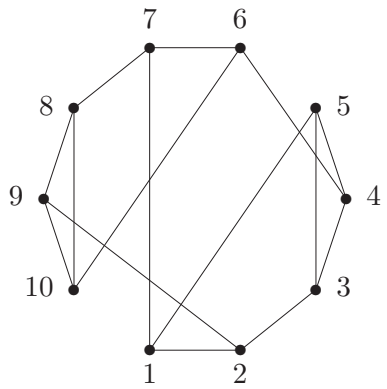
Vašim úkolem je mezi nimi najít všechny isomorfní dvojice a zdůvodnit svou odpověď.

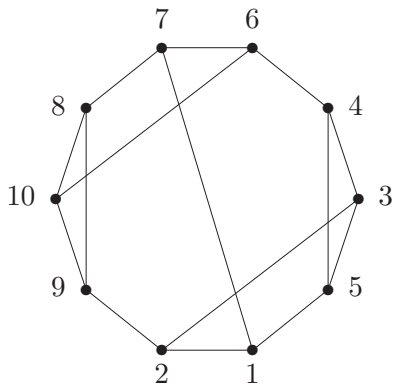
Zadané grafy si opět (po pár pokusech) pěkně názorně překreslíme, například takto (procvičte si sami vyznačení isomorfismu mezi obrázky nahoře a dole):



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

Příklad 1.19. *Vašim úkolem je zjistit, kolik vzájemně neisomorfních grafů může vzniknout z následujícího grafu vlevo přidáním jednoho nového vrcholu x a (jediné) hrany z x do některého původního vrcholu. Svou odpověď aspoň stručně zdůvodněte.*





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

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

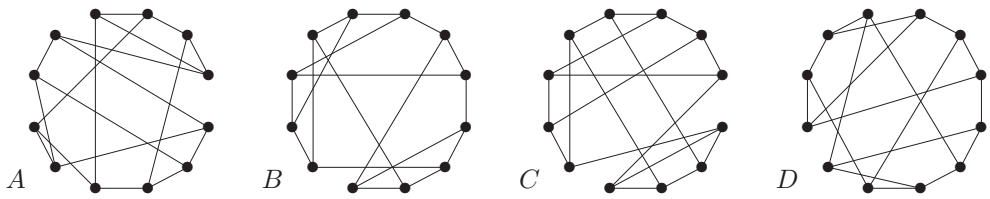
Takže odpověď je 5 vzájemně neisomorfních grafů. □

Úlohy k řešení

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

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

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



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

Část II
Základní Poznatky a Využití Grafů

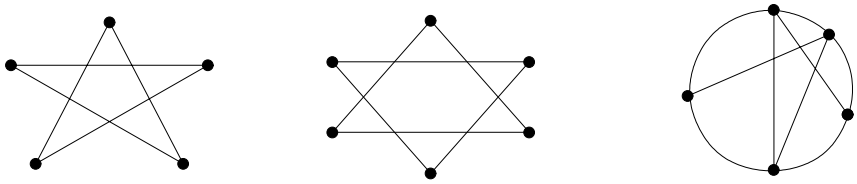
Znajíce již docela dobře, co to vlastně je ten „graf“, podíváme se nyní ve čtyřech lekcích na několik okruhů grafových pojmů a úloh, se kterými se asi nejčastěji setkáváme v aplikacích vázaných k informatice.

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. Při vyjadřování o grafech ale učitě nepoužívejme slovo „spojité“!) 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).

Potřebnou dovedností je umět celý souvislý graf algoritmicky procházet. Zde si uvedeme obecnou kostru algoritmu pro procházení grafu a zmíníme některé konkrétní variace tohoto algoritmu. (Znáte nějaký algoritmus pro procházení bludiště? V grafech je to v obecnosti podobné...) Také se zmíníme o vyšších stupních souvislosti obyčejného grafu, jako třeba o zálohování spojení v sítích pro případy výpadku, a o jiném pojetí souvislosti v orientovaných grafech. Závěr je pak věnován užitečné hříčce o „sedmi mostech“, čili kreslení jedním tahem.

Cíle

Tato lekce definuje a vysvětluje pojmy souvislosti a komponent grafu. Zmíněny jsou i vyšší stupně souvislosti a silná souvislost orientovaných grafů. V nepodlední řadě je ukázáno, jak se souvislostí grafu pracovat a jak také algoritmicky procházet graf po jeho hranách.

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).

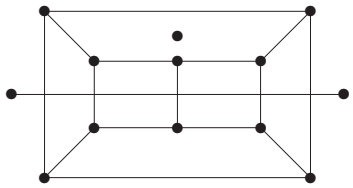
Lema 2.1. *Mějme relaci $\sim$ na množině vrcholů $V(G)$ libovolného grafu G takovou, že pro dva vrcholy $u \sim v$ právě když existuje v G sled začínající v u a končící ve v. Pak $\sim$ je relací ekvivalence.*

`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.

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



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

Připomeňme si, že *cesta v grafu* je vlastně sledem bez opakování vrcholů. Přirozeně pak vyplývá následující poznatek.

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. Jeho výhodou je konstruktivnost – vidíme, jak cestu získat.

`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. □

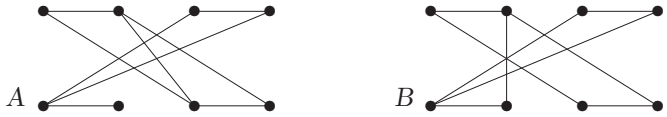
Závěrem se dostáváme k nejdůležitější definici souvislého grafu:

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

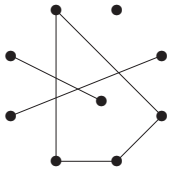
`Poznámka`: Prázdný graf je souvislý a má 0 komponent.

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

Způsob, kterým se vybírají vrcholy z úschovny ke zpracování, určuje variantu algoritmu procházení grafu. Zde pro větší obecnost dokonce v úschovně udržujeme vrcholy spolu s příchozími hranami. V prohledávaných vrcholech a hranách se pak provádějí konkrétní programové akce pro prohledání a zpracování našeho grafu.

Algoritmus 2.4. ***Generické procházení souvislé komponenty G grafu***
Algoritmus projde a zpracuje každou hranu a vrchol souvislého grafu G. Konkrétní uživatelské programové akce potřebné ke zpracování grafu jsou provedeny v pomocných funkcích `ZPRACUJ()`.

```
vstup < graf G;
stav(všechny vrcholy a hrany G) < iniciační;
úschovna U = {(∅, v0)}, pro libovolný vrchol v0 grafu G;
strom prohledávání T = ∅;
while (U je neprázdná) {
    zvolit (e,v) ∈ U;
    U = U \ {(e,v)};
    if (e ≠ ∅) ZPRACUJ(e);
    if (stav(v) ≠ zpracovaný) {
        foreach (f hrana vycházející z v) {
            w = opačný vrchol hrany f = vw;
            if (stav(w) ≠ zpracovaný)                    (*)
                U = U ∪ {(f,w)};
        }
        ZPRACUJ(v);
        stav(v) = zpracovaný;
        T = T ∪ {e,v};
    }
}
// případný přechod na další komponentu G'...
G je zpracovaný;
```

K průchodu nesouvislého grafu dochází po jednotlivých komponentách, po projití jedné komponenty algoritmem se libovolně přejde na vrchol další komponenty. V každé souvislé komponentě je automaticky vytvořen tzv. strom prohledávání T (DFS či BFS, apod) mající význam pro některé další aplikace prohledávání.
Podmínka $(*)$ je dodatečná a zajišťuje zpracování každé hrany jen v jednom směru (což je přirozené pro neorientované grafy). Pokud $(*)$ vypustíme, bude každá hrana zpracována v obou směrech.

`Komentář:` Konkrétní postup algoritmu procházení bude ilustrován ve Cvičení 2.5.

Algoritmus 2.5. ***Některé implementace procházení grafu;***
tj. konkrétně implementace kroku „`zvolit (e,v) ∈ U`“ úschovny `U`.

- *Procházení „do šířky“, BFS* – úschovna `U` je implementovaná jako fronta, neboli je voleno $(e, v) \in U$ od prvních vložených do úschovny.
- *Procházení „do hloubky“, DFS* – úschovna `U` je implementovaná jako zásobník, neboli je voleno $(e, v) \in U$ od posledních vložených do úschovny.
- *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, v Důsledku 3.4 a v Algoritmu 3.12.

  `Komentář:` Algoritmus 2.4 podává velmi obecné schéma a čtenář si povšimne, že v jednotlivých praktických případech BFS, DFS a jiných lze výsledný algoritmus značně zjednodušit: typicky tím, že úschovna bude udržovat pouze vrcholy v místo dvojic (e, v), což třeba zabrání opakování téhož vrcholu v U a uspoří pracovní paměť.

2.3 Vyšší úrovně souvislosti

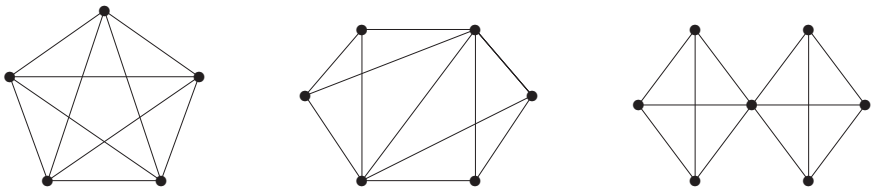
V *síťových aplikacích* nás často zajímá nejen, jestli se za normálních podmínek můžeme pohybovat mezi vrcholy/uzly, ale také, jaké spojení můžeme nalézt v případě lokálních výpadků (odolnost a redundance). Toto lze teoreticky podchytit zkoumáním „vyšších“ stupňů souvislosti grafu.

`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. 1-souvislý graf je pouhé synonymum pro souvislý.

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



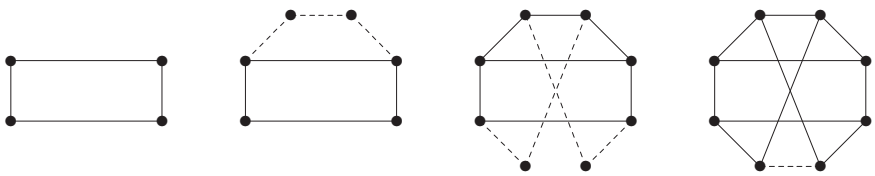
Na ilustračním obrázku má první graf vrcholovou souvislost 4 a snadno vidíme, že po odebrání tří vrcholů či hran zůstává souvislý. Z druhého grafu bychom museli odebrat nejméně 3 hrany, aby se stal nesouvislým, a proto je jeho hranová souvislost 3. Na druhou stranu však stačí odebrat 2 vrcholy, aby mezi jeho levým a pravým krajním vrcholem žádné spojení nezůstalo. (Vidíte, které dva?) A jak je tomu u třetího grafu?

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

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

Věta 2.6. *Libovolný obyčejný graf je 2-souvislý, právě když jej lze vytvořit z kružnice „přidáváním uší“; tj. iterací operace, kdy libovolné dva stávající vrcholy grafu jsou spojeny novou cestou libovolné délky (ale ne paralelní hranou).*

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



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

V druhém směru zvolíme maximální 2-souvislý podgraf G' v G, který lze sestrojit z nějaké kružnice přidáváním uší. G' je neprázdný, neboť 2-souvislý G obsahuje aspoň kružnici. Pokud $V(G') = V(G)$, pak lze jako uši přidávat všechny zbylé hrany G, tudíž z maximality plyne $G' = G$. Takže existuje vrchol $x \in V(G) \setminus V(G')$. Pak lze dokázat, že z x vedou dvě vnitřně disjunktní cesty do různých dvou vrcholů G', které tvoří další ucho přidané k G'. (Existence těchto dvou cest plyne třeba z Věty 2.7, ale elementární krátký argument zatím v této lekci nemáme.) Opět máme spor s maximalitou G', tudíž opět $G' = G$. □

Mengerova věta

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

Věta 2.7. *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.

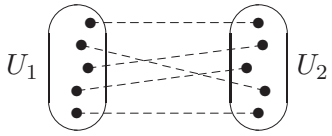
V duchu předchozí Mengerovy věty pokračujeme s následujícími poznatky.

Věta 2.8. *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 2.7 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:

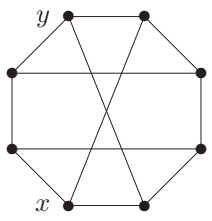
Věta 2.9. *Nechť G je vrcholově k-souvislý graf, $k \geq 1$. Pak pro každé dvě disjunktní množiny $U_1, U_2 \subset V(G)$, $|U_1| = |U_2| = k$ v G existuje k po dvou disjunktních cest z vrcholů U_1 do vrcholů U_2.*



Úlohy k řešení

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

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



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

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

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

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

2.4 Souvislost v orientovaných grafech

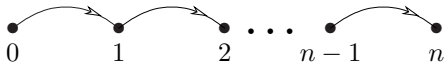
Třebaže orientované grafy jsou jen okrajovou součástí v našem výkladu látky, pojem orientované (silné) souvislosti je natolik fundamentálně odlišný (což je dané jeho „směrovostí“), že si zaslouží podrobný samostatný oddíl. Začneme analogicky Oddílu 2.1.

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

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

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

Věta 2.10. *Pokud mezi dvěma vrcholy grafu D existuje orientovaný sled, pak mezi nimi existuje orientovaná cesta.*

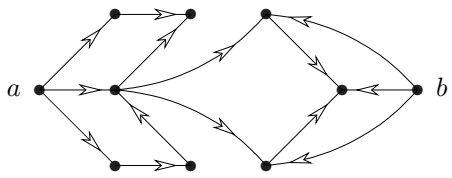


Pohledy na orientovanou souvislost

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

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

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

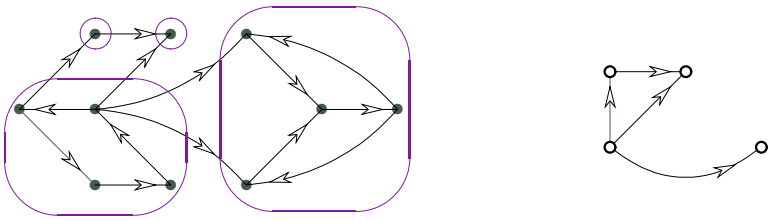


Nakonec „symetrizací“ přístupu dosažitelnosti se dobereme definici tzv. *silné souvislosti*, která je nejčastěji zmiňována u orientovaných grafů.

Lema 2.11. *Nechť $\approx$ je binární relace na vrcholové množině $V(D)$ orientovaného grafu G taková, že $u \approx v$ právě když existuje dvojice orientovaných sledů – jeden z u do v a druhý z v do u v grafu D. Pak $\approx$ je relace ekvivalence.*

Definice 2.12. ***Silné komponenty*** orientovaného grafu D
jsou třídy ekvivalance relace $\approx$ z Lematu 2.11. Orientovaný graf D je *silně souvislý* pokud má nejvýše jednu silnou komponentu.

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



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

Kondenzace orientovaného grafu

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

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

Tvrzení 2.13. *Kondenzace každého orientovaného grafu je acyklický orientovaný graf.*

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

Úlohy k řešení

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

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

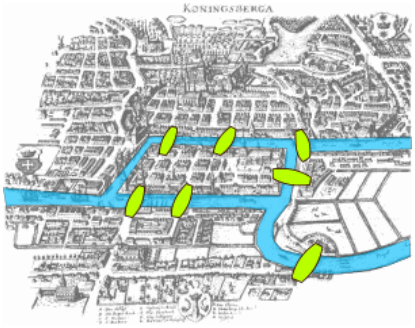
(2.4.3) *Kdy je v orientovaném grafu sám vrchol silnou komponentou?*

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

2.5 Jedním tahem: Eulerovské grafy

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

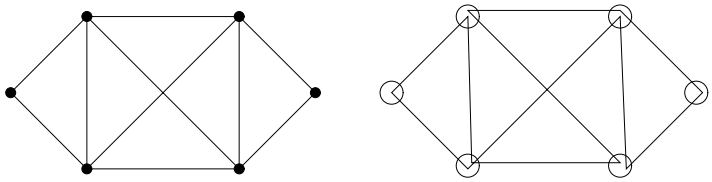
`Komentář:`



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

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

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



Onen slavný výsledek teorie grafů od Leonarda Eulera poté zní:

Věta 2.14. *Graf G lze nakreslit jedním uzavřeným tahem právě když G je souvislý a všechny vrcholy v G jsou sudého stupně.*

Důsledek 2.15. *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.

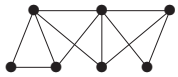
- Pro spor vezměme graf $G' = G - E(T)$, o kterém předpokládejme, že je neprázdný. Jelikož G' má taktéž všechny stupně sudé, je (z indukčního předpokladu) libovolná jeho hranově-neprázdná komponenta $C \subseteq G'$ nakreslená jedním uzavřeným tahem T_C.
- Vzhledem k souvislosti grafu G každá komponenta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

□

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

Úlohy k řešení

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



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

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

Rozšiřující studium

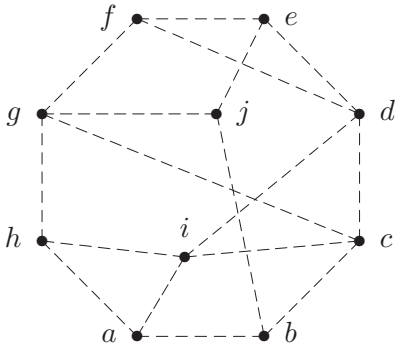
Základní souvislosti grafů se blíže teoreticky věnují [5, Oddíly 4.2, 4.7] a Eulerovským grafům obsáhleji [5, Oddíly 4.5, 4.6]. Algoritmy pro procházení grafu jsou podrobně popsány (včetně netriviálních aplikací) v [3] a demonstrovány třeba v `http://kam.mff.cuni.cz/~ludek/Algovision/Algovision.html`.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

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

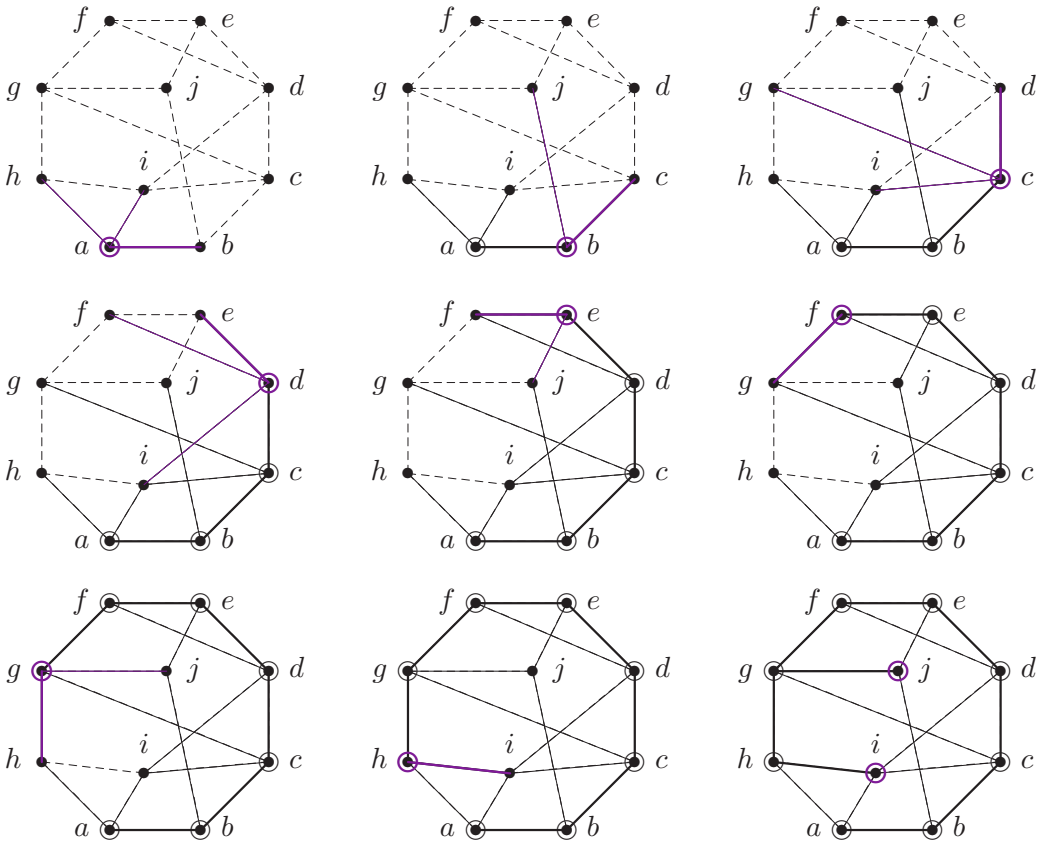
2.6 Cvičení: Procházení grafů a souvislost

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

Příklad 2.16. *Ukázka průchodu následujícím grafem do hloubky z vrcholu a.*

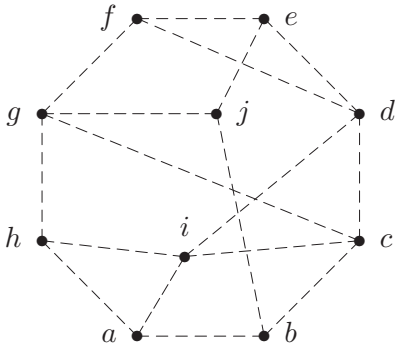


V této ukázce budeme obrázky znázorňovat stavy Algoritmu 2.4 v jednotlivých krocích takto: Neprohledané hrany jsou čárkované, prohledané hrany plnou čarou a hrany, které vedly k nalezení vrcholů, jsou tlustou čarou barevně zvýrazněny (tyto hrany tvoří strom prohledávání T). Nalezené a zpracované vrcholy jsou značeny kroužkem, právě zpracovávaný vrchol opět barevně zvýrazněn.

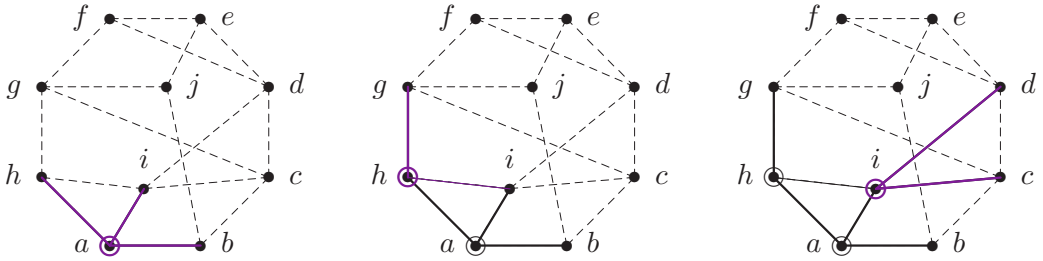


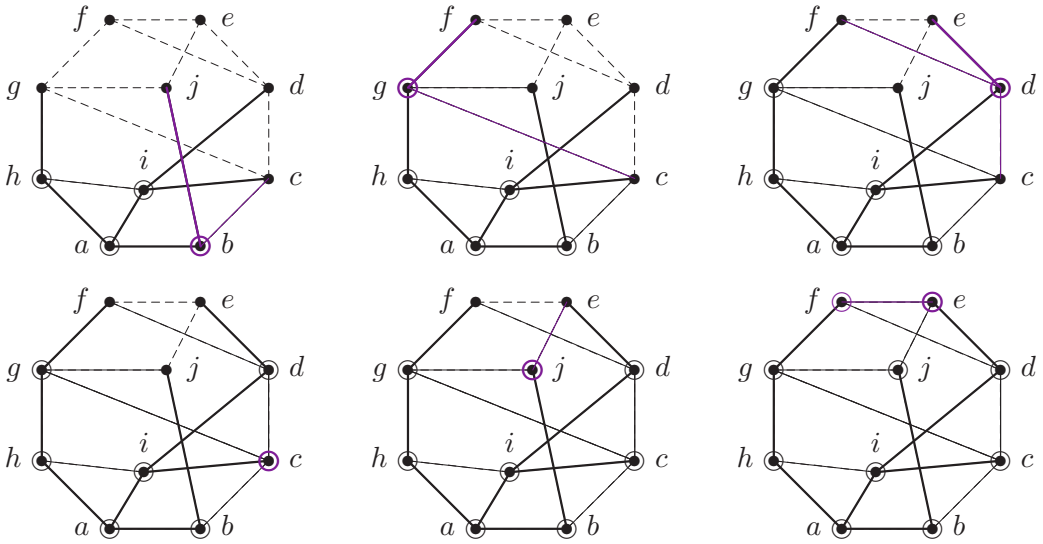
Tímto zpracování zadaného grafu skončilo. Mimo jiné jsme zjistili, že graf má jedinou komponentu souvislosti. □

Příklad 2.17. *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 2.18. *Jak byste za pomocí terminologie grafové souvislosti vyjádřili míru odolnosti například počítačové sítě S proti výpadkům jednotlivých spojení či jejích uzlů?*

V souladu s Příkladem 1.11 budeme síť S modelovat (multi)grafem tak, že jednotlivé uzly budou vrcholy a spojení budou hrany. Předpokládáme pro jednoduchost duplexní síť, tj. neorientovaný graf G_S.

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

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

Příklad 2.19. *Mějme graf H_3, jehož vrcholy jsou všechny podmnožiny množiny $\{1,2,3\}$ a hrany spojují právě disjunktní dvojice podmnožin. (Tj. H_3 má 8 vrcholů.) Rozhodněte, zda je H_3 souvislý graf, a napište, kolik má H_3 hran.*

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

Příklad 2.20. *Dokažte následující poznatek: Slabě souvislý orientovaný graf je silně souvislý právě tehdy, když každá jeho hrana leží v nějaké oritentované kružnici.*

Nechť je orientovaný graf D silně souvislý a $(u,v) \in E(D)$ je libovolná jeho hrana. Podle definice tudíž existuje v D orientovaný sled z v do u, a tudíž i orientovaná cesta P_{vu} podle Věty 2.2. Pak $P_{vu} \cup \{(u,v)\}$ je orientovaná kružnice.

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

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

Příklad 2.21. *Dokažte, že hrany každého* 6*-regulárního grafu lze zorientovat tak, aby z každého vrcholu vycházely právě* 3 *šipky.*

Myšlenka řešení je krátká, ale poměrně triková: Náš 6-regulární graf G je kreslitelný jedním uzavřeným tahem T podle Věty 2.14. Tento tah T si prostě zorientujeme – zvoleným směrem všechny jeho hrany. Jelikož T do každého vrcholu přichází třikrát a také odchází třikrát, dostaneme tři příchozí a tři odchozí šipky. □

Úlohy k řešení

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

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

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

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

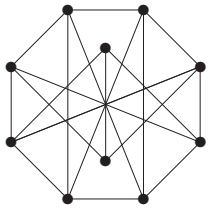
(2.6.5) *Kolik nejvíce hran může mít graf na* 10 *vrcholech,*
a) který se skládá ze tří komponent souvislosti,
b) jehož každá komponenta souvislosti má nejvíce tři vrcholy?
Tento graf také nakreslete.

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

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

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

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



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

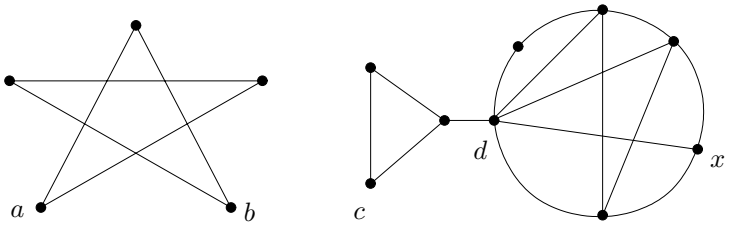
*(2.6.11)** *Dokažte, že každý* 4*-regulární graf se dá rozložit na dva* 2*-regulární grafy. (Rozklad je opakem operace množinového sjednocení, tj. sjednocením jejich množin hran i množin vrcholů dostaneme původní graf.)*

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

3 Vzdálenost a nejkratší cesty v grafech

Úvod

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



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

Z obecného pohledu při určování grafové vzdálenosti bereme do úvahy nezáporné(!) délky jednotlivých hran podél cesty. Problematika je pak beze změny aplikovatelná i na orientované grafy. Mimo potřebné definice si jako hlavní náplň lekce uvedeme principy, na kterých stojí dva klasické algoritmy pro výpočty vzdáleností v grafu: Floyd–Warshallův a Dijkstrův. Dijkstrův algoritmus je, mimo jiné aplikace, třeba základem programů vyhledávajících vlaková/autobusová spojení a ve vylepšené podobě i dnes velmi populárních GPS navigací.

Cíle

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

3.1 Vzdálenost v grafu

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

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

`Komentář:` Neformálně řečeno, vzdálenost mezi u, v je rovna *nejmenšímu počtu hran*, které musíme projít, pokud se chceme dostat z u do v. Speciálně $d_G(u,u) = 0$. Uvědomme si, že nejkratší sled je vždy cestou (vrcholy se neopakují) – Věta 2.2.

