

Drsná matematika III — 8. přednáška

Grafy a algoritmy: cesty a souvislost v grafech

Jan Slovák

Masarykova univerzita
Fakulta informatiky

6. 11. 2006

Obsah přednášky

- 1 Literatura
- 2 Algoritmy a reprezentace grafů
 - Reprezentace grafu
- 3 Prohledávání v grafech
 - Prohledávání do šířky a do hloubky
 - Souvislé komponenty grafu
- 4 Nejkratší cesty
 - Metrika na grafech
 - Dijkstrův algoritmus pro hledání nejkratších cest

Plán přednášky

- 1 Literatura
- 2 Algoritmy a reprezentace grafů
 - Reprezentace grafu
- 3 Prohledávání v grafech
 - Prohledávání do šířky a do hloubky
 - Souvislé komponenty grafu
- 4 Nejkratší cesty
 - Metrika na grafech
 - Dijkstrův algoritmus pro hledání nejkratších cest

Kde je dobré číst?

- Jiří Matoušek, Jaroslav Nešetřil, Kapitoly z diskrétní matematiky, Univerzita Karlova v Praze, Karolinum, Praha, 2000, 377 s.
- Petr Hliněný, Teorie grafů, studijní materiály, <http://www.fi.muni.cz/~hlineny/Vyuka/GT/>
- Bill Cherowitzo, Applied Graph Theory, Lecture Notes, <http://www-math.cudenver.edu/~wcherowi/courses/m4408/m4408f.html>

Plán přednášky

- 1 Literatura
- 2 Algoritmy a reprezentace grafů
 - Reprezentace grafu
- 3 Prohledávání v grafech
 - Prohledávání do šířky a do hloubky
 - Souvislé komponenty grafu
- 4 Nejkratší cesty
 - Metrika na grafech
 - Dijkstrův algoritmus pro hledání nejkratších cest

Abychom mohli algoritmy realizovat pomocí počítače, je třeba umět graf efektivně zadat. Uvedeme dva příklady:

Abychom mohli algoritmy realizovat pomocí počítače, je třeba umět graf efektivně zadat. Uvedeme dva příklady:

Definition

Hranový seznam (Edge List). Graf $G = (V, E)$ si v něm reprezentujeme jako dva seznamy V a E propojené ukazateli tak, že každý vrchol ukazuje na všechny z něj vycházející hrany (případně také na všechny do něj vcházející hrany u orientovaných grafů) a každá hrana ukazuje na svůj počáteční a koncový vrchol.

Abychom mohli algoritmy realizovat pomocí počítače, je třeba umět graf efektivně zadat. Uvedeme dva příklady:

Definition

Hranový seznam (Edge List). Graf $G = (V, E)$ si v něm reprezentujeme jako dva seznamy V a E propojené ukazateli tak, že každý vrchol ukazuje na všechny z něj vycházející hrany (případně také na všechny do něj vcházející hrany u orientovaných grafů) a každá hrana ukazuje na svůj počáteční a koncový vrchol.

Je vidět, že paměť potřebná na uchování grafu je v tomto případě $O(|V| + |E|)$, protože na každou hranu ukazujeme právě dvakrát a na každý vrchol ukazujeme tolikrát, kolik je jeho stupeň a součet stupňů je také roven dvojnásobku počtu hran. Až na konstantní násobek jde tedy stále o optimální způsob uchování grafu v paměti.

Definition (Matice sousednosti grafu)

Uvažme (neorientovaný) graf $G = (V, E)$, zvolme uspořádání jeho vrcholů $V = (v_1, \dots, v_n)$ a definujme matici $A_G = (a_{ij})$ nad \mathbb{Z}_2 (tj. zaplněnou jen nulami a jedničkami) takto:

$$a_{ij} = \begin{cases} 1 & \text{jestliže je hrana } e_{ij} = \{v_i, v_j\} \in E \\ 0 & \text{jestliže není hrana } e_{ij} = \{v_i, v_j\} \in E \end{cases}$$

Matici A_G nazýváme matice sousednosti grafu G .

Zadání grafu pomocí matice sousednosti potřebuje vždy $O(n^2)$ místa v paměti. Pokud je ale v grafu málo hran, dostáváme tzv. řídkou matici se skoro všemi prvky nulovými. Takové lze naopak reprezentovat pomocí „hranových seznamů“ odpovídajících grafů a to i včetně obecných číselných hodnot pro jednotlivé hrany. Příklad pro procvičení – násobení matic pomocí reprezentace hranovým seznamem.

Zadání grafu pomocí matice sousednosti potřebuje vždy $O(n^2)$ místa v paměti. Pokud je ale v grafu málo hran, dostáváme tzv. řídkou matici se skoro všemi prvky nulovými. Takové lze naopak reprezentovat pomocí „hranových seznamů“ odpovídajících grafů a to i včetně obecných číselných hodnot pro jednotlivé hrany. Příklad pro procvičení – násobení matic pomocí reprezentace hranovým seznamem.

