KATEDRA INFORMATIKY
PŘÍRODOVĚDECKÁ FAKULTA
UNIVERZITA PALACKÉHO

GRAFY A GRAFOVÉ ALGORITMY

ARNOŠT VEČERKA

VÝVOJ TOHOTO UČEBNÍHO TEXTU JE SPOLUFINANCOVÁN
EVROPSKÝM SOCIÁLNÍM FONDEM A STÁTNÍM ROZPOČTEM ČESKÉ REPUBLIKY

Olomouc 2007

Abstrakt

Tento text distančního vzdělávání seznamuje se základy teorie grafů a s grafovými algoritmy. Na začátku je popis jednotlivých typů grafů, popis různých způsobů jejich reprezentace a definice základních pojmů používaných v teorii grafů. Stěžejní částí studijního materiálu jsou grafové algoritmy, jež tvoří významnou třídu algoritmů a jsou prakticky používány při řešení úloh z různých oblastí.

Cílová skupina

Text je primárně určen pro posluchače prvního bakalářského studijního programu Aplikovaná informatika na Přírodovědecké fakultě Univerzity Palackého v Olomouci. Může však sloužit komukoli se zájmem o grafy a grafové algoritmy a jejich použití. Text předpokládá základní znalosti z algebry.

Obsah

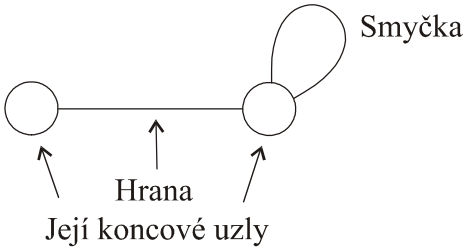
1 Grafy

Graf je poměrně obecný pojem. Vyskytuje se v různých významech. Zde bude tento pojem označovat grafický způsob vyjádření vztahů mezi nějakými objekty.

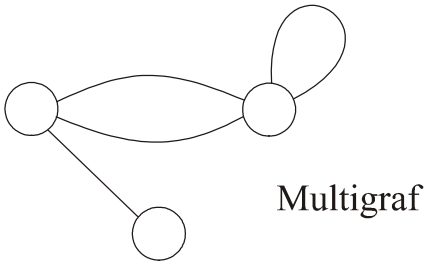
Objekty jsou v grafu reprezentovány uzly. Vedle pojmu uzel se také rovněž používá pojem vrchol. Vztahy jsou v grafu reprezentovány hranami. Skutečnost, že dva objekty jsou v určitém vztahu, se v grafu vyjádří spojením příslušných uzlů (vrcholů), jež tyto objekty v grafu reprezentují, hranou.

Na kreslení uzlů a hran nejsou žádná zvláštní omezení. Uzly zpravidla kreslíme kružnicemi, ale někdy také elipsami, obdélníky. Hrany kreslíme přímými čárami, oblouky nebo lomenými čárami. Způsob kreslení volíme především tak, aby graf byl přehledný.

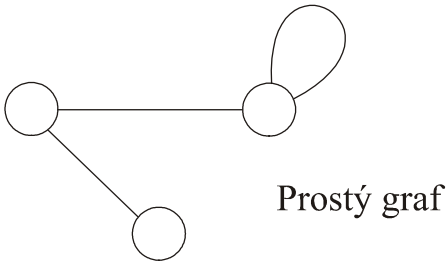
Hrana vždy začíná a končí v nějakém uzlu. Většinou jsou koncové uzly hrany různé, ale může to být i jeden uzel, pak takové hraně říkáme smyčka.



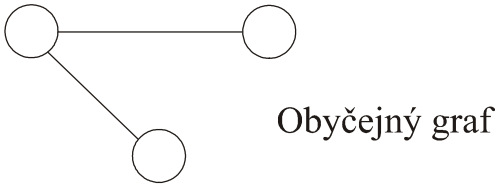
Dva uzly mohou být spojeny více hranami – takovým hranám říkáme násobné. Graf, ve kterém mezi některými uzly je více hran, nazýváme multigraf.



Graf bez násobných hran se nazývá prostý graf.



Prostý graf bez smyček je obyčejný graf.



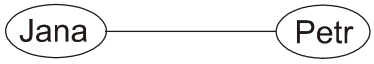
V praxi lze běžně vystačit s obyčejnými grafy. Proto my se v tomto studijním materiálu budeme zabývat jen tímto typem grafů. Od tohoto okamžiku pojem graf bude označovat obyčejný graf.

Nejčastěji se používají obyčejné grafy.

Hrany v grafu mohou být:

- Neorientované – reprezentují symetrické vztahy mezi uzly. Například jsou-li Petr a Jana bratr a sestra a hrana nám bude v grafu vyjadřovat vztah, že jsou sourozenci, bude tato hrana neorientovaná.

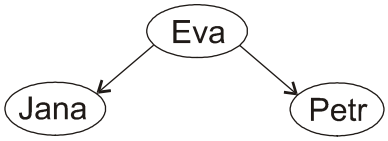
Neorientované grafy vyjadřují oboustranné vztahy.



Neorientovaná hrana

- Orientované – reprezentují jednosměrné, nesymetrické vztahy mezi uzly. Orientace hrany je vyznačena šipkou na jednom jejím konci. Například je-li Eva matkou Petra a Jany a hrana nám bude vyjadřovat vztah, že Eva je jejich rodičem, bude tato hrana orientovaná.

Jednostranné vztahy vyjadřují orientované grafy.



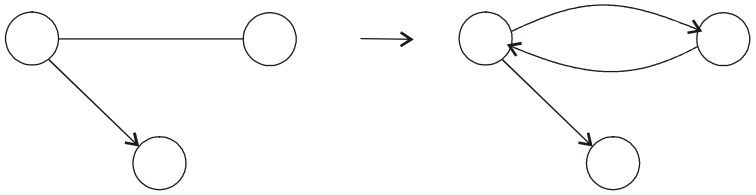
Orientované hrany

Podle typu hran dělíme grafy na:

- Neorientované – všechny jejich hrany jsou neorientované,
- Orientované – všechny jejich hrany jsou orientované.
- Smíšené – obsahují neorientované i orientované hrany.

Smíšené grafy se v praxi nepoužívají. Nahrazují se orientovanými grafy tak, že neorientované hrany se nahradí dvojicí opačně orientovaných hran.

Smíšený graf Ekvivalentní orientovaný graf



Nahrazení smíšeného grafu orientovaným grafem

V dalším textu pod pojmem graf budeme rozumět neorientovaný graf. Pokud se budeme zabývat orientovaným grafem, bude to explicitně uvedeno.

Průvodce studiem

Za zakladatele teorie grafů je považován známý švýcarský matematik Euler, který v roce 1736 jako první v historii pomocí grafu vyřešil jistou úlohu.Tato úloha nebyla z matematického hlediska nijak významná. Šlo o úlohu sedmi mostů města Královce. Euler hledal způsob, jak je projít postupně za sebou tak, aby po každém prošel jen jednou. Když se mu to nedařilo, pomocí grafu nakonec zjistil, že jejich topologie je taková, že to ani nejde.

1.1 Definice grafu

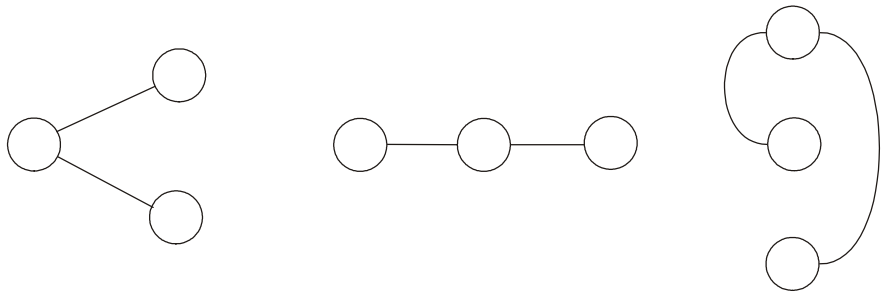
Studijní cíle: Zavést formální definici grafu a objasnit její význam.

Klíčová slova: Uzel, hrana, incidence, incidenční funkce, stupeň uzlu.

Potřebný čas: 1 hodina

Zatím jsme grafy reprezentovali jejich nakreslením. Tentýž graf můžeme nakreslit různě. Na následujícím obrázku je stejný graf třikrát různě nakreslen.

Graf nejčastěji reprezentujeme jeho nakreslením (diagramem).



Všechna tři nakreslení vyjadřují stejné vztahy mezi uzly. Aby se odlišilo vlastní nakreslení grafu od vztahů, které graf reprezentuje, někdy se konkrétnímu nakreslení grafu říká diagram grafu. Dále se zavádí formální definice grafu, která je nezávislá na nakreslení grafu (diagramu) a popisuje strukturu grafu a vztahy reprezentované grafem.

Definice grafu: Graf je trojice $G = (H, U, \rho)$, kde

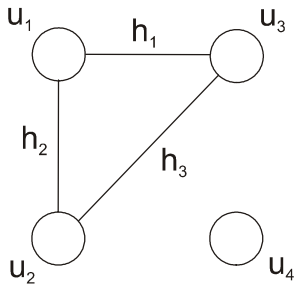
- H je množina hran $H = \{h_1, h_2, \ldots, h_n\}$
- U je množina uzlů $U = \{u_1, u_2, \ldots, u_m\}$
- ρ je incidenční zobrazení (incidenční funkce), které je dáno předpisem
  - pro neorientovaný graf

    $H \to U \otimes U$, kde $U \otimes U$ označuje množinu všech neuspořádaných dvojic prvků z množiny U.
  - a pro orientovaný graf

    $H \to U \times U$, kde $U \times U$ označuje kartézský součin (množinu všech uspořádaných dvojic prvků z množiny U).

Pro počet hran v množině H používáme zápis |H| (v předchozí definici je |H| = n) a pro pročet uzlů v množině U používáme zápis |U| (v předchozí definici je |U| = m).

Příklad. Graf zobrazený následujícím diagramem



lze definicí zapsat

$H = \{h_1, h_2, h_3\}$

$U = \{u_1, u_2, u_3, u_4\}$

$\rho(h_1) = \{u_1, u_3\}$

$\rho(h_2) = \{u_1, u_2\}$

$\rho(h_3) = \{u_2, u_3\}$

kde neuspořádané dvojice uzlů jsme zde zapsali jako dvouprvkové množiny.

Poznámka: Někdy se jako definice grafu užívá jen dvojice $G = (U, H)$, kde *U* je opět množina uzlů a *H* množina hran. Hrany jsou v tomto případě definovány dvojicemi svých koncových uzlů, přičemž u neorientovaného grafu hrany zapisujeme jako dvouprvkové množiny, u orientovaného jako uspořádané dvojice.

Například u grafu z předchozího příkladu tyto množiny budou:

$U = \{u_1, u_2, u_3, u_4\}$

$H = \{\{u_1, u_3\}, \{u_1, u_2\}, \{u_2, u_3\}\}$

Tato definice je sice úspornější, nicméně v mnoha případech se hůře používá.

Incidence je vztah mezi hranou a uzlem.

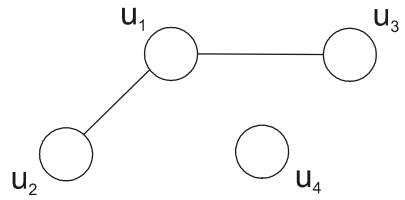
Definice. Stupeň uzlu je počet hran, které jsou s tímto uzel spojeny - mají tento uzel jako koncový. Pro skutečnost, že hrana má daný uzel jako koncový, používáme termín, že hrana s tímto uzlem *inciduje*.

Pro označení stupně uzlu *u* používáme zápis

$d(u)$

Písmeno *d* je zde od anglického slova *degree* – stupeň.

Příklad. Následující graf



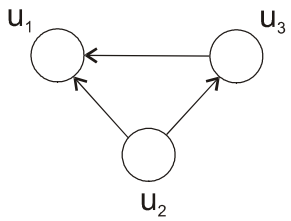
má stupně uzlů

$d(u_1)=2 \quad d(u_2)=1 \quad d(u_3)=1 \quad d(u_4)=0$.

U orientovaného grafu navíc rozeznáváme výstupní stupeň uzlu, což je počet hran, které z něho vychází, označujeme ho $d^-(u)$, a vstupní stupeň uzlu, což je počet hran, které do něho vchází, označuje ho $d^+(u)$. Zřejmě pro stupeň uzlu v orientovaném grafu platí

$d(u) = d^-(u) + d^+(u)$.

Příklad. Následující graf



má stupně uzlů

$d^-(u_1)=0 \quad d^+(u_1)=2 \quad d(u_1)=2$

$d^-(u_2)=2 \quad d^+(u_2)=0 \quad d(u_2)=2$

$d^-(u_3)=1 \quad d^+(u_3)=1 \quad d(u_3)=2$

Věta. Součet stupňů všech uzlů v grafu je sudé číslo.

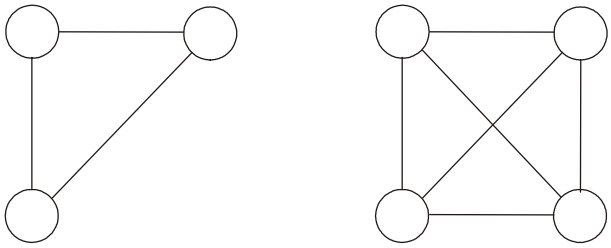
Důkaz: Každá hrana inciduje se dvěma uzly. Tedy součet stupňů všech uzlů v grafu je roven dvojnásobku počtu hran v grafu, čímž je sudé číslo.

Definice. V teorii grafů se používají termíny:

- Pro uzel, jehož stupeň je nulový, používáme název diskrétní uzel.
- Graf, jehož všechny uzly jsou diskrétní, nazveme diskrétní graf.
- Dva uzly, jež jsou spojeny hranou, nazveme sousedními uzly.
- Pro graf, jehož všechny uzly jsou sousední (má při daném počtu uzlů maximální počet hran), se používá označení úplný graf.

Sousednost je vztah mezi uzly.

Příklad. Na následujícím obrázku jsou dva úplné grafy, první se 3 uzly a druhý se 4 uzly.



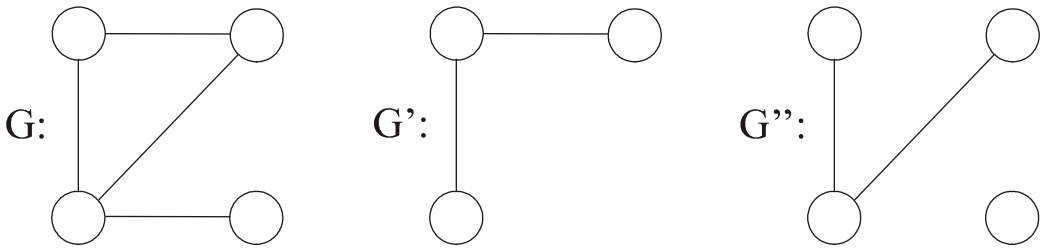
Definice podgrafu: Nechť je dán graf $G = (H, U, \rho)$. Pak graf $G' = (H', U', \rho')$ takový, že

- $H' \subseteq H$ (množina hran grafu *G'* je podmnožinou množiny hran grafu *G*) ,
- $U' \subseteq U$ (množina uzlů grafu *G'* je podmnožinou množiny uzlů grafu *G*) ,
- pro každou hranu $h \in H'$ platí $\rho'(h)=\rho(h)$ (incidenční funkce grafu *G'* je zúžením incidenční funkce grafu *G*)

nazveme podgrafem grafu *G*.

V případě, kdy je $U'=U$ (množina uzlů zůstane zachována), pro podgraf *G'* se používá název faktor grafu *G*.

Příklad. Na následujícím obrázku je graf *G* a jeho dva podgrafy *G'* a *G''*. Přičemž podgraf *G''* je zároveň i faktorem grafu *G* (zachovává jeho množinu uzlů).

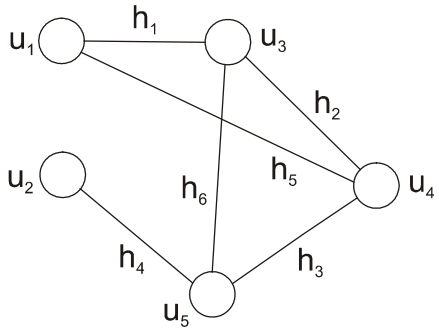


Kontrolní otázky

1. Jak je definována incidenční funkce neorientovaného grafu? A jak je definována u orientovaného grafu?
2. Co je stupeň uzlu grafu?

Cvičení

1. Napište incidenční funkci obyčejného grafu, jež je na následujícím obrázku.



1.
2. Napište stupně jednotlivých uzlů grafu z 1. cvičení.
3. Doplňte graf z 1. cvičení hranami tak, aby byl úplný. Kolik bude mít nyní hran?
4. Kolik může mít obyčejný graf, který má *m* uzlů, nejvýše hran?

Úkoly k textu

Orientovaným grafem lze reprezentovat binární relaci *R* a to tak, že *xRy* právě když existuje orientovaná hran z uzlu *x* do uzlu *y*. Zamyslete se nad tím, jak bude vypadat graf tranzitivní relace. A zkuste stanovit, jaký typ binární relace vyjadřuje neorientovaný graf.

Řešení

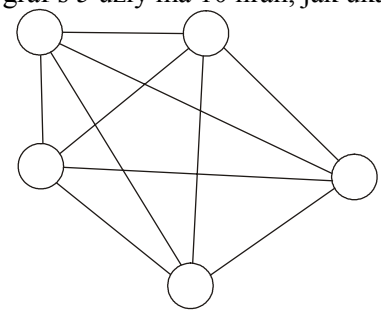
1. Incidenční funkce grafu je

$$\rho(h_1) = \{u_1, u_3\} \qquad \rho(h_2) = \{u_3, u_4\} \qquad \rho(h_3) = \{u_4, u_5\}$$
$$\rho(h_4) = \{u_2, u_5\} \qquad \rho(h_5) = \{u_1, u_4\} \qquad \rho(h_6) = \{u_3, u_5\}$$

2. Stupně uzlů jsou

$$d(u_1)=2 \quad d(u_2)=1 \quad d(u_3)=3 \quad d(u_4)=3 \quad d(u_5)=3$$

3. Úplný graf s 5 uzly má 10 hran, jak ukazuje následující obrázek.



4. Nejvíce hran má úplný graf. V něm každý jeho uzel sousedí se zbývajícími *m*-1 uzly. Uvážíme-li, že uzlů je celkově *m*, nicméně každá hrana je současně sdílena dvěma uzly, dostáváme pro počet hran obyčejného grafu vztah

$$počet\ hran \leq \frac{m*(m-1)}{2}.$$

1.2 Souvislost grafu

Studijní cíle: Vysvětlit pojem souvislého grafu.

Klíčová slova: Sled, tah, cesta.

Potřebný čas: 1 hodina

Definice. Nechť je dán graf $G = (H, U, \rho)$ a dva jeho uzly *u* a *v*. Sledem mezi uzly *u* a *v* nazveme posloupnost uzlů a hran

$$u, h_{i1}, u_{i1}, h_{i2}, u_{i2}, \ldots, u_{ik-1}, h_{ik}, v$$

pro kterou platí

$$\rho(h_{ir}) = \{u_{ir-1}, u_{ir}\} \quad \text{pro} \quad r = 1, \ldots, k \ .$$

Dále si označíme

$$u_{i0} = u \ \text{ a } \ u_{ik} = v \ .$$

Tedy sled je na sebe navazující posloupnost hran, kdy vždy dvě za sebou následující hrany ve sledu mají společný koncový uzel, který je ve sledu uveden mezi nimi.

Je-li $u=v$ (počáteční a koncový uzel sledu je stejný), jde o uzavřený sled.

Tah mezi uzly *u* a *v* je sled mezi těmito dvěma uzly, ve kterém žádná hrana se nevyskytuje vícekrát. Tj. $h_{ir} \neq h_{is}$ pro $r \neq s$. Je-li $u=v$, jde o uzavřený tah.

Cesta mezi uzly *u* a *v* je tah mezi těmito dvěma uzly, ve kterém se žádný jeho vnitřní uzel nevyskytuje vícekrát. Tj. $u_{ir} \neq u_{is}$ pro $r \neq s$, $r=0, \ldots, k$, $s=1, \ldots, k-1$. Uzavřená cesta (je-li $u=v$) je označována jako kružnice grafu.

Kružnice je uzavřená cesta.

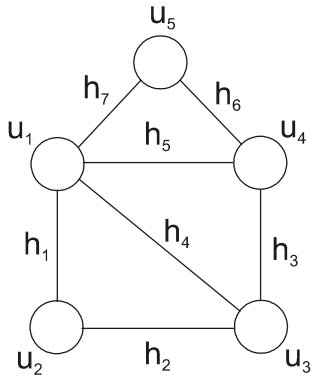
Pro orientované grafy mám pojmy:

Orientovaný sled - všechny jeho hrany mají stejnou orientaci – od počátečního uzlu *u* ke koncovému uzlu *v*. Pomocí incidenční funkce to zapíšeme

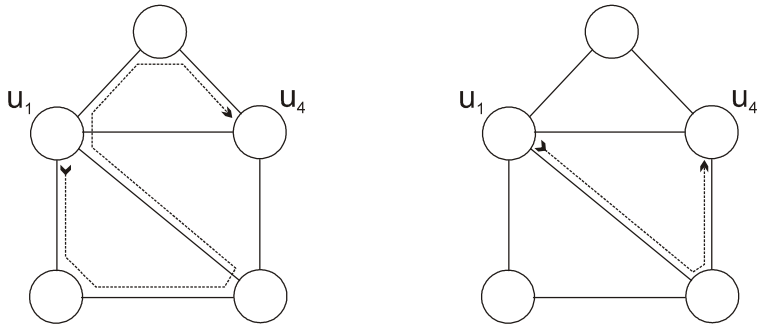
$$\rho(h_{ir}) = \langle u_{ir-1}, u_{ir} \rangle \quad \text{pro} \quad r = 1, \ldots, k, \ u_{i0} = u, \ u_{ik} = v$$

Dále jsou to pojmy orientovaný tah, orientovaná cesta a cyklus. Cyklus je označení pro orientovanou kružnici.

Příklad. V následujícím grafu



je $u_1, h_1, u_2, h_2, u_3, h_4, u_1, h_7, u_5, h_6, u_4$ je tah mezi uzly u_1 a u_4. A dále u_1, h_4, u_3, h_3, u_4 je cesta mezi těmito uzly. Jejich průběh ukazuje následující obrázek.



Věta. Z každého sledu spojující dva různé uzly u a v lze vybrat cestu, která je spojuje.

Z každého sledu lze vybrat cestu.

Důkaz: Vezměme sled

$$u=u_{i0}, h_{i1}, u_{i1}, \ldots, h_{ir}, u_{ir}, h_{ir+1}, \ldots, h_{is}, u_{is}, h_{is+1}, \ldots, u_{ik-1}, h_{ik}, u_{ik}=v$$

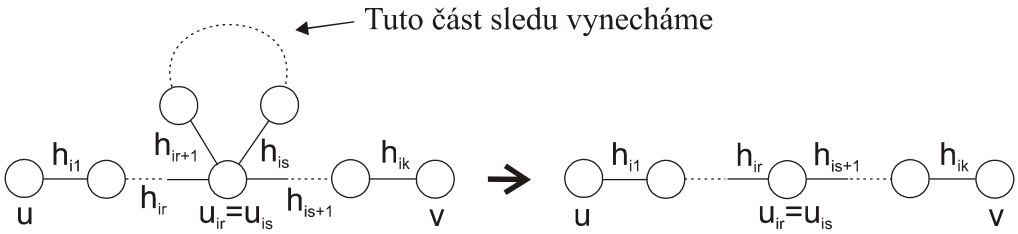
A nechť některý uzel u_{ir} je ve sledu stejný jako uzel u_{is}, kde $r<s$. Pak můžeme ve sledu jeho část

$$h_{ir+1}, \ldots, h_{is}, u_{is}$$

vynechat a dostaneme tím kratší sled spojující uzly u a v

$$u=u_{i0}, h_{i1}, u_{i1}, \ldots, h_{ir}, u_{ir}=u_{is}, h_{is+1}, \ldots, u_{ik-1}, h_{ik}, u_{ik}=v$$

Vynechání části ukazuje následující obrázek



Tento postup opakujeme tak dlouho, dokud ve sledu nejsou odstraněny všechny násobné výskyty uzlů. Tím se ze sledu stane cesta.

Příklad. V předchozím příkladu jsme měli tah $u_1, h_1, u_2, h_2, u_3, h_4, u_1, h_7, u_5, h_6, u_4$. V něm se uzel u_1 vyskytuje dvakrát. Podle důkazu předchozí věty v tahu vynecháme jeho část $h_1, u_2, h_2, u_3, h_4, u_1$. Zůstane nám u_1, h_7, u_5, h_6, u_4, což je už cesta mezi uzly u_1 a u_4.

Definice. Graf, mezi jehož každými dvěma uzly existuje cesta, nazveme souvislým grafem.

V souvislém grafu existuje cesta mezi každou dvojicí jeho uzlů.

Příklad.

Souvislý graf Nesouvislý graf

Věta. Souvislý (neprázdný) graf s m uzly má nejméně m-1 hran.

Důkaz: Matematickou indukcí podle počtu uzlů v grafu.

1. Pro m=1 je graf tvořen jen jedním uzlem. Hranu nemá žádnou, čímž je splněno, že má nejméně m-1 hran.
2. Nechť tvrzení věty platí pro nějaké m. Dokážeme, že platí i pro m+1. Souvislý graf s m+1 uzly vytvoříme tak, že jako výchozí vezmeme souvislý graf s m uzly. Ten má podle indukčního předpokladu nejméně m-1 hran. Nyní k němu přidáme ještě jeden uzel. Aby graf zůstal souvislý, musíme tento uzel spojit aspoň jednou hranou s některým uzlem ze stávajícího grafu. Po přidání uzlu a aspoň jedné hrany bude graf mít nejméně m hran, což můžeme napsat jako (m+1)-1 hran, tedy tvrzení věty platí i pro graf s m+1 uzly.

Definice. Komponentou grafu nazveme každý jeho maximální souvislý podgraf. Přitom souvislý podgraf daného grafu považujeme za maximální, jestliže ho už nelze zvětšit přidáním dalších hran, či uzlů daného výchozího grafu tak, aby podgraf byl stále souvislý. Tedy není vlastním podgrafem jiného souvislého podgrafu.

Věta. Nechť souvislý graf s m uzly má p komponent. Pak má nejméně m-p hran.

Důkaz: Označme $m_1, m_2, \ldots, m_p$ počty uzlů v jednotlivých komponentách grafu. Protože množiny uzlů jednotlivých komponent jsou disjunktní, zřejmě platí

$$m_1 + m_2 + \ldots + m_p = m \quad .$$

Komponenty jsou souvislými podgrafy. Tudíž podle předchozí věty platí, že

1. komponenta obsahuje nejméně m_1-1 hran

2. komponenta obsahuje nejméně m_2-1 hran

…….

k-tá komponenta obsahuje nejméně m_p-1 hran

Odtud sečtením dostaneme, že graf obsahuje nejméně hran:

$$(m_1 - 1) + (m_2 - 1) + \ldots + (m_k - 1) = m_1 + m_2 + \ldots + m_k - p = m - p$$

U orientovaného grafu rozeznáváme dva stupně souvislosti: souvislý a silně souvislý graf.

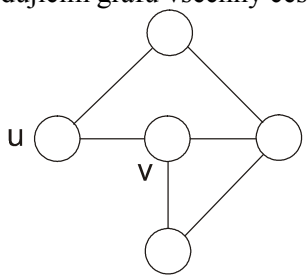
Definice. Orientovaný graf je souvislý, jestliže mezi každými jeho dvěma uzly *u* a *v* existuje orientovaná cesta buďto z uzlu *u* do uzlu *v* nebo z uzlu *v* do uzlu *u*. Orientovaný graf je silně souvislý, jestliže mezi každými jeho dvěma uzly *u* a *v* existuje orientovaná cesta z uzlu *u* do uzlu *v* a rovněž opačná cesta z uzlu *v* do uzlu *u*.

Kontrolní otázky

1. *Jaký je rozdíl mezi tahem a cestou?*
2. *Co je kružnice v grafu a co cyklus?*
3. *Kolik má souvislý graf s m uzly nejméně hran?*

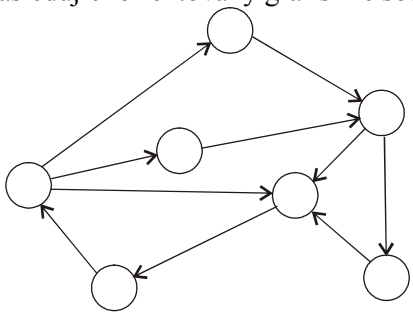
Cvičení

1. Najděte v následujícím grafu všechny cesty mezi uzly *u* a *v*.



2.

2. Je následující orientovaný graf silně souvislý?



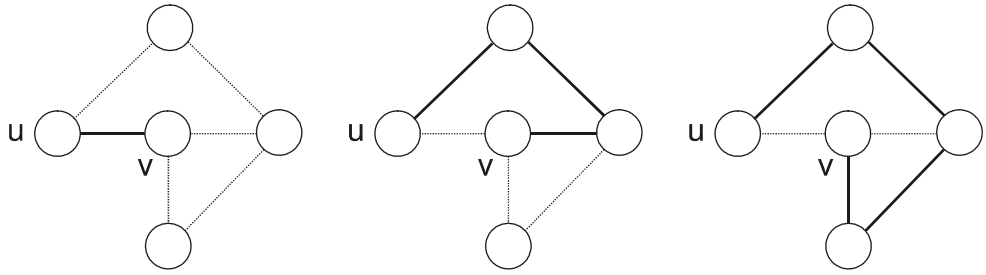
3.

Úkoly k textu

Zamyslete se nad tím, jak vypadá graf, který má stejný počet komponent jako hran. A jak vypadá graf, který má více komponent než hran.

Řešení

1. V grafu jsou celkem 3 různé cesty mezi uzly *u* a *v*.



2. Graf je silně souvislý.

1.3 Reprezentace grafů

Studijní cíle: Seznámit studujícího s různými možnostmi reprezentace grafu.

Klíčová slova: Matice incidence, matice sousednosti.

Potřebný čas: 1 a půl hodiny

Graf můžeme reprezentovat (popsat) různými způsoby:

- diagramem (nakreslením)
- definicí
- maticemi
- datovými strukturami (v programech)

První dva způsoby jsme už poznali. Věnujme se nyní těm zbývajícím dvěma.

1.3.1 Maticový popis grafu

Maticový popis grafu vychází ze dvou základních vztahů v grafu. Tím prvním je vztah mezi hranou a její koncovým uzlem, který jsme nazvali vztahem incidence. Ten popisuje matice incidence grafu. Druhým vztahem je sousednost uzlů. Ten popisuje matice sousednosti grafu. Zde uvedeme jen matici sousednosti, neboť ta má větší význam než matice incidence.

Maticí sousednosti lze reprezentovat graf.

Definice. Nechť je dán obyčejný graf $G = (H, U, \rho)$ s množinou hran $H = \{h_1, h_2, \ldots, h_n\}$ a množinou uzlů $U = \{u_1, u_2, \ldots, u_m\}$. Matice sousednosti je čtvercová matice řádu *m*, kterou si označíme *S*. Její prvky s_{jk} jsou dány předpisem:

U neorientovaného grafu

- s_{jk}=1, jestliže uzly u_j a u_k jsou sousední
- s_{jk}=0, jestliže uzly u_j a u_k sousední nejsou

U orientovaného grafu

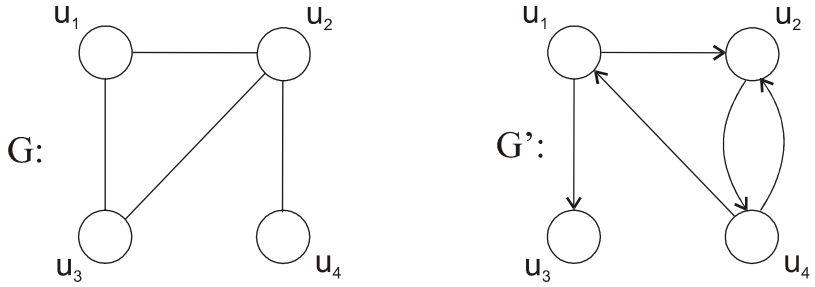
- s_{jk}=1, jestliže je hrana z uzlu u_j do uzlu u_k
- s_{jk}=0, jestliže není hrana z uzlu u_j do uzlu u_k

Z definice plynou některé vlastnosti matice sousednosti *S*:

- na hlavní diagonále jsou nuly (uvažujeme jen obyčejné grafy, které nemají smyčky)

- u neorientovaného grafu je symetrická podle hlavní diagonály
- počet jedniček v matici je u neorientovaného grafu roven dvojnásobku počtu hran
- počet jedniček v matici je u orientovaného stejný jako počet hran

Příklad. Pro následující dva grafy



sestavíme jejich matice sousednosti

$$S = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \qquad S' = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

1.3.2 Reprezentace grafu v programech

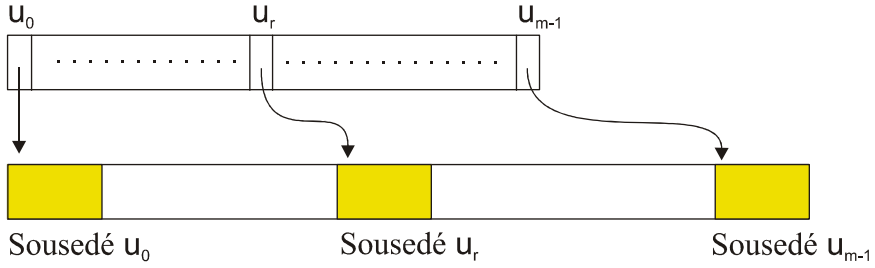
Graf si můžeme v programu reprezentovat různými způsoby. Můžeme si ho například uložit jako matici sousednosti. Ovšem u rozsáhlejších grafů s větším počtem uzlů je tato matice značně velká a navíc její použití v algoritmech je poměrně neefektivní, neboť v ní třeba musíme pracně . Uveďme dva často používané způsoby reprezentace grafu v programech.

1.3.2.1 Reprezentace pomocí polí

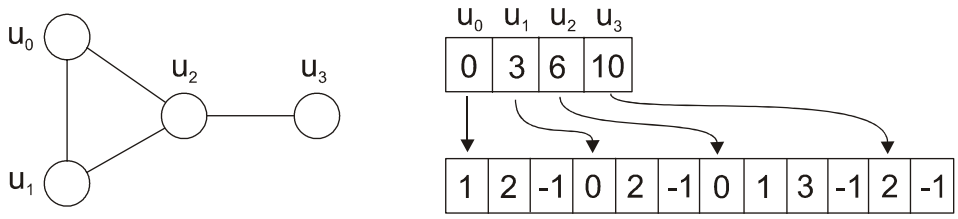
V programu můžeme graf reprezentovat dvěma poli.

Struktura grafu je uložena ve dvou polích. První pole má stejný počet prvků, jako je počet uzlů v grafu. Každému uzlu odpovídá jeden prvek pole. V něm je uložena hodnota indexu, od kterého v druhém poli začíná seznam uzlů, jež jsou sousedé tohoto uzlu.

Přepokládejme že množina uzlů grafu je $U = \{u_0, u_2, \ldots, u_{m-1}\}$. Tedy uzly jsou číslovány od 0 do *m*-1. Dále předpokládejme, že počáteční indexy polí jsou 0.



Na následujícím příkladu je graf se 4 uzly a jeho reprezentace pomocí polí. Hodnoty –1 na konci seznamů sousedů slouží k tomu, aby se poznalo, kde seznamy sousedů končí.



1.3.2.2 Reprezentace dynamickou datovou strukturou

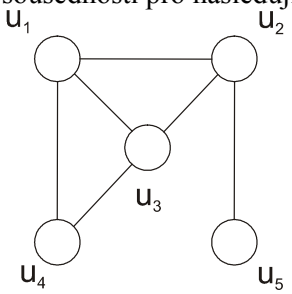
Další možnost je uzel grafu reprezentovat jako strukturovaný datový typ. Každý uzel obsahuje pole (seznam) ukazatelů na sousední uzly.

Kontrolní otázky

1. *Jaký je definována matice sousednosti neorientovaného grafu?*
2. *Jak reprezentujeme graf v programu?*

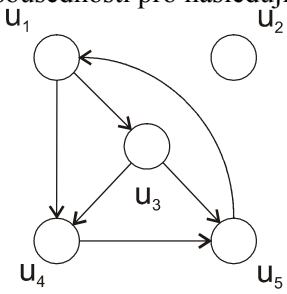
Cvičení

1. Sestavte matici sousednosti pro následující graf.



4.

2. Sestavte matici sousednosti pro následují orientovaný graf.



5.

Úkoly k textu

Matice sousednosti neorientovaného grafu ho až na označení uzlů a hran zcela určuje. Nepochybně se z ní dá poznat, zda graf je souvislý, a dále i kolik má komponent. Zkuste najít způsob, jak byste to z matice sousednosti určili.

Řešení

1. Matice sousednosti je

$$S = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

2. Matice sousednosti daného orientovaného grafu je

$$S = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

2 Průchod grafem (prohledání grafu)

Studijní cíle: Vysvětlit způsoby systematického průchodu grafem.

Klíčová slova: Průchod do hloubky, průchod do šířky.