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

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

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. □

Základní 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 $d_G(u,v') = d - 1$. Obdobně uvažme libovolného souseda w' vrcholu w. Pak

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

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

Důsledek 3.4. *Algoritmus procházení grafu do šířky lze použít pro výpočet grafové vzdálenosti z daného vrcholu u.*

Toto je poměrně jednoduchá aplikace, kdy počátečnímu vrcholu u přiřadíme vzdálenost 0, a pak vždy každému dalšímu nalezenému vrcholu v přiřadíme vzdálenost o 1 větší než byla vzdálenost vrcholu, ze kterého jsme jej právě nalezli. Podle Věty 3.3 se totiž nelze později dostat k v z vrcholu bližšího k počátku u.

Komentář: Důsledek 3.4 „funguje“ jen pro vzdálenosti s jednotkovou délkou všech hran. My si dále ukážeme obecnější Dijkstrův algoritmus, který obdobným postupem počítá nejkratší vzdálenost při libovolně kladně ohodnocených délkách hran.

Některé další pojmy

Definice 3.5. Mějme graf G. Definujeme (vzhledem k G) následující pojmy a značení:

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

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

Úlohy k řešení

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

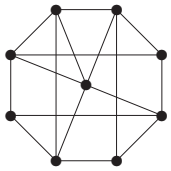
(3.1.2) *Jaká je největší vzdálenost dvou různých vrcholů v úplném bipartitním grafu $K_{33,44}$?*

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

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

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

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



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

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

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

3.2 Výpočet metriky

Než se podíváme na samotný postup výpočtu obecné vzdálenosti Dijkstrovým Algoritmem 3.12, uvedeme si ještě „globální“ pohled na soubor všech vzdáleností v grafu, tedy metriku grafu. Přínosem tohoto pohledu je především fakt, že pro výpočet celé metriky existuje velice jednoduchý tzv. dynamický Algoritmus 3.8 Floyda a Warshalla, jehož chytrá základní myšlenka je užitečná i v jiných oblastech a na jinak zadaných problémech.

`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. ***Dynamický výpočet metriky skládáním cest*** *v grafu G na množině vrcholů $V(G) = \{v_0, v_1, \ldots, v_{N-1}\}$.*

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

Věta 3.7. *Metoda 3.6 v poli* `d[i,j]` *správně vypočte vzdálenost mezi vrcholy v_i, v_j v $N = |V(G)|$ iteracích.*

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

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

Po provedení kroku $t = n - 1$ jsme hotovi. □

`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 3.8. ***Výpočet metriky grafu; Floyd–Warshall***
Tento algoritmus, pro vstupní graf G s N vrcholy zadaný svou maticí sousednosti, vypočte celou jeho metriku $d[,]$ podle postupu Metody 3.6.

```
vstup < matice sousednosti G[,] grafu na N vrcholech (číslovaných 0...N-1),
    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]);
}
výstup 'Matice vzdáleností d[,]';
```

`Komentář:` Algoritmus 3.8 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...

Úlohy k řešení

(3.2.1) *Vypočtěte si dle Algoritmu 3.8 metriku kružnice C_4.*

(3.2.2) *Zamyslete se nad implementací Algoritmu 3.8 v místě výpočtu* `d[i,t]+d[t,j]` *– není zde problém, že tyto hodnoty někdy jsou už nynější iterací změněné (kdežto jindy ještě nejsou)?*

3.3 Vážená (ohodnocená) vzdálenost

V dalším textu se již budeme věnovat cestám v grafech s obecně „dlouhými“ hranami. Pro celý zbytek lekce můžeme implicitně pracovat jak s neorientovanými, tak i s orientovanými grafy; všechny definice a výsledky platí i v orientovaném případě.

`Definice`: 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 3.9. ***(vážená vzdálenost)*** 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

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

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

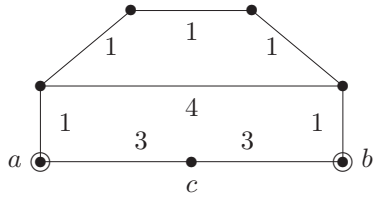
$$d^w_G(u, v) = \min\{d^w_G(S) : S \text{ je sled s konci } u, v\}\,.$$

Důležitým faktem je, že stejné definice a dobré vlastnosti zůstávají v platnosti i pro vzdálenost na orientovaných grafech. Obdobně Oddílu 3.1 nyní snadno dokážeme:

`Fakt`: Nejkratší sled v kladně váženém grafu je vždy cestou (příp. orientovanou).

Lema 3.10. *Vážená vzdálenost v nezáporně vážených grafech (i orientovaných grafech) splňuje trojúhelníkovou nerovnost.*

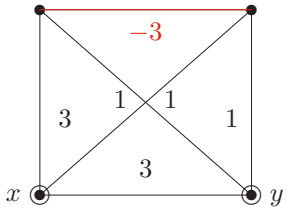
Příklad 3.11. *Podívejme se na následující ohodnocený graf (čísla u hran udávají jejich váhy–délky.)*



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

Záporné délky hran

`Komentář:` Doposud jsme důsledně vyžadovali nezápornost (lépe kladnost) ohodnocení délky hran; ale co je na hranách záporné délky tak „špatného“?



Takže jaká je v našem grafu vzdálenost mezi x, y? Je to 3 nebo 1? Ne, vzdálenost z x do y je $-\infty$ (sled může zápornou hranu libovolně krát opakovat…).

Stejný problém nastane, kdykoliv je možno opakovat záporné hrany, tedy vždy v neorientovaných grafech a částečně i v orientovaných grafech obsahujících „záporné cykly“.

Uvedené příklady problémů se zápornými délkami sledů nás vedou k úplnému vyloučení záporných vah hran v našem výkladu. V jistých omezených případech přesto má smysl hledat nejkratší cesty i v záporných vahách, ale tomu se v našem textu nebudeme věnovat z důvodu nedostatku místa.

Úlohy k řešení

(3.3.1) *Jaká je nejdelší vzdálenost mezi dvěma vrcholy v grafu z Příkladu 3.11?*

(3.3.2) *O kterém vrcholu v grafu z Příkladu 3.11 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?*

(3.3.3) *Zamyslete se, v jaké obsáhlé třídě grafů lze bez problémů korektně definovat vzdálenosti i za libovolné přítomnosti záporně vážených hran?*

3.4 Hledání nejkratší cesty

Pro nalezení nejkratší (vážené) cesty mezi dvěma vrcholy kladně váženého grafu se používá tradiční *Dijkstrův algoritmus* či jeho vhodná vylepšení. Takové algoritmy se například používají při vyhledávání vlakových spojení. Pravděpodobně se i vy někdy dostanete do situace, kdy budete nejkratší cestu hledat, proto si popsaný algoritmus včetně jeho vylepšení A^* zapamatujte.

`Poznámka`: Dijkstrův algoritmus je sice poněkud složitější než Algoritmus 3.8, ale na druhou stranu je *výrazně rychlejší*, pokud nás zajímá jen nejkratší vzdálenost z jednoho vrcholu místo všech dvojic vrcholů.

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 3.12. ***Dijkstrův pro nejkratší cestu v grafu***
Tento algoritmus nalezne nejkratší cestu mezi vrcholy u a v kladně váženého grafu G (i orientovaného), daného maticí vzdáleností sousedních vrcholů.

