## KMA/TGD1

# Teorie grafů a diskrétní optimalizace 1

### Pracovní texty přednášek

http://www.kma.zcu.cz/TGD1

Obsahem předmětu KMA/TGD1 jsou základy algoritmické teorie grafů a výpočetní složitosti. Předmět navazuje na předmět KMA/DMA „Diskrétní matematika“, zařazený do prvního ročníku bakalářského studia FAV, a znalosti v rozsahu základních skript [1] jsou podmínkou porozumění tomuto textu. Kapitoly 1 – 7 rozšiřují a prohlubují předchozí poznatky z předmětu „Diskrétní matematika“ a dávají orientaci v základních otázkách teorie grafů se zaměřením především na její algoritmickou stránku. Z hlediska celkové koncepce předmětu TGD1 tyto kapitoly vytvářejí aparát pro kapitoly 8 – 9.

Těžištěm předmětu jsou kapitoly 8 – 9, jejichž obsahem je úvod do teorie výpočetní složitosti a NP-úplnosti. Pochopení filosofie výpočetní složitosti, NP-úplnosti a vlastností problémů z tříd P, NP a NPC je hlavní dovedností, kterou by si student měl z tohoto předmětu odnést.

Na předmět KMA/TGD1 v letním semestru navazuje předmět KMA/TGD2, v němž se využijí znalosti pojmů z teorie grafů a výpočetní složitosti. Jeho obsahem je další prohloubení poznatků z teorie grafů se zaměřením především na optimalizační problémy na grafech a sítích a jejich algoritmické aspekty.

Tento text je pomocným textem k přednáškám, a v žádném případě si neklade ambice být považován za ucelený učební text typu učebnice či skript. Obsahuje sice všechny základní pojmy, tvrzení a jejich důkazy, ale chybí v něm mnohé komentáře, které jsou pro pochopení látky nezbytné. Na konci textu je uveden přehled základní literatury, která o teorii grafů a NP-úplnosti existuje v českém (a v jednom případě slovenském) jazyce. Z této literatury lze doporučit k doplňujícímu studiu zejména knihu [6], jíž je tento text blízký zejména v kapitolách 8 a 9.

Děkuji studentovi FAV (a nyní již kolegovi) Přemyslu Holubovi za přepis mých pracovních poznámek do TEXu, který se stal základem tohoto textu, a kolegovi Tomáši Kaiserovi, který je autorem některých pasáží kapitoly 2.

3. ledna 2007,

Z. Ryjáček

V této verzi textu bylo opraveno několik drobných chyb.

5. února 2011,

Z. Ryjáček

# Obsah

**1 Algoritmy a výpočetní složitost – intuitivní přístup** **3**

**2 Toky v sítích** **5**
2.1 Síť, existence toku v síti . . . 5
2.2 Síť s jedním zdrojem a jedním stokem . . . 7
2.3 Maximální tok . . . 8
2.4 Ford–Fulkersonův algoritmus . . . 9
2.5 Edmonds–Karpův algoritmus . . . 11
2.6 Dokončení důkazu věty 2.1 . . . 11

**3 Grafy a matice** **12**
3.1 Distanční matice ohodnoceného orientovaného grafu a Floydův algoritmus . . . 12
3.2 Rozložitelnost matic a struktura orientovaných grafů . . . 13
3.3 Slabě rozložitelné matice . . . 16
3.4 Regulární matice a perfektní párování v bigrafu . . . 19
3.5 Strukturální matice a generická hodnost . . . 20

**4 Míry souvislosti grafu** **23**
4.1 Mosty, artikulace, bloky grafu . . . 23
4.2 Hranový a uzlový stupeň souvislosti grafu . . . 25
4.3 Charakterizační věty $k$-souvislých grafů . . . 27

**5 Prohledávání grafů a algoritmy $k$-souvislosti** **29**
5.1 Algoritmus prohledávání grafu . . . 29
5.2 Použití backtrackingu . . . 32
5.3 Algoritmy zjišťování $k$-souvislosti . . . 33
5.4 Backtracking pro generování hamiltonovských cest a cyklů . . . 33
5.5 Heuristiky pro hledání hamiltonovské kružnice . . . 34

**6 Nezávislost, dominance, klikovost a jádro grafu** **36**
6.1 Neorientované grafy . . . 36
6.2 Jádro orientovaného grafu . . . 42

**7 Barevnost grafu** **44**
7.1 $k$-obarvitelnost a chromatické číslo grafu . . . 44
7.2 Barvení map . . . 47
7.3 Hranové barvení a rozklady grafů . . . 48

**8 Modely výpočtu** **51**
8.1 Počítač s libovolným přístupem . . . 51
8.2 Časová a paměťová náročnost výpočtu . . . 53
8.3 Problémy (jazyky) třídy P . . . 54

**9 Teorie NP-úplnosti** **55**
9.1 Logické formule . . . 55
9.2 Problém splnitelnosti logických formulí – SAT . . . 57
9.3 Problém $k$-SAT a polynomialita problému 2-SAT . . . 57
9.4 Problém existence nezávislé množiny uzlů dané velikosti – IND . . . 60
9.5 Třída NP . . . 62
9.6 Polynomiální redukce a NP-úplné problémy . . . 65
9.7 NP-úplnost problému SAT – Cookova věta . . . 66
9.8 Některé další NP-úplné problémy . . . 69

# 1 Algoritmy a výpočetní složitost – intuitivní přístup

Z předmětu KMA/DMA Diskrétní matematika známe řadu prakticky použitelných („efektivních“) algoritmů, umožňujících řešit některé problémy z teorie grafů: Dijkstrův algoritmus pro nalezení minimální cesty mezi dvěma uzly grafu, algoritmus číslování uzlů pro testování acykličnosti grafu a další. Poznali jsme i problémy, u nichž jsme konstatovali, že „efektivní“ algoritmus pro jejich řešení není znám - sem patří například problém rozhodnout o tom, zda daný neorientovaný graf je hamiltonovský. Aparát, který umožní postavit tuto klasifikaci problémů z hlediska „efektivnosti“ příslušných algoritmů na pevný základ, vybudujeme v (klíčových) kapitolách 8 – 9 tohoto předmětu. Zde si úvodem na intuitivní úrovni naznačíme otázky, které budou hlavním předmětem našeho zájmu. Veškeré úvahy v tomto odstavci jsou pouze motivačního charakteru a bez nároku na přesnost; pojmy, které zde zavedeme, získají přesný obsah až v kapitolách 8 – 9.

Algoritmy, které bývají považovány za rychlé, potřebují ke zpracování vstupních dat velikosti $n$ čas, který bývá zpravidla shora omezen funkcemi typu $n$, $n \log n$, $n^2$, $n^3$ a podobně, tj. časovou náročnost výpočtu je možno shora odhadnout polynomem. Naproti tomu algoritmy, které pracují metodou „hrubé síly“, tj. v podstatě probíráním všech možností, vyžadují alespoň $2^n$ kroků, pokud probíráme všechny podmnožiny dané $n$-prvkové množiny, $n!$ kroků, pokud probíráme všechny permutace dané $n$-prvkové množiny, nebo dokonce $n^n$ kroků, pokud probíráme všechna zobrazení dané $n$-prvkové množiny do sebe. Funkce, udávající počet kroků algoritmu jako funkci $n$ (velikosti vstupních dat) je tedy alespoň exponenciální.

Dramatický rozdíl mezi polynomiálními a exponenciálními algoritmy dobře ilustruje následující tabulka udávající čas, který je potřebný ke zpracování vstupních dat velikosti $n$, jestliže je nutno provést $f(n)$ operací a provedení jedné operace trvá jednu mikrosekundu.

| velikost vstupních dat $n$ | počet operací $f(n)$ | | | | |
|---|---|---|---|---|---|
| | $n^2$ | $n^3$ | $n^4$ | $2^n$ | $n!$ |
| 20 | 0,4 ms | 8 ms | 0,2 s | 1 s | 77 000 let |
| 40 | 1,6 ms | 64 ms | 2,6 s | 12 dní | — |
| 60 | 3,6 ms | 0,2 s | 13 s | 36 600 let | — |
| 80 | 6,4 ms | 0,5 s | 41 s | $3,6 \cdot 10^9$ let | — |
| 100 | 10 ms | 1 s | 100 s | — | — |
| 200 | 40 ms | 8 s | 27 min | — | — |
| 500 | 0,25 s | 125 s | 17 hod | — | — |
| 1000 | 1 s | 17 min | 12 dní | — | — |

Ještě lépe je tento rozdíl patrný z následující tabulky. Vycházíme v ní z předpokladu, že jsme schopni daným algoritmem s časovou náročností $f(n)$ zpracovat v daném časovém limitu vstupní data velikosti $n = 100$ a ptáme se, jak se zvětší velikost úloh, které jsme schopni zpracovat ve stejném časovém limitu, jestliže zvýšíme rychlost výpočtu 10×, 100×, 1000×.

| zrychlení výpočtu | počet operací $f(n)$ | | | | |
|---|---|---|---|---|---|
| | $n^2$ | $n^3$ | $n^4$ | $2^n$ | $n!$ |
| 1× | 100 | 100 | 100 | 100 | 100 |
| 10× | 316 | 215 | 177 | 103 | 100 |
| 100× | 1000 | 464 | 316 | 106 | 100 |
| 1000× | 3162 | 1000 | 562 | 109 | 101 |

Názorně zde vidíme, že pro exponenciální algoritmy je typické, že nad jistou mezní velikostí vstupních dat narážíme na bariéru, nad níž ani zvýšení rychlosti výpočtu o několik řádů nemůže znatelně zvětšit velikost zpracovatelných úloh.

Bylo by jistě možno namítat, že algoritmy s časovou náročností řádu $n^{100}$, $2^{100}n$ a podobně jsou polynomiální, a přesto budou v praxi nepoužitelné, zkušenost ale ukazuje, že podaří-li se u některé úlohy nalézt polynomiální algoritmus, pak se zpravidla dalším vývojem podaří snížit stupeň i koeficienty polynomu na „rozumné“ hodnoty. Jedinou dosud známou výjimkou je úloha (reálného) lineárního programování, kde v praxi běžně používaný simplexový algoritmus (poznáme jej v navazujícím předmětu KMA/TGD2) má v nejhorším případě exponenciální časovou náročnost, ale přesto je v „běžných situacích“ výhodnější než nedávno nalezený polynomiální (Chačijanův) algoritmus.

Výpočetní složitost daného algoritmu budeme tedy měřit pomocí funkce $f(n)$, která udává časovou náročnost výpočtu (počet kroků algoritmu) jako funkci velikosti vstupních dat. Algoritmus budeme považovat za *efektivní*, jestliže tuto funkci $f(n)$ lze shora odhadnout polynomem $p(n)$. Pokud takový odhad neexistuje, budeme algoritmus považovat za *neefektivní*.

Problémy, řešitelné polynomiálními (efektivními) algoritmy budeme nazývat *problémy třídy P*. Do třídy P patří většina problémů, které jsme dosud poznali: minimální kostra, minimální a kritická cesta, acykličnost grafu, distanční matice. Všimneme si zde toho, že algoritmy, které řeší tyto úlohy, pracují vesměs *deterministicky*, tj. v každém kroku je jednoznačně určen krok následující.

Na druhé straně, u řady úloh prostě z principiálních důvodů polynomiální algoritmus získat nelze. Jako příklad stačí uvést úlohu generovat všechny kostry daného grafu – víme, že neorientovaný graf na $n$ uzlech může mít až $n^{n-2}$ různých koster.

Mezi těmito skupinami je třetí skupina úloh, pro které dosud není nalezen efektivní algoritmus, ale také není dokázána jeho neexistence. Pro tyto úlohy je charakteristické, že podaří-li se nám „náhodou“ nalézt řešení, pak je možné v polynomiálním čase prověřit jeho správnost. K nalezení tohoto řešení jsou ovšem známy – při zachování požadavku polynomiality – pouze algoritmy *nedeterministické*, které pracují s náhodným generováním (jakožto modelem „uhádnutí“ hledaného řešení). Je zde ovšem možnost nalezení řešení deterministicky hrubou silou (probíráním všech možností), počet kroků takových algoritmů je ale nejméně exponenciální. Stejnou náročnost má zodpovězení otázky existence řešení v případě, kdy odpověď je záporná. Typickým představitelem úlohy z této třídy je úloha rozhodnout, zda v daném grafu existuje hamiltonovská kružnice: podaří-li se nám hamiltonovskou kružnici „uhádnout“, pak se v polynomiálním čase přesvědčíme o správnosti řešení – pro deterministické nalezení hledané kružnice nebo ověření její neexistence jsou ale známé jen neefektivní postupy typu hrubá síla. Další podobné úlohy poznáme v kapitolách 6 a 7.

Třída úloh, řešitelných nederministicky v polynomiálním čase, se nazývá *třída NP* (nedeterministicky polynomiální) a obecně platí $\mathrm{P} \subset \mathrm{NP}$. Není ale dosud známo, jestli je $\mathrm{P} = \mathrm{NP}$ nebo jestli je $\mathrm{P} \neq \mathrm{NP}$. A co více: ve třídě NP existuje celá řada úloh, o nichž je dokázáno, že kdyby se podařilo najít polynomiální algoritmus pro některou z nich, pak z tohoto algoritmu lze odvodit polynomiální algoritmy i pro všechny ostatní problémy této třídy. Úlohy této podskupiny jsou tedy co do efektivnosti řešení ekvivalentní - buďto jsou všechny řešitelné v polynomiálním čase, nebo neexistuje efektivní algoritmus pro žádnou z nich. Dosud ale není známo, která z těchto možností nastává. Tyto úlohy se nazývají *NP-úplné* a patří mezi ně například zmíněný problém existence hamiltonovské kružnice. Mezi NP-úplné úlohy patří mnohé důležité úlohy z nejrůznějších oblastí diskrétní matematiky, teoretické informatiky, z operační alalýzy, teorie kódování, z teorie čísel a dalších oblastí. Problém efektivní řešitelnosti těchto úloh, tj. otázka, zda platí $\mathrm{P} = \mathrm{NP}$ nebo $\mathrm{P} \neq \mathrm{NP}$, zvaný *NP-problém*, patří mezi nejdůležitější otevřené problémy moderní matematiky, a přes velké úsilí zůstává stále nevyřešen. Všeobecně se ale předpokládá, že tyto třídy rovny nejsou.

Poznamenejme, že to, co bylo dosud řečeno, se týká rozhodovacích úloh, tj. úloh, u nichž výstupem je odpověď ANO či NE na danou otázku. U optimalizačních úloh (najděte největší, nejmenší, nejlepší atd. objekt z dané množiny) je nutno úlohu nejprve převést na rozhodovací úlohu, která je z hlediska existence efektivního algoritmu s danou úlohou ekvivalentní. Konkrétní příklady poznáme ve druhé části tohoto textu.

# 2 Toky v sítích

Teorie toků v sítích je motivována úlohami z kategorie tzv. dopravních problémů, v nichž je cílem optimalizovat přepravu nějakého produktu v transportní síti, případně optimalizovat strukturu této sítě. Jedná se o úlohu s bohatými aplikacemi, a to jak teoretického charakteru v teorii grafů (mnohé grafové úlohy, v nichž „nikde nic neteče“ lze řešit převodem na toky), tak i při praktickém řešení optimalizačních problémů.

V tomto předmětu se omezíme na otázku existence toku v obecné síti a na problém maximálního toku v síti s jedním zdrojem a jedním stokem. Obecnou optimalizační úlohu, kdy je navíc u každé hrany zadána cena a minimalizujeme celkovou cenu toku, poznáme v navazujícím předmětu KMA/TGD2.

## 2.1 Síť, existence toku v síti

**Definice 2.1.** Síť *je orientovaný graf* $\vec{G}$ *s ohodnocením hran* $r : H(\vec{G}) \longrightarrow (0, \infty)$ *a ohodnocením uzlů* $a : U(\vec{G}) \longrightarrow R$.

Síť je tedy orientovaný graf s *kladným reálným* ohodnocením hran a s *reálným* (připouštíme i záporné hodnoty) ohodnocením uzlů.

V této kapitole budeme uzly grafu $\vec{G}$ považovat za očíslované čísly $1, \ldots, n$ (kde $n = |V(\vec{G})|$). Ohodnocení $a(i)$ uzlu $i \in U(\vec{G})$ budeme krátce značit $a_i$, a obdobně ohodnocení $r((i,j))$ hrany $(i,j) \in E(\vec{G})$ budeme krátce značit $r_{ij}$ $(i, j = 1, \ldots, n)$. Obdobný je i význam čísel $x_{ij}$ v následující definici.

**Definice 2.2.** *Buď* $\vec{G}$ *síť s ohodnocením uzlů* $a_i$ *a s ohodnocením hran* $r_{ij}$. Tok *v síti* $\vec{G}$ *je nezáporné hranové ohodnocení* $x : H(\vec{G}) \longrightarrow \langle 0, \infty)$, *splňující následující podmínky:*

1. *pro každý uzel* $i \in U(\vec{G})$ *platí*

$$\sum_{j;(i,j)\in H(\vec{G})} x_{ij} \; - \sum_{j;(j,i)\in H(\vec{G})} x_{ji} = a_i \ ,$$

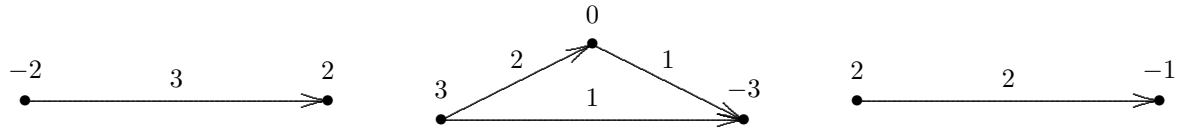
2. *pro každou hranu* $(i,j) \in H(\vec{G})$ *platí*

$$0 \leq x_{ij} \leq r_{ij} \ .$$

**Příklad.** Mějme jistou množinu měst, očíslovaných čísly $1, \ldots, n$, a tato města jsou spojena železniční sítí. Sestrojíme graf $\vec{G}$ na $n$ uzlech tak, že $(i,j) \in H(\vec{G})$ právě tehdy, jestliže $i$-té město je bezprostředně spojeno železnicí s $j$-tým městem (předpokládáme-li obousměrnost tratí, dostaneme vždy dvojici hran $(i,j)$ a $(j,i)$). Ve zmíněných městech se vyrábí a spotřebovává jistý produkt. Velikost výroby v $i$-tém městě za jednotku času označme $b_i$, velikost spotřeby za jednotku času označme $c_i$ a položme $a_i = b_i - c_i$, $i = 1, \ldots, n$. Označme $r_{ij}$ celkové množství produktu, které lze převézt po trati $(i,j)$ za jednotku času (obecně může být $r_{ij} \neq r_{ji}$). Jestliže čísla $x_{ij}$ mají význam skutečného množství produktu, které se za jednotku času přepraví po trati $(i,j)$, tj. z $i$-tého města do $j$-tého města, pak evidentně pro každou trať $(i,j) \in H(\vec{G})$ bude platit $0 \leq x_{ij} \leq r_{ij}$, což je podmínka 2 z definice 2.2. V každém uzlu $i$ přitom musí platit „zákon zachování“, podle něhož rozdíl celkového množství produktu vyvezeného („vytékajícího“) z uzlu $i$ po všech hranách $(i,j) \in H(\vec{G})$ a celkového množství produktu přivezeného („vtékajícího“) do uzlu $i$ po všech hranách $(j,i) \in H(\vec{G})$ musí být roven množství produktu, jež se v $i$-tém uzlu „nedostává“ (při $a_i < 0$), resp. jež v $i$-tém uzlu „přebývá“ (při $a_i > 0$), což je podmínka 1 z definice 2.2.

**Definice 2.3.** *Je-li $i \in U(\vec{G})$, pak se číslo $a_i$ nazývá* intenzita *uzlu $i$; pro $(i,j) \in H(\vec{G})$ se číslo $r_{ij}$ nazývá* propustnost *hrany $(i,j)$. Je-li $a_i > 0$, pak se uzel $i$ nazývá* zdroj, *při $a_i < 0$ se uzel $i$ nazývá* stok. *Je-li $a_i = 0$, budeme říkat, že uzel $i$ je* neutrální uzel.

**Příklad.**




Jak se snadno přesvědčíme, v žádné ze sítí na obrázku neexistuje tok (zdůvodněte si proč). Vidíme tedy, že v dané síti obecně nemusí tok existovat. Naším prvním úkolem bude vyjasnění podmínek, za nichž v dané síti existuje (alespoň jeden) tok.

Nejprve ale zavedeme některé potřebné pojmy a označení.

**Definice 2.4.** *Nechť $\vec{G}$ je síť, $A \subset U(\vec{G})$ je množina uzlů, a položme $\bar{A} = U(\vec{G}) \setminus A$. Množina hran*

$$(A, \bar{A}) = \{(x,y) \mid x \in A, y \in \bar{A}\}$$

*se nazývá* řez *sítě $\vec{G}$.*

**Poznámka.** Čtenář se snadno přesvědčí, že naše definice řezu je orientovanou analogií pojmu separace, známého z předmětu KMA/DMA (viz kap. 10.4 skript „Diskrétní matematika“). Používáme zde termín „řez“ (který byl vyhrazen pro minimální separace), protože je to v této souvislosti v literatuře obvyklejší. Uvidíme později, že v důležitých situacích budou řezy, které nás budou zajímat, splňovat podmínku minimality.

Nechť $\vec{G}$ je síť a $A \subset U(\vec{G})$ je množina jejích uzlů. Zavedeme následující označení:

je-li $f : U(\vec{G}) \longrightarrow R$ funkce na $U(\vec{G})$, označíme $f(A) = \sum_{i \in A} f_i$,

je-li $g : H(\vec{G}) \longrightarrow R$ funkce na $H(\vec{G})$, označíme $g(A, \bar{A}) = \sum_{(i,j) \in (A, \bar{A})} g_{ij}$.

**Tvrzení 2.1.** *Nechť $\vec{G}$ je síť, $x$ je tok v $\vec{G}$ a nechť $A \subset U(\vec{G})$ je množina uzlů $\vec{G}$. Pak platí*

$$a(A) = x(A, \bar{A}) - x(\bar{A}, A).$$

**Důkaz.** Podle definice toku (definice 2.2) pro množinu $A$ platí:

$$a(A) = \sum_{i \in A} a_i = \sum_{i \in A} \Big( \sum_{j;(i,j) \in H(\vec{G})} x_{ij} - \sum_{j;(j,i) \in H(\vec{G})} x_{ji} \Big) = \sum_{i \in A} \sum_{j;(i,j) \in H(\vec{G})} x_{ij} - \sum_{i \in A} \sum_{j;(j,i) \in H(\vec{G})} x_{ji}.$$

V tomto výrazu je první člen roven celkovém součtu hodnot toků, vytékajících ze všech uzlů množiny $A$ – z těchto toků některé směřují do jiných uzlů z $A$, některé do uzlů z $\bar{A}$; první výraz je tedy roven $x(A,A) + x(A,\bar{A})$. Obdobně druhý výraz je roven celkovém součtu hodnot toků, vtékajících do všech uzlů množiny $A$, z nichž některé přicházejí z uzlů z $A$, jiné z uzlů z $\bar{A}$, takže je roven $x(A,A) + x(\bar{A},A)$. Odtud dostáváme

$$a(A) = (x(A,A) + x(A,\bar{A})) - (x(A,A) + x(\bar{A},A)),$$

odkud

$$a(A) = x(A, \bar{A})) - x(\bar{A}, A).$$

□

**Věta 2.1.** *V síti existuje tok právě když* $a(U(\vec{G})) = 0$ *a pro každou množinu uzlů* $A \subset U(\vec{G})$ *je* $a(A) \leq r(A, \bar{A})$.

**Důkaz.** 1. Dokážeme, že podmínky věty jsou pro existenci toku nutné. Předpokládejme tedy, že v síti $\vec{G}$ existuje tok. Protože pro každou hranu $(i, j) \in H(\vec{G})$ platí $0 \leq x_{ij} \leq r_{ij}$, je $x(A, \bar{A}) \leq r(A, \bar{A})$ a $x(\bar{A}, A) \geq 0$. Z tvrzení 2.1 tedy vyplývá, že $a(A) \leq r(A, \bar{A})$. Speciálně pro $A = U(\vec{G})$ je $\bar{A} = \emptyset$, takže $a(U(\vec{G})) = 0$.

2. Tvrzení o tom, že podmínky věty jsou postačující, dokážeme později (v kapitole 2.6) po vyšetření jednoho důležitého speciálního případu. □

## 2.2 Síť s jedním zdrojem a jedním stokem

Jako speciální případ uvažujme síť $\vec{G}$ s jedním zdrojem $z$ a jedním stokem $s$; všechny ostatní uzly sítě $\vec{G}$ nechť jsou neutrální. Je-li intenzita zdroje $z$ rovna číslu $a \geq 0$, pak nutně musí mít stok intenzitu $-a$ (jinak by v síti $\vec{G}$ nemohl podle věty 2.1 existovat žádný tok). Jestliže v naší síti existuje nějaký tok $x$, pak příslušná hodnota intenzity zdroje vyjadřuje „množství produktu“, přepraveného po síti ze zdroje $z$ do stoku $s$. Toto číslo, tj. příslušná intenzita zdroje, se nazývá *velikost toku* $x$ a budeme ji značit $|x|$.

Položíme-li $a = 0$ a $x_{ij} = 0$ pro všechna $i, j$, pro něž $(i, j) \in H(\vec{G})$, pak zřejmě funkce $x$ splňuje podmínky definice 2.2 a tedy je tokem. Krátce řečeno - v každé síti existuje alespoň jeden tok, a sice tok nulové velikosti. Má tedy smysl ptát se na největší možnou velikost toku v dané síti.

**Definice 2.5.** *Nechť* $\vec{G}$ *je síť s jedním zdrojem* $z$ *a jedním stokem* $s$, *a nechť* $x$ *je tok v* $\vec{G}$. *Řekneme, že tok* $x$ *je* maximální tok v $\vec{G}$, *jestliže pro každý tok* $x'$ *v* $\vec{G}$ *platí*

$$|x'| \leq |x|.$$

**Definice 2.6.** *Nechť* $\vec{G}$ *je síť s jedním zdrojem* $z$ *a jedním stokem* $s$, *a nechť* $(A, \bar{A})$ *je řez sítě* $\vec{G}$.
*Číslo* $r(A, \bar{A})$ *se nazývá* propustnost řezu $(A, \bar{A})$.
*Řekneme, že řez* $(A, \bar{A})$ *je* minimální řez sítě $\vec{G}$, *jestliže pro každý řez* $(A', \bar{A}')$ *sítě* $\vec{G}$ *platí*

$$r(A, \bar{A}) \leq r(A', \bar{A}').$$

*Jsou-li* $u, v \in U(\vec{G})$ *dva uzly* $\vec{G}$, *pak řekneme, že řez* $(A, \bar{A})$ odděluje uzly $u$, $v$, *jestliže* $u \in A$ *a* $v \in \bar{A}$.

**Tvrzení 2.2.** *Nechť* $\vec{G}$ *je síť s jedním zdrojem* $z$ *a jedním stokem* $s$, *nechť* $(A, \bar{A})$ *je řez sítě* $\vec{G}$, *oddělující* $z$ *a* $s$, *a nechť* $x$ *je tok v* $\vec{G}$. *Pak platí:*
(*i*) $|x| = x(A, \bar{A}) - x(\bar{A}, A)$,
(*ii*) $|x| \leq r(A, \bar{A})$.

**Důkaz.** Tvrzení (*i*) je přímým důsledkem tvrzení 2.1, (*ii*) plyne ihned z tvrzení (*i*) a z vlastnosti 2 z definice toku. □

## 2.3 Maximální tok

Nás v této kapitole zajímá především maximální tok, jeho vlastnosti a možné postupy jeho hledání. Jeden pokus o takový postup by mohl vypadat například tak, že začneme s nulovým tokem a postupně jej modifikujeme podle následujícího pravidla: existuje-li orientovaná cesta $P$ ze $z$ do $s$, jejíž žádnou hranou zatím 'neteče' celý povolený objem (propustnost této hrany), pak přičteme k hodnotám toku na hranách cesty $P$ maximální možné číslo $c$, pro které nebude překročena propustnost žádné z hran. Protože přičítaná velikost je pro všechny hrany cesty stejná, snadno ověříme, že dostaneme opět tok. Jeho velikost bude navíc větší než velikost původního toku. Uvedenou úpravu opakujeme, dokud lze najít orientované cesty s požadovanou vlastností. Pokud taková cesta neexistuje, náš tok je jistě maximální — nebo ne?

Ne. Příklad, který ukazuje, že tato metoda je příliš naivní, je uveden na obrázku. Dejme tomu, že jsme v síti na obrázku (a) v prvním kroku našli tok, znázorněný tučně na obr. (b). Náš algoritmus skončil s tokem o velikosti 1, hledaná orientovaná cesta $P$ neexistuje. Maximální tok v síti $\vec{G}$ má však zjevně velikost 2.


Lze namítnout, že problém byl způsoben nevhodnou volbou cesty $P$ v prvním kroku. To je pravda, dá se totiž dokázat, že pro každou síť existuje posloupnost 'vhodných' výběrů cesty $P$, pro kterou nám skutečně vyjde maximální tok. Potíž je v tom, že nemáme po ruce žádný návod, jak poznat vhodný výběr od nevhodného. Proto nám nezbude než tento postup definitivně zavrhnout. V příštím oddílu si ukážeme tzv. Ford–Fulkersonův algoritmus, který danou úlohu řeší uspokojivě.

Zavedeme nyní několik pojmů, na kterých je tento algoritmus založen, a využijeme je v důkazu velmi důležité věty, Ford–Fulkersonovy věty o maximálním toku.

**Definice 2.7.** *Nechť $u, w \in U(\vec{G})$.* Polocesta *z $u$ do $w$ je posloupnost $u = v_0, h_1, v_1, h_2, \ldots, h_k, v_k = w$, kde $v_i$ jsou navzájem různé uzly, $h_i$ jsou hrany a pro každé $i = 1, \ldots, k$ platí buď $h_i = v_{i-1}v_i$ (pak jde o* souhlasnou *hranu dané polocesty) nebo $h_i = v_iv_{i-1}$* nesouhlasná *hrana).*

*Nechť $x$ je tok v síti $\vec{G}$.* Rezerva *polocesty $P$ je nezáporné číslo*

$$\Theta(P) = \min\Big(\{r_{ij} - x_{ij} |\ (i,j) \text{ je souhlasná hrana } P\} \cup \{x_{ij} |\ (i,j) \text{ je nesouhlasná hrana } P\}\Big).$$

*Polocesta $P$ je* rezervní, *jestliže $\Theta(P) > 0$.*

Řekneme-li, že hrana $(i,j)$ je *nasycená*, je-li $x_{ij} = r_{ij}$, a že je *nulová*, pokud $x_{ij} = 0$, pak polocesta je rezervní právě když neobsahuje žádnou souhlasnou nasycenou ani nesouhlasnou nulovou hranu.

Na význam této definice ukazuje následující tvrzení.

**Tvrzení 2.3.** *Nechť $\vec{G}$ je síť s jedním zdrojem $z$ a jedním stokem $s$, a nechť $x$ je tok v $\vec{G}$. Existuje-li v $\vec{G}$ rezervní polocesta ze $z$ do $s$ vzhledem k $x$, pak tok $x$ není maximální.*

**Důkaz.** Nechť $\Theta > 0$ je rezerva polocesty $P$ ze $z$ do $s$. Definujeme v $\vec{G}$ nový tok $x'$ předpisem:

$$x'_{ij} = \begin{cases} x_{ij} + \Theta & \text{je-li } (i,j) \text{ souhlasná hrana } P, \\ x_{ij} - \Theta & \text{je-li } (i,j) \text{ nesouhlasná hrana } P, \\ x_{ij} & \text{jinak.} \end{cases}$$

Snadno ověříme, že $x'$ je opět tok. Protože však $|x'| = |x| + \Theta$, tok $x$ není maximální. □

Nyní můžeme dokázat nejdůležitější tvrzení této kapitoly.

**Věta 2.2.** **[Ford, Fulkerson]** *Buď $\vec{G}$ síť s jedním zdrojem $z$ a jedním stokem $s$. Velikost maximálního toku v $\vec{G}$ je rovna propustnosti minimálního řezu, oddělujícího $z$ a $s$.*

**Důkaz.** Nechť $x$ je maximální tok v $\vec{G}$. Označme $R$ množinu všech uzlů $w$, pro které existuje rezervní polocesta ze $z$ do $w$. Jistě $z \in R$ (díky „prázdné“ polocestě) a $s \notin R$ (protože jinak by podle tvrzení 2.3 nemohl tok $x$ být maximální).

Každá hrana $(i,j) \in (R, \bar{R})$ je nutně nasycená, protože jinak bychom museli do množiny $M$ přidat její koncový uzel. Podobně každá hrana z $(\bar{R}, R)$ je nulová. Platí tedy $x(R,\bar{R}) = r(R,\bar{R})$ a $x(\bar{R},R) = 0$. Podle tvrzení 2.2 $(i)$ je

$$|x| = x(R,\bar{R}) - x(\bar{R},R) = r(R,\bar{R}).$$

Našli jsme tedy řez $(R,\bar{R})$, který odděluje $z$ a $s$ a jeho propustnost je rovna velikosti toku $x$. Podle tvrzení 2.2 $(ii)$ musí jít o minimální řez. □

Všimněme si, že jsme vlastně dokázali opačnou implikaci k tvrzení 2.3: neexistuje-li žádná rezervní polocesta ze zdroje do stoku, potom množina $R$ (definovaná jako v důkazu věty 2.2) neobsahuje stok a tedy $|x| = r(R,\bar{R})$. Podle tvrzení 2.2 $(ii)$ je $x$ nutně maximální tok.