Potřebný čas: 2 hodiny

Při některých použitích grafů máme v uzlech uložené určité údaje. Takovým grafům říkáme uzlově ohodnocené grafy. Průchod grafem řeší úlohu, kdy potřebujeme ve všech uzlech nad údaji provést nějakou operaci. Tedy potřebujeme projít všechny uzly grafu. Existují dva systematické průchody grafem – do hloubky a do šířky.

2.1 Průchod do hloubky

Průchod grafem znamená projít všechny uzly grafu.

Při průchodu grafem je nutné zajistit, abychom prošli všechny uzly a rovněž abychom žádný uzel neprocházeli vícekrát. Proto tyto účely se do uzlů přidává proměnná, podle které se pozná, zda v daném uzlu už jsme byli a provedli jsme v něm potřebné operace. Například do uzlů můžeme přidat celočíselnou proměnnou, kterou při vytváření reprezentace grafu nastavíme u všech uzlů na hodnotu 0. Označme tuto proměnnou *navst* (od slova navštíven). Dále si zavedeme proměnnou, kterou nazveme *pruchod* a na začátku do ní uložíme rovněž hodnotu 0. Tato proměnná bude sloužit jako čítač průchodů.

Při průchodu grafem do hloubky používáme zásobník.

Dále si zavedeme označení pro aktuální uzel (uzel, ve kterém právě jsme - který je právě navštíven). Budeme ho značit *u*.

Při průchodu do hloubky se k dočasnému uložení uzlů, které v dané chvíli nemůžeme navštívit používá zásobník.

Průchod můžeme začít od libovolného uzlu grafu. Počáteční uzel, od kterého začneme průchod, označíme u_0.

Popis algoritmu.

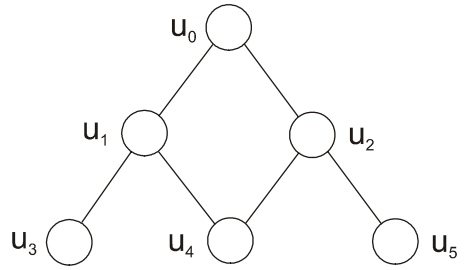
1. Počáteční nastavení:
   - Zásobník je na začátku prázdný.
   - Zvýšíme čítač průchodů o 1: $pruchod = pruchod + 1$.
   - Aktuální uzel nastavíme na počáteční uzel: $u = u_0$.
2. Krok navštívení aktuálního uzlu *u*.
   - V aktuálním uzlu *u* provedeme úkony, pro které jsme ho navštívili.
   - Nastavíme proměnnou *navst* na číslo průchodu jako příznak, že jsem v tomto uzlu již byli: $u.navst = pruchod$.
   - Zjistíme, kolik má aktuální uzel doposud nenavštívených sousedů. Mohou nastat případy:
     a) Nemá žádného nenavštíveného souseda. V tom případě přejdeme ke kroku 3., ve kterém jako další aktuální uzel vezmeme uzel, který je uložen na zásobníku.
     b) Má právě jednoho nenavštíveného souseda. Učiníme ho novým aktuálním uzlem *u* a přejdeme opět na začátek kroku 2 - navštívíme ho.

c) Má více nenavštívených sousedů. Jednoho z nich učiníme novým aktuálním uzlem u a zbývající uložíme na zásobník. A přejdeme na začátek kroku 2.

3. Krok odebírání uzlů ze zásobníku.

- Je-li zásobník prázdný, průchod je ukončen. Všechny uzly byly navštíveny.
- Odebereme ze zásobníku uzel, označme ho v. I když v době, kdy byl na zásobník ukládán, byl nenavštívený, mohl být od té doby navštíven po jiné cestě. Proto ověříme, zda nebyl již navštíven. V tom případě by hodnota jeho proměnné *v.navst* byla stejná jako hodnota proměnné *pruchod*. Pokud tomu tak je, přejdeme k odebrání dalšího uzlu ze zásobníku – přejdeme opět na začátek kroku 3.
- Pokud odebraný uzel v je doposud nenavštívený, učiníme ho novým aktuálním uzlem $u = v$ a navštívíme ho – přejdeme ke kroku 2.

Příklad. Máme projít uzly následujícího grafu.



1. Počáteční podmínky.

- Zásobník je prázdný
- $pruchod = pruchod + 1$
- Aktuální uzel nastavíme na u_0

2. Aktuální uzel: u_0, zásobník prázdný

$u_0.navst = pruchod$ (v uzlu u_0 nastavíme příznak, že už je navštíven)

Uzel u_0 má dva nenavštívené sousedy u_1 a u_2. První z nich učiníme aktuálním uzlem a druhý uložíme na zásobník. Přejdeme opět ke kroku 2.

2. Aktuální uzel: u_1, zásobník: u_2

$u_1.navst = pruchod$ (v uzlu u_1 nastavíme příznak, že už je navštíven)

Uzel u_1 má dva nenavštívené sousedy u_3 a u_4. První z nich učiníme aktuálním uzlem a druhý uložíme na zásobník. Přejdeme opět ke kroku 2.

2. Aktuální uzel: u_3, zásobník: u_2 u_4

$u_3.navst = pruchod$

Uzel u_3 nemá žádného nenavštíveného souseda. Přejdeme ke kroku 3.

3. Odebereme uzel u_4 ze zásobníku. Uzel u_4 není doposud navštíven, učiníme ho aktuálním uzlem a přejdeme ke kroku 2.

2. Aktuální uzel: u_4, zásobník: u_2

$u_4.navst = pruchod$

Uzel u_4 nemá žádného nenavštíveného souseda. Přejdeme ke kroku 3.

3. Odebereme uzel u_2 ze zásobníku. Uzel u_2 není doposud navštíven, učiníme ho aktuálním uzlem a přejdem ke kroku 2.

2. Aktuální uzel: u_2, zásobník prázdný

   $u_2.navst = pruchod$

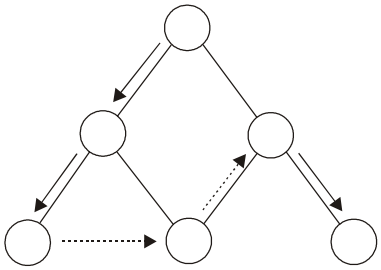
   Uzel u_2 má jednoho nenavštíveného souseda u_5. Učiníme ho aktuálním uzlem a přejdeme opět ke kroku 2.

2. Aktuální uzel: u_5, zásobník prázdný

   $u_5.navst = pruchod$

   Uzel u_5 nemá žádného nenavštíveného souseda. Přejdeme ke kroku 3.

3. Zásobník je prázdný – průchod je grafem je ukončen. Následující obrázek ukazuje, v jakém pořadí byly v tomto příkladu procházeny jednotlivé uzly grafu. Plné čáry naznačují přímý přechod k dalšímu uzlu, čárkované ukazují přechod k uzlu, který byl odebrán ze zásobníku.



2.2 Průchod do šířky

Při průchodu grafem do šířky používáme frontu.

Při průchodu grafem do šířky je datovou strukturou, do které se ukládají doposud nenavštívené uzly, fronta. Na rozdíl od průchodu do hloubky se zde jako další navštěvovaný uzel nevybírá soused aktuálního uzlu, ale vždy se bere uzel z fronty. Uzly jsou proto navštěvovány v tom pořadí, v jakém byly ukládány do fronty.

Pro popis algoritmu použijeme stejná označení, jaká byla použita v popisu algoritmu průchodu do hloubky.

Popis algoritmu.

1. Počáteční podmínky.
   - Fronta je na začátku prázdná.
   - Zvýšíme o 1 čítač průchodů: $pruchod = pruchod + 1$.
   - Aktuální uzel nastavíme na počáteční uzel: $u = u_0$.
2. Krok navštívení aktuálního uzlu u.
   - V aktuálním provedeme úkony, pro které jsme ho navštívili.
   - Nastavíme proměnnou *navst* na číslo průchodu jako příznak, že jsem v tomto uzlu již byli: $u.navst = pruchod$.
   - Všechny nenavštívené sousedy aktuálního uzlu (má-li nějaké) uložíme do fronty.
   - Přejdeme ke kroku 3., ve kterém stanovíme další aktuální uzel.
3. Krok odebírání uzlů z fronty.
   - Je-li fronta prázdná, průchod je ukončen. Všechny uzly byly navštíveny.
   - Odebereme z fronty uzel, označme ho v. I když v době, kdy byl do fronty ukládán, byl nenavštívený, mohl být od té doby navštíven po jiné cestě. Proto ověříme, zda nebyl již

navštíven. V tom případě by hodnota jeho proměnné *v.navst* byla stejná jako hodnota proměnné *pruchod*. Pokud tomu tak je, přejdeme k odebrání dalšího uzlu z fronty – přejdeme opět na začátek kroku 3.

- Pokud uzel je doposud nenavštívený, učiníme ho novým aktuálním uzlem $u = v$ a navštívíme ho – přejdeme ke kroku 2.

Příklad. Máme projít uzly stejného grafu jako v příkladu uvedeném u průchodu do hloubky.

1. Počáteční nastavení:
   - Fronta je prázdná
   - $pruchod = pruchod + 1$
   - Aktuální uzel nastavíme na u_0
2. Aktuální uzel: u_0, fronta prázdná

   $u_0.navst = pruchod$ (v uzlu u_0 nastavíme příznak, že už je navštíven)

   Uzel u_0 má dva nenavštívené sousedy u_1 a u_2. Uložíme je do fronty a přejdeme ke kroku 3.
3. Fronta: u_1 u_2

   Odebereme uzel u_1 z fronty. Uzel u_1 není doposud navštíven, učiníme ho aktuálním uzlem a přejdeme ke kroku 2.
2. Aktuální uzel: u_1, fronta: u_2

   $u_1.navst = pruchod$

   Uzel u_1 má dva nenavštívené sousedy u_3 a u_4. Uložíme je do fronty a přejdeme ke kroku 3.
3. Fronta: u_2 u_3 u_4

   Odebereme uzel u_2 z fronty. Uzel u_2 není doposud navštíven, učiníme ho aktuálním uzlem a přejdeme ke kroku 2.
2. Aktuální uzel: u_2, fronta: u_3 u_4

   $u_2.navst = pruchod$

   Uzel u_2 má dva nenavštívené sousedy u_4 a u_5. Uložíme je do fronty a přejdeme ke kroku 3.
3. Fronta: u_3 u_4 u_4 u_5

   Odebereme uzel u_3 z fronty. Uzel u_2 není doposud navštíven, učiníme ho aktuálním uzlem a přejdeme ke kroku 2.
2. Aktuální uzel: u_3, fronta: u_4 u_4 u_5

   $u_3.navst = pruchod$

   Uzel u_3 nemá žádného nenavštíveného souseda. Přejdeme ke kroku 3.
3. Fronta: u_4 u_4 u_5

   Odebereme uzel u_4 z fronty. Uzel u_4 není doposud navštíven, učiníme ho aktuálním uzlem a přejdeme ke kroku 2.
2. Aktuální uzel: u_4, fronta: u_4 u_5

   $u_4.navst = pruchod$

   Uzel u_4 nemá žádného nenavštíveného souseda. Přejdeme ke kroku 3.
3. Fronta: u_4 u_5

Odebereme uzel u_4 z fronty. Uzel u_4 byl již navštíven, přejdeme opět ke kroku 3.

3. Fronta: u_5

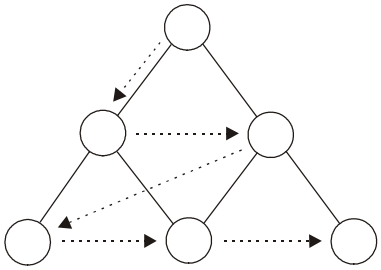
   Odebereme uzel u_5 z fronty. Uzel u_5 není doposud navštíven, učiníme ho aktuálním uzlem a přejdeme ke kroku 2.

3. Aktuální uzel: u_5, fronta prázdná

   $u_5.navst = pruchod$

   Uzel u_5 nemá žádného nenavštíveného souseda. Přejdeme ke kroku 3.

3. Fronta je prázdná – průchod je grafem je ukončen. Následující obrázek ukazuje, v jakém pořadí byly v tomto příkladu procházeny jednotlivé uzly grafu.

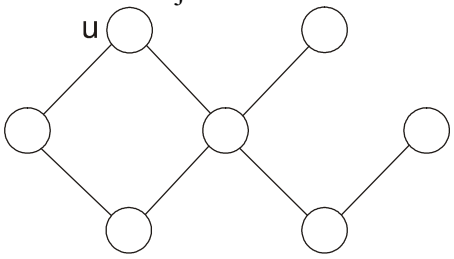


Kontrolní otázky

1. *Jakou abstraktní datovou strukturu použijeme pro průchod grafem do hloubky?*
3. *Jakou abstraktní datovou strukturu použijeme pro průchod grafem do šířky?*

Cvičení

1. Určete v jakém pořadí budou navštěvovány uzly v následujícím grafu při průchodu do hloubky. Průchod začínám od uzlu, který je na grafu označen *u*. Aby průchod byl jednoznačný, tak v případě více možností navštivte u uzlů, jež jsou vedle sebe, nejdříve levý uzel a u uzlů nad sebou nejdříve horní uzel.



6.

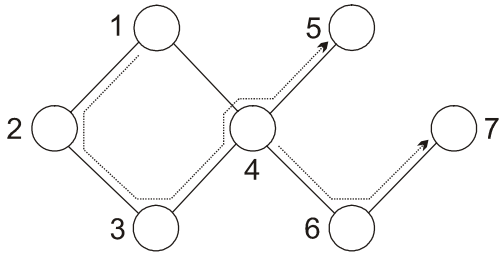
2. V jakém pořadí budou navštěvovány uzly v grafu z předchozího cvičení, jestliže půjde o průchod do šířky. Průchod začíná opět od uzlu *u* a jako první do fronty je vždy ukládán levý uzel anebo horní uzel.

Úkoly k textu

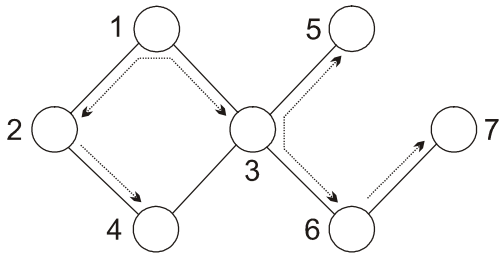
Dovedli byste najít graf (s aspoň pěti uzly) , ve kterém jsou uzly navštíveny při průchodu do hloubky i při průchodu do šířky ve stejném pořadí?

Řešení

1. Pořadí, v jakém jsou navštíveny uzly při průchodu do hloubky, ukazuje následující obrázek.



2. Pořadí navštívení uzlů při průchodu do šířky je na dalším obrázku.



3 Nezávislost, klikovost, dominance, barvení grafu

3.1 Nezávislost, klikovost, dominance

Studijní cíle: Seznámit s důležitými podmnožinami množiny uzlů grafu. Poskytnout algoritmy pro nalezení těchto podmnožin v grafu.

Klíčová slova: Nezávislá podmnožina, dominující podmnožina, klika.

Potřebný čas: 3 hodiny

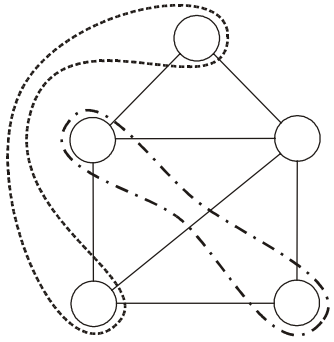
Tato část studijního textu se zabývá významnými podmnožinami množiny uzlů v grafu.

Definice. Podmnožinu N množiny uzlů grafu $G = (H, U, \rho)$ nazveme nezávislou, jestliže žádné dva uzly v N nejsou sousední.

V nezávislé podmnožině nejsou žádné dva uzly sousední.

Podmnožinu uzlů grafu nazveme maximální nezávislou podmnožinou, jestliže už k ní nelze přidat žádný uzel grafu, aniž by byla porušena vlastnost nezávislosti podmnožiny. Tedy není vlastní podmnožinou jiné nezávislé podmnožiny.

Příklad. Na následujícím grafu G jsou vyznačeny dvě jeho maximální nezávislé podmnožiny množiny uzlů.



Definice. Nezávislost grafu G je mohutnost (počet uzlů v množině) té největší nezávislé podmnožiny množiny uzlů grafu. Označujeme ji $\alpha(G)$. Tudíž

$$\alpha(G) = \max_{N \text{ je nezávislá podmnožina}} |N|$$

Příklad. V grafu z předchozího příkladu je zřejmě $\alpha(G) = 2$.

Určit nezávislost grafu je NP-obtížná úloha.

Určení nezávislosti grafu je úloha patřící do třídy NP-obtížných úloh. Není znám jiný postup jejího řešení než postupně procházet jednotlivé podmnožiny množiny uzlů grafu a hledat, která z nich je největší. Má-li graf m uzlů, pak je celkem 2^m podmnožin množiny uzlů grafu. Tedy počet podmnožin exponenciálně roste s počtem uzlů grafu. Projít tyto podmnožiny je zvládnutelné v přijatelném čase jen pro grafy s nízkým počtem uzlů (řádově do 30 uzlů). V praxi je ale nutné tuto a i další NP-obtížné úlohy řešit pro značně rozsáhlejší grafy. V tom případě se používají heuristické algoritmy. Jsou to algoritmy, které mají polynomiální časovou složitost, což znamená, že danou úlohu řeší v přijatelném (polynomiálním) čase. Jejich výsledek je ale běžně o něco horší, než by poskytl přesný algoritmus. Heuristický algoritmus zpravidla vychází z intuitivní myšlenky, jak danou úlohu řešit. Mnohdy máme pro stejnou úlohu více heuristický algoritmů lišících se vzájemně svou komplikovaností, časovou složitostí a přesností výsledků. Zpravidla algoritmy, které jsou komplikovanější a pracují déle, poskytují výsledky, jež se více přibližují k optimálnímu řešení než algoritmy jednoduché. O kolik horší

výsledek můžeme u jednotlivých heuristických algoritmů očekávat, se většinou nedá z nich odvodit, zjišťuje se to empirickými testy.

Heuristický algoritmus pro nalezení maximální nezávislé podmnožiny množiny uzlů grafu je založen na jednoduché myšlence, že velká nezávislá podmnožina uzlů se sestaví snadněji z uzlů, které mají v grafu málo sousedů, než z uzlů, které mají hodně sousedů. Protože zařazením uzlu do nezávislé podmnožiny se u všech jeho sousedů vyloučí možnost, že by mohli být v této množině. A čím má zařazovaný uzel více sousedů (větší stupeň), tím více uzlů ztrácí tak možnost být v nezávislé podmnožině. Následující algoritmus sestavuje maximální nezávislou podmnožinu postupným přidáváním uzlů k ní. V každém okamžiku z uzlů, které lze ještě uvažovat, vybere uzel s nejmenším stupněm, přidá ho k podmnožině a vyloučí tento uzel a všechny jeho sousedy dalšího uvažování.

Označení použitá v algoritmu:

G – výchozí graf.

G' - podgraf obsahující uzly, které lze ještě přidat k sestavované nezávislé podmnožině.

N – sestavovaná maximální nezávislá podmnožina uzlů.

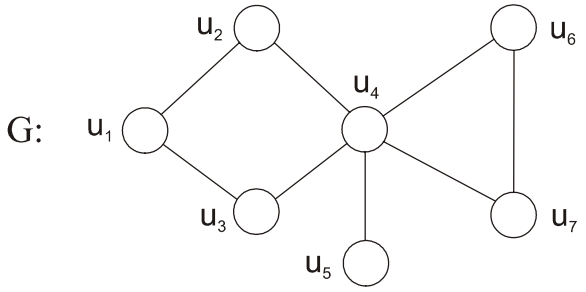
Popis algoritmus

1. Počáteční podmínky.
   - $N=\emptyset$ - sestavovaná nezávislá podmnožina je na začátku prázdná.
   - $G'=G$ - podgraf G' obsahující uzly, které lze ještě přidat k podmnožině, je na začátku celý výchozí graf .
2. Průběžný krok.

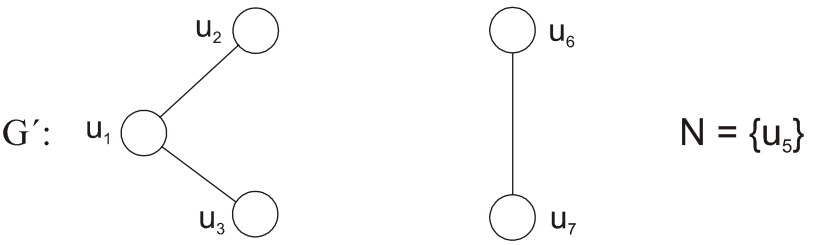
   V podgrafu G' najdeme uzel u s nejmenším stupněm. Je-li v něm více takových uzlů, vezmeme libovolný z nich. Uzel u přidáme k sestavované nezávislé podmnožině, $N = N \cup \{u\}$ a z podgrafu G' odstraníme tento uzel a rovněž všechny uzly, které s ním sousedí.

   Krok 2. provádíme tak dlouho, dokud podgraf G' není prázdný.

Příklad. Máme sestavit maximální nezávislou podmnožinu uzlů pro následující graf.

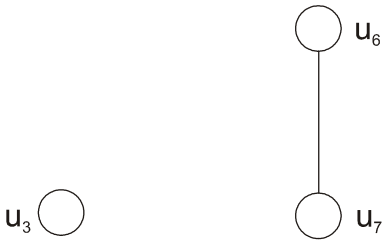


Nejmenší stupeň má uzel u_5. Dáme ho do množiny N a vytvoříme podgraf odstraněním uzlu u_5 a jeho souseda uzlu u_4.



Nejmenší stupeň mají nyní dva uzly u_2,u_3,u_6 a u_7. Pro přidání zvolíme uzel u_2.

G´: $N = \{u_2, u_5\}$



Nejmenší stupeň má nyní uzel u_3. Přidáme ho do množiny N.

G´: $N = \{u_2, u_3, u_5\}$

Nejmenší stupeň mají nyní dva uzly u_6 a u_7. Pro přidání zvolíme uzel u_6.

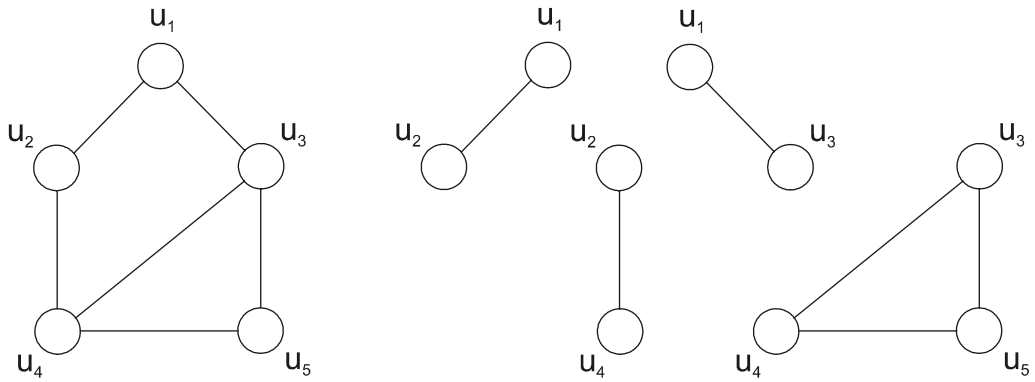
G´: je prázdný $N = \{u_2, u_3, u_5, u_6\}$

Podgraf $G´$ je prázdný, čímž je sestavení nezávislé podmnožiny množiny N dokončeno.

Definice. Klikou grafu $G = (H, U, \rho)$ nazveme každý jeho maximální úplný podgraf (připomeňme, že úplný podgraf je takový, že všechny uzly v něm jsou navzájem sousední).

Klika je maximální úplný podgraf.

Příklad. Na následujícím obrázku je graf G a 4 jeho různé kliky.



Definice. Klikovost grafu G je mohutnost množiny uzlů v největší klice grafu G. Označujeme ji $\omega(G)$.

Příklad. V grafu z předchozího příkladu je zřejmě $\omega(G) = 3$.

Klikovost grafu je opět NP-obtížná úloha a souvisí s nezávislostí grafu. Spojovacím článkem je zde doplněk grafu. Nejprve definice doplňku grafu.

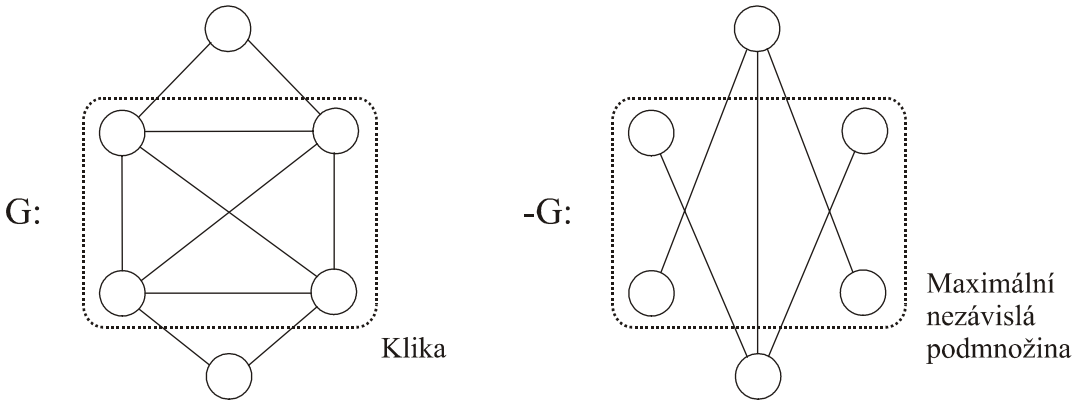
Stanovit klikovost grafu je rovněž NP-obtížná úloha.

Definice. Doplněk grafu G je graf, který má stejnou množinu uzlů a pro každé dva uzly v něm platí, že jsou sousední právě když tyto uzly nejsou sousední ve výchozím grafu G. Doplněk značíme $-G$.

Věta. Pro každý graf G graf platí $\omega(G) = \alpha(-G)$. (Klikovost grafu je rovna nezávislosti doplňku grafu.)

Důkaz: Uzly spolu s příslušnými hranami tvoří v grafu G úplný podgraf právě když v jeho doplňku $-G$ tvoří tyto uzly nezávislou podmnožinu množiny uzlů. Dále úplný podgraf grafu G je maximálním úplným podgrafem (klikou) právě když množina uzlů podgrafu je maximální nezávislou podmnožinou v doplňku $-G$.

Příklad. Uzly kliky vyznačené v následujícím grafu *G* jsou viditelně maximální nezávislou podmnožinou uzlů v jeho doplňku -*G*.

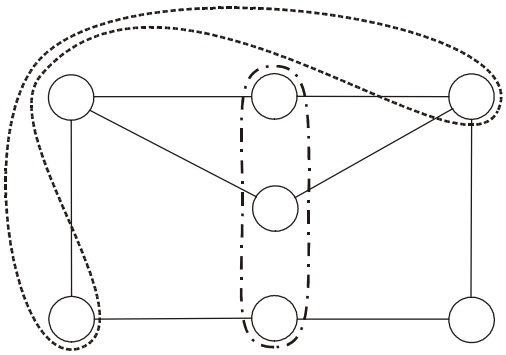


Z předchozí věty plyne, že algoritmy pro hledání maximálních nezávislých podmnožin uzlů lze využít pro hledání klik tak, že místo kliky v grafu hledáme maximální nezávislou podmnožinu v jeho doplňku.

Definice. Dominující podmnožina *D* množiny uzlů grafu G = (H, U, ρ) je taková podmnožina množiny uzlů grafu, která se svými sousedy pokrývá celý graf (zahrnuje celou množinu uzlů *U*).

Minimální dominující podmnožina množiny uzlů grafu je taková dominující podmnožina, jejíž žádná vlastní podmnožina není již dominující (nelze z ní odebrat žádný uzel bez ztráty vlastnosti dominující podmnožiny).

Příklad. Na následujícím grafu jsou vyznačeny dvě minimální dominující podmnožiny množiny uzlů grafu.



Definice. Dominance grafu *G* je mohutnost té nejmenší dominující podmnožiny množiny uzlů grafu. Označujeme ji $\beta(G)$. Tedy

$$\beta(G) = \min_{D \text{ je dominující podmnožina}} |D|$$

Věta. Nezávislá podmnožina množiny uzlů grafu je maximální právě když je dominující podmnožinou množiny uzlů grafu.

Důkaz: Nezávislá podmnožina je maximální právě když každý uzel, který v ní není, je sousední s některým uzlem v této množině. A to je právě vlastnost dominující podmnožiny.

Věta. Pro nezávislost α a dominanci β grafu *G* platí

$$\beta(G) \leq \alpha(G) .$$

Důkaz: Nechť *N* je libovolná maximální nezávislá podmnožina množiny uzlů grafu. Protože podle předchozí věty je zároveň dominující podmnožinou, dominance grafu je nejvýše rovna mohutnosti této množiny, tedy

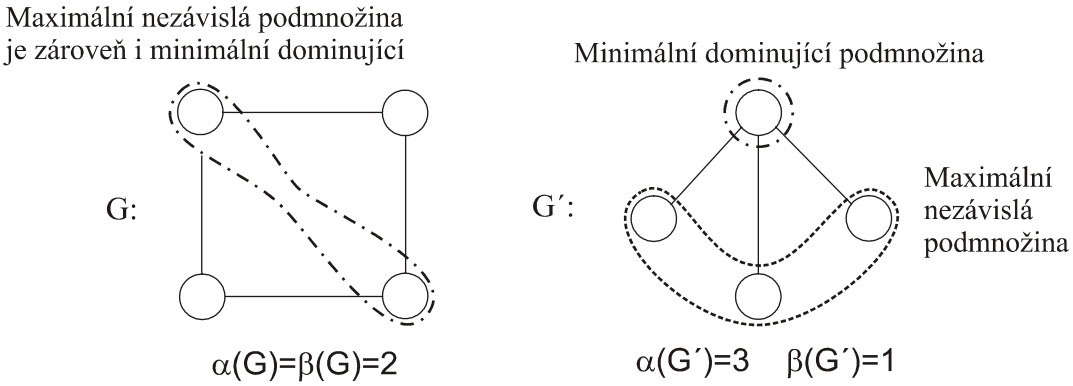
$$\beta(G) \leq |N| \ .$$

Dále protože N je nezávislá podmnožina množiny uzlů, nezávislost grafu není menší než mohutnost této množiny, tedy

$$|N| \leq \alpha(G) \ .$$

Spojením těchto dvou nerovností dostaneme tvrzení věty.

Příklad. Na následujícím obrázku jsou dva grafy. U jednoho je dominance stejná jako nezávislost, u druhého je dominance menší než nezávislost.



I stanovení dominance grafu je NP-obtížná úloha.

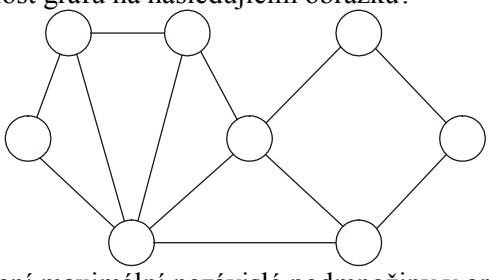
I dominance grafu patří do třídy NP-obtížných úloh. Vzhledem k tomu, že dominující podmnožina je zároveň maximální nezávislou podmnožinou, lze zde využít heuristického algoritmu pro hledání maximální nezávislé podmnožiny. Nyní ale hledáme maximální nezávislou podmnožinu s nejmenší mohutností, proto algoritmus upravíme tak, že v každém kroku budeme vybírat uzel s největším stupněm místo uzlu s nejmenším stupněm.

Kontrolní otázky

1. *Co je nezávislá podmnožina uzlů grafu a co je dominující podmnožina uzlů grafu?*
2. *Jak je definována nezávislost grafu a jak dominance grafu? Je mezi nimi nějaký vztah?*
3. *Co je klika grafu?*

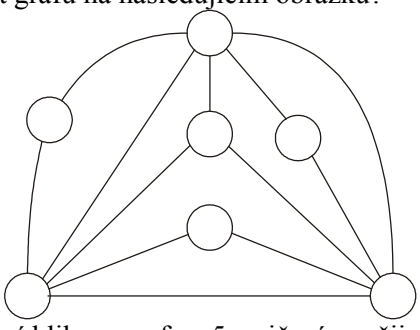
Cvičení

1. Jaká je nezávislost grafu na následujícím obrázku?



7.
2. Když pro nalezení maximální nezávislé podmnožiny v grafu z 1. cvičení použijeme heuristický algoritmus popsaný v této kapitole, dostaneme nezávislou podmnožinu, jejíž mohutnost (počet uzlů v ní) odpovídá nezávislosti grafu?
3. Jaká je dominance grafu, který je na obrázku v 1. cvičení?
4. Když pro nalezení dominující podmnožiny v grafu z 1. cvičení použijeme heuristický algoritmus naznačený v této kapitole, dostaneme dominující podmnožinu, jejíž mohutnost odpovídá dominanci grafu?

5. Jaká je klikovost grafu na následujícím obrázku?



8.

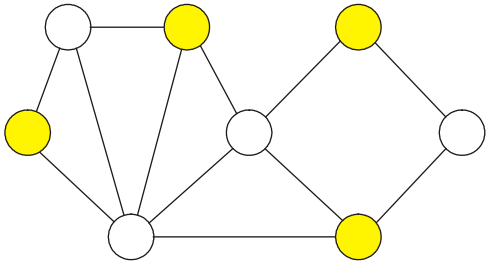
6. Když pro nalezení kliky v grafu z 5. cvičení použijeme heuristický algoritmus popsaný v této kapitole, dostaneme kliku, jejíž mohutnost odpovídá klikovosti grafu?

Úkoly k textu

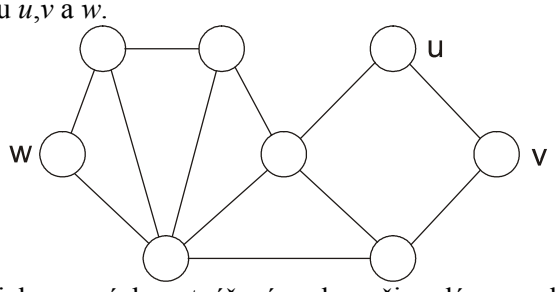
Uvažme, že byste sestavili graf, ve kterém uzly budou tvořit všechny osoby, které znáte. A každé dva uzly by byly sousední právě když dotyčné osoby by se vzájemně znaly. Našli byste v tomto grafu kliky? Které skupiny osob by je tvořily?

Řešení

1. Nezávislost daného grafu je 4. Uzly příslušné maximální nezávislé podmnožiny jsou barevně vyznačeny na následujícím obrázku.

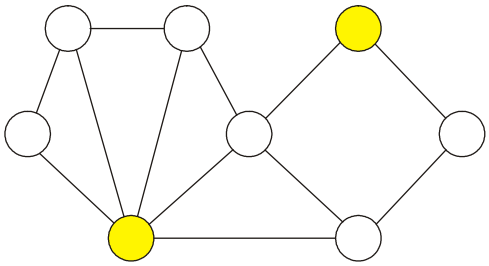


2. V tomto případě bude výsledek záviset na výběru uzlu s nejmenším stupněm, který zařadíme do vytvářené nezávislé podmnožiny. Na začátku to jsou 3 uzly označené na obrázku *u*,*v* a *w*.

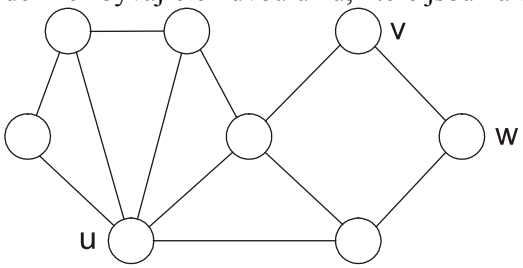


Pokud jako první do vytvářené podmnožiny dáme uzel *u*, dostaneme nezávislou podmnožinu se 4 uzly. Pokud začneme uzlem *v*, dostaneme nezávislou podmnožinu jen se 3 uzly. Začneme-li s uzlem *w*, bude to záležet na výběru dalších uzlů.

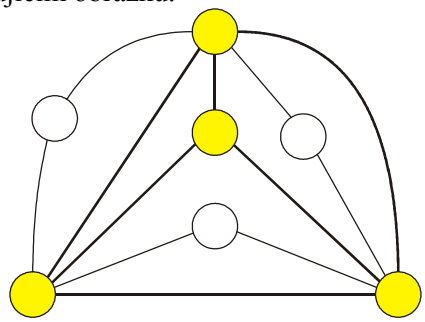
3. Dominance daného grafu je 2. Uzly příslušné dominující podmnožiny jsou barevně vyznačeny na následujícím obrázku.



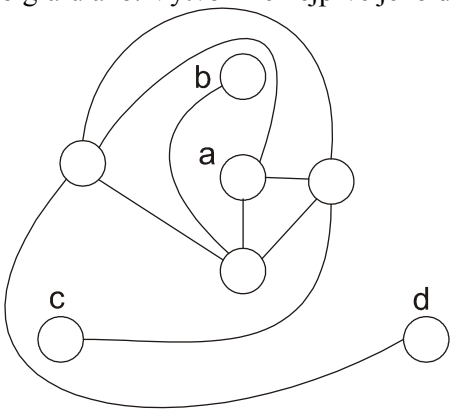
4. V tomto případě ano, neboť do vytvářené dominující podmnožiny jako první dáme uzel s největším stupněm, což je uzel označený na obrázku *u*. Následně pak do podmnožiny dáme jeden ze zbývajících dvou uzlů, které jsou na obrázku označené *v* a *w*.