`vstup` < *graf na* `N` *vrcholech daný maticí vzdáleností sousedů* `del[,]` ≥ 0,
kde `del[i,j]`$=\infty$ *pokud* `j` *není sousedem* `i`
`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; stav[i] = inic.; }
vzdal[u] = 0;
while (stav[v]!=zprac.) {
    // zde nalezneme nejbližší nezpracovaný vrchol j
    m = argmin (vzdal[i]: 0≤i<N, stav[i]==inic.)
    if (vzdal[m]==MAX) return 'Cesta neexistuje';
    // vrchol m poté zpracujeme, upravíme jeho sousedy
    foreach (k soused m)
        if (vzdal[m]+del[m,k]<vzdal[k]) {
            prich[k] = m;
            vzdal[k] = vzdal[m]+del[m,k];
        }
        // prich[.] uchovává, odkud jsme se do kterého vrcholu dostali
    stav[m] = zprac.;
}
výstup 'Cesta délky vzdal[v], uložená zpětnými odkazy v poli prich[]';
```

`Poznámka`: Uvědomme si, že pokud necháme tento algoritmus proběhnout až do zpracování všech vrcholů, získáme ve `vzdal[i]` nejkratší vzdálenosti z počátečního vrcholu do všech ostatních vrcholů i.

Celkový počet kroků potřebný v Algoritmu 3.12 k nalezení nejkratší cesty z `u` do `v` je v základní „hloupé“ implementaci zhruba N^2, kde N je počet vrcholů grafu. Na druhou stranu, při lepší implementaci grafu seznamem sousedů a použití vhodné úschovny nezpracovaných vrcholů (implementované *haldou* s nalezenou vzdáleností jako klíčem) lze dosáhnout i mnohem rychlejšího běhu tohoto algoritmu na řídkých grafech – času téměř úměrného počtu hran grafu, přesněji $O\big(|E(G)| + N \log N\big)$.

Správnost Algoritmu 3.12 dokážeme snadno indukcí pomocí následujícího tvrzení.

Věta 3.13. *V každé iteraci Algoritmu 3.12 (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]`*==zprac.*

Důkaz: 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 `m` ke zpracování ten, který má ze všech nezpracovaných vrcholů nejkratší nalezenou vzdálenost od počátku `u`. To znamená, že žádná kratší cesta z `u` do `m` nevede, neboť každá „oklika“ přes jiné nezpracované vrcholy musí být delší dle výběru `m` a indukčního předpokladu. (V tomto bodě potřebujeme nezápornost ohodnocení `del[,]`.)

  Naopak každá nová nejkratší cesta z `u` do nezpracovaného vrcholu `k` procházející přes `m` musí mít `m` coby předposlední vrchol, tj. poslední hranu `mk`, a proto je upravená hodnota `vzdal[k]` správná i po přidání `m` mezi zpracované vrcholy.

□

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

Komentář: Vysvětleme si podrobněji srovnání obou předchozích algoritmů. Floyd–Warshallův Algoritmus 3.8 pro výpočet metriky je implementačně jednodušší a přímočařejší a hodí se tam, kde potřebujeme spočítat všechny dvojice vzdáleností v grafu najednou. Velkou nevýhodou však je, že vždy musíme provést N^3 kroků celého výpočtu, i když počítáme vzdálenost jen dvou blízkých vrcholů. Dijkstrův Algoritmus 3.12 na druhou stranu počítá mnohem rychleji, pokud nás zajímá jen vzdálenost z jednoho vrcholu (zhruba N^2 nebo i méně kroků).

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

Úlohy k řešení

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

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

3.5 Pokročilé plánování cest

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

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

Algoritmus A^*

- Je pouhou reimplementací Dijkstrova algoritmu (hledajícího nejkratší cestu z vrcholu u do vrcholu v v orientovaném grafu) s „vhodně“ upravenými délkami hran.

- Nechť „*potenciál*“ $p_v(x)$ udává libovolný dolní odhad vzdálenosti z vrcholu x do cíle v. Každá (orientovaná!) hrana xy grafu G, w dostane nové délkové ohodnocení $w'(xy) = w(xy) + p_v(y) - p_v(x)$. Potenciál p_v je *přípustný*, pokud všechna upravená ohodnocení jsou nezáporná, neboli $w(xy) \geq p_v(x) - p_v(y)$.

- Upravená délka libovolného sledu S z u do v pak je $d_G^{w'}(S) = d_G^w(S) + p_v(v) - p_v(u)$, což je konstantní rozdíl oproti původní délce S. Takže S je optimální pro původní délkové ohodnocení w, právě když je optimální pro nové w'.

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

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

Myšlenka „dosahu“ (reach)

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

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

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

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

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

- V řádku “`foreach (k` *soused* `m)`” (viz Algoritmus 3.12) prostě akceptujeme pouze ty sousedy `k` pro které platí $reach_G(\texttt{mk}) \geq \texttt{vzdal[m]}$.

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

Úlohy k řešení

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

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

Rozšiřující studium

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

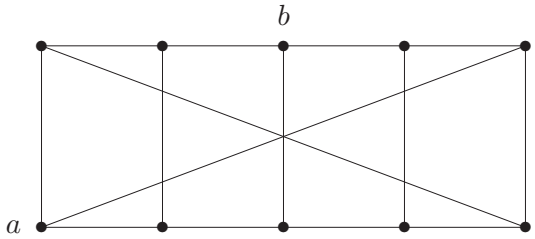
Problematika pokročilého plánování cest je v dnešní době mnohem bohatší, než jak jsme stačili naznačit v krátkém Oddíle 3.5, a čtenáře odkazujeme především na internetové zdroje. V prvé řadě je to `http://en.wikipedia.org/wiki/A*_search_algorithm` *a obousměrné varianty na* `http://en.wikipedia.org/wiki/Bidirectional_search`. *Z dalších přístupů lze kromě zmíněného klíčového slova „reach" hledat i přístupy založené na „separators", „highway / contraction hierarchies", „landmarks", „transit nodes", či různé kombinované a heuristické přístupy.*

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

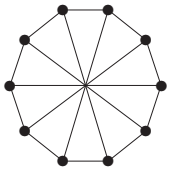
3.6 Cvičení: O cestách a grafových vzdálenostech

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

Příklad 3.14. *Určete průměr níže zakresleného grafu na 10 vrcholech a odpovězte, kolik různých dvojiv vrcholů v něm má vzdálenost rovnou průměru.*



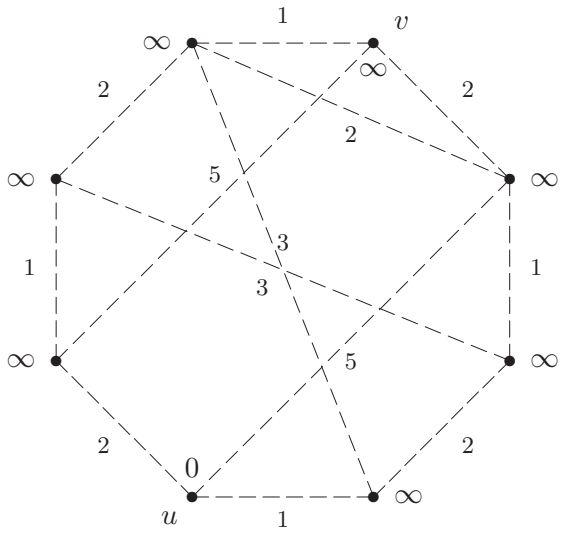
Snadným pohledem zjistíme, že například vrcholy a, b vyznačené v obrázku mají vzdálenost 3. Poté můžeme rozebráním pár možností argumentovat, že větší vzdálenost se v grafu nevyskutuje, tudíž jeho průměr je 3. Obdobně můžeme rozebráním možností spočítat, že vzdálenost 3 se nabývá právě mezi 10 dvojicemi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



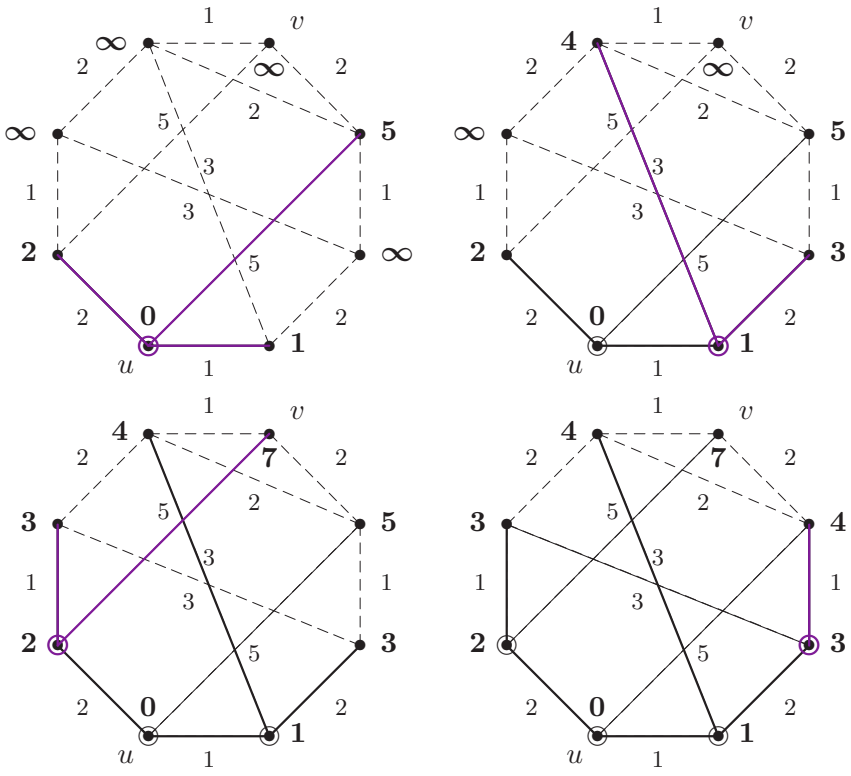
Správně doporučeným řešením je, jak už bylo zdůrazňováno několikrát, nalézt vhodnější obrázek našeho grafu. V tomto případě zde (výše). Nyní je okamžitě vidět, že všechny

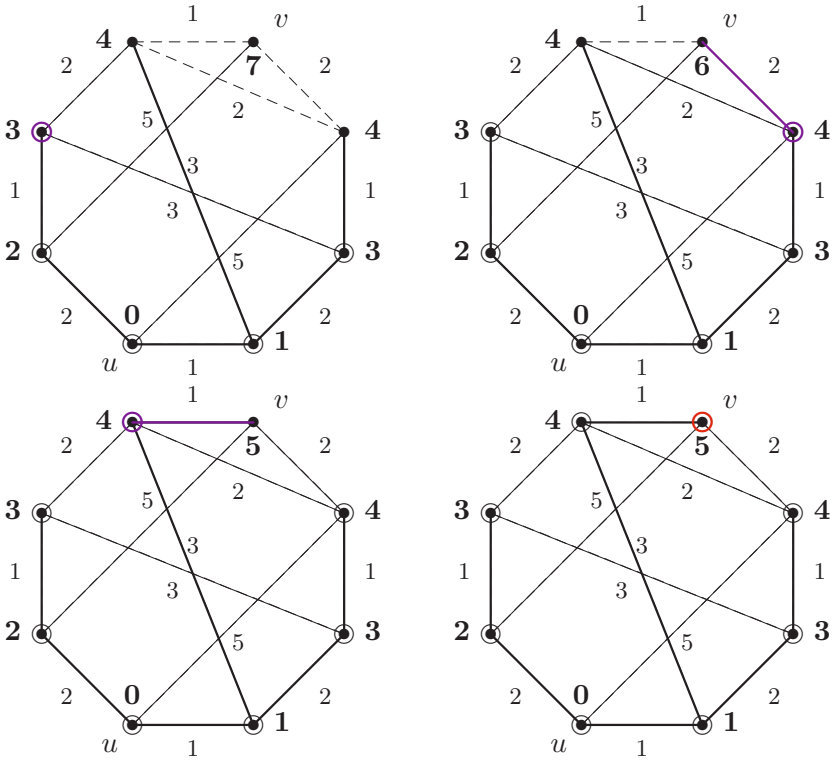
vrcholy jsou si navzájem symetrické, takže prostým jedním prohledáním do hloubky zjistíme nejvyšší vzdálenost 3 a právě dva vrcholy v této vzdálenosti z libovolného počátečního vrcholu. Matematicky rigorózně tak potvrdíme výše získané poznatky. □

Příklad 3.15. *Ukázka běhu Dijkstrova Algoritmu 3.12 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 2.16 a navíc pro každý vrchol zakreslíme jeho okamžitou dočasnou vzdálenost `vzdal[x]`. (Tj. zpracované vrcholy budeme značit kroužkem a hrany plnou čarou.) Jednotlivé kroky následují:





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

Následují dvě doplňkové ukázky vztahující se k látce Oddílu 3.5. Vzhledem ke svému specifickému zaměření na úzkou (i když významnou a populární) aplikační oblast mohou být čtenáři přeskočeny.

Příklad 3.16. *Ukažme si podrobně, jak problém plánování nejkratší cesty v reálné cestní síti (viz GPS navigace) modelovat váženým grafem.*

V prvním přiblížení je problém velmi jednoduchý, prostě:

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

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

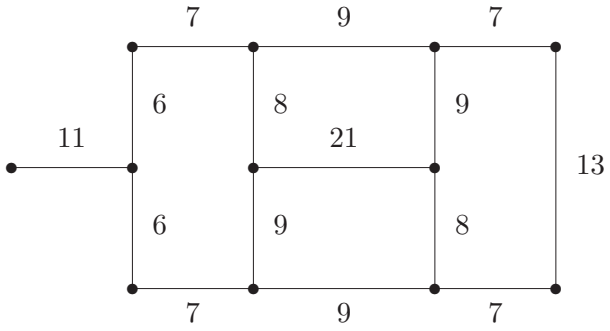
Ve druhé fázi tudíž věnujeme pozornost zmíněným dodatečným omezením přípustných cest. Ukazuje se, že jejich nejvhodnějším modelem je použití tzv. *manévrů*, které vyjadřují libovolné úseky vyžadující speciální pozornost. Formálně:

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

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

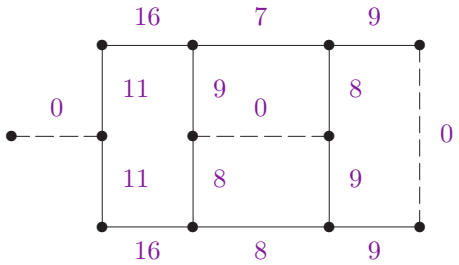
Za třetí je možno si všimnout ještě jednoho aspektu hledání najkratších cest, který nabývá na popularitě v navigačních zařízeních – váhy jednotlivých hran se v reálné cestní síti mohou v čase měnit. Jde o přirozený jev, kdy průjezdní doba na silnicích je výrazně delší v době dopravní špičky než mimo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

Příklad 3.17. *Určeme hodnotu dosahu (viz Oddíl 3.5) jednotlivých hran v následujícím váženém neorientovaném grafu:*



Které z hran se ukazují jako „bezvýznamné“?

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



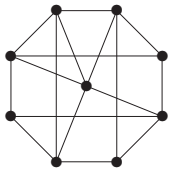
Velmi malých hodnot dosahu (konkrétně 0) nabývají vyznačené čárkované hrany, ty jsou tudíž z globálního pohledu plánování nevýznamné. Pochopitelně se jedná o příklad velmi malého grafu, takže výsledky jen lehce a dosti nepřesně naznačují skutečné chování parametru dosahu na velkých cestních sítích. □

Úlohy k řešení

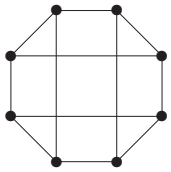
(3.6.1) *Jaká je největší vzdálenost mezi dvěma vrcholy v každém z následujících dvou grafů? Vyznačte tyto dva nejvzdálenější vrcholy.*

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

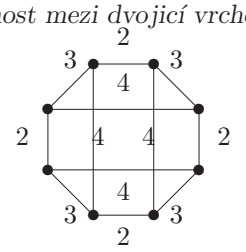
(3.6.3) *Kolik (neuspořádaných) dvojic vrcholů v následujícím grafu má vzdálenost právě* 3*?*



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



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



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

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

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

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

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

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

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

(3.6.13) *V návaznosti na Příklad 3.16 nalezněte ukázku cestní sítě s penalizovanými manévry, ve které existuje nejkratší sled (vzhledem k penalizované váze) opakující vrcholy.*

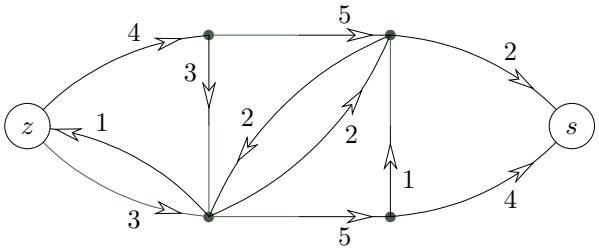
*(3.6.14) *Co by mělo být hlavní podstatnou změnou Dijkstrova algoritmu, aby jeho prostřednictvím bylo možno počítat nejkratší sledy vzhledem k manévrům (viz Příklad 3.16)?*

4 Toky v sítích

Úvod

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

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



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

Cíle

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

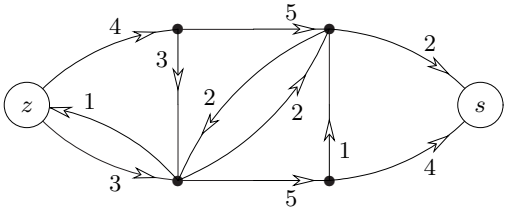
4.1 Definice sítě a toku

Základní strukturou pro reprezentaci sítí je orientovaný graf. Vrcholy grafu modelují jednotlivé uzly sítě a hrany jejich spojnice. Vzpomeňme si (Definice 1.10), že v orientovaných grafech je každá hrana tvořena uspořádanou dvojicí (u, v) vrcholů grafu, a tudíž taková hrana má *směr z vrcholu u do v*.

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

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

`Komentář:`



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

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

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

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

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

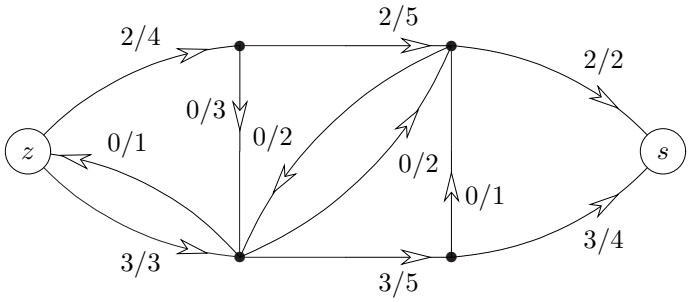
Definice 4.2. ***Tok*** v síti $S = (G, z, s, w)$
je funkce $f : E(G) \to \mathbb{R}_0^+$ splňující

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

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

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

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



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

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

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

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

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

Úlohy k řešení

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

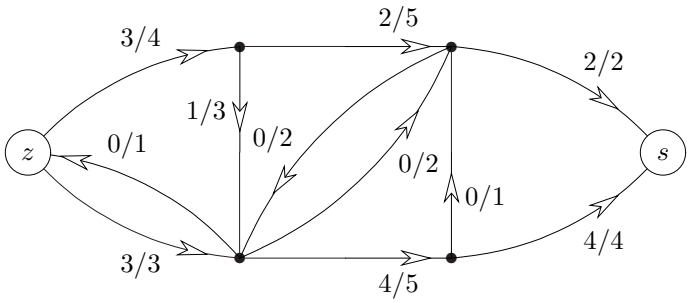
(4.1.2) *V dané síti máme dva různé toky stejné velikosti. Čím se od sebe „liší“?*

4.2 Hledání maximálního toku

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

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

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



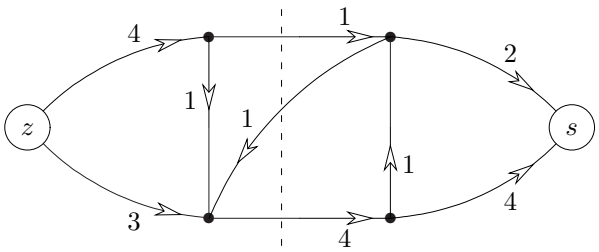
Jak však poznáme, že větší tok již v dané síti neexistuje? V této konkrétní ukázce to není obtížné, vidíme totiž, že obě dvě hrany přicházející do stoku mají součet kapacit $2 + 4 + 6$, takže více než 6 do stoku ani přitéct nemůže. V obecnosti lze použít obdobnou úvahu, kdy najdeme podmnožinu hran, které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

Pojem řezu v síti

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

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

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



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

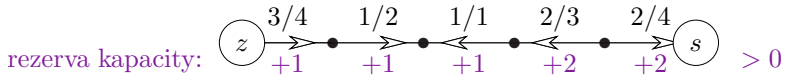
Důkaz Věty 4.5 bude proveden následujícím Algoritmem 4.7.

Nenasycené cesty v síti

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

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

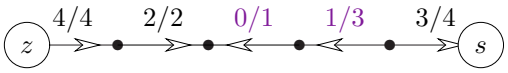
`Komentář:` Zde vidíme příklad nenasycené cesty ze zdroje do stoku s minimální rezervou kapacity +1.



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

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

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



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

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

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

Výsledný f§ tok pak bude opět přípustný.

Základní algoritmus nenasycených cest

Implementační detaily Metody 4.6 následují popisem konkrétního algoritmu.

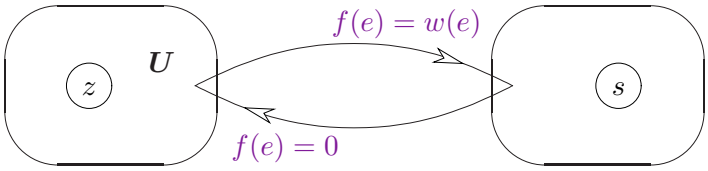
Algoritmus 4.7. ***Ford–Fulkersonův pro tok v síti.***

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

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

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

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



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

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

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

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

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

□

Z důkazu Algoritmu 4.7 odvodíme několik zajímavých faktů:

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

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

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

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

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

Vylepšení algoritmu nenasycených cest

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

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

Ještě lepších výsledků dosahují následující "chytré" algoritmy.

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

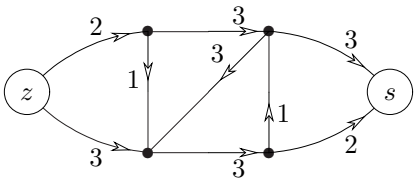
- Prohledáváním sítě do šířky nalezneme všechny nenasycené cesty nejkratší délky souběžně, které nám vytvoří "vrstvenou síť" (vrstvy odpovídají nenasycené vzdálenosti vrcholů od zdroje).
- Nalezené nenasycené cesty pak nasytíme novým prohledáním vrstvené sítě.

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

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

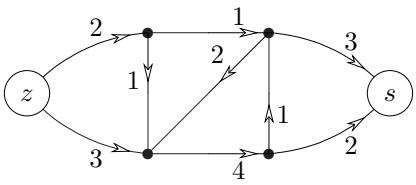
Úlohy k řešení

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



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

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



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

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

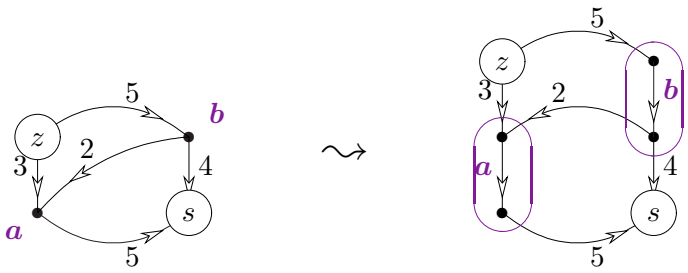
4.3 Zobecnění definice sítí

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

Sítě s kapacitami vrcholů

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

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



Sítě s dolními kapacitami

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

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

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

Tzv. vícekomoditní toky

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

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

Úlohy k řešení

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

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

4.4 Speciální aplikace sítí

Bipartitní párování

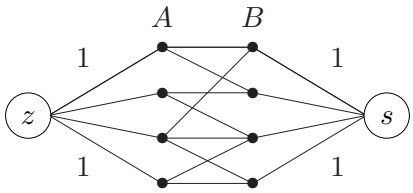
Biparitní grafy jsou grafy, jejichž vrcholy lze rozdělit do dvou množin tak, že všechny hrany vedou jen mezi těmito množinami, neboli podgrafy úplných bipartitních grafů.

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

> `Komentář:` Pojem (bipartitního) párování má přirozenou motivaci v mezilidských vztazích. Pokud 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, tradiční partnerské vztahy by měly představovat párování v popsaném grafu.

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

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



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

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

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

Vyšší grafová souvislost

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

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

Použitím tohoto tvrzení pro všechny dvojice vrcholů grafu snadno dokážeme dříve uvedenou důležitou Větu 2.7 (Mengerovu). Ještě jiné použití si ukážeme na problému výběru reprezentantů množin.

Výběr různých reprezentantů

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

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

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

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

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

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

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

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

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

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

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

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

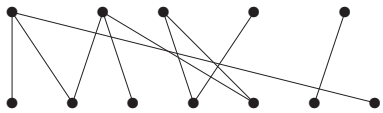
Segmentace obrazu

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

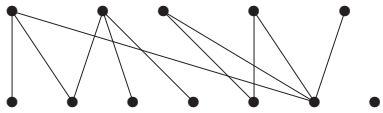
- Vstupem je matice obrazových bodů (pixelů), z nichž každý má přiřazeny hodnoty pravděpodobnosti bytí v popředí a bytí v pozadí. Dále jsou určeny „separační penalizace“ pro dvojice sousedních pixelů.
- Síť zkonstruhujeme přidáním univerzálního zdroje z a stoku s, přičemž hrany ze z do pixelů mají kapacitu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

Úlohy k řešení

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



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



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

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

Rozšiřující studium

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

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

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

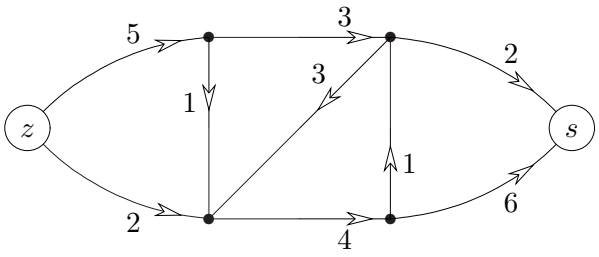
Teorie párování (Oddíl 4.4) tvoří samostatnou rozvinutou oblast grafů, mající zajímavé aplikace v jiných oborech jako statistické fyzice. Zájemce o obecnou teorii odkazujeme třeba na [1, Chapter 2] nebo internetově na `http://en.wikipedia.org/wiki/Matching`.

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

4.5 Cvičení: Příklady toků v sítích

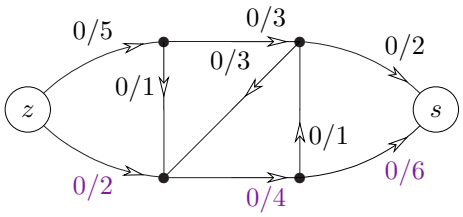
Mnohé příklady definic sítí a toků v nich byly již zmíněny v předchozím textu lekce, a proto ve cvičení přejdeme rovnou k ukázkovému řešení úloh souvisejících s toky v sítích. Cílem je nejen se naučit používat poměrně jednoduchý algoritmus hledání maximálního toku v síti, ale především se seznámit se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

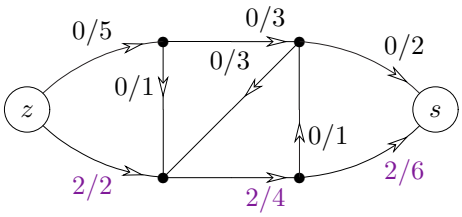
Příklad 4.16. *Jaký je maximální tok znázorněnou sítí ze z do s?*



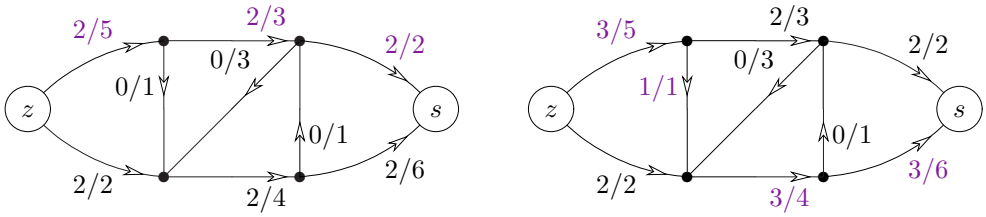
Na tomto příkladě si ukážeme průběh Algoritmu 4.7.

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

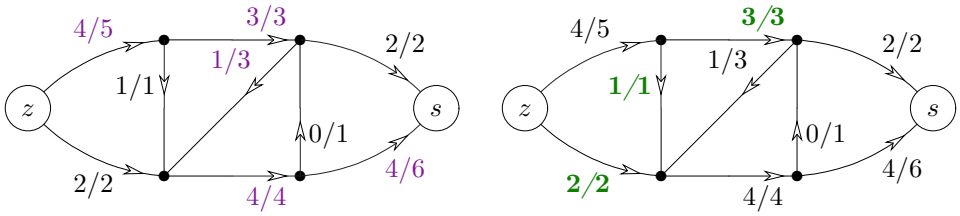




V dalším kroku najdeme nenasycenou cestu délky 3 vedoucí vrchem grafu. Její minimální rezerva kapacity je opět 2, takže ji o tolik nasytíme (viz další obrázek vlevo). Nyní již nenasycená cesta délky 3 neexistuje, takže najdeme nenasycenou cestu délky 4 s rezervou kapacity 1, kterou hned nasytíme (viz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

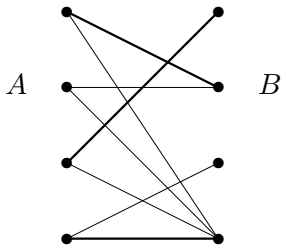


Obdobně ještě najdeme nenasycenou cestu délky 5, kterou také nasytíme o 1. Závěrečný obrázek nám ukazuje všechny nenasycené hrany, mezi kterými již nevede žádná nenasycená cesta ze z do s, takže máme maximální tok velikosti 6 v naší síti.



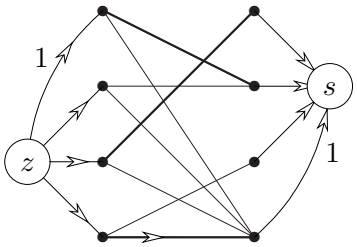
Zároveň je na posledním obrázku vyznačen i příslušný minimální řez velikosti 6. □

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



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

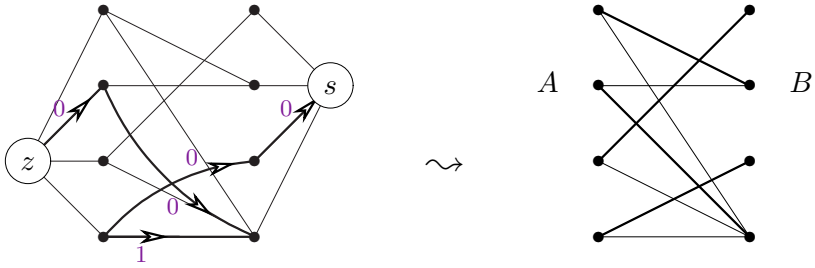
Síť konstruhovaná Algoritmem 4.13 vypadá (podle definice) následovně:



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

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

vznikne nové (již perfektní) párování vpravo:



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

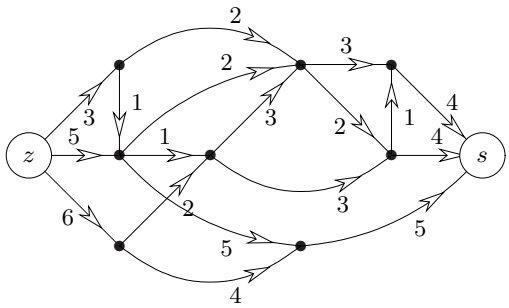
Příklad 4.18. *Mějme neorientovaný graf G. Pak hrany G lze zorientovat tak, aby z každého vrcholu vycházela nejvýše jedna šipka, právě když žádný podgraf v G neobsahuje více hran než vrcholů. Dokažte.*

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

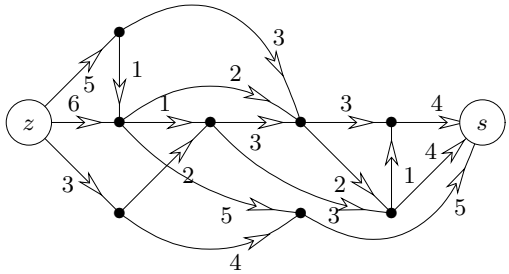
Podle Hallovy věty má systém hran G různé reprezentanty, právě když pro každé $F \subseteq E(G)$ platí, že $|\bigcup F| \geq |F|$, kde $\bigcup F$ zkráceně zapisuje podmnožinu těch vrcholů incidentních s některou hranou F. Potom pokud některý podgraf $H \subseteq G$ má více hran než vrcholů, je $|\bigcup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

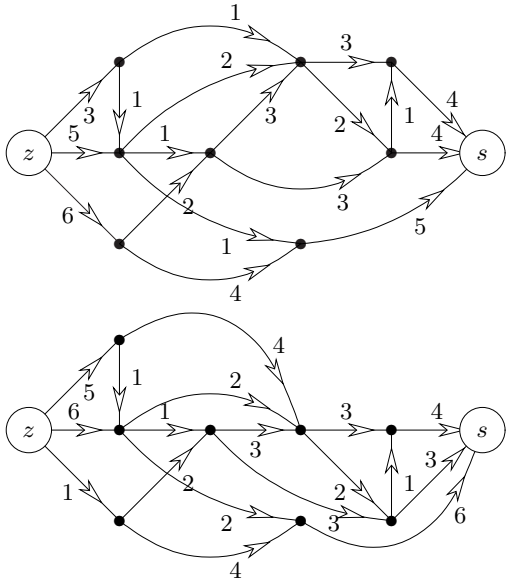
Úlohy k řešení

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



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





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

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

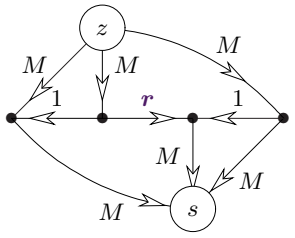
(4.5.4) *Má tento seznam množin systém různých reprezentantů? Pokud ano, vyznačte je v množinách, jinak správně zdůvodněte, proč systém různých reprezentantů nemají.*

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

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

(4.5.6) *Představme si graf G nakreslený do roviny bez křížení hran (viz Lekce 8 pro bližší definice). Nechť u, v jsou dva různé vrcholy v nakreslení. Pak z u do v lze dojít „procházkou“ po stěnách grafu s překročením nejvýše k jiných vrcholů, právě když neexistuje $k+1$ vzájemně disjunktních vnořených kružnic oddělujících u od v.*

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

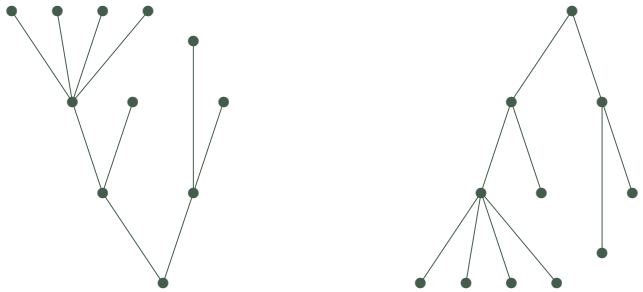


$r = \frac{\sqrt{5}-1}{2}$, $M \geq 2$

5 Stromy – grafy bez kružnic

Úvod