## 2.4 Ford–Fulkersonův algoritmus

Myšlenka důkazu věty 2.2 ukazuje metodu, která umožňuje efektivně ověřit, zda je daný tok $x$ maximální. Stačí zkonstruovat množinu $R$ všech uzlů, které jsou dosažitelné ze zdroje po rezervní polocestě. Pokud $s \in R$, pak náš tok $x$ není maximální a nalezená polocesta ze $z$ do $s$ umožňuje přejít k toku o větší velikosti. Pokud na druhou stranu $s \notin R$, pak jsme našli řez $(R,\bar{R})$ s propustností $|x|$. Podle tvrzení 2.2 $(ii)$ je tento řez důkazem, že $x$ je maximální tok.

Na myšlence rezervní polocesty je založen nejjednodušší algoritmus pro hledání maximálního toku, tzv. *Ford–Fulkersonův algoritmus.*

**Algoritmus 2.1.** **(Ford–Fulkersonův algoritmus)**

1. *Jako výchozí tok $x$ zvolme nulový tok: $x_{ij} := 0$ pro každou hranu $(i,j) \in H(\vec{G})$.*
2. *Jestliže v grafu $\vec{G}$ existuje nějaká rezervní polocesta $P$ ze $z$ do $s$, upravme podle ní tok $x$:*

$$x_{ij} := \begin{cases} x_{ij} + \Theta & \text{pokud } (i,j) \text{ je souhlasná hrana polocesty } P, \\ x_{ij} - \Theta & \text{pokud } (i,j) \text{ je nesouhlasná hrana polocesty } P, \\ x_{ij} & \text{pokud } (i,j) \text{ neleží na } P, \end{cases}$$

   *a pokračujme bodem (2).*

3. *V případě, že rezervní polocesta ze $z$ do $s$ neexistuje, je tok $x$ maximální.*

Způsob hledání rezervní polocesty není v tomto algoritmu specifikován — můžeme prostě vzít první, kterou najdeme.

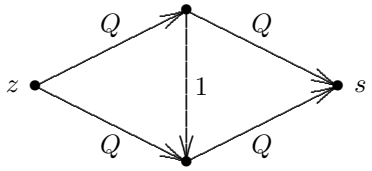
Aby byl tento postup použitelný, je nutné zaručit, že algoritmus po konečném počtu kroků skončí. V síti s celočíselnými propustnostmi hran to není tak těžké.

**Tvrzení 2.4.** *Jsou-li v síti $\vec{G}$ propustnosti všech hran celá čísla, pak Ford–Fulkersonův algoritmus skončí po konečném počtu kroků.*

**Důkaz.** Především si všimněme, že po provedení každého kroku algoritmu získáme tok, jehož hodnota na každé hraně je celočíselná. Výchozí nulový tok totiž tuto vlastnost má, a díky celočíselnosti propustností je celé i číslo $\Theta$, o které měníme hodnoty toku. Velikost toku v každém kroku vzroste právě o číslo $\Theta$, které je větší nebo rovno jedné. Velikost maximálního toku je přitom shora omezena propustností minimálního řezu. Počet kroků je tedy konečný. □

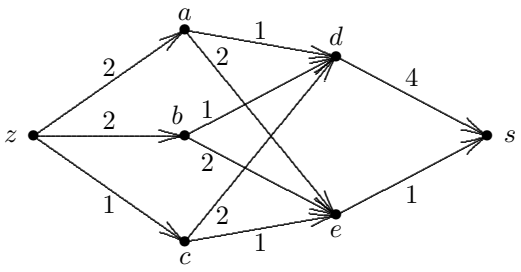
Dá se ukázat, že algoritmus je konečný i v případě, že propustnosti jsou racionální čísla. Pro obecné reálné propustnosti však (poněkud překvapivě) existují případy sítí, kdy Ford–Fulkersonův algoritmus konečný není. My se ve zbytku této kapitoly omezíme na sítě s celočíselnými propustnostmi hran.

Jinou slabinou našeho algoritmu je, že doba jeho provádění závisí nejen na počtu uzlů a hran sítě $\vec{G}$ (což je přirozené), ale také na propustnostech hran. Uvažme síť na následujícím obrázku, ve které $Q$ je nějaké velké číslo.

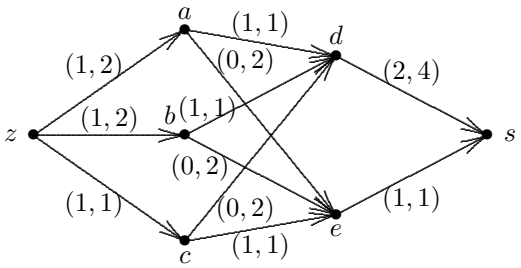



Pokud v každém kroku algoritmu nešťastnou náhodou použijeme rezervní polocestu, která prochází svislou hranou, změní se velikost toku vždy jen o 1, takže budeme potřebovat $2Q$ kroků.

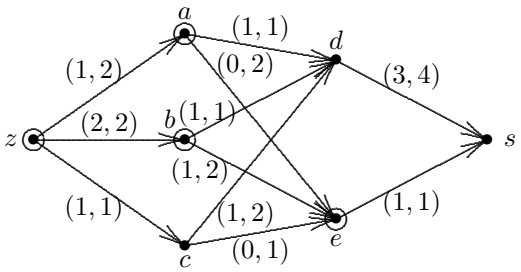
**Příklad.** Najděme maximální tok v síti na následujícím obrázku.




Počáteční nulový tok snadno zvětšíme podél několika orientovaných cest ze $z$ do $s$. Dejme tomu, že jsme přidali hodnotu 1 na cestách $zads$, $zces$ a $zbds$ a dostali jsme tok velikosti 3 na následujícím obrázku (zde dvojice čísel u hrany $(i,j)$ má význam $(x_{ij}, r_{ij})$).




Nyní už žádná orientovaná cesta ze $z$ do $s$ s nenulovou rezervou neexistuje, a mohli bychom se domnívat, že jsme našli maximální tok. Pokud ale sestrojíme množinu $R$ jako v důkazu Ford–Fulkersonovy věty, zjistíme, že $s \in R$, a to kvůli polocestě $zbecds$ s rezervou 1. Úpravou toku podél této polocesty dostaneme následující tok velikosti 4.



Provedeme-li opět kontrolu maximality (konstrukci množiny $R$), zjistíme, že tentokrát $s \notin R$, takže tok je již skutečně maximální. Uzly množiny $R$ jsou označeny kroužky, minimální řez je $(R, \bar{R}) = \{(a,d),(b,d),(z,c),(e,s)\}$.

## 2.5 Edmonds–Karpův algoritmus

Nesnáz se závislostí časové náročnosti algoritmu na propustnostech řeší modifikace Ford–Fulkersonova algoritmu, známá jako *Edmonds–Karpův algoritmus*. Myšlenka je jednoduchá: zvolíme vždy nejkratší rezervní polocestu ze $z$ do $s$ (tj. takovou, která má minimální počet hran). Dá se ukázat (my to dělat nebudeme), že v takovém případě je počet opakování kroku (2) algoritmu nezávislý na propustnostech a činí v nejhorším případě zhruba $m^2 n$, kde $m = |H(\vec{G})|$ a $n = |U(\vec{G})|$.

Jak ale zmíněnou nejkratší rezervní polocestu najít? Například tzv. *prohledáváním do šířky*. Prohledávání grafů je postup, který je základem mnoha efektivních algoritmů a podrobněji se jím budeme zabývat v kapitole 5. Nyní si jen popíšeme jednu z iterací kroku (2).

Dejme tomu, že máme nějaký tok $x$ a hledáme nejkratší rezervní polocestu ze $z$ do $s$.

(a) Nechť $T$ je strom na jediném uzlu $z$. Dále mějme seznam uzlů $L$ s jedinou položkou $z$. Všechny uzly sítě $\vec{G}$ kromě zdroje jsou na začátku *neoznačené*, zdroj je *označený*.

(b) Je-li $L \neq \emptyset$, pak nechť $v$ je první uzel seznamu $L$.

- Je-li $v = s$, algoritmus končí. Jednoznačně určená polocesta spojující $z$ a $s$ ve stromu $T$ je hledaná nejkratší polocesta. Upravíme podél této polocesty tok $x$ jako ve Ford–Fulkersonově algoritmu.
- Jinak označíme všechny neoznačené sousedy $w$ uzlu $v$, pro něž $vw$ je nenasycená hrana nebo $wv$ je nenulová hrana, ve stromu $T$ je připojíme hranami k $v$, a přidáme je na konec seznamu $L$. Vyřadíme uzel $v$ ze seznamu $L$ a pokračujeme bodem (b).

(c) V případě $L = \emptyset$ rezervní polocesta spojující $z$ a $s$ neexistuje a tok $x$ je tedy maximální.

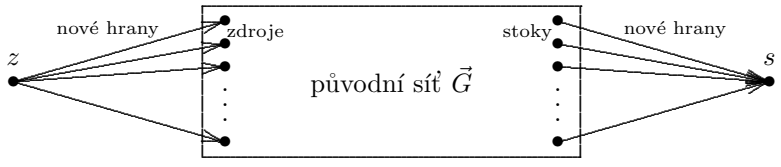
## 2.6 Dokončení důkazu věty 2.1

Vrátíme se nyní k obecnému případu sítě s více zdroji a více stoky a dokážeme tvrzení o tom, že podmínky věty 2.1 jsou postačující pro existenci toku. Tvrzení dokážeme tak, že hledaný tok sestrojíme převodem na větu 2.2.

Nechť tedy v síti $\vec{G}$ je $a(U(\vec{G})) = 0$ a pro každý řez $(A, \bar{A})$ je $a(A) \leq r(A, \bar{A})$. Sestrojíme novou síť $\vec{G}'$ s jedním zdrojem a jedním stokem tak, že ke $\vec{G}$ přidáme nový „fiktivní“ zdroj $z$ a nový „fiktivní“ stok $s$ a spojíme je s uzly sítě $\vec{G}$ podle následujících pravidel:

- je-li $a_i > 0$, pak přidej hranu $(z, i)$ s propustností $a_i$ a uzlu $i$ dej novou intenzitu $a_i = 0$,
- je-li $a_i < 0$, pak přidej hranu $(i, s)$ s propustností $-a_i$ a uzlu $i$ dej novou intenzitu $a_i = 0$,
- je-li $a_i = 0$, pak nedělej nic,

$i = 1, \ldots, |U(\vec{G})|$ (viz obrázek).




Z předpokladu $a(A) \leq r(A, \bar{A})$ vyplývá, že řez $(\{z\}, \overline{\{z\}})$ je minimálním řezem, oddělujícím $z$ a $s$. Podle Ford-Fulkersonovy věty 2.2 tedy v síti $\vec{G}'$ existuje tok, který nasycuje hrany řezu $(\{z\}, \overline{\{z\}})$. Je však zřejmé, že restrikce takto sestrojeného toku na původní síť $\vec{G}$ je hledaný tok v síti $\vec{G}$.

# 3 Grafy a matice

V předmětu KMA/DMA Diskrétní matematika jsme poznali základní způsoby popisu grafu a jeho vlastností pomocí matic (incidenční matice, matice sousednosti, matice vzdáleností). V tomto odstavci si ukážeme některé další souvislosti mezi grafovými pojmy a pojmy z lineární algebry.

## 3.1 Distanční matice ohodnoceného orientovaného grafu a Floydův algoritmus

Připomeňme si následující definici (viz [1], kap. 12.3, nebo [4], kap. 5.4).

**Definice 3.1.** *Nechť $\vec{G}$ je orientovaný ohodnocený graf s množinou uzlů $U(\vec{G}) = \{v_1, \dots, v_n\}$ a s hranovým ohodnocením $w : H(\vec{G}) \to (0, \infty)$. Matice $\mathbf{D}^w(\vec{G}) = [d^w_{i,j}]^n_{i,j=1}$, definovaná předpisem*

$$d^w_{i,j} = d^w(v_i, v_j), \qquad i, j = 1, \dots, n$$

*se nazývá* w-distanční matice grafu $\vec{G}$. [1]

Jednu metodu výpočtu matice $\mathbf{D}^w(\vec{G})$ (pomocí upraveného násobení matic) již známe - viz tvrzení 12.6 v textu [1]. Výpočetní složitost algoritmu založeného na této metodě je $O(n^4)$.

Jedním z nejlepších známých postupů pro řešení této úlohy je *Floydův algoritmus*, jehož výpočetní složitost je pouze $O(n^3)$.

**Algoritmus 3.1. (Floydův algoritmus)**

1. *Polož* $\mathbf{D}_0 = [d^0_{i,j}]^n_{i,j=1}$, *kde*

$$d^0_{ij} = \begin{cases} 0 & \text{pro } i = j, \\ w_{ij} & \text{pro } i \neq j\,, \ (i,j) \in H(\vec{G}), \\ \infty & \text{pro } i \neq j\,, \ (i,j) \notin H(\vec{G}). \end{cases}$$

2. *Pro* $k = 1, \dots, n$ *postupně vypočítáváme matice* $\mathbf{D}_k = [d^k_{i,j}]^n_{i,j=1}$, *kde*

$$d^k_{ij} = \min\{d^{k-1}_{ij}\,,\, d^{k-1}_{ik} + d^{k-1}_{kj}\}. \qquad (*)$$

3. $\mathbf{D}_n = \mathbf{D}^w(\vec{G})$.

**Věta 3.1.** *Algoritmus 3.1 nalezne w-distanční matici $\mathbf{D}^w(\vec{G})$ grafu $\vec{G}$ v čase $O(n^3)$.*

**Důkaz.** Indukcí podle $k$ dokážeme, že číslo $d^k_{i,j}$ je rovno minimální $w$-délce orientované cesty $\vec{P}$ takové, že pro její uzly platí $U(\vec{P}) \subset \{v_i, v_j\} \cup \{v_1, \dots, v_k\}$ (tj. $d^k_{ij}$ je délka minimální cesty množinou uzlů $\{1, \dots, k\}$).

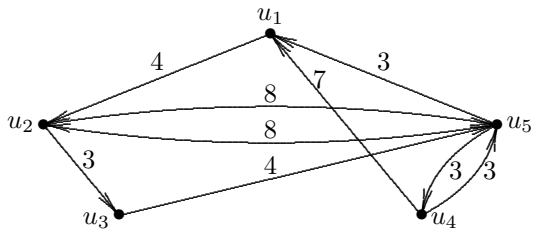
1. Pro $k = 0$ je tvrzení zřejmé.

2. Nechť tvrzení platí pro $k-1 < n$ a zvolme pevně indexy $i, j$ tak, že $1 \leq i, j \leq n$. Je-li $\vec{P}$ orientovaná cesta z $v_i$ do $v_j$ minimální $w$-délky taková, že $U(\vec{P}) \subset \{v_i, v_j\} \cup \{v_1, \dots, v_k\}$, pak buďto $u_k \notin U(\vec{P})$, a pak je podle indukčního předpokladu její $w$-délka rovna $d^{k-1}_{i,j}$, nebo $u_k \in U(\vec{P})$, a pak je její $w$-délka (opět podle indukčního předpokladu) rovna $d^{k-1}_{i,k} + d^{k-1}_{k,j}$.

[1] V textu [1] je používán název „matice vážených vzdáleností."

Algoritmus vypočítává $n$ matic $\mathbf{D}_1, \dots \mathbf{D}_n$ řádu $n$, přičemž délka výpočtu každého prvku (krok 2 algoritmu) nezávisí na $n$. Celkem tedy $n$-krát vypočítáváme $n^2$ prvků matice, výpočetní složitost je tedy $O(n^3)$. □

**Příklad.** Pro graf na následujícím obrázku postupně dostáváme:




$$\mathbf{D}_0 = \begin{bmatrix} 0 & 4 & \infty & \infty & \infty \\ \infty & 0 & 3 & \infty & 8 \\ \infty & \infty & 0 & \infty & 4 \\ 7 & \infty & \infty & 0 & 3 \\ 3 & 8 & \infty & 3 & 0 \end{bmatrix}; \qquad \mathbf{D}_1 = \begin{bmatrix} 0 & 4 & \infty & \infty & \infty \\ \infty & 0 & 3 & \infty & 8 \\ \infty & \infty & 0 & \infty & 4 \\ 7 & 11 & \infty & 0 & 3 \\ 3 & 7 & \infty & 3 & 0 \end{bmatrix};$$

$$\mathbf{D}_2 = \begin{bmatrix} 0 & 4 & 7 & \infty & 12 \\ \infty & 0 & 3 & \infty & 8 \\ \infty & \infty & 0 & \infty & 4 \\ 7 & 11 & 14 & 0 & 3 \\ 3 & 7 & 10 & 3 & 0 \end{bmatrix}; \qquad \mathbf{D}_3 = \begin{bmatrix} 0 & 4 & 7 & \infty & 11 \\ \infty & 0 & 3 & \infty & 7 \\ \infty & \infty & 0 & \infty & 4 \\ 7 & 11 & 14 & 0 & 3 \\ 3 & 7 & 10 & 3 & 0 \end{bmatrix};$$

$$\mathbf{D}_4 = \begin{bmatrix} 0 & 4 & 7 & \infty & 11 \\ \infty & 0 & 3 & \infty & 7 \\ \infty & \infty & 0 & \infty & 4 \\ 7 & 11 & 14 & 0 & 3 \\ 3 & 7 & 10 & 3 & 0 \end{bmatrix}; \qquad \mathbf{D}_5 = \mathbf{D}^w(\vec{G}) = \begin{bmatrix} 0 & 4 & 7 & 14 & 11 \\ 10 & 0 & 3 & 10 & 7 \\ 7 & 11 & 0 & 7 & 4 \\ 6 & 10 & 13 & 0 & 3 \\ 3 & 7 & 10 & 3 & 0 \end{bmatrix}.$$

**Poznámka.** Matici $\mathbf{D}^w(\vec{G})$ lze také nalézt v čase $O(n^3)$ pomocí Dijkstrova algoritmu. Dijkstrův algoritmus umožňuje nalézt vzdálenosti ze zvoleného pevného uzlu do všech ostatních uzlů grafu, tj. jeden řádek distanční matice, v čase $O(n^2)$. Opakujeme-li tento postup $n$-krát pro různé výchozí uzly, dostaneme matici $\mathbf{D}^w(\vec{G})$ v $O(n^3)$ krocích.

## 3.2 Rozložitelnost matic a struktura orientovaných grafů

**Definice 3.2.** *Čtvercová matice* $\mathbf{A}$ *se nazývá* rozložitelná, *lze-li ji napsat ve tvaru*

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{0} & \mathbf{A}_{22} \end{bmatrix},$$

*kde* $\mathbf{A}_{11}$ *a* $\mathbf{A}_{22}$ *jsou čtvercové matice řádu alespoň 1 a* $\mathbf{0}$ *je nulová matice, anebo lze-li ji do tohoto tvaru převést permutací řádků a stejnou permutací sloupců.*

*Čtvercová matice, která není rozložitelná, se nazývá* nerozložitelná.

**Poznámka.** 1. Ekvivalentně: $\mathbf{A}$ je rozložitelná, jestliže existuje permutační matice $\mathbf{P}$ tak, že

$$\mathbf{PAP}^T = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{0} & \mathbf{A}_{22} \end{bmatrix}.$$

2. Provádíme-li v matici permutaci řádků a stejnou permutaci sloupců, pak hovoříme o *simultánních permutacích.*

**Příklad.** 1. Matice

$$\begin{bmatrix} 3 & 10 & 4 & 8 \\ 0 & 1 & 0 & 6 \\ 2 & 5 & 2 & 1 \\ 0 & 3 & 0 & 1 \end{bmatrix}$$

je rozložitelná, protože prohozením 2. a 3. řádku a 2. a 3. sloupce se převede do tvaru

$$\begin{bmatrix} 3 & 4 & 10 & 8 \\ 2 & 2 & 5 & 1 \\ 0 & 0 & 1 & 6 \\ 0 & 0 & 3 & 1 \end{bmatrix}.$$

2. Matice

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 3 \\ 0 & 2 & 0 & 0 \\ 4 & 0 & 0 & 0 \end{bmatrix}$$

je nerozložitelná, ale prozatím jediná možnost jak to umíme dokázat je prověření všech permutací řádků a sloupců.

Následující věta poskytuje efektivní algoritmus pro testování rozložitelnosti matice převodem na grafovou úlohu.

**Věta 3.2.** *Buď $\vec{G}$ ohodnocený orientovaný graf. Platí:*

*a) Je-li $\vec{G}$ silně souvislý, pak je matice $\mathbf{W}(\vec{G})$ nerozložitelná.*

*b) Jsou-li $\vec{G}_1, \ldots, \vec{G}_k$ kvazikomponenty grafu $\vec{G}$, očíslované tak, že v kondenzaci $\vec{G}_C$ jsou pouze hrany $(\vec{G}_k, \vec{G}_\ell)$ pro $k < \ell$ a očíslujeme-li uzly grafu $\vec{G}$ souhlasně s očíslováním kvazikomponent, tj. tak, že je-li $i \in \vec{G}_k$ a $j \in \vec{G}_\ell$ pro $k < \ell$, pak $i < j$, pak matice $\mathbf{W}(\vec{G})$ má tvar*

$$\mathbf{W}(\vec{G}) = \begin{bmatrix} \mathbf{W}_{11} & \mathbf{W}_{12} & \mathbf{W}_{13} & \ldots & \mathbf{W}_{1k} \\ \mathbf{0} & \mathbf{W}_{22} & \mathbf{W}_{23} & \ldots & \mathbf{W}_{2k} \\ \mathbf{0} & \mathbf{0} & \mathbf{W}_{33} & \ldots & \mathbf{W}_{3k} \\ & & \ldots & & \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{W}_{kk} \end{bmatrix},$$

*kde $\mathbf{W}_{ii} = \mathbf{W}(\vec{G})_i$, $i = 1, \ldots, k$, a tyto matice jsou již nerozložitelné.*

**Důkaz.** a) Dokážeme ekvivalentní implikaci: je-li $\mathbf{W}(\vec{G})$ rozložitelná, pak $\vec{G}$ není silně souvislý.

Nechť tedy $\mathbf{W}(\vec{G})$ má (případně po permutaci) tvar

$$\mathbf{W}(\vec{G}) = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{0} & \mathbf{A}_{22} \end{bmatrix}.$$

Je zřejmé, že simultánní permutaci řádků a sloupců matice $\mathbf{W}(\vec{G})$ odpovídá přečíslování uzlů grafu $\vec{G}$. Buď $U_1$, resp. $U_2$, množina uzlů odpovídajících matici $\mathbf{A}_{11}$, resp. $\mathbf{A}_{22}$. Pak z žádného uzlu množiny $U_2$

nevede hrana do některého uzlu z $U_1$, a tedy z žádného uzlu množiny $U_2$ neexistuje orientovaná cesta do některého uzlu z $U_1$. Graf $\vec{G}$ tedy není silně souvislý.

b) Nechť $\vec{G}_1, \ldots, \vec{G}_k$ jsou kvazikomponenty grafu $\vec{G}$, očíslované podle věty 3.2. Pak mezi žádnými dvěma kvazikomponentami $\vec{G}_i$, $\vec{G}_j$ pro $i > j$ neexistuje hrana z $\vec{G}_i$ do $\vec{G}_j$. To znamená, že matice $\mathbf{W}(\vec{G})$ je následujícího tvaru:

| | *uzly* $\vec{G}_1$ | *uzly* $\vec{G}_2$ | *uzly* $\vec{G}_3$ | $\cdots$ | *uzly* $\vec{G}_k$ |
|---|---|---|---|---|---|
| *uzly* $\vec{G}_1$ | $\mathbf{W}(\vec{G}_1)$ | $\mathbf{W}_{12}$ | $\mathbf{W}_{13}$ | $\cdots$ | $\mathbf{W}_{1k}$ |
| *uzly* $\vec{G}_2$ | $\mathbf{0}$ | $\mathbf{W}(\vec{G}_2)$ | $\mathbf{W}_{23}$ | $\cdots$ | $\mathbf{W}_{2k}$ |
| *uzly* $\vec{G}_3$ | $\mathbf{0}$ | $\mathbf{0}$ | $\mathbf{W}(\vec{G}_3)$ | $\cdots$ | $\mathbf{W}_{3k}$ |
| $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ |
| *uzly* $\vec{G}_k$ | $\mathbf{0}$ | $\mathbf{0}$ | $\mathbf{0}$ | $\cdots$ | $\mathbf{W}(\vec{G}_k)$ |

Přitom každá kvazikomponenta $\vec{G}_i$ je silně souvislý graf a tedy podle (již dokázané) části a) jsou všechny matice $\mathbf{W}(\vec{G}_i)$ nerozložitelné. □

**Důsledek 3.1.** *Čtvercová matice* $\mathbf{A}$ *je nerozložitelná právě když její diagram* $\vec{G}(\mathbf{A})$ *je silně souvislý.*

**Příklad.** Diagramem matice

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 3 \\ 0 & 2 & 0 & 0 \\ 4 & 0 & 0 & 0 \end{bmatrix}$$

(viz příklad před větou 3.2) je cyklus délky 4, který je silně souvislý. Matice je tedy nerozložitelná.

**Příklad.** Převeďte následující matici

$$\begin{bmatrix} 0 & 1 & 0 & 2 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 \\ 0 & 5 & 6 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 & 8 & 0 \\ 9 & 10 & 0 & 0 & 11 & 12 \\ 0 & 13 & 0 & 0 & 0 & 14 \end{bmatrix}$$

simultánními permutacemi na horní blokově trojúhelníkový tvar.

Řešení: s použitím věty 3.2. Diagram matice má 3 kvazikomponenty s množinami uzlů $\{1, 4, 5\}$, $\{2, 6\}$ a $\{3\}$. Přečíslování uzlů definuje simultánní permutaci, kterou se matice převede do tvaru

$$\begin{bmatrix} 0 & 7 & 8 & 0 & 0 & 0 \\ 2 & 0 & 3 & 0 & 1 & 0 \\ 0 & 9 & 11 & 0 & 10 & 12 \\ 0 & 0 & 0 & 6 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 13 & 14 \end{bmatrix}.$$

## 3.3 Slabě rozložitelné matice

**Definice 3.3.** *Řekneme, že čtvercová matice* $\mathbf{A}$ *je* slabě rozložitelná, *jestliže existují permutační matice* $\mathbf{P}$ *a* $\mathbf{Q}$ *tak, že*

$$\mathbf{PAQ} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{0} & \mathbf{A}_{22} \end{bmatrix},$$

*kde* $\mathbf{A}_{11}$ *a* $\mathbf{A}_{22}$ *jsou čtvercové matice řádu alespoň 1 a* $\mathbf{0}$ *je nulová matice.*

*Čtvercová matice, která není slabě rozložitelná, se nazývá* úplně nerozložitelná.

**Poznámka.** 1. Každá rozložitelná matice je slabě rozložitelná, ale nikoliv naopak.

2. Ekvivalentní definice: $\mathbf{A}$ je slabě rozložitelná, jestliže existuje permutační matice $\mathbf{P}$ tak, že $\mathbf{PA}$ je rozložitelná.

**Příklad.** 1. Nerozložitelná matice

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 3 \\ 0 & 2 & 0 & 0 \\ 4 & 0 & 0 & 0 \end{bmatrix}$$

je slabě rozložitelná: stačí prohodit první sloupec s třetím a druhý sloupec se čtvrtým.

2. Matice

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

je úplně nerozložitelná, ale prozatím to neumíme dokázat (jinak než hrubou silou).

**Definice 3.4.** Bigraf *je orientovaný graf* $\vec{G}$*, jehož množinu uzlů lze rozložit na disjunktní neprázdné podmnožiny* $U_1$, $U_2$ [2] *tak, že pro každou hranu* $(u, v) \in H(\vec{G})$ *je* $u \in U_1$ *a* $v \in U_2$.

**Poznámka.** 1. Graf $\vec{G}$ je bigrafem právě když $\vec{G}$ je acyklický graf, jehož každý uzel je vstupní nebo výstupní.

2. Symetrizace bigrafu je bipartitní neorientovaný graf (tj. neorientovaný graf s chromatickým číslem 2 – blíže v kap. 7).

**Definice 3.5.** *Buď* $\mathbf{A} = [a_{ij}]$ *čtvercová matice řádu* $n$*; označme* $U_1$ *množinu řádkových indexů a* $U_2$ *množinu sloupcových indexů matice* $\mathbf{A}$. Bigraf matice $\mathbf{A}$ *je orientovaný graf* $\vec{B}(\mathbf{A})$ *s množinou uzlů*

$$U = U_1 \cup U_2$$

*a množinou hran*

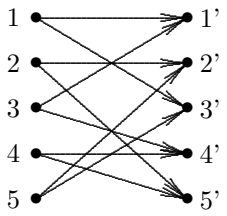
$$H = \{(i, j) |\ i \in U_1, j \in U_2, a_{ij} \neq 0\}.$$

[2] Tj. $U_1 \cap U_2 = \emptyset$, $U_1 \cup U_2 = U(\vec{G})$, $U_1 \neq \emptyset \neq U_2$.

**Příklad.** Matice

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

má bigraf

$\vec{B}(\mathbf{A}) =$ 


**Poznámka.** Je zřejmé, že také naopak, každému bigrafu lze zřejmým způsobem přiřadit matici.

**Definice 3.6.** *Buď $\vec{G}$ bigraf s množinou uzlů $U(\vec{G}) = U_1 \cup U_2$. Množina $V \subset U_1$, $\emptyset \neq V \neq U_1$, se nazývá* stabilní množina v $\vec{G}$, *jestliže pro množinu uzlů*

$$W = \{j \in U_2 |\ \exists i \in V \text{ tak, že } (i,j) \in H(\vec{G})\}$$

*platí*

$$|W| \leq |V|.$$

**Věta 3.3.** *Čtvercová matice $\mathbf{A}$ je slabě rozložitelná právě když v jejím bigrafu $\vec{B}(\mathbf{A})$ existuje stabilní množina.*

**Důkaz.** 1. Nechť $\mathbf{A}$ je slabě rozložitelná, tj. existují permutační matice $\mathbf{P}$ a $\mathbf{Q}$ tak, že

$$\mathbf{PAQ} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{0} & \mathbf{A}_{22} \end{bmatrix}.$$

Označíme-li $V$ množinu indexů druhého blokového řádku matice $\mathbf{PAQ}$ a $W$ množinu indexů druhého blokového sloupce matice $\mathbf{PAQ}$, pak v $\vec{B}(\mathbf{PAQ})$ všechny hrany, vycházející z uzlů ve $V$, vedou do uzlů v $W$ a protože $\mathbf{A}_{22}$ je čtvercová, je $|V| = |W|$, tj. $V$ je stabilní. Zbytek vyplývá z toho, že $\vec{B}(\mathbf{A})$ se od $\vec{B}(\mathbf{PAQ})$ liší jen číslováním řádků a sloupců, tj. uzlů v $U_1$ a v $U_2$.

2. Naopak, je-li $V$ stabilní množina $\vec{B}(\mathbf{A})$, tj. pro příslušnou $W$ je $|W| \leq |V|$, pak v $U_1$ očíslujeme nejprve uzly mimo $V$ a potom uzly množiny $V$, a obdobně v $U_2$ očíslujeme nejprve uzly mimo $W$ a potom uzly množiny $W$. Provedeme-li v matici $\mathbf{A}$ odpovídající permutaci řádků a sloupců, bude mít požadovaný tvar. □

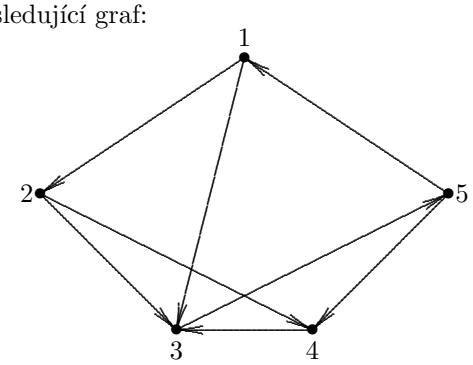
**Příklad.** Zjistěte, zda je matice

$$\mathbf{A} = \begin{bmatrix} 0 & 5 & -1 & 0 & 0 \\ 0 & 2 & -3 & 3 & 0 \\ 2 & 0 & 1 & 0 & 1 \\ 0 & 0 & 2 & -1 & 0 \\ 3 & 0 & 0 & -2 & 2 \end{bmatrix}$$

rozložitelná, slabě rozložitelná nebo úplně nerozložitelná.

1. Diagram matice **A** je následující graf:

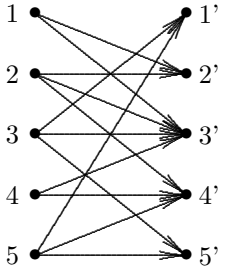
$\vec{G}(\mathbf{A})$:




(V obrázku nejsou zakresleny smyčky na uzlech 2, 3, 4 a 5, které nemají vliv na silnou souvislost grafu). Protože je $\vec{G}(\mathbf{A})$ silně souvislý, je **A** nerozložitelná.

2. Bigraf matice **A** je následující graf:

$\vec{B}(\mathbf{A})$ :




Snadno se přesvědčíme, že množina $V = \{1, 2, 4\}$ je stabilní, neboť odpovídající podmnožina množiny $U_2$ je $W = \{2', 3', 4'\}$ a tedy $|V| = |W|$. Po provedení příslušných permutací je matice **A** převedena do tvaru

$$\mathbf{PAQ} = \begin{bmatrix} 2 & 1 & 0 & 1 & 0 \\ 3 & 2 & 0 & 0 & -2 \\ 0 & 0 & 5 & -1 & 0 \\ 0 & 0 & 2 & -3 & 3 \\ 0 & 0 & 0 & 2 & -1 \end{bmatrix}$$

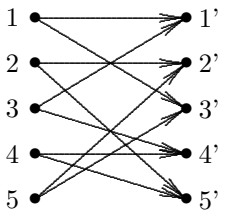
Matice **A** tedy je slabě rozložitelná.