5. Klikovost daného grafu je 4. Uzly příslušné kliky jsou barevně vyznačeny na následujícím obrázku.



6. U tohoto grafu ano. Vytvoříme nejprve jeho doplněk.



Uzly největší kliky obsažené v původním grafu jsou v jeho doplňku označeny písmeny *a*, *b*, *c*, *d*. Protože nyní heuristickým algoritmem v doplňku hledáme maximální nezávislou podmnožinu, vybereme do ní uzly s nejmenším stupněm. To jsou uzly *b*, *c*, *d*. Po jejich odstranění a odstranění jejich sousedů z grafu zůstane jen uzel a.

Ten jako poslední přidáme k nezávislé podmnožině. Nezávislá podmnožina obsahuje nyní uzly *a*, *b*, *c*, *d*, což jsou právě uzly největší kliky v původním grafu.

3.2 Barvení grafu

Studijní cíle: Poskytnout algoritmy pro barvení grafu.

Klíčová slova: Obarvení grafu, chromatické číslo.

Potřebný čas: 4 hodiny

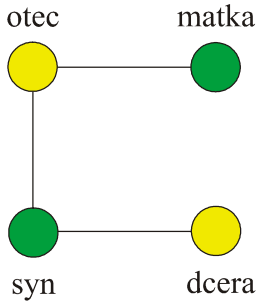
Tato část studijního textu se zabývá významnou, v praxi se často vyskytujícího úlohou, kterou je obarvení grafu. Tato grafová úloha se vyskytuje v souvislosti s přidělováním zdrojů. V ní máme objekty, které všechny potřebují jistý zdroj. Přitom charakter této úlohy je takový, že některé z nich mohou mít zdroj společný, zatímco jiné nemohou sdílet stejný zdroj. A úlohou je najít co nejmenší počet zdrojů vyhovující zadání úlohy.

Budeme-li tuto úlohu řešit grafem, objekty budou uzly grafu a vždy dva objekty, které nemohou mít stejný zdroj, budou v grafu sousední.

Příklad. Máme rodinu, jejíž příslušníci potřebují v určitých hodinách dne auto následovně.

- Otec autem odjíždí v 7 hodin autem do zaměstnání a vrací se domů v 16 hodin.
- Matka je doma a auto potřebuje dopoledne od 9 do 12 hodin na nákupy.
- Syn auto potřebuje odpoledne od 15 hodin, kdy jím odjíždí do fotbalového klubu .
- Dcera auto potřebuje od 18 hodin, kdy jede za svým přítelem.

Sestavíme graf, ve kterém dva uzly budou sousední, když dva rodinní nemohou používat stejné auto, protože doby, kdy ho potřebují, se překrývají. A uzly v grafu obarvíme tak, aby žádné dva sousední uzly neměly stejnou barvu a přitom jsme použili nejmenší možný počet barev.



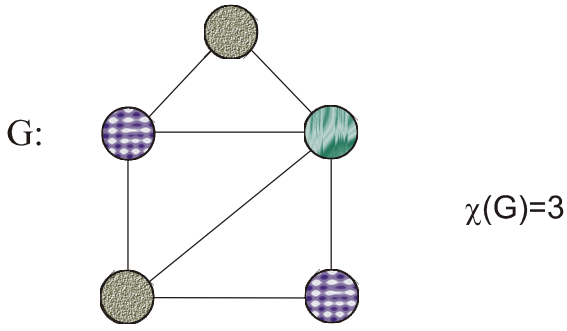
K obarvení grafu potřebujeme dvě barvy. Řešením úlohy tedy je, že rodina potřebuje dvě auta.

Definice. Obarvením grafu označujeme takové obarvení uzlů grafu, ve kterém žádné dva sousední uzly nemají stejnou barvu.

Při barvení grafu se sousední uzly barví různou barvou.

Definice. Chromatické číslo grafu *G* je nejmenší počet barev, kterým lze obarvit uzly grafu. Chromatické číslo značíme $\chi(G)$.

Příklad. Na následujícím obrázku je obarvení grafu.



Věta. Pro nezávislost α a chromatické číslo χ grafu G = (H, U, ρ) platí

$$\alpha(G)\,\chi(G) \geq |U| .$$

Důkaz: Označme U_1, U_2, …, $U_{\chi(G)}$ množiny uzlů obarvené stejnou barvou (počet těchto množin udává chromatické číslo $\chi(G)$). Uzly v každé z těchto množin nejsou sousední, tudíž každá z těchto množin je zároveň nezávislou podmnožinou množiny uzlů. Odtud plyne, že mohutnost žádné z těchto množin není větší než nezávislost grafu. Tedy

$$|U_1| \leq \alpha(G)$$

$$|U_2| \leq \alpha(G)$$

……

$$|U_{\chi(G)}| \leq \alpha(G)$$

Sečtením všech nerovností

$$|U_1| + |U_2| + \ldots + |U_{\chi(G)}| \leq \chi(G)\alpha(G) \ .$$

A protože množiny U_1, U_2, …, $U_{\chi(G)}$ jsou vzájemně disjunktní, platí

$$|U_1| + |U_2| + \ldots + |U_{\chi(G)}| = |U| \ ,$$

čímž dostáváme tvrzení věty

$$|U| \leq \chi(G)\alpha(G) \ .$$

Věta. Je-li k maximum ze stupňů všech uzlů grafu G, pak pro chromatické číslo grafu G platí

$$\chi(G) \leq k+1 \ .$$

Důkaz: Graf s uzly stupně nejvýše k lze vždy obarvit k+1 barvami, neboť každý uzel má nejvýše k sousedů a tito jsou obarveni nejvýše k barvami, čímž na obarvení uzlu vždy zůstává aspoň jedna barva, kterou není obarven žádný jeho soused. Tudíž chromatické číslo grafu nemůže být větší než k+1.

Stanovit chromatické číslo grafu je také NP-obtížná úloha.

Barvení grafu minimálním počtem barev je opět úloha patřící do třídy NP-obtížných úloh. Uvedeme 3 heuristické algoritmy barvení a pak ještě další algoritmus, kterým se výsledek obarvení heuristickými algoritmy dá v jistých případech zlepšit.

Označení použitá v algoritmech:

Jednotlivé barvy, které použijeme k barvení grafu, budeme značit přirozenými čísly (1,2,…).

Proměnná *B* bude označovat nejvyšší číslo použité barvy a zároveň i počet použitých barev, protože při barvení budeme používat čísla barev, jak jdou za sebou.

Pro barvení grafu existuje řada heuristických postupů.

Číslo barvy, kterou je obarven uzel *u*, budeme označovat funkcí *b*, tj. zápisem *b(u)*. Tento zápis použijeme i pro obarvení uzlu. Skutečnost, že uzel *u* obarvíme barvou *c*, zapíšeme přiřazením

b(u) = c.

Popis algoritmu

1. Počáteční podmínky.
   - B=0 – hodnotu proměnné, v níž máme číslo maximální použité barvy, na začátku nastavíme na nulu.
2. Průběžný krok.
   a) V grafu vyhledáme doposud neobarvené uzly. Je-li jich více, uplatníme následně další výběrové kritérium b):

b) Mezi doposud neobarvenými uzly vyhledáme uzel, jehož již obarvení sousedé jsou obarveni největším počtem různých barev (nikoliv uzel, který má nejvíce již obarvených sousedů). Zřejmě na obarvení takového uzlu v současnosti zůstává nejmenší počet barev. Je-li takových uzlů více, uplatníme ještě další výběrové kritérium c):

c) Mezi uzly vybranými v předchozím kritériu b) najdeme uzel, který má největší počet doposud neobarvených sousedů. Zřejmě pokud by byly následně obarveny různými barvami, bylo by obtížné najít volnou barvu pro obarvení tohoto uzlu. Proto tento uzel obarvíme přednostně - ještě předtím, než dojde k barvení jeho sousedů. Je-li i zde více uzlů splňující toto kritérium, pak pro obarvení vezmeme libovolný z nich.

Uzel, který byl předchozími kritérii vybrán jako následující uzel pro obarvení, označme u. Najdeme nejnižší barvu c takovou, že jí není obarven žádný z již obarvených sousedů uzlu u. Uzel u touto barvou obarvíme

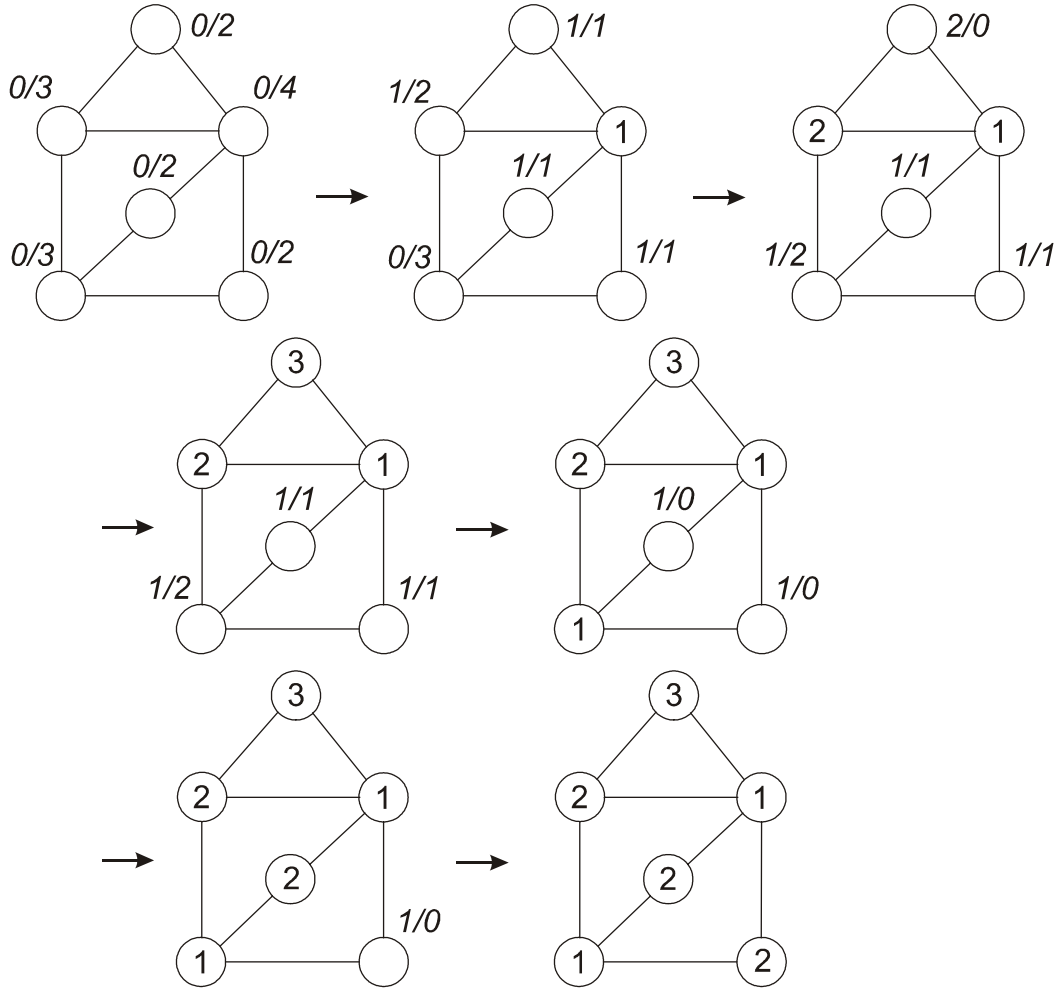
$$b(u) = c \quad .$$

A je-li $c>B$ (c je nová, doposud nepoužitá barva), nastavíme proměnnou B na číslo této barvy

$$B = c \quad .$$

3. Krok 2. opakujeme tak dlouho, dokud nejsou obarveny všechny uzly grafu. Na závěr je v proměnné B počet barev, jimiž je graf obarven.

Příklad. Na následujícím grafu je vyznačen postup barvení stejného grafu, jaký byl v předchozím příkladu. U neobarvených uzlů je dvojice čísel, z nichž první je počet barev, jimiž jsou obarveni sousedé uzlu, a druhé je počet doposud neobarvených sousedů uzlu.



Účinnost předchozího algoritmu heuristického barvení grafu je poměrně dobrá. Typicky počet použitých barev je o 5% větší než je chromatické číslo grafu.

Další heuristický algoritmus barvení je založen na skutečnosti, že množina uzlů, které jsou obarvené stejnou barvou, tvoří nezávislou podmnožinu množiny uzlů grafu. Algoritmus využívá již dříve uvedený heuristický algoritmus hledání maximálních nezávislých podmnožin.

Označení použitá v algoritmu:

b – proměnná označující aktuální barvu.

G´ - podgraf obsahující doposud neobarvené uzly.

Popis algoritmu

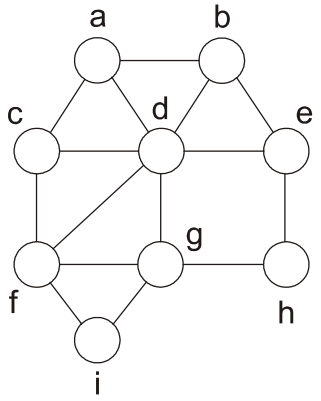
1. Počáteční podmínky.
   d) b=1 – aktuální barvu nastavíme na první barvu.
   e) G´=G – podgraf G´ obsahující doposud neobarvené uzly na začátku obsahuje celý výchozí graf.
2. Průběžný krok.

   V podgrafu *G´* již popsaným algoritmem najdeme maximální nezávislou podmnožinu uzlů *N,* obarvíme ji aktuální barvou *b* a uzly podmnožiny *N* z podgrafu *G´* odstraníme.
3. Pokud podgraf *G´* není ještě prázdný, zvýšíme číslo aktuální barvy

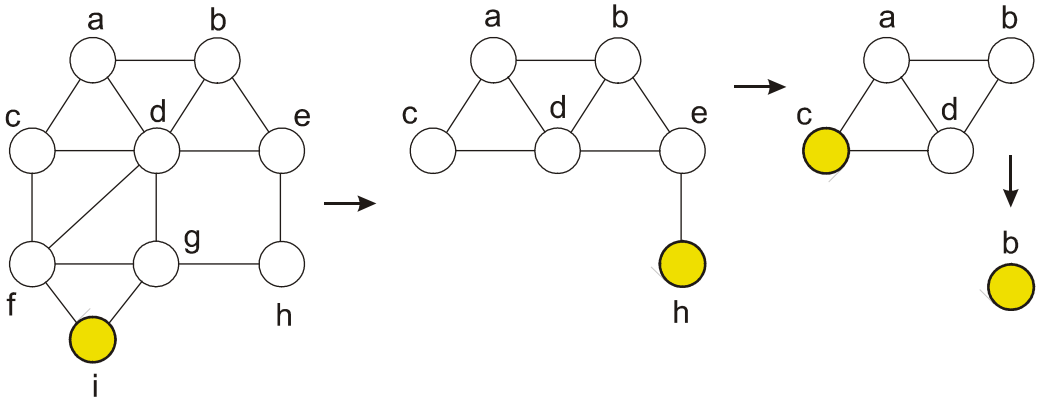
   $$b = b + 1$$

   a přejdeme znovu ke kroku 2. – barvení další barvou.

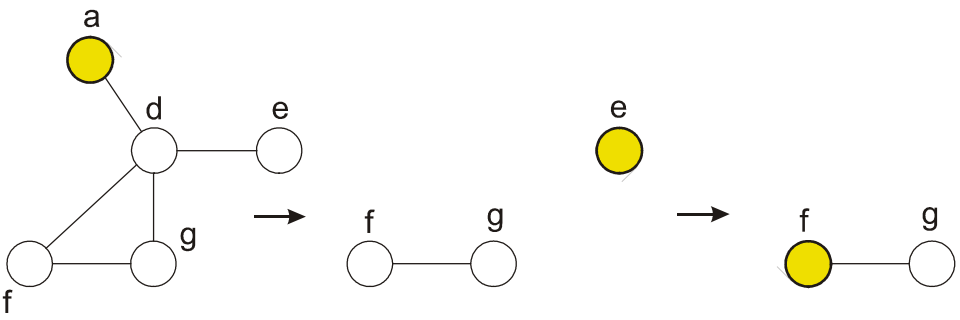
Příklad. Metodou hledání maximálních nezávislých množin máme obarvit následující graf.



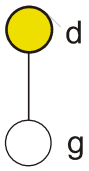
Postup hledání první nezávislé podmnožiny ukazuje následující obrázek.



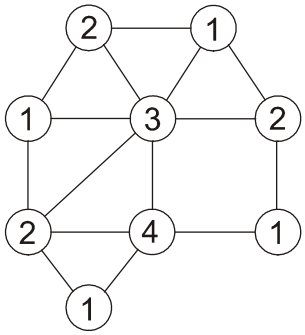
Ve zbylém podgrafu hledáme druhou nezávislou podmnožinu.



A zůstává podgraf se dvěma uzly, v němž hledáme třetí nezávislou podmnožinu.



A na konci zůstává podgraf obsahující jen jeden uzel - ten bude tvořit poslední, čtvrtou nezávislou podmnožinu množiny uzlů. Výsledné obarvení ukazuje následující obrázek.



Další heuristický algoritmus provádí spojování uzlů, které lze obarvit stejnou barvou, tedy uzlů, které jsou nesousední. Výsledek je úplný graf a ten lze snadno obarvit.

Popis algoritmu

1. První část – vytvoření pomocného úplného grafu.

   V grafu najdeme libovolné dva nesousední uzly *u* a *v*. Provedeme v grafu spojení těchto uzlů tak, že je nahradíme jedním uzlem, označme tento uzel *uv*. Nový uzel spojíme hranami se všemi původními sousedy uzlu *u* a rovněž se všemi původními sousedy uzlu *v*.

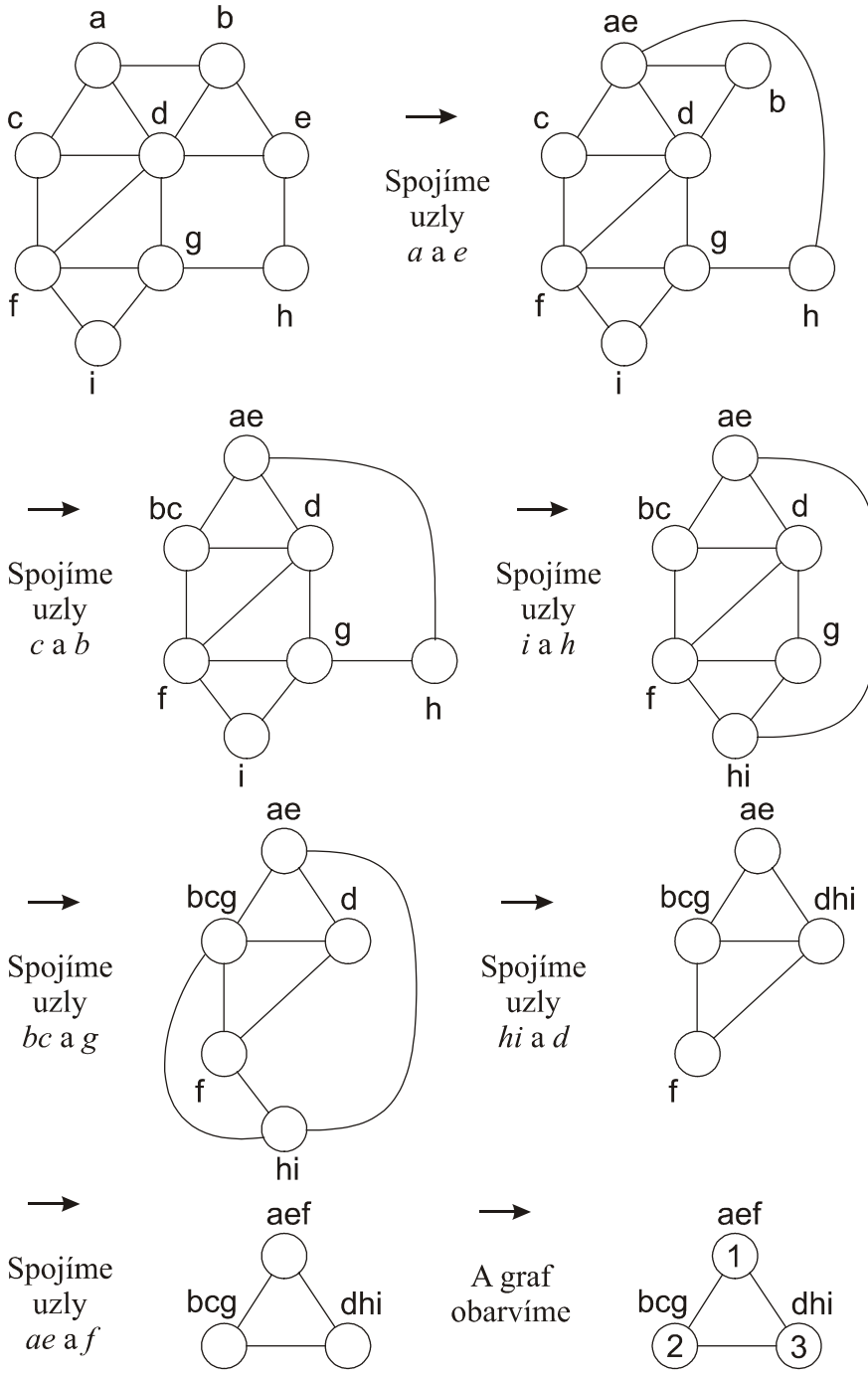
   Tento krok opakujeme tak dlouho, dokud lze spojovat nějaké dvojice uzlů. Po ukončení spojování dostaneme úplný graf.

2. Druhá část – vlastní obarvení.

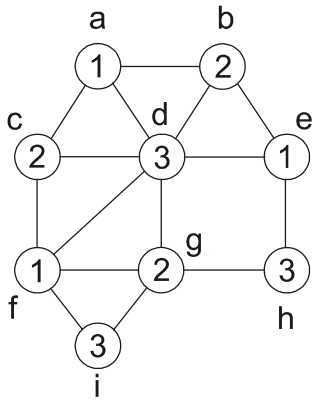
   Vytvořený úplný graf obarvíme. Úplný graf nelze obarvit jinak, než že každému jeho uzlu přiřadíme samostatnou barvu. Následně zpětně spojené uzly začneme rozdělovat, přičemž vždy stejnou barvu, jakou má uzel *uv*, dáme rovněž uzlům *u* a *v*, jejichž spojením uzel *uv* vznikl.

   Proces rozpojování uzlů opakujeme tak dlouho, dokud se nedostaneme k původnímu grafu.

Příklad. Vezmeme stejný graf, jaký byl v předchozím příkladu, a obarvíme ho metodou barvení spojováním uzlů.



Nyní obarvíme v původním grafu barvou 1 uzly *a*, *e* a *f*, barvou 2 uzly *b*, *c* a *g* a barvou 3 uzly *d*, *h* a *i*.



Poslední algoritmus, který uvedeme, není samostatným algoritmem barvení, ale dá se použít pro zlepšení výsledku jiných heuristických algoritmů, pokud je v nich ve vhodných okamžicích použit. Zejména je to situace, kdy musíme obarvit jediný uzel samostatnou barvou.

Algoritmus výměny barev sám neobarvuje uzly. Použije se jiným algoritmem v okamžiku, kdy použité barvy už jsou vyčerpány a nelze s nimi už obarvit další uzel. Algoritmus výměny barev zkouší, zda nelze v již obarvených uzlech zaměnit jejich obarvení tak, že by po záměně bylo možné některý zatím neobarvený uzel obarvit již použitou barvou. Typicky se použije, když už se barvení chýlí ke konci a zbývá už jeden nebo jen několik obarvených uzlů, které nelze obarvit, aniž bychom museli přidat další barvu. Každopádně, aby nějaká výměna mohla nastat, musí být v doposud obarvené části grafu nejméně dvě barvy.

Algoritmus lze rovněž použít i na zlepšení obarvení v již obarvených graf. Zde vybereme nějakou barvu, kterou je obarven jeden nebo jen několik uzlů a zkoušíme se ji odstranit tak, že výměnou barev postupně hledáme, zda tyto uzly by nebylo možné obarvit některou jinou barvou.

Popis algoritmu

1. Počáteční stav.
   - V s již obarvené části grafu G jsou použity už nejméně dvě barvy, tedy pro počet nejvyšší použitou barvu B platí

     $$B \geq 2 \quad .$$

   - V grafu je neobarvený uzel u, který nelze obarvit žádnou již použitou barvou, nebo máme uzel, jehož barvu b chceme odstranit tak, že ho obarvíme jinou již použitou barvou.
2. Zvolíme dvě různé barvy c, d, které byly již použity ve stávajícím obarvení. Nechť barva c je v této dvojici tou barvou, která má nižší číslo. To znamená, že čísla barev c a d vyhovují vztahu

   $$1 \leq c < d \leq B \quad .$$

   V případě, kdy odstraňujeme existující barvu b, volíme barvy c a d navíc tak, aby byly různé od této barvy

   $$c \neq b, \quad d \neq b \quad .$$

   V barveném grafu najdeme jeho podgraf, ve kterém jsou jen ty uzly, které jsou obarvené barvou c nebo d. Typicky tento podgraf je složen z více komponent. Ty rozdělíme do čtyř tříd.

   Třída A - bude obsahovat komponenty, jejichž žádný uzel ve výchozím grafu G nesousedí s uzlem u, který chceme obarvit nebo jeho barvu změnit.

   Třída B - bude obsahovat komponenty, které mají uzly, jež sousedí ve výchozím grafu G s uzlem u, který chceme obarvit nebo jeho barvu změnit, přičemž tyto uzly jsou všechny obarveny jen barvou c.

   Třída C - bude obsahovat komponenty, které mají uzly, jež sousedí ve výchozím grafu G s uzlem u, který chceme obarvit nebo jeho barvu změnit, přičemž tyto uzly jsou všechny obarveny jen barvou d.

   Třída D - bude obsahovat komponenty, které mají uzly, jež sousedí ve výchozím grafu G s uzlem u, který chceme obarvit nebo jeho barvu změnit, přičemž některé uzly jsou obarveny barvou c a jiné barvou d.

   Je-li třída D neprázdná, nelze u zvolené dvojice barev provést výměnu, která by vedla k požadovanému obarvení uzlu u. V tomto případě pokud je ještě jiná dvojice barev vyhovující vztahu $1 \leq c < d \leq B$ (a případně $c \neq b$, $d \neq b$), kterou jsme doposud

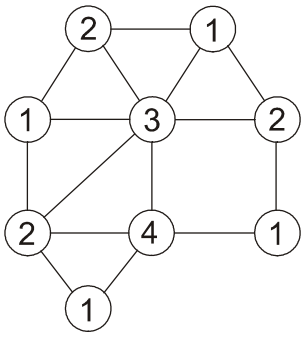
nevyzkoušeli, můžeme algoritmus zkusit znovu s ní, jinak algoritmus neúspěšně končí – nelze s ním obarvit uzel *u* požadovaným způsobem.

Je-li třída *D* prázdná (neobsahuje žádnou komponentu), pak:

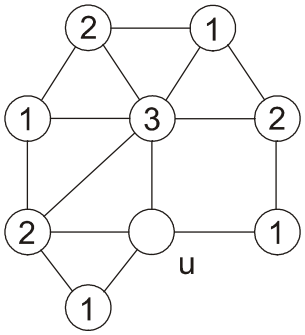
a) Buďto ve všech komponentách třídy *B* zaměníme barvy *c* a *d*, tj. uzly předtím obarvené barvou *c* budou nyní obarveny barvou *d* a naopak uzly doposud obarvené barvou *d* budou nyní obarvené barvou *c*. Po této výměně už žádný uzel sousedící s uzlem *u* nebude obarven barvou *c*, čímž uzel *u* nyní můžeme obarvit touto barvou.

b) Anebo ve všech komponentách třídy *C* zaměníme barvy *c* a *d*, tj. uzly předtím obarvené barvou *c* budou nyní obarveny barvou *d* a naopak uzly doposud obarvené barvou *d* budou nyní obarvené barvou *c*. Tím uzel *u* nyní můžeme obarvit barvou *d*, neboť žádný jeho soused už není obarven touto barvou.

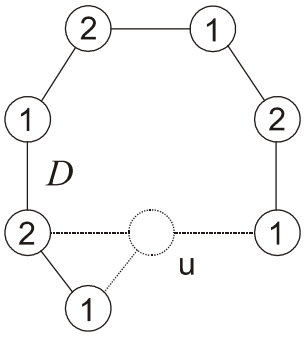
Příklad. Vraťme se k příkladu, kdy jsme graf barvili hledáním maximálních nezávislých podmnožin. Výsledek ukazuje následující obrázek.



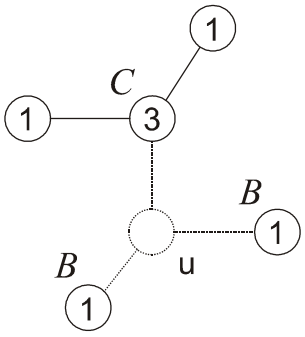
Odstraňme obarvení uzlu, jež je obarven barvou 4 a který jediný je touto barvou obarven, a zkusme ho obarvit jinou barvou metodou výměny barev.



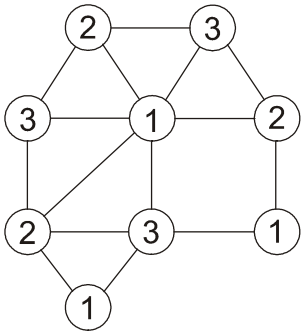
Zvolíme první možnou dvojici barev *c*=1 a *d*=2. Odstraníme z grafu uzly obarvené jinými barvami.



U této dvojice barev třída *D* není prázdná – obsahuje jednu komponentu. Zkusíme jinou možnou dvojici barev *c*=1 a *d*=3.



Zde už je třída *D* prázdná. Udělám výměnu barev třeba ve třídě *C* – ta obsahuje jednu komponentu. A uzel *u* pak obarvíme barvou 3. Výsledkem je graf obarvený třemi barvami.



Průvodce studiem

Jedno z neznámějších a velmi užitečných využití barvení grafu je v konstrukci překladačů. Metodou barvení grafu se zde v přeloženém programu řeší efektivní využití registrů procesoru při výpočtu.

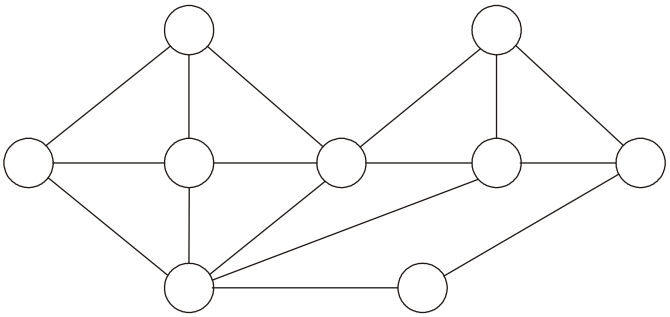
Kontrolní otázky

Kontrolní otázky

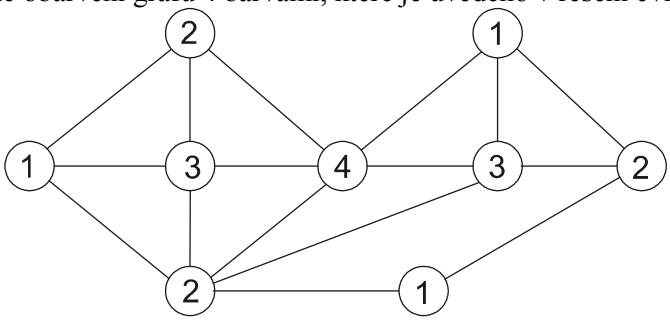
1. *Co je chromatické číslo grafu?*
2. *Jaké je chromatické číslo úplného grafu?*
3. *Proč pro barvení grafu používáme heuristické algoritmy a ne algoritmy, které jsou přesné a dají optimální výsledek?*

Cvičení

1. Jaké je chromatické číslo grafu na následujícím obrázku?



2. Pro obarvení grafu ze 7. cvičení použijeme první z heuristických algoritmů pro barvení, jež jsou popsány v této kapitole. Obarvíme ho stejným počtem, jako je chromatické číslo tohoto grafu?
3. Nyní pro obarvení grafu ze 7. cvičení použijte algoritmus hledání maximálních nezávislých podmnožin.
4. Jak dopadne obarvení grafu, když použijeme heuristický algoritmus založený na slepování uzlů?
5. Vezměte obarvení grafu 4 barvami, které je uvedeno v řešení cvičení 9,



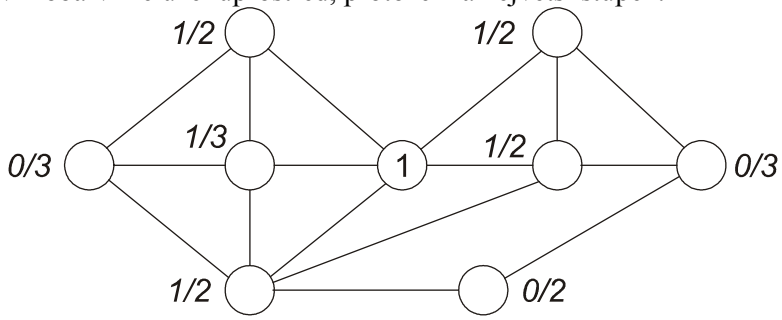
a zkuste, zda ho lze zlepšit algoritmem výměny barev.

Úkoly k textu

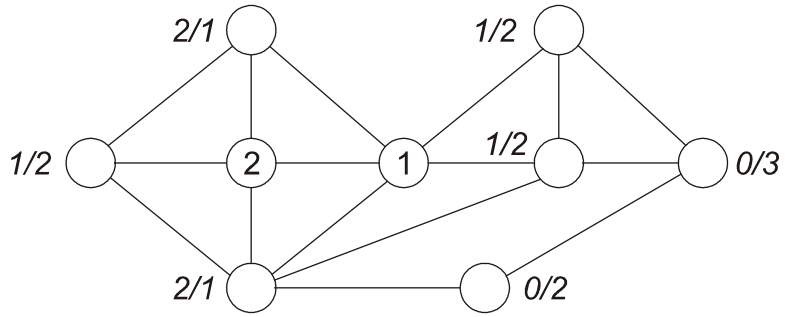
Máme graf s *m* uzly, o kterém víme, že jeho chromatické číslo je 2. Kolik může mít takový graf nejvýše hran?

Řešení

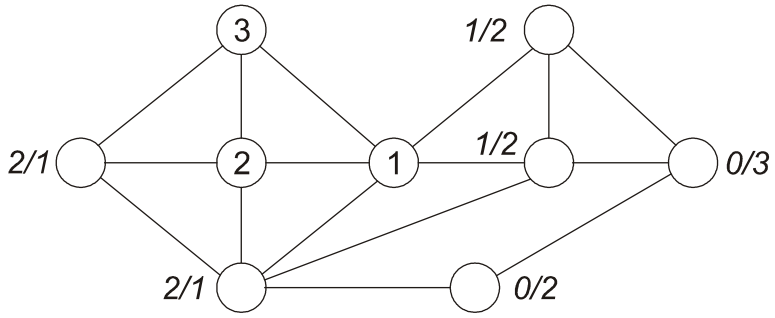
1. Daný graf lze obarvit 3 barvami. Chromatické číslo grafu je 3.
2. Jako první obarvíme uzel uprostřed, protože má největší stupeň.



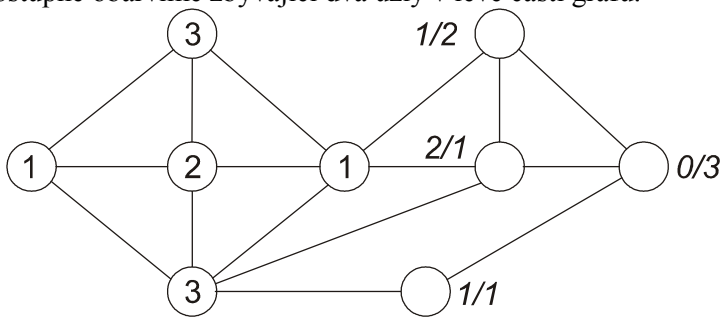
Nyní zřejmě obarvíme uzel vlevo od obarveného uzlu.



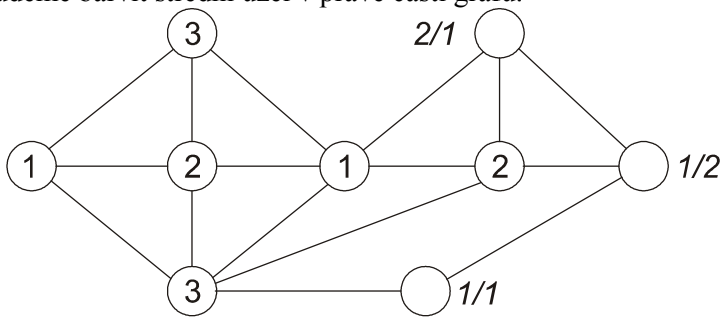
Ze dvou uzlů, které přicházejí nyní v úvahu k obarvení, obarvíme třeba ten, který je více nahoře.