Jednou ze základních podtříd všech grafů jsou tzv. stromy. Jedná se o minimální souvislé grafy, neboli souvislé grafy bez kružnic. Na druhou stranu lze stromy vnímat jako vhodné zobecnění cest, zachovávající mnoho ze „snadnosti“ v zacházení se stromy. Stromy se již nepřímo vyskytly v předchozích lekcích, jako třeba v případě stromu procházení grafu (Příklad 2.16) nebo u stromu nejkratších cest z vrcholu (Dijkstrův algoritmus). Co dále vidíme na následujícím obrázku?



Důležitost stromů – jak v matematice, tak i v informatických aplikacích – úzce souvisí s absencí kružnic a z ní vyplývající snadnosti jejich popisu a zpracování. Dobře je to vidět třeba v situacích vyžadujících jednoduchý popis hierarchických entit nebo v různých datových strukturách založených na stromech. Mimo uvedené ještě stojí za zmínku tradiční využití stromů v rodokmenech, které motivuje většinu zavedené terminologie na stromech.

Cíle

Úkolem této lekce je hlavně popsat pojem stromu coby minimálního souvislého grafu a grafu bez kružnic a ukázat základní vlastnosti stromů včetně jejich důkazů. Vedle toho je věnována pozornost kořenovým a uspořádaným stromům a je zaveden pojem koster grafu (na který navážeme i v Oddíle 6.1).

5.1 Základní vlastnosti stromů

Začněme definicí stromu a přehledem základních vlastností stromů.

Definice 5.1. ***Strom*** je jednoduchý souvislý graf T bez kružnic.

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

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

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

Následující vlastnosti stromů, která uvádíme i s jednoduchými důkazy, dokonce plně popisují stromy a mohou tudíž být případně použity místo uvedené Definice 5.1. Jedná se sice o dosti formální látku na úvod, avšak alespoň zběžné porozumění uvedeným vlastnostem je potřebné pro další práci se stromy a s jinými strukturami založenými na stromech.

Lema 5.2. *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 5.3. *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 5.2 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 5.4. *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 $H = P_1 \Delta P_2$ 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 5.2, spor. Proto cesta mezi u a v existuje jen jedna. □

Důsledek 5.5. *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 5.4. □

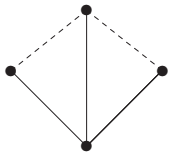
Věta 5.6. *Strom je minimální souvislý graf (na daných vrcholech).*

`Důkaz`: Strom je souvislý podle definice. Pokud by ale vypuštěním hrany $e = uv$ ze stromu T vznikl opět souvislý graf, pak by mezi u, v v T existovaly dvě cesty (dohromady kružnice) – hrana e a jiná cesta v $T - e$. To je ve sporu s Větou 5.4. Naopak, pokud by souvislý graf měl kružnici, zůstal by souvislý i po vypuštění některé hrany té kružnice. Proto každý minimální souvislý graf (na daných vrcholech) je stromem. Tudíž strom je právě minimálním souvislým grafem na daných vrcholech. □

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

Příklad 5.7. *Kolik nejvýše kružnic vznikne v grafu, který vytvoříme ze stromu přidáním dvou hran?*

Přidáním jedné hrany do stromu T vznikne jedna kružnice dle Důsledku 5.5. 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 po přidání dvou hran e, f do stromu T vzniknout nemohou: Podle Důsledku 5.5 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 5.4 dostáváme spor s faktem, že podgrafy stromů s hranami musí obsahovat vrchol stupně 1. □

Úlohy k řešení

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

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

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

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

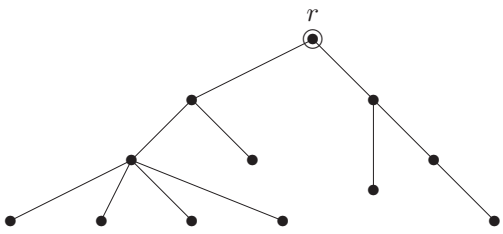
(5.1.5) *Kolik je neisomorfních stromů na čtyřech vrcholech? Nakreslete si je.*

5.2 Kořenové stromy

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

Definice 5.8. ***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:

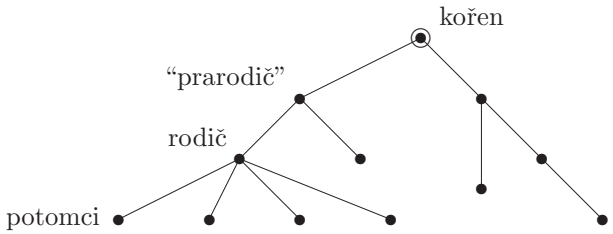


Zajímavostí je, že v informatice stromy většinou rostou od kořene směrem dolů. (Však také nejsme v biologii...)

`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.

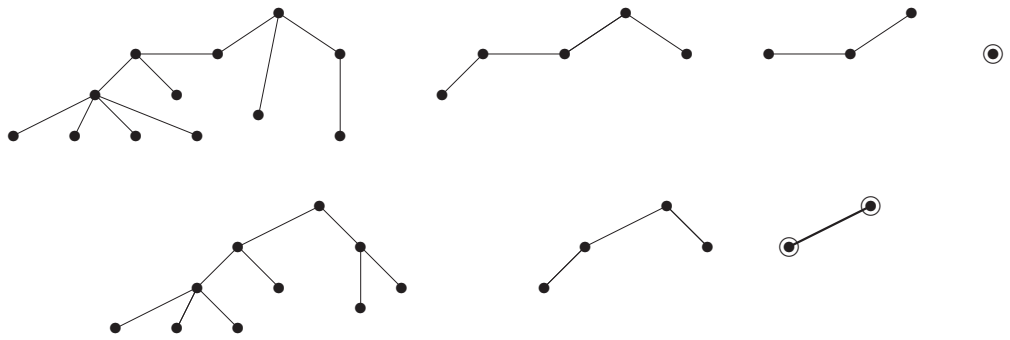
Centrum stromu

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

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

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

Příklad 5.9. *Ilustrací definice centra jsou následující dvě ukázky jeho nalezení (čtěme obrázky postupu zleva doprava):*



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

`Komentář:` Pokud chceme danému (abstraktnímu) stromu přiřadit jednoznačně kořen, je nejlepší jej přiřadit centru stromu. Speciálně, pokud je centrem hrana, bude kořenem nový

vrchol „rozdělující“ tuto hranu na dvě. Viz předchozí příklad dole:

Tvrzení 5.10. *Centrum stromu podle předchozí definice přesně odpovídá „vzdálenostnímu“ centru danému Definicí 3.5.*

`Důkaz` (náznak): Použijeme indukci vzhledem k poloměru stromu T s přímým srovnáním obou definic. □

Uspořádání na stromu

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

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

`Komentář:` Uspořádaný kořenový strom si jinak také můžeme představit jako strom s vyznačeným kořenem a pevně zvoleným nakreslením v rovině bez křížení hran. Nakreslení hran potomků vzhledem k hraně rodiče pak udává (ve zvolené orientaci) pořadí potomků. Tento pohled vede k názvu pěstovaný strom.

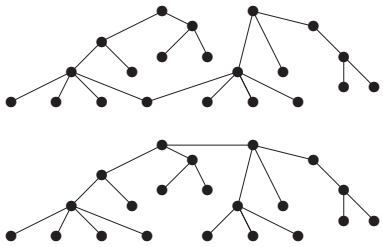
Uspořádání potomků vrcholu ve stromu je přirozeně požadováno v mnoha praktických situacích. Například ve stromových datových strukturách jsou často potomci explicitně seřazeni podle daného klíče, jako třeba ve vyhledávacích binárních stromech. I v případech, kdy uspořádání potomků ve stromě není dáno,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

Úlohy k řešení

(5.2.1) *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*

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

(5.2.2) *Určete centra následujících dvou stromů:*



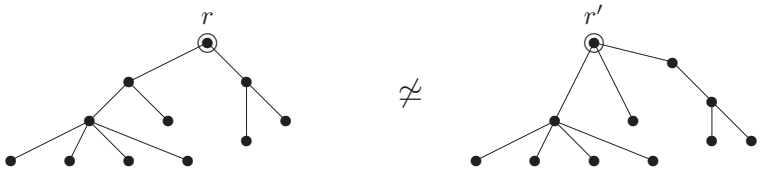
(5.2.3) *Mějme libovolný strom s 33 vrcholy. Kolik jeho listů postupně odebereme, než určíme centrum? (Je to jednoznačné?)*

5.3 Isomorfismus stromů

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

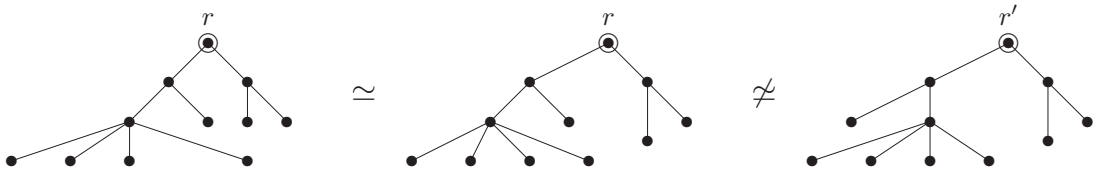
`Definice`: (Dva stromy jsou isomorfní pokud jsou isomorfní jako grafy.) Dva kořenové stromy T, r a T', r' jsou *isomorfní* pokud existuje isomorfismus mezi stromy T a T', který kořen r zobrazuje na kořen r'.

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



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

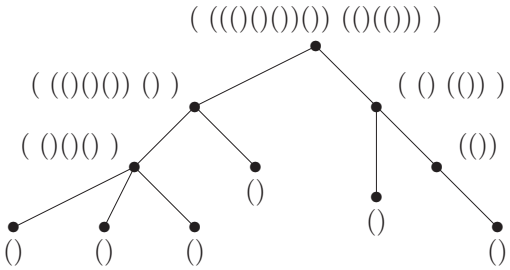
`Komentář:` Například z následujících tří isomorfních kořenových stromů jsou první isomorfní i coby uspořádané kořenové stromy, avšak třetí s nimi takto *isomorfní není*, neboť pořadí potomků levého syna kořene se liší.



Kódování uspořádaných kořenových stromů

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

`Definice`:



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

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

Klíčovým faktem o kódech pěstovaných stromů je toto tvrzení:

Lema 5.11. *Dva uspořádané kořenové (pěstované) stromy jsou isomorfní právě když jejich kódy získané podle předchozího popisu jsou shodné řetězce.*

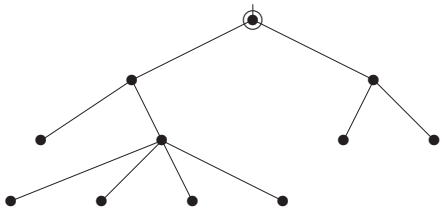
Důkaz: Postupujeme indukcí podle hloubky stromu (nejdelší vzdálenosti z kořene). Báze je zřejmá, strom hloubky 0 je jedinečný s kódem '()'. Dále podle definice jsou dva uspořádané kořenové stromy isomorfní, právě když v daném pořadí jsou po dvojicích isomorfní podstromy jejich kořenů, neboli podle indukčního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

Příklad 5.12. *Nakreslete jako pěstovaný strom ten odpovídající závorkovému kódu*

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

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

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



□

Efektivní algoritmus isomorfismu pro stromy

Při určování isomorfismu obecných stromů použijeme Lema 5.11 a jejich jednoznačnou reprezentaci uspořádanými kořenovými stromy, ve které kořen volíme v centru a potomky seřadíme podle jejich kódů vzestupně abecedně. Jinak se dá také říci, že kód přiřazený stromu T je lexikograficky nejmenší ze všech kódů uspořádaných stromů vzniklých z T s kořenem v jeho centru. Takový kód je zřejmě určující vlastností stromu T jako takového, a proto správnost následujícího algoritmu můžeme snadno nahlédnout níže.

Algoritmus 5.13. ***Určení isomorfismu dvou stromů.***
Pro dané dva obecné stromy T a U implementujeme algoritmus zjišťující isomorfismus $T \simeq^{?} U$ následovně v symbolickém zápisu:

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

Funkce minimalni_kod(strom X, vrchol r) {
    if (|V(X)|==1) return "()";
    d = počet komponent grafu X-r, tj. podstromů kořene r;
    for (i=1,...,d) {
```

```
        Y[i] =  i-tá souvislá komponenta grafu X-r;
        s[i] =  i-tý soused r, tj. kořen podstromu Y[i];
        k[i] = minimalni_kod(Y[i],s[i]);
    }
    sort k[1]<=k[2]<=...<=k[d] lexikograficky (abecedně);
    return "("+k[1]+...+k[d]+")";
}

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

Důkaz správnosti Algoritmu 5.13 je podán v následujícím tvrzení.

Věta 5.14. *Mějme dva stromy T, U o stejném počtu vrcholů a nechť (T', r) a (U', s) jsou po řadě jejich kořenové stromy získané v první části Algoritmu 5.13 (kde r, s jsou centra T, U). Pak platí:*

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

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

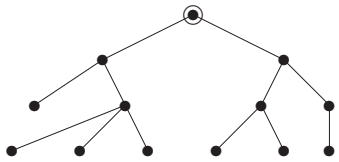
$$\texttt{minimalni\_kod}(\texttt{T}', \texttt{r}) = \texttt{minimalni\_kod}(\texttt{U}', \texttt{s}).$$

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

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

Úlohy k řešení

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



(5.3.2) *Nakreslete pěstovaný strom odpovídající závorkovému kódu*

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

5.4 Kostry grafů

Pohled na stromy coby minimální souvislé grafy přináší i pojem *kostry* grafu – minimálního propojení mezi všemi jeho vrcholy. V tomto oddíle se na problematiku

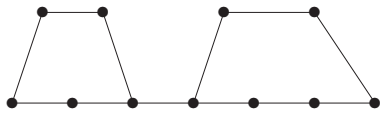
koster grafů podíváme pohledem hezké matematické hříčky (viz Věta 5.17). Na druhou stranu důležitosti problému hledání minimální kostry grafu (tzv. MST problém) bude věnován také pozdější Oddíl 6.1.

Definice 5.15. ***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 5.16. *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 Cayleye patří ke „krásným drahokamům“ teorie grafů.

Věta 5.17. *Úplný graf K_n pro $n > 0$ má přesně n^{n-2} koster.*

`Komentář:` Pro tento výsledek existuje několik různých velmi pěkných důkazů, které si zájemci mohou najít v literatuře [5]. My si dále uvedeme ještě jeden zcela obecný (a z pohledu výpočetní složitosti docela překvapivý) výsledek.

`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 5.18. *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

(5.4.1) *Kolik koster má graf K_1?*

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

*(5.4.3)** *Ověřte předchozí výsledek podle Věty 5.18.*

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

*(5.4.5)** *Odvoďte pomocí Věty 5.18, kolik koster má úplný bipartitní graf $K_{m,n}$.*

(5.4.6) *Dokážete nalézt souvislý graf s přesně* 2007 *kostrami bez kružnice* C_{2007} *coby podgrafem?*

Rozšiřující studium

Stromům, jejich isomorfismu a kódování se věnují [5, Oddíly 5.1, 5.2]. Zájemci o hlubší studium počítání koster v grafech a příslušné důkazy si mohou přečíst poměrně pokročilou [5, Kapitolu 8]. Tam lze také nalézt obtížný důkaz obecné Věty 5.18.

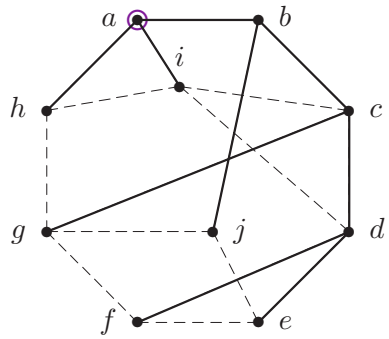
Pro informatika dále stromy představují základy většiny pokročilých datových struktur (například vyhledávací stromy, haldy, derivační stromy, apod.) a také některých algoritmických postupů jako například tzv. backtrackingu (viz také postupy prohledávání grafů v Lekci 2). Další informace o algoritmickém využití stromů najdeme v příští Lekci 6.

5.5 Cvičení: Příklady o stromech a kostrách

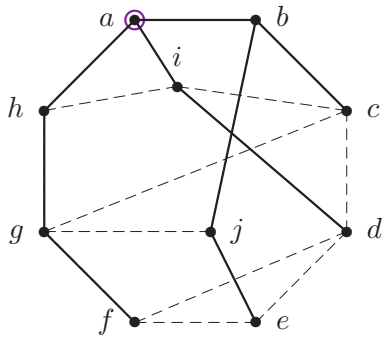
Příklad 5.19. *Zakresleme si stromy prohledávání grafu pro Příklady 2.16 a 2.17. Stromem prohledávání rozumíme kostru daného grafu tvořenou právě těmi hranami, jejichž projitím došlo k objevení nového vrcholu.*

Stromy prohledávání zmíněných instancí algoritmu prohledávání jsou zde:

DFS

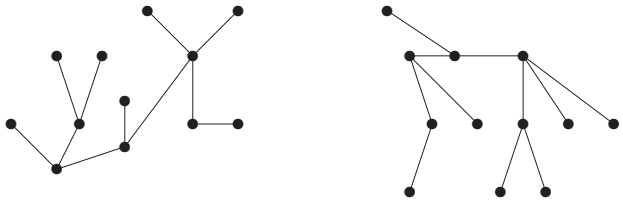


BFS

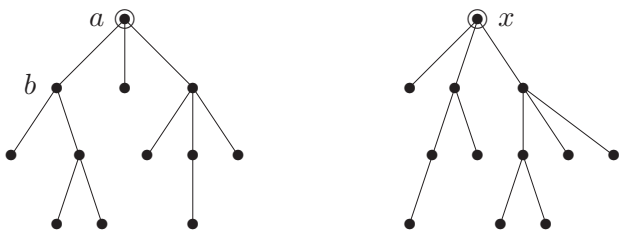


Je na nich zároveň hezky vidět základní strukturální rozdíl mezi prohledáváním do hloubky (DFS) a do šířky (BFS). □

Příklad 5.20. *Zjistěte, zda následující dva stromy jsou isomorfní pomocí Algoritmu 5.13.*



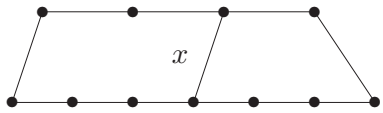
Nejprve si ověříme, že stromy mají stejný počet vrcholů (hran je pak také stejně). K daným dvěma stromům najdeme jejich centra a překreslíme si je jako kořenové stromy s kořeny v centrech.



Nyní přejdeme k určení minimálního závorkového kódu pro levý strom (Algoritmus 5.13): 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 5.21. *Kolik různých koster má tento graf?*

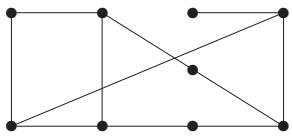


Jak vidíme, opět musíme vymazat dvě hrany z grafu, aby zbyla kostra. Nelze však vypouštět jednu hranu z levé kružnice a jednu z pravé, neboť by výběry nebyly nezávislé – hrana x je oběma sdílená. Podíváme se tedy na řešení jako na součet disjunktních možností; počtu koster obsahujících x a počtu koster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

Pokud hranu x vymažeme, zbude kružnice délky 11 a ta má 11 koster. Pokud naopak hranu x zachováme, musíme z "levého oblouku" kružnice od x vymazat jednu ze 6 hran a z "pravého oblouku" 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. □

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?*

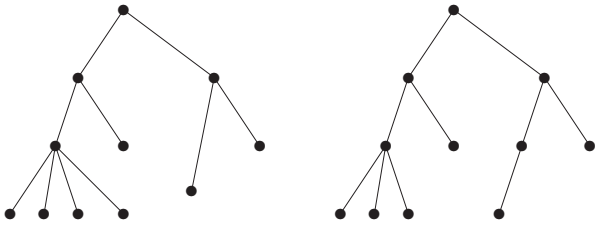
(5.5.4) *Kolik komponent souvislosti má les s* 2004 *vrcholy a* 1993 *hranami?*

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

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

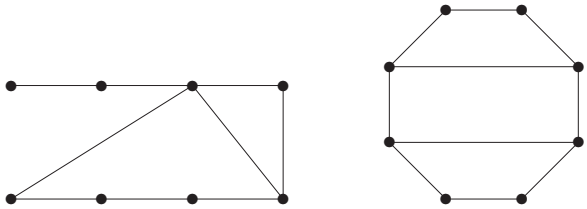
(5.5.7) *Mějme čísla* $\{10, 11, 13, 14, 15, 21\}$ *a spojme dvojice z nich hranami, pokud jsou soudělná. Je výsledný graf stromem? A aspoň lesem?*

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



(5.5.9) *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.*

*(5.5.10) *Kolik různých koster mají grafy na následujícím obrázku?*



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

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

*(5.5.13) *Dokážete nalézt souvislý graf s přesně* 2003 *kostrami bez kružnice C_{2003} coby podgrafem? Abyste si ušetřili čas,* 2003 *je prvočíslo.*

(5.5.14) *Ověřte řešení Příkladu 5.21 podle Věty 5.18.*

*(5.5.15) *Kolik koster má graf $K_n - e$ (tj. úplný po odebrání jedné hrany)?*

Návod: Zamyslete se, kolik koster prochází jednou (lib.) hranou K_n.

Část III
Další Užitečné Oblasti a Problémy

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

6 Minimální kostry, Hladový algoritmus

Úvod

Kromě teoretických "hrátek" mají kostry grafů (Oddíl 5.4) 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 5.6 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.

6.1 Hledání minimální kostry

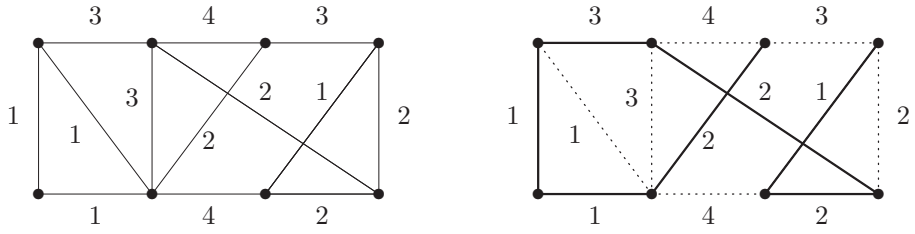
Problém 6.1. ***Problém minimální kostry (MST)***
Je dán (souvislý) vážený graf G, w s nezáporným ohodnocením hran w. Otázkou je najít kostru T v G, která má nejmenší možné celkové ohodnocení. Formálně

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

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

`Komentář:` Podrobnou ukázku průběhu hladového algoritmu čtenář najde v Příkladě 6.15.

`Důkaz` správnosti Algoritmu 6.2:
Nechť T je množina hran získaná v Algoritmu 6.2 a nechť hrany jsou již seřazené $w(e_1) \leq w(e_2) \leq \cdots \leq w(e_m)$. Vezměme množinu hran T_0 té minimální kostry (může jich být více se stejnou hodnotou), která se s T shoduje na co nejvíce prvních hranách. Pokud $T_0 = T$, algoritmus pracoval správně.

Předpokládejme tedy, že $T_0 \neq T$, a ukážeme spor, tj. že toto nemůže ve skutečnosti nastat. Označme $j > 0$ takový index, že se množiny T_0 a T shodují na prvních $j-1$ hranách $e_1, \ldots, e_{j-1}$, ale neshodují se na e_j. To znamená, že $e_j \in T$, ale $e_j \notin T_0$. (Jistě nemůže nastat $e_j \notin T$, ale $e_j \in T_0$.) Podle Důsledku 5.5 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 $\big(T_0 \setminus \{e_k\}\big) \cup \{e_j\}$ (vzniklá nahrazením hrany e_k hranou e_j) není horší než T_0 a měli jsme ji v naší úvaze zvolit místo T_0. To je hledaný spor. □

`Komentář:` Správný pohled na předchozí důkaz by měl být následovný: Předpokládali jsme, že nalezená kostra T se s některou optimální kostrou shoduje aspoň na prvních $j-1$ hranách. Poté jsme ukázali, že některou další hranu e_k v (předpokládané) optimální kostře lze zaměnit hranou e_j, a tudíž dosáhnout shodu aspoň na prvních j hranách. Dalšími iteracemi záměn ukážeme úplnou shodu naší nalezené kostry T s některou optimální kostrou. V našem důkaze jsme se vlastně zaměřili na fakt, že ta poslední iterace záměn nemůže selhat. Nakreslete si tento důkaz obrázkem!

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 6.3. ***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 6.4. ***Borůvkův pro minimální kostru*** (náznak).
Toto je poněkud složitější algoritmus, chová se jako Jarníkův algoritmus spuštěný zároveň ze všech vrcholů grafu najednou. Detaily lze nalézt v literatuře [5, Oddíl 5.4].

Doplňkové otázky

(6.1.1) *Co se stane, pokud v Algoritmu 6.2 seřadíme hrany naopak, tedy sestupně?*

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

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

6.2 Hladový algoritmus v obecnosti

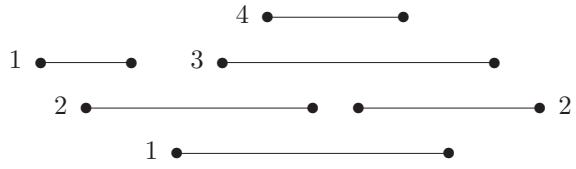
Asi nejprimitivnějším možným přístupem při řešení diskrétních optimalizačních úloh je postupovat stylem beru vždy to nejlepší, co se zrovna nabízí... Tento postup obecně v češtině nazýváme *hladovým algoritmem*, i když lepší by bylo použít správnější překlad anglického "greedy", tedy nenasystný algoritmus. 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 6.5. ***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?

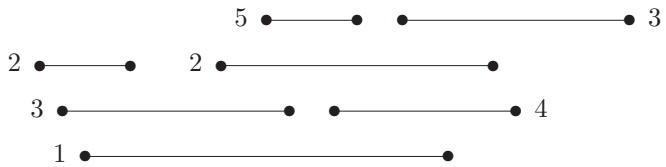
`Poznámka`: Uvedené zadání může být kombinatoricky popsáno také jako problém optimálního obarvení daného intervalového grafu (vrcholy jsou intervaly úkolů a hrany znázorňují překrývání intervalů).

Algoritmus 6.6. ***Hladový algoritmus rozdělení pracovních úkolů.***
Problém 6.5 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

(6.2.1) *Proč tedy nestačí méně než 4 pracovníci pro splnění pracovních úkolů v zadání za Problémem 6.5?*

(6.2.2) *Co kdybychom v hladovém řešení Problému 6.5 seřadili úkoly podle časů jejich ukončení?*

6.3 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 6.12).

Definice 6.7. ***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 6.8. *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 6.9. *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. □

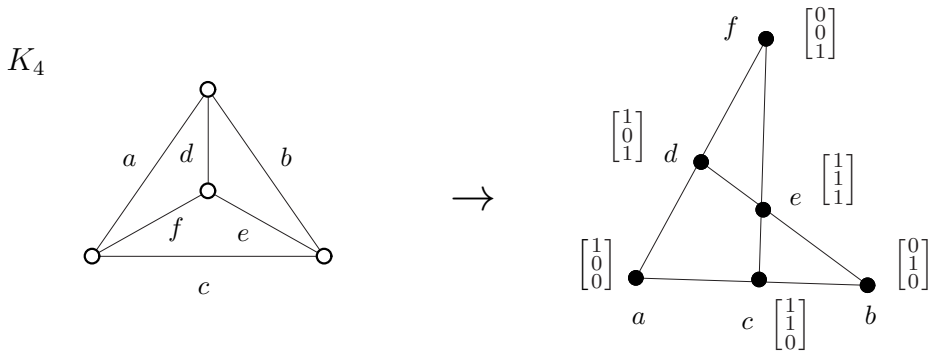
`Definice`: Řekneme, že podmnožina hran $F \subset E(G)$ je *acyklická*, pokud podgraf s vrcholy $V(G)$ a hranami z F nemá kružnici.

Lema 6.10. *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 6.9, 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 6.10 tvoří systém všech acyklických podmnožin hran v (libovolném) grafu G matroid. Tento matroid nazýváme *matroidem kružnic* grafu G.
V analogií 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 6.11. ***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 6.2 je přesně instancí Algoritmu 6.11.

Věta 6.12. *Algoritmus 6.11 (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 $w(x[1]) \leq \cdots \leq w(x[n])$. 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 6.2, který je jen specifickou instancí Algoritmu 6.11.

`Poznámka`: Požadavek, že hladový Algoritmus 6.11 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.)

6.4 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 6.13. *Nechť X je nosná množina se systémem "nezávislých" podmnožin $\mathcal{N}$ splňující podmínky (1,2) Definice 6.7. Pokud pro jakoukoliv váhovou funkci $w : X \to \mathbb{R}$ najde Algoritmus 6.11 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. □

Nakonec uvádíme dvě ukázky dobře známých kombinatorických úloh (viz Lekce 7), ve kterých hladový algoritmus výrazně selže:

Příklad 6.14. *Při následujících dvou použitích paradigmatu hladového algoritmu dojde k situacím, kdy získaný výsledek může být zcela jiný než optimální odpověď.*

Obarvení grafu. Problém obarvení grafu žádá přiřazení co nejméně barev vrcholům tak, aby sousední dvojice dostaly různé barvy. Jak jsem již poznamenali, v Problému 6.5 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. Viz. Příklad 6.18. □

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

(6.4.1) *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.

Rozšiřující studium

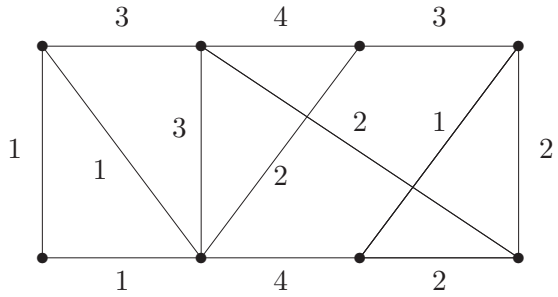
Problém minimální kostry a jeho historie jsou výborně shrnuty v [5, Oddíly 4.3,4.5]. Dosti obsáhlý rozbor algoritmů používaných na problém minimální kostry je v [4, Kapitoly 5,6]. Hladový postup je běžně zmiňován a používán v knihách zabývajících se algoritmy a optimalizací. Pro konkrétní algoritmické implementace odkazujeme i na [3].

V oblasti teorie matroidů je jen poskrovnu českých textů, ale obsáhlou a základní anglickou monografií je [8]. Krátký čtivý úvod do matroidů je volně dostupný na J. Oxley, What is a Matroid?, `http://www.math.lsu.edu/~preprint/2002/jgo2002e.pdf`.

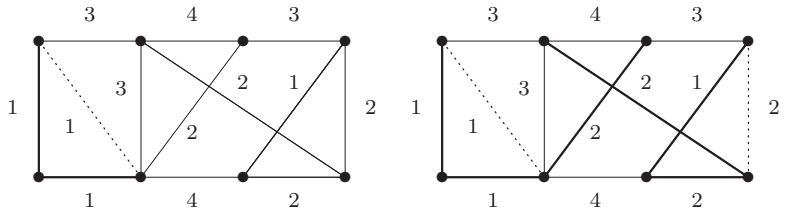
6.5 Cvičení: Ukázky použití hladových algoritmů

V následujícím cvičení se zaměříme primárně na několik snadných a názorných ukázek použití hladových algoritmů pro řešení kombinatorických problémů.

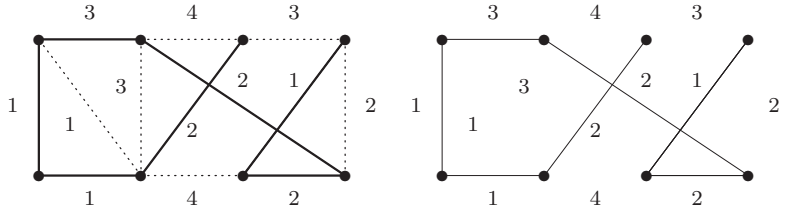
Příklad 6.15. *Najděme hladovým algoritmem minimální kostru v tomto váženém grafu (váhy hran jsou zapsány čísly v obrázku).*



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.

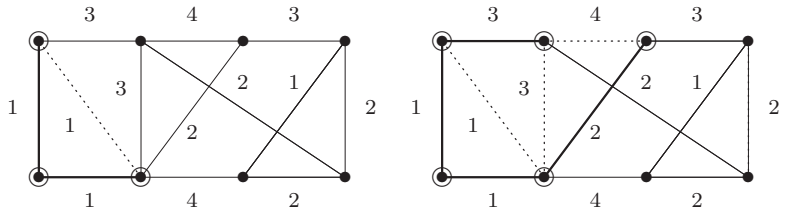


Získáme tak minimální kostru velikosti $1+2+2+3+1+1+2=12$, která je v tomto případě (náhodou) cestou, na posledním obrázku vpravo.

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.) □

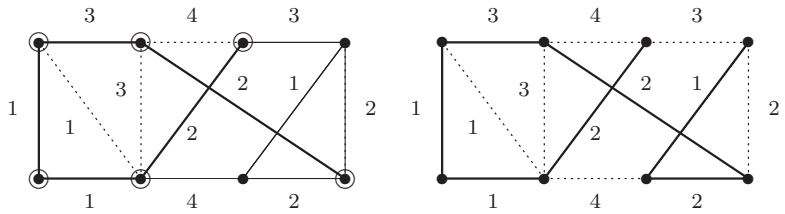
Příklad 6.16. *Najděme minimální kostru grafu z Příkladu 6.15 Jarníkovým algoritmem.*

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 6.15. 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 6.17. *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.

Vidíte přímou souvislost s Problémem 6.5? □

Příklad 6.18. *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):*

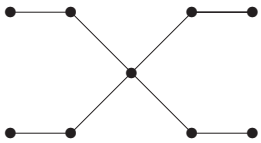
– *Začneme s prázdnou $X = \emptyset$.*

– *Vybereme vrchol v největšího stupně v G a přidáme jej do X.*

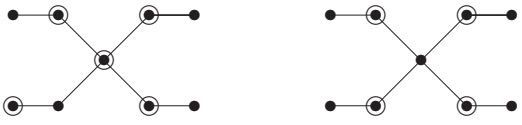
– *Vrchol v odebereme z grafu G (i s jeho hranami) a jdeme zpět na předchozí bod, dokud nezbude graf G prázdný.*

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