**Příklad.** Matice

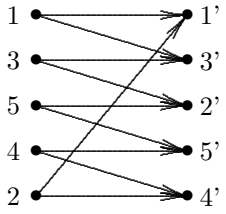
$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

je úplně nerozložitelná, protože její bigraf

$\vec{B}(\mathbf{A}) =$



nemá stabilní množinu. Neexistence stabilní množiny v $\vec{B}(\mathbf{A})$ je snadno vidět při jeho následujícím izomorfním nakreslení.

$\vec{B}(\mathbf{A}) =$ 


## 3.4 Regulární matice a perfektní párování v bigrafu

Poznatky z odstavce 3.3 ukazují, že bigraf matice by mohl být vhodným nástrojem studia těch vlastností matic, které nezávisí na permutacích řádků a sloupců. Jinou takovou vlastností je regularita matice. V tomto odstavci si ukážeme, zda a jak lze z bigrafu matice poznat, zda je tato matice regulární.

**Definice 3.7.** *Řekneme, že bigraf $\vec{G}$ je* lineární, *jestliže pro každý uzel $u \in U_1$ je $d^-(u) = 1$ a pro každý uzel $v \in U_2$ je $d^+(v) = 1$.*

**Poznámka.** Je-li $\vec{G}$ lineární bigraf, pak nutně $|U_1| = |U_2|$ a hrany $\vec{G}$ definují vzájemně jednoznačné zobrazení $U_1$ na $U_2$.

**Definice 3.8.** *Je-li $\vec{G}$ bigraf a $\vec{G}_1 \subset \vec{G}$ jeho lineární podbigraf, pak říkáme, že $\vec{G}_1$ je* párování *v $\vec{G}$.*

*Je-li $\vec{G}_1 \subset \vec{G}$ párování v $\vec{G}$ takové, že $U(\vec{G}_1) = U(\vec{G})$ (tj. $\vec{G}_1$ je faktorem bigrafu $\vec{G}$), pak říkáme, že $\vec{G}_1$ je* perfektní párování *v $\vec{G}$.*

**Věta 3.4.** *1. Je-li $\mathbf{A}$ regulární matice, pak její bigraf $\vec{B}(\mathbf{A})$ má perfektní párování.*

*2. Jestliže bigraf $\vec{G}$ má perfektní párování, pak existuje regulární matice $\mathbf{A}$ taková, že $\vec{B}(\mathbf{A}) = \vec{G}$.*

**Důkaz.** 1. Je-li $\mathbf{A}$ regulární, pak $\det(\mathbf{A}) \neq 0$ a tedy v sumě (definice determinantu)

$$\det(\mathbf{A}) = \sum_{\pi} \mathrm{zn}(\pi) a_{1,\pi(1)} \cdot \ldots \cdot a_{n,\pi(n)}$$

je pro alespoň jednu permutaci $\pi_0$ příslušný součin $a_{1,\pi_0(1)} \cdot \ldots \cdot a_{n,\pi_0(n)}$ nenulový. Hrany, odpovídající prvkům tohoto součinu, určují perfektní párování v $\vec{B}(\mathbf{A})$.

2. Nechť naopak $\vec{G}$ má perfektní párování $\vec{G}_1$. Sestrojme matici $\mathbf{A}$ následujícím předpisem:
– prvky matice $\mathbf{A}$ odpovídající hranám perfektního párování $\vec{G}_1$ jsou rovny jedné,
– prvky $\mathbf{A}$ odpovídající hranám bigrafu $\vec{G}$ ležícím mimo perfektní párování $\vec{G}_1$ jsou rovny $\varepsilon > 0$,
– ostatní prvky matice $\mathbf{A}$ jsou rovny 0.

Pak pro $|\det(\mathbf{A})|$ dostaneme dolní odhad

$$\begin{aligned} |\det(\mathbf{A})| &= |\sum \mathrm{zn}(\pi) a_{1,\pi(1)} \cdot \ldots \cdot a_{n,\pi(n)}| \\ &\geq |a_{1,\pi_0(1)} \cdot \ldots \cdot a_{n,\pi_0(n)}| - |\sum_{\pi'} \mathrm{zn}(\pi') a_{1,\pi'(1)} \cdot \ldots \cdot a_{n,\pi'(n)}|, \end{aligned}$$

kde $\pi_0$ je permutace, pro kterou prvky $a_{1,\pi_0(1)}, \ldots, a_{n,\pi_0(n)}$ odpovídají hranám perfektního párování $\vec{G}_1$, a v poslední sumě sčítáme přes všechny permutace $\pi'$ různé od $\pi_0$. Odtud dále dostaneme

$$|\det(\mathbf{A})| \geq 1 - |\sum_{\pi'} \mathrm{zn}(\pi') a_{1,\pi'(1)} \cdot \ldots \cdot a_{n,\pi'(n)}| \geq 1 - \varepsilon \cdot n! \quad .$$

Pro pevné $n$ je

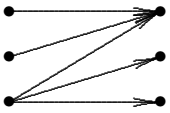
$$\lim_{\varepsilon \to 0} \varepsilon \cdot n! = 0 \ ,$$

a tedy při dostatečně malém $\varepsilon > 0$ bude $\det(\mathbf{A}) \neq 0$. □

**Příklad.** 1. Matice

$$\mathbf{A}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 0 & 0 \\ -1 & 1 & 2 \end{bmatrix}$$

je singulární, protože její bigraf

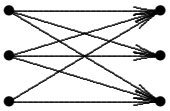
$\vec{B}(\mathbf{A}_1) =$ 

nemá perfektní párování.

2. Uvažujme matici

$$\mathbf{A}_2 = \begin{bmatrix} 1 & -1 & 2 \\ 2 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}.$$

Její bigraf

$\vec{B}(\mathbf{A}_2) =$ 

má perfektní párování, přestože $\mathbf{A}_2$ je singulární. Tento příklad ukazuje, že v prvním tvrzení vět z 3.4 neplatí ekvivalence.

**Poznámka.** Modifikujme matici $\mathbf{A}_2$ z předchozího příkladu následujícím způsobem.

$$\mathbf{A}_2(a) = \begin{bmatrix} 1 & -1 & 2 \\ 2 & 1 & 1 \\ 1 & 0 & a \end{bmatrix}$$

Pak zřejmě $\det(\mathbf{A}_2(a)) = 3a - 3$, a tedy $\mathbf{A}_2(a)$ je singulární právě když $a = 1$. Tedy $\mathbf{A}_2(a)$ je regulární pro „skoro všechny“ hodnoty parametru $a$. Tento příklad motivuje úvahy následujícího odstavce.

## 3.5 Strukturální matice a generická hodnost

Úvodem poznamenejme, že přesné provedení důkazů všech tvrzení tohoto odstavce by si vyžadovalo aparát, který přesahuje rámec látky vyučované na inženýrských oborech FAV, a proto bude výklad v některých částech více intuitivní než v jiných kapitolách. Odstavec je zařazen do předmětu protože ukazuje zajímavou souvislost mezi různými oblastmi matematiky, a má přímé aplikace v teorii řízení.

V předchozím odstavci jsme poznali souvislost mezi regularitou matice $\mathbf{A}$ a vlastnostmi jejího bigrafu $\vec{B}(\mathbf{A})$. Věta 3.4 ale není ekvivalence, pouze implikace. Podstata této „obtíže“ je v tom, že zatímco hodnost matice (a speciálně regularita) závisí nejen na struktuře rozmístění nulových a nenulových prvků v dané matici, ale i na jejich konkrétních hodnotách, bigraf $\vec{B}(\mathbf{A})$ popisuje pouze strukturu rozmístění nulových a nenulových prvků, a na jejich konkrétních hodnotách nezávisí. Chceme-li tedy dostat ekvivalenci, musíme najít maticový pojem s podobnými vlastnostmi.

*Strukturální matice řádu* $n \geq 1$ je čtvercová matice řádu $n$, u níž je dána pouze struktura nulových a nenulových prvků, ale nejsou určeny jejich konkrétní hodnoty (na strukturální matici tedy můžeme – poněkud přesněji – pohlížet jako na funkci hodnot jejích nenulových prvků). Strukturální matice budeme značit obvyklým způsobem s „křížkem“ na místě nenulových prvků. Příkladem takových strukturálních matic jsou tedy následující dvě matice.

$$\mathbf{A}_1 = \begin{bmatrix} 0 & 0 & \times \\ 0 & 0 & \times \\ \times & \times & \times \end{bmatrix}; \qquad \mathbf{A}_2 = \begin{bmatrix} \times & \times & 0 \\ \times & \times & 0 \\ 0 & 0 & \times \end{bmatrix}.$$

Povšimněme si již nyní toho, že zatímco u matice $\mathbf{A}_2$ existují hodnoty nenulových prvků, při nichž je vzniklá matice regulární, je matice $\mathbf{A}_1$ singulární pro každou volbu svých nenulových prvků.

Je-li $\mathbf{A}$ strukturální matice řádu $n$, mající $\ell$ nenulových prvků ($\ell \leq n^2$), pak její determinant $\det(\mathbf{A})$ je polynomem řádu $\ell$ v $\ell$ proměnných, jimiž jsou nenulové prvky $\mathbf{A}$. Například, snadno zjistíme že $\det(\mathbf{A}_2) = \alpha_{1,1}\alpha_{2,2}\alpha_{3,3} - \alpha_{1,2}\alpha_{2,1}\alpha_{3,3}$, zatímco $\det(\mathbf{A}_1)$ je nulový polynom.

Pokud $\det(\mathbf{A})$ není nulovým polynomem, pak je rovnice $\det(\mathbf{A}) = 0$ algebraickou rovnicí řádu $\ell$ v $\ell$ proměnných, a tedy množina všech jejích řešení je algebraickou varietou dimenze $r \leq \ell - 1$ v $\ell$-rozměrném prostoru $\mathbf{R}_\ell$. [3]

Z teorie míry [4] je známo, že pro $r < \ell$ má každá $r$-rozměrná varieta v $\mathbf{R}_\ell$ nulovou $\ell$-rozměrnou míru. [5] To znamená, že pokud $\det(\mathbf{A})$ není nulový polynom, pak množina všech řešení rovnice $\det(\mathbf{A}) = 0$ je podmnožinou nulové míry v prostoru $\mathbf{R}_\ell$. Odtud plyne, že je-li $\mathbf{A}$ strukturální matice řádu $n$ s $\ell$ nenulovými prvky $\alpha_1, \ldots, \alpha_\ell$ taková, že $\det(\mathbf{A})$ není nulový polynom, pak při náhodné volbě parametrů $\alpha_1, \ldots, \alpha_\ell$ [6] je pravděpodobnost jevu $\det(\mathbf{A}) = 0$ rovna nule. Komplementární jev $\det(\mathbf{A}) \neq 0$ má tedy pravděpodobnost 1.

Pro danou strukturální matici $\mathbf{A}$ jsou tedy pouze následující dvě možnosti:

($i$) polynom $\det(\mathbf{A})$ je nenulový, a při náhodné volbě nenulových prvků matice $\mathbf{A}$ je matice $\mathbf{A}$ regulární s pravděpodobností 1,

($ii$) polynom $\det(\mathbf{A})$ je nulový a matice $\mathbf{A}$ je singulární při každé volbě jejích nenulových prvků.

V prvním případě říkáme, že strukturální matice $\mathbf{A}$ je *genericky regulární*, ve druhém případě je $\mathbf{A}$ *genericky singulární*.

Nyní již snadno odvodíme následující tvrzení, které dává hledanou ekvivalenci.

**Věta 3.5.** *Nechť* $\mathbf{A}$ *je čtvercová strukturální matice. Pak je matice* $\mathbf{A}$ *genericky regulární právě když její bigraf* $\vec{B}(\mathbf{A})$ *má perfektní párování.*

**Důkaz.** Důkaz věty 3.5 plyne ihned z věty 3.4. □

**Příklad.** Strukturální matice

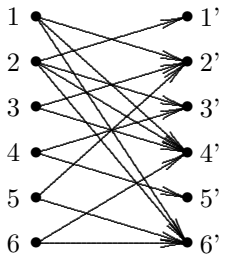
$$\mathbf{A} = \begin{bmatrix} 0 & \times & 0 & \times & 0 & \times \\ \times & 0 & \times & \times & 0 & \times \\ 0 & \times & 0 & \times & 0 & 0 \\ 0 & 0 & \times & 0 & \times & 0 \\ 0 & \times & 0 & 0 & 0 & \times \\ 0 & 0 & 0 & \times & 0 & \times \end{bmatrix}$$

[3] Uvedené pojmy zde přesně nedefinujeme (neboť by to přesáhlo rámec tohoto textu), a spolehláme na geometrickou představu a intuici čtenáře. Například – v prostoru $\mathbf{R}_3$ je množina všech řešení algebraické rovnice $xy - z = 0$ hyperbolický paraboloid, což je objekt dimenze 2 (parametrizovatelný dvěma parametry). Ve vyšších dimenzích je situace analogická, pouze obtížněji představitelná geometrickým názorem.

[4] Pro čtenáře, kteří neabsolvovali základy teorie míry: míra $\mu(M)$ množiny $M \subset \mathbf{R}_\ell$ je – velmi zhruba řečeno – číslo, které zobecňuje pojem délky, obsahu či objemu do libovolné dimenze. Tedy, míra (jednorozměrná) křivky je její délka, míra (dvourozměrná) plochy je její obsah, míra (trojrozměrná) tělesa je jeho objem atd.

[6] při libovolném „rozumném“ spojitém rozdělení náhodného vektoru $[\alpha_1, \ldots, \alpha_\ell]$

má bigraf

$\vec{B}(\mathbf{A}) =$ 


Povšimněme si toho, že v $\vec{B}(\mathbf{A})$ vedou z množiny uzlů $V = \{1, 3, 5, 6\} \subset U_1$ hrany pouze do uzlů množiny $W = \{2', 4', 6'\} \subset U_2$. Kdyby $\vec{B}(\mathbf{A})$ měl perfektní párování, tak by z každého uzlu množiny $V$ musela vycházet právě jedna hrana tohoto párování, což není možné, protože $|W| < |V|$. Bigraf $\vec{B}(\mathbf{A})$ tedy nemá perfektní párování a matice $\mathbf{A}$ je genericky singulární.

Předchozí úvahy je možno ještě trochu zobecnit. Z lineární algebry víme, že hodnost libovolné reálné matice $\mathbf{A}$ je rovna řádu její největší regulární podmatice (přesněji: největšímu číslu $k$, pro které v matici $\mathbf{A}$ existuje regulární podmatice řádu $k$). Tuto alternativní definici hodnosti je možno přenést na strukturální matice a generickou regularitu.

Nechť $\mathbf{A}$ je strukturální matice. Největší přirozené číslo $k$, pro které v matici $\mathbf{A}$ existuje genericky regulární podmatice řádu $k$, se nazývá *generická hodnost* matice $\mathbf{A}$ a značí se $\mathrm{gh}(\mathbf{A})$.

Počet hran největšího párování v bigrafu $\vec{B}$ se nazývá *párovací číslo* bigrafu $\vec{B}$ a značí se $\nu(\vec{B})$.

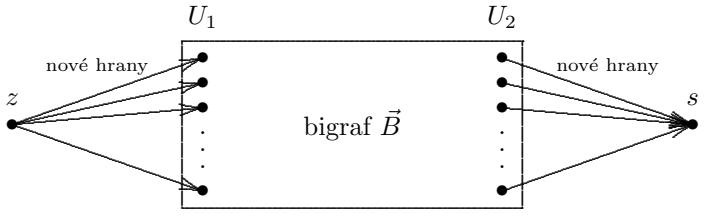
Aplikací věty 3.5 na největší genericky regulární podmatici ihned dostaneme následující tvrzení.

**Věta 3.6.** *Nechť* $\mathbf{A}$ *je strukturální matice. Pak* $\mathrm{gh}(\mathbf{A}) = \nu(\vec{B}(\mathbf{A}))$.

**Příklad.** Buď $\mathbf{A}$ matice z příkladu za větou 3.5. Víme již, že $\mathbf{A}$ je genericky singulární, protože $\vec{B}(\mathbf{A})$ nemá perfektní párování. Snadno ale v $\vec{B}(\mathbf{A})$ najdeme párování velikosti 5, a tedy $\mathrm{gh}(\mathbf{A}) = \nu(\vec{B}(\mathbf{A})) = 5$. To znamená, že při „náhodném“ dosazení nenulových prvků do matice $\mathbf{A}$ bude mít výsledná reálná matice s pravděpodobností 1 hodnost 5.

Závěrem tohoto odstavce zdůrazněme, že největší párování v libovolném bigrafu je možno najít v polynomiálním čase převodem úlohy na úlohu maximálního toku následující konstrukcí, jejíž obdobu již známe z důkazu věty 2.1 (viz str. 11).

Nechť $\vec{B}$ je bigraf s množinou uzlů $U(\vec{B}) = U_1 \cup U_2$. Bigrafu $\vec{B}$ přiřadíme síť s jedním zdrojem $z$ a jedním stokem $s$ tak, že k $\vec{B}$ přidáme nový uzel $z$ a nový uzel $s$, a přidáme
– hrany $(z, u)$ pro všechny uzly $u \in U_1$,
– hrany $(v, s)$ pro všechny uzly $v \in U_2$
(viz obrázek). Propustnosti všech hran jsou rovny jedné.




Najdeme-li v $\vec{G}$ (celočíselný) maximální tok, pak hrany bigrafu $\vec{B}$ s nenulovým tokem určují největší párování v $\vec{B}$.

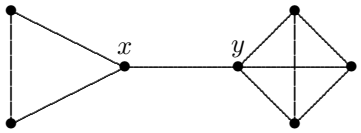
# 4 Míry souvislosti grafu

V celé následující kapitole termín "graf" znamená neorientovaný graf.

## 4.1 Mosty, artikulace, bloky grafu

**Definice 4.1.** *Hrana $\{x, y\} \in H(G)$ se nazývá* most *grafu $G$, jestliže v grafu $G$ neexistuje žádná kružnice, která ji obsahuje.*

**Příklad.** Každá hrana stromu je jeho mostem; příkladem mostu je též hrana $\{x, y\}$ na následujícím obrázku.




**Tvrzení 4.1.** *Je-li graf $G$ souvislý a hrana $\{x, y\}$ jeho most, pak graf $G - \{x, y\}$, vzniklý odstraněním hrany $\{x, y\}$ z $G$, je nesouvislý.*

**Důkaz.** Kdyby graf $G - \{x, y\}$ byl souvislý, existovala by v něm cesta $P$ z $x$ do $y$. Cesta $P$ by spolu s hranou $\{x, y\}$ tvořila v grafu $G$ kružnici, obsahující hranu $\{x, y\}$, což je spor. □

**Věta 4.1.** *Má-li souvislý graf $G$ most, pak má alespoň dva uzly lichého stupně.*

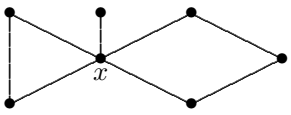
**Důkaz.** Označme $G'$ nesouvislý graf, který vznikne odstraněním mostu z grafu $G$. Kdyby měl graf $G$ stupně všech uzlů sudé, tak by v každé komponentě grafu $G'$ byl právě jeden uzel lichého stupně, což zřejmě nelze (uzlů lichého stupně je sudý počet). □

**Příklad.** Následující dva grafy ukazují, že uzly lichého stupně mohou být uzly mostu i některé jiné dva uzly grafu (uzly lichých stupňů jsou zakroužkovány).

Uzlovou analogií pojmu mostu grafu je pojem artikulace grafu.

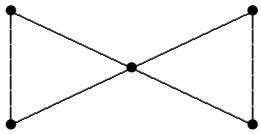
**Definice 4.2.** *Uzel $x \in U(G)$ je* artikulace *grafu $G$, jestliže existují hrany $\{x, y_1\}$ a $\{x, y_2\}$, které nepatří současně téže kružnici grafu $G$.*

**Příklad.**
($i$) Každý nekoncový uzel stromu je jeho artikulace.
($ii$) Uzel $x$ na následujícím obrázku je artikulací.

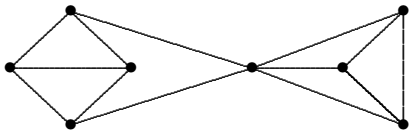


**Poznámka.**

1) Grafy na následujícím obrázku ukazují, že existence artikulace v grafu nemá (na rozdíl od mostu) žádný vliv na paritu stupňů uzlů.



sudé stupně všech uzlů



všechny uzly mají liché stupně

2) Je-li hrana $\{x, y\}$ mostem, pak oba uzly $x$, $y$ jsou artikulacemi. Jinak řečeno: nemá-li graf artikulaci, pak nemůže mít ani most.

**Definice 4.3.** *Buď $G$ graf, $G' \subset G$ jeho souvislý podgraf. Řekneme, že $G'$ je* blok *grafu $G$, jestliže:*

*a) $G'$ nemá artikulaci,*

*b) jestliže $G''$ je souvislý graf bez artikulace takový, že $G' \subset G'' \subset G$, pak $G'' = G'$.*

**Poznámka.**

1) Blok je maximální souvislý podgraf bez artikulace.
2) Je-li $\{x, y\}$ most, pak podgraf o jediné hraně $\{x, y\}$ je blok $G$ (každý most je blokem grafu).

**Tvrzení 4.2.** *Buď $G$ souvislý graf. Pak $G$ nemá artikulaci právě když pro každé dvě jeho hrany existuje kružnice, na níž obě leží.*

**Důkaz.** 1. Jestliže $G$ má artikulaci, pak podle definice artikulace existují hrany $\{x, y_1\}$, $\{x, y_2\}$, neležící na kružnici.

2. Nechť naopak existuje dvojice hran $h_1$, $h_2$, neležící na kružnici. Kdyby $h_1$, $h_2$ měly společný uzel $x$, pak $x$ je artikulací a jsme hotovi. Tedy každá taková dvojice hran $h_1$, $h_2$ je ve vzdálenosti alespoň 1. Zvolme $h_1$, $h_2$ tak, že jejich vzdálenost (tj. minimum vzdáleností jejich uzlů) je nejmenší možná, a nechť $P = u_1 u_2 \dots u_k$ je příslušná nejkratší cesta. Zvolme označení tak, že $h_1 = \{u_1, v_1\}$ a $h_2 = \{u_k, v_2\}$. Z minimality cesty $P$ plyne, že existuje kružnice $C_1$, obsahující hrany $\{u_1, v_1\}$ a $\{u_{k-1}, u_k\}$. Protože $u_k$ není artikulací, existuje kružnice $C_2$, obsahující $\{u_{k-1}, u_k\}$ a $\{u_k, v_2\}$. Podle předpokladu $C_2$ neobsahuje $h_1$. Množina hran $H(C_1) \cup H(C_2)$ tedy definuje uzavřený tah, obsahující hrany $h_1$, $h_2$ právě jednou. Z tohoto tahu lze zřejmým způsobem vybrat požadovanou kružnici. □

**Důsledek 4.1.** *Pro každé dvě hrany bloku, který není mostem, existuje kružnice, na níž obě leží.*

**Věta 4.2.** *Buďte $G_1$, $G_2$ dva bloky grafu $G$. Pak buďto $G_1 = G_2$, nebo $G_1$ a $G_2$ nemají žádnou společnou hranu.*

**Důkaz.** Nechť $h$ je hrana $G_1$ i $G_2$. Je-li $h$ most, tak je vše zřejmé. Nechť tedy hrana $h$ leží na alespoň jedné kružnici. Označme $G_h$ podgraf grafu $G$, tvořený těmi jeho hranami, jež leží spolu s hranou $h$ na některé kružnici. Blok $G_1$ obsahuje hranu $h$ a tedy dle důsledku 4.1 je $G_1 \subset G_h$. Obdobně i $G_2 \subset G_h$. Z konstrukce grafu $G_h$ a z tvrzení 4.2 ale plyne, že každé dvě hrany $G_h$ spolu leží na kružnici a tedy $G_h$ nemá artikulaci. Z maximality bloků pak již plyne rovnost $G_1 = G_2 = G_h$. □

**Poznámka.** Situace je podobná jako u komponent s tím rozdílem, že bloky nemusí být uzlově disjunktní, musí však být hranově disjunktní.

**Definice 4.4.** *Buď $G$ souvislý graf, $B_1, \ldots, B_r$ všechny jeho bloky a $x_1, \ldots, x_s$ všechny jeho artikulace. Graf $B(G)$, definovaný předpisem*

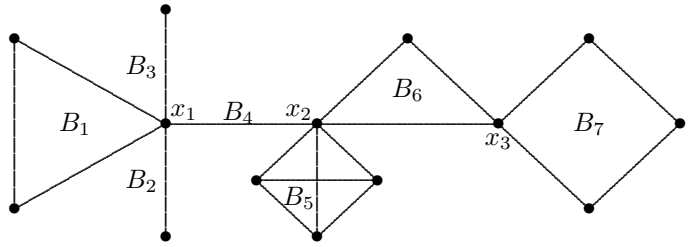
$U(B(G)) = \{x_1, \ldots, x_s, B_1, \ldots, B_r\}$,

$H(B(G)) = \{\{a, b\} \mid \exists i, j \text{ tak, že } a = x_i,\ b = B_j \text{ a } x_i \in U(B_j)\}$,

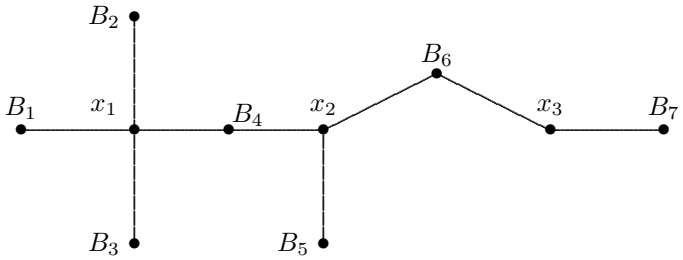
*se nazývá* blokový graf *grafu $G$.*

**Příklad.**

$G$




$B(G)$




Zřejmá, leč důležitá vlastnost blokového grafu je popsána v následující větě.

**Věta 4.3.** *Pro každý souvislý graf $G$ je blokový graf $B(G)$ stromem.*

## 4.2 Hranový a uzlový stupeň souvislosti grafu

Jednoduché a intuitivně zřejmé pojmy mostu a artikulace dostanou hlubší význam, jestliže je zobecníme na víceprvkové množiny.

**Definice 4.5.** *Buď $G$ souvislý; $x, y \in U(G)$. Množina $B \subset H(G)$ taková, že*

*1) každá cesta z uzlu $x$ do uzlu $y$ obsahuje alespoň jednu hranu množiny $B$,*

*2) žádná vlastní podmnožina množiny $B$ nemá vlastnost 1),*

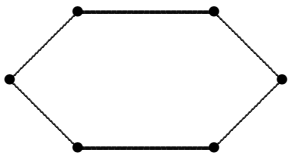
*se nazývá* hranový řez *grafu $G$ mezi uzly $x$ a $y$.*

**Poznámka.**

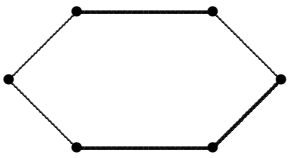
(*i*) Odstraněním množiny $B$ z grafu $G$ se přeruší všechny cesty mezi uzly $x$ a $y$, graf $G - B$ bude nesouvislý a uzly $x$ a $y$ budou ležet v různých komponentách.

(*ii*) Most je vlastně jednoprvkový hranový řez.

(*iii*) Zdůrazněme, že v definici řezu se požaduje minimalita – viz množiny silně vytažených hran na následujícím obrázku.



je řez

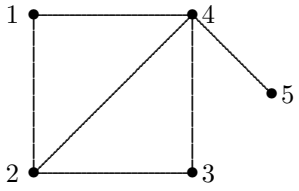


není řez - není minimální

**Definice 4.6.** *Nejmenší počet prvků hranového řezu mezi uzly $x$ a $y$ se nazývá* hranový stupeň souvislosti *grafu $G$ mezi uzly $x$ a $y$ a značí se $h_G(x, y)$.*

Jestliže očíslujeme uzly grafu čísly $1, \ldots, n$ a položíme $h_{i,j} = h_G(i, j)$, můžeme hranovou souvislost grafu plně popsat pomocí *matice hranové souvislosti* $H_G = [h_{i,j}]_{i,j=1}^{n}$ grafu $G$.

**Příklad.**




$$H_G = \begin{bmatrix} - & 2 & 2 & 2 & 1 \\ 2 & - & 2 & 3 & 1 \\ 2 & 2 & - & 2 & 1 \\ 2 & 3 & 2 & - & 1 \\ 1 & 1 & 1 & 1 & - \end{bmatrix}$$

Nyní si ukážeme analogické zobecnění pojmu artikulace.

**Definice 4.7.** *Buď $G$ souvislý graf, $x$, $y$ jeho uzly. Množina $A \subset U(G)$ taková, že*

*1) každá cesta z $x$ do $y$ obsahuje alespoň jeden uzel z množiny $A$,*

*2) žádná vlastní podmnožina množiny $A$ nemá vlastnost 1),*

*se nazývá* uzlový řez *grafu $G$ mezi uzly $x$ a $y$.*

**Poznámka.**

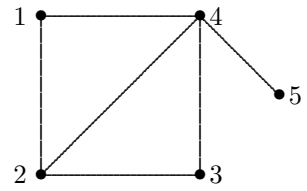
($i$) Odstraněním uzlového řezu dostaneme opět nesouvislý graf.

($ii$) Artikulace je jednoprvkový uzlový řez.

**Definice 4.8.** *Nejmenší počet prvků uzlového řezu, oddělujícího uzly $x$ a $y$, se nazývá* uzlový stupeň souvislosti *grafu $G$ mezi uzly $x$ a $y$ a značí se $u_G(x, y)$. Neexistuje-li uzlový řez mezi $x$ a $y$, tj. jsou-li uzly $x$ a $y$ sousední, klademe $u_G(x, y) = |U(G)| - 1$.*

Z uzlových stupňů souvislosti grafu můžeme analogicky sestavit matici uzlové souvislosti $U_G$ grafu $G$.

**Příklad.** Pro graf z předchozího příkladu dostáváme následující matici $U_G$.




$$U_G = \begin{bmatrix} - & 4 & 2 & 4 & 1 \\ 4 & - & 4 & 4 & 1 \\ 2 & 4 & - & 4 & 1 \\ 4 & 4 & 4 & - & 4 \\ 1 & 1 & 1 & 4 & - \end{bmatrix}$$

**Definice 4.9.** *Nejmenší z čísel $u_G(x, y)$ nazveme* uzlový stupeň souvislosti *grafu $G$ a budeme je značit $u(G)$. Nejmenší z čísel $h_G(x, y)$ nazveme* hranový stupeň souvislosti *grafu $G$ a budeme je značit $h(G)$. Řekneme, že graf $G$ je* uzlově (hranově) $k$-souvislý, *jestliže $u(G) \geq k$ $(h(G) \geq k)$.*

**Poznámka.**

(*i*) Často se pro jednoduchost používá pojem *k-souvislost* ve významu uzlové $k$-souvislosti. Pokud tedy v dalším textu bude řeč o hranové $k$-souvislosti, bude toto vždy explicitně uvedeno.

(*ii*) Pro nesouvislé grafy klademe $u(G) = h(G) = 0$ (příslušné řezy jsou pak prázdné množiny).

Symbolem $\delta(G)$ označíme *minimální stupeň grafu* $G$, tj. $\delta(G) = \min\{d(x)|x \in U(G)\}$.

**Věta 4.4.** *Pro každý graf $G$ platí*

$$u(G) \leq h(G) \leq \delta(G).$$

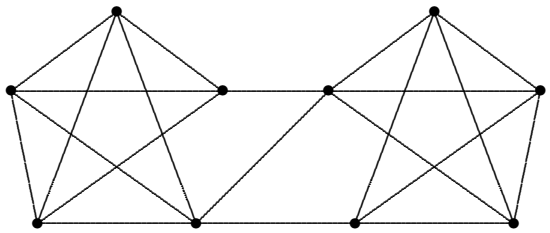
**Důkaz.** 1. Druhá nerovnost je zřejmá, neboť množina všech hran, obsahujících daný uzel, je hranovým řezem.

2. Dokážeme první nerovnost. Je-li $h(G) = 0$, pak zřejmě $u(G) = 0$, a je-li $h(G) = 1$, pak graf $G$ má most a tedy též $u(G) = 1$. Nechť tedy $h(G) > 1$, a nechť $B$ je minimální hranový řez, mající $k \geq 2$ hran. Odstraňme z $G$ některých $k - 1$ hran z řezu $B$. Tím vznikne graf s mostem $\{u, v\}$. Množinu $A \subset U(G)$ sestrojíme tak, že pro každou odstraněnou hranu vybereme jeden její uzel různý od uzlu $u$ i $v$ (množina $A$ má nejvýše $k - 1$ prvků).