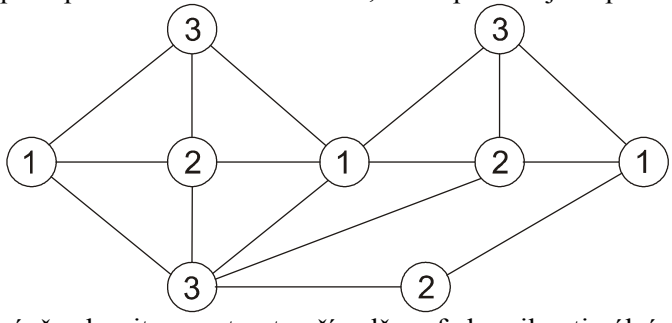
Nyní postupně obarvíme zbývající dva uzly v levé části grafu.



Nyní budeme barvit střední uzel v pravé části grafu.

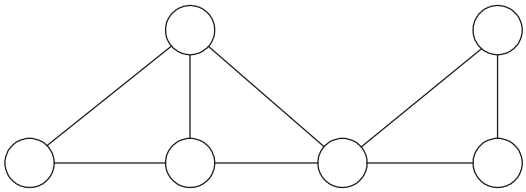


A nyní postupně obarvíme uzel nahoře, uzel vpravo a jako poslední obarvíme uzel dole.

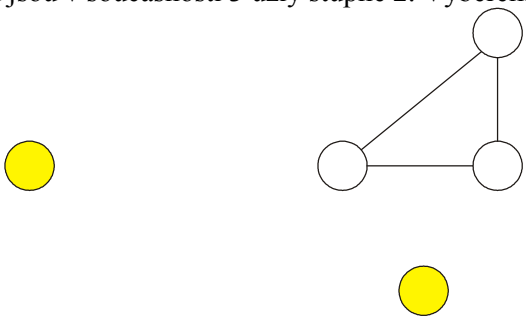


Je zřejmé, že algoritmus v tomto případě graf obarvil optimálním způsobem. Počet barev je stejný jako chromatické číslo grafu.

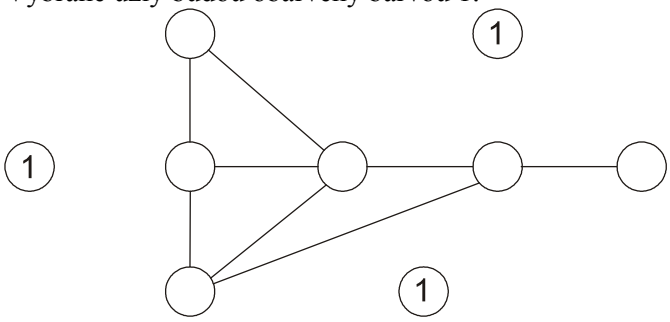
3. Nejnižší stupeň má uzel vpravo dole. Po jeho vybrání a odstranění jeho sousedů bude graf vypadat.



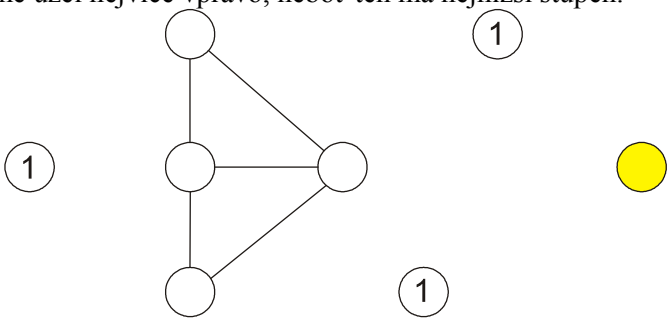
V grafu jsou v současnosti 3 uzly stupně 2. Vybereme třeba ten vlevo.



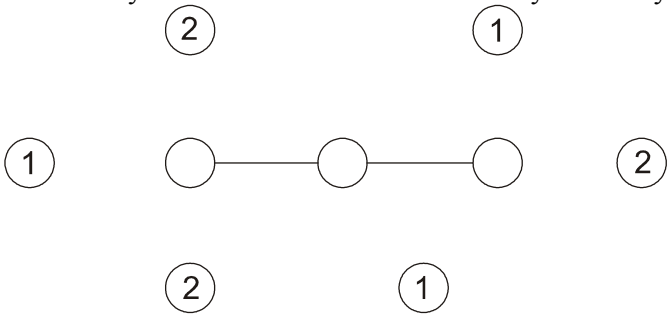
A nyní můžeme vybrat kterýkoliv ze zbývajících uzlů. Zkusme vybrat uzel, který je nahoře. Vybrané uzly budou obarveny barvou 1.



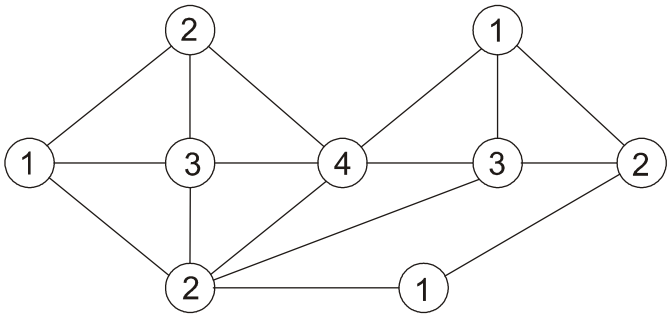
A můžeme pokračovat v hledání další maximální nezávislé podmnožiny. Zřejmě nyní vybereme uzel nejvíce vpravo, neboť ten má nejnižší stupeň.



A následně bude vybrán uzel nahoře a uzel dole. A vybrané uzly obarvíme barvou 2.

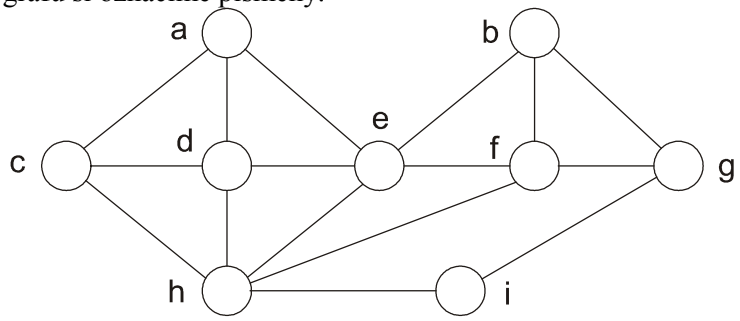


Je zřejmé, že nyní vybereme ještě další dvě nezávislé maximální podmnožiny. Výsledné obarvení grafu bude.

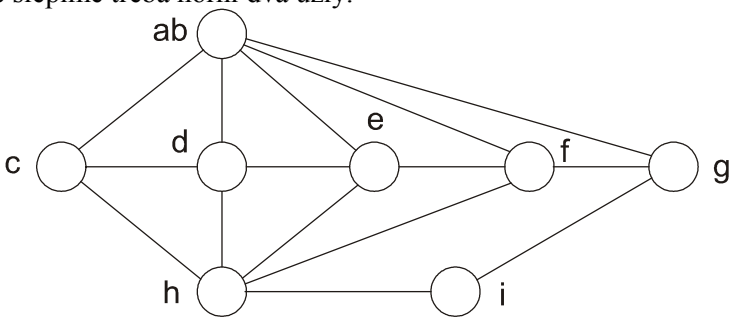


Graf je obarven větším počtem barev, než je optimální. Algoritmus v tomto případě dal horší výsledek. Ten ale také závisí na výběru uzlů do maximálních nezávislých podmnožin v situacích, kdy jsme mohli zvolit některý uzel z více uzlů. Pokud jste volili jiné uzly, mohli jste dostat jiný výsledek.

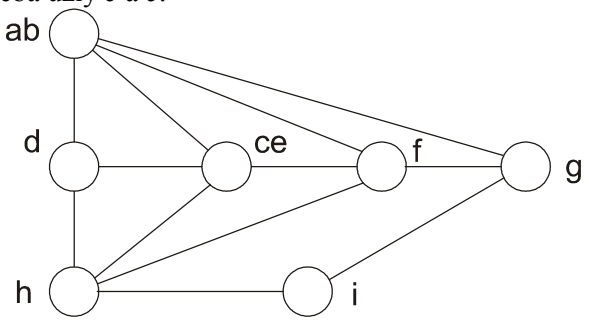
4. Uzly v grafu si označíme písmeny.



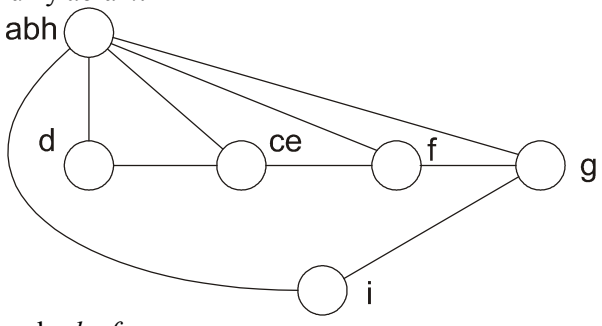
Nejprve slepíme třeba horní dva uzly.



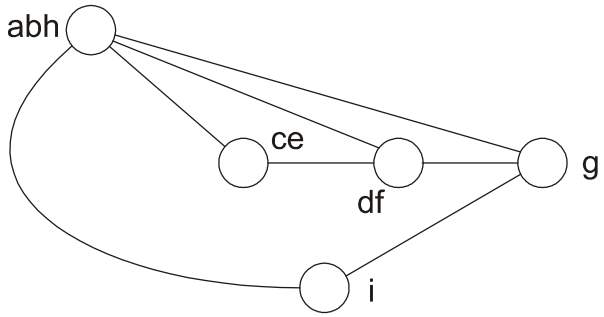
Nyní slepíme třeba uzly *c* a *e*.



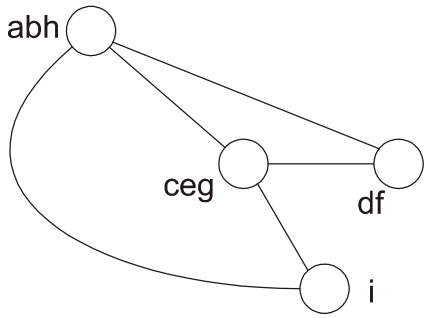
A třeba slepíme uzly *ab* a *h*.



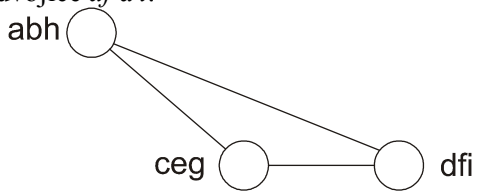
A třeba slepíme uzly *d* a *f*.



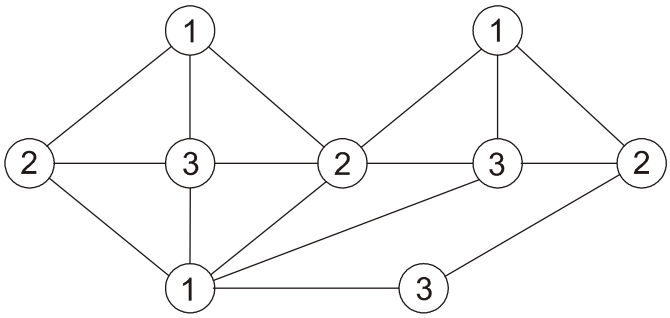
A uzly *ce* a *g*.



Zbývá dvojice *df* a *i*.

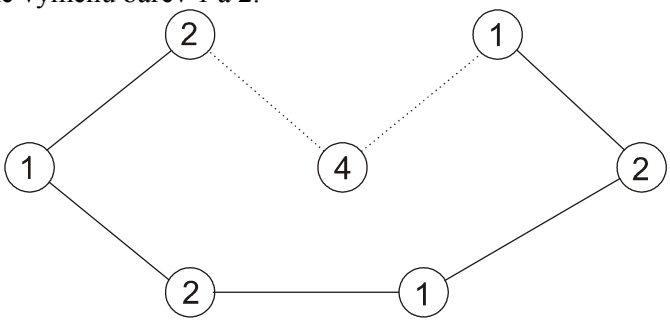


Nyní v původním grafu obarvíme uzly *a*,*b*,*h* barvou 1, uzly *c*,*e*,*g* barvou 2 a uzly *d*,*f*,*i* barvou 3.

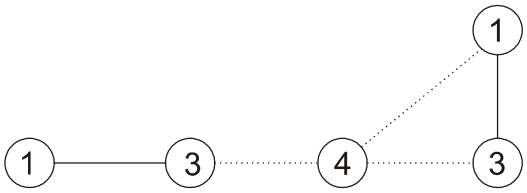


Až na čísla barev jsme dostali stejné obarvení, jaké bylo ve cvičení 8.

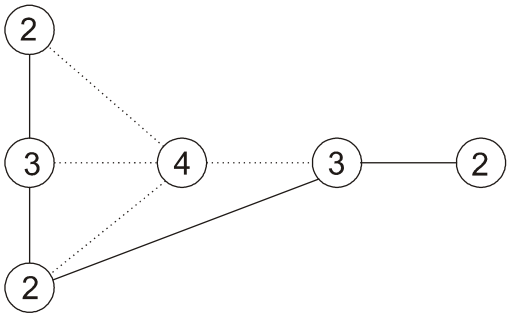
5. Budeme se snažit výměnou barev snížit barvu 4 ve středním uzlu na nižší barvu. Zkusíme výměnu barev 1 a 2.



Zde je jedna komponenta, jejíž uzly obou barev sousedí s uzlem s barvou 4. Tato možnost nevede k cíli. Zkusíme výměnu barev 1 a 3.



I zde je komponenta, jejíž obou barev sousedí s uzlem s barvou 4. Zbývá ještě možnost výměny barev 2 a 3.



Ani tato možnost nevede k cíli. Algoritmus v tomto případě neuspěl.

4 Minimální kostry grafu

Studijní cíle: Naučit algoritmy pro nalezení minimální kostry v grafu.

Klíčová slova: Kostra grafu.

Potřebný čas: 1 a půl hodiny

Kostra je faktor grafu, který

- má stejný počet komponent
- neobsahuje kružnice.

Připomeňme, že faktor grafu je jeho podgraf, který má stejnou množinu uzlů.

U hranově ohodnocených grafů se v praxi vyskytuje úloha nalezení minimální kostry. Formulace této úlohy je

Máme dán graf G = (H, U, ρ), jehož hrany jsou ohodnoceny kladnými reálnými čísly. Tedy existuje zobrazení *w*

$$w\colon\ H \to R^{+} \quad ,$$

kde R^{+} označuje množinu kladných reálných čísel.

Máme najít takovou kostru K = (H´, U, ρ´) grafu *G*, jejíž součet ohodnocení hran

$$s = \sum_{h \in H'} w(h)$$

je minimální ze všech možných koster grafu *G*.

Existují dva algoritmy. Jeden hrany přidává, nazveme ho zařazovací, a druhý hrany ubírá, nazveme ho vyřazovací.

4.1 Zařazovací algoritmus

Hrany grafu *G* uspořádáme tak, že jejich ohodnocení tvoří neklesající posloupnost.

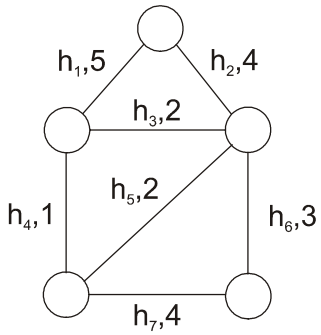
$$h_{i1}, h_{i2}, ..., h_{in}$$

$$w(h_{i1}) \le w(h_{i2}) \le ... \le w(h_{in}) \quad .$$

Algoritmus pro nalezení minimální kostry je poměrně jednoduchý.

Vezmeme diskrétní podgraf *K* výchozího grafu *G* (podgraf obsahující jen uzly grafu *G*). Postupně procházíme posloupnost hran od začátku směrem ke konci a u každé hrany ověřujeme, zda jejím přidáním k podgrafu *K* v něm nevznikne kružnice. Jestliže ne, hranu přidáme ke *K*.

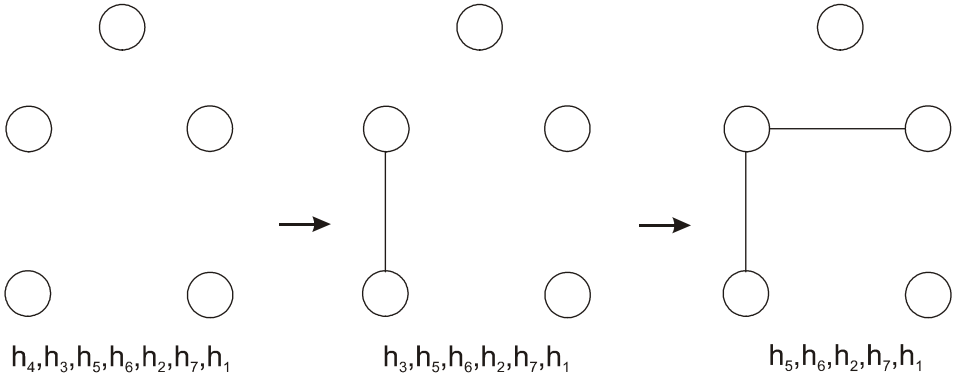
Příklad. V následujícím grafu máme najít jeho minimální kostru. Čísla vedle popisů hran označují ohodnocení hran.



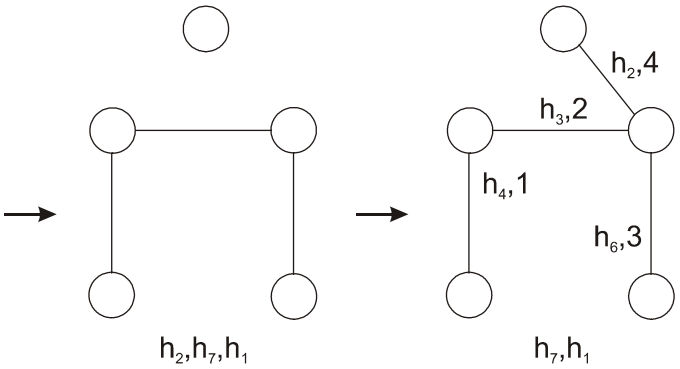
Hrany seřadíme v neklesající posloupnost

$h_4, h_3, h_5, h_6, h_2, h_7, h_1$

Postup přidávání hran ukazují následující obrázky.



Přidáním hrany h_5 by vznikla kružnice, proto ji nepřidáme.



Přidáním zbývajících hran h_7 a h_1 by vznikly kružnice, proto je už nepřidáme.

Ohodnocení sestavené kostry je

$$s = w(h_2) + w(h_3) + w(h_4) + w(h_6) = 4 + 2 + 1 + 3 = 10 \ .$$

4.2 Vyřazovací algoritmus

Hrany grafu G uspořádáme tak, že jejich ohodnocení tvoří nerostoucí posloupnost.

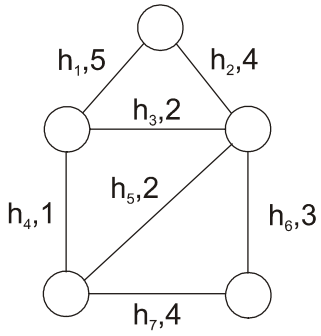
$$h_{i1}, h_{i2}, ..., h_{in}$$

$$w(h_{i1}) \geq w(h_{i2}) \geq ... \geq w(h_{in})$$

Na začátku jako podgraf K vezmeme celý výchozí graf G. Postupně procházíme posloupnost hran od začátku směrem ke konci a u každé hrany ověřujeme, zda leží na nějaké kružnici, tj. zda

jejím odebráním by se neporušila souvislost komponenty. Pokud se souvislost neporuší, hranu odebereme z *K*.

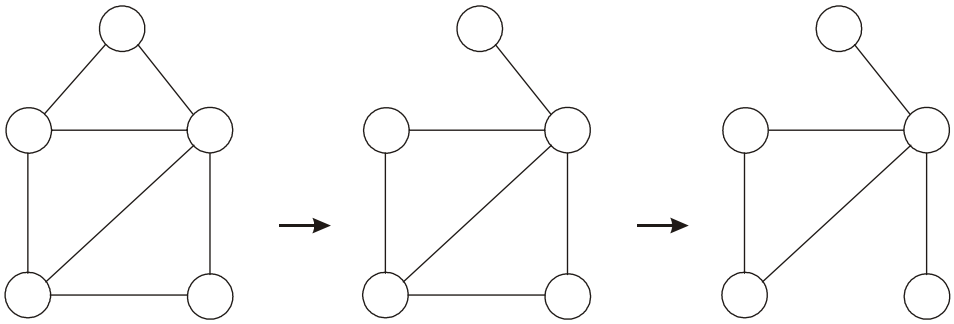
Příklad. Vezmeme stejný výchozí graf jako v příkladu u předchozího algoritmu.



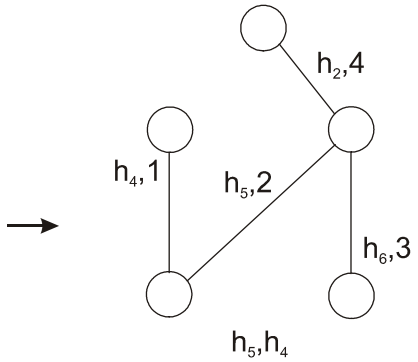
Hrany seřadíme v nerostoucí posloupnost

$h_1, h_2, h_7, h_6, h_3, h_5, h_4$

Postup přidávání hran ukazují následující obrázky.



Hranu h_2 nebylo možné odebrat, protože by se porušila souvislost grafu.



Zbývající hrany h_5 a h_4 už není možné odebrat, protože by se porušila souvislost grafu. Ohodnocení sestavené kostry je

$$s = w(h_2) + w(h_4) + w(h_5) + w(h_6) = 4 + 1 + 2 + 3 = 10 \ .$$

Dostali jsme sice jinou kostru než v minulém příkladě, ale její ohodnocení je stejné. Má-li graf více stejně ohodnocených hran, může mít více různých minimálních koster, záleží v jakém pořadí stejně ohodnocené hrany dáme do posloupnosti pro zařazování nebo vyřazování hran.

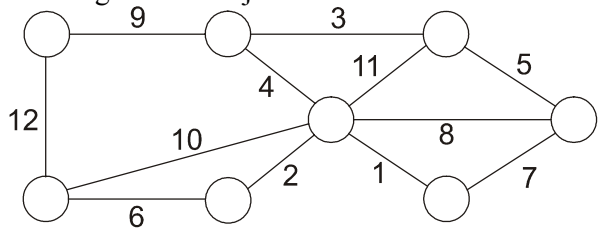
Kontrolní otázky

1. *Co je minimální kostra grafu?*

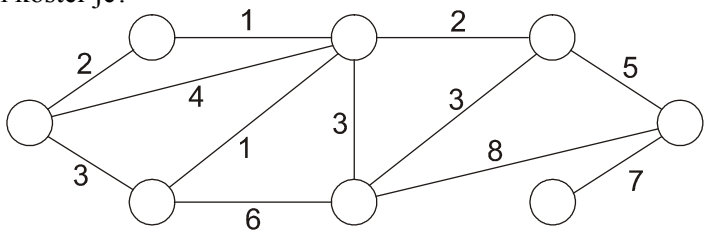
2. *Kdy může mít graf více různých minimálních koster?*

Cvičení

1. Zařazovacím algoritmem najděte minimální kostru následujícího grafu.



2. Vyřazovacím algoritmem najděte všechny minimální kostry následujícího grafu. Kolik různých koster je?

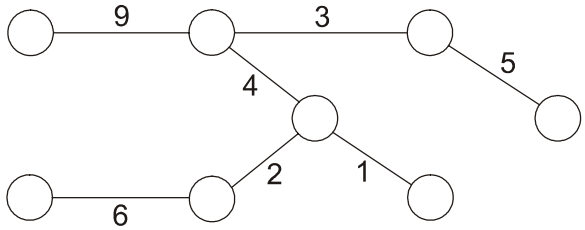


Úkoly k textu

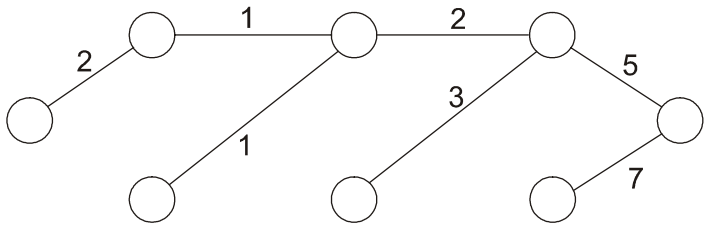
Co kdybychom v zařazovacím algoritmu podmínku, že "přidáním dané hrany k podgrafu *K* nesmí v něm vzniknout kružnice", nahradili podmínkou, že "přidáním dané hrany k podgrafu *K* se musí snížit počet jeho komponent". Bylo by to v pořádku? Pracoval by algoritmus s touto podmínkou stejně dobře jako s tou stávající?

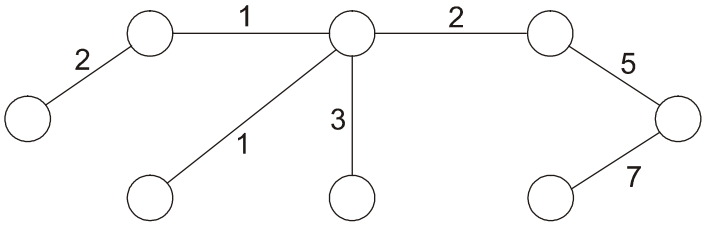
Řešení

1. Minimální kostra daného grafu je na následujícím obrázku.



2. Graf má dvě různé minimální kostry.




1
2
2
5
3
1
7

5 Kružnice v grafu

Studijní cíle: Seznámit s kružnicemi grafu a jejich fundamentální systémem.

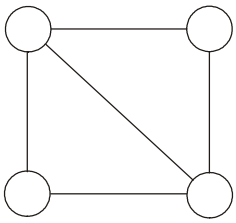
Klíčová slova: Fundamentální systém, cyklomatické číslo.

Potřebný čas: 1 hodina

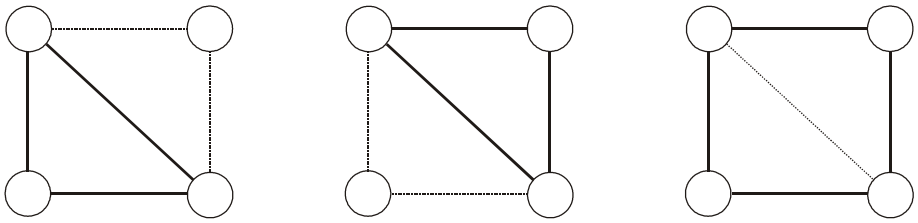
Kružnice je uzavřená cesta grafu. V grafu s kružnicemi lze najít podsystém kružnic takový, že všechny ostatní kružnice jsou sestaveny jen z hran obsažených v kružnicích tohoto podsystému. Tomuto podsystému kružnic říkáme fundamentální systém kružnic grafu.

Definice. Fundamentální systém kružnic grafu je nejmenší podmnožina kružnic grafu taková, že každá hrana grafu, jež leží na nějaké jeho kružnici, je součástí některé kružnice obsažené ve fundamentálním systému.

Příklad. Uvažujme následující graf.

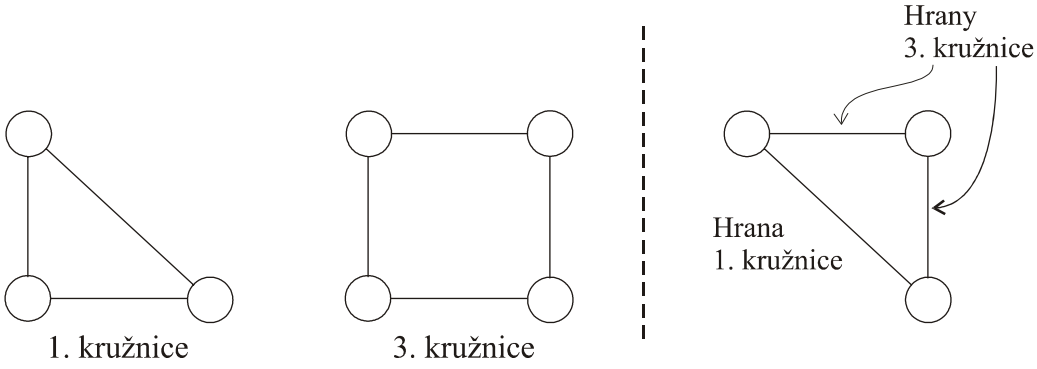


Tento graf obsahuje 3 kružnice, které jsou vyznačeny na následujícím obrázku



1. kružnice 2. kružnice 3. kružnice

Z nich lze vzít libovolné dvě jako fundamentální systém a zbývající třetí kružnici lze pak sestavit z jejich hran. Tím u tohoto grafu dostáváme 3 možné fundamentální systémy kružnic. Vezměme například jako fundamentální systém 1. a 3. kružnici na předchozím obrázku. Z jejich hran můžeme sestavit zbývající 2. kružnici.



1. kružnice 3. kružnice

Předchozí příklad naznačuje, že grafy s více kružnicemi mají běžně více různých fundamentálních systémů kružnic, nicméně počet kružnice ve všech fundamentálních systémech téhož grafu je stejný. Nalezení fundamentálního systému kružnic pro daný graf je poměrně jednoduché, jak ukazuje následující algoritmus.

Algoritmus pro nalezení fundamentálního systému kružnic grafu

Fundamentální systém kružnic grafu dostaneme přidáváním tětiv ke kostře grafu.

Mějme graf G = (H, U, ρ) s počtem hran n=|H|, počtem uzlů m=|U| a počtem komponent p. Vezmeme libovolnou kostru grafu. Kostra je takový podgraf daného, který má minimální počet hran a přitom počet komponent zůstává zachován. Podle věty, kterou jsme uvedli v první kapitole, je nejmenší počet hran v grafu, který má p komponent, roven m-p.

Označme hrany, které jsou obsaženy v kostře

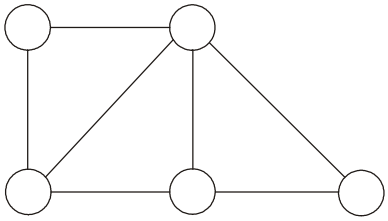
$$h_1^{'}, h_2^{'}, ..., h_{m-p}^{'}$$

a hrany výchozího grafu G, které nejsou v kostře (těch je n-m+p)

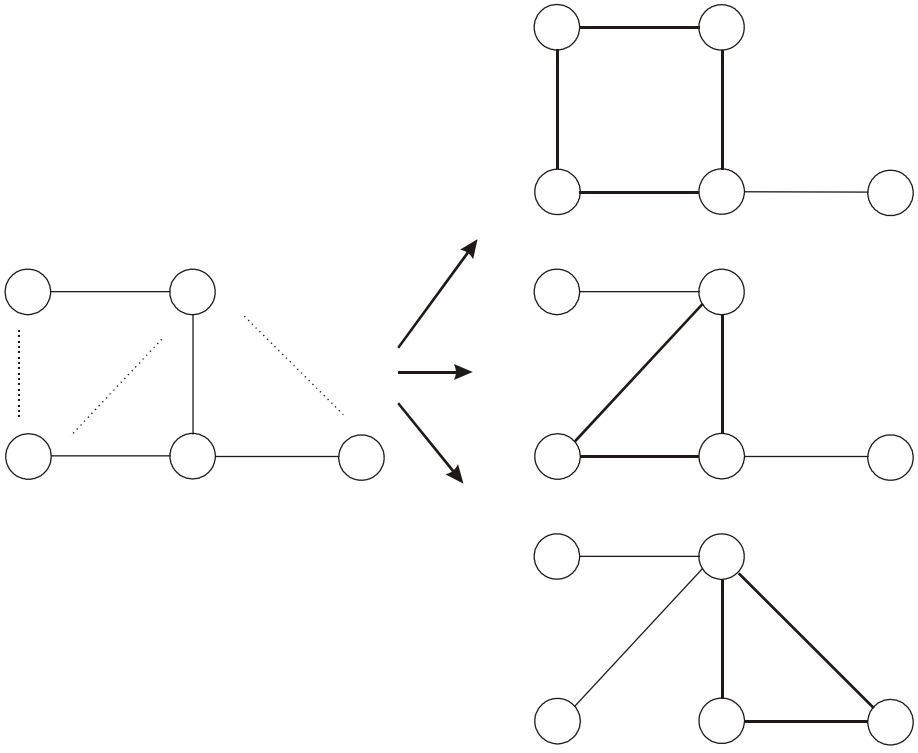
$$h_1^{*}, h_2^{*}, ..., h_{n-m+p}^{*}$$

Přidáním vždy jedné této hrany ke kostře vznikne v grafu jedna kružnice. Takto postupně dostaneme n-m+p kružnic, které budou tvořit fundamentální systém kružnic grafu G.

Příklad. Máme najít fundamentální systém kružnic následujícího grafu.



Následující obrázek ukazuje jednu z možných koster tohoto grafu a tři kružnice fundamentálního systému kružnic.



Které kružnice budou tímto algoritmem zařazeny do fundamentálního systému, záleží na tom, kterou kostru ze všech možných koster grafu zvolíme.

Cyklomatické číslo patří mezi významné charakteristiky grafu.

Definice. Počet kružnic ve fundamentálním systému kružnic grafu G nazýváme cyklomatické číslo grafu a značíme ho $\mu(G)$.

Z předchozího plyne, že cyklomatické číslo grafu je dáno vztahem

$$\mu(G) = |H| - |U| + p \quad ,$$

kde *p* je počet komponent grafu.

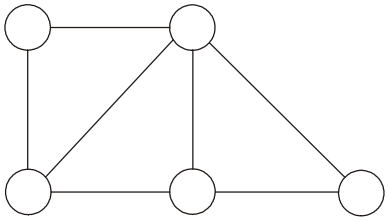
Dále se zavádí hodnost grafu, která se značí $h(G)$ a je definována vztahem

$$h(G) = |U| - p \quad .$$

Hodnost grafu viditelně vyjadřuje počet hran v kostře grafu. Použitím hodnosti lze cyklomatické číslo grafu napsat ve tvaru

$$\mu(G) = |H| - h(G) \quad .$$

Příklad. V grafu *G* z předchozího příkladu



je |H|=7, |U|=5 a p=1. Odtud cyklomatické číslo grafu je

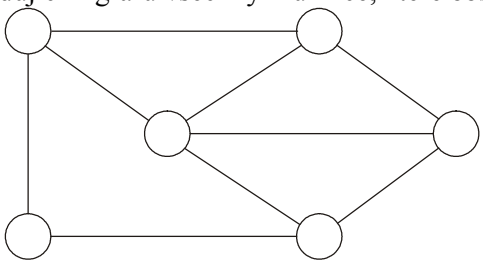
$$\mu(G) = 7 - 5 + 1 = 3 \quad .$$

Kontrolní otázky

1. *Co je to fundamentální systém kružnic grafu?*
2. *Jak se vypočítá cyklomatické číslo grafu?*

Cvičení

1. Najděte v následujícím grafu všechny kružnice, které obsahují největší počet hran.



9.

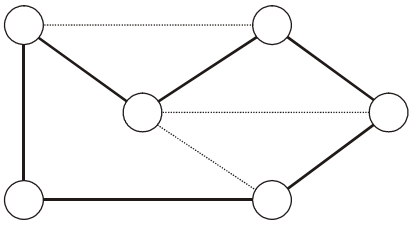
2. Sestavte souvislý graf se třemi kružnice, ve kterém všechny jeho kružnice jsou ve fundamentálním systému.

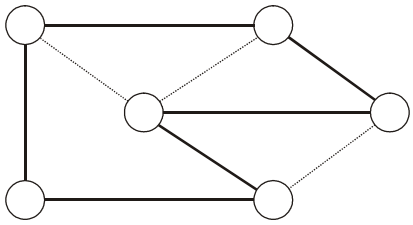
Úkoly k textu

Je dán graf, ve kterém je jen jedna kružnice. A já o něm tvrdím, že jeho chromatické číslo určitě není větší než 3. Zjistěte, zda mám pravdu.

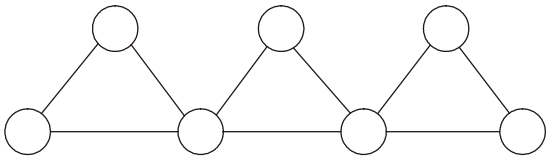
Řešení

1. Graf má 2 kružnice se 6 hranami.





2. Jeden z takový grafů je na obrázku.



6 Vzdálenosti v grafu

Studijní cíle: Vysvětlit algoritmy pro výpočet vzdáleností v grafech.

Klíčová slova: Délka cesty.

Potřebný čas: 2 hodiny

Mějme souvislý graf G = (H, U, ρ), jehož hrany jsou ohodnoceny nezápornými reálnými čísly. Toto ohodnocení zapíšeme jako zobrazení

$$w\colon\ H \to R_0^+$$

kde R_0^+ označuje množinu nezáporných reálných čísel.

Čísla, jimiž jsou jednotlivé hrany ohodnoceny, budeme nazývat délkami těchto hran.

Délkou cesty budeme nazývat součet délek všech hran obsažených na této cestě.

Definice. Vzdáleností d(u,v) dvou uzlů *u* a *v* souvislém grafu *G* nazveme délku nejkratší z cest mezi oběma uzly *u* a *v*.

Věta. Jsou-li délky hran v grafu kladná čísla, pak vzdálenost v grafu splňuje axiómy metriky. Tedy pro libovolné tři uzly *u*, *v* a *w* platí:

Vzdálenost v grafu má vlastnosti metriky.