(6.5.1) *V jakém (jednoduše spočitatelném) pořadí hladově barvit vrcholy bipartitního grafu, aby na celý graf vystačily pouze 2 barvy?*

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

7 Barevnost a další těžké problémy

Úvod

Pro motivaci této lekce se podíváme hlouběji do historie počátků grafů v matematice. 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, jak to přesně dokázat, jedná se totiž o nebývale obtížný problém...

Takto se zde (a dále v Oddíle 8.4) přirozeně dostáváme k široce studované problematice barvení grafů. 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. A také na několik jiných typických představitelů „obtížných" grafových problémů.

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. Studující bez zájmu o informatické aspekty mohou látku týkající se složitosti jednoduše přeskočit.

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

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

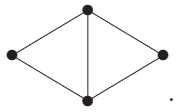
takové, že každé dva vrcholy spojené hranou dostanou různé barvy, tj. $c(u) \neq c(v)$ pro všechny $\{u, v\} \in E(G)$.

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 víc 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 jednoduché poznatky.

Lema 7.3. *Mějme jednoduchý graf (bez smyček) G a jeho libovolný podgraf $H \subseteq G$. Pak $\chi(G) \leq \chi(H)$.*

Důkaz: Zřejmě postačí použít restrikci obarvení grafu G na vrcholy podgrafu H. □

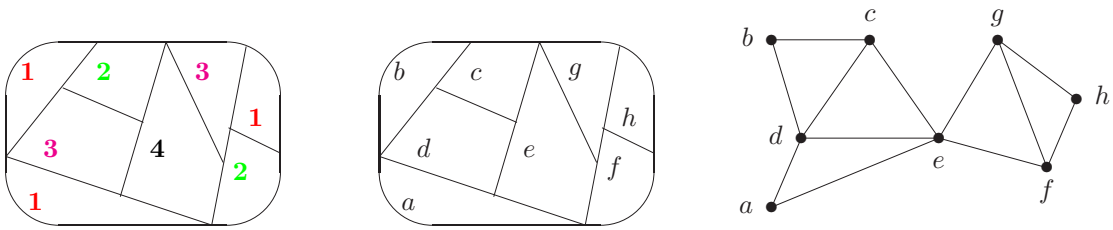
Říkáme, že barevnost grafu je *monotónní na podgrafy*.

Lema 7.4. *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 $\chi(G) < n$. □

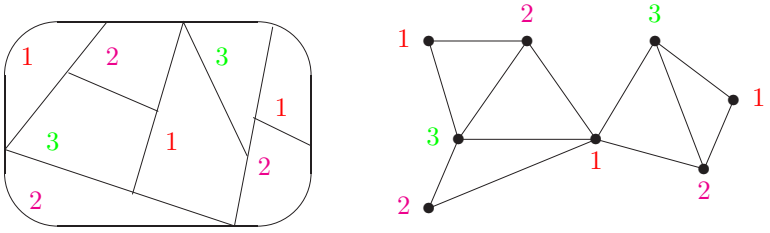
Příklad 7.5. *Vraťme se k příkladu „barvení“ mapy z úvodu lekce a ukažme si, jak mapy souvisejí s grafy a jejich barevností.*

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 7.6. *Neprázdný graf G má barevnost 1 právě když nemá žádné hrany. G má barevnost ≤ 2 právě když nemá žádnou kružnici liché délky jako podgraf.*

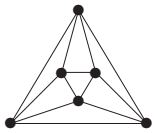
Důkaz: Pokud graf nemá hrany, můžeme všechny vrcholy obarvit stejnou barvou 1. Naopak pokud mají všechny vrcholy stejnou barvu, nemůže graf mít žádnou hranu.

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.

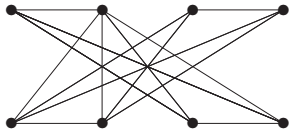
Pokud bychom tak získali třeba dva vrcholy spojené hranou f v sudé vzdálenosti od v, získáme uzavřený sled S liché délky přes f a v. Stejně tak pro dva vrcholy v liché vzdálenosti. Ponecháme-li ze sledu S ty hrany, které se opakují lichý počet krát, dostaneme Eulerovský podgraf T lichého počtu hran. Jak již víme (Oddíl 5.1), 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. □

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?*

7.2 Jak obarvit graf

Poněkud nepříjemnou skutečností je, že o barvení a určování barevnosti grafů nemůžeme v obecnosti přesně říci o mnoho více, než jsme uvedli výše v Oddíle 7.1. 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 6.2).

`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 7.7. *Každý k-degenerovaný graf lze správně hladově 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 (*Brooksovu větu*) si uvedeme nyní:

Věta 7.8. *Nechť G je souvislý jednoduchý graf maximálního stupně $k \geq 2$. Pak $\chi(G) \leq k$ až na případy, kdy G je úplný graf nebo lichá kružnice.*

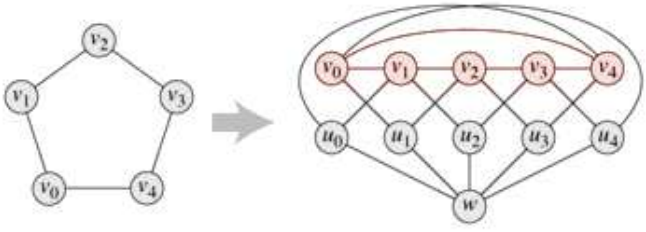
`Důkaz` (náznak): Pro $k = 2$ plyne tvrzení z Věty 7.6. Nechť tedy $k \geq 3$. V jednom směru je jasné, že $\chi(K_{k+1}) = k+1$. Naopak tedy předpokládejme, že G není úplný graf. Zároveň se omezme jen na případ, že G má všechny stupně rovné k, neboť jinak lze aplikovat postup z Věty 7.7.

- Prvním krokem nahlédneme, že pak G obsahuje dva nespojené vrcholy u, v se společným sousedem w. Pokud ale je graf $G - \{u, v\}$ nesouvislý, pak graf příslušně rozdělíme a indukcí po částech obarvíme.
- Přidejme tedy předpoklad, že $G - \{u, v\}$ je souvislý. Druhým krokem nahlédneme, že graf H vzniklý z $G - w$ ztotožněním u s v do jednoho vrcholu je $(k-1)$-degenerovaný.
- Tudíž graf H hladově obarvíme k barvami podle Věty 7.7. Po opětovném „rozpojení“ vrcholů u, v získáme obarvení $G - w$ k barvami takové, že u, v mají stejnou barvu. Nyní w má v sousedství nejvýše $k-1$ barev a G celý obarvíme. □

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. Třeba známá Mycielského konstrukce nám dává toto:

Tvrzení 7.9. *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 překvapivě silné tvrzení nalezené Erdősem:

Věta 7.10. *Pro každá $c, r > 0$ existuje graf s barevností alespoň c a neobsahující kružnice kratší než r.*

`Důkaz` (náznak): Třebaže k tomuto tvrzení dnes existují konstruktivní důkazy, stále nejkratším a z jistého pohledu nejsilnějším důkazem je původní pravděpodobnostní (nekonstruktivní) argument Erdőse:

- Ve vhodném pravděpodobnostním modelu sestrojíme náhodný graf H.
- Spočítáme, že s velkou pravděpodobností H nemá velké nezávislé množiny.
- Spočítáme, že střední hodnota počtu krátkých kružnic v H je velmi nízká.
- Proto existuje volba H, ve kterém je málo krátkých kružnic a malá hodnota nezávislosti. Tudíž z tohoto H lze odstraněním několika vrcholů všechny krátké kružnice zrušit a díky malé nezávislosti zůstane jeho barevnost vysoká.

□

Úlohy k řešení

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

(7.2.2) *Kolik nejméně barev vždy stačí na korektní obarvení grafu vzniklého z kružnice délky* 2006 *přidáním jedné nové hrany? A co přidáním dvou nových hran? Dokažte.*

(7.2.3) *Nechť dva (pod)grafy G a H sdílejí trojúhelník. Dokažte, že pokud jsou G i H 4-obarvitelné, tak i jejich sjednocení $G \cup H$ je 4-obarvitelné.*

*(7.2.4)** *Dokažte podrobně první krok důkazu Věty 7.8, když je graf $G - \{u, v\}$ nesouvislý.*

*(7.2.5)** *Dokažte podrobně druhý krok důkazu Věty 7.8, proč je graf H $(k-1)$-degenerovaný.*

(7.2.6) *Dokažte si Tvrzení 7.9.*

7.3 Variace na barevnost a jiné

Na problematiku barvení se lze dívat i z jiných úhlů, než ukázala základní Definice 7.1. Některé alternativní pohledy si nyní zběžně shrneme.

Definice 7.11. ***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 velmi dobře aproximovat za použití této Vizingovy věty.

Věta 7.12. *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 7.13. ***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 7.14. *Pro každé k nalezneme bipartitní graf s výběrovou barevností větší než k.*

`Fakt`: Hranová výběrová barevnost úplných bipartitních grafů úzce souvisí se známým problémem tzv. “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 souvisel s řešením problému čtyř barev. To je však mimo rámec našeho textu. Místo toho si ukážeme následující krásný výsledek Diraca:

Věta 7.15. *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. □

Úlohy k řešení

(7.3.1) *Jaká je hranová barevnost liché kružnice a proč?*

*(7.3.2)** *Jaká je hranová barevnost úplných grafů K_n v závislosti na n? Dokažte.*

(7.3.3) *Dokažte, že graf K_6 s jednou hranou podrozdělenou novým vrcholem má hranovou barevnost 6.*

(7.3.4) *Lze znění Věty 7.7 beze změny aplikovat na výběrovou barevnost?*

(7.3.5) *Určete výběrovou barevnost grafu $K_{2,4}$.*

*(7.3.6)** *Dokažte Tvrzení 7.14.*

*(7.3.7)** *Jaká je souvislost mezi existencí Hamiltonovské kružnice a hranovou barevností 3-regulárních grafů?*

(7.3.8) *Nalezněte 3-regulární jednoduchý souvislý graf bez Hamiltonovské kružnice.*

7.4 $\mathcal{NP}$-úplnost grafových problémů

Zatím prakticky všechny grafové problémy probírané v minulých lekcích – s drobnou výjimkou isomorfismu – byly řešitelné rozumně rychlými algoritmy. V této lekci (a více v Oddíle 7.5) se situace radikálně obrací a našim primárním zájmem se stávají obtížně řešitelné problémy. Studující ve stručnosti odkazujeme na formální definici $\mathcal{NP}$-úplnosti z oblasti algoritmické složitosti.

`Komentář`: Definice složitostní třídy $\mathcal{NP}$ se týká výhradně rozhodovacích problémů (s odpovědí „ANO/NE“). 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. Od jednoho $\mathcal{NP}$-úplného problému A se dostaneme k jinému B tzv. *polynomiálním převodem*: Ukážeme, jak bychom ze známého postupu řešení B efektivně nalezli řešení libovolné instance A.

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 7.16. ***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)?

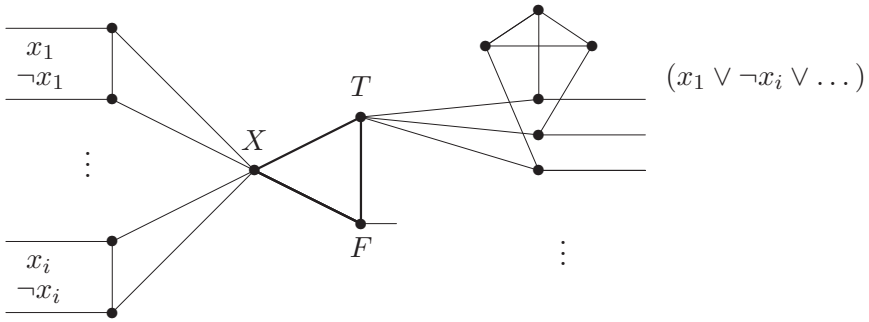
`Komentář:` Příkladem formule problému 3-SAT je třeba $\Phi \equiv (x_1 \vee \neg x_3 \vee x_4) \wedge (x_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_2 \vee x_3)$.

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

Problém 7.17. ***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.

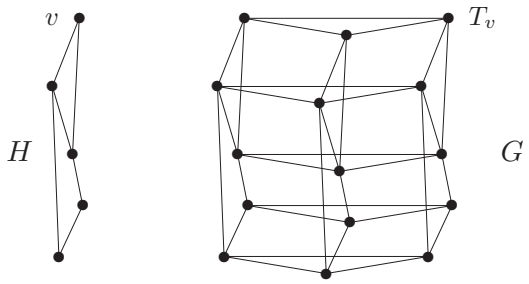
Pak G má 3-obarvení právě když je Φ splnitelná, jak si lze ověřit na obrázku. Tím jsme sestrojili polynomiální převod formule Φ z problému 3-SAT na graf G v problému 3-obarvení. $\mathcal{NP}$-úplnost nyní vyplývá z 7.16. □

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

Problém 7.18. ***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 7.19. ***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 7.27. □

Problém 7.20. ***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 7.28. □

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

Problém 7.21. ***HC (Hamiltonovský cyklus)***
Následující problém je $\mathcal{NP}$-úplný:

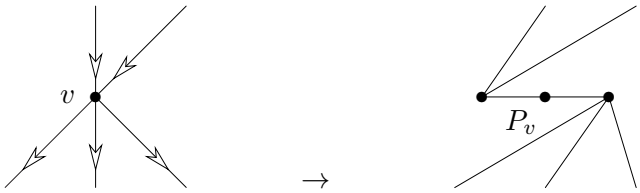
Vstup: Orientovaný graf G.
Výstup: Lze v G najít orientovanou kružnici (cyklus) procházející všemi vrcholy?

Tento převod pro jeho technickou obtížnost vynecháváme..

Problém 7.22. ***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 7.23. ***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

(7.4.1) *Rozhodněte, které z následujících problémů patří do třídy $\mathcal{P}$ všech polynomiálně řešitelných problémů. (Předpokládejte při tom $\mathcal{NP} \neq \mathcal{P}$.)*
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.

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

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

*(7.4.4)** *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?*

*(7.4.5)** *Rozhodněte, které z následujících problémů patří do třídy $\mathcal{NP}$. (Předpokládejte při tom, že negace $\mathcal{NP}$-úplných problémů už nepatří 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.

(7.4.6) *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.

(7.4.7) *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.

(7.4.8) *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 jediným 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.

7.5 Příběh problému vrcholového pokrytí

`Komentář:` Ač se to nezdá, někdy i zcela okrajová a poněkud banální otázka může nakonec vést k dalekosáhlým závěrům a novým teoriím... Co třeba proč zdánlivě velmi „podobné" problémy jako vrcholové pokrytí VC a dominující množina DOM mají (přestože oba $\mathcal{NP}$-úplné) tak rozdílné algoritmické chování? Vysvětlení [R. Downey and M. Fellows, Parameterized complexity, Springer 1999] by nás dovedlo až ke zcela novému pohledu na výpočetní složitost problémů, který jde „jaksi mimo" klasickou polynomiální hierarchii a umožňuje v jistých situacích 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, třeba opět $k = 10, 20$, dává skvěle použitelný lineární algoritmus!

Algoritmus 7.24. ***k-VC (vrcholové pokrytí)***
Pro fixní k „poměrně rychle" 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 nová množina hran F' vzniká 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 rekurze, takže ve výsledku bude čas výpočtu asymptoticky 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)

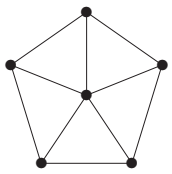
Rozšiřující studium

Problematika rovinného kreslení grafů a jejich barvení je úvodně pokryta v [5, Kapitola 6]. Další informace lze nalézt na web, třeba `http://en.wikipedia.org/wiki/Graph_coloring`*, a jako “rozcestník” pro další studium problematiky barvení grafů dobře poslouží monografie Jensena a Tofta na stránce* `http://www.imada.sdu.dk/Research/Graphcol/`*.*

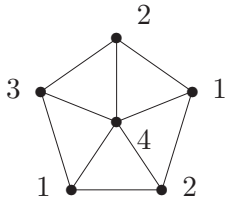
Více o $\mathcal{NP}$-úplnosti vybraných grafových problémů je uvedeno ještě v následujícím Apendixu. Pro případné hlubší studium složitosti a $\mathcal{NP}$-úplnosti grafových problémů odkazujeme na české [3], mnohé anglické monografie teoretické informatiky nebo na další internetové zdroje jako `http://en.wikipedia.org/wiki/NP-complete`*. Co se týče specificky parametrizované složitosti (lehce uvedené v Oddíle 7.5), lze mimo uvedenou již klasickou monografii [Downey and Fellows, Parameterized complexity, Springer 1999] odkázat na volně dostupnou práci R. Niedermeiera “Invitation to Fixed-Parameter Algorithms” na* `http://www-fs.informatik.uni-tuebingen.de/~niedermr/publications/habil.ps.gz` *nebo na novější monografii [Fluma–Groheho].*

7.6 Cvičení: Příklady o barevnosti grafů

Příklad 7.25. *Určete a zdůvodněte barevnost tohoto grafu:*



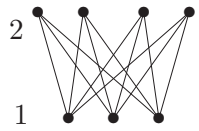
Vidíme, že obvodová kružnice tohoto grafu má délku 5, což je liché číslo, a proto je potřeba na její korektní obarvení použít aspoň tři barvy $1, 2, 3$. Pak je ale středový vrchol spojený s každou z těchto barev $1, 2, 3$, tudíž pro něj potřebujeme čtvrtou barvu 4.



To znamená, že daný graf má barevnost 4. □

Příklad 7.26. *Kolik nejméně hran musíte vypustit z úplného grafu na 7 vrcholech, aby se výsledný graf dal korektně obarvit dvěma barvama?*

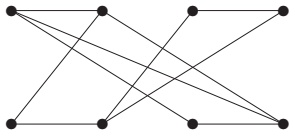
Musíme vypustit všechny hrany mezi vrcholy, kterým chceme dát stejnou barvu, aby výsledné obarvení bylo korektní. Jelikož všechny vrcholy úplného grafu jsou si ekvivalentní, stačí se rozhodnout, kolik z nich dostane barvu 1 a kolik barvu 2. Jak snadno zjistíme, optimální je barvy rozdělit napůl, tj.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



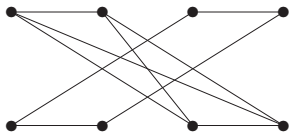
□

Úlohy k řešení

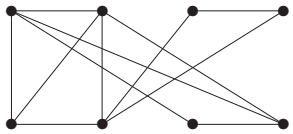
(7.6.1) *Kolik nejméně barev je třeba na korektní obarvení tohoto grafu? Příslušné obarvení vyznačte.*



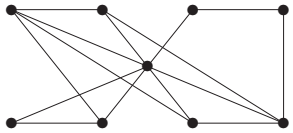
(7.6.2) *Kolik nejméně barev je třeba na korektní obarvení tohoto grafu? Příslušné obarvení vyznačte.*



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



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



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

*(7.6.6)** *Dokažte, že na obarvení grafu vzniklého z libovolného stromu přidáním dvou libovolných nových hran vždy stačí* 3 *barvy.*

(7.6.7) *Má graf z Příkladu 7.25 Hamiltonovskou kružnici?*

(7.6.8) *Pro která* m, n *má úplný bipartitní graf* $K_{m,n}$ *Hamiltonovskou kružnici?*

(7.6.9) *Nakreslete graf pravidelného dvanáctistěnu a vyznačte v něm Hamiltonovskou kružnici (to je původní Hamiltonův hlavolam).*

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

V tomto (nepovinném) obsáhlém dodatku si jednak probereme další 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. Jelikož se jedná o látku značně vzdálenou klasické teorii grafů, čtenáři bez zájmu o informatiku ji mohou klidně přeskočit. Motivací k zařazení této látky do učebnice je naše přesvědčení, že náhled na výpočetní složitost uvedených problémů umožní čtenáři mnohem lépe pochopit i jejich čistě matematické vlastnosti.

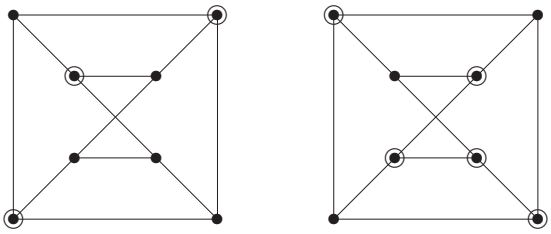
Polynomiální převody grafových problémů

`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 7.27. *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.)

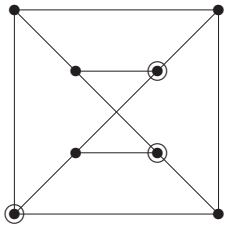


nezávislá množina vrcholové pokrytí

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.) □

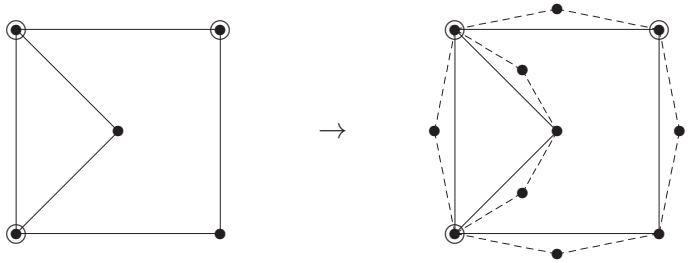
Příklad 7.28. *Problém dominující množiny se ptá, zda v grafu existuje podmnožina m vrcholů taková, že každý jiný vrchol je s některým z nich spojený hranou. Ukažme si podrobně, jak se problém vrcholového pokrytí polynomiálně převede na problém dominující množiny na souvislých grafech.*

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. □

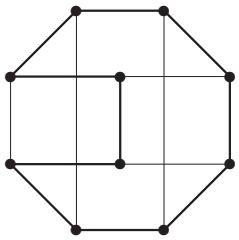
Problémy ve třídě $\mathcal{NP}$

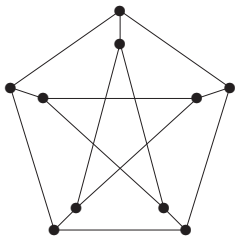
Nyní se budeme věnovat otázkám, proč některé grafové problémy náleží či nenáleží do třídy $\mathcal{NP}$.

`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 7.29. *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 7.30. *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.

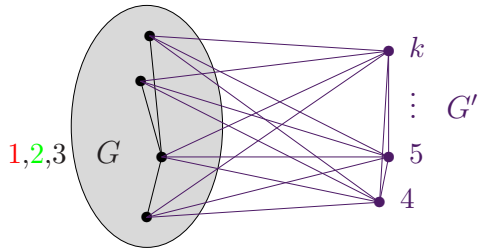
Takže v jednoduchých případech stačí vzít obecný (libovolný) vstup $\mathcal{V}$ známého těžkého problému P a "přeložit" jej na vstup $\mathcal{U}$ pro problém Q tak, že Q odpoví ANO na nový vstup $\mathcal{U}$ právě tehdy, když P odpověděl ANO na $\mathcal{V}$. (Všimněte si dobře, že musíte dokázat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

Příklad 7.31. *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 7.17, 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 7.32. *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.*

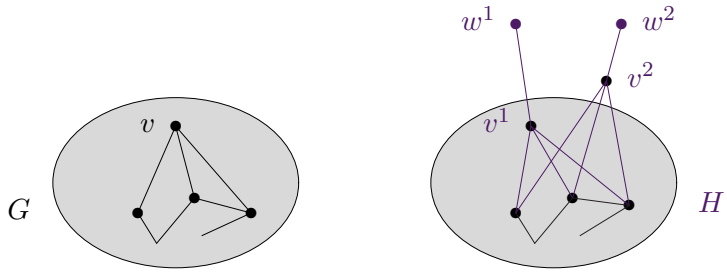
Již víme z Problému 7.22, zjištění existence Hamiltonovské kružnice je $\mathcal{NP}$-úplné. Problém Hamiltonovské cesty taktéž náleží do $\mathcal{NP}$, snadnou nápovědou existence je

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 7.33. *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 proměnné, 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 7.18. □

Úlohy k řešení

(7.7.1) *Analogicky k Příkladu 7.27 ukažte převod problému vrcholového pokrytí na nezávislou množinu. (Tj. opačný směr.)*

(7.7.2) *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?*

*(7.7.3)** *Dokážete najít převod problému dominující množiny na vrcholové pokrytí? (Tj. opačný směr k Příkladu 7.28.)*

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

(7.7.5) *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 Příkladě 7.32).*

8 Rovinnost a kreslení grafů

Úvod

V přímé návaznosti na předchozí Lekci 7 se zaměříme na druhý důležitý aspekt slavného problému čtyř barev; na otázku kreslení grafů do roviny bez křížení. Jak tedy taková obarvovaná politická mapa souvisí s kreslením grafů? Jednoduše – souvislé státy můžeme reprezentovat jako vrcholy grafu a hranami pak zaznamenat „sousednost" mezi státy. Důležité je, že takto vzniklý „duální" graf můžeme zřejmě zakreslit v rovině bez křížení hran a nazýváme jej proto rovinným grafem. Právě tato rovinnost je klíčová pro zmíněnou obarvitelnost grafu i pro jiné zajímavé vlastnosti rovinných grafů.

V našem výkladu si uvedeme (a zčásti odvodíme) mnohé základní vlastnosti rovinných grafů a sdělíme něco málo o kreslení grafů obecněji.

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 řekneme o obarvování rovinných grafů a v závěru uvedeme poznatky o praktickém kreslení i nerovinných grafů.

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.

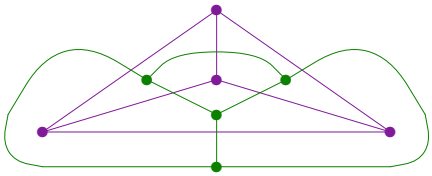
Komentář:

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 7.5 – 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:

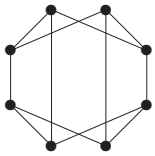
Definice. *Duální (multi)graf* rovinného nakreslení grafu G získáme tak, že stěny nahradíme vrcholy duálu a hranami spojíme sousedící dvojice stěn.

Komentář: Duální multigraf k rovinnému grafu je vždy rovinný, což je relativně snadné dokázat topologicky. Na druhou stranu však často bude obsahovat násobné hrany a dokonce i smyčky. Uvědomte si dobře, že počet hran duálního grafu je stejný jako u primárního grafu. Pro příklad odvození duálního grafu se podívejme na obrázek:



Úlohy k řešení

(8.1.1) *Nakreslete rovinně tento graf:*



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

(8.1.3) *Dokažte, že graf každého 3D konvexního mnohostěnu je rovinný.*

*(8.1.4)** *Dokažte, že graf každého 3D konvexního mnohostěnu je 3-souvislý.*

8.2 Eulerův vztah o počtu stěn

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*

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

Komentář: Čtenář by si měl povšimnout, že následující (zdánlivě jednoduchý) důkaz Eulerova vztahu závisí na docela obtížné a hluboké Jordanově větě o kružnici. To není náhoda, neboť jistým způsobem je Eulerův vztah Jordanově větě ekvivalentní – viz graf kružnice – a obě tvrzení tak jsou matematicky hluboká...

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 (viz Jordanova věta o kružnici) a dle Věty 5.3 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 $v + f - h = v + (f - 1) - (h - 1) = 2$.

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.

Eulerův 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 $h \geq \frac{1}{2} \cdot 3f$, neboli $\frac{2}{3}h \geq f$. Dosazením do vztahu Věty 8.2 získáme

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

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

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 $h \geq \frac{1}{2} \cdot 4f$, neboli $\frac{2}{4}h \geq f$. Dosazením do vztahu Věty 8.2 získáme

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

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

Tím jsme hotovi. □

`Poznámka`: Kdy nastává rovnost v Důsledku 8.3? Snadno analýzou důkazu zjistíme, že jednoduchý rovinný graf má $3v - 6$ hran, právě když všechny jeho stěny včetně vnější jsou trojúhelníky. Obdobně pro druhou část tvrzení se všemi stěnami velikosti 4.

Důsledek 8.4. *Každý jednoduchý rovinný graf obsahuje vrchol stupně nejvýše 5. Každý jednoduchý 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

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

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

*(8.2.3)** *Jak bude znít Eulerův vztah pro nesouvislý rovinný graf s k komponentami?*

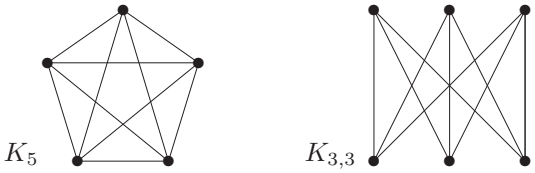
8.3 Rozpoznání rovinných grafů

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 ukázat v omezeném čase 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.

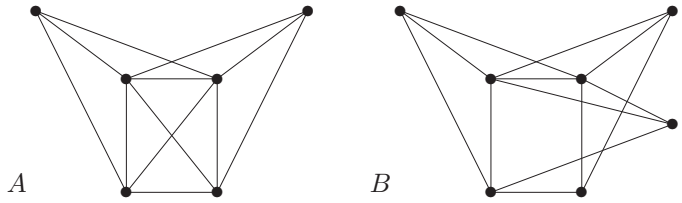
`Definice`: *Podrozdělením* grafu G rozumíme graf, který vznikne z G nahrazením některých hran novými cestami libovolné (kladné) délky.

Důležitý abstraktní popis všech rovinných grafů nalezl K.Kuratowski:

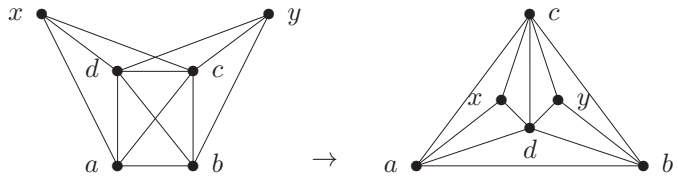
Věta 8.8. *Graf G je rovinný právě když neobsahuje podrozdělení grafů K_5 nebo $K_{3,3}$ jako podgrafy.*

`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.

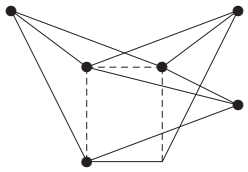
Příklad 8.9. *Které z následujících dvou grafů jsou rovinné? Najděte rovinné nakreslení (včetně očíslovaných vrcholů), nebo zdůvodněte nerovinnost grafu.*



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 poznatek 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.*

`Důkaz`: V jednom směru, pokud $G' = G - V(C)$ je souvislý, pak podle Jordanovy věty o kružnici leží celý G' uvnitř jedné oblasti C, tudíž druhá oblast C je stěnou v každém nakreslení G.

Naopak pokud C ohraničuje stěnu v některém rovinném nakreslení grafu G, dokážeme, že každé dva vrcholy mimo C lze spojit cestou disjunktní s C. Nechť tedy $x, y \in V(G) \setminus V(C)$ a označme X (či Y) množinu těch vrcholů C dosažitelných z x (z y) po cestách neprocházejících přes C. Jelikož x, y náleží stejné stěně kružnice C, množiny X, Y se na C „nepřekrývají“, přesněji je lze od sebe oddělit odebráním některých dvou vrcholů $c, d \in V(C)$. Pak však $\{c, d\}$ je řezem v grafu G oddělujícím x od y a to odporuje předpokladu 3-souvislosti, spor. □