Definition (Základní operace nad grafem)

- *odebrání hrany*
- *přidání hrany*
- *přidání vrcholu*
- *odebrání vrcholu*
- *dělení hrany nově přidaným vrcholem*

Jednoduchou aplikací maticového počtu je tvrzení:

Theorem

Nechť $G = (V, E)$ je graf s uspořádanými vrcholy $V = (v_1, \dots, v_n)$ a maticí sousednosti A_G . Označme $A_G^k = (a_{ij}^{(k)})$ prvky k -té mocniny matice A_G . Pak $a_{ij}^{(k)}$ je počet sledů délky k mezi vrcholy v_i a v_j .

Jednoduchou aplikací maticového počtu je tvrzení:

Theorem

Nechť $G = (V, E)$ je graf s uspořádanými vrcholy $V = (v_1, \dots, v_n)$ a maticí sousednosti A_G . Označme $A_G^k = (a_{ij}^{(k)})$ prvky k -té mocniny matice A_G . Pak $a_{ij}^{(k)}$ je počet sledů délky k mezi vrcholy v_i a v_j .

Corollary

Jsou-li $G = (V, E)$ a A_G jako v předchozí větě, pak lze všechny vrcholy G spojit cestou právě, když má matice $(A + \mathbb{I}_n)^{n-1}$ samé nenulové členy (zde \mathbb{I}_n označuje jednotkovou matici s n řádky a sloupci).

Jaký vliv má permutace našeho uspořádání uzlů V na matici sousednosti grafu?

Jaký vliv má permutace našeho uspořádání uzlů V na matici sousednosti grafu?

Permutace uzlů grafu G má za následek jednu a tutéž permutaci řádků a i sloupců matice A_G . Každou takovou permutaci můžeme zadat právě jednou tzv. permutační maticí, tj. maticí z nul a jedniček, která má v každém řádku a každém sloupci právě jednu jedničku a jinak nuly. Je-li P taková permutační matice, pak nová matice sousednosti izomorfního grafu G' bude

$$A_{G'} = P \cdot A_G \cdot P^T,$$

kde P^T značí transponovanou matici a tečkou označujeme násobení matic.

V případě permutačních matic je matice transponovaná zároveň maticí inverzní. Najdete souvislost matic sousednosti a matic lineárních zobrazení mezi vektorovými prostory?

Plán přednášky

- 1 Literatura
- 2 Algoritmy a reprezentace grafů
 - Reprezentace grafu
- 3 **Prohledávání v grafech**
 - Prohledávání do šířky a do hloubky
 - Souvislé komponenty grafu
- 4 Nejkratší cesty
 - Metrika na grafech
 - Dijkstrův algoritmus pro hledání nejkratších cest

Prohledávání v grafech

Algoritmy bývají založeny na postupném prohledávání všech vrcholů v grafu. Zpravidla máme zadaný počáteční vrchol nebo si jej na začátku procesu zvolíme. V průběhu procesu pak v každém okamžiku jsou vrcholy

- *již zpracované*, tj. ty, které jsme již při běhu algoritmu procházeli a definitivně zpracovali;
- *aktivní*, tj. ty vrcholy, které jsou detekovány a připraveny pro zpracovávání;
- *spící*, tj. ty vrcholy, na které teprve dojde.

Prohledávání v grafech

Algoritmy bývají založeny na postupném prohledávání všech vrcholů v grafu. Zpravidla máme zadaný počáteční vrchol nebo si jej na začátku procesu zvolíme. V průběhu procesu pak v každém okamžiku jsou vrcholy

- *již zpracované*, tj. ty, které jsme již při běhu algoritmu procházeli a definitivně zpracovali;
- *aktivní*, tj. ty vrcholy, které jsou detekovány a připraveny pro zpracovávání;
- *spící*, tj. ty vrcholy, na které teprve dojde.

Zároveň si udržujeme přehled o již zpracovaných hranách. V každém okamžiku musí být množiny vrcholů a/nebo hran v těchto skupinách disjunktním rozdělením množin V a E vrcholů a hran grafu G a některý z aktivních vrcholů je aktuálně zpracováván.

Základní postup:

- Na počátku máme jeden aktivní vrchol a všechny ostatní vrcholy jsou spící.

Základní postup:

- Na počátku máme jeden aktivní vrchol a všechny ostatní vrcholy jsou spící.
- V prvním kroku projdeme všechny hrany vycházející z aktivního vrcholu a jejich příslušným koncovým vrcholům, které jsou spící, změním je na aktivní.