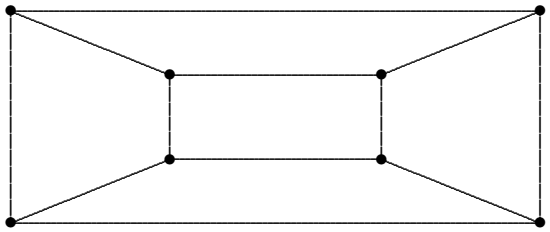
- Je-li graf $G'$, který vznikl odstraněním množiny $A$ z grafu $G$, nesouvislý, pak $u(G) \leq k - 1$.
- Je-li $G'$ souvislý, pak je to graf s mostem $\{u, v\}$. Ale graf $G$ má alespoň $k + 2$ uzlů (kdyby jich měl jen $k + 1$, tak by z rovnosti $h(G) = k$ plynulo, že se jedná o graf $K_{k+1}$ a $u(G) = k$), a tedy odstraněním $u$ nebo $v$ vznikne $k$-prvkový uzlový řez.

V každém případě tedy $u(G) \leq k$. □

**Příklad.** Následující dva grafy ukazují, že ve větě 4.4 mohou nastat rovnosti i ostré nerovnosti.



$u(G) = 2$, $h(G) = 3$, $\delta(G) = 4$



$u(G) = h(G) = \delta(G) = 3$

**Věta 4.5.** *V každém grafu $G$ platí*

$$h(G) \leq \frac{2|H(G)|}{|U(G)|}.$$

**Důkaz.** Platí známá rovnost $\sum d_G(u) = 2|H(G)|$. Protože $d(x) \geq \delta(G)$ pro každý uzel $x$, dostáváme $\sum d_G(u) \geq |U(G)|\delta(G)$, a porovnáním dostaneme $\delta(G) \leq \frac{2|H(G)|}{|U(G)|}$. Podle věty 4.4 je $h(G) \leq \delta(G)$. □

## 4.3 Charakterizační věty $k$-souvislých grafů

**Věta 4.6. (Ford, Fulkerson)** *Graf $G$ je hranově $k$-souvislý mezi uzly $a$ a $b$, $a \neq b$, právě když v něm existuje $k$ hranově disjunktních cest, vedoucích z $a$ do $b$.*

**Důkaz.** Důkaz provedeme převodem na problém maximálního toku v síti. Sestrojme síť $\vec{G}$ následujícím způsobem. Graf $\vec{G}$ je symetrická orientace grafu $G$, uzel $a$ je zdrojem a uzel $b$ stokem v síti $\vec{G}$, propustnosti všech hran jsou rovny jedné. V takto zkonstruované síti budeme hledat maximální tok.

1) Nechť je graf $G$ hranově $k$-souvislý mezi uzly $a$ a $b$. Pak neexistuje hranový řez o méně než $k$ hranách mezi těmito uzly v grafu $G$. Proto zároveň neexistuje takový hranový řez ani v $\vec{G}$. Podle Ford-Fulkersonovy věty o maximálním toku a minimálním řezu v síti $\vec{G}$ existuje celočíselný tok z uzlu $a$ do uzlu $b$ velikosti $k$. Protože propustnosti jsou rovny jedné, jsou hodnoty toku na všech hranách rovny 0 nebo 1. Množina hran s tokem velikosti 1 definuje požadovaný systém hranově disjunktních cest.
2) Nechť naopak v grafu $G$ existuje $k$ hranově disjunktních cest mezi uzly $a$ a $b$. Pak jsou to i cesty v síti $\vec{G}$. Po každé takové cestě v $\vec{G}$ pošleme tok o velikosti 1, vytvoříme tak tok velikosti $k$. Proto má každý hranový řez mezi uzly $a$ a $b$ alespoň $k$ hran (v síti $\vec{G}$ i v grafu $G$), a graf $G$ je tedy mezi uzly $a$ a $b$ hranově $k$-souvislý.

□

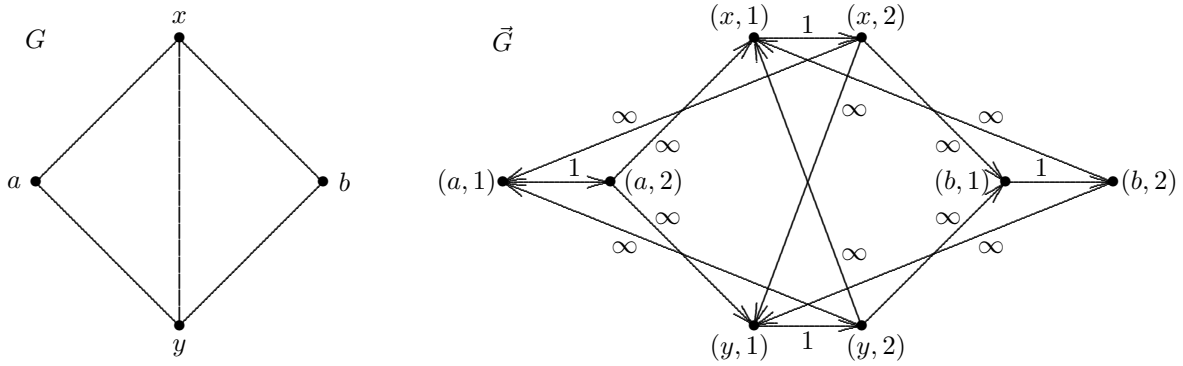
**Věta 4.7. (Menger)** *Graf $G$ je uzlově $k$-souvislý mezi nesousedními uzly $a$ a $b$, právě když v něm existuje $k$ uzlově disjunktních cest, vedoucích z $a$ do $b$.*

**Důkaz.** Zavedeme síť $\vec{G}$ následujícím předpisem:
$U(\vec{G}) = \{(x,i) |\ x \in U(G),\ i = 1,2\}$,
$H(\vec{G}) = \{((x,1),(x,2)) \mid x \in U(G)\} \cup \{((x,2),(y,1)) \mid \{x,y\} \in H(G)\}$.
(Názorně řečeno: každý uzel $x$ grafu $G$ rozdělíme na dva "polouzly" $(x,1)$ a $(x,2)$, a dále vytvoříme hrany dvojího druhu: hrany, spojující "půlky" uzlů, tj. $((x,1)\ (x,2))$, a hrany, vzniklé z hran v $G$, tj. hrany $((x,2)\ ,\ (y,1))$ ). Propustnosti hran zvolíme následujícím způsobem: u hran typu $((x,1)\ (x,2))$ položíme propustnost rovnu jedné a u hran druhého typu (tj. $((x,2),(y,1))$ ) bude propustnost nekonečná. Zdrojem je uzel $(a,2)$, stokem je uzel $(b,1)$. Pro takto sestrojenou síť bude důkaz analogický důkazu Ford-Fulkersonovy věty.

□

**Poznámka.** Myšlenky důkazů vět 4.6 a 4.7 mají samostatnou důležitost, neboť dávají algoritmy, umožňující určení hranové, resp. uzlové souvislosti grafu. Konstrukce z důkazu věty 4.7 (tj. "půlení uzlů") je standardní metoda, převádějící uzlovou souvislost na hranovou souvislost, umožňující použít aparát toků v sítích i na uzlovou souvislost. Pro vzniklé "půlky uzlů" se někdy používají termíny "vstupní polouzel" a "výstupní polouzel".

**Příklad.** Následující obrázek ilustruje konstrukci sítě $\vec{G}$ v důkazu věty 4.7.



# 5 Prohledávání grafů a algoritmy $k$-souvislosti

## 5.1 Algoritmus prohledávání grafu

Začneme obecným schématem prohledávání grafů, které je, jak uvidíme, užitečné jako základ mnoha algoritmů.

Naším úkolem je probrat systematicky všechny uzly a hrany grafu, ev. na nich provést nějaký úkol. Označme

$N$ množinu uzlů, které ještě nebyly probrány,
$M$ množinu hran, které ještě nebyly probrány,
$D$ množinu uzlů, kterých již bylo dosaženo, ale ze kterých ještě vedou neprobrané hrany.

**Algoritmus 5.1. (Prohledávání grafu)**

1. *$N := U(G)$, $M := H(G)$, $D := \emptyset$. (inicializace)*
2. *Je-li $N = \emptyset$, výpočet končí. (test ukončení)*
3. *Zvol $v \in N$ a polož $N := N \setminus \{v\}$, $D := \{v\}$. (volba prvního uzlu v komponentě)*
4. *Zvol libovolně $w \in D$. (volba počátečního uzlu hrany)*
5. *Hledej hranu v $M$ obsahující $w$: (test použitelnosti $w$)*
   - *neexistuje: $D := D \setminus \{w\}$, a*
     *je-li $D = \emptyset$, jdi na* **2**,
     *je-li $D \neq \emptyset$, jdi na* **4**.
   - *existuje: jdi na* **6**.
6. *Zvol $h = \{w, z\}$, "projdi" ji, (tj. proveď na ní určený úkol), a polož*
   *$M := M \setminus \{h\}$;*
   *je-li $z \in N$, pak $N := N \setminus \{z\}$, $D := D \cup \{z\}$,*

   *jdi na* **4**.

Snadno se ověří následující tvrzení.

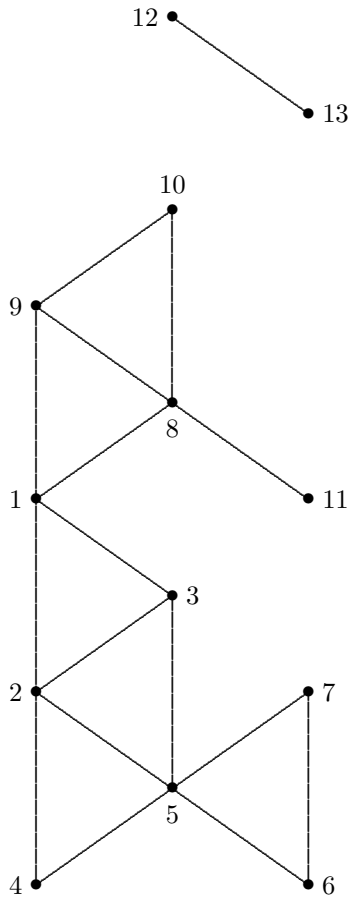
**Věta 5.1.** *Algoritmus 5.1 prohledá graf $G$ v čase $O(n+m)$, kde $n$ je počet uzlů a $m$ počet hran grafu $G$.*

**Poznámka.**

1. Bod **2** - test ukončení. Jedná se skutečně o ukončení algoritmu, neboť v tomto bodě je vždy $D = \emptyset$ a není tedy nic k dalšímu prohledávání.
2. V bodě 4 algoritmu jsou v zásadě tři možné implementace:
   - volbu provádět náhodně,
   - brát uzel, který je v množině $D$ nejdelší dobu, tj. interpretace množiny $D$ jako fronty,
   - brát uzel, který je v $D$ nejkratší dobu, tj. interpretace množiny $D$ jako zásobníku.

   Postup, který vznikne druhou interpretací ($D$ jako fronta) se nazývá *prohledávání do šířky*, poslední interpretace ($D$ jako zásobník) se nazývá *prohledávání do hloubky*. Oba tyto postupy mají svoji důležitost a, jak uvidíme, lze pomocí nich získat mnoho informací o zkoumaném grafu.

**Příklad.** Prohledávání grafu do šířky. Zde a v dalších podobných příkladech se pro jednoznačnost budeme řídit konvencí, podle níž v každém kroku, kdy je několik možností výběru uzlu, algoritmus zvolí uzel s nejmenším číslem.




| $D$ | $w$ | prohledávaná hrana |
|---|---|---|
| 1 | 1 | 1 2 |
| 1, 2 | 1 | 1 3 |
| 1, 2, 3 | 1 | 1 8 |
| 1, 2, 3, 8 | 1 | 1 9 |
| 1, 2, 3, 8, 9 | 1 | - |
| 2, 3, 8, 9 | 2 | 2 3 |
| 2, 3, 8, 9 | 2 | 2 4 |
| 2, 3, 8, 9, 4 | 2 | 2 5 |
| 2, 3, 8, 9, 4, 5 | 2 | - |
| 3, 8, 9, 4, 5 | 3 | 3 5 |
| 3, 8, 9, 4, 5 | 3 | - |
| 8, 9, 4, 5 | 8 | 8 9 |
| 8, 9, 4, 5 | 8 | 8 10 |
| 8, 9, 4, 5, 10 | 8 | 8 11 |
| 8, 9, 4, 5, 10, 11 | 8 | - |
| 9, 4, 5, 10, 11 | 9 | 9 10 |
| 9, 4, 5, 10, 11 | 9 | - |
| 4, 5, 10, 11 | 4 | 4 5 |
| 4, 5, 10, 11 | 4 | - |
| 5, 10, 11 | 5 | 5 6 |
| 5, 10, 11, 6 | 5 | 5 7 |
| 5, 10, 11, 6, 7 | 5 | - |
| 10, 11, 6, 7 | 10 | - |
| 11, 6, 7 | 11 | - |
| 6, 7 | 6 | 6 7 |
| 6, 7 | 6 | - |
| 7 | 7 | - |
| $\emptyset$ | | |

Nyní je sice množina $D$ prázdná, ale ještě nebyly probrány všechny uzly grafu. Provedeme tedy volbu nového uzlu, který zbyl v množině $N$ (krok 2).

| $D$ | $w$ | prohledávaná hrana |
|---|---|---|
| 12 | 12 | 12 13 |
| 12, 13 | 12 | - |
| 13 | 13 | - |
| $\emptyset$ | | |

Všimněme si, že tento postup probírá uzly postupně podle vzdálenosti od uzlu 1, tedy nejprve probere všechny uzly ve vzdálenosti 1, pak všechny uzly ve vzdálenosti 2, atd. Proto hovoříme o prohledávání do šířky.

**Příklad.** Vezmeme si tentýž graf, ale aplikujeme prohledávání do hloubky.

| $D$ | $w$ | prohledávaná hrana |
|---|---|---|
| 1 | 1 | 1 2 |
| 1, 2 | 2 | 2 3 |
| 1, 2, 3 | 3 | 3 1 |
| 1, 2, 3 | 3 | 3 5 |
| 1, 2, 3, 5 | 5 | 5 2 |
| 1, 2, 3, 5 | 5 | 5 4 |
| 1, 2, 3, 5, 4 | 4 | 4 2 |
| 1, 2, 3, 5, 4 | 4 | - |
| 1, 2, 3, 5 | 5 | 5 6 |
| 1, 2, 3, 5, 6 | 6 | 6 7 |
| 1, 2, 3, 5, 6, 7 | 7 | 7 5 |
| 1, 2, 3, 5, 6, 7 | 7 | - |
| 1, 2, 3, 5, 6 | 6 | - |
| 1, 2, 3, 5 | 5 | - |
| 1, 2, 3 | 3 | - |
| 1, 2 | 2 | - |
| 1 | 1 | 1 8 |
| 1, 8 | 8 | 8 9 |
| 1, 8, 9 | 9 | 9 1 |
| 1, 8, 9 | 9 | 9 10 |
| 1, 8, 9, 10 | 10 | 10 8 |
| 1, 8, 9, 10 | 10 | - |
| 1, 8, 9 | 9 | - |
| 1, 8 | 8 | 8 11 |
| 1, 8, 11 | 11 | - |
| 1, 8 | 8 | - |
| 1 | 1 | - |
| **12** | 12 | 12 13 |
| 12, 13 | 13 | - |
| 12 | 12 | - |
| $\emptyset$ | | |

Všimněme si, že kroky při prohledávání do hloubky jsou trojího typu:

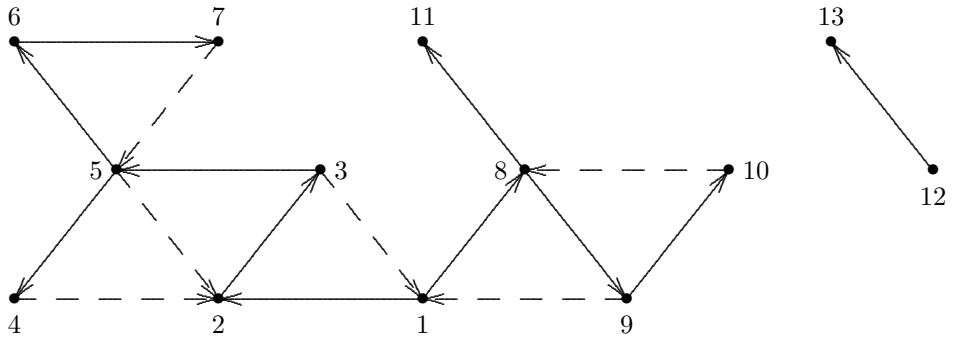
a) objevení nového uzlu,

b) nalezení hrany do již existujícího uzlu,

c) zpětný krok.

Hrany prohledávané při krocích a) tvoří strom zvaný *strom prohledávání* (pro nesouvislé grafy jde samozřejmě o les prohledávání). Hrany prohledávané při krocích b) se nazývají *chordy*. Je zvykem oba tyto druhy hran brát jako orientované, přičemž jejich orientace nám ukazuje směr prohledávání. Pak lze říci, že zpětné kroky jdou vždy po hranách stromu proti jejich orientaci. Podle zpětných kroků se algoritmu prohledávání do hloubky říká *backtracking*.

**Příklad.** Ukažme si strom prohledávání a chordy na grafu z předchozího příkladu. Hrany stromu jsou vyznačeny normální čarou, chordy jsou zobrazeny čárkovaně.




Prohledávání grafů lze použít pro řešení úloh souvisejících se souvislostí, komponentami apod. Se souvislostí a komponentami grafu je to jednoduché – každý průchod 3. krokem totiž určuje novou komponentu. Máme tedy ihned následující tvrzení.

**Tvrzení 5.1.** *Komponenty grafu lze nalézt* v *čase* $O(m+n)$.

## 5.2 Použití backtrackingu

### Artikulace grafu

V následujícím si ukážeme algoritmy na hledání artikulací v grafech. Tuto úlohu lze řešit i naivním přístupem, spočívajícím v postupném odstranění vždy jednoho uzlu a zkoumání, zda vzrostl počet komponent. Tento postup provádí $n$-krát backtracking a tedy má časovou složitost $O\left(n(m+n)\right)$. Jak si ukážeme, úlohu lze řešit podstatně rychleji jedním chodem backtrackingu, a tedy s časovou složitostí pouze $O(m+n)$.

Je-li uzel $v$ artikulace grafu $G$, pak *větev grafu* $G$ je maximální souvislá část $G$, v níž uzel $v$ není artikulací. Všimněme si následující vlastnosti backtrackingu: je zřejmé, že backtracking, jakmile do nějaké větve vejde, tak ji projde celou a pak ji teprve opustí. Nelze z takové větve vyjít jinudy, než právě artikulací a z vlastností backtrackingu ihned plyne, že větev bude probrána celá. Speciálně, vejde-li algoritmus založený na backtrackingu do koncového bloku, neopustí jej, dokud jej celý neprozkoumá. Této vlastnosti se využívá při hledání bloků a artikulací grafů. Otázkou však zůstává, jak při zpětném kroku poznáme, že jsme v artikulaci. Tarjan v roce 1972 navrhl následující postup.

1. Uzly očíslujeme pořadím, ve kterém byly zařazeny do stromu prohledávání, a tato čísla označíme $p(x)$.

2. Pro každý uzel $x$ souběžně postupně vypočítáváme *dolní číslo* $r(x)$ podle následujících pravidel:
   a) při objevení nového uzlu položíme $r(x) = p(x)$,
   b) při nalezení chordy $xy$ položíme $r(x) = \min\{p(y), r(x)\}$,
   c) při zpětném kroku z $x$ do $y$ položíme $r(y) = \min\{r(y), r(x)\}$.

Snadno je vidět, že číslo $r(x)$ udává nejmenší pořadové číslo uzlu, do něhož se lze z uzlu $x$ dostat orientovanou cestou, skládající se z hran stromu a (na konci) z právě jedné chordy. Nyní již lze poznat artikulaci podle následujícího pravidla.

Uzel $x$ s $p(x) > 1$ je artikulací, právě když existuje jeho následník $y$ ve stromu prohledávání takový, že platí $r(y) \geq p(x)$. Jinak řečeno, právě když existuje takový následník, z něhož se nelze dostat chordou do uzlu s nižším $p(u)$, než má uzel $x$.

Uvedli jsme, že tento postup lze použít i pro hledání bloků grafu. Chceme-li generovat bloky grafu, tak vždy, když při zpětném kroku nalezneme artikulaci $x$, dáme na výstup všechny hrany prozkoumané mezi prvním (dopředným) a druhým (zpětným) průchodem uzlu $x$ a tímto postupem nalezneme všechny bloky grafu.

#### Další modifikace backtrackingu

Artikulace a bloky jsme díky backtrackingu nalezli v čase $O(m + n)$. Existuje podobná metoda, která při stejné časové složitosti vyhledá kvazikomponenty a určí kondenzaci pro orientované grafy. Je opět založena na backtrackingu, konkrétně na jeho modifikaci pro grafy s orientovanými hranami. Určení kvazikomponent, resp. kondenzace, je sice možné i z distanční matice, ale samo nalezení distanční matice má časovou složitost $O(n^3)$.

## 5.3 Algoritmy zjišťování $k$-souvislosti

Backtracking lze využít i pro hledání biartikulací (dvouuzlových řezů) a 3-komponent (maximálních 3-souvislých podgrafů) a časová složitost je opět $O(m + n)$. Pro vyšší souvislosti grafů již podobné jednoduché postupy založené na backtrackingu nejsou známé. Vždy lze ale použít obecný postup, založený na převodu na problém nalezení maximálního toku v síti, který jsme poznali v důkazech Ford-Fulkersonovy a Mengerovy věty z kapitoly věnované mírám souvislosti grafu.

#### Hranová souvislost

Obecně je $h_G(a, b)$ rovna velikosti maximálního toku v síti, jež vznikne symetrickou orientací grafu $G$ s jednotkovou propustností všech hran, považujeme-li uzel $a$ za zdroj a uzel $b$ za stok. Touto metodou získáme řešení v čase $O(n^3)$ (s použitím nejrychlejšího známého – Dinicova – algoritmu na maximální tok). Chceme-li zjistit hranovou souvislost grafu, provedeme výše uvedený postup pro všechny dvojice uzlů a určíme tak $h(G)$ v čase $O(n^5)$. Nejlepší známé vylepšení (Even, Hopcroft, Tarjan) snižuje odhad časové složitosti na $O(m^2\sqrt{n})$, kde $m = |H(G)|$. Protože obecně platí $m \sim n^2$, časová složitost tohoto postupu je nejvýše $O(n^{4.5})$.

#### Uzlová souvislost

Uzlová souvislost mezi dvěma uzly se určí pomocí myšlenky z důkazu Mengerovy věty ("štěpení uzlů" a převod na toky). Časová náročnost tohoto postupu je, stejně jako v případě hranové souvislosti, $O(n^3)$ (odhady časové složitosti pro hranovou souvislost se jen násobí konstantou). Pro celý graf dostaneme složitost $O(n^{4.5})$, tedy stejnou, jako pro hranovou souvislost. Oba tyto postupy jsou nejlepší známé pro $k$-souvislost pro $k \geq 4$.

## 5.4 Backtracking pro generování hamiltonovských cest a cyklů

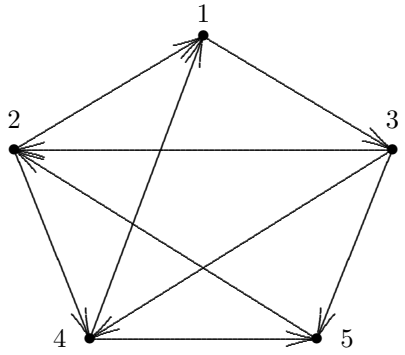
Prohledávání grafů umožňuje algoritmizovat i jiné problémy, kdy v grafu potřebujeme probírat celý strom prohledávání. Nyní si uvedeme příklad, ve kterém budeme modifikovat backtracking pro generování hamiltonovských cest a cyklů (tj. cest a cyklů procházejících všemi uzly grafu).

#### Modifikace backtrackingu

Obecné schema backtrackingu budeme modifikovat následujícími pravidly.

1) Nebereme v úvahu chordy,
2) při $|D| = n$ testujeme existenci hrany do počátečního uzlu; výsledkem je cesta (hrana neexistuje) nebo cyklus (hrana existuje),
3) při zpětném kroku se ptáme, zda existuje následník s vyšším číslem, jakmile jej ale nalezneme, tak při následném dopředném pohybu prozkoumáváme opět všechny uzly grafu, které nejsou v množině $D$.

**Příklad.** Mějme dán následující graf a hledejme hamiltonovské cesty a cykly.




| $D$ | $w$ | prohledávaná hrana |
|---|---|---|
| 1 | 1 | 1 3 |
| 1, 3 | 3 | 3 2 |
| 1, 3, 2 | 2 | 2 4 |
| 1, 3, 2, 4 | 4 | 4 5 |
| 1, 3, 2, 4, 5 | 5 | test $(5,1) \in H(\vec{G})$ |
| | | ne - cesta $1-3-2-4-5$ |
| 1, 3, 2, 4 | 4 | - |
| 1, 3, 2 | 2 | - |
| 1, 3 | 3 | 3 4 |
| 1, 3, 4 | 4 | 4 5 |
| 1, 3, 4, 5 | 5 | 5 2 |
| 1, 3, 4, 5, 2 | 2 | test $(2,1) \in H(\vec{G})$ |
| | | ano - cyklus $1-3-4-5-2-1$ |
| 1, 3, 4, 5 | 5 | - |
| 1, 3, 4 | 4 | - |
| 1, 3 | 3 | 3 5 |
| 1, 3, 5 | 5 | 5 2 |
| 1, 3, 5, 2 | 2 | 2 4 |
| 1, 3, 5, 2, 4 | 4 | test $(4,1) \in H(\vec{G})$ |
| | | ano - cyklus $1-3-5-2-4-1$ |
| 1, 3, 5, 2 | 2 | - |
| 1, 3, 5 | 5 | - |
| 1, 3 | 3 | - |
| 1, | 1 | - |
| $\emptyset$ | | KONEC |

Nalezli jsme jednu cestu a dva cykly. Kdybychom chtěli najít i cesty, začínající v jiných uzlech, tak bychom museli aplikovat algoritmus s příslušným výchozím uzlem. V tomto příkladu však jiné cesty neexistují. Nyní bychom mohli pro ohodnocený graf vyčíslit sumu cen hran na hamiltonovských cyklech a řešit úlohu obchodního cestujícího.

Tento algoritmus samozřejmě není efektivní. Backtracking sice pracuje v čase $O(m+n)$, ale hodnoty $m, n$ jsou hodnotami pro příslušný rozhodovací strom, nikoliv pro graf $G$. Je zřejmé, že počet uzlů rozdodovacího stromu obecně roste exponenciálně vzhledem k počtu uzlů grafu $G$.

Christofides uvádí pro podobné pokusy empirický vzorec, podle něhož za jednu sekundu je zpracován graf s přibližně 25 uzly a každé další zvýšení počtu uzlů o dva prodlouží výpočet více než na dvojnásobek původního času. Odtud vychází následující orientační tabulka časové náročnosti.

| počet uzlů | 50 | 60 | 70 | 80 |
|---|---|---|---|---|
| čas | 1.6 hodiny | 2.14 dne | 68.7 dne | 6 let |

## 5.5 Heuristiky pro hledání hamiltonovské kružnice

Heuristika je polynomiální postup, který neprobere všechny možnosti, ale v některých případech "umí" dát odpoveď. Má tzv. *orákulum*, které jí řekne, co probírat nemusí. Výsledek aplikace heuristiky je zpravidla buď kladná odpověď, nebo heuristika "neví" (tj. heuristika zpravidla neumí dát zápornou odpověď). I při odpovědi "nevím" však přesto může kladné řešení problému existovat. Uvedeme zde dvě heuristiky pro hledání hamiltonovské kružnice v grafu.

### Pósova heuristika

Pósova heuristika se snaží o přímočaré prodlužování cesty. Pokud to není možné, tedy pokud neexistuje hrana z koncového uzlu do nějakého volného, vezme hranu do některého uzlu cesty a provede modifikaci,

znázorněnou na následujícím obrázku.


Pósova heuristika tedy vybere uzel $x_1$, najde cestu do dalšího uzlu $x_2$ atd. Když dorazí do uzlu $z$, odkud přímé pokračování není možné, musí mít takový uzel hranu do některého z již prohledaných uzlů (např $x_i$). Odstraní se hrana $\{x_i, x_{i+1}\}$, přidá se $\{z, x_i\}$ a pokračuje se v uzlu $x_{i+1}$. Je však nutná ochrana proti zacyklení. Pokud tato heuristika nenalezne řešení, přečíslují se uzly a aplikuje se znovu. Testy této heuristiky ukázaly (i přes její jednoduchost) velmi dobré výsledky.

Jiná dobrá heuristika je založena na pojmu uzávěru grafu, majícím ovšem i samostatnou důležitost. Nejprve ale uvedeme několik tvrzení, potřebných pro zavedení pojmu uzávěru grafu.

**Věta 5.2. (Dirac)** *Nechť $G$ je graf na $n \geq 3$ uzlech. Je-li $\delta(G) \geq \frac{n}{2}$, pak je $G$ hamiltonovský.*

Diracova věta plyne ihned z následujícího obecnějšího tvrzení.

**Věta 5.3. (Ore)** *Jestliže pro všechny dvojice nesousedních uzlů $x, y$ grafu $G$ platí $d(x) + d(y) \geq n$, pak je graf $G$ hamiltonovský.*

Důkaz Oreho věty je založen na následujícím lemmatu.

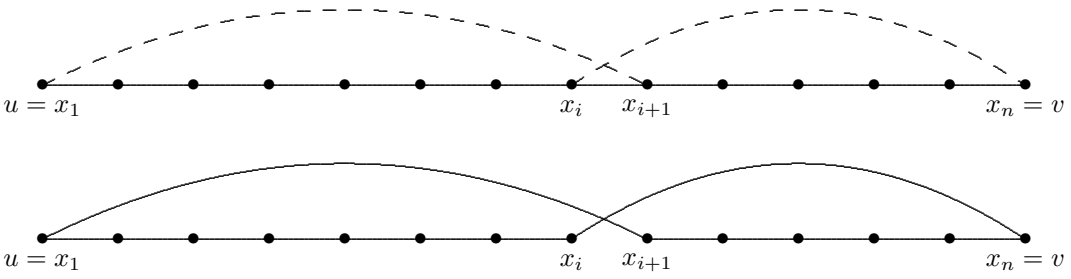
**Lemma 5.1. (Ore)** *Nechť $u, v$ jsou dva nesousední uzly grafu $G$ takové, že $d(u) + d(v) \geq n$. Pak $G$ je hamiltonovský, právě když graf $G + \{u, v\}$ (vzniklý přidáním hrany $\{u, v\}$ do $G$) je hamiltonovský.*

**Důkaz.**

1. Je-li $G$ hamiltonovský, je zřejmě $G + \{u, v\}$ též hamiltonovský.

2. Nechť naopak $G + \{u, v\}$ je hamiltonovský; chceme ukázat, že graf $G$ je rovněž hamiltonovský. V $G + \{u, v\}$ existuje hamiltonovská kružnice $C$. Není-li $\{u, v\}$ na $C$, pak je kružnice $C$ i v grafu $G$ a jsme s důkazem hotovi. Nechť tedy $\{u, v\} \in H(C)$. Odstraněním hrany $\{u, v\}$ získáme v grafu $G$ hamiltonovskou cestu $P$ mezi uzly $u$ a $v$. Očíslujme její uzly $u = x_1, x_2, \ldots, x_{n-1}, x_n = v$. Označíme:
   $$M = \{x_i |\ \{u, x_{i+1}\} \in H(G)\},$$
   $$N = \{x_i |\ \{x_i, v\} \in H(G)\}.$$
   Pak $|M| = d(u)$ a $|N| = d(v)$, a tedy podle předpokladu lemmatu je
   $$|M| + |N| = d(u) + d(v) \geq n.$$
   Je zřejmé, že uzel $v$ není v žádné z množin $M, N$, a tedy je nutně $|M \cup N| < n$. Proto je nutně $M \cap N \neq \emptyset$. Zvolme $x_i \in M \cap N$. Pak ale kružnice, určená posloupností uzlů $(u = x_1), x_2, \ldots, x_i, (v = x_n), x_{n-1}, \ldots, x_{i+1}, (x_1 = u)$ (viz obr.) je hamiltonovská kružnice v $G$.

□



Oreho věta se z tohoto lemmatu dokáže snadno. Protože podle předpokladu věty všechny dvojice nesousedních uzlů splňují podmínku $d(x) + d(y) \geq n$, můžeme přidat do grafu $G$ všechy chybějící hrany. Vzniklý úplný graf je samozřejmě hamiltonovský, a tedy je hamiltonovský i graf $G$.

Bondy s Chvátalem (1974) zjistili, že z Oreho lemmatu lze získat podstatně více.

**Definice 5.1.** Uzávěrem grafu $G$ *nazveme graf* $\mathrm{cl}(G)$*, který vznikne z grafu* $G$ *postupným rekurentním přidáváním všech hran* $\{u, v\}$*, splňujících podmínku Oreho lemmatu.*

Zdůrazněme zde, že při konstrukci uzávěru se v každém kroku před přidáním hrany $\{u, v\}$ testuje Oreho podmínka $d(u) + d(v) \geq n$ na aktuálním grafu (nikoliv tedy na původním grafu $G$). Tedy, uzávěr může být úplným grafem, i když v původním grafu $G$ zdaleka ne všechny dvojice nesousedních uzlů splňují Oreho podmínku. Snadným příkladem je graf $G = C_5 + h$ (tj. kružnice $C_5$ s jednou přidanou hranou), mající 5 uzlů, z nichž 3 jsou stupně 2, ale $\mathrm{cl}(G)$ je přesto úplný graf.