I. $d(u,v) \geq 0$, přičemž $d(u,v) = 0$ právě když $u = v$

II. $d(u,v) = d(v,u)$

III. $d(u,v) \leq d(u,w) + d(w,v)$

Důkaz:

Vlastnost *I.* je zřejmá. Pokud *u* a *v* jsou různé uzly, musí na cestě mezi nimi být aspoň jedna hrana.

Vlastnost *II.* je rovněž zřejmá. Nejkratší cesta z uzlu *u* do *v* je zároveň i nejkratší cesta z uzlu *v* do *u*.

I vlastnost *III.* je poměrně zřejmá. Neboť spojením cest mezi *u* a *w* a mezi *w* a *v* dostaneme cestu mezi *u* a *v*, jejíž délka, jež je součtem vzdáleností mezi *u* a *w* a mezi *w* a *v*, nemůže být menší než vzdálenost mezi koncovými uzly cesty *u* a *v*, což je délka nejkratší z cest mezi těmito dvěma uzly.

Následující algoritmus hledá nejkratší cestu (a tím i počítá vzdálenost) mezi dvěma zvolenými uzly grafu *u* a *v*. Algoritmus pracuje tak, že jeden z koncových uzlů zvolí jako výchozí, nechť je to třeba uzel *u*, a postupně v ostatních uzlech počítá délky cest mezi nimi a uzlem *u*. Pro tyto délky má v každému uzlu proměnnou *c*, ve které je v každém okamžiku uložena délka nejkratší z doposud nalezených cest mezí tímto uzlem a uzlem *u*. Dále si zavedeme množinu *S*, ve které budeme mít uzly, u kterých jsme zatím neprovedli výpočet délky cest, jež jdou přes tyto uzly k jejich sousedům.

Aktuální uzel, pro který právě počítáme délky cest jdoucí od uzlu *u* přes tento uzel k jeho sousedům, budeme značit *z*.

Popis algoritmu

1. Počáteční nastavení.
   - $c(u) = 0$ - pro uzel *u* je délka nejkratší nalezené cesty = 0.

Pro výpočet vzdálenosti dvou uzlů výpočet existuje účinný algoritmus.

- $c(w) = +\infty$ - pro všechny ostatní uzly $w \neq u$ je délka cesty mezi w a u na začátku nastavena na nekonečno. Tato hodnota vyjadřuje, že nebyla zatím vypočítána délka žádné cesty od uzlu u k uzlu w.
- $S=U$ - do množiny S na začátku dáme všechny uzly grafu.
- $z=u$ - aktuální uzel z, u kterého právě probíhá výpočet, na začátku nastavíme na výchozí uzel u.

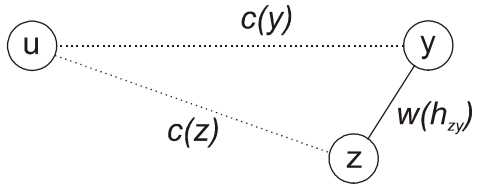
2. Výpočet délek cest, jež vedou od uzlu u přes aktuální uzel z k sousedům aktuálního uzlu z. Pro všechny uzly y, které
   - jsou sousedé aktuálního uzlu z
   - a zároveň patří do množiny S ($y \in S$)

   vypočítáme novou délku $c(y)$ cesty mezi uzlem y a výchozím uzlem u

   $$c(y) = \min(c(y), c(z) + w(h_{zy}))$$

   kde h_{zy} označuje hranu mezi uzly z a y a $w(h_{zy})$ označuje délku této hrany.

   Což znamená, že je-li nová cesta mezi u a y přes uzel z kratší než doposud nalezená nejkratší cesta mezi uzly u a y, pak proměnná c vyjadřující délku nejkratší doposud nalezené cesty se v uzlu y nastaví na délku nové cesty.



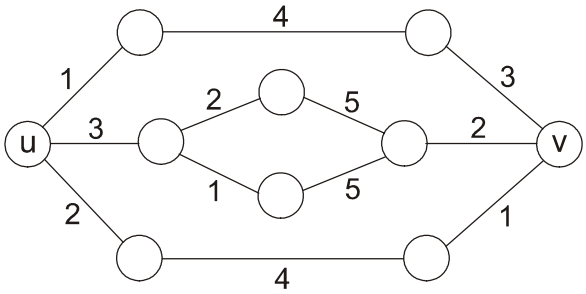
   Aktuální uzel z odebereme z množiny S, neboť délka cest jdoucích přes tento uzel k jeho sousedům už byla vypočítána

   $$S = S - \{u\} \quad .$$

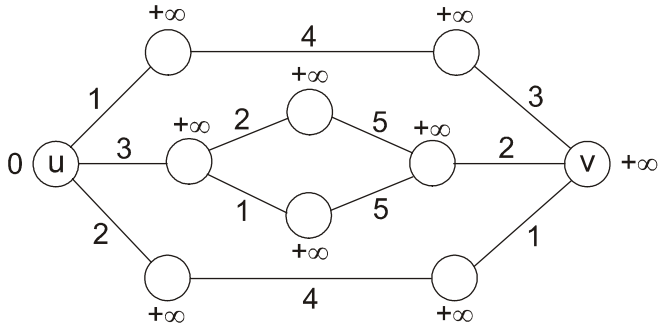
3. V množině najdeme uzel s nejnižší hodnotou c a ten učiníme novým aktuálním uzlem z. Je-li tímto uzlem druhý koncový uzel hledané cesty v, pak výpočet končí a vzdálenost mezi uzly u a v je rovna hodnotě proměnné c v tomto uzlu. Tedy $d(u,v) = c(v)$.

   Jinak přejdeme ke kroku 2.

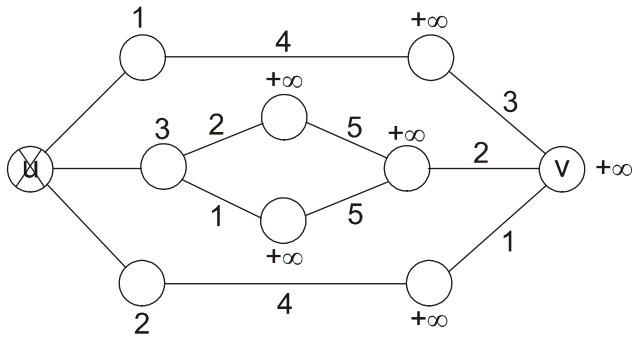
Příklad. V následujícím grafu G máme vypočítat vzdálenost mezi jeho uzly u a v.



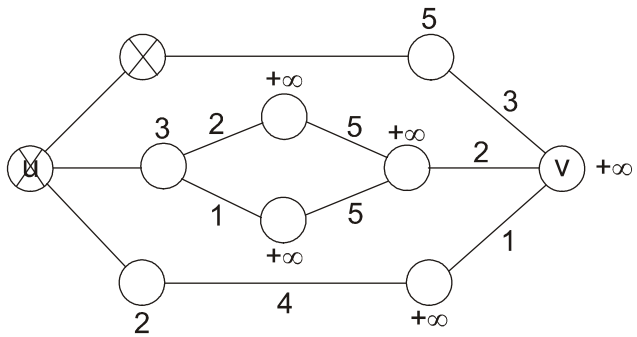
Nastavíme příslušně ve všech uzlech počáteční hodnoty proměnné c.



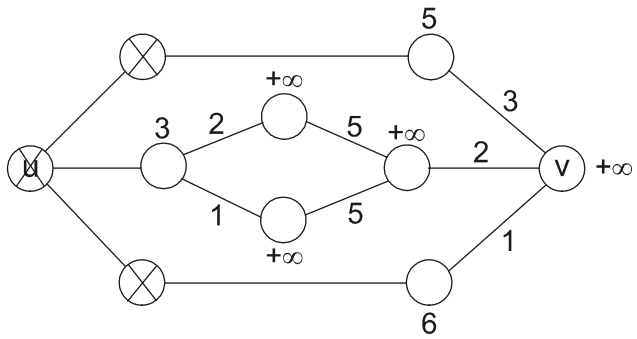
Aktuální uzel je počáteční uzel *u*. Provedeme pro něj výpočet a jeho vyřazení z množiny *S* označíme jeho přeškrtnutím.



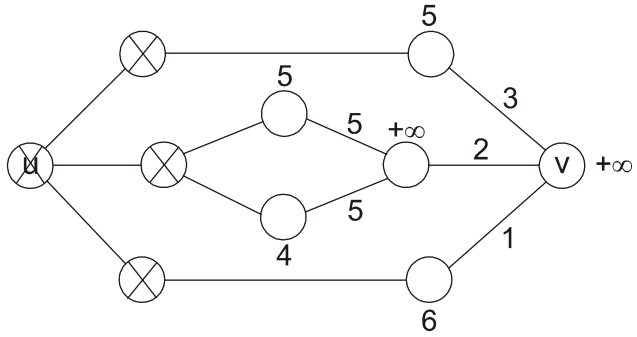
Uzel s nejmenší hodnotou *c* je uzel s hodnotou *1*. Učiníme ho aktuálním uzlem a provedeme pro něj výpočet.



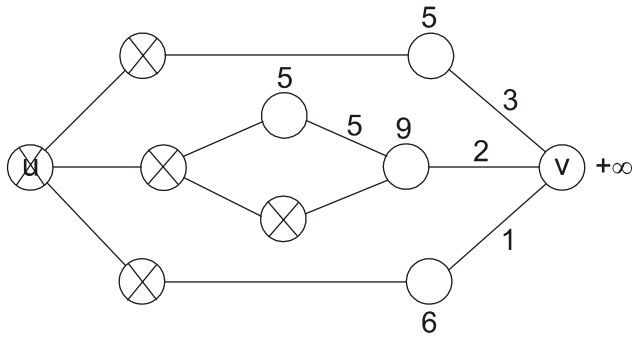
Uzel s nejmenší hodnotou *c* je uzel s hodnotou 2. Ten bude nyní aktuálním uzlem.



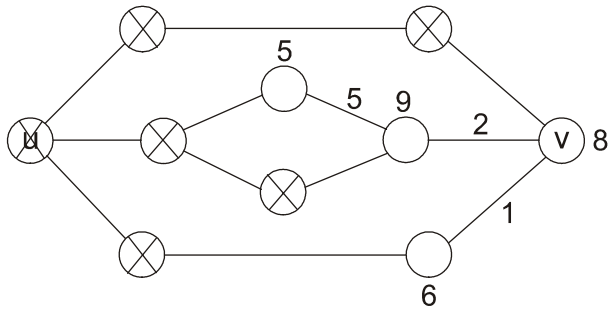
Uzel s nejmenší hodnotou *c* je uzel s hodnotou 3. Učiníme ho aktuálním uzlem.



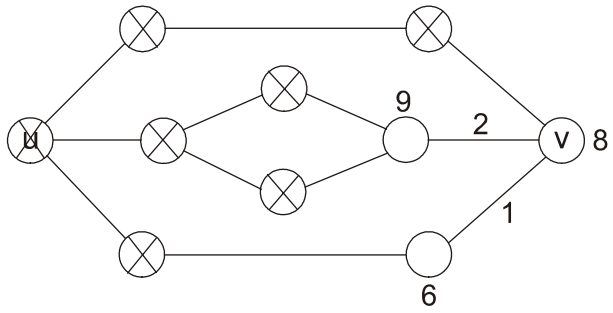
Uzel s nejmenší hodnotou *c* je uzel s hodnotou 4. Učiníme ho aktuálním uzlem.



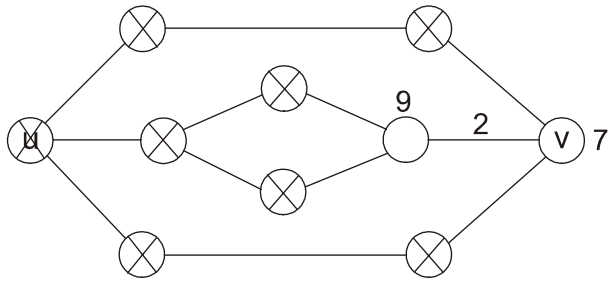
Nyní máme dva uzly s nejmenší hodnotou *c*. Jako aktuální vezmeme třeba ten, který je na obrázku nahoře.



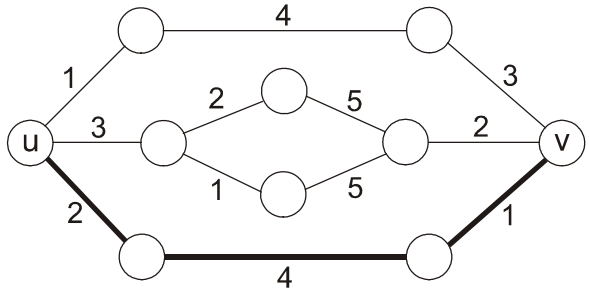
Uzel s nejmenší hodnotou *c* je uzel s hodnotou 5. Učiníme ho aktuálním uzlem.



Uzel s nejmenší hodnotou *c* je uzel s hodnotou 6. Učiníme ho aktuálním uzlem.



Uzel s nejmenší hodnotou *c* je nyní druhý koncový uzel cesty *v*. Výpočet tím končí a vzdálenost mezi uzly *u* a *v* je rovna 7. Nejkratší cestu ukazuje následující obrázek.



Pokud potřebujeme výpočet vzdáleností ne jen mezi dvěma určitými uzly grafu, ale mezi všemi dvojicemi uzlů grafu, lze sice použít předchozí algoritmus tak, že ho postupně aplikujeme na jednotlivé dvojice uzlů, nicméně v tomto případě je mnohem rychlejší všechny vzdálenosti počítat souběžně, což dělá následující algoritmus.

Následující algoritmus je původně navržen pro orientovaný graf. Dá se přirozeně použít i pro neorientovaný graf, jestliže neorientované hrany reprezentujeme dvojicemi opačně obrácených orientovaných hrany.

Mějme orientovaný graf G = (H, U, ρ), jehož hrany jsou ohodnoceny nezápornými reálnými čísly:

$w\colon\ H \to R_0^+$.

Množina uzlů grafu nechť je $U = \{u_1, u_2, \ldots, u_m\}$.

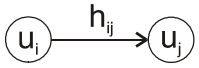
Výpočtem matice vzdáleností najdeme současně vzdálenosti všech uzlů v grafu.

Popis algoritmu.

Algoritmus počítá matici vzdáleností $D = (d_{ij})$, kde d_{ij} je vzdálenost z uzlu u_i do uzlu u_j, tj. $d_{ij} = d(u_i, u_j)$.

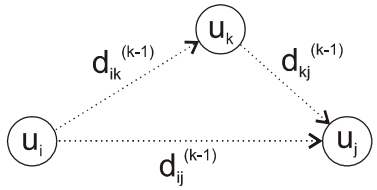
Výpočet začíná od matice ohodnocení hran $D^{(0)} = (d_{ij}^{(0)})$, jejíž prvky jsou dány předpisem:

- $d_{ij}^{(0)} = w(h_{ij})$, existuje-li hrana h_{ij} z uzlu u_i do uzlu u_j, přičemž $w(h_{ij})$ je ohodnocení této hrany.



- $d_{ij}^{(0)} = +\infty$, jestliže v grafu není hrana z uzlu u_i do uzlu u_j.
- $d_{ij}^{(0)} = 0$ (na diagonále matice jsou nuly)

Následně se postupně počítají matice $D^{(1)}, \ldots, D^{(m)}$, přičemž každá matice $D^{(k)}$ se počítá z předchozí matice $D^{(k-1)}$ tak, že uvažuje možnost cesty přes uzel u_k.



Jako nová délka cesty $d_{ij}^{(k)}$ z uzlu u_i do uzlu u_j se vezme

- Předchozí vypočítaná délka cesty $d_{ij}^{(k-1)}$, jestliže cesta přes uzel u_k není kratší.

- Délka cesty přes uzel u_k, která je $d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$, jestliže tato cesta je kratší než předchozí vypočítaná cesta s délkou $d_{ij}^{(k-1)}$.

Tedy matice $D^{(k)}$ se vypočítá dvěma vnořenými cykly

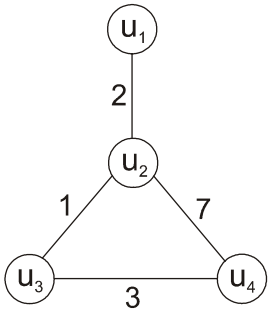
Cyklus pro *i* do 1 do *m*

Cyklus pro *j* od 1 do *m*

$$d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$$

Poslední vypočítaná matice D^m je hledanou maticí vzdáleností D, tj. $D = D^{(m)}$.

Příklad. V následujícím grafu G máme vypočítat vzdálenosti mezi uzly grafu.



Sestavíme počáteční matici $D^{(0)}$ délek hran mezi uzly.

$$D^{(0)} = \begin{pmatrix} 0 & 2 & +\infty & +\infty \\ 2 & 0 & 1 & 7 \\ +\infty & 1 & 0 & 3 \\ +\infty & 7 & 3 & 0 \end{pmatrix}$$

Protože graf je neorientovaný, matice je symetrická dle hlavní diagonály. Následně postupně počítáme matice $D^{(1)}$ až $D^{(4)}$.

$$D^{(1)} = \begin{pmatrix} 0 & 2 & +\infty & +\infty \\ 2 & 0 & 1 & 7 \\ +\infty & 1 & 0 & 3 \\ +\infty & 7 & 3 & 0 \end{pmatrix}$$

kde například $d_{23}^{(1)} = \min(1, 2+\infty) = 1$, $d_{34}^{(1)} = \min(3, \infty+\infty) = 3$.

$$D^{(2)} = \begin{pmatrix} 0 & 2 & 3 & 9 \\ 2 & 0 & 1 & 7 \\ 3 & 1 & 0 & 3 \\ 9 & 7 & 3 & 0 \end{pmatrix}$$

kde například $d_{13}^{(2)} = \min(\infty, 2+1) = 3$, $d_{34}^{(2)} = \min(3, 1+7) = 3$.

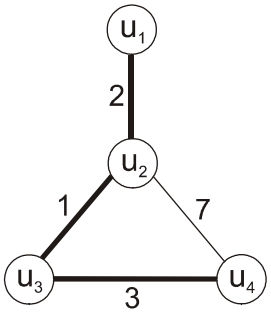
$$D^{(3)} = \begin{pmatrix} 0 & 2 & 3 & 6 \\ 2 & 0 & 1 & 4 \\ 3 & 1 & 0 & 3 \\ 6 & 4 & 3 & 0 \end{pmatrix}$$

kde například $d_{14}^{(3)} = \min(9, 3+3) = 6$, $d_{24}^{(3)} = \min(7, 1+3) = 4$.

$$D^{(4)} = \begin{pmatrix} 0 & 2 & 3 & 2 \\ 2 & 0 & 1 & 4 \\ 3 & 1 & 0 & 3 \\ 2 & 4 & 3 & 0 \end{pmatrix}$$

kde například $d_{12}^{(4)} = \min(2, 6+4) = 2$, $d_{13}^{(3)} = \min(3, 6+3) = 3$.

Z poslední matice plyne, že vzdálenost mezi uzly u_1 a u_4 je 6. Nejkratší cestu mezi těmito uzly ukazuje následující obrázek.

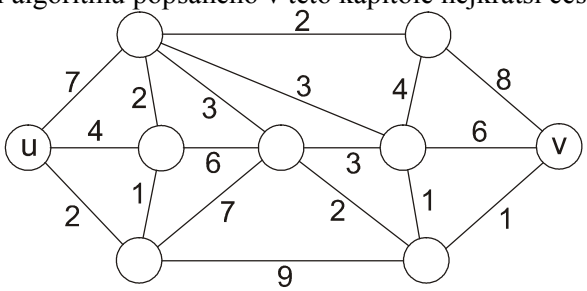


Kontrolní otázky

1. *Co je délka cesty mezi dvěma danými uzly?*
2. *Jaká je souvislost mezi délkou cesty mezi dvěma danými uzly a vzdáleností mezi těmito dvěma uzly?*

Cvičení

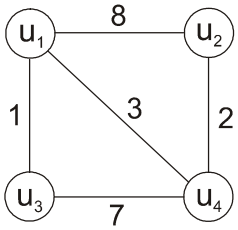
1. Najděte pomocí algoritmu popsaného v této kapitole nejkratší cestu mezi uzlu *u* a *v*.



10.

2. Vypočítejte maticí vzdáleností následujícího grafu.

11.



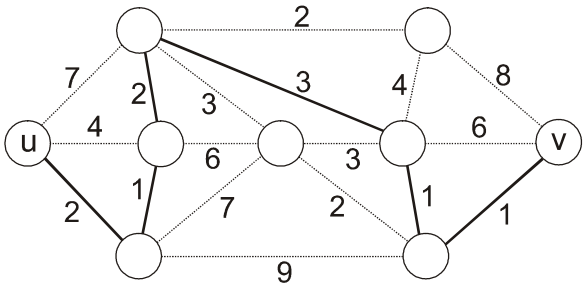
12.

Úkoly k textu

Máme graf, o kterém víme, že je souvislý a vzdálenost mezi všemi jeho dvojicemi uzlů je stejná. Zjistěte, jaký je to typ grafu.

Řešení

1. Nejkratší cesta je silně vytažena na následujícím obrázku.



2. Matice je

$$D^{(4)} = \begin{pmatrix} 0 & 5 & 1 & 3 \\ 5 & 0 & 6 & 2 \\ 1 & 6 & 0 & 4 \\ 3 & 2 & 4 & 0 \end{pmatrix}$$

7 Eulerovské grafy

Studijní cíle: Seznámit studujícího s eulerovskými grafy.

Klíčová slova: Pokrytí grafu, eulerovský tah.

Potřebný čas: 1 hodina

Tato část se zabývá pokrytím grafu tahy.

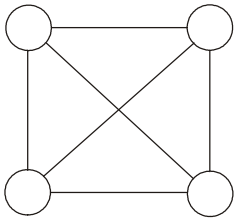
Definice. Pokrytí grafu $G = (H, U, \rho)$ je rozklad jeho množiny hran H na podmnožiny

$$H_1, \ldots, H_k$$

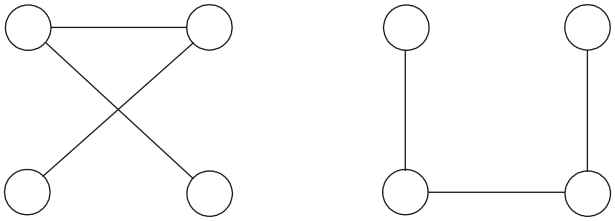
takový, že hrany v každé této podmnožině tvoří v grafu tah.

Rozklad s nejmenším možným počtem podmnožin nazveme minimálním pokrytím grafu.

Příklad. Následující graf



lze pokrýt dvěma tahy, jak ukazuje následující obrázek.



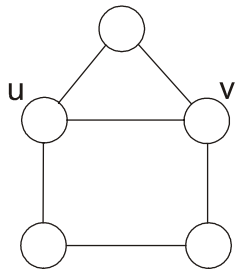
Problém, kdy lze graf pokrýt jedním tahem, řeší následující věta.

Věta. Nechť G je souvislý graf. Pak graf G lze pokrýt jedním tahem právě když obsahuje nejvýše dva uzly lichého stupně. Tento tah je uzavřený právě když všechny uzly v grafu mají sudý stupeň.

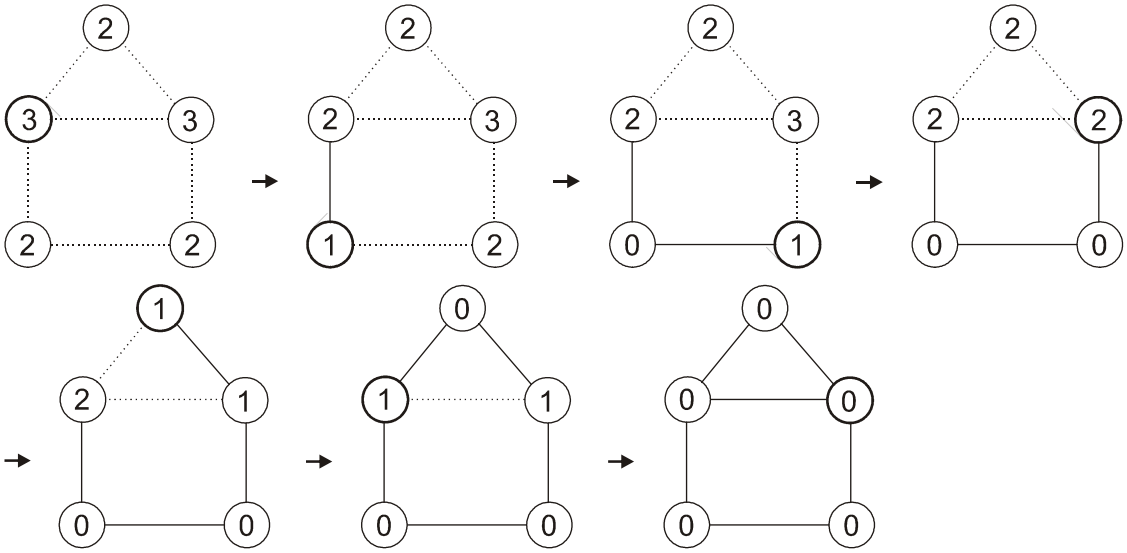
Důkaz: Nechť graf má nejvýše dva uzly lichého stupně. Už v první kapitole tohoto studijního materiálu jsme ukázali, že součet stupňů všech uzlů grafu je sudé číslo. Z toho plyne, že graf má v tomto případě má buďto dva uzly lichého stupně anebo žádný. Uvažujme, že má dva uzly lichého stupně, označme je u a v. Tyto uzly budou koncovými uzly tahu, který pokrývá graf.

Tah vytvoříme tak, že začneme v jeho koncovém uzlu u a vezmeme libovolnou hranu incidující s tímto uzlem a dáme ji do tahu. Nechť druhý koncový uzel přidané hrany je w. Nyní vezmeme, libovolnou hranu incidující s tímto uzlem, která ještě není v tahu, a přidáme ji také k tahu. Tak pokračujeme v přidávání hran k tahu tak dlouho, dokud lze nějakou hranu přidat. Tímto způsobem se nakonec po přidání všech hran dostaneme do druhého uzlu s lichým stupněm v. V uzlu se sudým stupněm skončit nemůžeme, protože v okamžiku, kdy se do něho dostaneme tak, že přidáme hranu s ním incidující k tahu, zůstává lichý počet hran s ním incidujících, které ještě v tahu nejsou. Tudíž existuje vždy další hrana, kterou lze přidat k tahu. Uzel v je jediný, ve kterém může nastat situace, že jsme přidali hranu s tímto uzlem na konci a už v tomto uzlu není žádná další hrana, která ještě není v tahu.

Ukažme tento postup konstrukce tahu na následujícím grafu.



V následujících obrázcích je postup konstrukce tahu, přičemž čísla v uzlech ukazují počet hran, s kterými uzel inciduje a které přitom ještě nejsou v tahu.



Mají-li všechny uzly grafu sudý stupeň, můžeme začít konstrukci tahu v libovolném uzlu a nakonec i v tomto uzlu skončíme. Dostaneme tak uzavřený tah pokrývající celý graf.

Předpokládejme naopak, že graf lze pokrýt jedním tahem. Pak nutně uzly grafu, které nejsou koncovými uzly tahu, musí mít sudý stupeň, protože každý jejich výskyt v tahu znamená incidenci se dvěma hranami – hranou v tahu před ním a hranou v tahu za ním. Tudíž vyskytuje-li se nějaký nekoncový uzel tahu v něm k-krát, jeho stupeň v grafu je $2*k$. Jsou-li koncové uzly tahu různé, musí mít liché stupně. Naopak je-li to stejný uzel, musí tento uzel mít sudý stupeň.

Eulerův graf se dá pokrýt jedním uzavřeným tahem.

Definice. Graf, který lze pokrýt jedním uzavřeným tahem nazýváme Eulerovým grafem.

Jak plyne z předchozí věty, graf je Eulerův právě když je souvislý a všechny jeho uzly mají sudý stupeň.

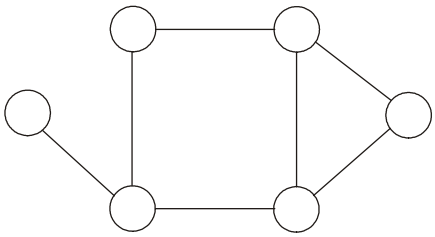
Zjistit, zda graf je Eulerův, je jednoduché.

Uveďme bez důkazu větu, která je určitým zobecnění předchozí věty.

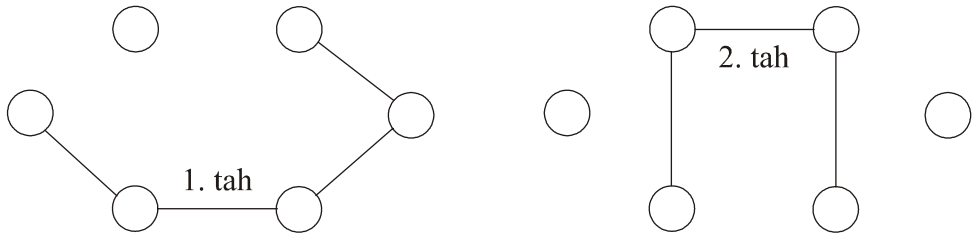
Věta. Nechť G je souvislý graf, který má $2*k$ uzlů lichého stupně, kde k je přirozené číslo. Pak graf G lze pokrýt k tahy.

Zřejmě tyto tahy budou mít konce v uzlech grafu, které mají lichý stupeň.

Příklad. Následující graf má 4 uzly s lichým stupněm.



Lze ho tedy pokrýt dvěma tahy. Mohou to být třeba tahy, které ukazuje následující obrázek.

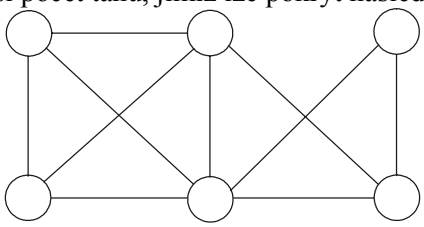


Kontrolní otázky

1. *Jaký graf se označuje přívlastkem "eulerovský"?*
2. *Jak se dá zjistit, zda graf je eulerovský?*

Cvičení

1. Jaký je nejmenší počet tahů, jimiž lze pokrýt následující graf? Najděte tyto tahy.

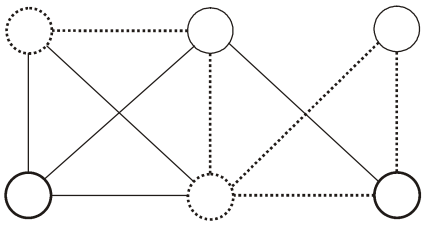
13. 

Úkoly k textu

Zjistěte, zda je pravdivé tvrzení, že v eulerovském grafu každá jeho hrana leží v nějaké kružnici grafu.

Řešení

1. Graf má 4 uzly lichého stupně, k jeho pokrytí jsou zapotřebí dva tahy. Jednu z možností, jak tyto tahy zvolit, ukazuje následující obrázek.



8 Hamiltonovské grafy

Studijní cíle: Seznámit studujícího s hamiltonovskými grafy. Poskytnout heuristické algoritmy pro řešení úlohy obchodního cestujícího.

Klíčová slova: Hamiltonovská kružnice, úloha obchodního cestujícího.

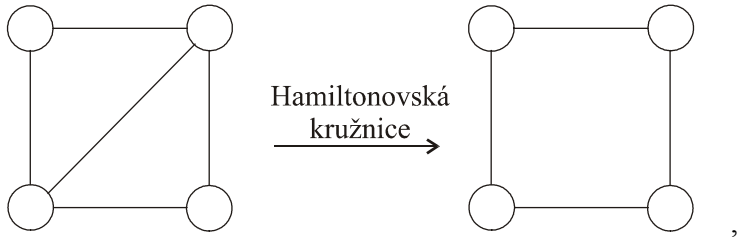
Potřebný čas: 3 hodiny

Zatímco u eulerovských grafů jsme hledali uzavřený tah, který obsahuje všechny hrany grafu, u hamiltonovských grafů hledáme uzavřenou cestu, tj. kružnici, která obsahuje všechny uzly grafu.

Hamiltonovská kružnice obsahuje všechny uzly grafu.

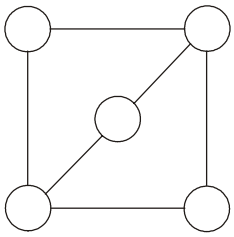
Definice. Kružnici nazveme hamiltonovskou, jestliže obsahuje všechny uzly grafu. Graf *G* nazveme hamiltonovský, pokud v něm existuje hamiltonovská kružnice.

Příklad. Následující graf je hamiltonovský



,

kdežto graf na dalším obrázku hamiltonovský není.



Zatímco zjistit, zda graf je eulerovský, je jednoduché, u hamiltonovských grafů jde o mnohem obtížnější problém. Je to úloha patřící do třídy NP-obtížných problémů.

Je zřejmé, že čím více má graf při daném počtu uzlů hran, tím větší je naděje, že bude hamiltonovský. Nejjednodušší situace je u úplných grafů. Úplný graf (s aspoň třemi uzly) je vždy hamiltonovský. Protože každý uzel v něm sousedí s každým uzlem. Při sestavení hamiltonovské kružnice lze vyjít z libovolného uzlu a sestavovat cestu postupným přidáváním dalších a dalších uzlů a po vyčerpání všech uzlů ji nakonec uzavřít v kružnici spojením koncového uzlu cesty s počátečním uzlem cesty.

Následující věta ukazuje, že má-li graf dosti velké stupně uzlů, je zaručeno, že je hamiltonovský.

Věta. Mějme graf $G = (H, U, \rho)$ s aspoň třemi uzly ($|U| \geq 3$). Jestliže pro součet stupňů každých dvou jeho nesousedních uzlů *u* a *v* platí

$$d(u) + d(v) \geq |U| \quad ,$$

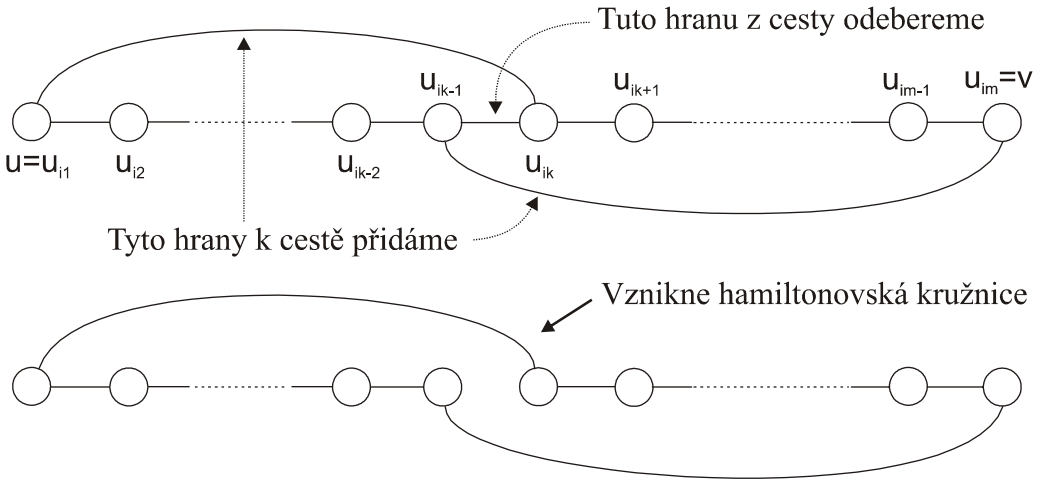
pak graf *G* je hamiltonovský.

Důkaz: Sporem – nechť existuje graf *G*, který splňuje podmínky věty a přitom není hamiltonovský.

Začneme mezi dvojice nesousedních uzlů grafu postupně přidávat hrany tak dlouho, až se graf stane hamiltonovský. To jednou musí nastat, protože přidáváním hran se přinejmenším můžeme dostat až k úplnému grafu, který hamiltonovský je. Nechť poslední přidaná hrana, po níž se graf stal hamiltonovský, spojila uzly grafu u a v a vytvořila tak hamiltonovskou kružnici. Tuto hranu opět odebereme a výsledný graf označíme G'. Graf G' není hamiltonovský, neexistuje v něm hamiltonovská kružnice, nicméně v něm existuje hamiltonovská cesta (cesta obsahující všechny uzly grafu), jejíž koncové uzly jsou u a v. Vezměme uzly grafu v tom pořadí, v jakém se vyskytují na této cestě

$$u = u_{i1}, u_{i2}, \ldots, u_{im-1}, u_{im} = v$$

Pro $k = 2, \ldots, m$ platí, že v grafu neexistují zároveň hrany mezi dvojicí uzlů u a u_{ik} a dvojicí uzlů u_{ik-1} a v. Kdyby obě hrany existovaly, pak, jak ukazuje následující obrázek, bychom přidáním těchto hran k hamiltonovské cestě a odebráním z ní hrany mezi uzly u_{ik-1} a u_{ik} dostali v grafu G' hamiltonovskou kružnici, což je spor s tím, že tento graf není hamiltonovský.



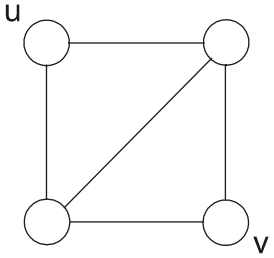
Z toho ovšem pro součet stupňů uzlů u a v plyne

$$d(u) + d(v) < |U|$$

neboť sousedí-li uzel u s k uzly, pak uzel v nemůže sousedit s k uzly, které jsou vlevo od nich. Tedy odečteme-li jejich počet od celkového počtu m-1 ostatních uzlů (uzlů, které nejsou uzlem v), dostaneme, že uzel v může sousedit s nejvýše s m-1-k uzly (kde m=|U|). Dohromady tedy oba dva uzly mají nejvýše k+m-1-k=m-1 sousedů, což je zároveň nejvýše součet jejich stupňů.