Základní postup:

- Na počátku máme jeden aktivní vrchol a všechny ostatní vrcholy jsou spící.
- V prvním kroku projdeme všechny hrany vycházející z aktivního vrcholu a jejich příslušným koncovým vrcholům, které jsou spící, změním statut na aktivní.
- V dalších krocích vždy u zpracovávaného vrcholu probíráme ty z něho vycházející hrany, které dosud nebyly probrány a jejich koncové vrcholy přidáváme mezi aktivní. Tento postup aplikujeme stejně u orientovaných i neorientovaných grafů, jen se drobně mění význam adjektiv koncový a počáteční u vrcholů.

Základní postup:

- Na počátku máme jeden aktivní vrchol a všechny ostatní vrcholy jsou spící.
- V prvním kroku projdeme všechny hrany vycházející z aktivního vrcholu a jejich příslušným koncovým vrcholům, které jsou spící, změním statut na aktivní.
- V dalších krocích vždy u zpracovávaného vrcholu probíráme ty z něho vycházející hrany, které dosud nebyly probrány a jejich koncové vrcholy přidáváme mezi aktivní. Tento postup aplikujeme stejně u orientovaných i neorientovaných grafů, jen se drobně mění význam adjektiv koncový a počáteční u vrcholů.

V konkrétních úlohách se můžeme omezovat na některé z hran, které vychází z aktuálního vrcholu. Na principu to ale nic podstatného nemění.

Pro realizaci algoritmů je nutné se rozhodnout, v jakém pořadí zpracováváme aktivní vrcholy a v jakém pořadí zpracováváme hrany z nich vycházející. V zásadě přichází v úvahu dvě možnosti zpracovávání vrcholů:

- 1 vrcholy vybíráme pro další zpracování ve stejném pořadí, jak se stávaly aktivními (fronta)
- 2 dalším vrcholem vybraným pro zpracování je poslední zaktivněný vrchol (zásobník).

V prvním případě hovoříme o *prohledávání do šířky*, ve druhém o *prohledávání do hloubky*.

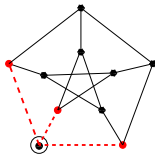
Na první pohled je zřejmá role volby vhodných datových struktur pro uchování údajů o grafu. Hranový seznam umožňuje projít všechny hrany vycházející z právě zpracovávaného vrcholu v čase lineárně úměrném jejich počtu. Každou hranu přitom diskutujeme nejvýše dvakrát, protože má právě dva konce. Zjevně tedy platí:

Na první pohled je zřejmá role volby vhodných datových struktur pro uchování údajů o grafu. Hranový seznam umožňuje projít všechny hrany vycházející z právě zpracovávaného vrcholu v čase lineárně úměrném jejich počtu. Každou hranu přitom diskutujeme nejvýše dvakrát, protože má právě dva konce. Zjevně tedy platí:

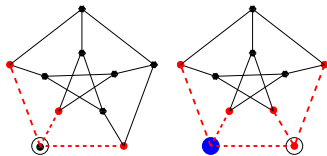
Theorem

*Celkový čas realizace vyhledávání do šířky i do hloubky v čase $O((n + m) * K)$, kde n je počet vrcholů v grafu, m je počet hran v grafu a K je čas potřebný na zpracování jedné hrany, resp. jednoho vrcholu.*

Ilustrace prohledávání do šířky:

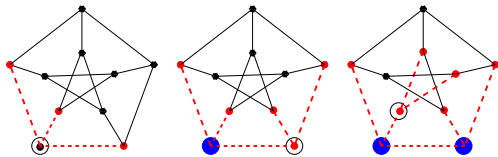


Ilustrace prohledávání do šířky:



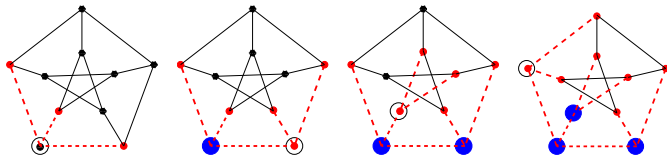
Zakroužkovaný vrchol je ten právě zpracovávaný, modré velké puntíky jsou již zpracované uzly, čárkované červené hrany jsou již zpracované a červené drobné uzly jsou ty aktivní (poznají se také podle toho, že do nich již vede některá zpracovaná hrana). Hrany zpracováváme v pořadí orientace proti hodinovým ručkám, přičemž za „první“ bereme směr „kolmo dolů“.

Ilustrace prohledávání do šířky:



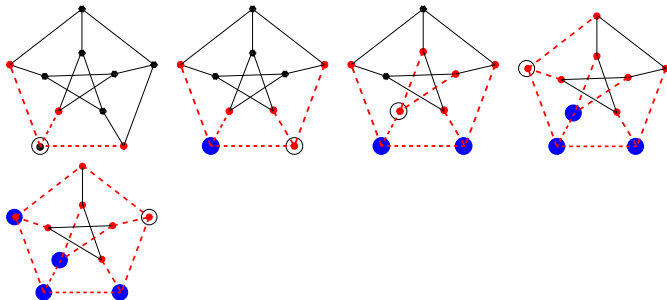
Zakroužkovaný vrchol je ten právě zpracovávaný, modré velké puntíky jsou již zpracované uzly, čárkované červené hrany jsou již zpracované a červené drobné uzly jsou ty aktivní (poznají se také podle toho, že do nich již vede některá zpracovaná hrana). Hrany zpracováváme v pořadí orientace proti hodinovým ručkám, přičemž za „první“ bereme směr „kolmo dolů“.

