Drsná matematika III — 8. přednáška
Grafy a algoritmy: cesty a souvislost v grafech

Jan Slovák

Masarykova univerzita
Fakulta informatiky

7. 11. 2011

Plán přednášky

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 ostaní vrcholy jsou spící.

Základní postup:

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

Základní postup:

- Na počátku máme jeden aktivní vrchol a všechny ostaní vrcholy jsou spící.
- V prvním kroku projdeme všechny hrany vycházející z aktivního vrcholu a jejich příslušným koncovým vrcholům, které jsou spící, změníme 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 ostaní vrcholy jsou spící.
- V prvním kroku projdeme všechny hrany vycházející z aktivního vrcholu a jejich příslušným koncovým vrcholům, které jsou spící, změníme 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 algortimů 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**.

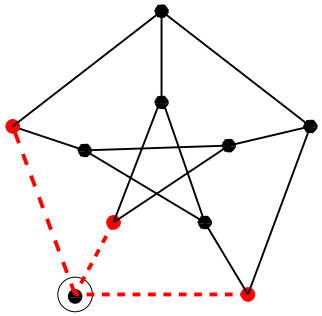
Na první pohled je zřejmá role volby vhodných datových struktur pro uchovávání údajů o grafu. Hranový seznam umožňuje projít všechny hrany vycházející z právě zpracovávaného vrcholu v čase lineárně úměrném jejich počtu. Každou hranu přitom diskutujeme nejvýše dvakrát, protože má právě dva konce. Zjevně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

Na první pohled je zřejmá role volby vhodných datových struktur pro uchovávání 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:

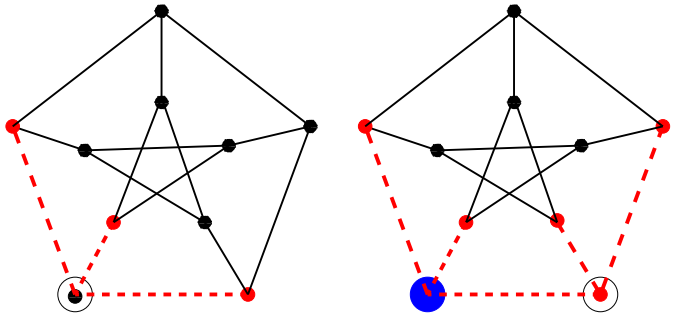
Theorem

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:

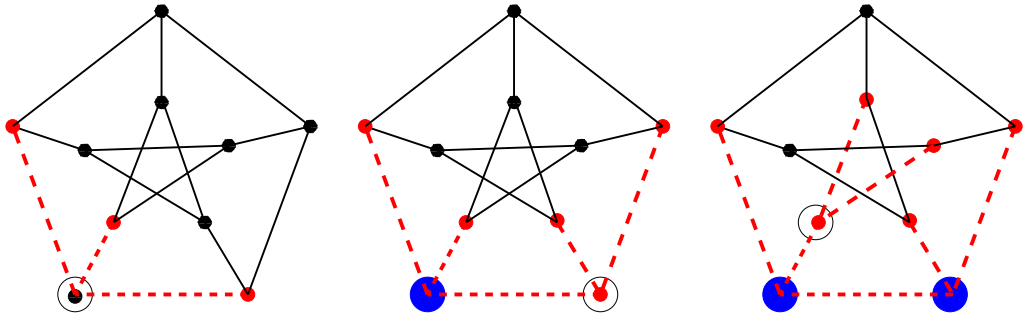


Ilustrace prohledávání do šířky:



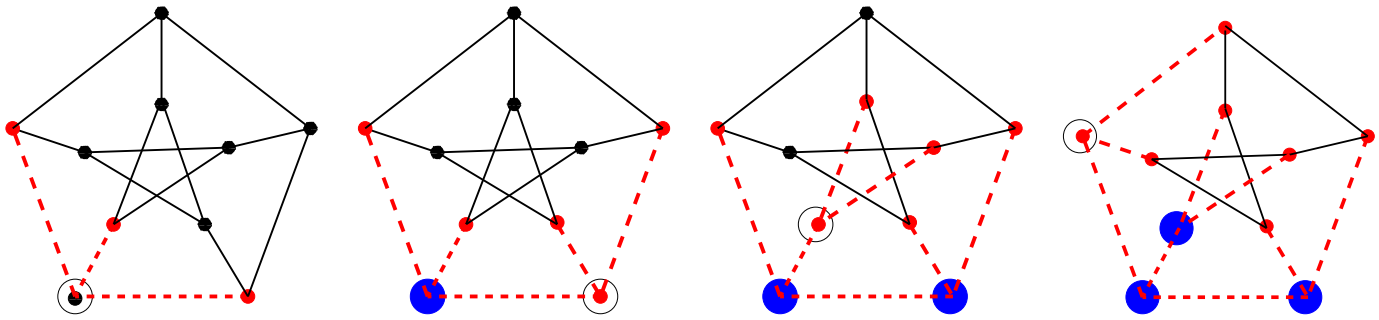
Zakroužkovaný vrchol je ten právě zpracovávaný, modré velké puntíky jsou již zpracované uzly, čárkované červené hrany jsou již zpracované a červené drobné uzly jsou ty aktivní (poznají se také podle toho, že do nich již vede některá zpracovaná hrana). Hrany zpracováváme v pořadí orientace proti hodinovým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

Ilustrace prohledávání do šířky:



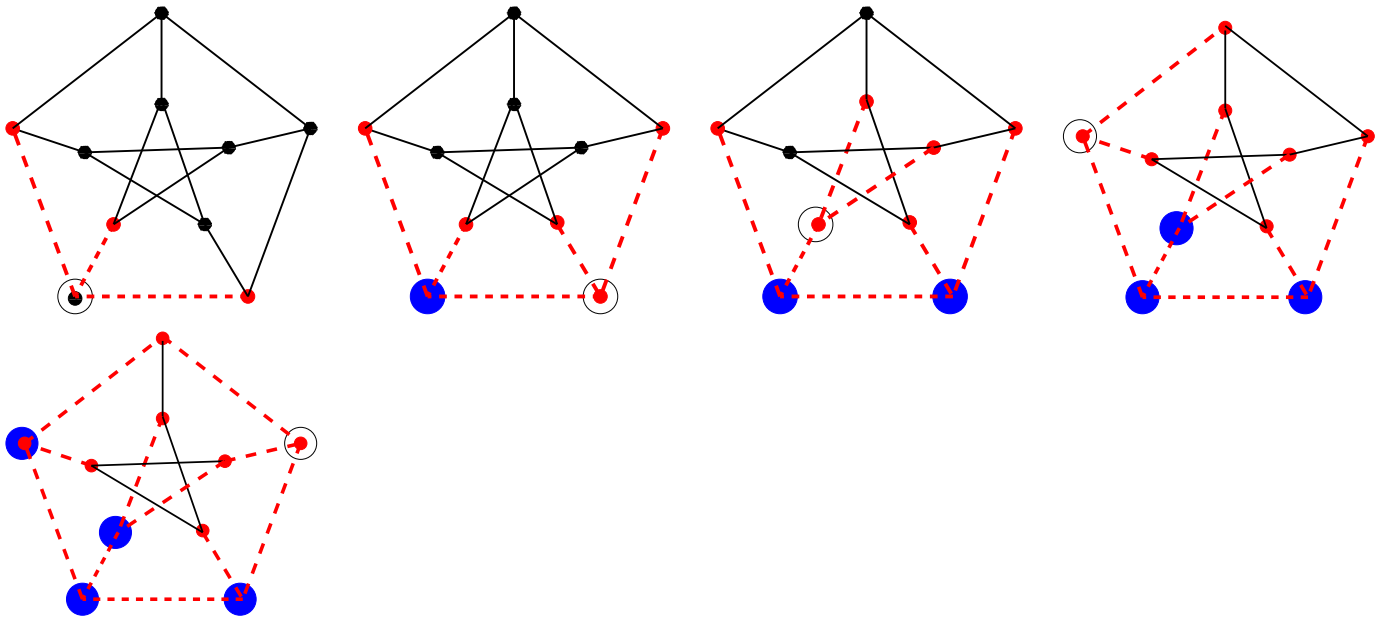
Zakroužkovaný vrchol je ten právě zpracovávaný, modré velké puntíky jsou již zpracované uzly, čárkované červené hrany jsou již zpracované a červené drobné uzly jsou ty aktivní (poznají se také podle toho, že do nich již vede některá zpracovaná hrana). Hrany zpracováváme v pořadí orientace proti hodinovým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