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

Zajímavou aplikací Důsledku 8.11 je algoritmus pro rozpoznávání isomorfismu rovinných grafů, který mimo porovnávání rovinných nakreslení 3-souvislých komponent používá metody rozkladu grafu na "více-souvislé" komponenty a testování isomorfismu stromů:

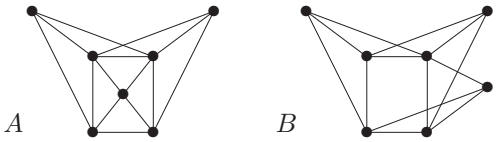
Věta 8.12. *Problém isomorfismu rovinných grafů je řešitelný v lineárním čase.*

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

Věta 8.13. *Každý jednoduchý rovinný graf lze nakreslit v rovině (bez křížení hran) tak, že hrany jsou úsečky.*

Úlohy k řešení

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



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

(8.3.3) *Kdy je úplný bipartitní graf $K_{m,n}$ rovinný?*

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

*(8.3.5)** *Jak vypadají grafy bez podrozdělení $K_{3,3}$?*

8.4 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 101). 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 znovu v roce 1993 Robertson, Seymour, Sanders a Thomas, dokázali tuto větu, která rozřešila problém čtyř barev a která je jedním z nejslavnějších výsledků diskrétní matematiky vůbec:

Věta 8.14. *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 počítač), a proto si uvedeme slabší a mnohem jednodušší tvrzení:

Tvrzení 8.15. *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 7.7. Druhou část dokážeme obdobně, když nalezneme vrchol stupně ≤ 3. □

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

Věta 8.16. *Každý rovinný graf bez smyček má výběrovou barevnost nejvýše* 5.

`Důkaz` (náznak): Poměrně přímočarou indukcí lze dokázat následující zesílené tvrzení:
Nechť rovinný graf G s vnější stěnou ohraničenou kružnicí C má všechny ostatní stěny trojúhelníky. Nechť každý vrchol mimo C má přiřazen seznam 5 *barev, vrcholy C mají seznamy* 3 *barev a jisté dva sousední vrcholy x, y na C mají přímo předepsané (různé) barvy. Pak G lze výběrově obarvit.*

- Nechť $z \neq y$ je druhý soused x na C. Pokud některá hrana f ze z nenáležející C má druhý konec také na C, pak podél f „rozdělíme“ G na podgraf G_1 obsahující x, y a podgraf G_2 sdílející hranu f s G_1. Indukcí nejprve obarvíme G_1, pak G_2 taktéž splní indukční předpoklad a i jej dobarvíme.
- Jinak budeme indukcí barvit podgraf $G_3 = G - z$; přičemž všem sousedům z uvnitř G odebereme ze seznamu (jejich pěti) barev dvě z barev seznamu u vrcholu z různé od barvy x. Následně dobarvíme vrchol z, pro nějž máme tři možnosti a jen jeho dva sousedé na C s ním mohou být v konfliktu.

□

Úlohy k řešení

(8.4.1) *Jaká je barevnost grafu pravidelného dvacetistěnu?*

*(8.4.2)** *Nechť rovinný graf má každou stěnu délky* 4. *Jaká je jeho barevnost? Dokažte.*

(8.4.3) *Předpokládejme, že každý vrchol rovinného grafu leží na vnější stěně. Dokažte, že pak je jeho barevnost nejvýše* 3.

*(8.4.4)** *Dokážete najít rovinný graf, jehož výběrová barevnost je* 5*?*

(8.4.5) *Proč každý rovinný graf na* 13 *vrcholech musí obsahovat nezávislou množinu velikosti* 4*?*

8.5 Praktické „pružinové“ kreslení grafů

Závěrem se podívejme na trochu jinou problematiku – jak prakticky nakreslit daný (nerovinný) graf, aby vše „vypadalo hezky“. Jeden ze základních heuristických přístupů ke kreslení grafů se dá shrnout následovně:

Metoda 8.17. ***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.

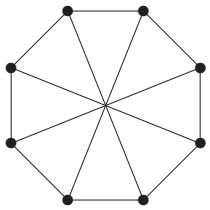
Rozšiřující studium

Problematika rovinného kreslení grafů a jejich barvení je úvodně pokryta v [5, Kapitola 6]. Zájemce o přístupný český popis (poněkud pomalejšího, kvadratického) algoritmu na rovinné kreslení odkazujeme na [3]. Jeden z existujících lineárních algoritmů na rovinné kreslení je vysvětlen třeba na `http://www.math.gatech.edu/~thomas/planarity.ps`. *Zábavné hraní si s rovinností grafů je pak k dispozici na* `http://www.planarity.net/`.

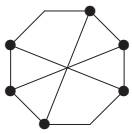
Zájemci o podrobný popis historie i náznaku řešení problému čtyř barev si mohou přečíst přímo `http://www.math.gatech.edu/~thomas/FC/fourcolor.html`. *K pružinovému modelu kreslení grafů ("spring layout") existuje na webu mnoho popisů i dostupných implementací, i když ne vždy kvalitních, dokonce je najdeme i mezi demonstračními aplety Javy na* `http://java.sun.com/applets/jdk/1.4/`.

8.6 Cvičení: Příklady na rovinnost grafů

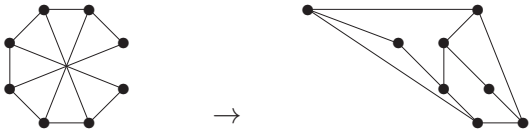
Příklad 8.18. *Kolik nejméně hran je třeba vypustit z následujícího* 8*-vrcholového grafu, aby vznikl rovinný graf? Zdůvodněte.*



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



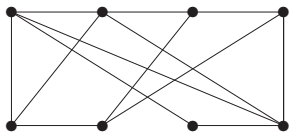
Tento obrázek zároveň ukazuje ještě jednu zajímavost – náš graf nebude rovinný, ani když vypustíme libovolnou z "tětiv" obvodové kružnice. Na druhou stranu není těžké objevit, že stačí vypustit třeba pravou svislou hranu a získáme rovinné nakreslení:



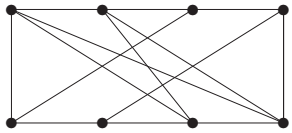
Stačí tedy vypustit jednu hranu a získáme rovinný graf. □

Úlohy k řešení

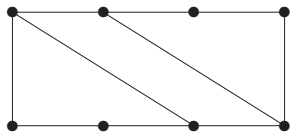
(8.6.1) *Je tento graf rovinný?*



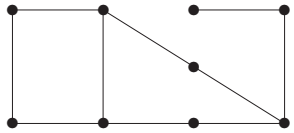
(8.6.2) *Je tento graf rovinný?*



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



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



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

(8.6.6) *Kolik nejméně hran je nutno odebrat z úplného grafu* K_7*, aby se stal rovinným?*

(8.6.7) *Dokažte, že rovinný graf, jehož každá stěna je trojúhelník, musí být* 3*-souvislý.*

(8.6.8) *Je možné, aby rovinný* 2*-souvislý graf na* 1000 *vrcholech měl více než* 1000 *neekvivalentních nakreslení (s jinými soubory stěnových kružnic)?*

Část IV
Vybrané Pokročilé Partie

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

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

9 Povídání o průnikových grafech

Úvod

Naším prvním výběrem pokročilého tématu jsou průnikové grafy. Jedná se o 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 (třeba intervalové nebo chordální grafy). Jedná se však také o oblast celkem zajímavou i z pohledu historie a vývoje teorie grafů.

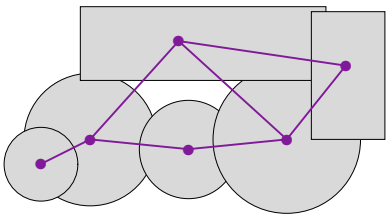
Cíle

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

9.1 Průnikové a intervalové grafy

Oblast průnikových grafů je obsáhle studovaná a má mnoho zajímavých zákoutí a pěkných výsledků, jak v oblasti teorie, tak i po algoritmické stránce. My si několik takových v této lekci ukážeme. Nejprve uvedeme základní všeobecnou definici a poznatky o průnikových reprezentacích grafů.

Definice 9.1. ***Průnikovým grafem*** množinového systému $\mathcal{M}$ nazveme graf $I_{\mathcal{M}}$ na vrcholech $V = \mathcal{M}$ s množinou hran $E = \big\{\{A, B\} \subset \mathcal{M} : A \cap B \neq \emptyset\big\}$.



Poznamenáváme také, že nejčastěji studované typy průnikových grafů mají „geometrickou povahu", tj. jejich množiny jsou definovány jako geometrické objekty.

`Fakt`: Třídy průnikových grafů (urč. typu) jsou vždy uzavřené na indukované podgrafy.

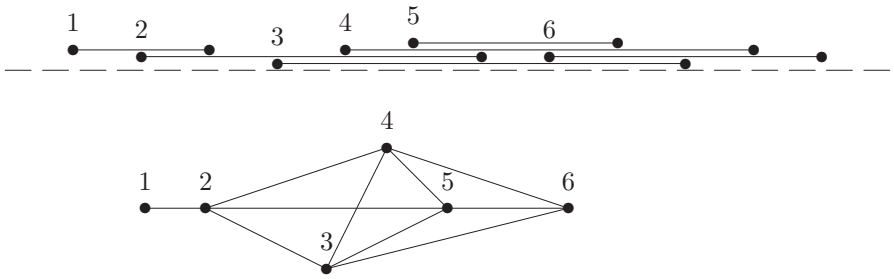
Tvrzení 9.2. *Každý jednoduchý graf je isomorfní průnikovému grafu nějakého vhodného systému množin.*

`Důkaz` (náznak): Stačí pro každý vrchol x našeho grafu zvolit soubor hran incidentních s x za množinu M_x reprezentující x. □

Intervalové grafy

Z celého spektra možných typů průnikových grafů se blíže podíváme na průnikové grafy intervalů na přímce – *intervalové grafy* (zkratka *INT*).

`Komentář:` Jedná se o jeden z nejstarších historických příkladů průnikových grafů, a přitom intervalové grafy jsou intenzivně studovány dodnes a mají zajímavé aplikace, třeba v DNA sekvencování. Zde je jednoduchý příklad intervalového grafu:



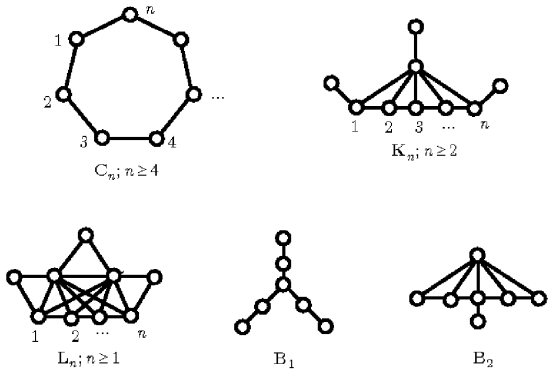
Vzpomeňte si, že s intervalovými grafy jsme se vlastně setkali už v Problému 6.5 řešícím přidělení pracovních úkolů, tj. barvení příslušného intervalového grafu.

Stručně a bez důkazů si uvedeme tato klíčová tvrzení:

Lema 9.3. *Každá kružnice délky větší než tři v intervalovém grafu má chordu, tj. indukuje hranu spojující nesousední vrcholy kružnice.*

Věta 9.4. *Třídu všech intervalových grafů lze charakterizovat pomocí jednoho z následujících tvrzení.*

- *Graf je intervalový právě když neobsahuje žádný z následujících zakázaných indukovaných podgrafů:*



- *Jednoduchý graf je intervalový, právě když neobsahuje indukovanou kružnici C_4 a jeho doplněk má tranzitivní orientaci.*

Úlohy k řešení

(9.1.1) *Dokažte si sami, že kružnice délky větší než 3 není intervalovým grafem.*

*(9.1.2) *Dokažte si sami, že další zakázané grafy z Věty 9.4 nejsou intervalové*

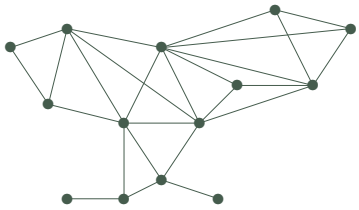
*(9.1.3) *Jak byste charakterizovali grafy, které jsou průnikovými grafy intervalů stejné délky?*

(9.1.4) *Jak byste jednoduše nahlédli, že doplněk intervalového grafu má tranzitivní orientaci?*

9.2 Chordální grafy

Chordální grafy představují zajímavou a užitečnou třídu grafů zobecňujících na jednu stranu intervalové grafy a na druhou stranu stromy. Podívejte se proto na ně později znovu také z druhého zmíněného pohledu Lekce 10.

Definice 9.5. ***Chordální graf*** (také zván *triangulovaný*) G je takový, který neobsahuje žádnou indukovanou kružnici delší než tři jako svůj podgraf.

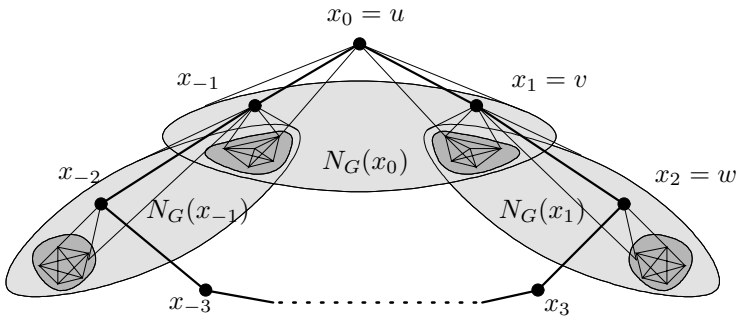


Úplně základním poznatkem o chordálních grafech je následující věta, poprvé dokázaná už v jednom z prvních článků o intervalových grafech. Dnes existují i krátké důkazy a my si zde uvedeme jeden alternativní a striktně elementární důkaz.

Věta 9.6. *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 chordálním grafu G existuje hrana f taková, že $E(C) \setminus \{e\} \cup \{f\}$ obsahuje trojúhelník.
2. Nechť uv je hranou G a $N_G(v)$, tj. sousedé v, tvoří (indukují) bisimpliciální podgraf v G. 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. Tudíž pokud G není bisimpliciální, ale sousedé každého jeho vrcholu indukují bisimpliciální podgraf, pak G obsahuje kružnici C odporující bodu 1 (na vrcholech $x_0, x_1, x_2, \ldots$ jako na obrázku níže).
4. Tudíž G je bisimpliciální.



□

Přímým důsledkem Věty 9.6 je existence *simpliciální dekompozice* libovolného chordálního grafu:

Důsledek 9.7. *Vrcholy každého chordálního grafu G lze seřadit do posloupnosti $v_1, v_2, \ldots, v_n$ tak, že každé v_i, $i = 2, \ldots, n$, je simpliciální v podgrafu G indukovaném na $\{v_1, \ldots, v_{i-1}\}$.*

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

Průniková reprezentace

Docela zajímavý je ještě jeden pohled na chordální grafy – ukazující, proč je lze považovat za přirozené zobecnění intervalových grafů.

Věta 9.8. *Graf G je chordální právě když existuje strom T takový, že G je průnikovým grafem vhodné kolekce podstromů v T.*

`Důkaz` (náznak): Ve směru doleva pouze stručně konstatujeme, že každý průnikový graf podstromů v nějakém stromě nutně musí být chordální. Naopak povedeme důkaz indukcí podle počtu vrcholů G, přičemž báze pro jeden vrchol je zřejmá.

Jinak nechť v je simpliciální vrchol chordálního grafu G. Indukcí sestrojíme průnikovou reprezentaci také chordálního grafu $G - v$. Pak sousedé v tvoří kliku, tudíž v průnikové reprezentaci grafu $G - v$ se v některém uzlu stromu všichni překrývají. Na tomto místě přidáme nový list stromu, který bude reprezentovat v a bude překryt reprezentanty sousedů v. □

Úlohy k řešení

(9.2.1) *Řekněte jednoduše, proč každý intervalový graf obsahuje aspoň dva simpliciální vrcholy?*

(9.2.2) *Jak byste snadno zdůvodnili, že daný graf je chordální? Přes definici to přímo nejde, neboť byste museli zkontrolovat až exponenciálně mnoho kružnic.*

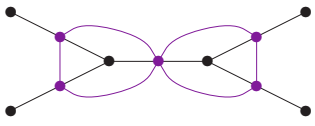
(9.2.3) *V důkaze Věty 9.8 se využívá, že pokud průniková reprezentace podstromů ve stromě tvoří kliku, tak se tyto stromy musí překrývat ve společném vrchole. Proč?*

9.3 Další třídy průnikových grafů

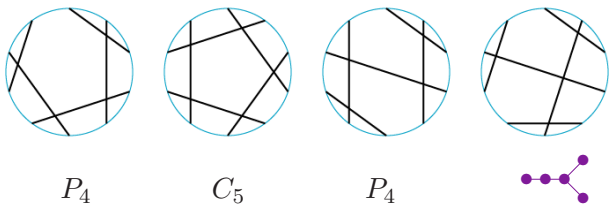
Ve výkladu pokračujeme stručným neformálním přehledem některých jiných typů geometrických průnikových grafů, které jsou běžně studovány. Náš výběr ukázek není nijak specifický, jen ilustruje bohatost a oblíbenost této výzkumné oblasti grafů.

`Definice`: Zavedeme následující třídy průnikových grafů:

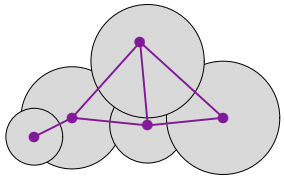
- *Hranový* graf $L(G)$ je průnikovým grafem hran $E(G)$ v běžném grafu G.



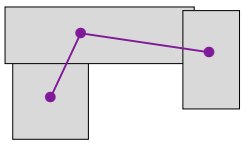
- *Kruhově-intervalové* grafy (CA) jsou průnikovými grafy intervalů na kružnici.
- *Kružnicové* grafy (CIR) jsou průnikovými grafy tětiv v kružnici.



- *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.

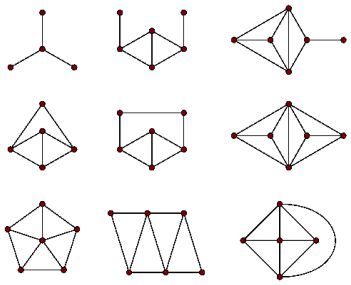


Složitost rozpoznávání

Častou a důležitou otázkou u (jakýchkoliv) typů reprezentací grafů je, které abstraktní grafy mají reprezentaci daného typu a jak lze takovou reprezentaci algoritmicky sestrojit. Problému se říká *rozpoznávání daného typu grafů*.

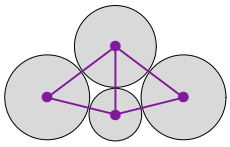
Věta 9.9. *Hranové, CA a CIR grafy lze rozpoznávat algoritmy v polynomiálním čase. Problémy, zda daný abstraktní graf je DISC nebo BOX, jsou NP-úplné.*

Věta 9.10. *Daný graf je hranovým grafem nějakého jednoduchého grafu, právě když neobsahuje žádný z následujících indukovaných podgrafů:*



Jiné druhy reprezentace

- *Dotykové* ... grafy jsou variantou průnikových grafů geometrických objektů, ve které se požaduje, aby vnitřky objektů byly po dvou disjunktní.

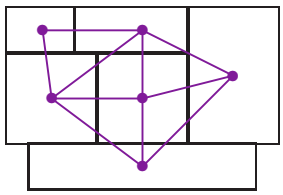


Moc hezkým výsledkem v oblasti dotykových grafů je následující, který je už mnoho let starý (Koebe) a několikrát byl nezávisle znovu objevován.

Věta 9.11. *Graf je rovinný právě když je dotykovým grafem kruhů v rovině.*

Vedle prostých dotykových grafů lze zavést ještě striktnější reprezentace, jako například „zápalkové“ grafy v příští lekci nebo následující ukázku.

- *Obdélníkové „duály“* – jedná se o průnikové dotykové reprezentace grafu pomocí nepřekrývajících se obdélníků se stranami rovnoběžnými osám.



  - V této reprezentaci není dovoleno setkání čtyř obdélníků v jednom rohu.
  - Ve striktním podání musí obdélníky reprezentace vyplnit plochu „bez děr“.

`Fakt`: Pouze rovinné grafy mohou mít obdélníkový duál, avšak ne všechny rovinné grafy je mají. Navíc striktní obdélníkové duály vždy reprezentují *kvazitriangulace* (všechny stěny až na vnější jsou trojúhelníky).

Úlohy k řešení

(9.3.1) *Najděte nějaký graf, který není CA.*

(9.3.2) *Najděte nějaký graf, který není CIR.*

(9.3.3) *Najděte nějaký graf, který není DISC.*

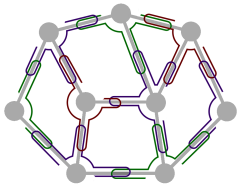
*(9.3.4)** *Najděte nějaký graf, který není průnikovým grafem koulí v 3D.*

*(9.3.5)** *Dokažte formálně, že dotykové grafy kruhů v rovině jsou vždy rovinné.*

9.4 Průnikové grafy křivek a úseček

Na závěr výkladu se trochu podrobněji podíváme na následující typy průnikových grafů. (Jejich výběr není reprezentativní, ani nemá nějaký hlubší význam, jen že jsou zajímavé a blízké autorovu dřívějšímu výzkumu v oblasti průnikových grafů.)

`Definice`: *Křivkovými (niťovými) grafy* nazveme průnikové grafy obyčejných křivek v rovině.



Tvrzení 9.12. *Každý rovinný graf je niťový.*

Následující tvrzení [Kratochvíl] je poněkud překvapivé a samo o sobě naznačuje, že studium niťových grafů nebude lehké.

Tvrzení 9.13. *Existují grafy, které jsou niťové, ale každá jejich taková reprezentace obsahuje dvojici křivek majících exponenciálně mnoho (k počtu vrcholů) vzájemných průsečíků.*

Složitost rozpoznávání

Co se týče algoritmické složitosti, je pak rozpoznání niťových grafů těžké. Vzhledem k Tvrzení 9.13 je mnohem obtížnější dokázat příslušnost problému do třídy $\mathcal{NP}$, než jeho těžkost (což je poměrně neobvyklý fenomén v teorii složitosti), viz [Kratochvíl / Schaeffer a Štefankovič].

Věta 9.14. *Problém rozpoznat, zda daný graf je niťový, je $\mathcal{NP}$-úplný.*

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

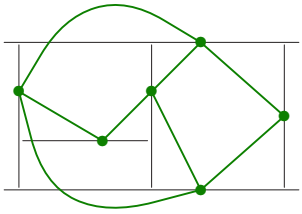
Tvrzení 9.15. *Existují grafy, které jsou úsečkové, ale každá jejich taková reprezentace obsahuje úsečku, k zápisu jejíž souřadnic je třeba exponenciálně mnoho (k počtu vrcholů) bitů.*

Ve srovnání s jednoduchým Tvrzením 9.12 vynikne tento docela dlouho otevřený problém, nedávno dořešený [Chalopin, Goncalves]:

Věta 9.16. *Každý rovinný graf je úsečkovým grafem.*

„Zápalkové“ grafy

Výše jsem zmínili obecný pojem geometrických dotykových grafů, nyní se z tohoto úhlu pohledu podíváme na *dotykové grafy úseček* v rovině: Tímto pojmem nazýváme ty průnikové grafy úseček v rovině, u nichž je dodatečnou podmínkou, že žádné dvě úsečky se neprotínají ve svých vnitřních bodech. (Jakoby „zápalkové“ reprezentace v rovině.)



Věta 9.17. *Graf je dotykovým grafem disjunktních horizontálních a disjunktních vertikálních úseček, právě když se jedná o rovinný bipartitní graf.*

V obecném případě (bez omezení na pouhé dva směry úseček) se však opět jedná o obtížně popsatelnou a výpočetně těžkou třídu grafů:

Věta 9.18. *Problém rozpoznat dotykový graf úseček je $\mathcal{NP}$-úplný.*

Úlohy k řešení

(9.4.1) *Dokažte si sami Tvrzení 9.12.*

(9.4.2) *Najděte graf, který není průnikovým grafem úseček majících pouhé tři směry.*

(9.4.3) *Najděte nějaký graf, který je křivkový, ale ne úsečkový.*

(9.4.4) *Proč každý graf na 6 vrcholech je niťový?*

(9.4.5) *Najděte jakýkoliv rovinný graf, který není dotykovým grafem úseček.*

(9.4.6) *Co myslíte, bylo by možné (viz Věta 9.16) elementárně dokázat, že každý rovinný graf je průnikem úseček majících pouhé čtyři směry?*

Rozšiřující studium

Jelikož jsou průnikové a chordální grafy obsáhle studovány, množství další informací lze snadno nalézt na internetu pod heslem "intersection graphs". Specificky je třídám průnikových grafů věnována třeba monografie [McKee, McMorris, Topics in Intersection Graph Theory, SIAM monographs on discrete mathematics and applications, 1999]. Specificky intervalové a chordální (a také CIR) grafy mají zajímavý vedlejší vztah k odlišnému tématu šířkových parametrů příští lekce.

Různé typy průnikových grafů ve své podstatě vycházejí z praktických aplikací, jako například DISC grafy z problematiky přidělování radiových frekvencí, INT grafy ve vztahu k sekvencování genomu, apod. Vhodným rozcestníkem k dalšímu samostatnému studiu je stránka `http://en.wikipedia.org/wiki/Intersection_graph`.

10 Dekompozice grafů, algoritmy a minory

Úvod

Další autorův výběr tématu nás zavede ke strukturální teorii grafů – k různým jejich dekompozicím, grafovým minorům a navazujícím efektivním algoritmům pro jinak těžké problémy. Obecnou inspirací nám je známý fakt, že většinu jinak těžkých problémů lze řešit snadno a efektivně na stromech. Podobná situace nastává třeba u intervalových grafů nebo obecně u chordálních grafů. Proto se podíváme na grafy, které jsou svým způsobem „stromům blízké", ve smyslu existence jejich vhodné dekompozice. Stromová dekompozice přitom má definičně blízko i ke dříve zmiňovaným chordálním grafům.

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. Důraz bude kladen především na stromovou šířku a její návaznost na dříve definované pojmy jako třeba chordální grafy. Závěrem je definován pojem minoru a formulována Robertson–Seymourova věta o dobrém kvaziuspořádání konečných grafů vzhledem k minorům.

10.1 Obtížné problémy na speciálních grafech

V Lekci 7 jsme uvedli některé „neřešitelně obtížné", přesněji $\mathcal{NP}$-těžké grafové problémy, například 3-obarvení grafu (Problém 7.17) nebo nezávislá množina vrcholů (Problém 7.18). Přesto i takto obtížné problémy dokážeme řešit efektivně na některých specifických třídách grafů – a to nejen na stromech.

Stačí si vzpomenout na Problém 6.5 o přidělování pracovních úkolů, neboli o barevnosti intervalových grafů v terminologii Lekce 9, který jsme řešili rychlým hladovým algoritmem. Obdobně vyřešíme i nezávislou množinu na intervalových grafech:

Algoritmus 10.1. ***Nalezení nezávislé množiny v intervalovém grafu***
Předpokládejme, že graf G je daný svou intervalovou reprezentací (v případě potřeby je možno tuto reprezentaci efektivně sestrojit). Maximální nezávislou množinu nalezneme následovně.

- Uspořádáme intervaly reprezentující G podle jejich pravých konců.
- Do nezávislé množiny hladově (v tomto uspořádání) vložíme vždy první interval neprotínající se s předchozími vybranými.

Také na obecnějších chordálních grafech můžeme navrhovat rychlé a povětšinou i snadné algoritmy.

Algoritmus 10.2. ***Určení barevnosti chordálního grafu***

- Snadno zkonstruhujeme simpliciální dekompozici našeho grafu G (Věta 9.6).
- Graf G je k-degenerovaný, kde $k = \omega(G) - 1$ je největší stupeň simpliciálního vrcholu v některém kroku simpliciální dekompozice G. Hladově tudíž můžeme G obarvit $k+1$ barvami a to je optimální (neboť máme v G kliku velikosti $k+1$).

Algoritmus 10.3. ***Nalezení nezávislé množiny chordálního grafu***

- Opět zkonstruhujeme simpliciální dekompozici našeho grafu G (Věta 9.6).

- V pořadí této dekompozice hladově přidáváme vrcholy do nezávislé množiny. (Tento postup je přímým zobecněním Algoritmu 10.1, viz také Věta 9.8.)

Možná zobecnění?

Zajímavou a užitečnou otázkou teď je, jak takové postupy zobecnit na širší třídy grafů, které mají nějakou specifickou omezující (ale ne příliš) vlastnost – „parametr“.

Úlohy k řešení

*(10.1.1) *Jak byste efektivně nalezli nejmenší dominující množinu na intervalovém grafu?*

(10.1.2) *Jak byste efektivně nalezli nejmenší vrcholové pokrytí v chrodálním grafu?*

*(10.1.3) *Dokažte správnost Algoritmu 10.3.*

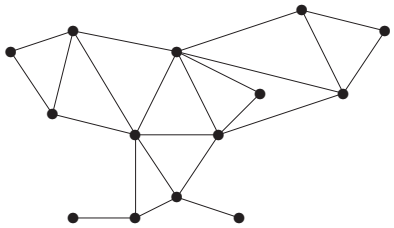
10.2 Tree-width – čtyři definice

Výlet za strukturálními parametry grafů začneme pojmem, který už lze dnes prohlásit za klasický. Název „tree-width“ byl zaveden Robersonem a Seymourem počátkem 80-tých let, ale pak se ukázalo, že ekvivalentní definice již uvažovali matematici léta před nimi, například v souvislosti s takzvanými „k-trees“ nebo se simpliciálními dekompozicemi. (Důsledkem tohoto vývoje je také bohatost různých definic stejného pojmu...)

Připomeňme, že velikost největší kliky v grafu G se označuje $\omega(G)$.

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

`Komentář:` Například každý podgraf následujícího chrodálního grafu má tree-width ≤ 2:

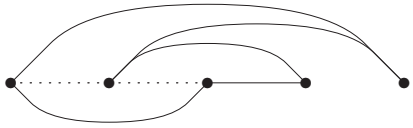


Kde je však v této definici nějaký „strom“? Je, ale skrytý – podívejte se na Větu 9.8 popisující chordální grafy jako průnikové grafy podstromů ve stromě.

Jinou možnost popisu ukazuje:

`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 $V(G)$.

`Komentář:` Všimněte si, že uspořádání vrcholů z této definice je vlastně zpětnou simpliciální dekompozicí chordálního grafu $H \supseteq G$ z předchozí definice (viz také Věta 9.6).



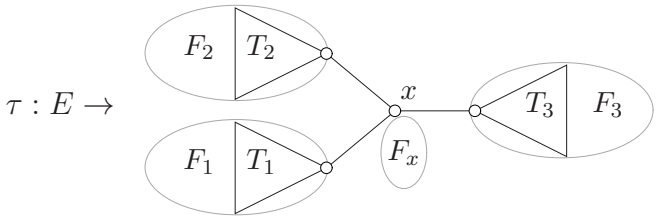
V nakresleném příkladě grafu C_5 vidíme uspořádání vrcholů se šířkou 2. (Tečkované hrany ukazují tzv. *chordální doplnění* grafu, relevantní k první definici tree-width.)

Ještě další, značně odlišný, přístup má tato definice:

`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 $F_i = \tau^{-1}\big(V(T_i)\big)$. 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.



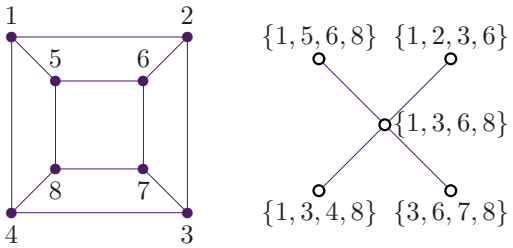
Třetí *stromovou šířkou* grafu G pak definujeme jako nejmenší možnou, přes všechny dvojice T, τ, hodnotu výrazu $\max_{t \in V(T)} \ell_\tau(t)$.