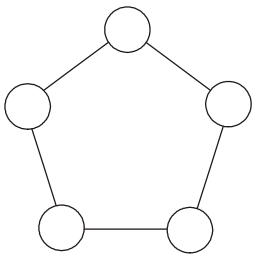
Graf G' vznikl přidáním hran ke grafu G a uzly u, v jsou přitom v grafu G' nesousední. Z toho plyne, že tyto dva uzly jsou i v grafu G nesousední. Protože v grafu G' pro tyto uzly platí $d(u) + d(v) < |U|$, musí pro ně i v grafu G platit $d(u) + d(v) < |U|$. To je ale spor s předpokladem věty.

Příklad. V následujícím grafu jsou jediné dva nesousední uzly u a v.



Jejich stupně jsou $d(u) = 2$, $d(v) = 2$. Počet uzlů v grafu je $|U| = 4$. Graf je hamiltonovský, neboť platí 2+2≥4.

Věta uvádí postačující podmínku existence hamiltonovského grafu. To, že tato podmínka není nutná, ukazuje následující graf, který je viditelně hamiltonovský a přesto v něm ani jedna dvojice nesousedních uzlů nesplňuje podmínku předchozí věty.



Zjistit, zda graf je hamiltonovský, je obtížná úloha.

Protože nalezení hamiltonovské kružnice je NP-obtížný problém, přichází v úvahu hlavně heuristický algoritmus.

Následující algoritmus se snaží postupným přidáváním uzlů sestavit hamiltonovskou cestu a tu konci uzavřít v kružnici. V případě, že se dostane do situace, kdy cesta ještě není kompletní a nelze k ní přidat další uzel, zkouší záměnou pořadí uzlů v ní najít jinou její variantu. Aby opakovaně nezkoušel stejné varianty, používá množinu *S*, do které si ukládá uzly, jejichž použití ke změně cesty už vyzkoušel.

Popis algoritmu

1. Počáteční krok.

   Jako počáteční uzel sestavované hamiltonovské cesty zvolíme libovolný uzel v grafu. Označme ho u_{i1}.

2. Průběžný krok.

   Nechť je již sestavena určitá část cesty a ta nechť je tvořena uzly

   $u_{i1}, u_{i2}, \ldots, u_{ik}$

   Nyní mohou nastat případy:

   - Je $k<m$ (cesta ještě neobsahuje všechny uzly).

     Jestliže poslední uzel v cestě u_{ik} má souseda - uzel u_{ik+1}, který není doposud v cestě obsažen ($u_{ik+1} \neq u_{ij}$ pro $j=1,\ldots,k$), pak tento uzel přidáme k cestě

     $u_{i1}, u_{i2}, \ldots, u_{ik}, u_{ik+1}$.

     Množinu *S* vyzkoušených uzlů nastavíme na prázdnou množinu $S=\varnothing$.

     Přejdeme opět na začátek kroku 2.

     Jestliže se nám nepodařilo cestu prodloužit, přejdeme ke kroku 3., v kterém je hledána jiná varianta cesty.

   - Je $k=m$ (cesta už obsahuje všechny uzly).

     Hledáme-li jen hamiltonovskou cestu (nikoliv kružnici), algoritmus zde úspěšně končí.

     Jinak, existuje-li hrana mezi koncovými uzly cesty u_{i1} a u_{im}, přidáme ji k cestě, čímž vznikne hamiltonovská kružnice a úloha je dokončena.

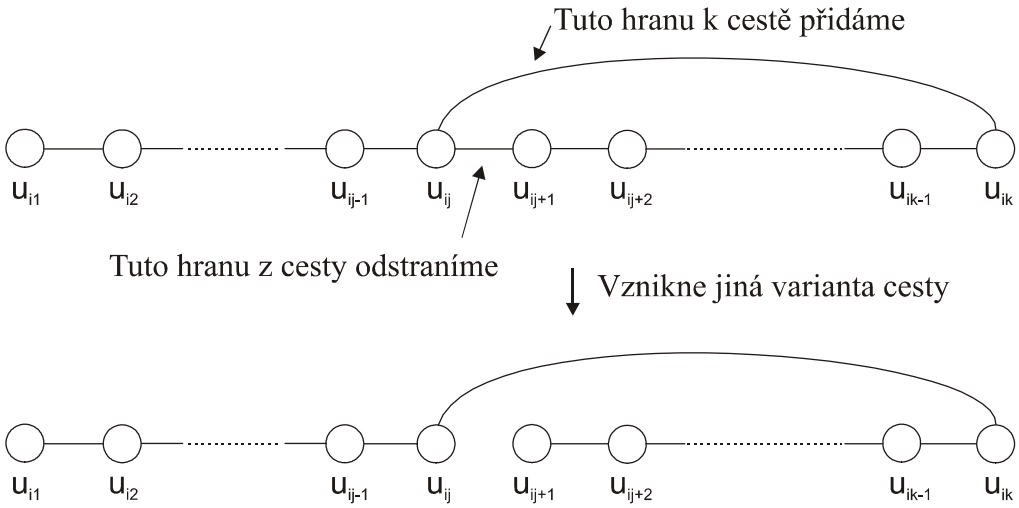
     Neexistuje-li taková hrana, přejdeme ke kroku 3.

3. Hledáme v existující části cesty uzel u_{ij}, $j<k$ takový, že

   - Existuje hrana mezi uzlem u_{ij} a současným koncovým uzlem cesty u_{ik}.
   - Uzel u_{ij+1} není v množině vyzkoušených uzlů *S* ($u_{ij+1} \notin S$).

   Jestliže takový uzel u_{ij} najdeme, pak z cesty odebereme hranu, která je mezi uzly u_{ij}

a u_{ij+1}. Přidáme k cestě hranu, jež je mezi uzly u_{ij} a u_{ik}.



Nyní cesta bude tvořena posloupností uzlů

$u_{i1}, \ldots, u_{ij}, u_{ik}, u_{ik-1}, u_{ik}, \ldots, u_{ij+2}, u_{ij+1}$.

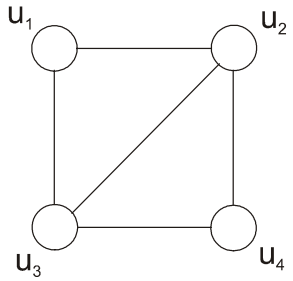
Předchozí koncový uzel u_{ik} přidáme do množiny již zkoušených koncových uzlů S

$S = S \cup \{u_{ik}\}$.

A znovu přejdeme ke kroku 2.

Pokud se nám takový uzel u_{ij} nepodaří najít, algoritmus neúspěšně končí. Nenašel hamiltonovskou cestu, či hamiltonovskou kružnici. Můžeme ho případně opakovat s jiným výchozím uzlem.

Příklad. Předchozím algoritmem hledáme hamiltonovskou kružnici v následujícím grafu.



Jako počáteční uzel cesty zvolíme uzel u_1.

Cesta = u_1.

Existuje soused uzlu u_1 uzel u_2, který ještě v cestě není, přidáme ho k cestě.

Cesta = $u_1\ u_2$. S - prázdná.

Existuje soused uzlu u_2 uzel u_3, který ještě v cestě není, přidáme ho k cestě.

Cesta = $u_1\ u_2\ u_3$. S - prázdná.

Existuje soused uzlu u_3 uzel u_4, který ještě v cestě není, přidáme ho k cestě.

Cesta = $u_1\ u_2\ u_3\ u_4$. S - prázdná.

Cesta již obsahuje všechny uzly, ale nelze ji uzavřít v kružnici – v grafu neexistuje hrana mezi koncovými uzly cesty. Avšak existuje hrana mezi uzlem cesty u_2 a koncovým uzlem u_4. Provedeme změnu cesty způsobem popsaným v algoritmu.

Cesta = $u_1\ u_4\ u_3\ u_2$. $S = \{u_4\}$.

Nyní existuje hrana mezi koncovými uzly cesty u_1 a u_2, přidáním které je dokončeno sestavení hamiltonovské kružnice v grafu.

8.1 Úloha obchodního cestujícího

S hamiltonovskou kružnicí souvisí úloha obchodního cestujícího. Jejím cílem je najít nejkratší trasu pro obchodního cestujícího, který vyjíždí z určitého počátečního místa, má navštívit jistá obchodní místa (každé jednou) a nakonec se má vrátit na původní místo, odkud vyjel. Vyjádříme-li tuto úlohu grafem, pak uzly v něm reprezentují výchozí místo a místa, která má obchodní cestující navštívit. Dále každé dva uzly v grafu jsou spojeny hranou, která je ohodnocená vzdáleností, kterou je zapotřebí urazit, abychom se dostali z místa reprezentovaného jedním z těchto dvou uzlů do místa, jež reprezentuje druhý z uzlů. Úloha je tedy zadána úplným hranově ohodnoceným grafem. V úplném grafu (s aspoň třemi uzly) lze vždy snadno najít hamiltonovskou kružnici. Cílem této úlohy je ale najít tu hamiltonovskou kružnici, jejíž délka, která je vyjádřena součtem ohodnocení hran v ní, je nejmenší ze všech hamiltonovský kružnic v grafu. Tato hamiltonovská kružnice je zároveň i nejkratší trasou pro obchodního cestujícího a je tedy řešením úlohy obchodního cestujícího.

I úloha obchodního cestujícího patří mezi NP-obtížné problémy, což znamená, že se pro ni používají především heuristické algoritmy dávající přibližné řešení. Těchto algoritmů existuje více, neboť úloha obchodního cestujícího je jednou z velmi důležitých grafových úloh.

Úloha obchodního cestujícího se řeší heuristickými algoritmy.

První popsaný algoritmus není heuristickým algoritmem ve smyslu, že by dával přibližný výsledek. Jeho negativní stránka je v tom, že někdy může skončit neúspěchem. Pokud ale nalezne řešení, je toto řešení optimálním řešením úlohy obchodního cestujícího.

Algoritmus je založen na skutečnosti, že hamiltonovská kružnice je specifickým případem tzv. 1-stromu, což je souvislý podgraf daného grafu, jež má právě jednu kružnici.

Definice. Mějme souvislý graf G. Jeho 1-strom je každý jeho prodgraf G', který splňuje podmínky:

- Obsahuje všechny uzly grafu G (je faktorem grafu G).
- Je souvislý.
- Obsahuje právě jednu kružnici.

Je-li navíc výchozí graf hranově G ohodnocený, pak jeho minimálním 1-strom je ten z jeho 1-stromů, který má nejnižší součet ohodnocení svých hran.

Na 1-strom se dá dívat jako na souvislý podgraf s cyklomatickým číslem 1. Dosadíme-li do vztahu pro jeho cyklomatické číslo

$$\mu(G') = |H'| - |U| + p$$

počet kružnic $\mu(G') = 1$ a počet komponent $p = 1$, vyjde nám, že 1-strom má právě tolik hran, kolik má uzlů.

Algoritmus vychází z úplného hranově ohodnoceného grafu G, který je zadáním úlohy obchodního cestujícího. Najde jeho minimální 1-strom G'. Pokud by tento minimální 1-strom byl kružnicí, je tím úloha vyřešena, neboť by to byla minimální hamiltonovská kružnice grafu G. Typicky tomu ale tak nebude. Proto algoritmus cíleně provádí změny ohodnocení hran tak, aby minimální 1-strom grafu G se stal kružnicí. Přitom tyto změny ohodnocení jsou dělány takovým způsobem, aby získaná kružnice byla zároveň i hamiltonovskou kružnicí, která má nejmenší ohodnocení z hamiltonovský kružnic ve výchozím grafu, tedy byla řešením úlohy obchodního cestujícího.

Popis algoritmu

1. V grafu G najdeme jeho minimální 1-strom G'. Můžeme k tomu použít již popsané algoritmy pro nalezení minimální kostry, které mírně modifikujeme. Vezmeme třeba zařazovací algoritmus. Tj. seřadíme hrany v neklesající posloupnost, začneme od diskrétního podgrafu a postupně je přidáváme tak, aby nevznikla více než jedna kružnice. Je-li nalezený minimální 1-strom zároveň i kružnicí, je úloha ukončena. Tato kružnice je shodná s hamiltonovskou kružnicí s nejmenším ohodnocením v původním grafu G.

   Jinak přejdeme ke kroku 2., kde uděláme změny v odhodnocení grafu v grafu G.

2. 1-strom je kružnicí právě když všechny uzly v něm mají stupeň 2. Vyjdeme z této skutečnosti a budeme se snažit u uzlů, které mají ve stávajícím minimálním 1-stromu stupeň jen 1, snížit ohodnocení hran, které s ním incidují, aby při následné nové konstrukci minimálního 1-stromu se do něho dostalo více hran s ním incidujících. A naopak u uzlů, které mají v současném minimálním 1-stromu stupeň větší než 2, zvýšíme ohodnocení hran, které s ním incidují, aby při další konstrukci 1-stromu provedené po této změně se už do něho nedostalo tolik hran incidujících s tímto uzlem. Nové ohodnocení hran grafu G stanovíme následujícím způsobem

   $$w'(h_{ij}) = w(h_{ij}) + t_i + t_j \quad ,$$

   kde h_{ij} je hrana spojující uzlu u_i a u_j, $w(h_{ij})$ je stávající ohodnocení hrany h_{ij}, $w'(h_{ij})$ je nové ohodnocení hrany h_{ij}, a t_i a t_j jsou korekční hodnoty přiřazené koncovým uzlům hrany u_i a u_j.

   Přičemž korekční hodnoty t_i vypočítáme pro každý uzel u_i grafu G podle zásad:

   - Je-li v současném minimálním 1-stromu stupeň uzlu u_i menší než 2 (nižší než požadovaný), hodnotu t_i stanovíme zápornou. Její vliv na hrany incidující s tímto uzlem bude takový, že sníží jejich ohodnocení.
   - Je-li v současném minimálním 1-stromu stupeň uzlu u_i roven 2, hodnota t_i bude 0, neboť tento uzel má požadovaný stupeň a není důvod měnit ohodnocení hran s ním incidujících.
   - Je-li v současném minimálním 1-stromu stupeň uzlu u_i větší než 2 (vyšší než požadovaný), hodnotu t_i stanovíme kladnou a to tím větší, čím větší je stupeň uzlu u_i v současném minimálním 1-stromu. Její vliv na hrany incidující s tímto uzlem bude takový, že zvýší jejich ohodnocení.

   Takovou funkcí, která počítá hodnoty t_i podle uvedených zásad, může být třeba funkce

   $$t_i = c * \left(d'(u_i) - 2\right) \quad ,$$

   kde c je kladná konstanta a $d'(u_i)$ je stupeň uzlu u_i v současném minimálním 1-stromu.

   Po provedení změny ohodnocení hran přejdeme opět k 1. kroku – najdeme nový minimální 1-strom.

Ověřme, že i přes změny ohodnocení hran v grafu G je kružnice nalezená předchozím algoritmem v 1-stromu G' hledanou hamiltonovskou s nejnižším ohodnocením i ve výchozím grafu s původním ohodnocením.

Nechť K_1, a K_2 jsou dvě hamiltonovské kružnice v grafu G. Označme $w(K_1)$ a $w(K_2)$ jejich ohodnocení v G. A nechť pro ně platí

$$w(K_1) \leq w(K_2) .$$

Po změně ohodnocení hran v G pomocí hodnot t_i se ohodnocení w kružnic změní na ohodnocení w', které je dáno vztahy:

$$w'(K_1) = w(K_1) + 2 * \left(t_1 + t_2 + \ldots + t_m\right)$$

$$w'(K_2) = w(K_2) + 2*(t_1 + t_2 + ... + t_m)$$

neboť každý uzel u_i na hamiltonovské kružnici inciduje se dvěma hranami a přispívá do celkového součtu změnou ohodnocení o t_i u každé z obou hran, tedy celkově mění ohodnocení kružnice o hodnotu $2*t_i$.

Platila-li před změnou ohodnocení nerovnost

$$w(K_1) \leq w(K_2) \ ,$$

její úpravou dostaneme

$$w(K_1) + 2*(t_1 + t_2 + ... + t_m) \leq w(K_2) + 2*(t_1 + t_2 + ... + t_m)$$

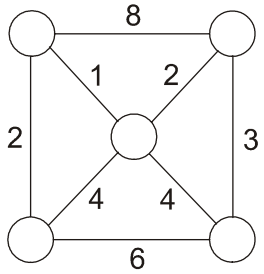
a tudíž po změně ohodnocení platí stejná nerovnost

$$w'(K_1) \leq w'(K_2) \ .$$

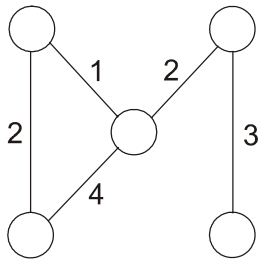
Z čehož plyne, že hamiltonovská kružnice nalezená minimálním 1-stromem v předchozím algoritmu je zároveň hamiltonovskou kružnicí, jejíž ohodnocení v původním grafu je menší nebo stejné než ohodnocení jakékoliv jiné hamiltonovské kružnice v něm, čímž je i hledanou hamiltonovskou kružnicí s nejnižším ohodnocením v původním grafu a tedy řešením úlohy.

O uvedeném algoritmu jsme se zmínili, že nemusí nalézt řešení. To se projeví tak, že výpočet nikdy neskončí. Proto pokud výpočet trvá neúměrně dlouho, ukončíme ho sami. Pak buďto můžeme tento algoritmus zkusit znovu s jinou funkcí pro výpočet hodnot t_i anebo pro vyřešení úlohy zvolíme jiný heuristický algoritmus.

Příklad. Mějme dán následující graf, ve kterém máme najít hamiltonovskou kružnici s nejnižším ohodnocením. Poznamenejme, že tento graf na rozdíl od grafu v zadání úlohy obchodního cestujícího není úplný. Pro vlastní algoritmus to ale nevadí.



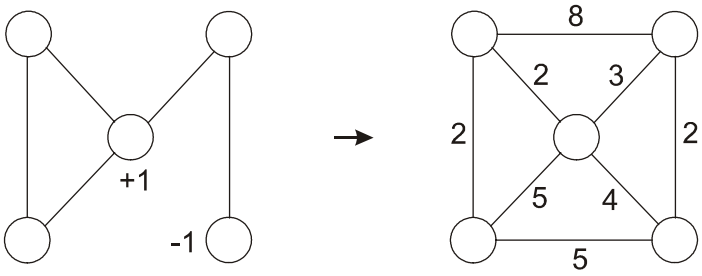
V grafu najdeme minimální 1-strom. Vyjdeme z diskrétního podgrafu a k němu postupně přidáme hrany s ohodnoceními 1, 2, 2 a 3. V této chvíli máme kostru. K ní přidáme jednu z hran s ohodnoceními 4. Zvolme pro přidání například levou z těchto hran.



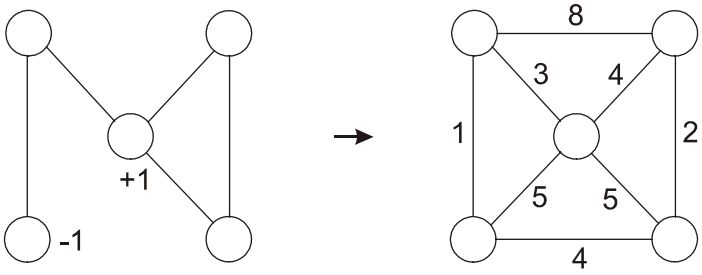
Použitím vytvořeného minimálního 1-stromu vypočítáme hodnoty t_i. Pro výpočet použijeme funkci

$$t_i = d'(u_i) - 2 \ .$$

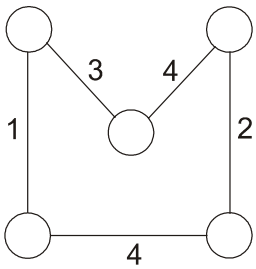
Vypočítané hodnoty jsou na dalším obrázku uvedeny u uzlů 1-stromu. Hodnoty, které vyjdou nulové, jsou pro přehlednost vynechány. Pomocí těchto hodnot nyní změníme ohodnocení původního grafu.



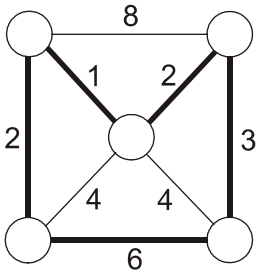
Následně po této změně ohodnocení najdeme nový minimální 1-strom. Opět vyjdeme z diskrétního podgrafu a přidáme k němu postupně hrany s ohodnoceními 2, 2, 2, 3 a 4. Tento 1-strom opět není kružnicí a znovu změníme ohodnocení grafu.



Znovu sestavíme minimální 1-strom přidáváním hran s ohodnoceními 1, 2, 3, 4 a 4. Ten už je hamiltonovskou kružnicí.



Vyznačíme ji na původním grafu, abychom viděli, že i v něm je hamiltonovskou kružnicí s nejmenším ohodnocením.



Další popisovaný algoritmus pracuje tak, že na začátku v grafu zvolíme malou kružnici. Pak k ní postupně přidáváme uzly, které v ní ještě nejsou, a to vždy na místo v kružnici, kde přidání zvýší co nejméně ohodnocení kružnice.

Popis algoritmu

1. Počáteční podmínky.

   Na začátku v grafu zvolíme malou kružnici (stačí se 3 hranami).

2. Průběžný krok.

   Nechť už je sestavena nějaká kružnice, která obsahuje uzly

   $u_{i1}, u_{i2}, \ldots, u_{ik}$.

Nechť u_r je uzel, který v této kružnici ještě není. Pokud tento uzel vložíme do kružnice mezi její dva první sousední uzly u_{i1} a u_{i2}, pak se ohodnocení kružnice změní o

$$\Delta_1 = w(h_{i1,r}) + w(h_{i2,r}) - w(h_{i1,i2})$$

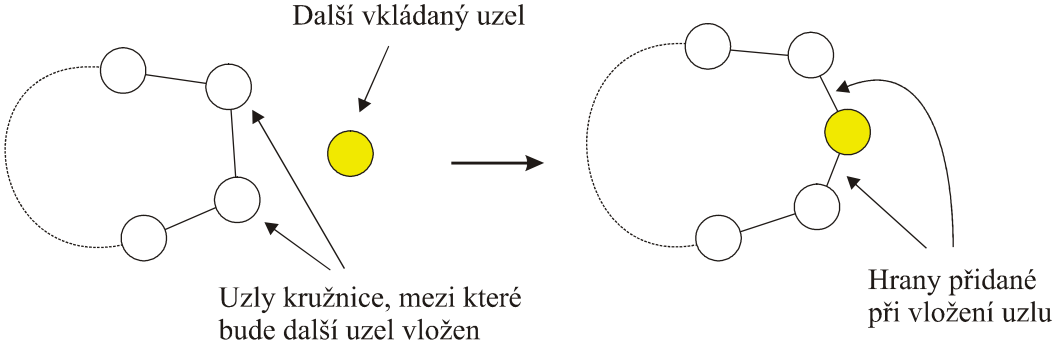
neboť jsem z kružnice odebrali hranu $h_{i1,i2}$ spojující uzly uzly u_{i1} a u_{i2} a přidali hranu $h_{i1,r}$ spojující vkládaný uzel u_r s uzlem u_{i1} a hranu $h_{i2,r}$ spojující uzel u_r s uzlem u_{i2}.

Podobně další možné pozice vložení mezi dva sousední uzly kružnice dostáváme změny ohodnocení kružnice

$$\Delta_2 = w(h_{i2,r}) + w(h_{i3,r}) - w(h_{i2,i3})$$

…

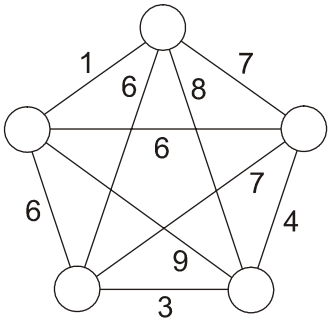
$$\Delta_k = w(h_{ik,r}) + w(h_{i1,r}) - w(h_{ik,i1})$$



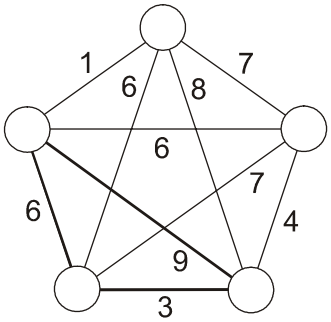
Ve vypočítaných hodnotách Δ_1, …, Δ_k najdeme nejmenší z nich a na místo v kružnici, které odpovídá nejmenší hodnotě, uzel u_r vložíme.

Průběžný krok opakujeme tak dlouho, dokud kružnice neobsahuje všechny uzly (nestane se hamiltonovskou kružnicí).

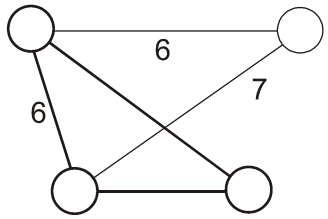
Příklad. Mějme dán následující graf, ve kterém máme najít hamiltonovskou kružnici s nejnižším ohodnocením.



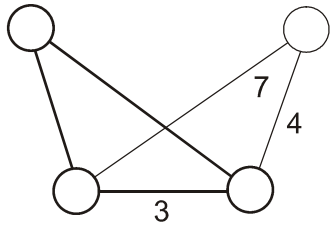
Nejprve zvolíme malou kružnici. Na dalším obrázku je vyznačena silnými čárami.



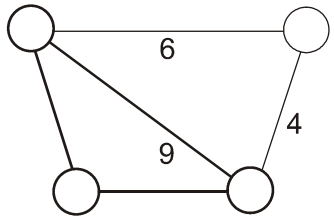
Budeme nyní vkládat uzel, který je na grafu nejvíce vpravo. Jsou možné tři pozice vložení do stávající kružnice.



$\Delta = 6+7-6 = 7$

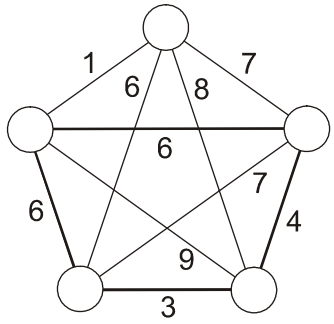


$\Delta = 7+4-3 = 8$

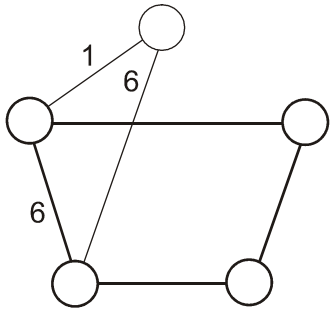


$\Delta = 6+4-9 = 1$

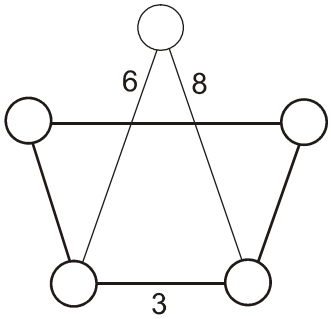
Je zřejmé, že nejpříznivější pozice vložení je ta poslední. Jí se stávající ohodnocení kružnice zvýší o 1.



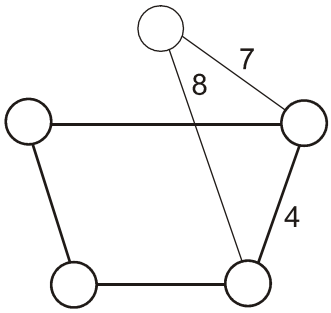
Zbývá do kružnice vložit poslední uzel. V kružnici jsou 4 možné pozice vložení.

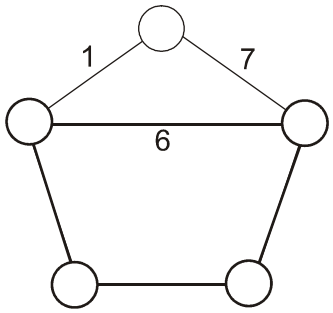


$\Delta = 1+6-6 = 1$



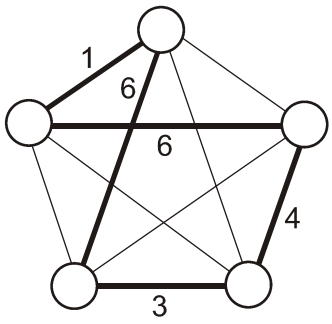
$\Delta = 6+8-3 = 11$





$\Delta = 8+7-4 = 11$ $\Delta = 1+7-6 = 2$

Je vidět optimální je první uvedená pozice vložení. Výsledná hamiltonovská kružnize je na dalším obrázku.

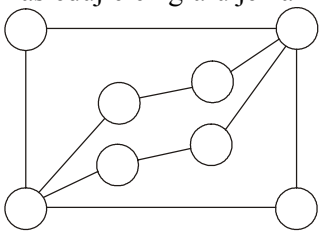


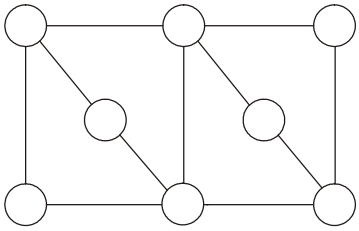
Kontrolní otázky

1. *Kdy je graf hamiltonovský?*
2. *Znáte nějakou postačující podmínku, aby graf byl hamiltonovský?*
3. *Jaký problém řeší úloha obchodního cestujícího?*

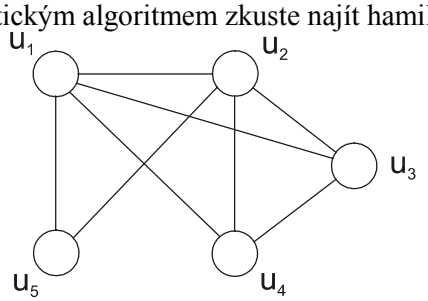
Cvičení

1. Který z následujících grafů je hamiltonovský?

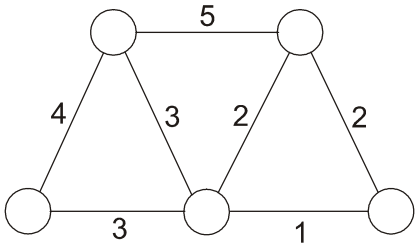




2. Heuristickým algoritmem zkuste najít hamiltonovskou kružnici v následujícím grafu.



3. Následující graf ukazuje místa, která musí navštívit obchodní cestující. Na grafu jsou vyznačeny jen spojující ulice, které jsou mezi místy, s jejich délkami. Graf doplňte na úplný graf vzdáleností mezi jednotlivými místy a řešte heuristickým algoritmem úlohu obchodního cestujícího.



Úkoly k textu

Zřejmě graf, jehož všechny uzly mají stupeň 2, je hamiltonovský. Je i hamiltonovský graf, jehož všechny uzly mají stejný stupeň, který je ale větší než 2?

Řešení

1. Žádný.
2. Jako počáteční uzel cesty zvolíme uzel u_1.

   Cesta = u_1.

   Existuje soused uzlu u_1 uzel u_2, který ještě v cestě není, přidáme ho k cestě.

   Cesta = $u_1\ u_2$. S - prázdná.

   Existuje soused uzlu u_2 uzel u_3, který ještě v cestě není, přidáme ho k cestě.

   Cesta = $u_1\ u_2\ u_3$. S - prázdná.

   Existuje soused uzlu u_3 uzel u_4, který ještě v cestě není, přidáme ho k cestě.

   Cesta = $u_1\ u_2\ u_3\ u_4$. S - prázdná.

   Zbývá uzel u_5, ten nelze přidat. Existuje hrana mezi uzlem cesty u_2 a koncovým uzlem u_4. Provedeme změnu cesty způsobem popsaným v algoritmu.

   Cesta = $u_1\ u_2\ u_4\ u_3$. $S = \{u_4\}$.

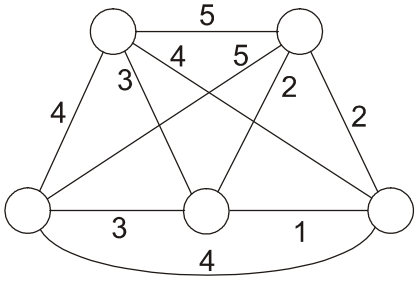
   Stále nelze přidat zbývající uzel u_5. Existuje hrana mezi uzlem cesty u_1 a koncovým uzlem u_3. Provedeme změnu cesty způsobem popsaným v algoritmu.

   Cesta = $u_1\ u_3\ u_4\ u_2$. $S = \{u_3, u_4\}$.

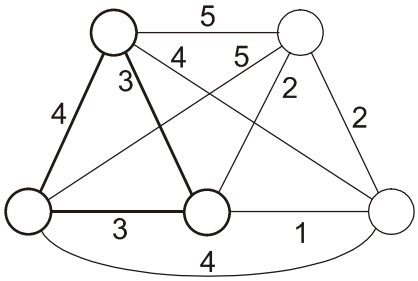
   Existuje soused uzlu u_2 uzel u_5, který ještě v cestě není, přidáme ho k cestě.

   Cesta = $u_1\ u_3\ u_4\ u_2\ u_5$. S - prázdná.

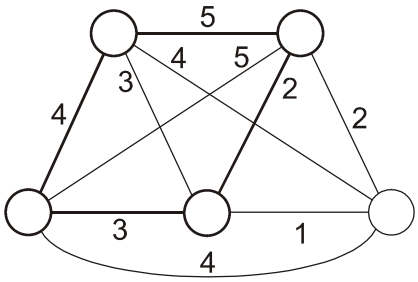
   Cesta již obsahuje všechny uzlu a existuje hrana mezi počátečním uzlem cesty u_1 a koncovým uzlem u_5. Cestu lze uzavřít v kružnici.
3. Úplný graf vzdáleností je na následujícím obrázku.



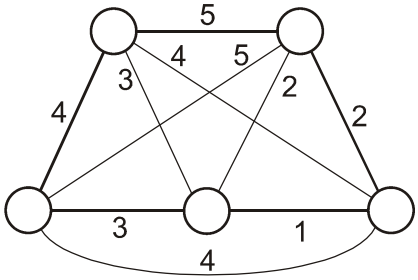
Úlohu budeme řešit druhým uvedeným heuristickým algoritmem, neboť tento je jednodušší. Zvolíme počáteční kružnici.



A budeme vkládat druhý uzel, který je nahoře. Jsou dvě možné pozice vložení s nejmenší nárůstem ohodnocení kružnice. Jednu z nich zvolíme.



Zbývá vložit dolní pravý uzel. Po vyhodnocení změn ohodnocení pro jednotlivé možnosti vložení dostáváme výsledek.



9 Párování

Studijní cíle: Seznámit studujícího s úlohou párování a algoritmem pro její řešení.

Klíčová slova: Párování, maximální párování, perfektní párování.

Potřebný čas: 4 hodiny

Párování řeší úlohu výběru dvojic objektů.

Další významnou grafovou úlohou je párování. Jejím cílem je vybrat dvojice (páry) ze zadané množiny objektů. Objekty tvoří uzly grafu. Hrany v grafu spojují ty dvojice objektů, které mohou tvořit páry. Najít párovaní znamená najít podmnožinu hran takovou, že hrany v ní jsou uzlově disjunktní – žádné dvě hrany v této podmnožině nemají společný žádný koncový uzel. Vybrané páry jsou tvořeny koncovými uzly vybraných hran, respektivně objekty, které tyto koncové uzly reprezentují.

Definice. Mějme dán graf $G = (H, U, \rho)$. Párování je podmnožina jeho množiny hran P taková, že žádné dvě hrany v P nemají společný žádný uzel.

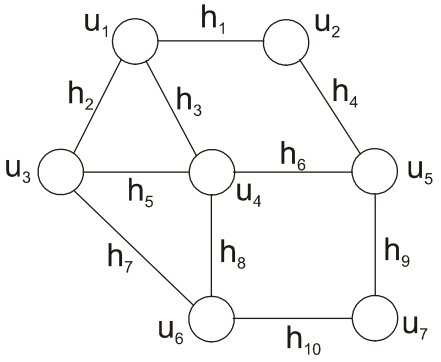
Maximální párování je takové, že obsahuje největší počet hran. Tedy v grafu neexistuje párování s větším počtem hran.

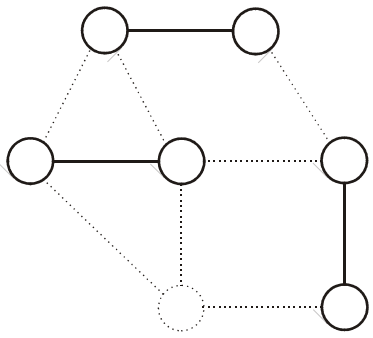
Definice. O uzlu říkáme, že je obsažen v párování P, nebo také se říká, že je nasycen párováním, jestliže je koncovým uzlem některé hrany obsažené v P.

Perfektní párování nasycuje všechny uzly grafu.

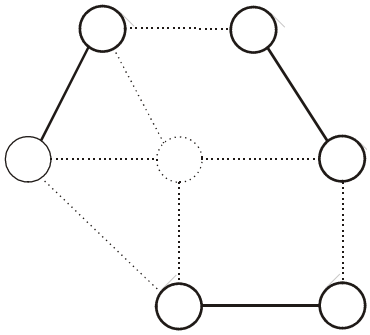
Perfektní párování je párování, které zahrnuje (nasycuje) všechny uzly grafu.