**Tvrzení 5.2.** *Pro každý graf* $G$ *je jeho uzávěr* $\mathrm{cl}(G)$ *určen jednoznačně.*

**Důkaz.** Nechť graf $G$ má dva uzávěry $G_1$ a $G_2$; nechť $G_1$ vznikl přidáním hran $e_1, e_2, \ldots, e_m$ a $G_2$ vznikl přidáním hran $f_1, f_2, \ldots, f_n$. Ukážeme, že každá hrana $e_i$ je obsažena v $G_2$ a každá hrana $f_i$ je v $G_1$. Důkaz provedeme sporem, budeme tedy předpokládat, že existuje hrana $\{u, v\} = e_{k+1}$, která není v grafu $G_2$ a všechny předcházející hrany $e_i, i \leq k$, v $G_2$ jsou. Označíme graf $H = G \cup \{e_1, \ldots, e_k\}$. Podle definice grafu $G_1$ musí být $d_H(u) + d_H(v) \geq n$, ale, podle volby hrany $e_{k+1}$, všechny předchozí hrany $e_i$, $i \leq k$ jsou v $G_2$. Graf $H$ je tedy podgrafem grafu $G_2$. Tedy i v grafu $G_2$ musí být splněna podmínka $d_{G_2}(u) + d_{G_2}(v) \geq n$. To znamená, že v $G_2$ existuje hrana $\{u, v\} = e_{k+1}$, což je spor.
Stejný způsobem bychom ukázali, že každá hrana $f_j$ je v grafu $G_1$, proto $G_1 = G_2$. □

**Věta 5.4. (Bondy, Chvátal)** *Nechť* $G$ *je graf s alespoň třemi uzly. Je-li* $\mathrm{cl}(G)$ *úplný graf, potom je graf* $G$ *hamiltonovský.*

**Poznámka.** Myšlenka důkazu Oreho lemmatu poskytuje polynomiální algoritmus na nalezení hamiltonovské kružnice v grafu $G$, jehož uzávěrem je úplný graf. Jak již bylo uvedeno dříve, obecně je problém existence hamiltonovské kružnice NP-úplný.

# 6 Nezávislost, dominance, klikovost a jádro grafu

## 6.1 Neorientované grafy

V tomto odstavci termín "graf" znamená neorientovaný graf.

**Definice 6.1.** *Množina* $A \subset U(G)$ *se nazývá* nezávislá množina *grafu* $G$ *(někdy "vnitřně stabilní"), jestliže žádné dva uzly množiny* $A$ *nejsou spojeny hranou.*

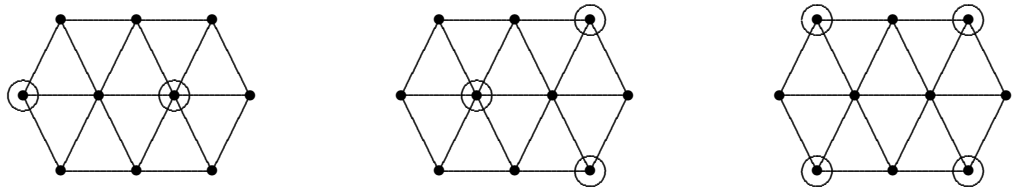
**Příklad.** Klasickým příkladem aplikace nezávislé množiny je tzv. úloha o vysílačích. Uzly grafu odpovídají televizním vysílačům, hrany spojují uzly-vysílače, které se vzájemně ruší (a tedy nemohou vysílat na stejném kanálu). Hledáme maximální množinu vysílačů, které se vzájemně neruší.

**Poznámka.** Je-li $A \subset G$ nezávislá a $A' \subset A$, pak $A'$ je také nezávislá. Má tedy smysl ptát se na maximální nezávislé množiny.

**Definice 6.2.** Nezávislost grafu $G$ *(značíme* $\alpha(G)$*) je největší počet prvků nezávislé množiny grafu* $G$*.*

**Poznámka.** Zdůrazněme: *největší*, nikoliv *maximální* ! V tom je zásadní rozdíl, jak nám ukazuje následující příklad.

**Příklad.**



Zakroužkované uzly jsou uzly patřící do nezávislých množin. Ve všech třech případech jsou tyto nezávislé množiny maximální, ale největší je jen v posledním případě. Tedy, $\alpha(G) = 4$.

**Poznámka.** V anglicky psané literatuře se maximální × největší rozlišuje jako *maximal* × *maximum* (takže například "největší nezávislá množina" se anglicky řekne "a maximum independent set").

**Poznámka.** I z hlediska algoritmické složitosti je rozdíl mezi hledáním největší a maximální nezávislé množiny velmi zásadní. Nalezení největší nezávislé množiny je NP-těžký problém (s jedinými známými postupy typu "hrubá síla" se složitostí $O(2^n)$), zatímco nalezení maximální nezávislé množiny je polynomiální problém, řešitelný snadno postupy typu hladového algoritmu.

**Definice 6.3.** *Množina $B \subset U(G)$ se nazývá (uzlové)* pokrytí *grafu $G$, jestliže pro každou hranu $\{x, y\} \in H(G)$ je $x \in B$ nebo $y \in B$ (mohou být i oba).*

**Poznámka.** Je-li $B$ pokrytí $G$ a $B' \supset B$, pak $B'$ je také pokrytí. Má tedy smysl hledat nejmenší pokrytí grafu $G$.

**Definice 6.4.** Pokrývací číslo *grafu $G$ (značíme $\beta(G)$) je počet prvků nejmenšího pokrytí grafu $G$.*

**Příklad.**

1. Dopravní úlohy, kontrola všech spojnic z minimálního počtu uzlů (policisté na křižovatkách, minimalizujeme jejich počet tak, aby kontrolovali všechny ulice).
2. Doplňky nezávislých množin z obrázků příkladu na minulé straně jsou pokrytí grafu $G$.

Předchozí příklad přímo motivuje následující větu.

**Věta 6.1.** *Množina $A \subset U(G)$ je nezávislá právě když $B = U(G) \setminus A$ je pokrytí $G$.*

**Důkaz.** $A$ je nezávislá $\Leftrightarrow$ žádná hrana nemá oba konce v $A \Leftrightarrow$ každá hrana má alespoň jeden konec v $B \Leftrightarrow B$ je pokrytí. □

**Důsledek 6.1.** *Pro každý graf $G$ platí*

$$\alpha(G) + \beta(G) = |U(G)|.$$

**Důkaz.** Je-li $A$ největší nezávislá množina v $G$, pak $U(G) \setminus A$ je nejmenší pokrytí. □

Následující věta ukazuje jednu zajímavou souvislost mezi nezávislostí a dalšímu vlastnostmi grafu.

**Věta 6.2. (Chvátal, Erdős)** *Nechť $G$ je $k$-souvislý graf ($k \geq 2$) s $\alpha(G) \leq k$. Pak je $G$ hamiltonovský.*

**Důkaz.** Předpokládejme, že $G$ není hamiltonovský, a nechť $C$ je nejdelší kružnice v $G$. Podle předpokladu $C$ není hamiltonovská kružnice v $G$, a tedy graf $G - U(C)$ má alespoň jednu neprázdnou komponentu $H$. Podle Mengerovy věty 4.7 [7] existuje $k$ uzlově disjunktních cest $P_1, \ldots, P_k$ tak, že $P_i$ má koncové uzly $x_i \in U(H)$ a $y_i \in U(C)$, a vnitřní uzly $P_i$ jsou mimo $U(H) \cup U(C)$, $i = 1, \ldots, k$. Označme $z_i \in U(C)$ následníka uzlu $y_i$ na $C$ (při nějaké pevně zvolené orientaci kružnice $C$), $i = 1, \ldots, k$, a nechť $z_0$ je libovolný uzel v $H$. Pak se snadno přesvědčíme, že množina $M = \{z_0, z_1, \ldots, z_k\}$ je nezávislá množina v $G$, protože v opačném případě bychom snadno sestrojili kružnici, která je delší než $C$. Pak ale $M$ je $(k+1)$-prvková nezávislá množina v $G$, což je spor s předpokladem $\alpha(G) \leq k$. □

Obdobnou technikou lze dokázat i následující větu, tematicky související s tvrzeními odstavce 5.5.

**Věta 6.3. (Dirac)** *Nechť $G$ je $k$-souvislý graf ($k \geq 2$). Pak pro každou množinu $M \subset U(G)$, $|M| \leq k$, existuje v grafu $G$ kružnice $C$ taková, že $M \subset U(C)$.*

**Důkaz.** Předpokládejme, že taková kružnice v $G$ neexistuje, zvolme $C$ tak, aby obsahovala co nejvíce prvků množiny $M$, a nechť $H$, $P_i$, $x_i$ a $y_i$ má stejný význam jako v důkazu věty 6.2. Protože $C$ neobsahuje všechny prvky $M$, lze $H$ zvolit tak, že existuje $z \in U(H) \cap M$. Uzly $y_1, \ldots, y_k$ dělí kružnici $C$ na $k$ cest a protože $|M| \leq k$, alespoň jedna z těchto cest neobsahuje žádný prvek množiny $M$. Pak lze ale snadno nalézt kružnici $C'$ takovou, že $(U(C) \cap M) \cup \{z\} \subset U(C')$, což je spor s volbou kružnice $C$. □

Pokusme se nyní hledat odpověď na otázku, která je v jistém smyslu opakem nezávislosti grafu: hledejme podmnožiny uzlů, které jsou *všechny* spojeny hranami.

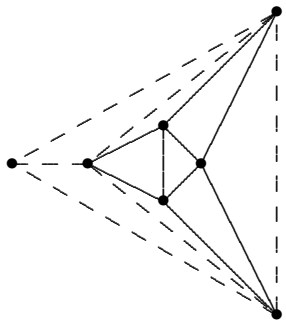
**Definice 6.5.** *Podgraf $K \subset G$ se nazývá* klika *grafu $G$, jestliže*

*a) $K$ je úplný podgraf*

*b) jestliže $K \subset K' \subset G$ a $K'$ je úplný podgraf, pak $K = K'$.*

Tedy, jinak řečeno, klika je maximální úplný podgraf.

**Příklad.** V tomto grafu je každý z plně vytažených trojúhelníků $K_3$ klikou (je maximální), ale není největší. Největší klika je $K_4$ - v obrázku je zakreslena čárkovaně.



**Definice 6.6.** Klikovost grafu *(značíme $\omega(G)$) je největší počet uzlů kliky grafu $G$.*

[7] Přesněji vzato – potřebujeme zde obecnější verzi Mengerovy věty, která říká, že v $k$-souvislém grafu existuje $k$ uzlově disjunktních cest i mezi každými dvěma disjunktními souvislými podgrafy. Důkaz tohoto tvrzení je analogický důkazu věty 4.7.

**Příklad.** Pro graf z minulého příkladu je $\omega(G) = 4$.

**Definice 6.7.** *Nechť $G$ je graf. Potom graf $\bar{G} = \left(U(G), \binom{U(G)}{2} \setminus H(G)\right)$ se nazývá* doplněk *grafu $G$.*

**Příklady:**

1) $G$: $\bar{G} = P_4$

2) $G = P_3$ $\bar{G} = P_3 \simeq G$

3) $G = C_5$ $\bar{G} = C_5 \simeq G$

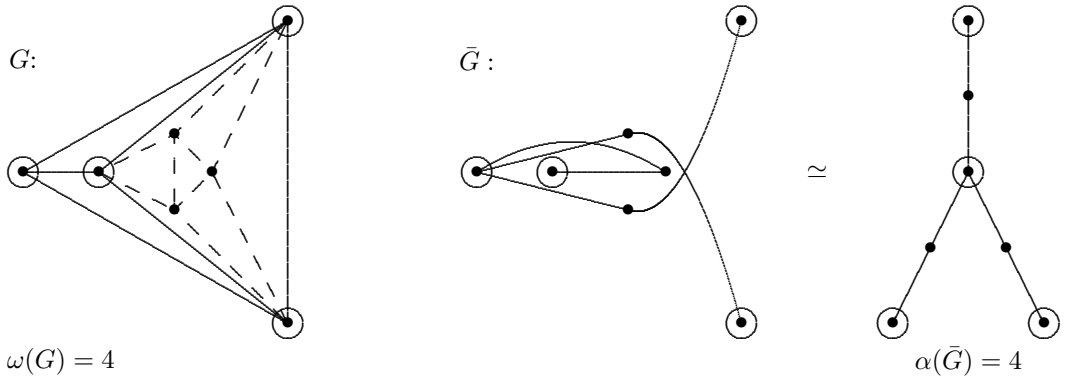
**Poznámka.** Pokud je graf izomorfní se svým doplňkem, hovoříme o autokomplementárním grafu (např. $P_3$, $C_5$). Např. na šesti uzlech však takový graf neexistuje (graf na šesti uzlech spolu se svým doplňkem má celkem $\binom{6}{2} = 15$ hran, což je lichý počet).

**Věta 6.4.** *Nechť $G$ je graf. Pak*

$$\alpha(G) = \omega(\bar{G}).$$

**Důkaz.** Množina $A \subset U(G)$ je nezávislá, právě když žádné dva její uzly nejsou spojeny hranou, tedy právě když v $\bar{G}$ jsou každé dva její uzly spojeny hranou. Odtud plyne, že $A$ je maximální nezávislá množina, právě když $A$ tvoří v grafu $\bar{G}$ kliku. □

**Příklad.**




**Definice 6.8.** *Množina $B \subset U(G)$ se nazývá* dominantní množina *(též vnějškově stabilní) grafu $G$, jestliže pro každý uzel $x \notin B$ existuje uzel $y \in B$ takový, že $\{x, y\} \in H(G)$.*

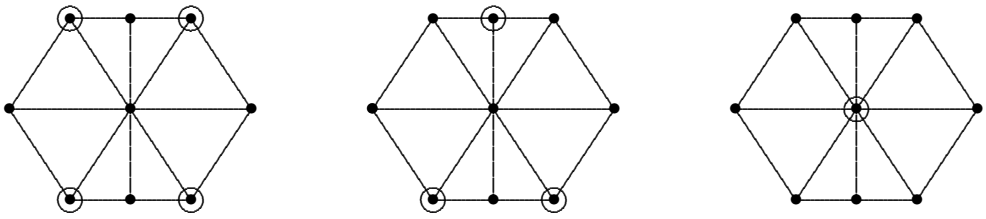
Jinak řečeno, $B$ je dominantní, jestliže každý uzel leží v $B$, nebo v ní má souseda.

**Poznámka.** Je-li $B$ dominantní a $B' \supset B$, pak $B'$ je také dominantní. Má tedy smysl ptát se na minimální a na nejmenší dominantní množinu.

**Definice 6.9.** *Počet prvků nejmenší dominantní množiny grafu $G$ se nazývá* číslo dominance *grafu $G$ a značí se $\gamma(G)$.*

**Příklad.** Opět je potřeba důsledně rozlišovat mezi minimální a nejmenší dominantní množinou:

Minimální Minimální Nejmenší $\gamma(G) = 1$



**Věta 6.5.** *Nechť $A \subset U(G)$ je nezávislá množina grafu $G$. Pak $A$ je maximální nezávislá množina, právě když $A$ je dominantní.*

**Důkaz.** Nechť $A$ je nezávislá množina.

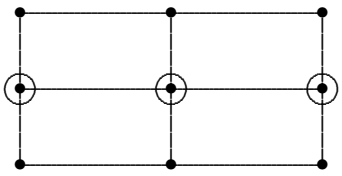
1) Je-li $A$ dominantní, potom každý uzel mimo $A$ má v $A$ souseda, z čehož vyplývá, že žádný takový uzel již nelze přidat do $A$ při zachování nezávislosti. $A$ je tedy maximální.
2) Je-li $A$ je maximální nezávislá množina, pak do ní nelze přidat žádný uzel při zachování nezávislosti. Odtud plyne, že každý uzel nepatřící do $A$ má v $A$ souseda, a tedy $A$ dominantní.

□

**Věta 6.6.** *Každá maximální nezávislá množina je minimální dominantní množina grafu $G$.*

**Důkaz.** Podle předchozí věty je maximální nezávislá množina dominantní. Je tedy třeba dokázat, že je minimální dominantní. Kdyby nebyla minimální, tak v ní existuje uzel $x$ takový, že $A \setminus \{x\}$ je dominantní. Potom uzel $x$ je sousedem nějakého uzlu z množiny $A$. Ale to znamená, že $A$ by už nebyla nezávislá, což je spor. □

**Poznámka.** Implikaci ve větě nelze obrátit. Minimální dominantní množina totiž nemusí být nezávislá (viz obrázek).

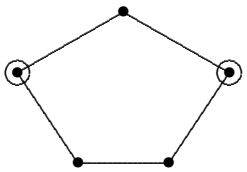


**Důsledek 6.2.** *Pro každý graf $G$ platí* $\gamma(G) \leq \alpha(G)$.

**Důkaz.** Plyne z předchozího. □

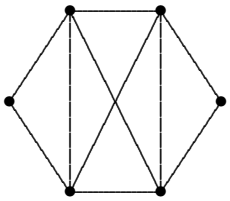
**Příklady.** Následující příklady ukazují, že v nerovnosti v důsledku 6.2 může nastat rovnost i ostrá nerovnost.

1)



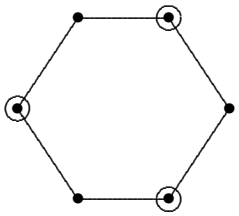
$\alpha(G) = \gamma(G) = 2$

2)

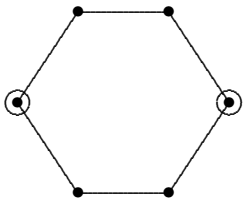


$\alpha(G) = \gamma(G) = 2$

3)

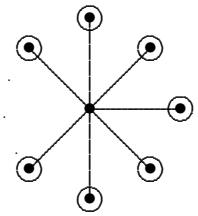


$\alpha(G) = 3$

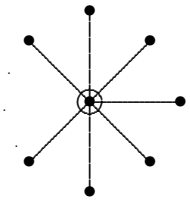


$\gamma(G) = 2$

4)



$\alpha(G) = n$



$\gamma(G) = 1$

**Definice 6.10.** *Množina uzlů $C \subset U(G)$, která je současně nezávislá i dominantní, se nazývá* jádro *grafu $G$.*

**Věta 6.7.** *Každý neorientovaný graf má jádro.*

**Důkaz.** Jádrem je vlastně každá maximální nezávislá množina, protože podle věty 6.6 je i dominantní. □

**Poznámka.** Připomeňme, že rozhodnutí, zda v daném grafu $G$ existuje nezávislá množina dané velikosti, je klasický NP-úplný problém, a tudíž nejsou známy žádné efektivní algoritmy, umožňující nalezení čísla $\alpha(G)$. (Podrobněji se těmito otázkami budeme zabývat v kapitole 9.) Naproti tomu, nalezení maximální nezávislé množiny je snadné hladovým algoritmem (zvolíme libovolný uzel, odstraníme jej i s jeho sousedy a ve zbylém grafu postup opakujeme). Tento postup je jednak polynomiálním algoritmem pro nalezení jádra neorientovaného grafu, ale zároveň heuristikou, poskytující dolní odhad čísla $\alpha(G)$.

Situace je obdobná s klikovostí, dominancí i pokrývacím číslem. Vždy je možno snadno najít maximální, resp. minimální množiny v polynomiálním čase, zatímco příslušná otázka pro největší, resp. nejmenší množiny je NP-těžká.

## 6.2 Jádro orientovaného grafu

**Definice 6.11.** *Buď $\vec{G}$ orientovaný graf. Říkáme, že množina $A \subset U(\vec{G})$ je* nezávislá, *jestliže žádné dva její uzly nejsou spojeny hranou.*

**Poznámka.** Definice nezávislé množiny je tedy shodná pro orientované i neorientované grafy.

**Definice 6.12.** *Buď $\vec{G}$ orientovaný graf, $C \subset U(\vec{G})$. Množina $C$ se nazývá* jádro *grafu $\vec{G}$, jestliže*

*(i) $C$ je nezávislá množina,*

*(ii) pro každý $x \in U(\vec{G}) \setminus C$ existuje $y \in C$ tak, že $(x, y) \in H(\vec{G})$.*

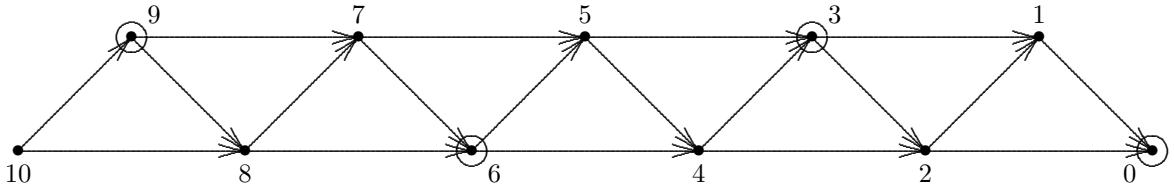
Názorně – podmínka *(ii)* říká, že z každého uzlu mimo jádro existuje hrana do jádra.

**Příklad.** Je-li $\vec{G}$ cyklus liché délky, pak $\vec{G}$ nemá jádro. To se nahlédne snadno: zvolíme-li (symetrie) první uzel jádra libovolně, pak z *(i)* a *(ii)* vyplývá, že v $C$ leží "každý druhý" uzel cyklu, ale poslední uzel nelze zahrnout do $C$ ani nechat mimo $C$.

Jedna z motivací pojmu jádra pochází z teorie her. Uvedeme si dva jednoduché příklady.

**Příklad.** Na hromádce je deset zápalek, hráči střídavě ubírají jednu nebo dvě. Vyhrává ten, který vezme poslední zápalku.

Pozicím hry (počet zápalek, které zůstaly ve hře) přiřadíme uzly grafu a všechny možné tahy nám vytvoří hrany grafu (viz následující obrázek).

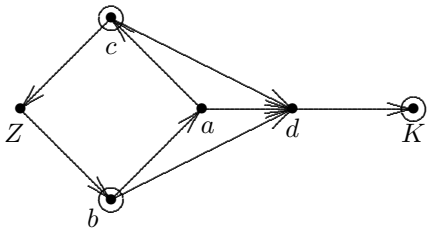



Uzly patřící do jádra jsou opět zakroužkovány.

Hráč, který první táhne do jádra, má možnost při jakémkoliv tahu soupeře táhnout zpět do jádra (2. vlastnost), ale soupeř tuto možnost nemá (1. vlastnost). Hráč, který začíná, tedy nemůže prohrát (samozřejmě, neudělá-li chybu).

**Obecně:** Jestliže graf, odpovídající hře, má jádro, pak hráč, který první táhl do jádra, má strategii, zaručující to, že neprohraje (může ovšem dojít k remíze - při pohybu v cyklu).

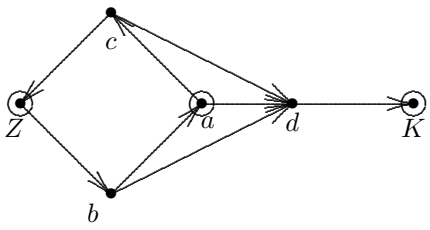
**Příklad.**




$Z$ – začátek hry, $K$ – konec hry. Vyhrává ten, který první táhne do $K$.

Začínající hráč A si nalezne jádro → táhne do jádra. Protihráč B: kdyby na $d$, tak A vyhraje. Tedy na $a$. A: kdyby na $d$, vyhraje B. Tedy na $c$. B: kdyby na $d$, vyhraje A. Tedy na $Z$, čímž si vynutí remízu cyklem.

To je tím, že B má také tah do jádra, zvolíme-li jiné jádro, jaké nám ukazuje následující obrázek:




**Otázka:** Kdy má orientovaný graf jádro? Například lichý cyklus jádro nemá, zatímco každá symetrická orientace neorientovaného grafu ano.

**Věta 6.8.** *Každý acyklický graf má jádro.*

**Důkaz.** Očíslujme uzly podle věty o acyklických grafech a definujme funkci $f : U(\vec{G}) \to \mathbf{N}_0$ (ohodnocení uzlů celými nezápornými čísly):

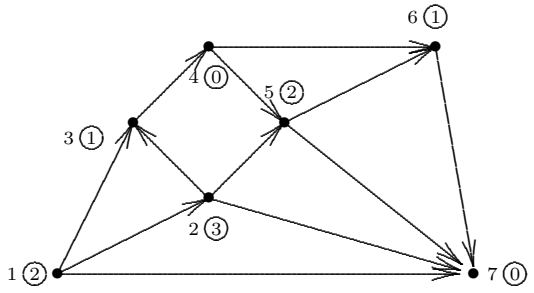
$f(u_n) = 0$

$f(u_i) = \min\{k \in \mathbf{N}_0;\ k \notin \{f(v_j);\ (u_i, v_j) \in H(\vec{G})\}\},\ \ i = n-1, \ldots, 1.$

Tato funkce se nazývá *Grundyho funkce.*

Označme $C = \{x \in U(\vec{G});\ f(x) = 0\}$. Pak $C$ je jádro. □

**Příklad.** Na následujícím obrázku jsou hodnoty Grundyho funkce jednotlivých uzlů zakroužkovány.




Platí ještě více.

**Věta 6.9.** *Každý orientovaný graf bez cyklů liché délky má jádro.*

**Důkaz.** (myšlenka). Jádro grafu bez lichých cyklů lze nalézt následujícím postupem.

1. Nechť $\vec{G}_0$ je výstupní kvazikomponenta grafu $\vec{G}$ (tj. kvazikomponenta, odpovídající výstupnímu uzlu kondenzace). Zvolme libovolně $x \in U(\vec{G}_0)$ a položme

$$S_0 = \{y \in U(\vec{G}_0)|\ \text{ z } y \text{ do } x \text{ existuje orientovaný sled sudé délky}\}.$$

(Poznamenejme, že sled nulové délky je také sudý sled, a tedy $x \in S_0$.) Díky předpokladu neexistence lichých cyklů je $S_0$ jádrem $\vec{G}_0$.

2. Je-li $\vec{G}_0 = \vec{G}$ (tj. $\vec{G}$ je silně souvislý), jsme hotovi. V opačném případě položíme

$$\vec{G}_1 = \vec{G} - (S_0 \cup \{y \in U(\vec{G})|\ \exists z \in S_0 \text{ tak, že } (y, z) \in H(\vec{G})\}),$$

a pro graf $\vec{G}_1$ celý postup opakujeme.

3. Skončí-li uvedený postup po $k$ krocích, je jádro $S$ grafu $\vec{G}$ sjednocením takto nalezených množin: $S = S_0 \cup S_1 \cup \ldots S_k$. □

**Poznámka.** Myšlenky důkazů vět 6.8 a 6.9 dávají zároveň algoritmy pro nalezení jádra (v těchto speciálních případech).

Poznamenejme ještě, že charakterizační věta orientovaných grafů s jádrem není známa, a otázka existence jádra v obecném orientovaném grafu je NP-úplný problém.

# 7 Barevnost grafu

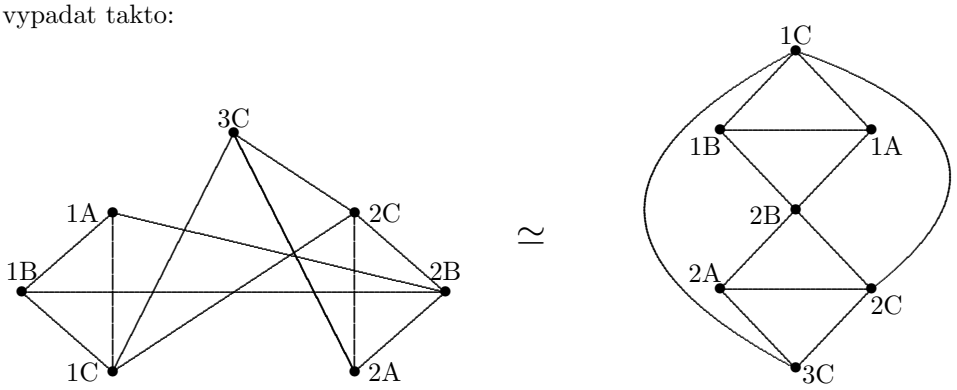
**Příklad. (rozvrh hodin)** Je dán kurs, obsahující $M$ přednášek, přičemž některé přednášky nemohou probíhat současně (stejný učitel, stejní studenti, stejná specializovaná laboratoř apod.) Úkolem je zjistit, v jakém nejkratším čase lze odpřednášet celý kurs.

| | A | B | C |
|---|---|---|---|
| 1 | U | U | W |
| 2 | V | U | V |
| 3 | | | V |

U, V, W ... učitelé
A, B, C ... předměty
1, 2, 3 ... třídy

Předmět C má speciální učebnu

Strukturu výuky popíšeme grafem tak, že jednotlivým vyučovacím předmětům přiřadíme uzly grafu, a dvojicím předmětů, které nelze přednášet ve stejnou dobu, přiřadíme hrany grafu. Graf této úlohy pak bude vypadat takto:




Uzlům přidělíme vyučovací hodiny tak, že sousední uzly nebudou mít stejná čísla (hrany grafu říkají, že dané přednášky nesmí probíhat současně). Po chvíli pokusů zjistíme, že – překvapivě – daný kurs nelze narozvrhovat do 3 hodin; potřebujeme alespoň 4 vyučovací hodiny.

**Příklad. (síť vysílačů)** Vytvoříme následující graf: uzly grafu odpovídají vysílačům TV, hrany spojují dvojice vysílačů, které se navzájem ruší a tedy nemohou vysílat na stejném kanálu. Cílem je nalézt minimální počet kanálů pro vysílání.

## 7.1 $k$-obarvitelnost a chromatické číslo grafu

**Definice 7.1.** *Graf $G$ se nazývá* $k$-obarvitelný, *jestliže každému jeho uzlu lze přiřadit jednu z "barev" $1 \dots k$ tak, že žádné dva sousední uzly nemají stejnou barvu.*

**Poznámka.** Každý graf je $|U(G)|$-obarvitelný (každý uzel jinou barvou).

**Definice 7.2.** *Nejmenší přirozené číslo $k$, pro které je graf $G$ $k$-obarvitelný, se nazývá* chromatické číslo (barevnost) *grafu $G$ a značí se $\chi(G)$.*

**Příklad.** Graf z příkladu o rozvrhu hodin má barevnost 4.

**Poznámka.** Úloha nalezení chromatického čísla grafu je ekvivalentní úloze rozložení množiny uzlů grafu na minimální počet nezávislých podmnožin.

**Příklad.**
Je-li $G$ kružnice sudé délky, pak $\chi(G) = 2$.
Je-li $G$ kružnice liché délky, pak $\chi(G) = 3$.
Je-li $G$ strom, pak $\chi(G) = 2$.
Je-li $G$ úplný graf, pak $\chi(G) = n$.

**Tvrzení 7.1.** *Nechť $G$ obsahuje jako podgraf úplný graf $K_k$. Pak $\chi(G) \geq k$.*

**Důkaz.** $K_k$ má chromatické číslo $k$, $G$ tedy musí mít chromatické číslo $\geq k$. □

**Poznámka.** Graf z příkladu o rozvrhu hodin ukazuje, že ve větě může platit ostrá nerovnost. Tento graf neobsahuje jako podgraf žádný $K_4$, ale přesto $\chi(G) = 4$.

Horní odhad chromatického čísla dává následující tvrzení.

**Věta 7.1.** *Pro každý graf $G$ platí $\chi(G) \leq \Delta(G) + 1$, kde $\Delta(G)$ je maximální stupeň grafu $G$.*

**Důkaz.** Indukcí podle $n = |U(G)|$.

1. pro $n = 1$ je tvrzení zřejmé: $\Delta(G) = 0$ a $\chi(G) = 1$.
2. nechť tvrzení platí pro každý graf s $n-1$ uzly a $G$ má $n$ uzlů. Sestrojme $G'$ odstraněním libovolného uzlu $u$. Pak $G'$ má $n - 1$ uzlů a maximální stupeň $\leq \Delta(G)$; podle indukčního předpokladu je obarvitelný $\Delta(G) + 1$ barvami. Uzel $u$ má stupeň nejvýše $\Delta(G)$ a tedy jeho sousedi mají nejvýše $\Delta(G)$ barev – lze jej tedy obarvit tou z $\Delta(G) + 1$ barev, kterou žádný z jeho sousedů nemá.

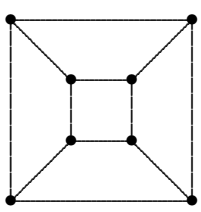
□

Tato věta se dá ještě zesílit:

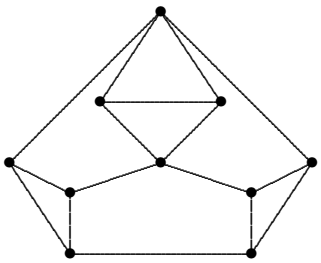
**Věta 7.2.** **(Brooks)** *Pro každý graf $G$ platí $\chi(G) \leq \Delta(G)$ až na tyto dvě výjimky:*
*I. $G$ má komponentu $K_{(\Delta(G)+1)}$,*
*II. $\Delta(G) = 2$ a $G$ má za komponentu kružnici liché délky.*

**Důsledek 7.1.** *Je-li $G$ souvislý graf, který není úplným grafem ani kružnicí liché délky, pak $\chi(G) \leq \Delta(G)$.*

**Příklad.**



$\chi(G) = 2$
$\Delta(G) = 3$



$\chi(G) = \Delta(G) = 4$

Přirozenou otázkou je, kdy má graf malou barevnost.
První krok je triviální: $\chi(G) = 1 \iff H(G) = \emptyset$.
Naproti tomu, grafy barevnosti 2 již tvoří důležitou a poměrně bohatou třídu.