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

Definice 10.4. ***Stromová dekompozice*** grafu G.
Stromovou dekompozicí grafu G nazveme strom T spolu se systémem množin $\mathcal{X}_t$ (zvaných „balíky“) 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.



Přes veškerou zdánlivou odlišnost uvedených definic platí následovné.

Věta 10.5. *Všechny čtyři výše uvedené definice stromové šířky definují přesně tutéž hodnotu, pokud graf G má neprázdnou množinu hran.*

Důkaz plného znění této věty je však nad rámec našeho textu.

`Komentář`: Parametr stromové šířky omezuje pouze „globální“ strukturu grafu, grafy omezené tree-width mohou sice lokálně vypadat téměř jakkoliv, ale z celkového (globálního) pohledu musí dodržovat jistá velice striktní omezení. Neformálně lze říci, že když se na velký graf omezené tree-width podíváme z velké dálky (kdy už se jednotlivé vrcholy budou slévat), obrázek nám bude připomínat velký strom.

O četnících a zloději

Na závěr si představme *hru na četníky a zloděje* s těmito pravidly: Zloděj se bleskovou rychlostí pohybuje po hranách grafu přes vrcholy neobsazené četníky (jeho pohyb vždy skončí ve vrcholu, ne „uprostřed“ hrany). Naopak četníci se grafem vůbec nepohybují, jen přilétají do a odlétají z vrcholů helikoptérou. Zloděj je chycen ve svém vrcholu z přiletivším četníkem, pokud jsou i všichni sousedé z zrovna obsazeni četníky.

Věta 10.6. *Nejmenší počet četníků potřebných k zaručenému chycení zloděje v grafu G je roven stromové šířce G plus* 1.

Úlohy k řešení

(10.2.1) *Dokažte si sami ekvivalenci prvních dvou definic tree-width. Vyjděte přitom z existence simpliciální dekompozice chordálního grafu (Věta 9.6).*

(10.2.2) *Dokažte, že pro každou stromovou dekompozici grafu G platí, že každá klika v G se nachází celá v jednom některém balíku.*

(10.2.3) *Které grafy mají vysokou tree-width?*

*(10.2.4)** *Dokážete nalézt rovinný graf s vysokou tree-width?*

(10.2.5) *Jak mohou 3 četníci chytit zloděje na kružnici? A proč 2 nestačí?*

10.3 Některé další parametry

Pro ukázku si uvedeme jiné dva šířkové parametry, které místo „stromového tvaru“ strukturu grafu „linearizují“.

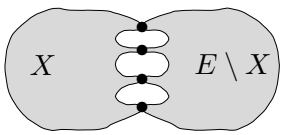
`Definice`: *Cestní* dekompozici a šířku (path-width) grafu G definujeme stejně jako v Definici 10.4, jen požadujeme navíc, aby T byla cesta.

`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 $\max_{v_iv_j \in E(G)} |i-j|$ přes všechny permutace vrcholů $V(G)$.

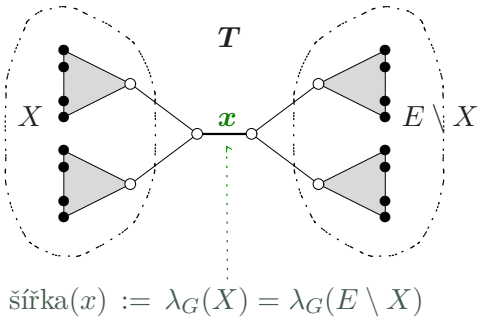
Větvené dekompozice

Další definice se váží k zásadně jinému, a přesto v konečném důsledku ekvivalentnímu pohledu na stromovou šířku. Graf je *kubický*, pokud má všechny vrcholy stupně 3. Strom je *podkubický*, pokud má všechny vrcholy stupně ≤ 3.

`Definice`: Pro libovolný graf G a podmnožinu $X \subseteq E(G)$ definujeme *funkci souvislosti* $\lambda_G(X)$ jako počet vrcholů G, které jsou konci některých hran z X i hran z $E(G) \setminus X$ (*separace* množiny X).

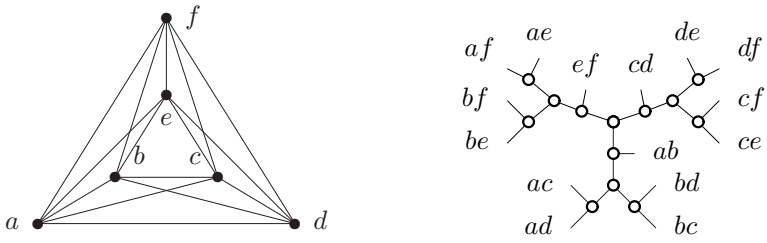


Definice 10.7. ***Větvená dekompozice*** grafu G
Nechť T je podkubický strom a $\tau : E(G) \to L(T)$ je bijekce hran grafu G do listů $L(T)$ stromu T. Pro každou hranu x stromu T definujeme šířku x jako $\lambda_G(X)$, kde $X = \tau^{-1}\big(V(T_1)\big)$ pro jednu z komponent T_1, T_2 lesa $T - x$.



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 5.)



Věta 10.8. *Pokud graf G má stromovou šířku t a větvenou šířku $b > 1$, tak*

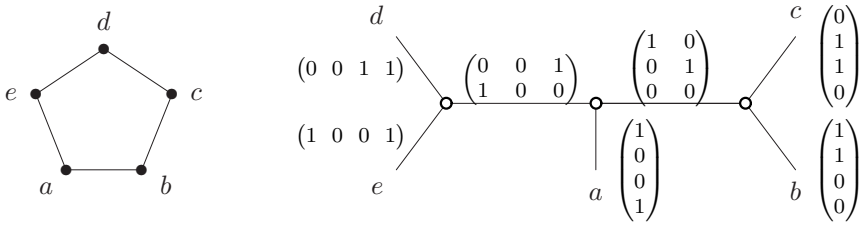
$$b \leq t + 1 \leq \left\lfloor \frac{3}{2} b \right\rfloor .$$

Jiný přístup

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

Definice 10.9. ***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 $X = \tau^{-1}\big(V(T_1)\big)$ 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.

`Komentář:` Následuje ukázka rankové dekompozice šířky 2 pro kružnici C_5.



`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.

Úlohy k řešení

(10.3.1) *Může graf omezené path-width mít vrcholy velkého stupně?*

*(10.3.2)** *Jak může souviset path-width grafu s hrou na četníky a zloděje?*

(10.3.3) *Jaká je bandwidth kružnice? Dokažte.*

(10.3.4) *Dokažte, že graf omezené bandwidth má omezený maximální stupeň. Jaký je přesný vztah max. stupně k bandwidth?*

(10.3.5) *Dokažte, že větvená šířka grafu K_4 je právě* 3.

(10.3.6) *Podobně dokažte, že větvená šířka grafu K_6 je právě* 4.

*(10.3.7)** *Dokažte Větu 10.8.*

(10.3.8) *Nalezněte příklady grafů, které dokazují těsnost obou nerovností ve Větě 10.8.*

(10.3.9) *Jaká je ranková šířka úplných bipartitních grafů?*

(10.3.10) *Najděte nějaký dosti velký graf, který není úplný ani úplný bipartitní, a přesto má rankovou šířku* 1.

(10.3.11) *Jaká je ranková šířka kružnic?*

*(10.3.12)** *Umíte najít nějaký graf velké rankové šířky?*

10.4 Efektivní algoritmy na dekompozicích

V této části si uvedeme pár ukázek jednoduchých algoritmů běžících „dynamicky“ na vhodné dekompozici grafu. Samotné řešené problémy jsou sice nedůležité, ale účelem je ukázat principy návrhu takových efektivních algoritmů.

Již víme z 7.18, že určení velikosti největší nezávislé množiny v grafu je $\mathcal{NP}$-úplný problém. Stromová dekompozice fixní šířky však tento problém umožňuje řešit velice snadno.

Algoritmus 10.10. ***Nezávislá množina na stromové dekompozici***
Danou stromovou dekompozici vstupního grafu si libovolně „zakořeníme“.

- V každém listě dekompozice vyřešíme problém hrubou silou v konstantním čase.
- Ve směru od listů ke kořeni sbíráme následující informaci:
  V každém balíku B dekompozice, pro každou $X \subseteq B$, velikost největší nezávislé množiny I v grafu indukovaném na podstromu pod B takové, že $I \cap B = X$.
- Vzhledem k interpolační vlastnosti naší dekompozice lze výše popsanou informaci v každém balíku dekompozice zjistit v konstantním čase pouze ze znalosti stejné informace ze všech jeho potomků v dekompozici.

Výsledný algoritmus pracuje v čase úměrném počtu uzlů dekompozice, tedy v lineárním čase!

Další problém hledání (maximálního) párování v grafu sice je polynomiálně řešitelný, ale zjišťování počtu všech párování už je *$\#\mathcal{P}$-úplné*, neboli stejně těžké (jinými slovy beznadějné) jako výpočet permanentu matice nebo spočítání všech řešení SAT problému.

Algoritmus 10.11. ***Počet párování na větvené dekompozici***
Opět si větvenou dekompozici vstupního grafu libovolně „zakořeníme“.

- V každém listě dekompozice je řešení triviální.
- Ve směru od listů ke kořeni sbíráme následující informaci:
  V každé hraně dekompozice, pro každou podmnožinu $X \subseteq S$ vrcholů separace S indukované touto hranou v dekompozici, počet všech párování, která ze separace S „obsazují“ právě vrcholy X.

- Opět lze výše popsanou informaci na každé hraně dekompozice zjistit v konstantním čase pouze ze znalosti stejné informace z obou jejich podstromů v dekompozici (vzájemným vynásobením a sečtením počtů).

Výsledný algoritmus pracuje v čase úměrném počtu hran dekompozice, tedy opět v lineárním čase.

Jeden všeobecný výsledek

Nápadná vzájemná podobnost předchozích algoritmů určitě není náhodná, chtěli jsme jimi ilustrovat jeden důležitý obecný princip, který objevili postupně [Courcelle / Arnborg, Lagergren, Seese / Borie, Parker, Tovey].

Věta 10.12. *Každá vlastnost grafů, která je vyjádřitelná v tzv. (E)MSO jazyce, se dá vypočítat v lineárním čase pro všechny grafy omezené stromové šířky.*

Pro zjednodušení nebudeme přesně definovat, co (E)MSO jazyk znamená, ale zhruba jde o jazyk, který má dovoleno kvantifikovat přes podmnožiny vrcholů a hran grafu a enumerovat počty prvků množin. Většina grafových vlastností, které jsme zatím probírali, spadá do této kategorie.

Úlohy k řešení

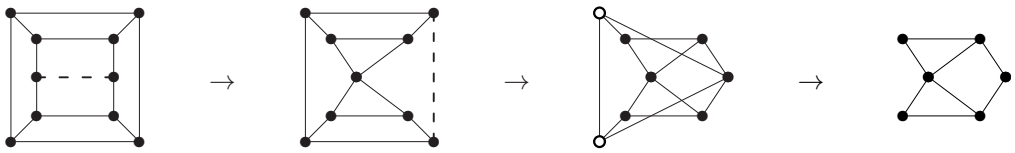
*(10.4.1) *Dokázali byste efektivně nalézt Hamiltonovskou kružnici v grafu na jeho větvené dekompozici fixní šířky?*

10.5 Minory v grafech

Závěrem nahlédneme do (nedozírných?) hlubin strkturální teorie grafů. Klíčem k ní je pojem minoru, který zobecňuje běžné podgrafy,

`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 10.13. *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!

Úlohy k řešení

(10.5.1) *Jak vypadají grafy, ve kterých zakážeme coby minor graf K_3?*

(10.5.2) *Jak vypadají grafy, ve kterých zakážeme coby minor graf K_4?*

*(10.5.3)** *Jak vypadají grafy, ve kterých zakážeme coby minory grafy K_4 a $K_{2,3}$?*

*(10.5.4)** *A teď 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

Pro zájemce o další studium strukturálních aspektů grafů doporučujeme výbornou (i když pokročilou) anglickou učebnici [1]. Specificky stromové šířce a teorii grafových minorů je věnována Kapitola 12 této knihy. Na FI MU se můžete o strukturální teorii grafů mnohem více dozvědět v navazujícím výběrovém předmětu MA052. Další zmínku o zajímavých souvislostech grafových minorů naleznete také v Lekci 11.

Šířkové parametry grafů se významně využívají v teorii parametrizované složitosti algoritmů, jak jsme naznačili v Oddíle 10.4. Něco více o použití různých šířkových parametrů v parametrizovaných algoritmech se můžete dozvědět v přehledovém článku [Width Parameters Beyond Tree-width and Their Applications; by PH–Oum–Seese–Gottlob, `http://dx.doi.org/10.1093/comjnl/bxm052`*].*

11 Pokročilé kreslení grafů

Úvod

Další výběrová lekce se soustřeďuje kolem problematiky vhodných nakreslení grafů, které nejsou rovinné. Pro takové případy se nabízejí dva základní směry zobecňování pojmu rovinného nakreslení grafu. Jeden směr zachová definici nakreslení tak, jak ji známe z rovinného případu, ale umožní kreslit na tzv. „vyšší plochy" jako torus nebo projektivní rovina, apod. Při výrazném navýšení kreslících možností nám zůstanou zachovány mnohé užitečné vlastnosti známé z rovinných grafů.

Na druhou stranu můžeme zůstat při kreslení v běžné rovině, ale povolit obecnější způsoby zobrazení grafu, třeba kreslení s (pokud možno malým počtem) křížením hran. Ještě jiné možnosti „obrázkové" reprezentace nakonec ilustrujeme tzv. rovinným pokrytím, které uvádíme jako zajímavou kuriozitu k zamyšlení.

Cíle

Úkolem první části této lekce je definovat nakreslení grafu na vyšších plochách (kdy si stručně uvedeme i klasifikaci ploch) a naučit se s nakresleními pracovat. Druhá část pak předestře některé další problémy vztahující se ke kreslení nerovinných grafů, třeba problematiku minimalizace počtu křížení hran (tzv. průsečíkové číslo).

11.1 Co jsou to plochy

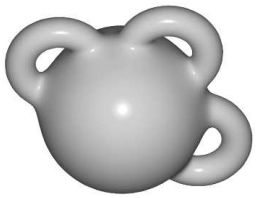
Mimo tradičního kreslení grafů „na papír", tj. do roviny, si lze představit kreslení grafů na složitější povrchy (*plochy*), třeba na povrch duše pneumatiky. Co nám takovéto rozšířené kreslení grafů přinese nového? Především možnosti nakreslit i nerovinné grafy bez křížení hran.

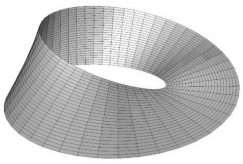
Nejprve si stručně uvedeme důležitý výsledek klasické topologie – *klasifikaci ploch*.

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

- $\mathcal{S}_0$ *sféře,*
- $\mathcal{S}_h$ *sféře s* h *přidanýma „ušima" (handle),*
- $\mathcal{N}_k$ *sféře s* k *přidanými „křížícími místy" (crosscap).*

`Komentář:` *Přidávání uší* na sféru je snadno představitelná konstrukce (viz následující ilustrace nalevo). Avšak *crosscap* je velmi obtížné vizualizovat v Euklidovské geometrii, takže pro ilustraci si jej téměř ekvivalentně můžeme nahradit připojováním *Möbiova proužku* (viz ilustrace napravo) k hranici polosféry.

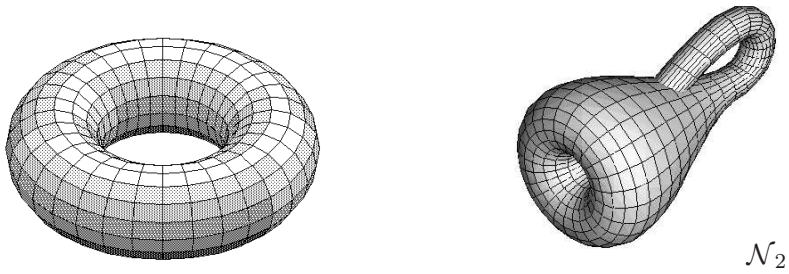




`Definice`: *Crosscap* na ploše je kružnice, jejíž protilehlé dvojice bodů jsou ztotožněny (vnitřek kruhu přitom ploše už nepatří).

`Značení`: Plocha $\mathcal{S}_1$ je známý *torus* (ilustrace následující vlevo), neboli povrch duše kola. Plocha $\mathcal{N}_1$ je *projektivní rovina* a vypuštěním kruhu z $\mathcal{N}_1$ vzniká zmíněný Möbiův proužek. Plocha $\mathcal{N}_2$ je tzv. *Kleinova láhev* (ilustrace vpravo), jejímž podélným

rozříznutím vzniknou dva Möbiovy proužky.



$\mathcal{S}_1$ $\mathcal{N}_2$

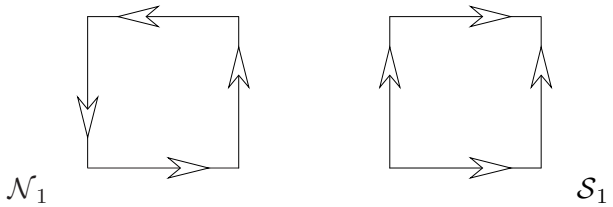
Plochy $\mathcal{S}_0$ a $\mathcal{S}_h$ jsou *orientovatelné*, kdežto $\mathcal{N}_k$ jsou *neorientovatelné*.

Všimněte si, že Věta 11.1 mluví jen o separátním přidávání uší nebo crosscapů. Co za plochu však vzniká kombinovaným přidáváním uší a crosscapů? Na to je snadná odpověď, vzniknou jen znovu už výše popsané plochy.

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.*

`Značení`: Klasická topologie používá následující způsob reprezentace vyšších ploch – plocha je zobrazena jako pravidelný mnohoúhelník, jehož strany jsou v naznačených orientacích po dvojicích ztotožněny.

Například ztotožněním protilehlých dvojic stran čtverce v naznačených směrech vznikají (zleva) projektivní rovina a torus.



$\mathcal{N}_1$ $\mathcal{S}_1$

11.2 Kreslení grafů na plochy

Přirozeným zobecnění rovinného nakreslení grafu je následující definice:

Definice 11.3. ***Nakreslením*** grafu G na plochu Σ (viz také Definice 8.1) 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. Čtenář by si však měl uvědomit, že na vyšších plochách nemusí stěny být „jen placaté“, a proto pro zajištění jisté konzistence s rovinným případem je třeba následujícího pojmu.

`Definice`: Nakreslení grafu G na plochu Σ je *buňkové*, pokud je každá jeho stěna (bez své hranice) homeomorfní otevřenému disku.

`Fakt`: Buňkové nakreslení 2-souvislého 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 u stěnových kružnic.)

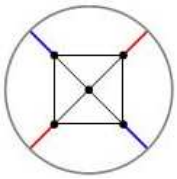
Tvrzení 11.4. *Buňkové nakreslení grafu G na orientovatelnou plochu Σ je jednoznačně určeno rotačním schématem vycházejících hran u svých vrcholů. (V případě neorientovatelných ploch je třeba ještě přidat jistá „znaménka“.)*

Rotační schéma u každého vrcholu v nakreslení určuje cyklické pořadí hran (v globálně zvolené orientaci) vycházejících z tohoto vrcholu v našem nakreslení.

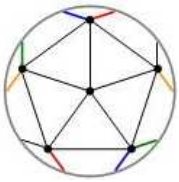
Kreslení na určenou plochu

První přirozenou otázkou o kreslení grafů na vyšší plochy je, zda dokážeme takto nakreslit i jiné než rovinné grafy. Například úplný graf K_5 nelze nakreslit do roviny.

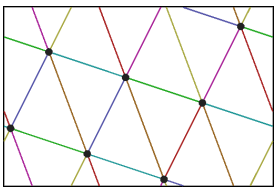
Tvrzení 11.5. *Do projektivní roviny lze bez křížení hran nakreslit úplné grafy K_5 i K_6, na torus také K_7, kdežto na Kleinovu láhev K_7 nakreslit nelze.*



K_5



K_6



K_7

Také mnohé jiné poznatky o kreslitelnosti grafů lze zobecnit z rovinných na vyšší plochy. Z těch jednoduchých je nejdůležitější Eulerův vztah (Věta 8.2).

Věta 11.6. *Nechť buňkové nakreslení neprázdného souvislého grafu G na ploše Σ má f stěn. Pak*

$$|V(G)| + f - |E(G)| = \chi(\Sigma)\,,$$

kde $\chi(\Sigma)$ (Eulerova charakteristika plochy) je $2-2h$ pro $\Sigma = \mathcal{S}_h$ a $2-k$ pro $\Sigma = \mathcal{N}_k$.

Obdobně rovinnému případu, z Eulerova vztahu vyplývají důležitá omezení na maximální počet hran jednoduchých grafů nakreslitelných na určené plochy. Například jednoduchý n-vrcholový graf nakreslený na toru nebo Kleinově láhvi nemůže mít více než $3n$ hran.

11.3 Překážky kreslení na plochy

Podle Věty 8.5 lze rovinnost zadaného grafu poměrně rychle algoritmicky rozhodnout i najít nakreslení. I tento silný výsledek má stejně silné zobecnění na vyšší plochy (Mohar), ale už bohužel není vhodný pro praktické implementace.

Věta 11.7. *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.*

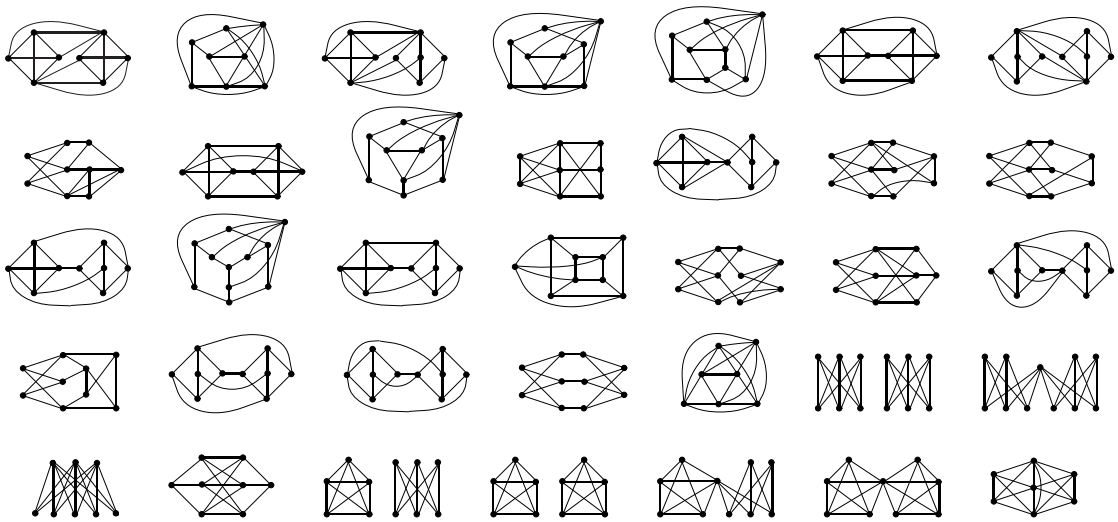
`Poznámka`: Za zmínku stojí fakt, že obecnější problém určit nejjednodušší plochu, na kterou lze daný graf nakreslit, je už $\mathcal{NP}$-těžký.

Z jiné strany lze zobecňovat Kuratowského Větu 8.8 na vyšší plochy.

`Fakt`: Vlastnost grafu být nakreslitelný na určenou plochu se zachovává pro všechny jeho podgrafy i minory.

Třebaže je známo, že zobecnění Kuratowského věty s konečným počtem překážek je platné pro každou plochu, konkrétní seznam zakázaných minorů či podrozdělení známe pouze u jediné vyšší plochy (Archdeacon):

Věta 11.8. *Graf G je nakreslitelný do projektivní roviny, právě když neobsahuje žádný minor isomorfní některému z následujících 35 grafů.*



Význam grafů na plochách

`Komentář:` Původní motivace výzkumu kreslení grafů na plochy je přirozená – byla to především snaha o řešení problému čtyř barev. Nyní však již máme Větu o čtyřech barvách, takže co dále motivuje výzkum kreslení grafů na vyšší plochy, mimo „pěkných obrázků“?

S grafy nakreslenými na vyšších plochách se setkáme ve dvou základních teoretických oblastech:

- Algebraické – studium pravidelných „map“ na plochách.
- Strukturální grafové – celá Robertson–Seymourova teorie grafových minorů, třebaže na první pohled s kreslením grafů nemá nic společného, stojí na grafech nakreslených na plochách.

Abychom si poslední poněkud překvapivý poznatek blíže vysvětlili, uvedeme si stručně následující asi nejdůležitější mezivýsledek Robertson–Seymourovy teorie.

Věta 11.9. *Mějme nerovinný graf H. Pak každý graf G, který neobsahuje minor isomorfní H, má stromovou dekompozici (Definice 10.4) následující vlastnosti: Každý její balík (bez ohledu na velikost) indukuje podgraf, který je až na omezeně mnoho „lokálních výjimek“ nakreslitelný na nějakou plochu Σ takovou, že H na Σ nakreslit nelze.*

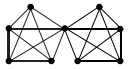
Úlohy k řešení

(11.3.1) *Nalezněte nakreslení grafu $K_{4,4}$ na torus. (Vezměte si k tomu jako pomůcku nějaký fyzický model toru, třeba posilovací kolečko.)*

*(11.3.2)** *Dokažte, že graf K_7 nelze nakreslit bez křížení na Kleinovu láhev.*

(11.3.3) *Určuje nám Eulerova charakteristika (Věta 11.6) jednoznačně plochu, na kterou je graf nakreslený? Kdy?*

(11.3.4) *Lze nějaký graf nakresit buňkově na různé plochy?*

(11.3.5) *Proč tento graf*  *nelze nakreslit do projektivní roviny?*

(11.3.6) *Proč graf $K_{3,5}$ nelze nakreslit do projektivní roviny?*

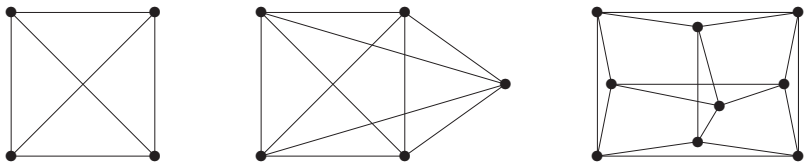
*(11.3.7)** *Uměli byste o dalších, složitějších, grafech z Věty 11.8 dokázat, že nejsou projektivní?*

11.4 O průsečíkovém čísle grafů

Mimo Definici 11.3 je přirozené uvažovat ještě jedno zobecnění rovinných nakreslení: Jak přistupovat k hezkým kreslením nerovinných grafů do roviny?

Definice 11.10. ***Obecným nakreslením*** (s křížením hran) grafu G do roviny (viz také Definice 8.1) 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 se dvojice hran v tom bodě „křížila“ (ne jednostranný dotyk).

Příklad 11.11. *Podívejme se na následující tři (obecná) nakreslení do roviny. Jsou všechna „optimální“, tj. je počet jejich křížení nejmenší možný?*



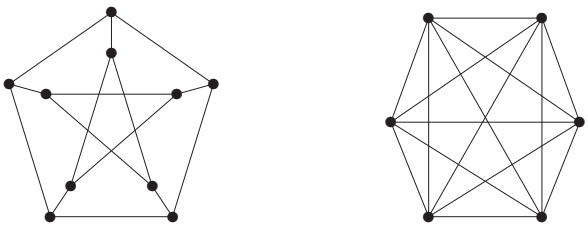
Snadno vidíme, že první graf lze nakreslit i bez křížení a druhý graf jen s jedním křížením. Naopak třetí graf už s méně kříženími nakreslit nelze. Uměli byste toto dokázat? □

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

`Komentář:` Původ problému průsečíkového čísla spadá do doby druhé světové války, kdy matematik P. Turán byl na nucených pracech v cihelně. Jeho úkolem bylo tlačit vozíky s cihlami po kolejnicích a na každé křižovatce kolejí potkával 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 11.12. *Určeme průsečíková čísla následujících dvou grafů:*



Asi snadno každý nalezne nakreslení prvního grafu (Petersenův) s pouze dvěma kříženími hran. Jak však dokázat, že jej nelze nakreslit s jedním křížením? Všimněme si, že každá jeho hrana je „ekvivalentní“ s každou jinou. To znamená, že by ostranění jedné zvolené hrany z Petersenova grafu mělo vytvořit rovinný graf, ale to není pravda.

Druhý graf K_6 lze nakreslit s 3 kříženími – začněme s kružnicí délky 5 a posledním vrcholem spojeným uvnitř. Zbylých 5 hran pak už dokážeme dokreslit s vytvořením tří křížení. Zkuste si to! Proč není možných méně křížení? Předpokládejme, že dvě křížení hran postačují, pak bychom vypuštěním dvou dotčených hran získali rovinný graf. Ten by však měl 6 vrcholů a $13 > 3 \cdot 6 - 6$ hran, spor. □

Ač prozatím může studovaná problematika vypadat jako hříčka či hračka, ve skutečnosti se jedná o nezvykle obtížný grafový problém. To ostatně nejlépe ilustruje už následující fakt:

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

Co se týče výpočetní složitosti problému, během let bylo dokázáno následovné (výsledky postupně Garey–Johnson 1978, autor 2004 a nakonec Cabello–Mohar 2010):

Věta 11.13. *Problém určit, zda průsečíkové číslo $cr(G) \leq k$ pro G a k na vstupu je $\mathcal{NP}$-úplný. Toto platí i když G je kubický 3-souvislý graf.*

Dokonce je-li dán rovinný graf G a dvojice jeho nespojených vrcholů u, v, tak problém určit pro k na vstupu, zda průsečíkové číslo $cr(G + uv) \leq k$, je $\mathcal{NP}$-úplný.

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

`Komentář:` V praxi se ukazuje, že určení průsečíkového čísla je přímo zoufale těžký problém, ještě mnohem beznadějnější než třeba zjištění barevnosti. Snad jediným existujícím „pozitivním“ (i když zcela nepraktickým) přesným algoritmickým výsledkem je následující výsledek Groheho a poté Kawarabayashi–Reeda:

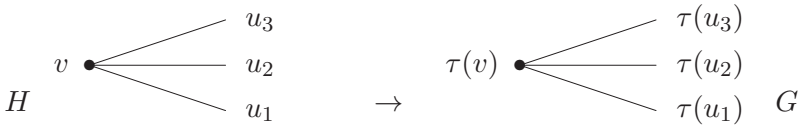
Věta 11.14. *Pro fixní k lze otestovat, zda $cr(G) \leq k$, v lineárním čase vzhledem k počtu vrcholů grafu (závislost na parametru k je však doslova „brutální“).*

Další existující pozitivní algoritmické výsledky se pak týkají už jen aproximací průsečíkového čísla nebo výpočtů pro konkrétní malé grafy (řádově v desítkách vrcholů). To však překračuje rozsah našeho textu.

11.5 O problému rovinného pokrytí

Na závěr si jako kuriozitu uvedeme zajímavý, třebaže okrajový, problém známý od 80-tých let pod názvem *Negamiho hypotéza planárních pokrytí* nebo Negamiho $12\,\infty$ hypotéza. Avšak k velmi podobné otázce nezávisle ve stejné době dospěl i Fellows. Problém je hezký především svou „chytlavostí“ a jednoduchostí zadání.

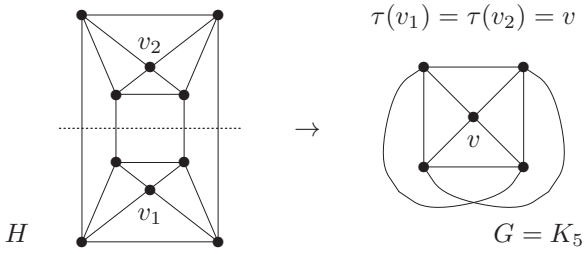
`Definice`: Říkáme, že graf H *pokrývá* graf G, pokud existuje surjektivní zobrazení $\tau : V(H) \to V(G)$ takové, že sousedé každého vrcholu v grafu H jsou bijektivně zobrazeny na sousedy vrcholu $\tau(v)$ grafu G.



Negamiho hypotéza rovinných pokrytí

`Hypotéza 11.15.` 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 pojmenováním “centrálních” vrcholů.



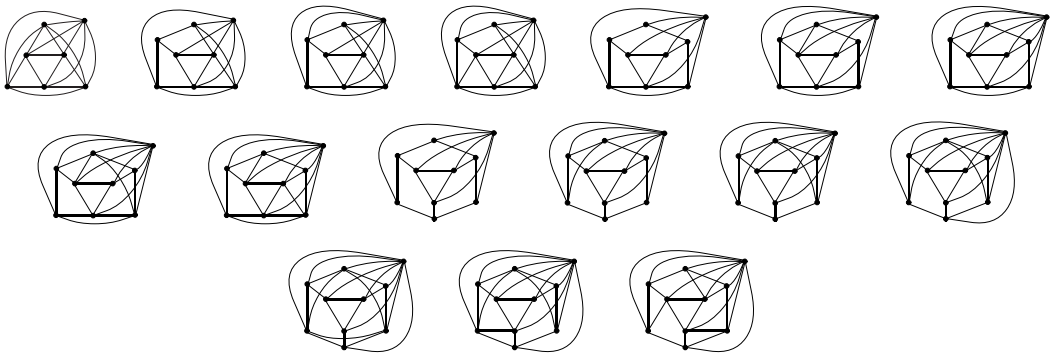
`Fakt`: Je-li G nakreslitelný do projektivní roviny, pak univerzální pokrytí projektivní roviny sférou okamžitě dá nakreslení dvojitého rovinného pokrytí grafu G.

Lema 11.16. *K důkazu Hypotézy 11.15 stačí ověřit, že žádný z* 32 *souvislých zakázaných minorů z Věty 11.8 nemá konečné rovinné pokrytí.*

Kupodivu pro většinu z oněch 32 grafů to lze dokázat velmi snadno. Další výsledky, na kterých se podíleli Archdeacon, Fellows, Negami, Thomas a autor, vedly k dále uvedeným poznatkům, které jsou doposud tím nejsilnějším, co o řešení Negamiho hypotézy víme.

Věta 11.17. *Pokud graf $K_{1,2,2,2}$ nemá konečné rovinné pokrytí, tak je Hypotéza 11.15 pravdivá.*

Věta 11.18. *Existuje jen* 16 *následujících konkrétních grafů (až na triviální modifikace), pro které by Hypotéza 11.15 mohla být nepravdivá.*

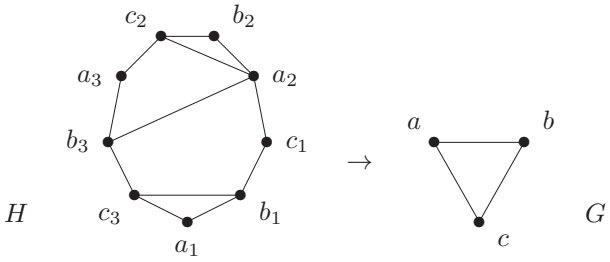


O rovinných emulátorech

Mírnou modifikací konceptu planárního pokrytí představuje tato definice, podaná Fellowsem nezávisle na Negamim.

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