Ilustrace prohledávání do šířky:



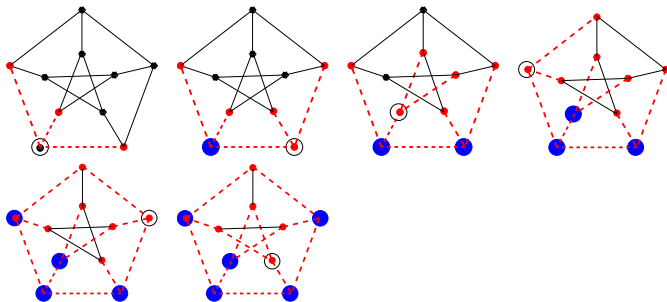
Zakroužkovaný vrchol je ten právě zpracovávaný, modré velké puntíky jsou již zpracované uzly, čárkované červené hrany jsou již zpracované a červené drobné uzly jsou ty aktivní (poznají se také podle toho, že do nich již vede některá zpracovaná hrana). Hrany zpracováváme v pořadí orientace proti hodinovým ručkám, přičemž za „první“ bereme směr „kolmo dolů“.

Ilustrace prohledávání do šířky:



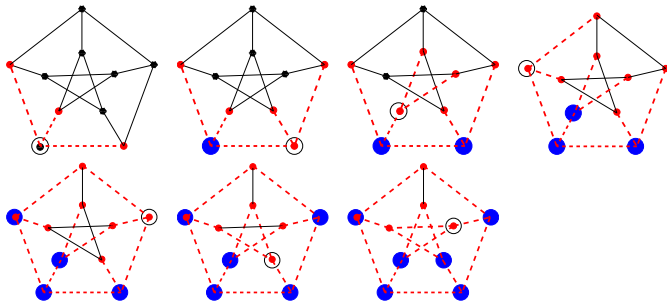
Zakroužkovaný vrchol je ten právě zpracovávaný, modré velké puntíky jsou již zpracované uzly, čárkované červené hrany jsou již zpracované a červené drobné uzly jsou ty aktivní (poznají se také podle toho, že do nich již vede některá zpracovaná hrana). Hrany zpracováváme v pořadí orientace proti hodinovým ručkám, přičemž za „první“ bereme směr „kolmo dolů“.

Ilustrace prohledávání do šířky:



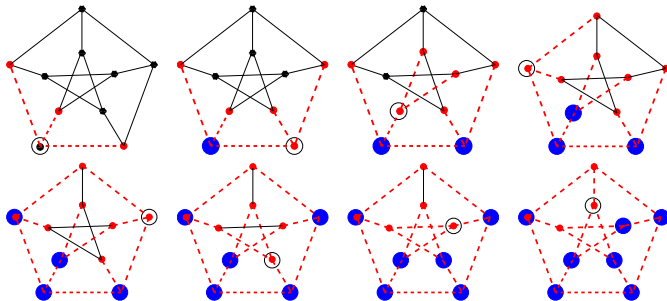
Zakroužkovaný vrchol je ten právě zpracovávaný, modré velké puntíky jsou již zpracované uzly, čárkované červené hrany jsou již zpracované a červené drobné uzly jsou ty aktivní (poznají se také podle toho, že do nich již vede některá zpracovaná hrana). Hrany zpracováváme v pořadí orientace proti hodinovým ručkám, přičemž za „první“ bereme směr „kolmo dolů“.

Ilustrace prohledávání do šířky:



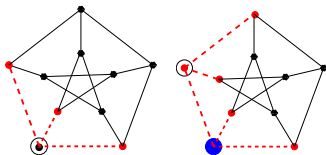
Zakroužkovaný vrchol je ten právě zpracovávaný, modré velké puntíky jsou již zpracované uzly, čárkované červené hrany jsou již zpracované a červené drobné uzly jsou ty aktivní (poznají se také podle toho, že do nich již vede některá zpracovaná hrana). Hrany zpracováváme v pořadí orientace proti hodinovým ručkám, přičemž za „první“ bereme směr „kolmo dolů“.