**Věta 7.3.** *$\chi(G) = 2$ právě když $H(G) \neq \emptyset$ a $G$ neobsahuje kružnici liché délky.*

**Důkaz.**

1. Má-li $G$ kružnici liché délky, pak $G$ zřejmě není 2-obarvitelný.
2. Nechť $G$ nemá kružnici liché délky, indukcí podle počtu kružnic dokážeme 2-obarvitelnost.
   (*i*) Jestliže $G$ nemá kružnici, pak je to strom a $\chi(G) = 2$.
   (*ii*) Nechť každý graf bez lichých kružnic s nejvýše $k-1$ kružnicemi je 2-obarvitelný; $G$ má $k$ sudých kružnic. Odstraňme hranu $\{x, y\}$ některé kružnice $C$, pak zbylý graf je podle indukčního předpokladu 2-obarvitelný; protože $C$ je sudá, mají $x$ a $y$ různé barvy a obarvení lze přenést na graf $G$.

□

**Poznámka.** Každý graf barevnosti 2 je podgrafem některého úplného bipartitního grafu $K_{m,n}$. (Důkaz: každou monochromatickou třídu uzlů grafu $G$ umísti do jedné partity grafu $K_{n,m}$). Proto se jim říká *bipartitní grafy*. Je zřejmé, že o tom, zda je daný graf bipartitní, je možno rozhodnout v polynomiálním čase (například hladovým algoritmem).

Pro $\chi(G) \geq 3$ je to ale už horší, o čemž nás přesvědčí následující věta:

**Věta 7.4.** *Úloha: "určete, zda je daný graf $G$ 3-obarvitelný" je NP-úplná.*

**Důkaz.** Větu dokážeme v kapitole 9. □

**Poznámka.** Úloha 3-obarvitelnosti je NP-úplná dokonce i pro rovinné grafy; dokonce i pro rovinné grafy s $\Delta(G) \leq 4$ (které jsou 4-obarvitelné podle Brooksovy věty).

**Poznámka.** Určení barevnosti grafu je zřejmě ekvivalentní s nalezením rozkladu $U(G)$ na minimální počet nezávislých množin. Odtud plyne následující tvrzení:

**Věta 7.5.** *Pro graf $G$ s chromatickým číslem $\chi(G)$ a nezávislostí $\alpha(G)$ platí:*

1. $\chi(G)\alpha(G) \geq |U(G)|$,
2. $\chi(G) + \alpha(G) \leq |U(G)| + 1$.

**Důkaz.**

1. V optimálním obarvení grafu $G$ $\chi(G)$ barvami je každá monochromatická třída uzlů nezávislou množinou o nejvýše $\alpha(G)$ prvcích.
2. Největší nezávislou množinu obarvíme 1. barvou a zbylé uzly každý jinou barvou různou od té první. Pak $\alpha(G)$ + počet barev $= |U(G)| + 1$ ; ale pravděpodobně existuje lepší obarvení grafu $G$, proto $\leq$.

□

**Poznámka.** Nalezení $\chi(G)$ je ovšem NP-těžká úloha. Myšlenka druhé části důkazu poskytuje následující heuristiku:

- hledej v $G$ nezávislou množinu,
- obarvi ji 1 barvou a vyhoď,
- ve zbytku grafu postup opakuj.

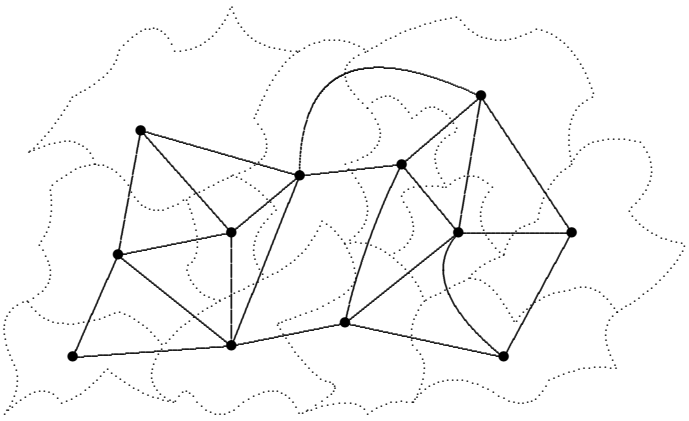
Heuristika, přes svoji jednoduchost, je používaná a dává slušné výsledky.

Jiná používaná heuristika, tzv. "sekvenční barvení", je založena na myšlence důkazu věty 7.1. V každém kroku barví uzel nejnižší barvou, kterou ještě nemá žádný z jeho sousedů (varianty jsou ve volbě barveného uzlu: náhodně, od největších stupňů apod.)

## 7.2 Barvení map

**Příklad.** Pokusíme se obarvit mapu tak, jak je zvykem na tzv. "politických mapách", tj. tak, aby sousední státy měly různé barvy. Sousedními státy přitom rozumíme ty, které spolu sousedí v nekonečně mnoha bodech. Snažíme se samozřejmě minimalizovat počet použitých barev.

Barvení mapy se dá převést na určení barevnosti (duálního) rovinného grafu (viz obrázek). Rovinnost grafů zatím budeme uvažovat jen intuitivně, později se k jejich definici vrátíme.



V daném případě k obarvení potřebuji 4 barvy. Dá se ukázat, že 4 barvy také vždy stačí:

**Věta 7.6.** *Každý rovinný graf je 4-obarvitelný.*

Tato věta dává řešení jednoho z nejproslulejších problémů celé historie teorie grafů – tzv. *Problému čtyř barev.*

V rovině tedy stačí 4 barvy. Jak je tomu na plochách vyšších rodů, nám ukazuje následující věta:

**Věta 7.7. (Heawoodův vzorec)** *Je-li $G$ uložitelný na plochu rodu $\gamma$, pak*

$$\chi(G) \leq \left[\frac{7+\sqrt{1+48\gamma}}{2}\right].$$

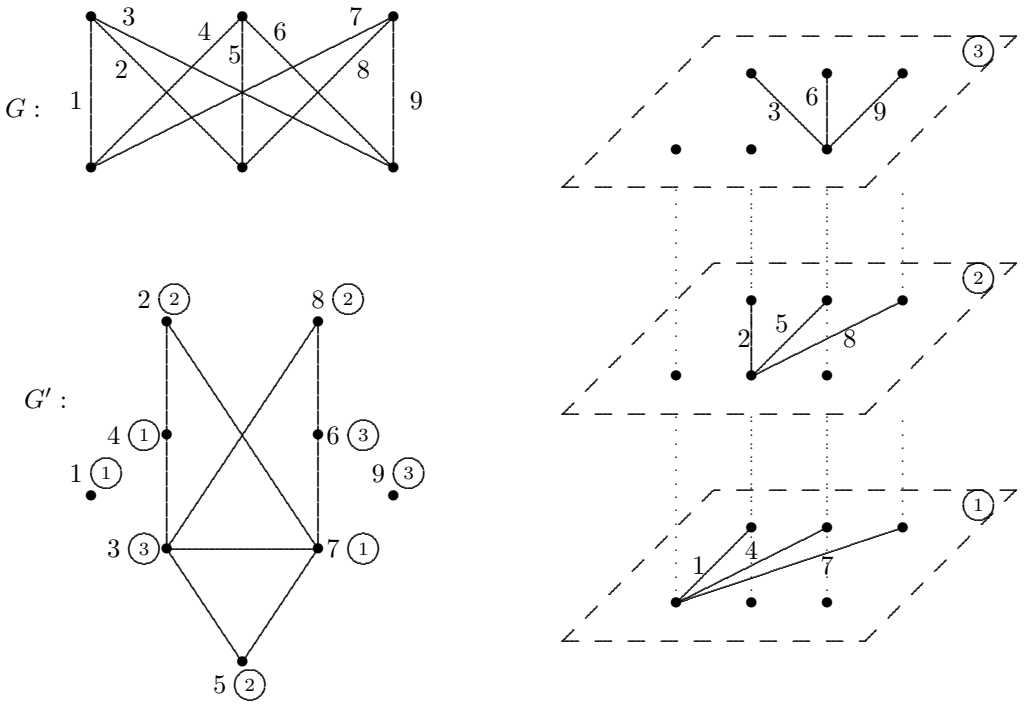
Vzorec uvedl Heawood už roku 1890, důkaz byl kompletně dokončen (pro rod 0, tj. rovinu) až v roce 1968.

Tento odstavec zakončíme příkladem jedné aplikace barvení grafu.

**Příklad. (rozklad grafu na vrstvy rovinných grafů)**
Nechť je dán graf $G$ a jeho nakreslení v rovině (při němž se některé hrany mohou "protínat" mimo uzel). Očíslujeme hrany grafu $G$ a sestrojíme graf $G'$ tak, že uzly $G'$ odpovídají hranám $G$ a dva uzly $G'$ jsou sousední právě když odpovídající hrany grafu $G$ se protínají.

Nalezneme-li obarvení grafu $G'$, pak každá monochromatická třída uzlů v $G'$ odpovídá množině neprotínajících se hran grafu $G$, tj. rovinnému podgrafu. To znamená, že jsme $H(G)$ rozložilli na $\chi(G')$

rovinných podgrafů. Ukážeme si to na příkladu grafu $K_{3,3}$ na následujícím obrázku (čísla v kroužcích znamenají barvy uzlů v obarvení grafu $G'$).




Samozřejmě, při praktickém použití metody je třeba nejdříve nakreslit graf $G$ s pokud možno malým počtem průsečíků hran a pak teprve konstruovat $G'$. Pak lze graf $K_{3,3}$ (dokonce i $K_6$ nebo $K_{4,4}$) rozložit jen na dvě vrstvy.

## 7.3 Hranové barvení a rozklady grafů

Hranové obarvení grafu lze do jisté míry považovat za hranovou analogii nám již známého pojmu uzlového obarvení.

**Definice 7.3.** Hranové obarvení *grafu $G$ je zobrazení $H(G)$ do* **N**. Dobré hranové obarvení *grafu $G$ je takové hranové obarvení grafu $G$, při němž žádné dvě hrany téže barvy nemají společný uzel. Graf $G$ je* hranově $k$-obarvitelný, *jestliže existuje jeho hranové obarvení $k$ barvami.* Chromatický index $\chi'(G)$ *grafu $G$ je nejmenší číslo $k$, pro něž je graf $G$ hranově $k$-obarvitelný.*

**Poznámka.** Někdy se místo "dobré hranové obarvení" používá termín "přípustné hranové obarvení" (angl. "proper edge coloring").

**Příklad.** Z Brooksovy věty víme, že chromatické číslo grafu $G$ je (až na dvě výjimky) shora omezeno maximálním stupněm $\Delta(G)$. Naproti tomu, graf $G = K_{n,n}$ má $\Delta(G) = n$, ale $\chi(G) = 2$ – pro *uzlové* obarvení tedy neexistuje dolní mez, která by byla funkcí $\Delta(G)$. Následující věta ukazuje, že u hranového obarvení je situace odlišná.

**Věta 7.8. (Vizing)** *Pro každý graf $G$ platí*

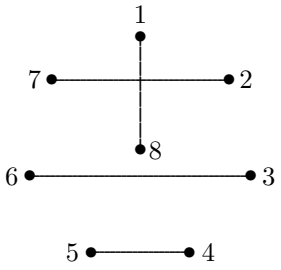
$$\Delta(G) \leq \chi'(G) \leq \Delta(G) + 1.$$

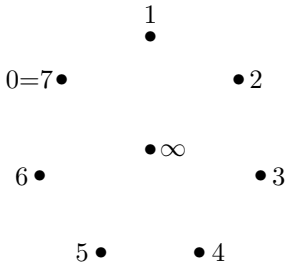
**Důkaz.** První nerovnost je zřejmá, druhou nebudeme dokazovat. □

**Příklad.** Určete číslo $\chi'(K_8)$.

Podle Vizingovy věty je $\chi'(K_8) \geq \Delta(K_8) = 7$. Ukážeme, že dobré obarvení 7 barvami existuje.

*Metoda první – geometrická.* Očíslujeme uzly grafu $G$ čísly $1, \ldots, 8$, nakreslíme uzel 8 "dovnitř" a spojíme dle obrázku (levá část). Vzniklý obrázek sedmkrát rotujeme, přičemž při každé rotaci dáme hranám další barvu. Sjednocením vytvořené množiny hran dostaneme graf $K_8$, hranově obarvený 7 barvami. Povšimněme si toho, že každá monochromatická třída hran tvoří ve vzniklém grafu párování (v tomto případě perfektní).




*Metoda druhá – algebraická.* Pracujeme v aritmetice modulo 7. Uzly očíslujeme podle obrázku (pravá část). Chromatické třídy jsou tvořeny následujícími množinami hran.

| třída | hrany | | | |
|---|---|---|---|---|
| 1 | $\{1,\infty\}$ | $\{2,0\}$ | $\{3,6\}$ | $\{4,5\}$ |
| 2 | $\{2,\infty\}$ | $\{3,1\}$ | $\{4,0\}$ | $\{5,6\}$ |
| 3 | $\{3,\infty\}$ | $\{4,2\}$ | $\{5,1\}$ | $\{6,0\}$ |
| 4 | $\{4,\infty\}$ | $\{5,3\}$ | $\{6,2\}$ | $\{0,1\}$ |
| 5 | $\{5,\infty\}$ | $\{6,4\}$ | $\{0,3\}$ | $\{1,2\}$ |
| 6 | $\{6,\infty\}$ | $\{0,5\}$ | $\{1,4\}$ | $\{2,3\}$ |
| 7 | $\{0,\infty\}$ | $\{1,6\}$ | $\{2,5\}$ | $\{3,4\}$ |

Povšimněme si toho, že při této konstrukci další chromatické třídy dostaneme z první postupným přičítáním jedné.

Definujeme-li v předchozím příkladu *délku hrany* $\{i,j\}$ předpisem $\min\{i-j, j-i\} \pmod 7$, pak v každé chromatické třídě jsou hrany všech různých délek $1, 2, 3, \infty$. Toto pozorování umožňuje naši konstrukci zobecnit: barvíme hrany grafu $K_{2n}$, pracujeme v aritmetice modulo $2n-1$, délka hrany je číslo $\min\{i-j, j-i\} \pmod{2n-1}$, délky hran v každé chromatické třídě jsou $1, 2, \ldots, n, \infty$, a vše funguje obdobně. Tím je dokázána část následujıcího tvrzení.

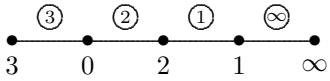
**Věta 7.9.** $\chi'(K_{2n-1}) = \chi'(K_{2n}) = 2n-1$.

**Důkaz.** a) Pro $K_{2n}$ je věta už dokázána v předchozím příkladu (našli jsme obarvení $2n-1$ barvami a lépe to podle Vizingovy věty nejde).

b) Podle Vizingovy věty je $\chi'(K_{2n-1}) \leq 2n-1$ (příslušné obarvení dostaneme ihned např. tak, že z $K_{2n}$ s obarvením z předchozího příkladu vynecháme jeden uzel). Zbývá dokázat, že $K_{2n-1}$ nelze obarvit $\Delta(G) = 2n-2$ barvami. Nechť naopak takové obarvení existuje. Pak je množina $H(K_{2n})$ rozložena na $2n-2$ monochromatických tříd, a v každé z nich je nejvýše $n-1$ hran. Ale počet hran grafu $K_{2n-1}$ je $|H(K_{2n-1})| = \binom{2n-1}{2} = \frac{(2n-1)(2n-2)}{2} = (2n-1)(n-1) > (2n-2)(n-1)$, což je spor. □

Jak jsme již poznamenali, je-li graf $K_{2n}$ obarven $2n-1$ barvami, pak každá monochromatická třída hran je 1-faktorem (perfektním párováním). Dostáváme tak *rozklad grafu* $K_{2n}$ *na* $2n-1$ *1-faktorů*. Tuto konstrukci rozkladu grafu $K_{2n}$ na izomorfní grafy lze dále zobecnit.

**Příklad.** Graf $K_8$ lze rozložit na 7 kopií grafu $P_4$. Rozklad dostaneme analogickým postupem z jedné kopie $P_4$ s ohodnocením dle obrázku postupným přičítáním jedné (ovšemže v aritmetice modulo 7). Čísla v kroužcích znamenají délky hran.




Z tohoto příkladu je ihned vidět myšlenka důkazu následujícího tvrzení (zde symbolem $\ell(h)$ značíme délku hrany $h$).

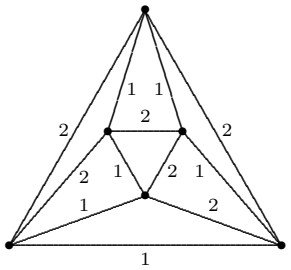
**Věta 7.10.** *Buď $G$ graf o $n$ hranách. Existuje-li ohodnocení uzlů grafu $G$ čísly $0, 1, \ldots, n-1, \infty$ takové, že*

*(i) pro každé dvě hrany $h_1, h_2 \in H(G)$ je $\ell(h_1) \neq \ell(h_2)$,*
*(ii) $\{\ell(h_1), \ldots, \ell(h_n)\} = \{1, \ldots, n-1, \infty\}$,*
*pak existuje rozklad grafu $K_{2n}$ na $2n-1$ kopií grafu $G$.*

**Poznámka.** Uzlové ohodnocení s vlastnostmi z věty 7.10 se nazývá "graceful labelling" (česky snad "půvabné ohodnocení"). Jedna ze známých a dosud otevřených hypotéz říká, že každý strom má půvabné ohodnocení.

Uvedeme si ještě jeden příklad podobné obecné konstrukce rozkladu grafu.

**Příklad.** Máme graf osmistěnu a pokoušíme se o jeho rozložení na dva faktory 2. stupně. Řešení je na následujícím obrázku (čísla 1 a 2 u hran nám udávají, ve kterém faktoru se hrana nalézá).




Úspěšně jsme tedy graf 4. stupně rozložili na dva podgrafy 2. stupně. Jde to vždy? Odpověď nám dá následující věta.

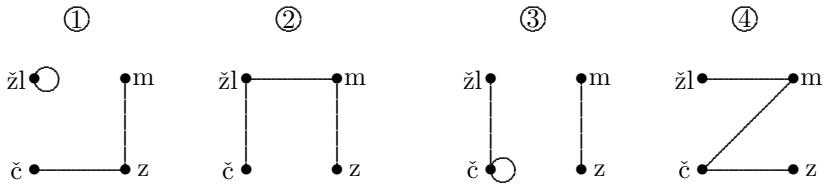
**Věta 7.11.** *Každý pravidelný graf 4. stupně se dá rozložit na dva kvadratické (= pravidelné 2. stupně) faktory.*

**Důkaz.** $G$ je eulerovský; sestrojíme tedy eulerovský tah a jeho hrany očíslujeme střídavě 1 a 2. Toto očíslování udává příslušnost hran k faktorům. □

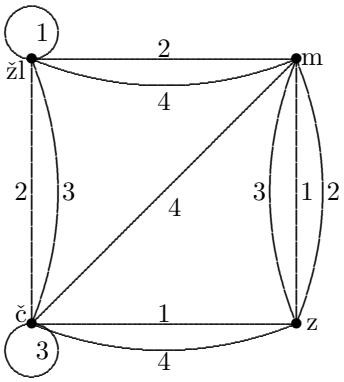
**Příklad.** Na závěr odstavce si ukážeme, jak lze rozklad grafu použít na řešení jednoho klasického hlavolamu.

Máme 4 kostky se stěnami, obarvenými čtyřmi barvami (žlutou, modrou, červenou a zelenou). Úkolem je sestavit kostky do sloupečku tak, aby na žádné straně nebyly dvě stěny stejné barvy.

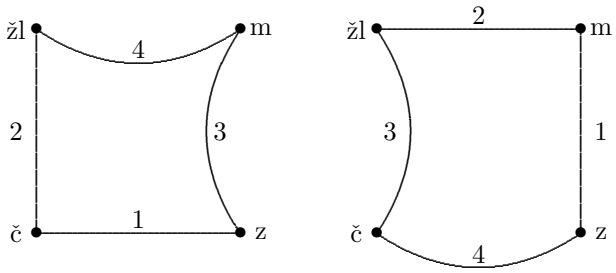
Po sestavení bude každá barva jednou na každé stěně vzniknuvšího hranolu. Pro každou kostku nakreslíme graf, v němž barvy (uzly) jsou spojeny hranou právě tehdy, jsou-li na protilehlých stěnách.




Tyto grafy poskládáme do jediného grafu s tím, že ohodnotíme hrany čísly podle toho, z kterého grafu byly použity.




Úkolem je nalézt dva hranově disjunktní podgrafy, které jsou pravidelné 2. stupně a mají hrany všech čtyř původních grafů. Tyto podgrafy ukazuje následující obrázek.




Z těchto grafů již sestavíme řešení úlohy.

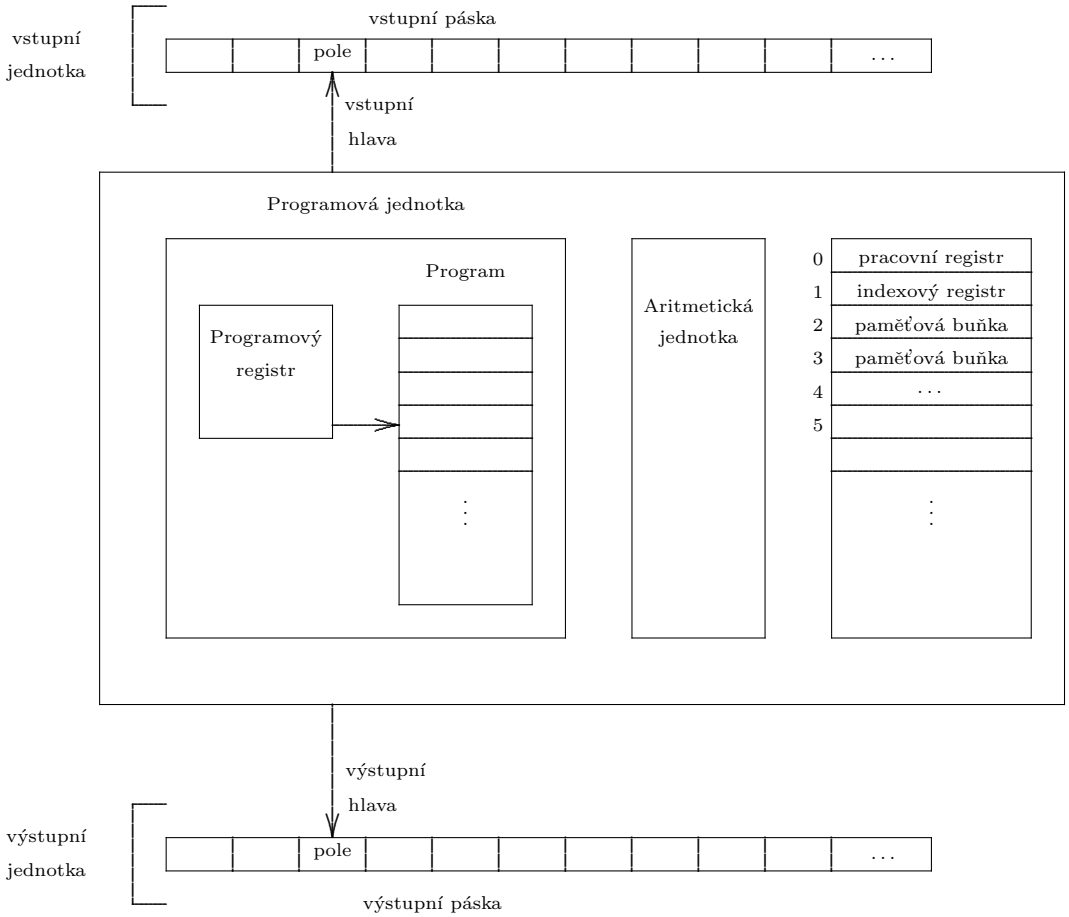
# 8 Modely výpočtu

V této a následující kapitole využijeme dosud poznané pojmy a poznatky z teorie grafů jako aparát, na němž vybudujeme základy teorie výpočetní složitosti. Zejména dáme přesný obsah pojmu NP-úplnosti (který jsme dosud chápali spíše intuitivně), a závěrem i NP-úplnost některých základních problémů dokážeme.

Chceme-li precizovat pojmy jako "časová náročnost výpočtu", "polynomiální problém", "efektivní algoritmy" a podobně, musíme nejprve najít vhodný abstraktní model pro samotný výpočet, popsaný algoritmem. Je mnoho možných definic algoritmu (např. Turingův počítač, rekurzivní funkce, program v některém programovacím jazyku apod.). Tzv. "Churchova teze" říká, že každá úloha, algoritmizovatelná podle některé definice, je algoritmizovatelná i podle všech ostatních definic.

## 8.1 Počítač s libovolným přístupem

My budeme používat jako model výpočtu tzv. *počítač s libovolným přístupem*, jehož model je zobrazen na následujícím obrázku. Jeho popis je sice komplikovanější než např. u Turingova počítače, ale tato nevýhoda je kompenzována mnohem snazší prací s modelem (který je bližší reálnému počítači). V anglické literatuře bývá tento model označován zkratkou RAM ("Random Access Machine").




V polích vstupní i výstupní pásky jsou celá čísla libovolně velká.
Paměťových buněk je neomezený počet a lze do nich vkládat neomezeně velká čísla.
Čísla před registry a paměťovými buňkami udávají adresu.

Nadále pojem "počítač" nebo "stroj" znamená vždy tento abstraktní model. Přehledně uvedeme definice základních pojmů, týkajících se počítače s libovolným přístupem.

*Konfigurace počítače:* přiřazení, které každému
- poli vstupní pásky
- poli výstupní pásky
- paměťové buňce
- programovému registru

přiřazuje celé číslo (= popis okamžitého stavu počítače).

*Počáteční konfigurace:* existuje $n$ takové, že
- pole vstupní pásky s adresami $n$, $n+1$, ...
- všechna pole výstupní pásky
- všechny paměťové buňky

obsahují nuly a programový registr má hodnotu 1. [8]

[8] Tedy na polích vstupní pásky s adresami $0, \ldots, n-1$ jsou vstupní data.

*Výpočet počítače:* posloupnost konfigurací $C_0$, $C_1$, ... taková, že $C_0$ je počáteční konfigurace, a *krok* je dán některým z *příkazů* (viz dále).

*Program počítače:* konečná posloupnost $p_1$, ... , $p_n$ příkazů.

*Příkazy:*
- *přesuny v paměti*
  - LOAD *operand* do pracovního registru uloží hodnotu operandu (ostatní nezměněno)
  - STORE *operand* do paměť. buňky s adresou rovnou adrese operandu uloží obsah prac. registru
- *aritmetické příkazy*
  - ADD *operand* k obsahu pracovního registru se přičte hodnota operandu
  - SUBTRACT *operand* od obsahu prac. registru se odečte hodnota operandu
  - MULTIPLY *operand* obsah pracovního registru se násobí hodnotou operandu
  - DIVIDE *operand* obsah prac. registru se dělí hodnotou operandu
- *vstupy, výstupy*
  - READ do prac. registru dá obsah aktuálního pole vstupní pásky a posune hlavu o 1 vpravo
  - WRITE obsah prac. registru uloží do aktuálního pole výstupní pásky a posune hlavu o 1 vpravo
- *skoky*
  - JUMP *návěští* uloží návěští do programového registru
  - JZERO *návěští* provede příkaz JUMP, pokud je obsah pracovního registru roven nule
  - JGE *návěští* provede příkaz JUMP, pokud obsah prac. registru je $\geq 0$

  Návěštím rozumíme číslo instrukce či příkazu programu, tj. přirozené číslo.
- *zastavení*
  - STOP ukončí výpočet
  - ACCEPT ukončí výpočet, u rozhodovacích úloh má pravdivostní hodnotu 1
  - REJECT totéž jako ACCEPT, ale dává hodnotu 0

*Způsoby zadání operandu:*
- $j$ ($j$ je přirozené číslo nebo nula) - adresa operandu je $j$, hodnota operandu je obsah buňky s adresou $j$.
- $*j$ ($j$ je celé číslo) - adresa operandu je $i+j$, kde $i$ je obsah indexového registru, hodnota je obsah buňky s adresou $i+j$.
- $=j$ ($j$ je celé číslo) - hodnota je $j$, adresa není definována.

*Adresovací chyba:* nastane, když se u operandu $*j$ objeví okamžitá hodnota $i+j$ záporná. Výpočet se v takovémto případě zastaví.

## 8.2 Časová a paměťová náročnost výpočtu

**Definice 8.1.** *Řekneme, že výpočet počítače* trval dobu $t$, *jestliže*
- *v $t$-tém kroku došlo k:*
  - *provedení příkazu zastavení, nebo*
  - *adresovací chybě, nebo*
  - *dělení nulou,*
- *v krocích* $0, 1, \ldots, t-1$ *žádný z uvedených případů nenastal.*

*Řekneme, že výpočet počítače* pracoval s pamětí velikosti $m$, *jestliže*
- *nebyl proveden příkaz s adresou* $> m$,
- *byl proveden příkaz s adresou* $= m$.

**Poznámka.** To znamená, že používáme tzv. *uniformní hodnocení* paměti, tj. takové, při němž se nepřihlíží k obsahu buněk a jeho velikosti.

Toto hodnocení je zpravidla lepší než logaritmické (při němž se při odhadu doby, potřebné pro provedení příkazu, vychází z délky binárního zápisu čísla, s nímž se operuje), má však háček: fakt, že nepřihlížíme k velikosti obsahu buněk, lze využít ke "zrychlení" tím, že např. řádek matice uložíme jako jedno číslo. Tato "úspora" je samozřejmě jen zdánlivá. Aby však nedocházelo k podobným nepřesnostem, dohodneme se takto:

**Omezení 1.** *Nechť $p$ je pevně daný polynom. Připouštíme jen výpočty, pro něž v žádné buňce paměti není číslo v absolutní hodnotě větší než $p(\max\{n, |c_1|, |c_2|, \ldots, |c_n|\})$, kde $c_1, \ldots, c_n$ jsou vstupní data.*

Kdyby nastal případ, že toto omezení je příliš silné, rozdělíme velká čísla do několika buňek a přesuny v paměti provedeme posloupností příkazů.

Jak ale vyřešíme omezení rozsahu paměti? Zřejmě platí, že počet buněk paměti nebude větší než počet kroků výpočtu. Ale rozsah paměti je dán největší adresou, nikoliv počtem buněk, není tedy záruka, že uvedená nerovnost bude splněna. Platí ale následující věta.

**Věta 8.1.** *Nechť $f$ je funkce a $M$ počítač, který každou vstupní posloupnost délky $n$ zpracuje v čase $f(n)$. Pak existuje počítač $M'$, který zpracuje týž vstup v čase $O\left((f(n))^2\right)$ a v paměti $O(f(n))$ a dá týž výstup.*

**Důkaz.** (myšlenka)
Počítač $M'$ kopíruje výpočet počítače $M$ s tím rozdílem, že kdykoliv $M$ ukládá číslo $c$ na adresu $a$, počítač $M'$ ukládá do dvojice prvních volných paměťových buněk číslo $a$ a $c$ (technické detaily simulace nejsou pro nás až tak důležité, nebudeme se jimi blíže zabývat). Jasné je, že:

- $M'$ pracuje v paměti $O(f(n))$, neboť ukládá nejvýše dvojici údajů pro každý krok počítače $M$,
- $M'$ pracuje v čase $O\left((f(n))^2\right)$, neboť potřebuje $O(f(n))$ kroků a provede je nejvýše $f(n)$ krát.

□

**Důsledek 8.1.** *Jestliže existuje počítač, který každou vstupní posloupnost délky $n$ zpracuje v polynomiálním čase, pak existuje počítač, který každou vstupní posloupnost zpracuje v polynomiálním čase i paměti.*

Tedy - při definici polynomiality se stačí omezit na *časovou* složitost.

## 8.3 Problémy (jazyky) třídy P

**Definice 8.2.** Vstupními daty *nebo* slovem *nazveme konečnou posloupnost nul a jedniček.* Délkou slova *rozumíme počet členů posloupnosti vstupních dat.* Jazykem *nazveme konečnou (nebo i nekonečnou) množinu slov.*

**Definice 8.3.** Přijímací počítač *je počítač, který má následující dvě vlastnosti:*

(*i*) *jeho program neobsahuje příkazy WRITE ani STOP,*

(*ii*) *pro každé slovo $w$ se výpočet zastaví po konečném počtu kroků provedením příkazů ACCEPT nebo REJECT (aniž by došlo k adresovací chybě nebo dělení nulou).*

*Řekneme, že přijímací počítač* přijímá *slovo $w$, pokud se výpočet zastaví příkazem ACCEPT* , a odmítá *slovo $q$, pokud výpočet skončí příkazem REJECT.*
*Množina přijímaných slov se nazývá* jazyk přijímaný počítačem.

**Poznámka.** Smysl právě zavedeného pojmu je v tom, že přijímací počítač je abstraktním modelem algoritmu, řešícího rozhodovací problém (např. existence hamiltonovské kružnice v daném grafu).

**Definice 8.4.** *Nechť $J$ je jazyk a $f : \mathbf{N} \to \mathbf{N}$. Časová (paměťová) složitost jazyka $J$ je nejvýše $f$, jestliže existuje přijímací počítač $M$, který přijímá $J$ a každé slovo jazyka $J$ délky $n$ zpracuje v čase (paměti) $f(n)$.*

**Definice 8.5.** Třída P *je třída všech jazyků $J$, pro něž existuje polynom $p$ takový, že časová složitost jazyka $J$ je nejvýše $p$.*

**Příklady** jazyků třídy P: posloupnosti vstupních dat odpovídající:
- souvislým grafům,
- $k$-souvislým grafům $\forall k \geq 1$,
- acyklickým orientovaným grafům, ...