Příklad. Na následujícím obrázku je graf a vedle něho je vyznačené párování $P=\{h_1, h_5, h_9\}$, které je maximální a nasycuje uzly $u_1, u_2, u_3, u_4, u_5, u_7$.



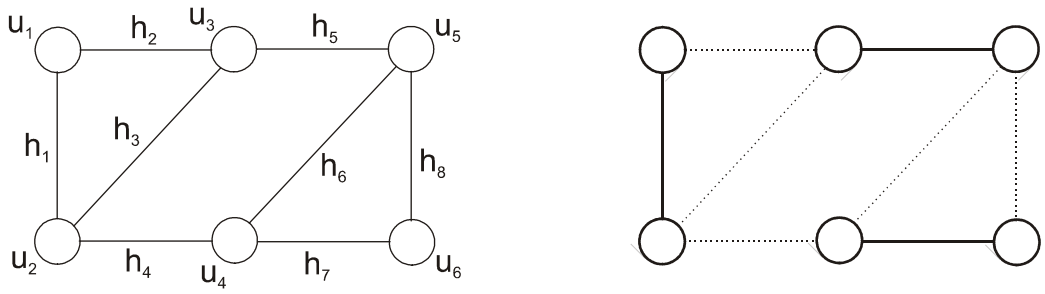


Na dalším obrázku je jiné maximální párování $P'=\{h_2, h_4, h_{10}\}$ stejného grafu.



Perfektní párování tento graf nemá a mít ani nemůže, neboť k němu je zapotřebí, aby graf měl sudý počet uzlů.

Příklad. Na dalším obrázku je graf a vedle něho je vyznačené perfektní párování $P=\{h_1, h_5, h_7\}$, které je maximální a nasycuje uzly $u_1, u_2, u_3, u_4, u_5, u_6$.

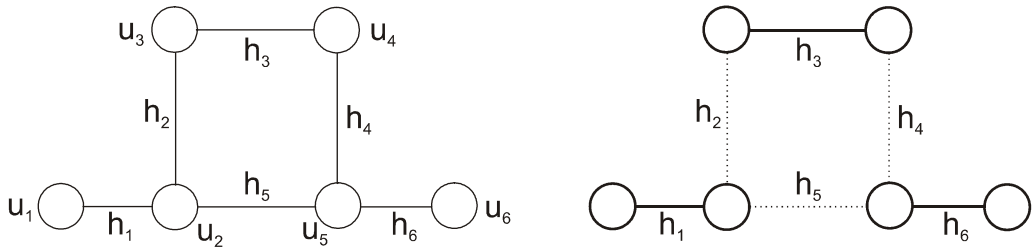


Definice. Mějme graf G a jeho párování P. Střídavá cesta v grafu G vzhledem k párování P je cesta s vlastnostmi:

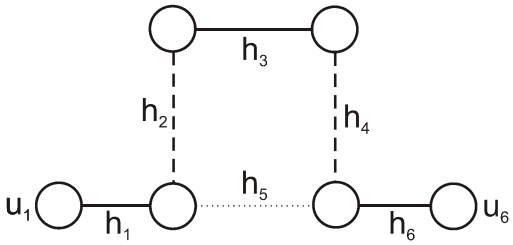
- Hrany cesty střídavě jsou a nejsou v párování P.
- Pro krajní uzly cesty platí, že je-li krajní uzel v párování P nasycen, pak hrana, která ho nasycuje, je součástí cesty.

Střídavá kružnice je uzavřená střídavá cesta.

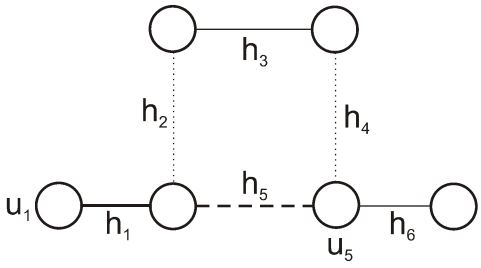
Příklad. Na dalším obrázku je graf a vedle něho je vyznačené párování $P=\{h_1, h_3, h_6\}$.



Cesta h_1, h_2, h_3, h_4, h_6 :

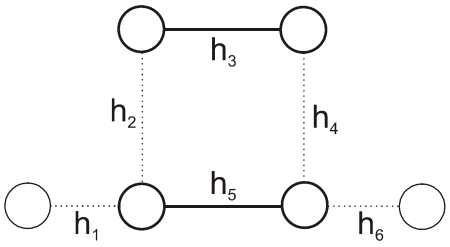


je střídavou cestou vzhledem k párování P. Oba její koncové uzly jsou nasycené a leží na cestě. Naproti tomu cesta h_1, h_5 :

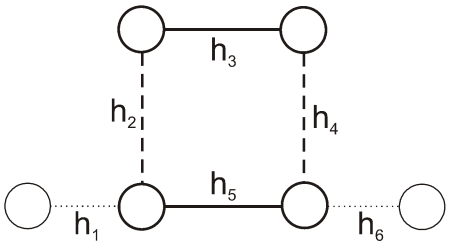


není střídavou cestou vzhledem k párování P, neboť její koncový uzel u_5 je nasycený v párování hranou h_6, která ale není součástí cesty.

Vezměme jiné párování $P'=\{h_3, h_5\}$.



Kružnice tvořená hranami h_2, h_3, h_4, h_5 :



Z jednoho párování lze dostat jiné pomocí střídavých cest.

je střídavou kružnicí vzhledem k párování P':

Věta. Nechť graf G má párování P. Pro každé jiné párování P' lze v grafu G lze najít soustavu střídavých cesta a kružnic vzhledem k párování P s vlastnostmi:

- Střídavé kružnice a cesty jsou disjunktní (žádné dvě z nich nemají společný ani jeden uzel).
- Změnou příslušnosti hran k párování v nich (hrany, které do párování patřily, nyní nebudou patřit a naopak hrany, které do párování nepatřily, nyní budou patřit) lze z párování P získat párování P'.

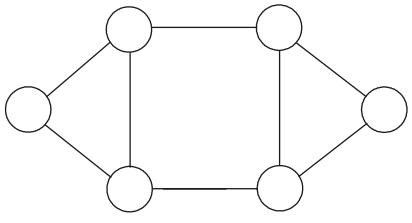
Důkaz: Uvažujme faktor G' grafu G (připomeňme, že faktor je podgraf zachovávající množinu uzlů) s množinou hran

$$(P - P') \cup (P' - P) .$$

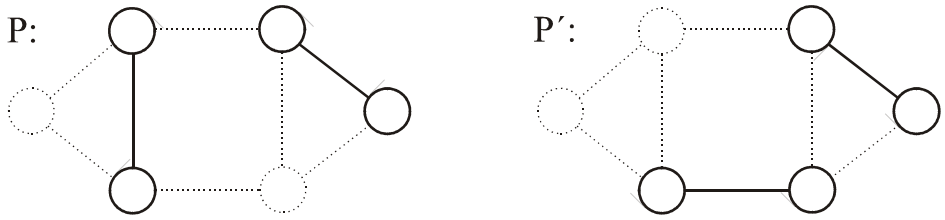
Pro uzly ve faktoru G' mohou nastat 3 případy:

1. Uzel je izolovaný. Jsou dvě situaci, kdy to může nastat.
   a) Uzel není nasycen ani v jednom z párování P a P'. Pak není obsažen ani v množině P-P' a ani v množině P'-P.
   b) Uzel je nasycen oběma párováními P a P' a to stejnou hranou. Pak tato hrana není ani v množině P-P' a ani v množině P'-P.
2. Uzel má stupeň 1. To nastane v případě, kdy uzel je nasycen právě jedním z obou párování P a P'. Pak je obsažen buďto v množině P-P' anebo v množině P-P'.
3. Uzel má stupeň 2. To nastane v případě, kdy uzel je nasycen oběma párováními P a P' a to různými hranami. Pak jedna z hran je obsažena v množině P-P', zatímco druhá hrana je v množině P-P'.

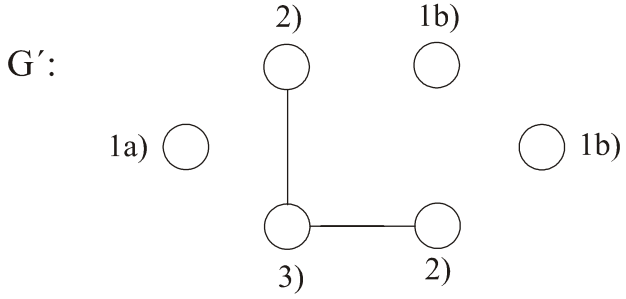
Uvažujme následující graf

G: 

a jeho dvě párování

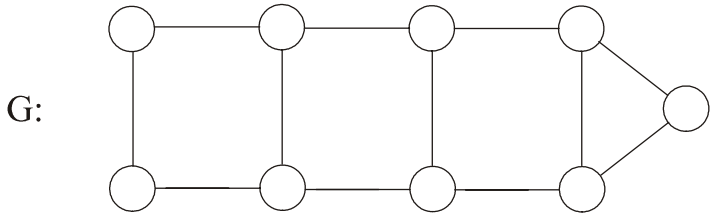


Na dalším obrázku je faktor grafu popsaný ve větě, ve kterém je u jednotlivých uzlů ukázáno, které případy popsané v tomto důkazu u nich nastaly.

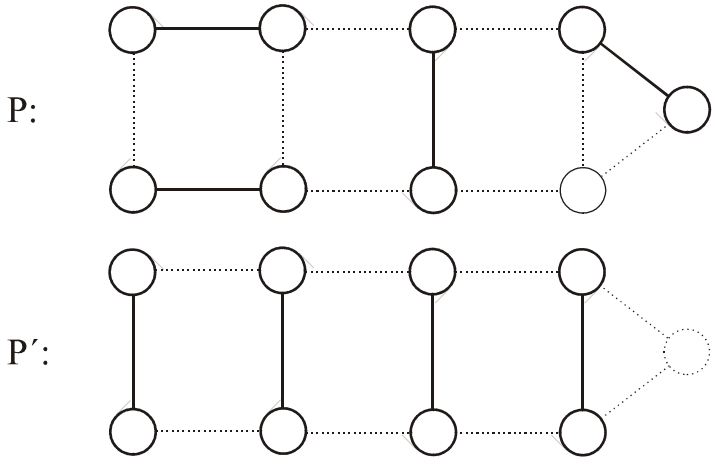


Nyní vezmeme jednotlivé komponenty faktoru *G′*, které nejsou izolovanými uzly. Ty jsou střídavými cestami nebo kružnicemi vzhledem k párování *P*. Záměnou příslušnosti hran k párování lze v nich přejít z párování *P* k párování *P′*.

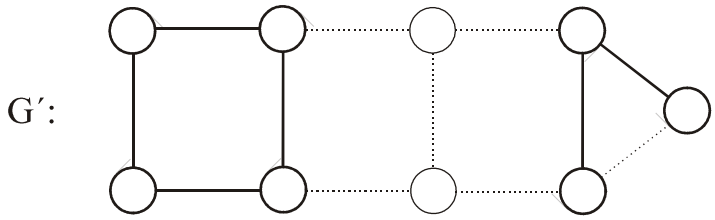
Příklad. Uvažujme graf



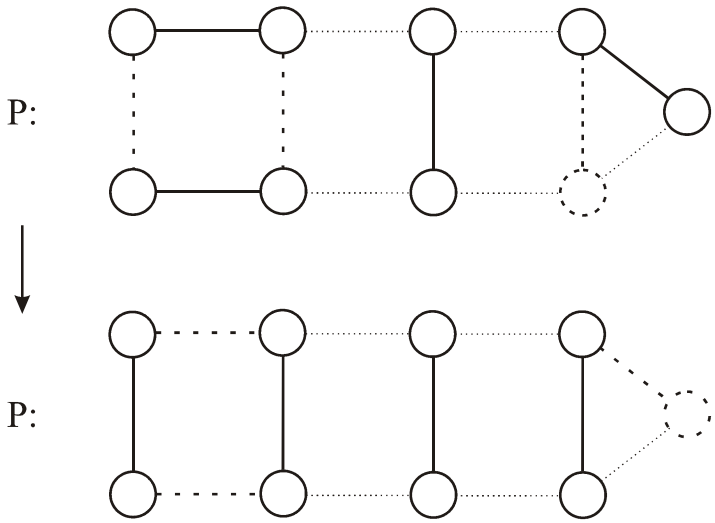
a jeho dvě párování



Faktor G′ popsaný v důkazu věty je



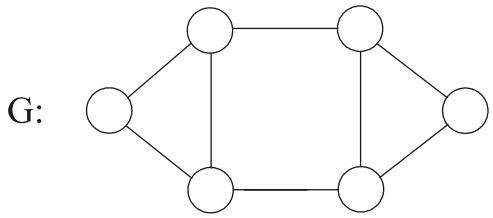
a obsahuje jednu kružnici a jednu cestu. Následující obrázek ukazuje, jak z nich vytvořit střídavou kružnici a střídavou cestu, u kterých záměnou příslušnosti hran lze přejít od párování P k párování P'.



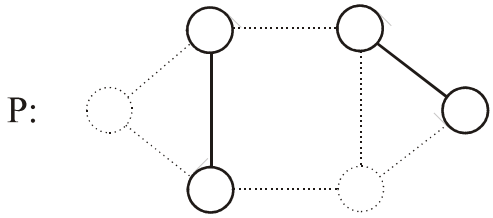
Věta. Párování P v grafu G je maximální právě když v grafu G není střídavá cesta vzhledem k P, jejíž oba koncové uzly jsou v párování P nenasycené.

Důkaz. Nechť taková střídavá cesta existuje. Nutně musí mít lichý počet hran. Přičemž počet těch hran v cestě, které do párování P nepatří, je 1 hranu větší, než počet hran v cestě, které do párování patří. Záměnou příslušnosti hran v cestě k párování dostaneme nové párování, které bude mít o 1 hranu více. Tedy párování P není maximální.

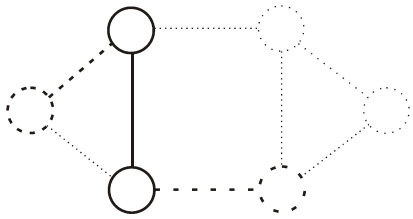
Vezměme graf



a jeho párování

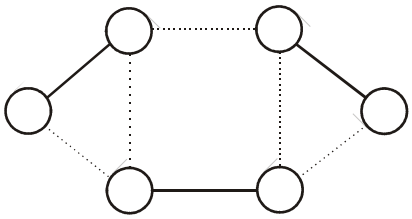


V grafu existuje střídavá cesta vzhledem k párování, jejíž oba koncové uzly jsou nenasycené, jak ukazuje další obrázek.



Záměnou příslušnosti hran k párování v této cestě dostaneme párování P', které má o jednu hranu více než párování P.

P′:



Naopak je-li párování P maximální, nemůže v grafu existovat střídavá cesta vzhledem k tomuto párování s oběma koncovými uzly nenasycenými, neboť záměnou příslušnosti hran k párování v ní bychom dostali párování větší, což by byl spor s tím, že párování P je maximální.

Na hledání střídavých cest je založen následující algoritmus pro nalezení maximálního párování.

Algoritmus pro nalezení maximálního párování je založen na hledání střídavých cest.

1. Počáteční podmínky.

   Je dán graf G, ve kterém hledáme maximální párování. Předpokládáme, že má nějaké hrany (není diskrétní).

   Množina hran párování P je na začátku prázdná.

2. Průběžný krok.

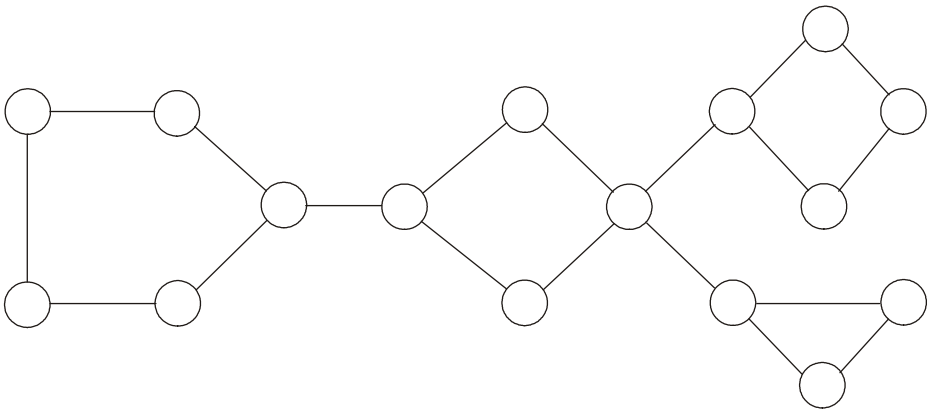
   V grafu najdeme uzel u, který není izolovaný (má stupeň aspoň 1). Vezmeme libovolnou hranu incidující s ním a začneme od uzlu u vytvářet střídavou cestu. Přičemž uzel u bude nenasyceným uzlem střídavé cesty. Cestu vytváříme tak dlouho, dokud k ní lze přidávat nějaké hrany.

   Když už nejde ke střídavé cestě přidat žádnou hranu, podíváme se, zda i její druhý koncový uzel není nenasycený. Jestliže ano, provedeme v cestě záměnu příslušnosti hran k párování (tím se počet hran patřících k párování zvýší o 1).

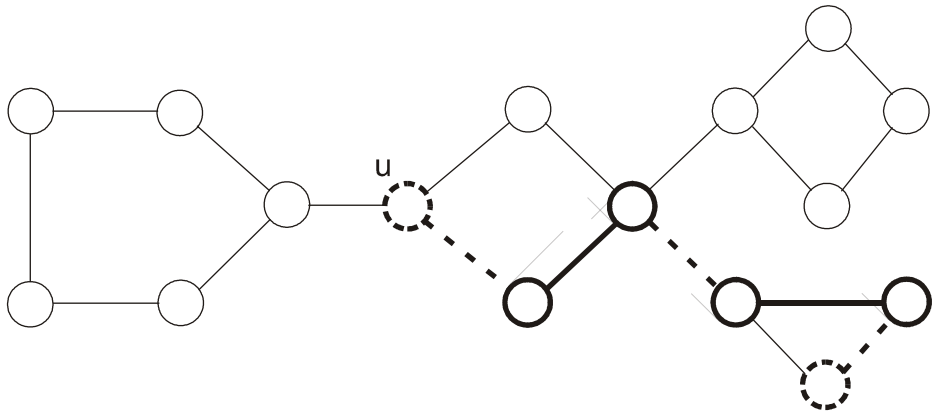
   Hrany na střídavé cestě, které přísluší k párování, přidáme k množině P. Následně z grafu G odstraníme všechny uzly patřící k hranám právě přidaným k množině P.

3. Krok 2. opakujeme tak dlouho, dokud v grafu zbývají ještě nějaké hrany. Po ukončení je množina P maximálním párováním ve výchozím grafu.

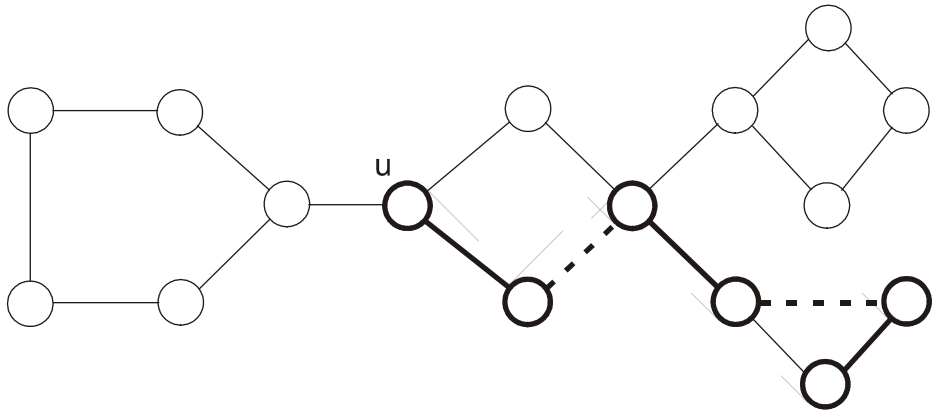
Příklad. Vezměme následující graf.



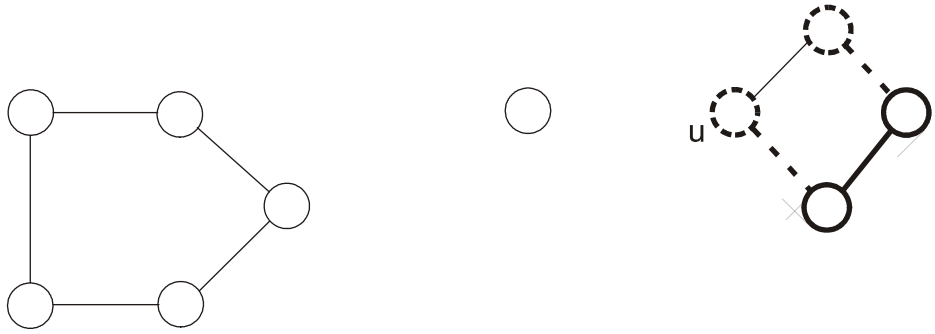
Zvolme jeden uzel jako počáteční uzel u a sestavme střídavou cestu. Třeba podle následujícího obrázku.



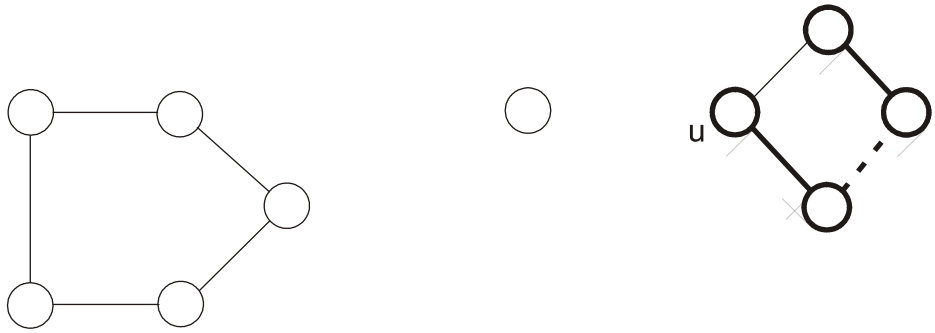
Oba koncové uzly střídavé cesty jsou nenasycené, proto provedeme záměnu příslušnosti hran hran k párování.



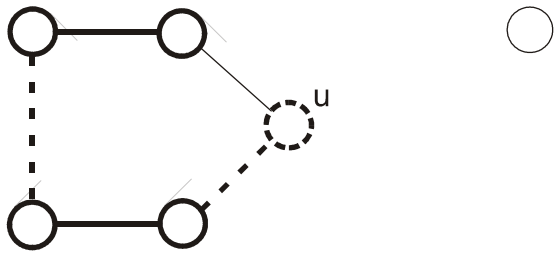
Hrany na střídavé cestě patřící k párování přidáme do sestavovaného párování a odstraníme z grafu. A zvolíme nový počáteční uzel *u* a sestavíme střídavou cestu. Třeba podle následujícího obrázku.



Opět oba koncové uzly střídavé cesty jsou nenasycené, proto provedeme na cestě záměnu příslušnosti hran k párování.



Hrany na střídavé cestě patřící k párování přidáme do sestavovaného párování a odstraníme z grafu. A opět zvolíme nový počáteční uzel *u* a sestavíme střídavou cestu. Třeba podle následujícího obrázku.



Zde druhý koncový uzel střídavé cesty není nenasycený, tudíž jen přidáme příslušné hrany obsažené na cestě do párování. Po jejich odstranění z grafu zbudou v grafu jen dva izolované uzly, čímž je vytváření maximálního párování ukončeno.

9.1 Úloha čínského pošťáka

S párováním souvisí úloha čínského pošťáka. Jejím cílem je najít nejkratší trasu pro pošťáka při doručování pošty v jeho obvodě. Předpokládáme, že pošťák vyjde z nějakého počátečního místa v obvodě (třeba pošty), při doručování pošty projde všechny ulice doručovacího obvodu a vrátí se zpět na místo, odkud vyšel.

Doručovací obvod je převeden do souvislého grafu *G*, ve kterém ulice jsou reprezentovány hranami, přičemž tyto hrany jsou ohodnoceny délkami příslušných ulic. Uzly jsou koncová místa ulic nebo místa, kde se ulice setkávají (křižovatky).

Cílem úlohy čínského pošťáka je najít uzavřený sled, který pokrývá celý graf (obsahuje všechny hrany grafu) a je z takovýchto sledů nejkratší. Přitom délka sledu je definována jako součet ohodnocení všech hran v něm. Nepochybně nejkratší možná trasa pro pošťáka je, když každou ulicí projde právě jednou. V grafové podobě to znamená, že by musel existovat uzavřený tah, který pokrývá celý graf. Tuhle otázku jsme již řešili a výsledkem bylo, že graf by musel být eulerovský. A graf je eulerovský, jestliže všechny jeho uzly mají sudý stupeň. To ovšem většinou splněno není, což znamená, že pošťák bude muset některými ulicemi projít dvakrát.

Při řešení tohoto problému vyjdeme ze skutečnosti, že problém v grafu způsobují uzly s lichým stupněm. Protože u lichého uzlu vždy některou z hran, s kterými uzel inciduje, musíme projít dvakrát. Uzlů s lichým stupněm je v grafu sudý počet, neboť součet stupňů všech uzlů grafu je sudé číslo. Problém, že každý uzel s lichým stupněm inciduje vždy s hranou, kterou je nutné projít dvakrát, vyřešíme tak, že uzly s lichými stupni rozdělíme do dvojic

Lichý uzel vždy inciduje s ulicí, kterou pošťák musí projít dvakrát.

$$(u_{i1},u_{j1}), (u_{i2},u_{j2}), \ldots, (u_{ik},u_{jk})$$

a po nejkratších cestách, jež spojují tyto dvojice uzlů, pošťák projde dvakrát. Takových rozdělení do dvojic je více. Budeme se snažit zvolit z nich to, u kterého je nejmenší součet vzdáleností mezi uzly v jednotlivých dvojicích, tedy nejmenší je součet

$$d(u_{i1},u_{j1}) + d(u_{i2},u_{j2}) + \ldots + d(u_{ik},u_{jk}) \quad ,$$

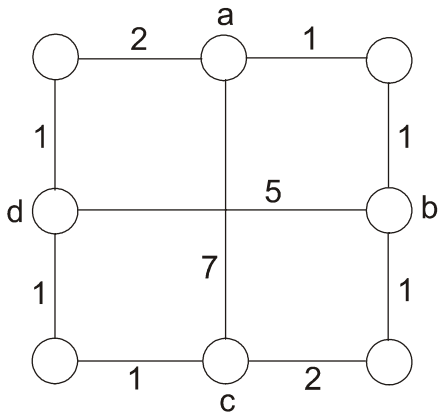
kde $d(u_{ir},u_{jr})$ označuje vzdálenost (délku nejkratší cesty) mezi uzly u_{ir} a u_{jr}.

Tuto úlohu lze řešit přes párování. Nejprve sestavíme pomocný graf G', jehož uzly budou uzly výchozího grafu, které mají lichý stupeň. Graf bude úplný, což znamená, že každá dvojice uzlů v něm bude spojena hranou. Tato hrana bude ohodnocena vzdáleností těchto uzlů ve výchozím grafu. K tomu se dají využít algoritmy pro hledání vzdáleností mezi uzly, které už jsme rovněž měli.

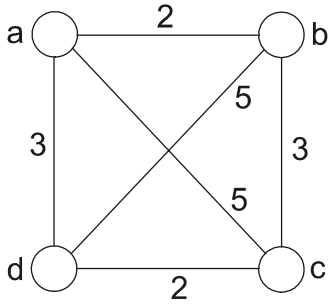
V takto sestaveném pomocném grafu G' vezmeme párování s nejmenším ohodnocením. To je takové perfektní párování, u něho součet ohodnocení všech jeho hran je ze všech perfektních

párování nejnižší. Nyní se vrátíme k původnímu grafu *G*. Vezmeme koncové uzly hran zvoleného párování. Cesty mezi páry uzlů v grafu G vyznačíme jako trasy, kterými pošťák musí projít dvakrát.

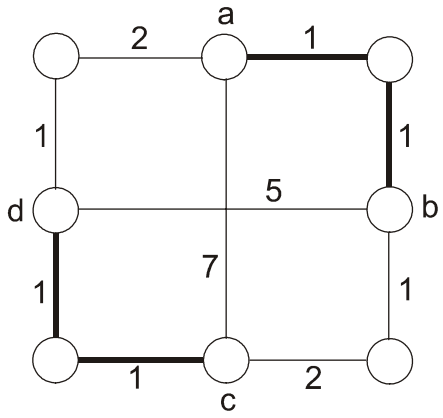
Příklad. Nechť doručovací obvod pošťáka je dán následujícím grafem.



Uzly s lichým stupněm jsou na grafu označeny písmeny. Sestavíme pomocný graf vzdáleností mezi uzly s lichým stupněm.



Je zřejmé, že perfektní párování s nejmenším ohodnocením zde tvoří hrana s koncovými uzly *a*, *b* a hrana s koncovými uzly *c*, *d*. Nejkratší cesty spojující koncové uzly těchto hran vyznačíme na původním grafu jako ulice, které je nutno projít dvakrát.

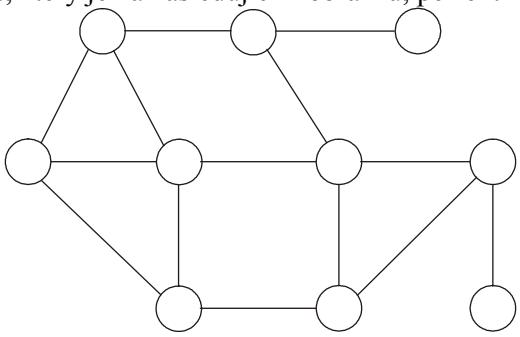


Kontrolní otázky

1. *Které párování se označuje jako maximální a které jako perfektní?*
2. *Jak zjistíme, zda párování je maximální?*
3. *Jaký problém řeší úloha čínského pošťáka?*

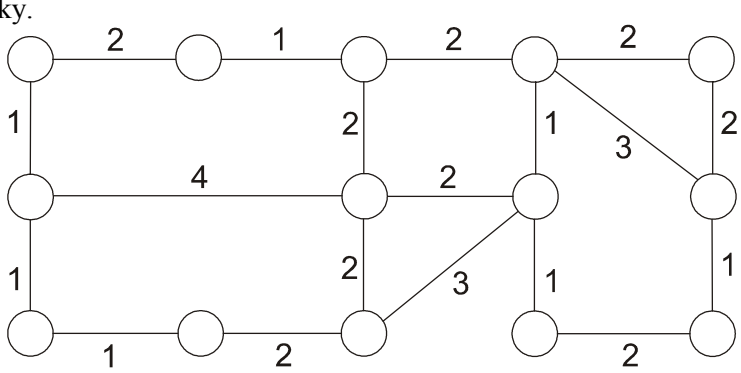
Cvičení

1. Existuje v grafu, který je na následujícím obrázku, perfektní párování?



14.

2. Najděte algoritmem uvedeným v této kapitole maximální párování v grafu z 1. cvičení.
3. Na následujícím grafu je doručovací obvod pošťáka. Najděte pro něho nejkratší trasu obchůzky.



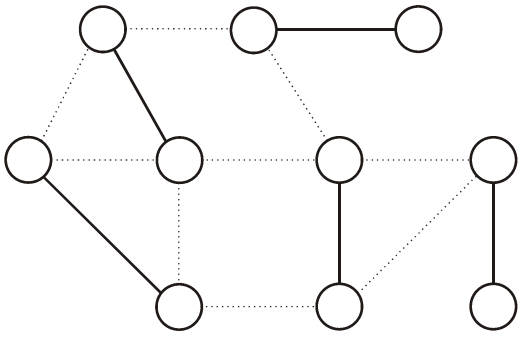
15.

Úkoly k textu

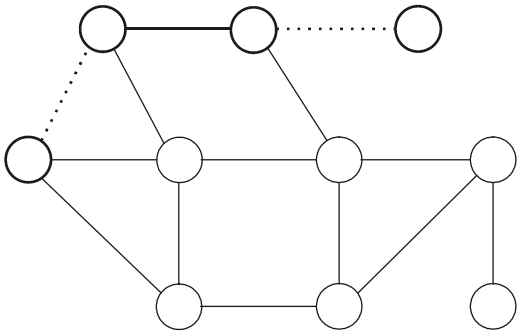
Tvrdím, že v souvislém grafu, v němž všechny uzly mají sudý stupeň, vždy existuje perfektní párování. Zjistěte, zda mám pravdu.

Řešení

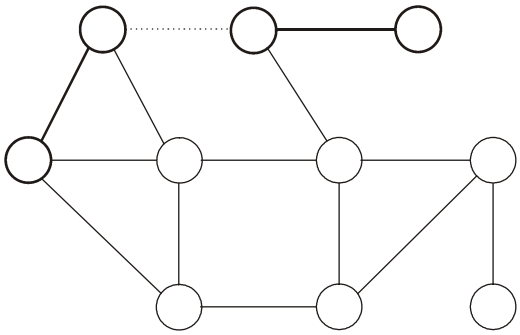
1. Existuje. Je vyznačeno na následujícím obrázku.



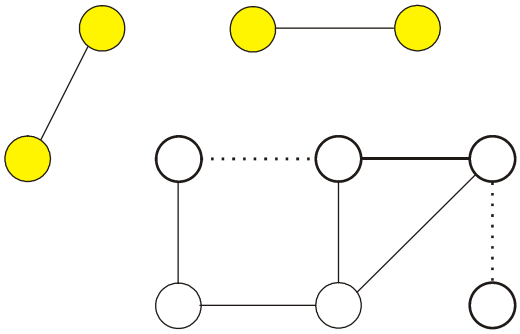
2. Začneme od nejlevějšího uzlu a najdeme první střídavou cestu.



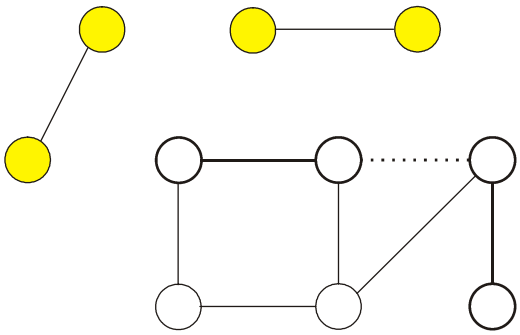
Oba její koncové uzly jsou nenasycené. Provedeme na cestě záměnu příslušnosti hran k párování.



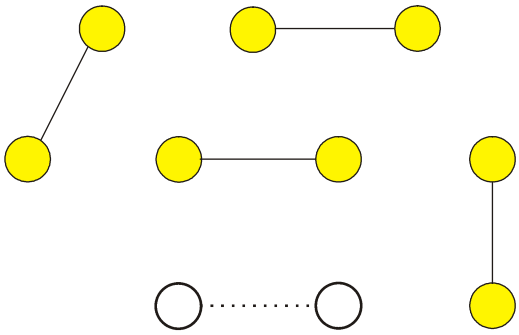
A hledáme další střídavou cestu.



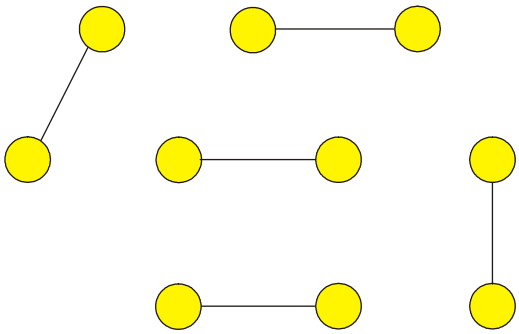
Opět jsou oba dva koncové uzly nenasycené. Provedeme záměnu příslušnosti hran k párování.



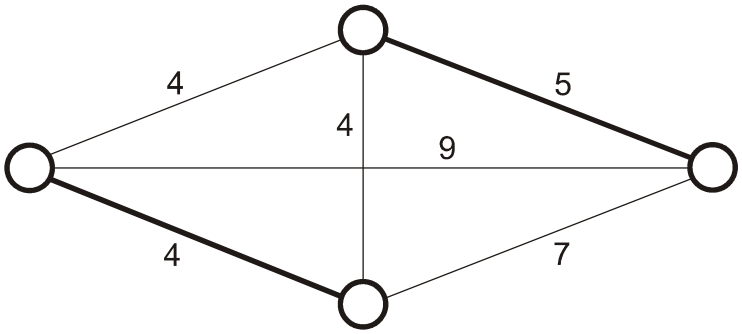
A hledáme poslední střídavou cestu.



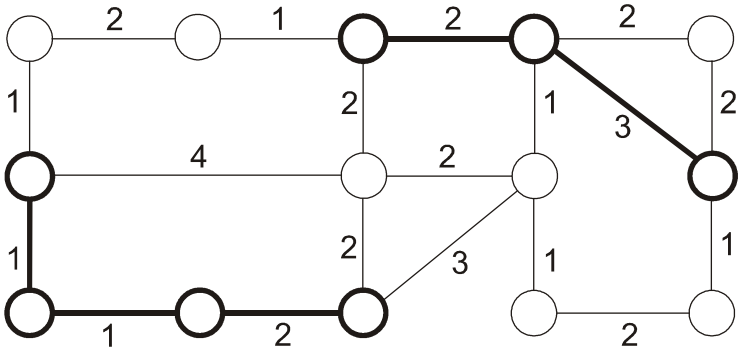
Opět její oba dva koncové uzly jsou nenasycené. Záměnou příslušnosti hrany k párování dostaneme výsledné párování.