Ilustrace prohledávání do šířky:

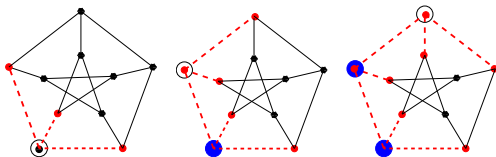


Zakroužkovaný vrchol je ten právě zpracováváný, modré velké puntíky jsou již zpracované uzly, čárkované červené hrany jsou již zpracované a červené drobné uzly jsou ty aktivní (poznají se také podle toho, že do nich již vede některá zpracovaná hrana). Hrany zpracováváme v pořadí orientace proti hodinovým ručkám, přičemž za „první“ bereme směr „kolmo dolů“.

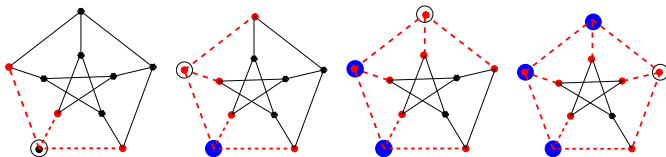
Totéž postupem „do hloubky“. Všimněte si, že první krok je stejný jako v předchozím případě.



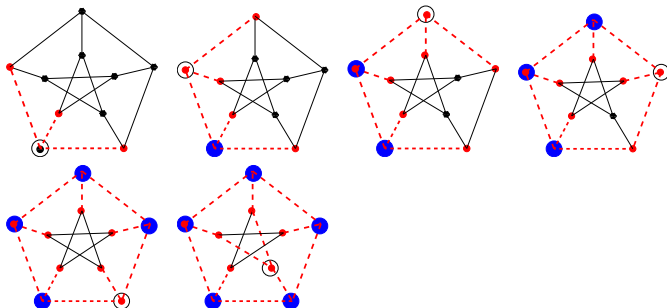
Totéž postupem „do hloubky“. Všimněte si, že první krok je stejný jako v předchozím případě.



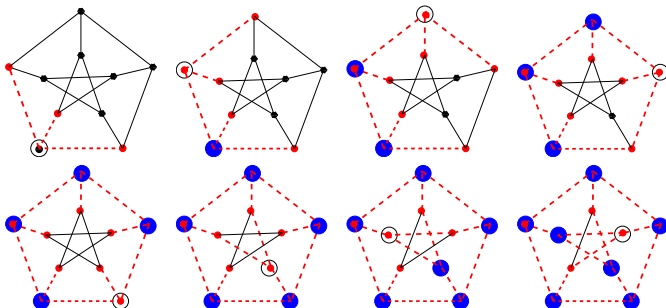
Totéž postupem „do hloubky“. Všimněte si, že první krok je stejný jako v předchozím případě.



Totéž postupem „do hloubky“. Všimněte si, že první krok je stejný jako v předchozím případě.



Totéž postupem „do hloubky“. Všimněte si, že první krok je stejný jako v předchozím případě.



Souvislé komponenty grafu

Definition

Nechť je $G = (V, E)$ neorientovaný graf. Na množině vrcholů grafu G zavedeme relaci \sim tak, že $v \sim w$ právě když existuje cesta z v do w . Promyslete si, že tato relace je dobře definovaná a že se jedná o ekvivalenci. Každá třída $[v]$ této ekvivalence definuje indukovaný podgraf $G_{[v]} \subset G$ a disjunkttní sjednocení těchto podgrafů je ve skutečnosti původní graf G . Podgrafům $G_{[v]}$ říkáme *souvislé komponenty grafu G* .

Souvislé komponenty grafu

Definition

Nechť je $G = (V, E)$ neorientovaný graf. Na množině vrcholů grafu G zavedeme relaci \sim tak, že $v \sim w$ právě když existuje cesta z v do w . Promyslete si, že tato relace je dobře definovaná a že se jedná o ekvivalenci. Každá třída $[v]$ této ekvivalence definuje indukovaný podgraf $G_{[v]} \subset G$ a disjunkttní sjednocení těchto podgrafů je ve skutečnosti původní graf G . Podgrafům $G_{[v]}$ říkáme *souvislé komponenty grafu G* .

Pro orientované grafy postupujeme stejně, pouze u \sim požadujeme aby cesta existovala z uzlu v do uzlu w nebo naopak z uzlu w do uzlu v .

Souvislé komponenty grafu

Definition

Nechť je $G = (V, E)$ neorientovaný graf. Na množině vrcholů grafu G zavedeme relaci \sim tak, že $v \sim w$ právě když existuje cesta z v do w . Promyslete si, že tato relace je dobře definovaná a že se jedná o ekvivalenci. Každá třída $[v]$ této ekvivalence definuje indukovaný podgraf $G_{[v]} \subset G$ a disjunktí sjednocení těchto podgrafů je ve skutečnosti původní graf G . Podgrafům $G_{[v]}$ říkáme *souvislé komponenty grafu G* .

Pro orientované grafy postupujeme stejně, pouze u \sim požadujeme aby cesta existovala z uzlu v do uzlu w nebo naopak z uzlu w do uzlu v .