**Poznámka.** Jestliže pojem přijímacího počítače je abstraktním modelem algoritmu, řešícího rozhodovací problém, pak pojem jazyka je modelem tohoto problému. Zdůrazněme zde, že při zkoumání příslušnosti jazyka ( = problému) do některé z tříd musí jít vždy o rozhodovací problém (např. "je daný graf souvislý?"). Optimalizační problémy musíme nejprve na rozhodovací úlohu převést. Např. místo úlohy nalezení minimální kostry musíme uvažovat rozhodovací problém zda existuje kostra ceny $\leq k$ pro nějaké předem pevně dané $k$.

# 9 Teorie NP-úplnosti

Začneme zdánlivě jinde - u logických formulí.

## 9.1 Logické formule

Pro účely tohoto odstavce vystačíme s nejjednodušší definicí booleovské proměnné a logické formule.
*Logická (booleovská) proměnná* je proměnná, která nabývá hodnot 0 (false) a 1 (true).
Definice *logické formule* je konstruktivní:

($i$) konstanty 0 a 1 a každá logická proměnná jsou logickými formulemi,

($ii$) jsou-li $f$, $g$ logické formule, pak je logická formule i výraz $\bar{f}$, $f \wedge g$, $f \vee g$, $f \Rightarrow g$, $f \Leftrightarrow g$, $f \oplus g$. (Zde symbol $f \oplus g$ označuje tzv. "vylučovací nebo", které má hodnotu 1 právě když má hodnotu 1 právě jeden z argumentů $f$, $g$.)

**Definice 9.1.** *Má-li formule pro dané hodnoty proměnných hodnotu 1, říkáme, že je splněna. Formule, která je splněna pro všechny hodnoty proměnných, se nazývá* tautologie.

*Řekneme, že formule $f$ proměnných $x_1, x_2, \ldots, x_n$ je* splnitelná, *jestliže existují hodnoty $x_1, x_2, \ldots, x_n$, pro které je $f$ splněna.*

**Příklad.** Uvažujme logickou formuli $f(x, y, z) = (x \Rightarrow \bar{y}) \wedge (\bar{x} \Rightarrow z)$. Zjistíme, pro které hodnoty proměnných je formule $f$ splněna.

| $x$ | $y$ | $z$ | $\bar{x}$ | $\bar{y}$ | $x \Rightarrow \bar{y}$ | $\bar{x} \Rightarrow z$ | $f$ |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |

Naše formule je tedy splnitelná a je splněna pro vektory proměnných $[0,0,1], [0,1,1,], [1,0,0,]$ a $[1,0,1]$.

**Příklad.** Formule $f(x,y,z) = \overline{(x \Rightarrow (y \vee z))} \wedge (y \oplus z)$ nabývá hodnoty 0 pro všechny vektory $[x,y,z]$ (ověřte tabulkou) a tedy není splnitelná.

**Definice 9.2.** *Proměnné a jejich negace se nazývají* literály. *Literály a disjunkce dvou či více literálů se nazývají* (disjunktivní) klauzule.
*Je-li formule disjunktivní klauzulí nebo konjunkcí dvou či více disjunktivních klauzulí, říkáme, že formule je v* konjunktivní normální formě (tvaru).
*Je-li formule v konjunktivní normální formě a každá klauzule obsahuje literály všech proměnných, říkáme, že formule je v* úplné konjunktivní normální formě (tvaru).

**Poznámka.** Název "konjunktivní normální forma", resp. "úplná konjunktivní normální forma" budeme v následujícím zkracovat KNF, resp. ÚKNF.

**Příklad.** Formule $f(x,y,z) = (x \Rightarrow \bar{y}) \wedge (\bar{x} \Rightarrow z)$ není v konjunktivní normální formě (KNF).
Víme ale (viz příklad za definicí 9.1), kdy je tato formule splněna a kdy ne. Vezmeme ty hodnoty proměnných, kdy formule splněna není a složíme formuli $(x \vee y \vee z) \wedge (x \vee \bar{y} \vee z) \wedge (\bar{x} \vee \bar{y} \vee z) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$, která je v ÚKNF.
Poslední dva členy lze zapsat ve tvaru $\bar{x} \vee \bar{y}$, podobně i první dva členy sloučíme do tvaru $x \vee z$. Formule pak získá následující podobu: $f(x,y,z) = (x \vee z) \wedge (\bar{x} \vee \bar{y})$. Formule je v KNF, ale již ne v ÚKNF.

**Poznámka.** Lze dokázat, že pro danou formuli existuje vždy jen jediná ÚKNF (je tedy určena jednoznačně). Pro KNF toto obecně neplatí – téže formuli může odpovídat několik různých KNF. Např. formuli z předchozího příkladu lze zapsat rovněž ve tvaru $f(x,y,z) = (x \vee y \vee z) \wedge (x \vee \bar{y} \vee z) \wedge (\bar{x} \vee \bar{y})$, což je také KNF (nikoliv však úplná).

Analogicky lze definovat disjunktivní normální formu (DNF):

**Definice 9.3.** *Literály a konjunkce dvou či více literálů se nazývají* (konjunktivní) klauzule. *Je-li formule konjunktivní klauzulí nebo disjunkcí dvou či více konjunktivních klauzulí, říkáme, že formule je v* disjunktivním normálním tvaru (formě).

**Poznámka.** Analogicky můžeme též definovat úplnou DNF (ÚDNF), a analogické tvrzení platí pro jednoznačnost, resp. nejednoznačnost ÚDNF, resp. DNF (plyne z principu duality).

**Příklad.** ÚDNF formule uvedené v příkladu za definicí 9.1 je $f(x,y,z) = (\bar{x} \wedge \bar{y} \wedge z) \vee (\bar{x} \wedge y \wedge z) \vee (x \wedge \bar{y} \wedge \bar{z}) \vee (x \wedge \bar{y} \wedge z)$. Opět je možno slučovat dvojice klauzulí až do tvaru $f(x,y,z) = (\bar{x} \wedge z) \vee (x \wedge \bar{y})$, což je DNF, ale již ne úplná.

**Věta 9.1.** *Každou nekonstantní logickou formuli lze vyjádřit v ÚDNF i ÚKNF.*

**Důkaz.** Myšlenka důkazu je naznačena v předešlých příkladech, kde je vlastně popsána konstrukce těchto úplných forem. □

## 9.2 Problém splnitelnosti logických formulí – SAT

Následující problém, nazývaný "problém splnitelnosti logických formulí" (anglicky "satisfiability problem" – odtud zkratka "SAT") bude mít pro nás zásadní důležitost.

**SAT**
**Vstup:** logická formule $f(x_1, x_2, \ldots, x_n)$ v proměnných $x_1, x_2, \ldots, x_n$ v KNF (tj. $f = c_1 \wedge c_2 \wedge \ldots \wedge c_m$, kde $c_i$ jsou klauzule proměnných $x_1, x_2, \ldots, x_n$).
**Úkol:** zjistit, zda je formule $f$ splnitelná.

**Příklad.** Mějme logickou formuli $f = (x \vee y \vee \bar{z}) \wedge (\bar{x} \vee \bar{y}) \wedge (\bar{x} \vee z) \wedge (x \vee \bar{z}) \wedge (\bar{x} \vee \bar{y} \vee z) \wedge (y \vee z)$. Podaří-li se nám "uhádnout" hodnoty $x = 1$, $y = 0$, $z = 1$, pak snadno ověříme, že $f(1, 0, 1) = 1$, a víme, že je splnitelná.

Naproti tomu formule $g(x, y, z) = (x \vee y \vee z) \wedge (x \vee \bar{z}) \wedge (x \vee \bar{y} \vee z) \wedge (\bar{x} \vee y \vee z) \wedge (\bar{x} \vee \bar{y}) \wedge (\bar{x} \vee y \vee \bar{z})$ splnitelná není. Tento fakt ale neumíme algoritmicky efektivně zjistit, museli bychom prozkoumat $2^3$ možností (všechna možná přiřazení hodnot proměnným $x, y, z$).

I další úlohy, např. $k$-obarvitelnost, hamiltonovskost, existence kliky velikosti $k$, mají obdobný charakter. Pokud řešení existuje a "náhodou" se nám ho podaří uhodnout, pak ověření, že to, co jsme uhodli, opravdu splňuje podmínky úlohy, je možno provést v polynomiálním čase. Pokud však řešení úlohy neexistuje, pak tento fakt nejsme schopni algoritmicky efektivně (tj. v polynomiálním čase) ověřit. V odstavci 9.4 si ukážeme, že tato shoda není vůbec náhodná, protože, jak uvidíme v odst. 9.7, problém SAT je jedním ze základních NP-úplných problémů.

Nejprve si ale ukážeme, že omezíme-li se na formule, v nichž každá klauzule má délku nejvýše 2, pak je úloha splnitelnosti takových formulí řešitelná v polynomiálním čase.

## 9.3 Problém $k$-SAT a polynomialita problému 2-SAT

Úloha $k$-SAT je speciálním případem úlohy SAT, v němž se omezujeme na formule, u nichž každá klauzule má délku nejvýše $k$ (kde $k$ je předem dané přirozené číslo). V tomto odstavci si ukážeme, že úloha 2-SAT je řešitelná v polynomiálním čase. Poznamenejme, že v odstavci 9.8 poznáme, že již pro $k = 3$ je úloha 3-SAT NP-úplná.

**$k$-SAT**
**Vstup:** logická formule tvaru

$$f(x_1, x_2, \ldots, x_n) = \bigwedge_{i=1}^{m} \left( \bigvee_{j=1}^{k} a_{ij} \right),$$

kde každé $a_{ij}$ je rovno $x_\ell$ nebo $\bar{x}_\ell$ pro vhodné $\ell = 1, \ldots, n$ (tj. $f$ je formule v KNF, která má $m$ klauzulí délky $k$).
**Úkol:** zjistit, zda je formule $f$ splnitelná.

**Poznámka.** Úlohu $k$-SAT jsme zformulovali pro formule s klauzulemi délky právě $k$. Je však snadno vidět, že každou formuli s klauzulemi délky *nejvýše* $k$ lze ekvivalentně zapsat tak, že má klauzule délky *právě* $k$ tak, že v „kratších" klauzulích opakujeme některý literál.

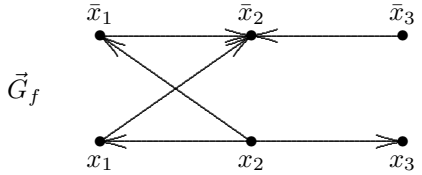
Speciálním případem úlohy $k$-SAT pro $k = 2$ je následující problém.

**2-SAT**
**Vstup:** logická formule $f(x_1, \ldots, x_n) = (a_1 \vee b_1) \wedge (a_2 \vee b_2) \wedge \ldots, \wedge (a_m \vee b_m)$, kde každé $a_i$, $b_i (i = 1, \ldots m)$ je rovno $x_\ell$ nebo $\bar{x}_\ell$ pro vhodné $\ell = 1, \ldots, n$ (tj. $f$ je formule v KNF s klauzulemi délky 2).
**Úkol:** zjistit, zda je formule $f$ splnitelná.

**Věta 9.2.** *2-SAT $\in$ P.*

**Důkaz.** Důkaz věty spočívá v tom, že ukážeme myšlenku konstrukce polynomiáního algoritmu pro úlohu 2-SAT.

Nechť $f = C_1 \wedge C_2 \wedge, \ldots, \wedge C_m$ je formule v proměnných $x_1, \ldots, x_n$ s klauzulemi o 2 literálech. Sestrojíme orientovaný graf $\vec{G}_f$ následující konstrukcí: $U(\vec{G}_f) = \{x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n\}$, a pro každou klauzuli $C_i = (a \vee b)$ budou v $\vec{G}_f$ obě hrany $(\bar{a}, b)$ a $(\bar{b}, a)$ (uvědomme si, že $\bar{\bar{b}} = b$). Například, pro formuli $f(x_1, x_2, x_3) = (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_2) \wedge (\bar{x}_2 \vee x_3)$ je příslušný graf $\vec{G}_f$ na následujícím obrázku.




Před vlastním důkazem věty 9.2 nejprve dokážeme několik pomocných tvrzení.

**Lemma 9.1.** *Existuje-li v $\vec{G}_f$ orientovaný sled z $a$ do $b$, pak v $\vec{G}_f$ existuje i orientovaný sled z $\bar{b}$ do $\bar{a}$.*

**Důkaz** lemmatu 9.1 plyne ihned z konstrukce grafu $\vec{G}_f$, neboť zřejmě $(a, b) \in H(\vec{G}_f)$ právě když $(\bar{b}, \bar{a}) \in H(\vec{G}_f)$. □

*Splňující přiřazení* je každý vektor $t \in \{0, 1\}^n$, pro který $f(t) = 1$ (tj. takové přiřazení hodnot proměnným $x_1, \ldots, x_n$, že pro tyto hodnoty je formule $f$ splněna).

Je-li $a$ literál a $t \in \{0, 1\}^n$ přiřazení hodnot proměnným $x_1, \ldots, x_n$, pak hodnotu literálu $a$ po dosazení vektoru hodnot $t$ budeme značit $a(t)$.

**Lemma 9.2.** *Existuje-li v $\vec{G}_f$ cesta z $a$ do $b$, pak pro každé splňující přiřazení $t$ je $a(t) = 1 \Rightarrow b(t) = 1$.*

**Důkaz.** Je-li $(a, b)$ hrana $\vec{G}_f$, pak v $f$ existuje klauzule $(\bar{a} \vee b)$. Protože $t$ je splňující, má (po dosazení vektoru $t$) každá klauzule hodnotu 1. Tedy i $(\bar{a}(t) \vee b(t)) = 1$. Je-li $a(t) = 1$, pak $\bar{a}(t) = 0$ a tedy nutně $b(t) = 1$. Zbytek důkazu plyne snadnou indukcí. □

**Lemma 9.3.** *Je-li $t$ splňující přiřazení, pak pro každou kvazikomponentu $\vec{G}_i$ grafu $\vec{G}_f$ a pro každé uzly $a, b \in U(\vec{G}_i)$ je $a(t) = b(t)$ (a tedy také $\bar{a}(t) = \bar{b}(t)$).*

**Důkaz** lemmatu 9.3 plyne ihned z lemmat 9.1 a 9.2. □

**Lemma 9.4.** *Formule $f$ je splnitelná právě když žádná kvazikomponenta grafu $\vec{G}_f$ neobsahuje současně některou proměnnou i její negaci.*

**Důkaz.** a) Nechť $t$ je splňující přiřazení. Je-li v některé kvazikomponentě grafu $\vec{G}_f$ současně $a = x_i$ i $b = \bar{x}_i$, pak podle lemmatu 9.3 musí být $x_i(t) = \bar{x}_i(t)$, což je spor. Podmínka věty je tedy nutná.

b) Ukážeme, že podmínka věty je postačující. Nechť tedy žádná kvazikomponenta grafu $\vec{G}_f$ neobsahuje současně žádnou proměnnou i její negaci; sestrojíme splňující přiřazení $t$.

Označme $\vec{G}_1, \ldots, \vec{G}_s$ kvazikomponenty grafu $\vec{G}_f$ v acyklickém očíslování (tj. $\vec{G}_1$ je vstupní, $\vec{G}_s$ je výstupní a hrany z $G_i$ do $G_j$ existují pouze pro $i < j$).

*Konstrukce přiřazení $t$.* Postupujeme pro $j = s, s-1, \ldots, 1$, pro každou proměnnou $x_i$ určíme největší index $j_0$ takový, že $x_i$ nebo $\bar{x}_i$ je v $\vec{G}_{j_0}$, a položíme

$x_i = 1$, jestliže $x_i \in U(\vec{G}_{j_0})$,

$x_i = 0$, jestliže $\bar{x}_i \in U(\vec{G}_{j_0})$.

Protože v $\vec{G}_{j_0}$ nemůže být současně $x_i$ i $\bar{x}_i$, je toto přiřazení jednoznačné.

Zbývá dokázat, že $t$ je splňující, tj. že každá klauzule má hodnotu 1. Předpokládejme naopak, že nějaká klauzule $C = (a \vee b)$ má hodnotu 0. Pak $a$ nemá hodnotu 1, protože v okamžiku přiřazení už $\bar{a}$ měl hodnotu 1 v nějaké komponentě s vyšším indexem. Označíme:

$\vec{G}_{i_1}$ kvazikomponentu obsahující $a$,

$\vec{G}_{i_2}$ kvazikomponentu obsahující $b$,

$\vec{G}_{i_3}$ kvazikomponentu obsahující $\bar{a}$,

$\vec{G}_{i_4}$ kvazikomponentu obsahující $\bar{b}$.

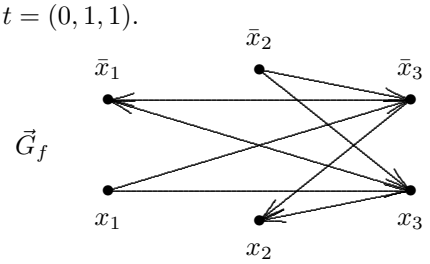
Pak $a$ nemá hodnotu 1, protože v okamžiku přiřazení už $\bar{a}$ měl hodnotu 1 v $\vec{G}_{i_3}$, a tedy $i_1 < i_3$. Obdobně $b$ nemá hodnotu 1, protože v okamžiku přiřazení už $\bar{b}$ měl hodnotu 1 v $\vec{G}_{i_4}$, odkud $i_2 < i_4$. Ale protože $(\bar{a}, b) \in H(\vec{G}_f)$, je $i_3 \leq i_2$, a obdobně z $(\bar{b}, a) \in H(\vec{G}_f)$ plyne $i_4 \leq i_1$. Celkem tedy $i_1 < i_3 \leq i_2 < i_4 \leq i_1$, což je spor. Každá klauzule má tedy hodnotu 1, tj. $f$ je splněna. □

Nyní můžeme dokončit důkaz věty 9.2. Graf $G_f$ má $|U(G_f)| = 2n$ uzlů a $|H(G_f)| = 2m$ hran, tj. jeho velikost je polynomiální funkcí rozměru formule $f$. Protože kvazikomponenty, kondenzaci i acyklické číslování lze nalézt v polynomiálním čase, dává lemma 9.4 polynomiální algoritmus pro rozhodnutí, zda $f$ je splnitelná. □

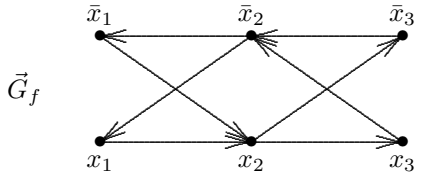
**Příklad.** Nechť $f(x_1, x_2, x_3) = (\bar{x}_1 \vee \bar{x}_3) \wedge (x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_3) \wedge (x_2 \vee x_3)$. Graf $G_f$ (viz obrázek) je acyklický, a pro acyklické očíslování jeho uzlů dané tabulkou:

| uzel | $x_1$ | $x_2$ | $x_3$ | $\bar{x}_1$ | $\bar{x}_2$ | $\bar{x}_3$ |
|---|---|---|---|---|---|---|
| acyklické očíslování | 1 | 5 | 4 | 6 | 2 | 3 |

dostaneme splňující přiřazení $t = (0, 1, 1)$.




**Příklad.** Nechť $f(x_1, x_2, x_3) = (x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_2) \wedge (\bar{x}_2 \vee x_3) \wedge (\bar{x}_2 \vee \bar{x}_3)$. Graf $G_f$ (viz obrázek) je silně souvislý a tedy $f$ není splnitelná.




V následujícím odstavci uvidíme, že bez omezení délky klauzulí je problém SAT ekvivalentní s úlohou nalézt v daném neorientovaném grafu nezávislou množinu předepsané velikosti .

## 9.4 Problém existence nezávislé množiny uzlů dané velikosti – IND

Problémem IND (angl "independent set" - nezávislá množina) rozumíme následující úlohu.

**IND**
**Vstup:** neorientovaný graf $G$ na $n$ uzlech a přirozené číslo $k \leq n$.
**Úkol:** zjistit, zda v grafu $G$ existuje nezávislá množina uzlů velikosti alespoň $k$.

**Poznámka.** Úloha IND je často zaměňována s úlohou zjištění čísla nezávislosti $\alpha(G)$ daného grafu $G$. Určení hodnoty $\alpha(G)$ (stejně jako kteréhokoliv jiného parametru grafu) je však optimalizačním, a nikoliv rozhodovacím problémem, a tedy nemá smysl hovořit o jeho příslušnosti či nepříslušnosti do třídy P. Úlohu lze však snadno převést na $n$-násobné provedení úlohy IND pro různé hodnoty konstanty $k$. Odtud ihned plyne, že $\alpha(G)$ lze určit v polynomiálním čase právě když příslušná rozhodovací úloha IND patří do třídy P, a tedy z hlediska polynomiality či nepolynomiality jsou obě úlohy ekvivalentní.

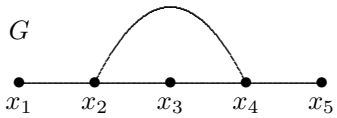
**Příklad.** Je dán graf $G$ s uzly $x_1, x_2, \ldots, x_n$ a číslo $k$. Sestrojíme formuli $f(u_1, u_2, \ldots, u_n)$ takovou, že $f$ je splněna právě když množina $N = \{x_i \in U(G) | u_i = \text{TRUE}\}$ je nezávislá množina o alespoň $k$ prvcích.

Položíme $f(u_1, \ldots, u_n) = f_1(u_1, \ldots, u_n) \wedge f_2(u_1, \ldots, u_n)$, kde

(*i*) $f_1(u_1, \ldots, u_n) = \bigwedge_{(x_i, x_j) \in H(G)} (\bar{u}_i \vee \bar{u}_j)$,

(*ii*) $f_2(u_1, \ldots, u_n)$ je formule, která má hodnotu TRUE právě tehdy, když alespoň $k$ z proměnných $u_1, \ldots, u_n$ má hodnotu TRUE (na toto existují standardní konstrukce, které zde nebudeme uvádět).

Například, pro graf $G$ uvedený na obrázku je formule $f_1$ tvaru $f_1 = (\bar{u}_1 \vee \bar{u}_2) \wedge (\bar{u}_2 \vee \bar{u}_3) \wedge (\bar{u}_3 \vee \bar{u}_4) \wedge (\bar{u}_4 \vee \bar{u}_5) \wedge (\bar{u}_2 \vee \bar{u}_4)$.




Je zřejmé, že $f_1$ je splněna právě tehdy, když množina $N = \{x_i | u_i = 1\}$ je nezávislá.

**Poznámka.** Všimněme si, že délka formule $f_1$ je polynomiální funkcí $n$, neboť počet klauzulí je roven počtu hran grafu (který je nejvýše $O(n^2)$). Pro formuli $f_2$ platí totéž. To znamená, že jsme našli *polynomiální redukci IND na SAT.*

Z tohoto faktu vyplývají následující důsledky. Zformulujeme si je již nyní, i když poněkud nepřesně – pojmy v nich použité dostanou přesný obsah v odstavci 9.6.

**Důsledek 9.1.** *Úloha SAT je alespoň tak těžká jako úloha IND.*

**Důsledek 9.2.** *Kdybychom měli polynomiální algoritmus na SAT, bylo by možné vytvořit polynomiální algoritmus i na IND.*

**Příklad.** Nechť je nyní naopak dána logická formule v KNF tvaru

$$f(u_1, u_2, \ldots, u_n) = \bigwedge_{i=1}^{m} \left( \bigvee_{j=1}^{m_i} a_{ij} \right),$$

kde každé $a_{ij}$ je tvaru $u_k$ nebo $\bar{u}_k$ pro vhodné $k$.

Sestrojíme graf $G$ následující konstrukcí:

$U(G) = \{x_{ij} |\ i = 1, \ldots, m\,;\ j = 1, \ldots, m_i\}$,

$H(G) = \{\{x_{ij}, x_{pq}\} |\ i = p \text{ a } j \neq q\} \cup \{\{x_{ij}, x_{pq}\} |\ i \neq p,\ j = q \text{ a } a_{ij} = \bar{a}_{pq}\}$.

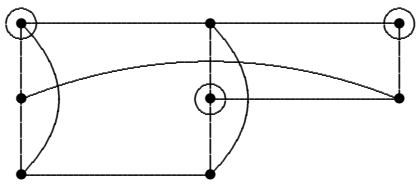
Ukážeme, že v $G$ existuje nezávislá množina velikosti $m$ právě když $f$ je splnitelná.

a) Je-li $f(u_1, \ldots, u_k) = 1$, pak v každé klauzuli existuje alespoň jeden literál, který je roven jedné, a příslušné uzly v $G$ jsou nezávislé.

b) Naopak, je-li $M$ nezávislá množina velikosti $m$ v $G$, pak každý uzel je z jiné kliky (tj. z jiné klauzule). V příslušné klauzuli polož příslušný literál = 1 (z nezávislosti $M$ plyne, že nemůže nikde dojít ke sporu definováním stejné proměnné dvěma způsoby). Případné zbylé proměnné dodefinujeme libovolně. Pak každá klauzule = 1 a tedy i $f = 1$.

Nalezli jsme tedy *polynomiální redukci SAT na IND.*

Konstrukci grafu $G$ ilustrujme na formuli $f = (u_1 \vee \bar{u}_2 \vee u_3) \wedge (\bar{u}_1 \vee \bar{u}_2 \vee \bar{u}_3) \wedge (u_1 \vee u_2)$. Graf $G$, sestrojený pomocí výše uvedené konstrukce, a pravdivostní tabulka formule $f$ jsou na obrázku.



| $u_1$ | $u_2$ | $u_3$ | $c_1$ | $c_2$ | $c_3$ | f |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 |

Jak snadno prověříme, nezávislé množiny velikosti 3 v $G$ odpovídají vektorům booleovských proměnných, pro něž je $f$ splněna (jedna z nezávislých množin v $G$ je v obrázku tvořena zakroužkovanými uzly).

Z nalezené polynomiální redukce SAT na IND ihned vyplývají následující dvě skutečnosti.

**Důsledek 9.3.** *Úloha IND je alespoň tak těžká jako úloha SAT.*

**Důsledek 9.4.** *Kdybychom měli polynomiální algoritmus na IND, bylo by možné vytvořit polynomiální algoritmus i na SAT.*

Porovnáním důsledků 9.2 a 9.4 pak dostáváme následující překvapivé zjištění.

**Důsledek 9.5.** *SAT* $\in P$ *právě když* *IND* $\in P$ .

Nyní již asi tušíme některé hlubší souvislosti, ale k jejich formulaci potřebujeme precizovat základní pojmy:

- co to je polynomiální redukce dvou problémů,
- jak to je s tím "lehkým uhádnutím" či "těžkým zamítnutím".

## 9.5 Třída NP

Začněme nejprve tou druhou otázkou.

**Definice 9.4.** Nedeterministický algoritmus *se v některých krocích může libovolně rozhodnout pro některé z několika možných různých pokračování.*

**Příklad.** Uvažujme problém IND a následující algoritmus.

Vstup: graf $G$ s uzly $u_1, \ldots, u_n$ a přirozené číslo $k \leq n$.

1) $X := 0$ (*inicializace*)
2) pro $i = 1, 2, \ldots, n$ proveď:
   a) buďto $X := X \cup \{u_i\}$,
   b) nebo $X := X$

   (*nedeterministický krok*)
3) Jestliže $\{u_i, u_j\} \in H(G)$ pro některé $u_i, u_j \in X$, pak REJECT (*test nezávislosti*)
4) Je-li $|X| \geq k$ pak ACCEPT, jinak REJECT (*test* $|X| \geq k$)

V důsledku nedeterminističnosti 2. kroku je $2^n$ možných různých výpočtů. Podstatné ale je to, že v $G$ existuje nezávislá množina o alespoň $k$ uzlech právě když existuje výpočet, končící ACCEPT. Proto definujeme:

**Definice 9.5.** Nedeterministický přijímací počítač *má (navíc oproti dříve definovanému přijímacímu počítači) příkaz* CHOOSE L1, L2, *kde L1, L2 jsou návěští.*

Smysl příkazu CHOOSE L1, L2 lze ekvivalentně zapsat takto:

vyber si JUMP L1 nebo JUMP L2.

**Definice 9.6.** *Řekneme, že slovo $w$* je přijímáno *nedeterministickým přijímacím počítačem $M$, jestliže* existuje *přípustný výpočet počítače $M$ s počáteční konfigurací danou slovem $w$ a končící ACCEPT. V opačném případě řekneme, že nedeterministický přijímací počítač* odmítá *slovo $w$.*
*Řekneme, že $M$* přijímá jazyk *$J$, jestliže pro každé slovo $w$ je $w \in J \Leftrightarrow M$ přijímá $w$.*

Jak je to s délkou výpočtu?

- když $M$ přijímá $w$: při prvním ACCEPT to víme a nemusíme dál počítat,
- když $M$ odmítá $w$: musíme pokračovat, dokud neukončíme všechny přípustné výpočty.

Proto definujeme:

**Definice 9.7.** *Řekneme, že nedeterministický přijímací počítač $M$* zpracuje slovo $w$ v čase nejvýše $t$, *jestliže:*

- *buďto $M$ přijímá $w$,* a existuje *přípustný výpočet určený slovem $w$ a končící ACCEPT po nejvýše $t$ krocích,*
- *nebo $M$ odmítá $w$* a každý *přípustný výpočet končí po nejvýše $t$ krocích.*

**Definice 9.8.** *Řekneme, že nedeterministický přijímací počítač $M$* pracuje v polynomiálně omezeném čase, *jestliže existuje polynom $p$ takový, že libovolné slovo délky $n$ zpracuje v čase nejvýše $p(n)$.*

**Příklad.** Mějme nedeterministický přijímací počítač $M$ z příkladu před definicí 9.5. Každý přípustný výpočet proběhne v polynomiálním čase, ale možných přípustných výpočtů je $2^n$.

**Příklad.** $M$ definuji takto:
- vstup: graf $G$ na $n$ uzlech
- náhodně vygeneruj faktor grafu $G$
- test souvislosti: ne $\rightarrow$ REJECT
- test kružnice (pravidelný graf stupně 2):
  ne $\rightarrow$ REJECT
  ano $\rightarrow$ ACCEPT

Každý výpočet je polynomiální; přijímaná slova jsou hamiltonovské grafy (přijímaný jazyk).

**Definice 9.9.** *Třída NP je třída všech jazyků $J$ takových, že existuje nedeterministický přijímací počítač, který pracuje v polynomiálně omezeném čase a přijímá jazyk $J$.*

**Poznámka.** Jak už jsme poznamenali na konci odstavce 8.3, pojem přijímacího počítače je abstraktním modelem algoritmu, řešícího rozhodovací problém, a pojem jazyka je modelem tohoto řešeného problému. Z definice 9.9 tedy vyplývá, že všechny problémy z třídy NP (tj. problémy, jejichž modelem je jazyk patřící do třídy NP) musí být problémy rozhodovací. V případě optimalizačního problému je nutno uvažovaný problém nejprve převést na problém rozhodovací (tak jak jsme to například učinili při definici problému IND v odstavci 9.4).

**Příklad.** Uvažujme následující úlohu.

**HAM**
**Vstup:** neorientovaný graf $G$ na $n$ uzlech.
**Úkol:** zjistit, zda v grafu $G$ existuje hamiltonovská kružnice.

Pak algoritmus z příkladu před definicí 9.9 dokazuje, že HAM $\in$ NP. Poznamenejme, že podobně algoritmus z příkladu před definicí 9.5 dokazuje že IND $\in$ NP.

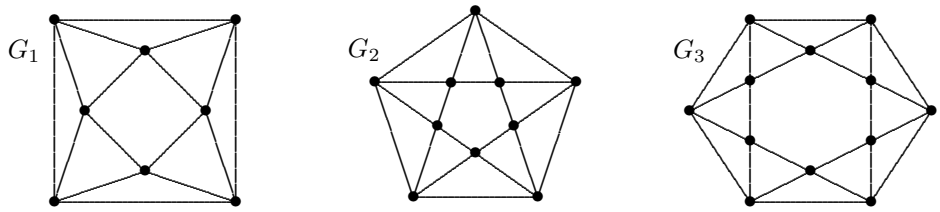
**Věta 9.3.** $P \subset NP$

**Důkaz.** Deterministický přijímací počítač je speciální případ nedeterministického. □

**Poznámka.** Jak jsme již poznamenali v kapitole 1, pro problémy třídy NP je typické, že podaří-li se nám náhodou „uhádnout“ řešení, pak ověření jeho správnosti je možno provést v polynomiálním čase. Na úrovni abstraktního počítače jakožto modelu algoritmu je toto „uhádnutí“ modelováno pojmem nedeterminismu.

Pro problémy (jazyky) z třídy P tedy umíme řešení v polynomiálním čase *nalézt*, a pokud již řešení máme, umíme též v polynomiálním čase *verifikovat* jeho správnost. U problémů třídy NP již rezignujeme na požadavek polynomiálního nalezení řešení (to je na úrovni abstraktního počítače modelováno nedeterminismem jakožto modelem „uhádnutí“), stále však zůstává v platnosti druhý požadavek – když už jsme řešeni „uhádli“ (neterministicky vygenerovali), umíme v polynomiálním čase *ověřit* (*verifikovat*) jeho správnost. Proto se o problémech třídy NP také často hovoří jako o *polynomiálně verifikovatelných* problémech.

**Příklad.** Uvažujme posloupnost grafů $\{G_i\}_{i=1}^{\infty}$, jejíž první tři členy vidíme na následujícím obrázku.




Tyto grafy mají zajímavou vlastnost: pro každý jejich uzel $x$ platí, že množina $N(x)$ všech sousedů uzlu $x$ indukuje vždy tentýž graf, izomorfní s cestou délky 3. Navíc, vidíme, že graf s touto vlastností může být libovolně veliký. Pokusme se tuto otázku otočit, a uvažujme následující rozhodovací úlohu.