3. Sestavíme graf vzdáleností mezi lichými uzly grafu doručovacího obvodu a najdeme v něm perfektní párování s nejmenším ohodnocením. Sestavený graf a vybrané párování ukazuje následující obrázek.



Ulice, které pošťák musí projít dvakrát, jsou vyznačeny silně ně následující grafu.



10 Planární grafy

Studijní cíle: Seznámit studujícího s vlastnostmi planárních grafů a algoritmem, jak nakreslit rovinný diagram planárního grafu.

Klíčová slova: Planární graf.

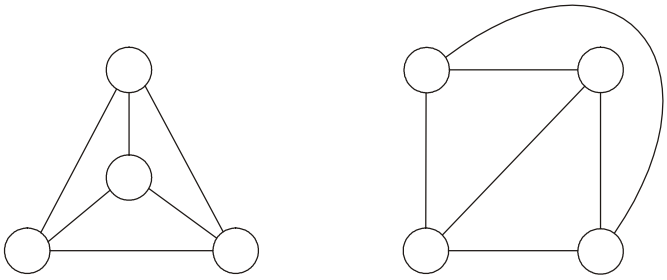
Potřebný čas: 3 hodiny

Planární graf lze nakreslit v rovině bez protínání hran.

Na začátku jsme zavedli pojem diagram grafu označující konkrétní nakreslení grafu. Zatím jsme tento pojem nepoužívali, protože způsob nakreslení grafu nebyl podstatný. Nyní se dostáváme k části, která se zabývá určitým způsobem nakreslení grafu, a to rovinným nakreslením grafu.

Definice. Graf nazveme planární (rovinný), jestliže existuje jeho nakreslení v rovině (diagram grafu) takové, že žádné dvě hrany se v něm neprotínají.

Příklad. Úplný graf se 4 uzly je planární, neboť ho lze v rovině nakreslit bez protínání jeho hran. Dvě možná taková nakreslení ukazuje následující obrázek.



Zřejmě nesouvislý graf je planární, jestliže jsou planární všechny jeho komponenty. Tudíž stačí se zabývat planárností jen souvislých grafů (jednotlivých komponent u nesouvislých grafů).

Planární nakreslení grafu rozděluje rovinu na stěny.

Graf v planárním nakreslení rozděluje rovinu na části, kterým říkáme stěny. Jednak jsou to stěny vnitřní, které jsou ohraničeny kružnicemi grafu (má-li nějaké), a pak jedna stěna vnější, což je zbývající, nekonečná část roviny.

Kružnice, které tvoří hranice vnitřních stěn, viditelně tvoří fundamentální systém kružnic grafu. Jejich počet je dán cyklomatickým číslem grafu. Tudíž celkový počet stěn grafu G, včetně vnější stěny, je

$$r = \mu(G) + 1 \quad .$$

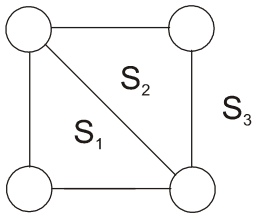
Dosadíme-li vztah pro cyklomatické číslo souvislého grafu

$$\mu(G) = |H| - |U| + 1 \quad ,$$

dostaneme pro počet stěn vztah

$$r = |H| - |U| + 2 \quad .$$

Příklad. Následující planární graf má dvě vnitřní stěny, jsou označeny S_1 a S_2. Vnější stěna je označena S_3.



Počet stěn v rovinném nakreslení grafu je podle předchozího vztahu r=5-4+2=3 .

Zřejmě čím bude mít graf při daném počtu uzlů více hran, tím obtížnější bude jeho rovinné nakreslení. Následující věta ukazuje, že pokud počet hran překročí určitou mez, graf nelze nakreslit rovinným způsobem.

Překročí-li počet hran v grafu určitou mez, graf nemůže být planární.

Věta. Pro souvislý planární graf G = (H, U, ρ) s aspoň třemi uzly platí vztah

$$|H| \le 3|U| - 6 \quad .$$

Důkaz: Vezměme nejprve případ, kdy graf nemá kružnice. Pro něj platí vztah

$$|H| = |U| - 1 \quad .$$

Dosazením do vztahu v tvrzení věty máme

$$|U| - 1 \le 3|U| - 6$$

a úpravou

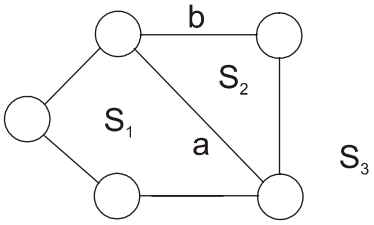
$$5 \le 2|U| \quad ,$$

což je vzhledem k předpokladu věty (|U|≥3) splněno.

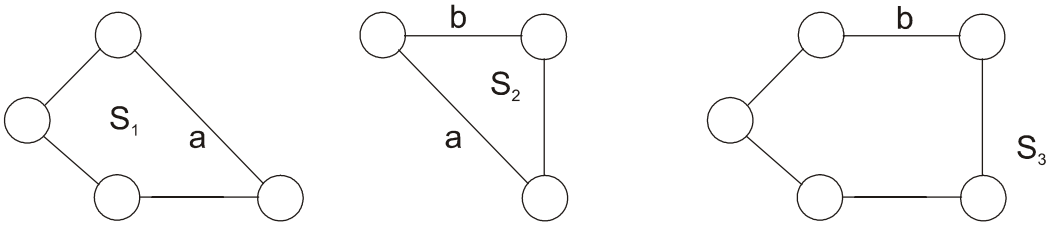
U grafu s kružnicemi hranici každé stěny (vnitřní i vnější) tvoří kružnice. Přitom

- kružnice má nejméně tři hrany
- každá hrana takové kružnici je obsažena současně v hranicích dvou stěn

Například graf



má tři hranice stěn



Je zřejmé, že hrana *a* je současně v hranicích stěn S_1 a S_2, zatímco hrana *b* je současně v hranicích stěn S_2 a S_3.

Odtud pro počet stěn dostáváme vztah

$$r \le 2\frac{|H|}{3} \quad .$$

Dosazením již odvozeného vztahu pro počet stěn

$$r = |H| - |U| + 2$$

dostáváme

$$|H| - |U| + 2 \leq 2\frac{|H|}{3} \quad .$$

A jeho úpravami

$$3|H| - 3|U| + 6 \leq 2|H|$$

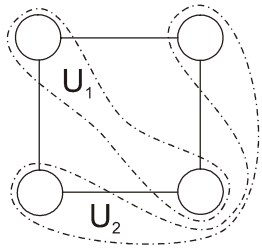
$$|H| \leq 3|U| - 6 \quad .$$

Zřejmě je-li graf planární, pak je i planární každý jeho podgraf. Stačí vzít rovinné nakreslení výchozího grafu a pak v něm odstranit hrany a uzly, které do podgrafu nepatří. Nebo naopak je-li nějaký podgraf daného grafu neplanární, pak není planární ani tento graf. Na tom je založena věta, která říká, že neplanární graf vždy obsahuje jako podgraf jeden ze dvou základních neplanárních grafů. Jeden z těchto základních neplanárních grafů je bipartitní graf. Nejprve definice bipartitního grafu.

Definice. Graf G = (H, U, ρ) je bipartitní, jestliže existuje rozklad jeho množiny hran na dvě podmnožiny U_1 a U_2 takový, že žádné dva uzly patřící do stejné podmnožiny rozkladu nejsou sousední.

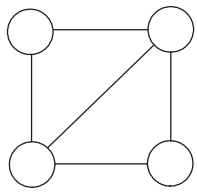
Bipartitní graf je úplný bipartitní graf, jestliže každé dva uzly, které nejsou ve stejné podmnožině rozkladu, jsou sousední.

Příklad. Následující graf je úplný bipartitní graf.



Bipartitní grafy mají poměrně jednoduché vlastnosti. Například chromatické číslo bipartitního grafu není větší než 2. Neboť všechny uzly ve stejné množině rozkladu lze obarvit stejnou barvou, protože žádné dva z nich nejsou sousední. Kružnice v bipartitním grafu mají vždy sudou délku (sudý počet hran).

Příklad. Následující graf není bipartitní – obsahuje kružnici liché délky.



Věta. Graf *G* je planární právě když neobsahuje podgraf *G´* homeomorfní s některým z následujících dvou grafů:

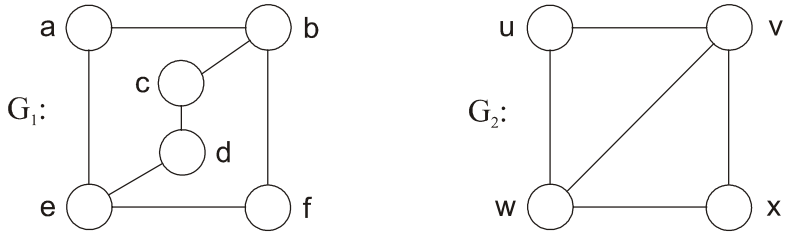
- Úplný graf s 5 uzly
- Úplný bipartitní graf s 3+3 uzly

Důkaz: Není jednoduchý…

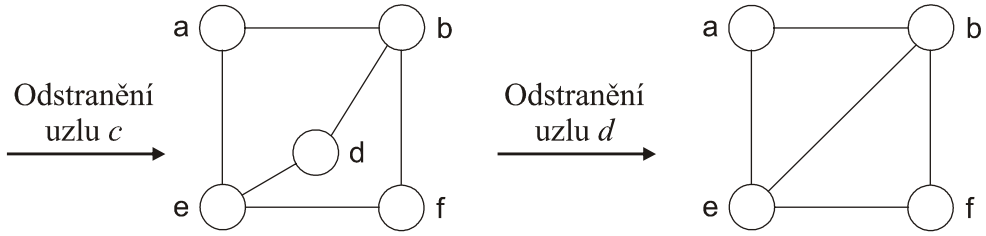
Homeomorfismus uvedený ve větě znamená;, že podgraf G' je buďto izomorfní s některým z uvedených úplných grafů anebo z něho některý izomorfní graf dostaneme postupným odstraňováním všech uzlů stupně 2, přičemž u každého odstraněného uzlu dvě hrany, které s ním incidují, spojíme v jednu hranu.

Izomorfismus dvou grafů znamená, že tyto grafy jsou až na označení stejné (lze je nakreslit stejným diagramem).

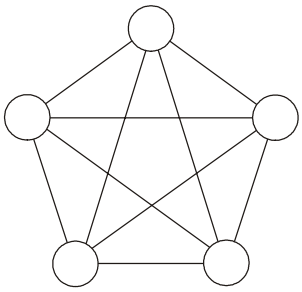
Příklad. Následující dva grafy jsou homoeomorfní.



Z grafu G_1 postupně odstraníme uzly c a d a dostaneme graf, který je viditelně až na označení stejný jako graf G_2:

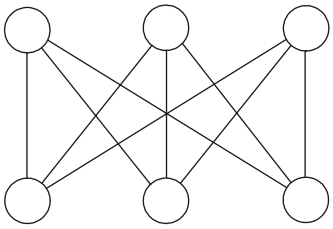


Úplný graf s 5 uzly uvedený v předchozí větě je na následujícím obrázku.



Počet uzlů a hran je v něm |U|=5 a |H|=10. Není v něm splněna nutná podmínka planárnosti odvozená na začátku této kapitoly $|H| \leq 3|U| - 6$ a tudíž planární tento graf ani nemůže být.

Úplný bipartitní graf s 3+3 uzly je na dalším obrázku.



V něm je |U|=6 a |H|=9 a je tedy v něm splněn vztah $|H| \leq 3|U| - 6$. Přesto ale tento graf planární není. Z toho plyne, že uvedená podmínka planárnosti je jen nutná, nikoliv postačující.

Abychom se přesvědčili, že tento graf skutečně planární není, odvodíme si nutnou podmínku planárnosti pro bipartitní grafy.

Věta. Pro souvislý planární bipartitní graf G = (H, U, ρ) s aspoň dvěma uzly platí vztah

$$|H| \le 2|U| - 4 \quad .$$

Důkaz. Bipartitní graf má všechny kružnice sudé délky. Tudíž každá kružnice, která je hranicí stěny planárního bipartitního grafu, má nejméně 4 hrany. Odtud pro počet stěn dostáváme obdobnou úvahu, jakou jsme měli v předminulé větě, vztah

$$r \le 2\frac{|H|}{4} \quad .$$

Dosazením vztahu pro počet stěn

$$r = |H| - |U| + 2$$

dostáváme

$$|H| - |U| + 2 \le 2\frac{|H|}{4} \quad .$$

A jeho úpravami

$$2|H| - 2|U| + 4 \le |H|$$

$$|H| \le 2|U| - 4 \quad .$$

Tento vztah v úplném bipartitním grafu s 3+3 uzly, v němž počet uzlů a hran je |U|=6 a |H|=9, viditelně splněn není. Tím máme ověřeno, že ani druhý z grafů uvedený ve větě o planárnosti není planární.

Použít předchozí větu ke zjišťování planárnosti grafu tím způsobem, že bychom hledali, zda neobsahuje podgraf homeomorfní s některým ze dvou neplanárních grafů uvedených ve větě, by bylo velmi pracné. Existují algoritmy, které buďto přímo graf rovinným způsobem nakreslí anebo zjistí, že graf planární není.

Následující popisovaný algoritmus sice není nejrychlejší z nich, zato je ale poměrně jednoduchý. Algoritmus začíná od určitého podgrafu *H* kresleného grafu *G* a postupně vykresluje další a další hrany až buďto celý graf rovinně nakreslí anebo zjistí, že graf není planární.

Pojmy použité v algoritmu.

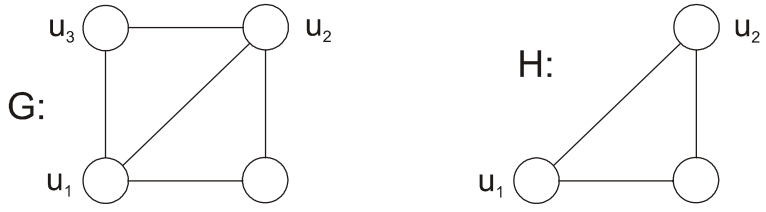
Pro nakreslení planárního diagramu grafu existují účinné algoritmy.

G v algoritmu označuje výchozí graf, *H* označuje jeho již nakreslenou část – podgraf.

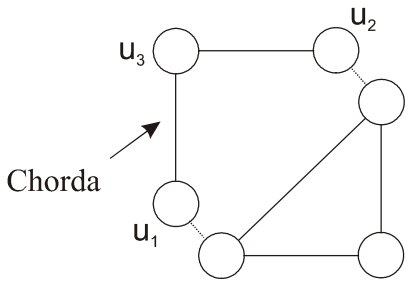
Chorda v *G* vzhledem k podgrafu *H* je cesta v *G* s vlastnostmi

- Koncové uzly cesty jsou v *H*.
- Žádný další uzel cesty a žádná hrana cesty nepatří do *H*.

Na následujícím obrázku je graf *G* a jeho podgraf *H*.



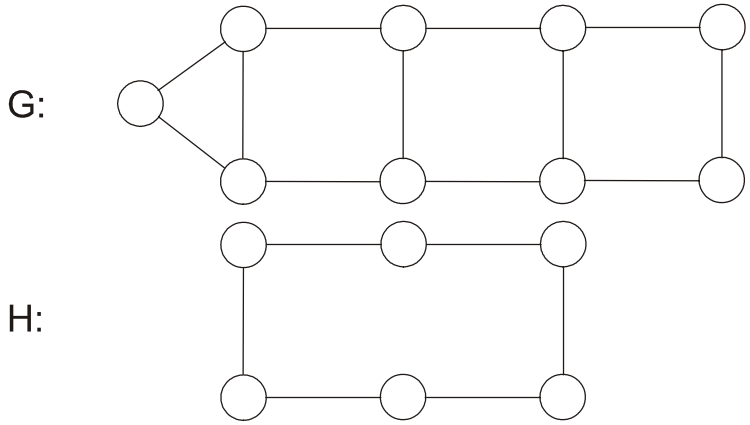
Další obrázek ukazuje chordu v *G* vzhledem k *H* (je tvořena cestou danou uzly u_1, u_2, u_3).



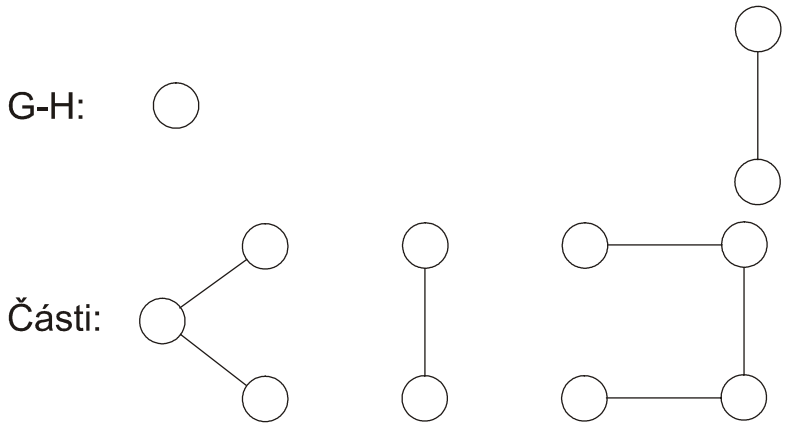
Část grafu *G* vzhledem k podgrafu *H* je

- Hrana, která do *H* nepatří, ale její koncové uzly jsou v *H*.
- Nebo je to komponenta grafu *G-H* (rozdílu grafů *G* a *H*, kde rozdíl dostaneme odebráním *H* z *G*), ke které jsou přidány všechny hrany, jejichž jeden uzel je v komponentě a druhý uzel je v *H*.

Na následujících obrázcích je graf *G* a jeho podgraf *H*.



Na dalších obrázcích je rozdíl *G-H* (2 komponenty) a části grafu *G* vzhledem k *H*.



Části jsou celkem tři. Dvě z nich (ty krajní) vznikly přidáním hran ke komponentám rozdílu *G-H*, které je spojují s podgrafem *H*. Prostřední část je tvořena hranou, jejíž oba koncové uzly patří do *H*, ale hrana sama do *H* nepatří.

Popis algoritmu

1. Počáteční krok.

   Je dán výchozí graf *G*, který chceme rovinně nakreslit. Předpokládáme, že je souvislý a jsou v něm nějaké kružnice. (Nakreslit rovinně graf bez kružnic je velmi snadné, na to není zapotřebí žádný algoritmus.)

   Jako podgraf *H* zvolíme na začátku libovolnou kružnici grafu *G*.

2. Průběžný krok.

   Nechť C_1, C_2, …, C_p jsou části grafu G vzhledem k H.

   Dále nechť S_1, S_2, …, S_r jsou stěny v planárně nakresleném podgrafu H (včetně vnější stěny).

   U všech částí C_1, C_2, …, C_p ověříme, do kterých stěn v podgrafu H je lze nakreslit.

   Mohou nastat případy:

   a) Existuje část C_i, kterou nelze nakreslit do žádné stěny. Algoritmus zde končí. Zjistil, že graf G není planární.

   b) Existuje část C_i, kterou lze nakreslit jen do jedné ze stěn, označme tuto stěnu S_k. Má-li část s již nakresleným podgrafem H společné aspoň dva uzly, vybereme v části C_i nějakou chordu vzhledem k H a tu nakreslíme do stěny S_k. Má-li část C_i s podgrafem H společný jen jeden uzel, najdeme v ní cestu, jejíž jeden koncový uzel je tímto společným uzlem, a nakreslíme ji do stěny S_k.

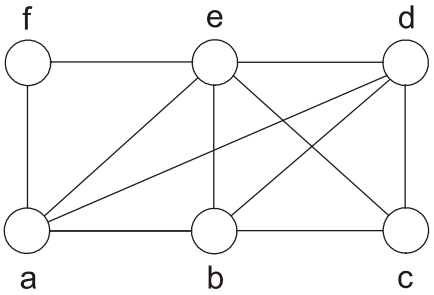
   c) Pro všechny části platí, že je lze nakreslit do více než jedné stěny. Vybereme libovolnou z částí, označme ji C_i, a vybereme jednu ze stěn, do kterých lze tuto část nakreslit, označme ji S_k. Opět v části C_i vybereme nějakou chordu vzhledem k H a nakreslíme ji do stěny S_k, má-li část s podgrafem H společné aspoň dva uzly, anebo vybereme v části C_i cestu s koncovým uzlem v H a zakreslíme ji do S_k, má-li část s podgrafem H společný jen jeden uzel.

3. Krok 2. opakujeme tak dlouho, dokud se buďto nezjistí, že graf není planární anebo celý graf G není rovinně nakreslen, tj. dokud podgraf H není celým grafem G.

Zbývá dořešit, jak zjistit, do kterých stěn lze danou část nakreslit. Stačí k tomu vzít množinu stykových uzlů části, tj. těch uzlů části, které patří do H, a vzít množiny uzlů stěn, což jsou uzly ležící na hraničních kružnicích jednotlivých stěn anebo uvnitř těchto stěn. Jestliže množina stykových uzlů je podmnožinou množiny uzlů dané stěny, pak část lze do této stěny nakreslit.

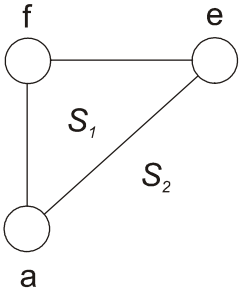
Příklad. Následující graf G máme nakreslit rovinným způsobem.

G:



Jako počáteční podgraf H zvolíme kružnici, která je v grafu G vlevo nahoře. Tato kružnice rozděluje rovinu na dvě stěny S_1 a S_2. Hranice těchto stěn obsahují uzly a, e, f.

H:

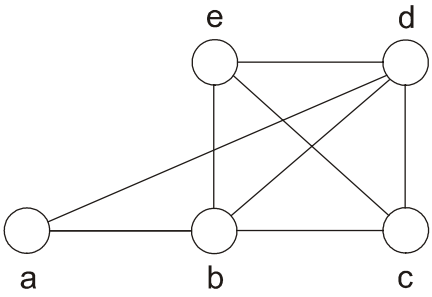


S_1: {a,e,f}

S_2: {a,e,f}

Graf G obsahuje jednu část vzhledem k H. Její stykové uzly (uzly společné s H) jsou a a e.

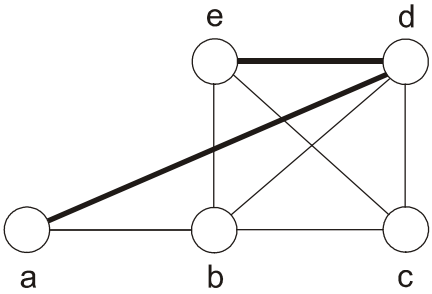
C:



C: {a,e}

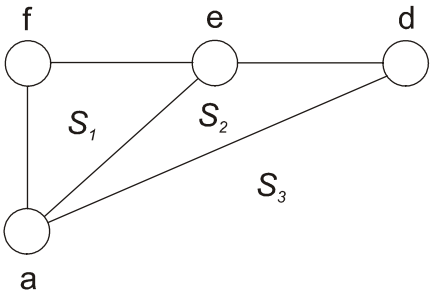
Je zřejmé, že tuto část lze umístit do obou stěn S_1 a S_2. Zvolíme její umístění do vnější stěny S_2. A pro zakreslení zvolíme třeba její chordu, která je zvýrazněna na dalším obrázku.

Chorda:



Nyní graf *H* má už tři stěny.

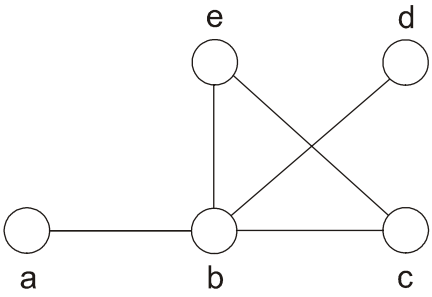
H:



S_1: {a,e,f}
S_2: {a,d,e}
S_3: {a,d,e,f}

Graf *G* opět obsahuje jednu část vzhledem k *H*.

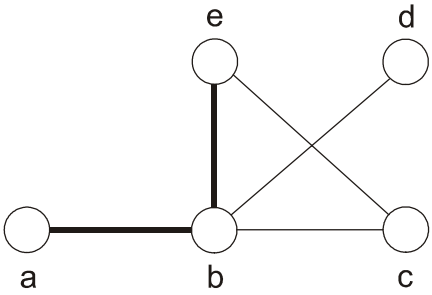
C:



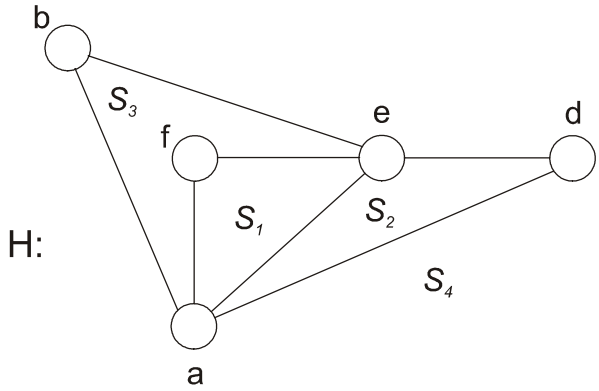
C: {a,d,e}

Je zřejmé, že ji lze umístit do stěn S_2, neboť $\{a,d,e\} \subseteq \{a,d,e\}$, a S_3, neboť $\{a,d,e\} \subseteq \{a,d,e,f\}$. Zvolíme vložení do vnější stěny S_3. Umístíme do ní třeba chordu, která je zvýrazněna na dalším obrázku.

Chorda:

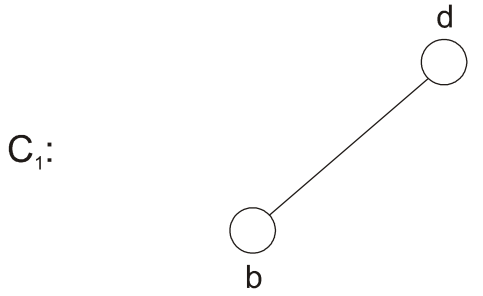


Po vložení chordy má graf *H* čtyři stěny.

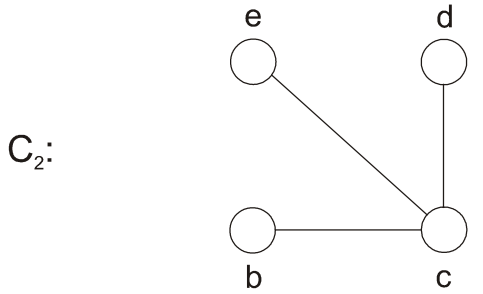


S_1: {a,e,f}
S_2: {a,d,e}
S_3: {a,b,e,f}
S_4: {a,b,d,e}

Graf *G* obsahuje dvě části vzhledem k *H*. Přitom jedna z těchto částí je tvořena hranou, jejíž oba koncové uzly jsou v *H*, ale hrana sama v *H* není. Druhá část je vytvořena tak, že k podgrafu *G-H*, který je tvořen již jen uzlem *c*, jsou přidány hrany spojující ho s *H*.

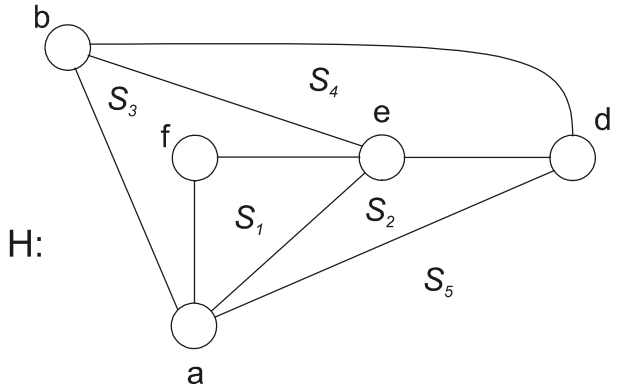


C_1: {b,d}



C_2: {b,d,e}

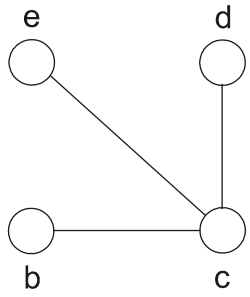
Část C_1 lze umístit jen do stěny S_4. Část C_2 lze rovněž umístit jen do stěny S_4. Umístíme třeba část C_1.



S_1: {a,e,f}
S_2: {a,d,e}
S_3: {a,b,e,f}
S_4: {b,d,e}
S_5: {a,b,d}

Ještě zbývá umístit část grafu *G* vzhledem k *H*, která je na dalším obrázku.

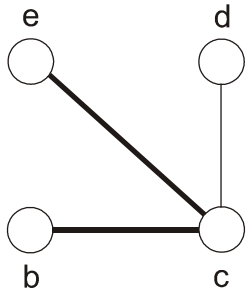
C:



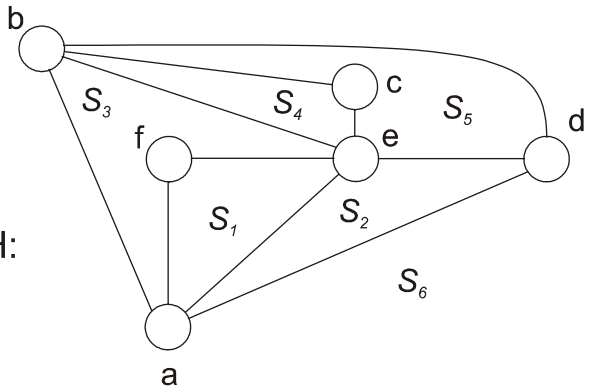
C: {b,d,e}

Část C_1 lze umístit jen do stěny S_4. Pro umístění zvolíme chordu zvýrazněnou na dalším obrázku.

Chorda:



H:



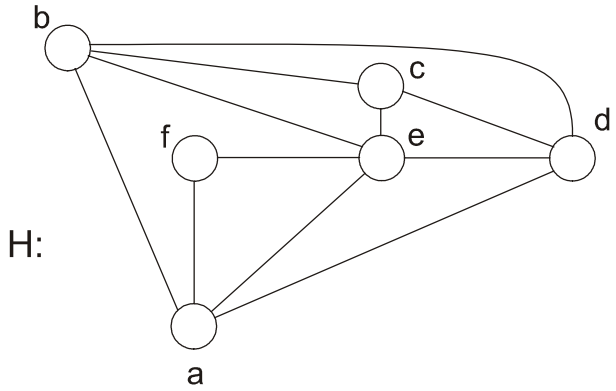
S_1: {a,e,f}
S_2: {a,d,e}
S_3: {a,b,e,f}
S_4: {b,c,e}
S_5: {b,c,d,e}
S_6: {a,b,d}

Nyní je podgraf *G*-*H* graf prázdný, nicméně zůstává část tvořená hranou mezi uzly *c* a *d*, která ještě v kresleném grafu *H* není.

C:

C: {c,d}

Tu lze umístit jen do stěny S_5. Když ji do ní dáme, je graf dokreslen.



Průvodce studiem

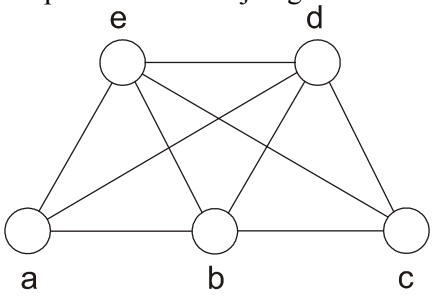
Velmi známý je problém obarvení mapy tak, aby žádné dva sousední státy neměly stejnou barvu. Zobrazíme-li státy uzly a vzájemně spojíme uzly sousedních států hranami, jde o úlohu obarvení planárního grafu. I když se dlouho soudilo, že chromatické číslo planárního grafu je nejvýše 4, dokázáno to bylo teprve v roce 1976 a to pomocí programu, kterým na počítači bylo ověřeno necelých 2000 různých případů.

Kontrolní otázky

1. *Co je to planární graf?*
2. *Znáte nějaké významné neplanární grafy?*
3. *Při jakém počtu hran vzhledem k počtu uzlů už je jisté, že graf není planární?*

Cvičení

1. Nakreslete planárně následující graf.

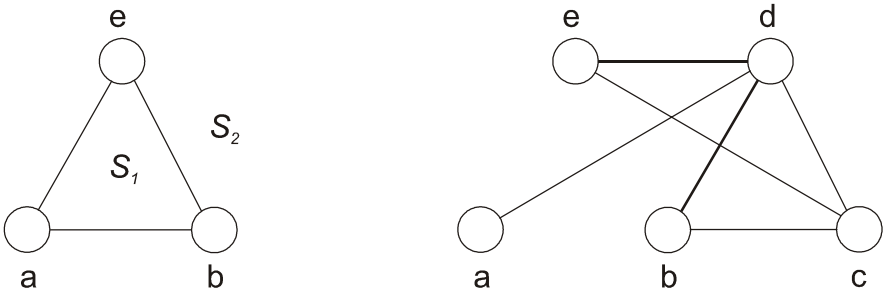


Úkoly k textu

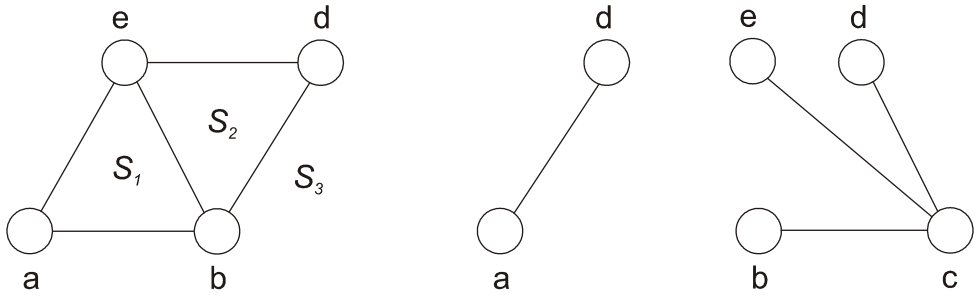
Dokažte, že úplný bipartitní graf s 3+3 uzly uvedený ve větě o neplanárnosti není planární. Vyjděte z toho, že tento graf neobsahuje kružnice délky 3. A dokažte, že planární graf, jehož kružnice mají nejmenší délku 4, musí splňovat nerovnost $|H| \leq 2|U| - 4$.

Řešení

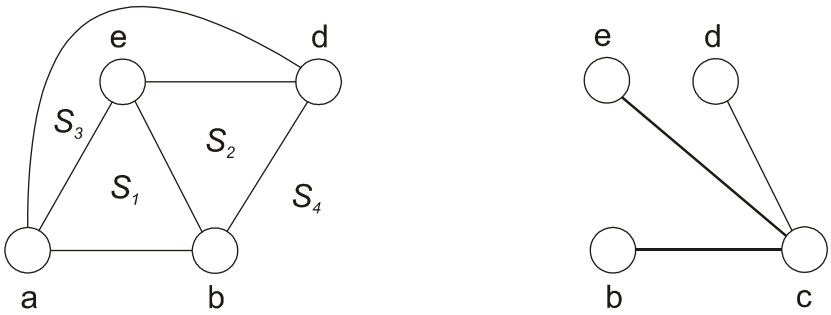
1. Zvolíme v grafu kružnici. Ta rozdělí rovinu na dvě stěny.



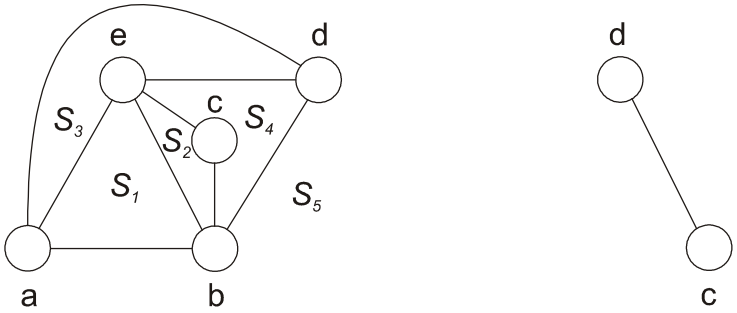
Graf má jednu část vzhledem ke zvolené kružnici, která je ne obrázku vpravo. Tu lze umístit do obou stěn. Zvolíme umístění do vnější stěny S_2. Umístíme do ní třeba chordu, která je v části vyznačena tučně.



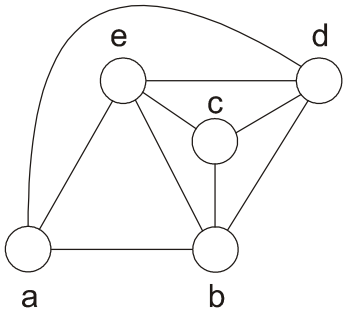
Nakreslený planární podgraf rozděluje nyní rovinu na 3 stěny. Graf má dvě části vzhledem k planárně nakreslenému podgrafu. První je tvořena jednou hranou a lze ji viditelně umístit jen do vnější stěny S_3. Druhou lze umístit do stěn S_2 a S_3. Musíme v tomto kroku umístit tu část, kterou lze umístit jen do jedné stěny.



Graf má nyní jednu část vzhledem k nakreslenému planárnímu podgrafu. Lze ji umístit jen do stěny S_2. Umístíme do ní chordu, která je v části silně vyznačena.



Graf má nyní část vzhledem k nakreslenému planárnímu podgrafu. Je to zbývající hrana a lze ji umístit jen do stěny S_4. Umístíme ji tam. Dokončené planární nakreslení grafu je na následujícím obrázku.


e
d
c
a
b

11 Rejstřík