Jinými slovy: Každý graf $G = (V, E)$ se přirozeně rozpadá na disjunktí podgrafy G_i takové, že vrcholy $v \in G_i$ a $w \in G_j$ jsou spojeny nějakou cestou právě, když $i = j$. To jsou právě souvislé komponenty grafu G .

Pro hledání všech souvislých komponent v grafu lze užít
prohledávání — dodatečnou informací, kterou musíme zpracovávat
je, kterou komponentu aktuálně procházíme.

Pro hledání všech souvislých komponent v grafu lze užít prohledávání — dodatečnou informací, kterou musíme zpracovávat je, kterou komponentu aktuálně procházíme.

Definition

Řekneme, že graf $G = (V, E)$ je

- *souvislý*, jestliže má právě jednu souvislou komponentu;
- *vrcholově k -souvislý*, jestliže má alespoň $k + 1$ vrcholů a bude souvislý po odebrání libovolné podmnožiny $k - 1$ vrcholů;
- *hranově k -souvislý*, jestliže bude souvislý po odebrání libovolné podmnožiny $k - 1$ hran.

Pro hledání všech souvislých komponent v grafu lze užít prohledávání — dodatečnou informací, kterou musíme zpracovávat je, kterou komponentu aktuálně procházíme.

Definition

Řekneme, že graf $G = (V, E)$ je

- *souvislý*, jestliže má právě jednu souvislou komponentu;
- *vrcholově k -souvislý*, jestliže má alespoň $k + 1$ vrcholů a bude souvislý po odebrání libovolné podmnožiny $k - 1$ vrcholů;
- *hranově k -souvislý*, jestliže bude souvislý po odebrání libovolné podmnožiny $k - 1$ hran.

Silnější souvislost grafu je žádoucí např. u síťových aplikací, kdy klient požaduje značnou redundanci poskytovaných služeb v případě výpadku některých linek (tj. hran) nebo uzlů (tj. vrcholů).

Pro hledání všech souvislých komponent v grafu lze užít prohledávání — dodatečnou informací, kterou musíme zpracovávat je, kterou komponentu aktuálně procházíme.

Definition

Řekneme, že graf $G = (V, E)$ je

- *souvislý*, jestliže má právě jednu souvislou komponentu;
- *vrcholově k -souvislý*, jestliže má alespoň $k + 1$ vrcholů a bude souvislý po odebrání libovolné podmnožiny $k - 1$ vrcholů;
- *hranově k -souvislý*, jestliže bude souvislý po odebrání libovolné podmnožiny $k - 1$ hran.

Silnější souvislost grafu je žádoucí např. u síťových aplikací, kdy klient požaduje značnou redundanci poskytovaných služeb v případě výpadku některých linek (tj. hran) nebo uzlů (tj. vrcholů). Speciální případ: 2-souvislý graf je takový souvislý graf o alespoň třech vrcholech, kdy vynecháním libovolného vrcholu nenarušíme jeho souvislost.

Theorem

Pro graf $G = (V, E)$ s alespoň třemi vrcholy jsou následující podmínky ekvivalentní:

- *G je (vrcholově) 2-souvislý;*
- *každé dva vrcholy v a w v grafu G leží na společné kružnici;*
- *graf G je možné vytvořit z trojúhelníku K_3 pomocí postupných dělení hran.*

Theorem

Pro graf $G = (V, E)$ s alespoň třemi vrcholy jsou následující podmínky ekvivalentní:

- *G je (vrcholově) 2-souvislý;*
- *každé dva vrcholy v a w v grafu G leží na společné kružnici;*
- *graf G je možné vytvořit z trojúhelníku K_3 pomocí postupných dělení hran.*

Obecnější je tzv. *Mengerova věta*:

Theorem

Pro každé dva vrcholy v a w v grafu $G = (V, E)$ je počet hranově různých cest z v do w roven minimálnímu počtu hran, které je třeba odstranit, aby se v a w ocitly v různých komponentách vzniklého grafu.

Plán přednášky

- 1 Literatura
- 2 Algoritmy a reprezentace grafů
 - Reprezentace grafu
- 3 Prohledávání v grafech
 - Prohledávání do šířky a do hloubky
 - Souvislé komponenty grafu
- 4 **Nejkratší cesty**
 - Metrika na grafech
 - Dijkstrův algoritmus pro hledání nejkratších cest

Na každém (neorientovaném) grafu definujeme *vzdálenost uzlů* v a w jako číslo $d_G(v, w)$, které je rovno počtu hran v nejkratší možné cestě z v do w . Pokud cesta neexistuje, píšeme $d_G(v, w) = \infty$.