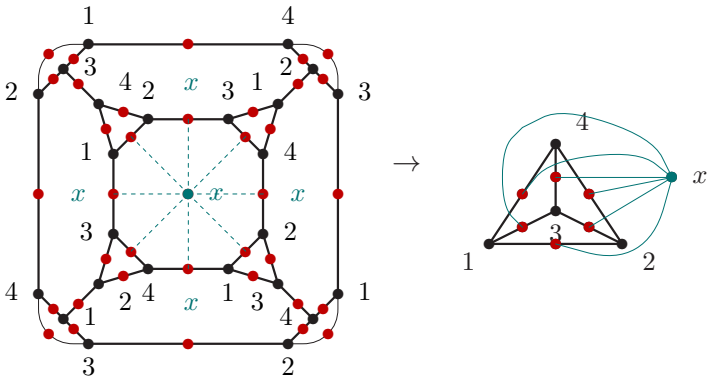
`Komentář`: Na rozdíl od rovinných pokrytí mohou mít emulátory poměrně bohatší strukturu, přesněji řečeno sousedé mohou být "duplikováni", viz tento příklad emulátoru trojúhelníka:



Jelikož je na první pohled z definice "jasné", že duplikace sousedů nemůže být přínosná pro existenci planárních emulátorů (ve srovnání s pokrytími), již Fellows vyslovil domněnku, že souvislý graf má konečné rovinné pokrytí právě tehdy, když má konečný rovinný emulátor. Přesto se na závěr roku 2008 objevilo skutečné překvapení:

Věta 11.19. (Rieck a Yamashita) *Existuje graf, který není projektivní a nemá konečné rovinné pokrytí a přesto má konečný rovinný emulátor.*

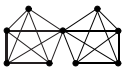
Jiné příklady grafů z Věty 11.19 byly následně objeveny Chimanim a autorem a jeden z nich (až překvapivě jednoduchý a malý) ukazuje náznakem tento obrázek:



Problematika, které grafy mají konečné rovinné emulátory, je stále široce otevřená.

Úlohy k řešení

(11.5.1) *Dokažte si sami Lema 11.16.*

(11.5.2) *Proč tento graf*  *nemá konečné rovinné pokrytí?*

(11.5.3) *Dokažte, že graf $K_{3,5}$ nemá konečné rovinné pokrytí.*

(11.5.4) *Řešte předchozí dvě otázky pro emulátory místo pokrytí.*

Rozšiřující studium

Při popisu kreslení grafů na plochy vycházíme z výborné monografie Mohara a Thomassena [7]. Na této monografii také zakládáme pokročilý výběrový předmět MA051 na FI MU, který zájemcům o grafy vřele doporučujeme. V této krátké lekci však problematiku velmi zjednodušujeme z důvodu nedostatku místa, takže pro zájemce může být přínosné si přečíst, jak kreslení grafů na plochy hluboce souvisí s fundamentálními problémy klasické topologie v Thomassenově skvělém podání.

Co se týče Negamiho hypotézy a planárních pokrytí, obsáhlý ucelený popis lze nalézt v autorově dizertační práci (Georgia Tech, 1999)
`http://www.fi.muni.cz/~hlineny/Research/papers/gtthesis.pdf` *a doplnit nejnovější informace lze ze shrnujícího článku (2009)*
`http://www.fi.muni.cz/~hlineny/Research/papers/plcover20-gc.pdf`.

K průsečíkovému číslu grafů (není nám o něm známa žádná tištěná monografie) lze nalézt množství doplňujících informací na internetu, jako třeba
`http://en.wikipedia.org/wiki/Crossing_number`.

Část V

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.

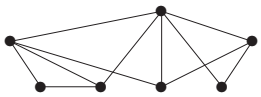
(1.1.4) Pro $m = n = 2$, délky čtyři.

(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.

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

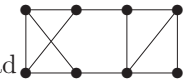
(1.3.1) Ano, ale musí mít délku aspoň o 2 kratší než délka kružnice. Vidíte proč?

(1.3.2) 3, více nelze, neboť na obvodě kružnice by se musely ob-jeden střídat.

(1.3.3) 3

(1.3.4) 7

(1.3.5)* 5, dokážete ji vůbec najít?

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

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

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

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

(1.6.1) Ano.

(1.6.2) Ne.

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

(1.6.4) Dva.

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

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

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

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

(1.6.9) Vlevo 4, vpravo 3.

(1.6.10) Vlevo 3, vpravo 3.

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

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

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

(1.6.14) Ano, třeba v úloze 1.6.7 vlevo.

(1.6.15) ...

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

2 zanedbat a zbytek musí být K_4.

(1.6.17)* 20, po vynechání jednoho vrcholu jsou vždy dvě možnosti na vyplnění zbytku kružnicí.

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

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

(1.6.20)* 11

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

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

(1.7.1) 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.

(1.7.2) 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.

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

(1.7.4)* 5

(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) Prostě začněte z C_5.

(2.3.4) 1 do kružnice.

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

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

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

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

(2.4.3) Právě když neleží v žádné orientované kružnici.

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

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

(2.5.2) Ne, má čtyři vrcholy lichého stupně.

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

(2.6.1) 5, samé trojúhelníky.

(2.6.2) 6, samé úplné grafy K_5.

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

(2.6.4) 18, stupně ≥ 3, třeba jako šestiboký hranol.

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

(2.6.6)* Některý vrchol má stupeň aspoň 3, jinak vezmeme doplněk. Pak ti tři sousedi jsou nezávislí...

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

(2.6.8)* Pokud máme vrchol, ve kterém žádná šipka nekončí, položíme jej první v řadě a pokračujeme indukcí. Jinak v grafu nalezneme orientovanou kružnici (neznámé délky) a z ní lze minimalizací získat kružnici délky 3.

(2.6.9) Ano, ale jen otevřeným.

(2.6.10) Není to strom, proto obsahuje kružnici.

(2.6.11)* Pomůže vám nakreslení uzavřeným tahem; co tak vybrat z něj každou druhou hranu?

(2.6.12) Jde o Eulerovský tah, který musí být uzavřený, neboť všechny stupně jsou sudé 6 (plus

2 za smyčku).

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

(3.1.2) 2

(3.1.3) 5

(3.1.4) 7

(3.1.5) $n-1$ – tj. počet hran kostry.

(3.1.6) 3, najdete příslušnou dvojici vrcholů?

(3.1.7) Poloměr 2, centrum obsahuje právě 3 vrcholy (najdete je?).

(3.1.8) Třeba: 

(3.1.9)* $\mathrm{rad}(G) \leq \mathrm{diam}(G) \leq 2 \cdot \mathrm{rad}(G)$, rovnost v prvním pro K_n, v druhém pro cestu.

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

(3.2.2) Není, hodnoty s jedním indexem t se přece v iteraci t nezmění nikdy.

(3.3.1) 5

(3.3.2) Jsou to oba vrcholy „nad“ a, b, do každého dalšího vrcholu je z nich vzdálenost nejvýše 4. Lépe to nejde.

(3.3.3) Například ve třídě všech acyklických orientovaných grafů.

(3.4.1) Krok výběru dalšího vrcholu ke zpracování – i když bereme nejbližší nezpracovaný vrchol, z jiného, vzdálenějšího, vrcholu se lze později přes hranu záporné délky dostat do vybraného vrcholu “kratší” cestou.

(3.4.2) Bohužel nebude, důvod je stejný jako v předchozí otázce. Je nutno použít jiné algoritmy.

(3.5.1) $w(xy) + p_v(y) \geq ||x, y|| + p_v(y) \geq p_v(x)$

(3.5.2) Pak (za samozřejmého předpokladu $p_v(v) = 0$) nutně bude upravená délka některé hrany na optimální cestě z x do v záporná. Neboli býti dolním odhadem je nutná podmínka pro přípustný potenciál, bohužel ale obecně nedostačující.

(3.6.1) Vlevo 3 a vpravo 2.

(3.6.2) Vlevo 3 a vpravo 5.

(3.6.3) 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.

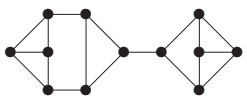
(3.6.4) 2 – jedná se o graf krychle, tak přidáme dvě úhlopříčky na jednu stěnu–čtverec. Pak budou vzdálenosti ≤ 2. Naopak jedna hrana nestačí, protože po přidání libovolné hrany e stále najdeme dva protilehlé vrcholy krychle, které hraně e nepatří, a tyto dva vrcholy zůstanou ve vzdálenosti 3.

(3.6.5) $2 + 3 + 2 = 7$

(3.6.6) 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!

(3.6.7) Vzdálenost 14 mezi 7 a 5. Dokážete zdůvodnit, že větší vzdálenost v grafu není?

(3.6.8)* Spočítejte si, kolik vrcholů může být do vzdálenosti 3...

(3.6.9) 

(3.6.10)* Ano, musí.

(3.6.11) Ano.

(3.6.12)* Osmi.

(3.6.13) Například zakázané odbočení na křižovatce může vynutit u optimální trasy prijetí

křižovatky přímo a pak třeba návrat z opačného směru, ze kterého již odbočit lze. Je i spousta jiných příkladů.

(3.6.14)* Klíčovou myšlenkou je, že do úschovny nalezených vrcholů ke zpracování se nebudou ukládat jednotlivé vrcholy samotné, nýbrž spolu s vhodným sufixem sledu, kterým byly dosaženy. Jeden vrchol se tak v úschovně může vyskytnou mnohokrát v závislosti na tom, v jakých se nachází manévrech.

(4.1.1) V podstatě nijak, pouze touto smyčkou může cirkulovat jisté množství substance bez vlivu na velikost toku.

(4.1.2) Rozdílem těchto dvou toků je tzv. cirkulace, splňující zachování substance ve všech vrcholech včetně z, s.

(4.2.1) Tok 5.

(4.2.2) Jen zdroj z.

(4.2.3) Řez velikosti 4 oddělující dva vrcholy vpravo nahoře.

(4.2.4) Protože rezervy kapacit jsou celočíselné, neboli aspoň +1 v každé iteraci, takže v konečném počtu kroků se dostaneme k maximálnímu toku.

(4.2.5)* Myšlenka: Pokud se jednou nasycená hrana vrátí mezi nenasycené, bude to určitě na delší nenasycené cestě.

(4.3.1) Snadné...

(4.3.2)* Klíčové je dokázat, že získaný tok v první fázi po navýšení vytváří cirkulaci na původním grafu G (součty ve všech vrcholech 0).

(4.4.1) Velikosti 5:

(4.4.2) Velikosti 4:

(4.4.3) Ano, reprezentanti jsou zapsaní první: $(1, 2, 3)$, $(2, 1, 4)$, $(3, 1, 4)$, $(4, 2, 3)$.

(4.4.4) Ne, všech podmnožin je $\binom{5}{3} = 10$, ale prvků k reprezentaci jen 5.

(4.5.1) 12

(4.5.2) 13, 11 a 9, po řadě.

(4.5.3) Nemá – 6 z množin má sjednocení $\{3, 4, 5, 6, 7\}$, kde se najde jen 5 různých prvků pro reprezentanty.

(4.5.4) Nemá – 6 z množin má sjednocení $\{1, 2, 3, 4, 9\}$, kde se najde jen 5 různých prvků pro reprezentanty.

(4.5.5)* Stejně jako v Příkladě 4.18, jen je třeba zdvojit každý vrchol grafu (aby mohl reprezentovat dvě vycházející šipky).

(4.5.6)* Jedná se o další zajímavé použití toků v síti, kde hledaná procházka odpovídá vlastně minimálnímu řezu ve vhodně odvozeném grafu z G a oddělující kružnice jsou naopak získány z maximálního celočíselného toku.

(4.5.7) Viz http://en.wikipedia.org/wiki/Ford-Fulkerson_algorithm#Non-terminating_example

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

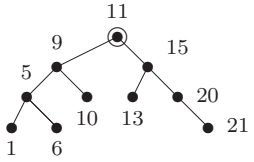
(5.1.2) Jsou to cesty P_n a úplné bipartitní grafy $K_{1,n}$.

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

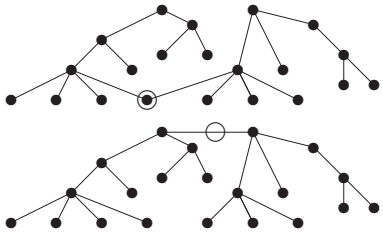
(5.1.4) 3, jeden bez hran, jeden s jednou hranou a poslední strom s dvěma hranami.

(5.1.5) 2, cesta P_3 a hvězda $K_{1,3}$.

(5.2.1)



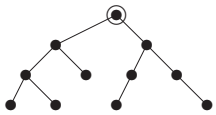
(5.2.2) Centra jsou vyznačená:



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

(5.3.1) ((()(()()())) ((()())(())))

(5.3.2)



(5.4.1) 1, bez hran.

(5.4.2) 2004

(5.4.3)* Je to jen o procvičení symbolických úprav velkých matic...

(5.4.4) 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.

(5.4.5)* $m^{n-1}n^{m-1}$

(5.4.6) Třeba kružnice C_9 a C_{223} sdílející jeden vrchol, neboť $9 \cdot 223 = 2007$.

(5.5.1) 3

(5.5.2) Ano, 2005 vrcholů podle Věty 5.3.

(5.5.3) $2009 - 1 - 1 - 1 - 1 = 2005$

(5.5.4) 11, počítejte hrany v každé komponentě – stromu.

(5.5.5)* Nenajdete, to by bylo v rozporu s Důsledkem 5.5.

(5.5.6)* Jen 4 (odtrhejte 8 vrcholů stupně 1, co zbude?).

(5.5.7) Graf není souvislý, 13 není spojeno s ničím. Není ani lesem, neboť $10, 14, 21, 15$ tvoří kružnici.

(5.5.8) Vlevo B, vpravo A.

(5.5.9) 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.

(5.5.10)* Levý graf 14. Pravý 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.

(5.5.11) $\binom{n}{2} - n + 1$

(5.5.12)* 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.

(5.5.13)* Třeba kružnice C_{503} s tětivou tvořící kružnici délky 4. Zamyslete se, jak k takovému řešení lze systematicky dospět...

(5.5.14) Platí.

(5.5.15)* Všechny kostry K_n mají dohromady $n^{n-2} \cdot (n-1)$ hran, takže na jednu hranu (které je symetrická všem ostatním) připadá $2n^{n-3}$ koster. $K_n - e$ tudíž má $n^{n-2} - 2n^{n-3}$ koster.

(6.1.1) Nalezneme kostru s největším celkovým ohodnocením.

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

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

(6.2.1) V jistém časovém okamžiku vidíme, že se zpracovávají 4 úkoly najednou, a proto méně než 4 pracovníci nestačí.

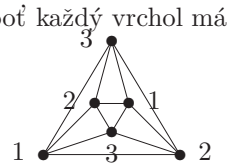
(6.2.2) Také to bude fungovat, dokažte si sami.

(6.4.1) Jakkoliv špatně, pro každý zvolený počet barev se dá nalézt špatný příklad, zkuste si to...

(6.5.1) V pořadí jakéhokoliv souvislého prohledávání našeho grafu.

(6.5.2) Jednoduše, prostě hladový algoritmus spustíme a on nikdy nepoužije barvu vyšší než $k+1$, neboť každý vrchol má jen k (možná) obarvených sousedů.

(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) 0 pro prázdný, 1 pokud má jeden vrchol a jinak 2 podle Věty 7.6 – stromy totiž nemají žádné kružnice.

(7.2.1) 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.

(7.2.2) 3 oboje.

(7.2.3) Na sdíleném trojúhelníku má každý z G, H tři různé barvy. Tudíž barvy H permutujeme tak (nezávisle na G), aby byly shodné s barvami G na tomto trojúhelníku.

(7.2.4)* Jestliže jsou V_1, V_2 vrcholové komponenty $G-\{u,v\}$ a u, v dohromady mají více sousedů ve V_1, obarvíme nejprve podgraf indukovaný na $V_1 \cup \{u,v\}$. Pokud u, v získají stejnou barvu, obarvíme indukčně podgraf indukovaný na $V_2 \cup \{u,v\}$ se ztotožněnými vrcholy u, v, jinak tento podgraf s hranou uv navíc. Nakonec obě obarvení ztotožníme vhodnou permutací barev.

(7.2.5)* Nechť F je podgraf H. Jelikož je nyní $G-\{u,v\}$ souvislý, v grafu $H \cup \{w\}$ vede cesta z w do nejbližšího vrcholu t podgrafu F mimo $u=v$, takže t má v F stupeň menší než k.

(7.2.6) Představte si w obarvené barvou 1, která pak nemůže být v žádném ze sousedů. Dále dokážeme, pro každé i, že vrchol v_i můžeme nakonec obarvit (či přebarvit) stejně, jako vrchol u_i, a tudíž dostáváme spor s barevností původního grafu.

(7.3.1) 3, stejně jako u vrcholové barevnosti.

(7.3.2)* $n-1$ pro sudá n a n pro lichá mimo 1.

(7.3.3) Pokud by stačilo 5 barev, jedna barva by se musela vyskytnout $|E(G)|/5 = 16/5$ tj. aspoň 4-krát, což na 7 vrcholech nelze.

(7.3.4) Ano, včetně důkazu.

(7.3.5) 3, neboť pro seznamy 12 a 34 na jedné straně a 13, 14, 23, 24 na druhé straně není obarvení.

(7.3.6)* Podobně předchozí úloze...

(7.3.7)* 3-regulární graf má sudý počet vrcholů, tudíž hrany Ham. kružnice umíme střídavě obarvit barvami 1, 2 a barvu 3 použít na zbylé hrany. Proto 3-regulární graf s Ham. kružnicí má hranovou barevnost 3.

(7.3.8) Stačí vzít nějaký s vrcholem, jehož vypuštění poruší souvislost.

(7.4.1) A,B,C,F

(7.4.2) Ano, patří, tyto kružnice napovíme a snadno ověříme.

(7.4.3) Ani nemůže patřit, neboť se nejedná o rozhodovací problém!

(7.4.4)* 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 napovězené 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.

(7.4.5)* A,D

(7.4.6) Je to kupodivu ještě jednodušší než v Tvrzení 7.20. 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.

(7.4.7) V převodu připojte ke každému vrcholu nový vrchol stupně 1.

(7.4.8)* 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.

(7.6.1) 3

(7.6.2) 4, všimněte si, že graf obsahuje K_4.

(7.6.3) 3

(7.6.4) 1

(7.6.5) 14, pokud vypustíme tři hrany z jednoho vrcholu, nebo 13, pokud vypustíme tři hrany jednoho trojúhelníku, nebo 12, pokud vypustíme tři disjunktní hrany.

(7.6.6)* Dokážeme, že takto vzniklý graf je 2-degenerovaný, neboť každý graf stupně nejméně 3 musí ke stromu přidat aspoň 3 hrany.

(7.6.7) Ano.

(7.6.8) Jen pro $m = n \geq 2$.

(7.6.9) 

(7.7.1) Je to stejný převod, neboť to funguje v obou směrech.

(7.7.2) 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.

(7.7.3)* Toto je fakt obtížné...

(7.7.4)* 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.

(7.7.5) 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) Graf pravidelného osmistěnu.

(8.1.3) Užijte projekci z bodu velmi blízkého zvolené stěně.

(8.1.4)* Obtížné...

(8.2.1) $v = 12$, $f = 20$, $v + f - h = 2$, tedy $h = 12 + 20 - 2 = 30$.

(8.2.2) 3

(8.2.3)* Jako $v - h + f = 1 + k$. Všimněte si, že za každou novou komponentu sice na pravé straně vztahu "přibude" 2, ale jedna vnější stěna se sdílí dohromady mezi všechny komponenty, a proto skutečně se pravá strana s každou komponentou zvýší o $+1$.

(8.3.1) Rovinný jen B.

(8.3.2) Pro $n = 1, 2, 3, 4$.

(8.3.3) Jen pokud $m \leq 2$ nebo $n \leq 2$, jinak v sobě obsahuje $K_{3,3}$.

(8.3.4) 3 – takto z kružnice délky 6 vytvoříme graf $K_{3,3}$, nakreslete si to.

(8.3.5)* Takové grafy jsou buď rovinné nebo jsou to podrozdělení K_5, nebo vznikají spojením menších na řezech velikosti ≤ 2.

(8.4.1) 4

(8.4.2)* ≤ 2

(8.4.3) Stačí dokázat, že takový graf je 2-degenerovaný.

(8.4.4)* Viz článek DOI: 10.1016/0012-365X(95)00216-J, M. Voigt.

(8.4.5) Protože každý rovinný graf lze obarvit 4 barvami a mezi 13 vrcholy čtyř barev je jedna barva aspoň na 4 vrcholech, mezi kterými pak nemohou být hrany.

(8.6.1) Ano, stačí dolní levé dva vrcholy překreslit nahoru.

(8.6.2) Ne, obsahuje podrozdělení $K_{3,3}$, najdete jej?

(8.6.3) 8

(8.6.4) 11

(8.6.5) Má vždy 10 vrcholů podle Eulerova vztahu.

(8.6.6) $\binom{7}{2} - (3 \cdot 7 - 6) = 6$

(8.6.7)* Sporem, co bude se stěnou přilehlou ke dvojici vrcholů tvořících řez?

(8.6.8) Ano, stačí 2-řez, který jej rozdělí na mnoho komponent a každou z nich lze nezávisle zrcadlově převracet.

(9.1.1) Podívejte se, sporem, na interval nejvíce vlevo – jeho dva sousedé na kružnici se pak musí překrývat...

(9.1.2)* Opět si pomožte úvahami o položení a nuceném překrývání intervalů v reprezentaci...

(9.1.3)* Intervalové a zároveň neobsahující indukované $K_{1,3}$.

(9.1.4) Neprotínající-se intervaly uspořádáme podle jejich polohy na ose (zleva doprava).

(9.2.1) Podívejte se na interval, který má první pravý konec zleva – všichni jeho sousedé se překrývají. Totéž zprava.

(9.2.2) Podáte jeho simpliciální dekompozici.

(9.2.3)* Dokazujte sporem – pokud se jeden vrchol v kliky nepřekrývá se všemi, lze velký strom rozpojit vynechání hrany, ale pak podstrom reprezentující v ukazuje na existenci kružnice.

(9.3.1) Například velká kružnice s "ocáskem" délky 2.

(9.3.2) Například kružnice délky 5 s dalším vrcholem spojeným všude.

(9.3.3) Například $K_{7,7}$. Umíte toto krátce zdůvodnit?

(9.3.4)* Například $K_{13,13}$, souvisí s problémem zvaným "kissing number"...

(9.3.5)* Spojte úsečkama středy kruhů a dokažte, že se takové hrany neprotínají.

(9.4.1) Vezměte rovinné nakrelsení a kolem každého jeho vrcholu "obejděte půlhrany" z něj vycházející jeho křivkou.

(9.4.2) K_4

(9.4.3) ...

(9.4.4)* Rovinný nebo rozbor speciálních případů.

(9.4.5) Třeba do každé trojúhelníkové stěny K_4 přidejte nový vrchol spojený do vrcholů toho trojúhelníka.

(9.4.6)* To by byl velký výsledek, neboť by implikoval Větu o čtyřech barvách!

(10.1.1)* Uspořádáme intervaly podle jejich konců a vždy do dominující množiny vložíme poslední interval protínající se s prvním dosud nepokrytým.

(10.1.2) Jako doplněk nezávislé množiny, kterou umíme.

(10.1.3)* Myšlenka je následovná: Je-li x simpliciální vrchol, tak ze sousedů x může být v nezávislé množině jen jeden (tvoří kliku), takže si neuškodíme, pokud dáme do nezávislé množiny přímo x.

(10.2.1) Docela přímočaré – uspořádání v druhé definici je přesně postup odtrhávání simpliciálních vrcholů od konce...

(10.2.2) Jinak by se porušila interpolační vlastnost; každá dvojice kliky má hranu a ta musí být ve společném balíku.

(10.2.3) Třeba úplné podle předchozí otázky...

(10.2.4)* Třeba velká "mříž". Podle hry na četníky a zloděje je třeba více četníku než je počet řádků / sloupců mříže. Proč?

(10.2.5) Jeden četník se postaví napevno a další dva vzájemným přeskakováním projdou zbytek kružnice. Pokud jsou jen dva četníci, v některém okamžiku se jeden z nich musí zvednou v helikoptéře a tehdy zloděj uteče kamkoliv jinam – hra je zas na začátku a nikam nevede.

(10.3.1) Ano.

(10.3.2)* Je potřeba přesně path-width plus 1 četníků k chycení zloděje, kterého četníci nevidí dokud ho nedopadnou.

(10.3.3) 2, menší být nemůže a uspořádání na 2 si najdete sami...

(10.3.4) $\Delta \leq 2 \cdot \mathit{bandwidth}$

(10.3.5) Horní odhad snadný, pro dolní je třeba nahlédnout, že každý podkubický strom s 6 listy má hranu oddělující aspoň dva listy na každé straně.

(10.3.6) Využijte, že každý podkubický strom s 15 listy má hranu oddělující na každé straně

aspoň 5 listů.

(10.3.7)* Převádějte dekompozice jednu na druhou...

(10.3.8) Zprava třeba úplné grafy, zleva úplné bipartitní s odebraným párováním.

(10.3.9) 1

(10.3.10) Třeba cesty.

(10.3.11) 1 pro C_3, C_4, jinak 2.

(10.3.12)* Třeba velká mříž, podobně jako s tree-width.

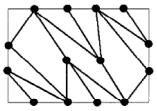
(10.4.1)* Ano, ale není to lehké... Je třeba ukládat všechny možnosti spárování hraničních vrcholů (pro separace v dekompozici) pomocí systému cest pokrývajících všechny vrcholu v tomto podstromu dekompozice. To je vzhledem k fixní šířce dekompozice konstantně velká informace.

(10.5.1) Jsou to lesy, acyklické grafy.

(10.5.2) Jsou to tzv. sériově–paralelní grafy.

(10.5.3)* Jsou to grafy, které lze nakreslit do roviny tak, aby všechny vrcholy ležely na vnější stěně.

(10.5.4)* No, měli bychom důkaz $P = NP$, ale žádný použitelný algoritmus k tomu. Zajímavá situace...

(11.3.1) 

(11.3.2)* "Rozviňte" si pro spor takové nakreslení do roviny, dostanete podle Eulerova vztahu triangulaci (nekonečnou) roviny, a odvoďte následně fakt, že směr rotace hran kolem vrcholů se v této triangulaci musí zachovávat. To je spor s neorientovatelností Kleinovy láhve. (Těžké k přečtení, že? Ale krátké.)

(11.3.3) Jenoznačně pro hodnoty 2 a všechny liché, pro ostatní sudé $0, -2, -4, \ldots$ nerozliší mezi příslušnou orientovatelnou a neorientovatelnou plochou.

(11.3.4) Samozřejmě, nakrelete si třeba K_4 i na torus. Vždy však lze definovat nejmenší a největší takovou plochu pro daný graf.

(11.3.5) Obsahuje dvě hranově disjunktní kopie K_5, jen jedna z nich může ke svému nakreslení "využít" crosscap, ta druhá by byla rovinná, spor.

(11.3.6) Z Eulerova vztahu pro projektivní rovinu získáme odhad počtu hran $h \leq 2v - 2$ pro grafy bez trojúhelníků, ale $15 > 16 - 2$, spor.

(11.3.7)* ...

(11.5.1) Není-li graf projektivní, obsahuje jeden ze zakázaných minorů (32 souvislých), přitom vlastnost míti rovinné pokrytí se jednoduše dědí na minory.

(11.5.2) Obdobná úvaha jako v 11.3.5.

(11.5.3)* Opět se počítá vhodně počet hran proti Eulerovu vztahu...

(11.5.4)* První je v podstatě stejná, ale řešení pro $K_{3,5}$ vyžaduje pomocné tvrzení, že vrcholy emulátoru zobrazované na vrcholy stupně 3 mohou být také upraveny na stupně 3.

Reference

[1] R. Diestel, Graph Theory (third edition), Springer, 2005.
Online http://www.math.uni-hamburg.de/home/diestel/books/graph.theory/

[2] J.L. Gross, J. Yellen, eds. Handbook of Graph Theory, CRC Press, 2004.

[3] L. Kučera, Kombinatorické Algoritmy, SNTL, Praha, 1983.

[4] M. Mareš, Krajinou grafových algoritmů, ITI MFF UK, 2007.
http://iti.mff.cuni.cz/series/files/iti330.pdf.

[5] J. Matoušek a J. Nešetřil, Kapitoly z Diskrétní Matematiky. čtvrté vydání, Karolinum, UK Praha, 2009.

[6] J. Matoušek and J. Nešetřil, Invitation to Discrete Mathematics, second edition. Oxford University Press, 2008.

[7] B. Mohar, C. Thomassen, Graphs on Surfaces. Johns Hopkins, 2001.

[8] J. Oxley, Matroid Theory, Oxford University Press, 1992.