**Vstup:** neorientovaný graf $H$ na $n$ uzlech.
**Úkol:** zjistit, zda existuje graf $G$, v němž okolí každého uzlu indukuje podgraf izomorfní s grafem $H$.

Tento problém, v literatuře známý jako Trachtěnbrot-Zykovův problém, je příkladem rozhodovacího problému, který nepatří do třídy NP. Potíž je v tom, že i při malém grafu $H$ (tj. při malé velikosti vstupních dat) může být hledaný graf $G$ veliký, a nemáme předem žádný odhad jeho velikosti. I kdyby se nám tedy podařilo graf $G$ „náhodně uhádnout“, nemůžeme zaručit, že ověření jeho vlastností bude proveditelné v čase, který je polynomiální funkcí velikosti vstupních dat.

Tento příklad znovu ukazuje, že zjištění, že nějaký problém je ve třídě NP, je zjištění pozitivní: nevyplývá z něj sice ještě řešitelnost v polynomiálním čase, ale je zde jistá šance (jen kdyby se podařilo odstranit ten nedeterminismus ...), a jsou problémy, které jsou ještě podstatně horší.

**Poznámka.** Máme-li za úkol „prakticky“ vyřešit nějaký problém, popsaný jazykem $J$, o němž se domníváme, že nejspíše nebude ve třídě P, setkáváme se s několika různými otázkami, na něž odpovědí bude vhodný algoritmus:

- $(i)$ ptáme se, zda je $J \in \mathrm{P}$,
- $(ii)$ ptáme se, zda je alespoň $J \in \mathrm{NP}$ (nebo je ještě hůře ...),
- $(iii)$ hledáme přesné řešení postupy typu „hrubá síla“ (probírkou všech možností),
- $(iv)$ rezignujeme na přesné řešení (protože postupy typu ad $(iii)$ jsou časově příliš náročné), použijeme heuristiku a spokojíme se s přibližným řešením.

Pro odpověď na každou z těchto otázek potřebujeme jiný typ algoritmu. Základní vlastnosti těchto algoritmů shrnuje následující tabulka.

| | Algoritmus | Deterministický | Polynomiální | Přijímá $J$ |
|---|---|---|---|---|
| $(i)$ | Dokazuje $J \in \mathrm{P}$ | ANO | ANO | ANO |
| $(ii)$ | Dokazuje $J \in \mathrm{NP}$ | NE | ANO | ANO |
| $(iii)$ | „Hrubá síla“ | ANO | NE | ANO |
| $(iv)$ | Heuristika | ANO | ANO | NE |

Vidíme, že nalezení algoritmu, který má všechny tři „pěkné“ vlastnosti (deterministický, polynomiální, přijímá $J$) je možné [9] pouze ve třídě P, a mimo P musíme na jednu z těchto vlastností rezignovat.

[9] Samozřejmě při současné úrovni poznání; kdyby bylo P = NP, tak by situace vypadala jinak.

## 9.6 Polynomiální redukce a NP-úplné problémy

Nejprve precizujeme pojem "vzájemného převodu" úloh.

**Definice 9.10.** Překládací počítač *je počítač $M$ s libovolným přístupem takový, že pro každý jeho výpočet (daný libovolnou počáteční konfigurací) platí:*

- *každé provedení WRITE zapíše na výstupní pásku číslo* 0 *nebo* 1,
- *po konečném počtu kroků se výpočet zastaví provedením STOP.*

*Posloupnost nul a jedniček, zapsanou na výstupní pásce při výpočtu určeném slovem $w$, označíme $M(w)$.*

Jinak řečeno - překládací počítač poskytuje funkci $z$ množiny všech slov *do* sebe.

**Definice 9.11.** *Řekneme, že jazyk $J_1$ je* redukovatelný *na jazyk $J_2$ (značíme $J_1 \triangleleft J_2$), jestliže existuje deterministický překládací počítač $M$ pracující v polynomiálně omezeném čase takový, že pro každé slovo $w$ je $w \in J_1 \Leftrightarrow M(w) \in J_2$.*

Nejprve několik snadných tvrzení.

**Věta 9.4.**

*1)* $J_1 \triangleleft J_2,\ J_2 \triangleleft J_3 \Rightarrow J_1 \triangleleft J_3$ *(tranzitivita)*
*2)* $J_1 \triangleleft J_2,\ J_2 \in NP \Rightarrow J_1 \in NP$
*3)* $J_1 \triangleleft J_2,\ J_2 \in P \Rightarrow J_1 \in P$

**Důkaz.**

1) Jestliže $M_1$ realizuje $J_1 \triangleleft J_2$ a $M_2$ realizuje $J_2 \triangleleft J_3$, pak $M$ realizující $J_1 \triangleleft J_3$ získáme jako složení $M_1$ a $M_2$:
   - program $M_2$ zapíšeme za program $M_1$,
   - příkaz STOP v programu $M_1$ všude nahradíme skokem na první instrukci $M_2$,
   - upravíme návěští v programu $M_2$, vstupy a výstupy, tak aby odpovídaly adresám a číslům příkazů.
2) , 3) - obdobným způsobem.

□

**Poznámka.** $J_1 \triangleleft J_2$ znamená, že $J_2$ je alespoň tak těžký jako $J_1$ (kdybychom měli polynomiální algoritmus na $J_2$, měli bychom jej i na $J_1$ – získali bychom jej složením s algoritmem, jenž je popsán počítačem, realizujícím převod $J_1 \triangleleft J_2$).

Příklady z odstavce 9.4 dávají v právě zavedené terminologii polynomiální redukce IND $\triangleleft$ SAT a SAT $\triangleleft$ IND.

Nyní jsme připraveni na zavedení pojmu NP-úplnosti, hlavního tématu tohoto oddílu.

**Definice 9.12.** *Jazyk $J$ se nazývá* NP-úplný, *jestliže platí*

*1)* $J \in NP$
*2) pro každý* $J' \in NP$ *je* $J' \triangleleft J$.

*Třídu NP-úplných jazyků označíme NPC (NP-complete).*

Jinými slovy: problém je NP-úplný, jestliže náleží do třídy NP a kterýkoliv jiný problém z třídy NP lze na něj polynomiálně převést. Tedy – NP-úplné problémy jsou "nejtěžší" problémy ze třídy NP.

**Věta 9.5.** *Platí: buď $P = NP$, nebo $P \cap NPC = \emptyset$.*

**Důkaz.** Kdyby $K \in P \cap NPC$, tak by pro libovolný $J \in NP$ bylo $J \lhd K$ (neboť $K \in NPC$), a tedy $J \in P$ (neboť $K \in P$). □

**Poznámka.**

1. Je-li $J_1, J_2 \in NPC$, pak $J_1 \lhd J_2$ i $J_2 \lhd J_1$.
2. NPC je třída "nejtěžších" úloh v NP. Nyní vidíme, že tyto úlohy jsou co do obtížnosti ekvivalentní.
3. Jsou tedy dvě možnosti:
   (*i*) $P \subset NP$, $NPC \subset NP$, $P \cap NPC = \emptyset$,
   (*ii*) $P = NPC = NP$.
   Neví se však, která z uvedených možností platí.
4. Zdůrazněme, že podle definice 9.12 každý NP-úplný problém patří do třídy NP, a tedy je rozhodovacím problémem. Optimalizační problémy („určete $\alpha(G)$, $\omega(G)$, $\chi(G)$" apod.) nemohou být NP-úplné, protože nepatří do NP, a je nutno je nejprve na rozhodovací problémy převést.
5. O jazycích $J$, které splňují podmínku 2) definice 9.12 (tj. pro každý $J' \in NP$ je $J' \lhd J$), ale nejsou nutně ze třídy NP, se někdy říká že jsou *NP-těžké*. Speciálně tedy jsou NP-těžkými problémy optimalizační verze NP-úplných problémů které poznáme dále (určení $\alpha(G)$, $\omega(G)$, $\chi(G)$ atd.)

## 9.7 NP-úplnost problému SAT – Cookova věta

Až do této chvíle nevíme, zda vůbec nějaký NP-úplný problém existuje. Jak uvidíme, Cookova věta identifikuje problém SAT jako prvního "obyvatele" třídy NP. Otázkou však je, jak se dá NP-úplnost nějakého problému vůbec dokázat – jak dokázat, že *všechny* problémy z třídy NP se dají na daný problém převést? Zde se ukáže síla aparátu abstraktního počítače, který jsme zavedli: protože každý problém ze třídy NP je popsatelný abstraktním počítačem (nedeterministickým polynomiálním přijímacím), stačí ukázat, že se na SAT dá převést obecný popis každého takového počítače. Tato idea je zároveň myšlenkou důkazu Cookovy věty.

**Věta 9.6. (Cook)** SAT ∈ NPC.

**Poznámka.** Uvážíme-li definici NP-úplnosti, pak – v řeči poznámky na str. 63 – Cookova věta říká, že každý polynomiálně verifikovatelný problém lze redukovat na problém splnitelnosti logických formulí. V této formulaci lépe vynikne síla Cookovy věty: ověřit, že daný problém je polynomiálně verifikovatelný (tj. patří do třídy NP) je často velmi snadné, a tento často téměř zřejmý fakt má hluboce netriviální důsledek – redukovatelnost na SAT.

**Důkaz.** 1. SAT ∈ NP. To je celkem zřejmé. Nedeterministicky přiřadíme proměnným hodnoty 0, 1 a určíme hodnotu formule $f$ pro tyto hodnoty proměnných (což lze provést v čase úměrném délce formule). Tedy SAT ∈ NP.

2. "Zbývá" dokázat, že pro každý $J \in$ NP je $J \lhd$ SAT. Nechť tedy:
   - $J \in$ NP,
   - $M$ je nedeterministický přijímací počítač, přijímající $J$ v čase omezeném polynomem $p$.

Úkol: pro každé slovo $w \in J$ nalézt formuli $f_w$ v KNF tak, že $f_w$ je splnitelná, právě když $M$ přijímá $w$. Označíme:
- $n$ - délka slova $w$,
- $m = p(n)$,
- $q$ - počet příkazů programu počítače $M$.

Lze předpokládat, že

- $p$ je současně polynom z Omezení 1 z odst. 8.2 (jinak vezmeme maximum),
- výpočet pracuje v paměti nejvýše $m$ (což podle věty 8.1 lze).

To znamená, že

- pouze prvních $n$ polí vstupní pásky obsahuje čísla $\neq 0$,
- pouze prvních $m$ paměťových buněk obsahje číslo $\neq 0$,
- v žádné paměťové buňce není číslo větší než $m$,
- počet kroků počítače je nejvýše $m$.

K popisu všech konfigurací stačí znát hodnoty logických proměnných:

| | | | |
|---|---|---|---|
| $a_i$, | $i = 1, \ldots, n$ : | $a_i = 1 \Leftrightarrow$ | v $i$-tém poli vstupní pásky je číslo 1, |
| $b_{it}$, | $i = 1, \ldots, n$ :<br>$t = 0 \ldots, m$ | $b_{it} = 1 \Leftrightarrow$ | po $t$-tém kroku je vstupní hlava na $i$-tém poli pásky, |
| $c_{ijt}$, | $i = 0 \ldots m$ :<br>$j = 0, \pm 1, \ldots, \pm m$<br>$t = 0, \ldots m$ | $c_{ijt} = 1 \Leftrightarrow$ | v $t$-té konfiguraci je v $i$-té paměťové buňce číslo $j$, |
| $d_{jt}$, | $j = 1, \ldots, q$ :<br>$t = 0, \ldots, m$ | $d_{jt} = 1 \Leftrightarrow$ | v $t$-té konfiguraci je v programovém registru číslo $j$. |

Počet proměnných je

$$n + n(m+1) + (m+1)^2(2m+1) + (m+1)q = O(m^3) = O\left(p^3(n)\right),$$

tj. je polynomiálně omezen.

Mají-li proměnné popisovat kroky počítače $M$, který v $m$ krocích přijme slovo $w$, musí splňovat následující podmínky:

1) hodnoty $a_i$ se shodují s 0 a 1 ve slově $w$,
2) pro každé $t$ existuje nejvýše jedno $i$ tak, že $b_{it} = 1$ (mělo by vlastně být "právě jedno $i$", ale to, že hlava musí někde být, vyplyne z popisu příkazů a počáteční konfigurace),
3) pro každé $i, t$ existuje právě jedno $j$ tak, že $c_{ijt} = 1$ (tj. v každé buňce je právě jedno číslo),
4) pro každé $t$ existuje právě jedno $j$ tak, že $d_{jt} = 1$ (stejný důvod pro programový registr),
5) jestliže $d_{jm} = 1$, pak $j$ je návěští některého příkazu ACCEPT,
6) vztah hodnot proměnných mezi $t-1$ a $t$ musí odpovídat prováděnému příkazu počítače.

Položíme $f_w = f_{w1} \wedge f_{w2} \wedge f_{w3} \wedge f_{w4} \wedge f_{w5} \wedge f_{w6}$, kde $f_{wi}$ je formule daná podmínkou $i$, $i = 1, \ldots, 6$. Ukážeme, jak formule $f_{wi}$ sestrojit.

1) Polož $A_i = \{ \begin{smallmatrix} a_i \\ \bar{a}_i \end{smallmatrix}$ , je-li $i$-tý prvek $w$ roven $\{ \begin{smallmatrix} 1 \\ 0 \end{smallmatrix}$ , a $f_{w1} = A_1 \wedge A_2 \wedge \ldots \wedge A_n$.
2) Nejprve definujeme pomocnou formuli $g(x_1, \ldots, x_r)$ vztahem $g(x_1, x_2, \ldots, x_r) = \bigwedge_{1 \le i < j \le r} (\bar{x}_i \vee \bar{x}_j)$. Zřejmě $g(x_1, x_2, \ldots, x_r)$ je splněna právě když *nejvýše jedno* $x_i = 1$. Pomocí této formule nyní sestavíme formuli $f_{w2}$:
$$f_{w2} = \bigwedge_{t=0}^{m} g(b_{1t}, \ldots b_{nt}).$$
3) Polož $h(x_1, \ldots x_r) = g(x_1, \ldots x_r) \wedge (x_1 \vee x_2 \vee \ldots \vee x_r)$. Protože $x_1 \vee x_2 \vee \ldots \vee x_r = 1$ právě když *alespoň jedno* $x_i = 1$, je $h(x_1, \ldots x_r) = 1 \Leftrightarrow$ *právě jedno* $x_i = 1$. Pomocí těchto formulí lze napsat:
$$f_{w3} = \bigwedge_{i=0}^{m} \bigwedge_{t=0}^{m} h(c_{i,-m,t}, \ldots, c_{i,0,t}, \ldots, c_{i,m,t}).$$

4) Obdobně získáme formuli pro $f_{w4}$:

$$f_{w4} = \bigwedge_{t=0}^{m} h(d_{1,t}, \dots d_{q,t}).$$

5) $f_{w5} = \bigvee_{j \in A} d_{jm}$, kde $A$ je množina návěští všech výskytů příkazu ACCEPT.

6) Toto je nejsložitější. Formálně položme $f_{w6} = \bigwedge_{r=1}^{q} f_{w6r}$, kde formule $f_{w6r}$ popisuje působení $r$-tého příkazu programu. Ukážeme si tuto část důkazu pouze na příkladu jednoho konkrétního příkazu. Nechť $r$-tý příkaz programu je ADD 8, což znamená, že do pracovního registru přičteme obsah buňky s adresou 8. Pro všechna $t = 1, \dots, m$ tedy musí platit:

   je-li $d_{r,t-1} = 1$ (v konfiguraci $t-1$ je v programovém registru návěští $r$ příkazu ADD 8),
   pak z $c_{0,x,t-1} \wedge c_{8,z,t-1} = 1$ (v pracovním registru je $x$, na adrese 8 je $z$)
   plyne $c_{0,x+z,t} = 1$ (po $t$-tém kroku je v pracovním registru číslo $x + z$).

   Dostaneme tak formuli:

$$\bigwedge_{t=1}^{m} \bigwedge_{x=-m}^{m} \bigwedge_{z=-m}^{m} (d_{r,t-1} \Rightarrow ((c_{0,x,t-1} \wedge c_{8,z,t-1}) \Rightarrow c_{0,x+z,t}))$$

   a po úpravě s užitím vztahu $(A \Rightarrow B) \Leftrightarrow (\bar{A} \vee B)$ a de Morganova zákona:

$$f_1 = \bigwedge_{t=1}^{m} \bigwedge_{x=-m}^{m} \bigwedge_{z=-m}^{m} \left(\bar{d}_{r,t-1} \vee \bar{c}_{0,x,t-1} \vee \bar{c}_{8,z,t-1} \vee c_{0,x+z,t}\right).$$

   Tím je popsán účinek příkazu ADD 8; ještě je třeba popsat, že jestliže je $d_{r,t-1} = 1$, pak se kromě obsahu pracovního a programového registru nic jiného nezmění. K tomu použijeme pomocnou formuli:

$$k(x, y) = (x \vee \bar{y}) \wedge (\bar{x} \vee y).$$

   Je zřejmé, že $k(x, y) = 1 \Leftrightarrow x = y$. Pomocí této formule zapíšeme požadavky splnění následujících formulí:

   $f_2 = \bigwedge_{i=1}^{m} \bigwedge_{j=-m}^{m} \bigwedge_{t=1}^{m} (d_{r,t-1} \Rightarrow k(c_{i,j,t-1}, c_{i,j,t}))$ (kromě pracovního registru se obsah paměti nemění),

   $f_3 = \bigwedge_{i=1}^{n} \bigwedge_{t=1}^{m} (d_{r,t-1} \Rightarrow k(b_{i,t-1}, b_{i,t}))$ (se vstupní hlavou se nic nestalo),

   $f_4 = \bigwedge_{j=1}^{q-1} \bigwedge_{t=1}^{m} (d_{r,t-1} \Rightarrow k(d_{j,t-1}, d_{j+1,t}))$ (v programovém registru se číslo zvětší o 1).

   Formule $f_2$, $f_3$ a $f_4$ zřejmým způsobem upravíme do KNF (obdobně jako jsme upravili formuli $f_1$), a formuli $f_{w6r}$ pak zapíšeme jako konjunkci $f_1 \wedge f_2 \wedge f_3 \wedge f_4$.

Pro dokončení důkazu bychom museli podobně popsat všechny další příkazy.

Dále je třeba prověřit polynomialitu našeho převodu. Jak jsme již uvedli, formule $f_w$ má $O(p^3(n))$ proměnných, a z konstrukce vidíme, že každá z námi sestrojených formulí $f_{w1}, \dots, f_{w5}$ má nejvýše $O(m^2) = O(p^2(n))$ klauzulí, a formule $f_{w6r}$ má $O(m^3) = O(p^3(n))$ klauzulí. Po dokončení popisu zbývajících příkazů programu bychom se přesvědčili, že i počet klauzulí všech ostatních formulí $f_{w6j}$, $j = 1, \dots, q$, a tedy i formule $f_{w6}$, je omezen polynomem v proměnné $n$. □

**Poznámka.** Například, nedeterministický krok (příkaz CHOOSE $L_1, L_2$) se popíše požadavkem splnění formule $d_{r,t-1} \Rightarrow (d_{L_1,t} \vee d_{L_2,t})$, neboli ekvivalentně v KNF $\bar{d}_{r,t-1} \vee d_{L_1,t} \vee d_{L_2,t}$ .

**Poznámka.**

1. Nyní již víme, že třída NPC je neprázdná, neboť podle Cookovy věty je SAT $\in$ NPC.
2. Jakmile víme, že NPC$\neq \emptyset$, je dokazování NP-úplnosti dalších problémů zpravidla snazší. Je-li $J \in$ NP, což bývá zřejmé, pak k důkazu, že $J \in$ NPC, stačí ukázat, že $K \triangleleft J$ pro některý vhodný jazyk $K \in$ NPC: z NP-úplnosti $K$ plyne, že všechny jazyky z NP lze redukovat na $K$, a z tranzitivity (tvrzení 1 věty 9.4) plyne redukovatelnost na $J$.

## 9.8 Některé další NP-úplné problémy

**Problém 3-splnitelnosti logických formulí – 3-SAT.**

**3-SAT**
**Vstup:** logická formule $f(x_1, \ldots, x_n) = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \ldots, \wedge (a_k \vee b_k \vee c_k)$, kde každé $a_i$, $b_i$, $c_i$ $(i = 1, \ldots k)$ je rovno $x_\ell$ nebo $\bar{x}_\ell$ pro vhodné $\ell = 1, \ldots, n$ (tj. $f$ je formule v KNF s klauzulemi délky 3).
**Úkol:** zjistit, zda je formule $f$ splnitelná.

**Věta 9.7.** *3-SAT $\in$ NPC.*

**Důkaz.** 1. 3-SAT $\in$ NP - to je zřejmé, neboť 3-SAT je speciální případ SAT.

2. K důkazu 3-SAT $\in$ NPC stačí redukovat SAT $\triangleleft$ 3-SAT (tvrzení 1 věty 9.4). K tomu nám poslouží malý trik. Všimněme si nejprve tohoto:

$u_1 \vee u_2 \vee \ldots \vee u_m = 1 \Leftrightarrow$ existuje $v$ takové, že platí $(u_1 \vee \ldots u_{m-2} \vee v) \wedge (u_{m-1} \vee u_m \vee \bar{v}) = 1$
(důkaz je celkem zbytečný, rovnost je patrná na první pohled). Ve formuli na pravé straně ekvivalence je první klauzule o jeden literál kratší a druhá má délku 3. Postupným prováděním můžeme upravit libovolnou formuli v KNF tak, aby všechny klauzule měly nejvýše tři literály. Toho, aby všechny klauzule měly délku právě 3, dosáhneme zřejmým způsobem - opakováním literálu. □

**Nezávislá množina – IND**

**Věta 9.8.** *IND $\in$ NPC.*

**Důkaz.** Redukce SAT $\triangleleft$ IND byla uvedena v odstavci 9.4. □

**Poznámka.** Uvažujme následující dvě varianty problému IND, definovaného v odstavci 9.4.

**IND$_=$**
**Vstup:** neorientovaný graf $G$ na $n$ uzlech a přirozené číslo $k \leq n$.
**Úkol:** zjistit, zda v grafu $G$ existuje nezávislá množina uzlů velikosti $k$.

**IND$_k$**
**Vstup:** neorientovaný graf $G$ na $n$ uzlech.
**Úkol:** zjistit, zda v grafu $G$ existuje nezávislá množina uzlů velikosti $k$.

Problém IND$_=$ se od problému IND liší pouze tím, že se ptáme na existenci nezávislé množiny velikosti *právě* $k$ (místo *alespoň* $k$ jako u IND). Snadno je vidět, že problémy IND a IND$_=$ jsou ekvivalentní: každá

množina velikosti $k$ je zároveň množinou velikosti alespoň $k$, a naopak, existuje-li v $G$ nějaká nezávislá množina velikosti $k' > k$, pak každá její podmnožina velikosti $k$ je také nezávislá. To znamená, že platí následující fakt.

**Tvrzení 9.1.** *$IND_= \in NPC$.*

Problém $\text{IND}_k$ se od problému $\text{IND}_=$ liší „pouze“ tím, že číslo $k$ není součástí vstupních dat, ale je pevně dané (tedy - hodnotu $k$ nám nezadává „nepřítel“, ale známe ji předem). Protože $n$-prvková množina má $\binom{n}{k}$ $k$-prvkových podmnožin a $\binom{n}{k} = O(n^k)$, řešení problému $\text{IND}_k$ „hrubou silou“ (probírkou všech možností) má složitost $O(n^k)$, a $k$ je konstanta. Dokázali jsme tak následující fakt:

**Tvrzení 9.2.** *$IND_k \in P$.*

Vidíme překvapivou skutečnost: zdánlivě „nepatrný“ rozdíl – zda je číslo $k$ pevně dané nebo zda je součástí vstupních dat – může rozhodnout o tom, jestli je problém polynomiální nebo NP-úplný. Je možno namítnout, že „nepřítel“ nám u problému $\text{IND}_=$ také přece vždy zadá konstantní $k$. Kde je tedy rozdíl? Tento rozdíl vynikne tehdy, uvědomíme-li si, že je-li $k$ součástí vstupních dat, pak nemusí být konstantní, ale může záviset na konkrétní hodnotě $n$. Představme si, že nám „nepřítel“ bude zadávat posloupnost grafů $G_1, G_2, \ldots$ takovou, že $|V(G_i)| = n = 2i$, a ke každému grafu $G_i$ (na $2i$ uzlech) nám zadá hodnotu $k = i = \frac{n}{2}$, $i = 1, 2, \ldots$. Pak počet všech $k$-prvkových podmnožin $n$-prvkové množiny $V(G_i)$ bude roven

$$\binom{n}{k} = \binom{2i}{i} = \frac{2i(2i-1)\ldots(i+1)}{i(i-1)\ldots 1} = \frac{2i}{i}\frac{2i-1}{i-1}\ldots\frac{i+1}{1} > 2^i = 2^{\frac{n}{2}} = (\sqrt{2})^n,$$

což znamená, že složitost algoritmu „hrubá síla“ je na těchto vstupních datech alespoň exponenciální.

Hranice mezi oběma světy (polynomiálním a NP-úplným) je velmi ostrá, a často velmi záleží i na přesné formulaci problému. Totéž platí i pro následující problémy COV a CLIQUE.

### Uzlové pokrytí – COV

**COV**
**Vstup:** neorientovaný graf $G$ na $n$ uzlech a přirozené číslo $k \leq n$.
**Úkol:** zjistit, zda v grafu $G$ existuje pokrytí velikosti nejvýše $k$.

**Poznámka.** Zdůrazněme opět, že úloha je formulována jako rozhodovací problém. O převodu optimalizačních úloh na rozhodovací jsme se již zmínili u úlohy IND, zde (a u dalších úloh) platí totéž.

**Věta 9.9.** *$COV \in NPC$.*

**Důkaz.** 1. Příslušnost do třídy NP je zřejmá (zformulujte si příslušný algoritmus!). Je tedy třeba dokázat existenci redukce.

2. $\text{IND} \lhd \text{COV}$, neboť $M \subset U(G)$ je $k$-prvková nezávislá množina právě tehdy, když $U(G) \setminus M$ je $n - k$ prvkové pokrytí. □

### Existence kliky předepsané velikosti – CLIQUE

**CLIQUE**
**Vstup:** neorientovaný graf $G$ na $n$ uzlech a přirozené číslo $k \leq n$.
**Úkol:** zjistit, zda v grafu $G$ existuje klika velikosti alespoň $k$.

**Věta 9.10.** *CLIQUE ∈ NPC.*

**Důkaz.** Příslušnost do třídy NP je opět zřejmá, a IND $\triangleleft$ CLIQUE, neboť $M \subset U(G)$ je $k$-prvková nezávislá množina v $G$ právě tehdy, když $M$ je $k$-prvková klika v $\bar{G}$. □

### 3-obarvitelnost grafu – 3-COL

**$k$-COL**
**Vstup:** neorientovaný graf $G$ na $n$ uzlech a přirozené číslo $k \leq n$.
**Úkol:** zjistit, zda je graf $G$ $k$-obarvitelný.

Pro $k = 1$ a $k = 2$, tj. pro případ 1- a 2-obarvitelnosti, již víme z kapitoly 7, že 1-COL$\in P$ a 2-COL$\in P$. Pro $k = 3$ je už ale vše jinak.

**Věta 9.11.** *3-COL ∈ NPC.*

**Důkaz.** 1. Zřejmě 3-COL ∈ NP (zformulujte si opět příslušný algoritmus).

2. Ukážeme, že 3-SAT $\triangleleft$ 3-COL. Nechť tedy $f(x_1, \ldots, x_n) = (a_1 \vee b_1 \vee c_1) \wedge \ldots \wedge (a_k \vee b_k \vee c_k)$; sestrojíme graf $G$, který je 3-obarvitelný právě tehdy, je-li formule $f$ 3-splnitelná. Barvy budeme označovat čísly 0, 1, 2.

Provedeme nejprve dvě pozorování na malých grafech (viz obrázek), které nám poslouží jako "stavební kameny" při konstrukci hledaného grafu.


V $G_1$ je zřejmé, že:
- jsou-li $a, b$ obarveny stejnou barvou, pak $d$ má tutéž barvu,
- jsou-li $a, b$ obarveny různými barvami, pak $d$ má libovolnou barvu.

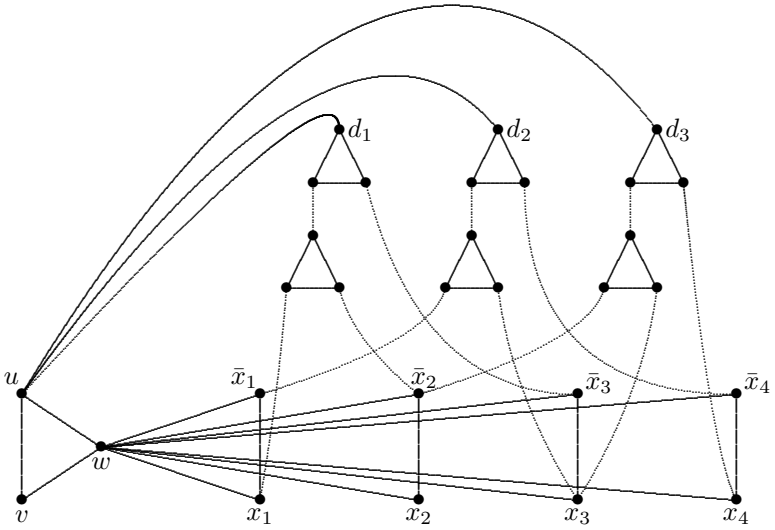
Obdobně, v $G_2$ je zřejmé, že:
- jsou-li $a, b, c$ obarveny stejnou barvou, pak $d$ má tutéž barvu,
- má-li alespoň jeden z uzlů $a, b, c$ jinou barvu než ostatní, pak $d$ může být obarven barvou 1.

Graf $G$ nyní sestrojíme touto konstrukcí:
- pro každou proměnnou $x_i$ sestrojíme dvojici uzlů $x_i, \bar{x}_i$ a spojíme ji hranou,
- přidáme tři uzly $u, v, w$ tvořící trojúhelník,
- uzel $w$ spojíme se všemi uzly $x_i, \bar{x}_i$,
- pro každou klauzuli formule $f$ vytvoříme jednu kopii grafu $G_2$ přičemž uzly $a, b, c$ budou totožné s uzly literálů klauzule, a uzel $d$ bude sousední s uzlem $u$.

Je třeba ještě dokázat, že $G$ je 3-obarvitelný právě když $f$ je splnitelná. Myšlenku této části důkazu ilustrujeme na příkladu. Mějme dánu logickou formuli:

$$f(x_1, \ldots, x_4) = (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_3 \vee \bar{x}_4) \wedge (\bar{x}_2 \vee x_3 \vee x_4)$$

Sestrojíme graf $G$:




Ukážeme, že $G$ je 3-obarvitelný, právě když $f$ je splnitelná.

1. Především: $G$ je obarvitelný $\Leftrightarrow$ $G$ je 3-obarvitelný tak, že uzel $u$ má barvu 0, uzel $v$ barvu 1 a uzel $w$ barvu 2 (pokud tomu tak není, vhodně přečíslujeme barvy, abychom dostali tento výsledek).

2. Uzel $w$ má nyní barvu 2. Potom uzly $x_i, \bar{x}_i$ mají barvy z množiny $\{0, 1\}$ a lze je interpretovat jako hodnoty logických proměnných.

3. A nyní již lze psát: Formule $f$ je 3-splnitelná $\Leftrightarrow$ existuje přiřazení hodnot 0, 1 proměnným $x_i$ tak, že v každé klauzuli je alespoň jednou jednička $\Leftrightarrow$ všechny uzly $d_i$ mohou mít barvu různou od 0 $\Leftrightarrow$ graf $G$ je 3-obarvitelný. □

**Poznámka.**

1. NP-úplnost problému $k$-obarvitelnosti pro $k > 3$ se dokáže analogicky, ale důkaz je složitější (základní konstrukce je obdobná, ale grafy, používané jako „stavební kameny“, jsou komplikovanější).
2. Mezi základní NP-úplné problémy patří také problém HAM (existence hamiltonovské kružnice). Příslušnost do třídy NP jsme dokázali v příkladu před definicí 9.9; NP-úplnost dokazovat nebudeme.
3. Poznamenejme ještě, že rozhodnutí, zda pro daný neorientovaný graf $G$ platí $\chi'(G) = \Delta(G)$ nebo $\chi'(G) = \Delta(G) + 1$ (viz Vizingova věta - Věta 7.8), je také NP-úplný problém.

# Reference


[1] R. Čada, T. Kaiser, Z. Ryjáček: Diskrétní matematika. Skripta ZČU Plzeň, 2004.

[2] J. Demel: Grafy a jejich aplikace. Academia, Praha 2002

[3] M. Fiedler: Speciální matice a jejich použití v numerické matematice. Teoretická knižnice inženýra, SNTL, Praha 1981.

[4] J. Holenda, Z. Ryjáček: Lineární algebra II – Úvod do diskrétní matematiky. Skripta ZČU Plzeň, 1995.

[5] J. Kolář, O. Štěpánková, M. Chytil: Logika, algebry a grafy. SNTL, Praha 1989.

[6] L. Kučera: Kombinatorické algoritmy. Matematický seminář SNTL, Praha 1989.

[7] J. Nešetřil: Teorie grafů. Matematický seminář SNTL, Praha 1979.

[8] J. Nešetřil, J. Matoušek: Kapitoly z diskrétní matematiky. Karolinum, Praha 2000.

[9] J. Plesník: Grafové algoritmy. Veda, Bratislava, 1983.