Na každém (neorientovaném) grafu definujeme *vzdálenost uzlů* v a w jako číslo $d_G(v, w)$, které je rovno počtu hran v nejkratší možné cestě z v do w . Pokud cesta neexistuje, píšeme $d_G(v, w) = \infty$. Budeme v dalším uvažovat pouze souvislé graf G . Pak pro takto zadanou funkci $d_G : V \times V \rightarrow \mathbb{N}$ platí obvyklé tři vlastnosti vzdálenosti:

- $d_G(v, w) \geq 0$ a přitom $d_G(v, w) = 0$ právě, když $v = w$;
- vzdálenost je symetrická, tj $d_G(v, w) = d_G(w, v)$;
- platí trojúhelníková nerovnost, tj. pro každou trojici vrcholů v, w, z platí

$$d_G(v, z) \leq d_G(v, w) + d_G(w, z).$$

Na každém (neorientovaném) grafu definujeme *vzdálenost uzlů* v a w jako číslo $d_G(v, w)$, které je rovno počtu hran v nejkratší možné cestě z v do w . Pokud cesta neexistuje, píšeme $d_G(v, w) = \infty$. Budeme v dalším uvažovat pouze souvislé graf G . Pak pro takto zadanou funkci $d_G : V \times V \rightarrow \mathbb{N}$ platí obvyklé tři vlastnosti vzdálenosti:

- $d_G(v, w) \geq 0$ a přitom $d_G(v, w) = 0$ právě, když $v = w$;
- vzdálenost je symetrická, tj $d_G(v, w) = d_G(w, v)$;
- platí trojúhelníková nerovnost, tj. pro každou trojici vrcholů v, w, z platí

$$d_G(v, z) \leq d_G(v, w) + d_G(w, z).$$

Říkáme, že d_G je *metrika na grafu* G .

Na každém (neorientovaném) grafu definujeme *vzdálenost uzlů* v a w jako číslo $d_G(v, w)$, které je rovno počtu hran v nejkratší možné cestě z v do w . Pokud cesta neexistuje, píšeme $d_G(v, w) = \infty$. Budeme v dalším uvažovat pouze souvislé graf G . Pak pro takto zadanou funkci $d_G : V \times V \rightarrow \mathbb{N}$ platí obvyklé tři vlastnosti vzdálenosti:

- $d_G(v, w) \geq 0$ a přitom $d_G(v, w) = 0$ právě, když $v = w$;
- vzdálenost je symetrická, tj $d_G(v, w) = d_G(w, v)$;
- platí trojúhelníková nerovnost, tj. pro každou trojici vrcholů v, w, z platí

$$d_G(v, z) \leq d_G(v, w) + d_G(w, z).$$

Říkáme, že d_G je *metrika na grafu* G .

Metrika na grafu splňuje navíc:

- $d_G(v, w)$ má vždy nezáporné celočíselné hodnoty;
- je-li $d_G(v, w) > 1$, pak existuje nějaký vrchol z různý od v a w a takový, že $d_G(v, w) = d_G(v, z) + d_G(z, w)$.

Každá funkce d_G s výše uvedenými pěti vlastnostmi na $V \times V$ je metrikou grafu s vrcholy G .

Nejkratší cestu v grafu, která vychází z daného uzlu v a končí v jiném uzlu w můžeme hledat pomocí prohledávání grafu do šířky. Při tomto typu prohledávání totiž postupně diskutujeme vrcholy, do kterých se umíme dostat z výchozího vrcholu po jediné hraně, poté projdeme všechny, které mají vzdálenost nejvýše 2 atd. Na této jednoduché úvaze je založen jeden z nejpoužívanějších grafových algoritmů – tzv. *Dijkstrův algoritmus*.

Nejkratší cestu v grafu, která vychází z daného uzlu v a končí v jiném uzlu w můžeme hledat pomocí prohledávání grafu do šířky. Při tomto typu prohledávání totiž postupně diskutujeme vrcholy, do kterých se umíme dostat z výchozího vrcholu po jediné hraně, poté projdeme všechny, které mají vzdálenost nejvýše 2 atd. Na této jednoduché úvaze je založen jeden z nejpoužívanějších grafových algoritmů – tzv. *Dijkstrův algoritmus*.

Tento algoritmus hledá nejkratší cesty v realističtější podobě, kdy jednotlivé hrany e jsou ohodnoceny „vzdálenostmi“, tj. kladnými reálnými čísly $w(e)$. Kromě aplikace na hledání vzdáleností v silničních nebo jiných sítích to mohou být také výnosy, toky v sítích atd.

- Vstupem algoritmu je graf $G = (V, E)$ s ohodnocením hran a počáteční vrchol v_0 .

- Vstupem algoritmu je graf $G = (V, E)$ s ohodnocením hran a počáteční vrchol v_0 .
- Výstupem je ohodnocení vrcholů čísla $d_w(v)$, která udávají nejmenší možný součet ohodnocení hran podél cest z vrcholu v_0 do vrcholu v .

Postup dobře funguje v orientovaných i neorientovaných grafech.

- Vstupem algoritmu je graf $G = (V, E)$ s ohodnocením hran a počáteční vrchol v_0 .
- Výstupem je ohodnocení vrcholů čísla $d_w(v)$, která udávají nejmenší možný součet ohodnocení hran podél cest z vrcholu v_0 do vrcholu v .

Postup dobře funguje v orientovaných i neorientovaných grafech. Je skutečně podstatné, že všechna naše ohodnocení jsou kladná. Zkusme si rozmyslet třeba cestu P_3 se záporně ohodnocenou prostřední hranou. Při procházení sledu mezi krajními vrcholy bychom „vzdálenost“ zmenšovali každým prodloužením sledu o průchod prostřední hranou tam a zpět.

Dijkstrův algoritmus vyžaduje jen drobnou modifikaci obecného prohledávání do šířky:

- U každého vrcholu v budeme po celý chod algoritmu udržovat číselnou hodnotu $d(v)$, která bude horním odhadem skutečné vzdálenosti vrcholu v od vrcholu v_0 .

Dijkstrův algoritmus vyžaduje jen drobnou modifikaci obecného prohledávání do šířky:

- U každého vrcholu v budeme po celý chod algoritmu udržovat číselnou hodnotu $d(v)$, která bude horním odhadem skutečné vzdálenosti vrcholu v od vrcholu v_0 .
- Množina již zpracovaných vrcholů bude v každém okamžiku obsahovat ty vrcholy, u kterých již nejkratší cestu známe, tj. $d(v) = d_w(v)$.

Dijkstrův algoritmus vyžaduje jen drobnou modifikaci obecného prohledávání do šířky:

- U každého vrcholu v budeme po celý chod algoritmu udržovat číselnou hodnotu $d(v)$, která bude horním odhadem skutečné vzdálenosti vrcholu v od vrcholu v_0 .
- Množina již zpracovaných vrcholů bude v každém okamžiku obsahovat ty vrcholy, u kterých již nejkratší cestu známe, tj. $d(v) = d_w(v)$.
- Do množiny aktivních (právě zpracovávaných) vrcholů W zařadíme vždy právě ty vrcholy y z množiny spících vrcholů Z , pro které je $d(y) = \min\{d(z); z \in Z\}$.

Dijkstrův algoritmus

Předpokládáme, že graf G má alespoň dva vrcholy.

- *Iniciační krok:* Nastavíme hodnoty u všech $v \in V$,

$$d(v) = \begin{cases} 0 & \text{pro } v = v_0 \\ \infty & \text{pro } v \neq v_0, \end{cases}$$

nastavíme $Z = V$, $W = \emptyset$.

Dijkstrův algoritmus

Předpokládáme, že graf G má alespoň dva vrcholy.

- *Iniciační krok:* Nastavíme hodnoty u všech $v \in V$,

$$d(v) = \begin{cases} 0 & \text{pro } v = v_0 \\ \infty & \text{pro } v \neq v_0, \end{cases}$$

nastavíme $Z = V$, $W = \emptyset$.

- *Test cyklu:* Pokud není ohodnocení všech vrcholů $y \in Z$ rovno ∞ , pokračujeme dalším krokem, v opačném případě algoritmus končí.

- *Aktualizace statutu vrcholů:*

- Najdeme množinu N všech vrcholů $v \in Z$, pro které $d(v)$ nabývá nejmenší možné hodnoty

$$\delta = \min\{d(y); y \in Z\};$$

- posledně zpracované aktivní vrcholy W přesuneme do množiny zpracovaných a za nové aktivní vrcholy zvolíme $W = N$ a odebereme je ze spících, tj. množina spících bude nadále $Z \setminus N$.

- *Aktualizace statutu vrcholů:*

- Najdeme množinu N všech vrcholů $v \in Z$, pro které $d(v)$ nabývá nejmenší možné hodnoty

$$\delta = \min\{d(y); y \in Z\};$$

- posledně zpracované aktivní vrcholy W přesuneme do množiny zpracovaných a za nové aktivní vrcholy zvolíme $W = N$ a odebereme je ze spících, tj. množina spících bude nadále $Z \setminus N$.
- *Tělo hlavního cyklu:* Pro všechny hrany v množině E_{WZ} všech hran vycházejících z některého aktivního vrcholu v a končících ve spícím vrcholu y opakujeme:
 - Vybereme dosud nezpracovanou hranu $e \in E_{WZ}$;
 - Pokud je $d(v) + w(e) < d(y)$, nahradíme $d(y)$ touto menší hodnotou.

Theorem

Pro všechny vrcholy v v souvislé komponentě vrcholu v_0 najde Dijkstraův algoritmus vzdálenosti $d_w(v)$. Vrcholy ostatních souvislých komponent zůstanou ohodnoceny $d(v) = \infty$. Algoritmus lze implementovat tak, že ukončí svoji práci v čase $O(n \log n + m)$, kde n je počet vrcholů a m je počet hran v grafu G .