Ilustrace prohledávání do šířky:



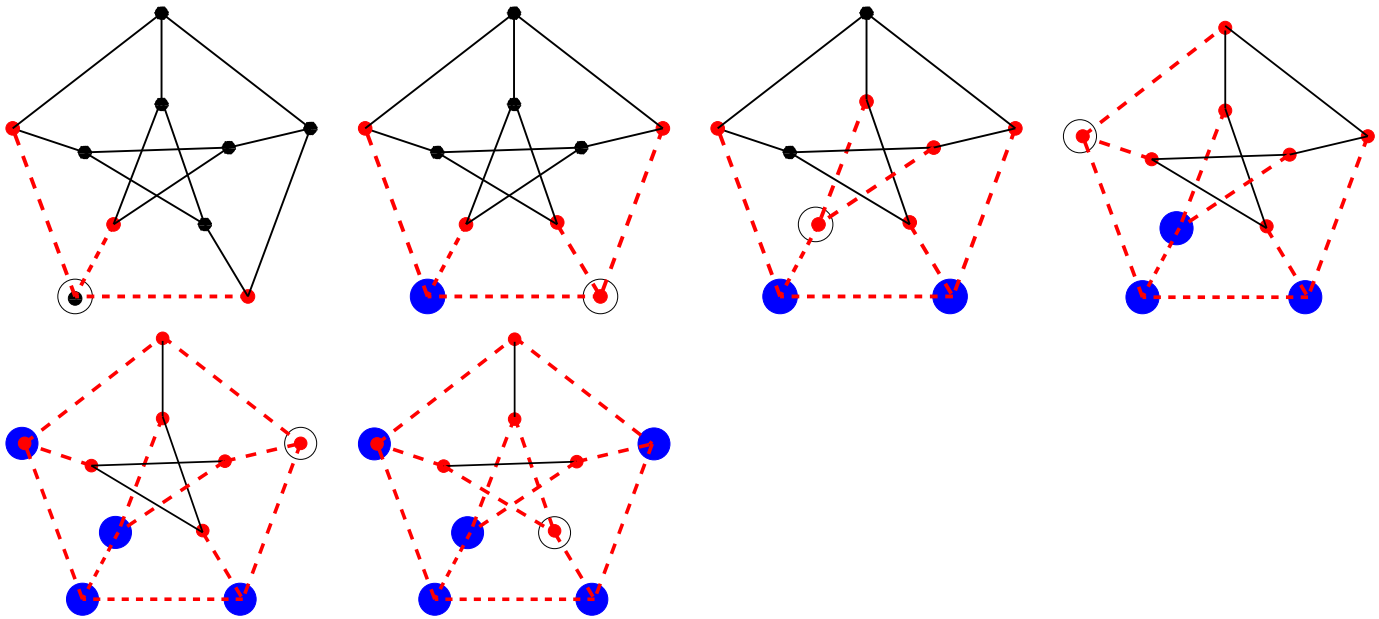
Zakroužkovaný vrchol je ten právě zpracovávaný, modré velké puntíky jsou již zpracované uzly, čárkované červené hrany jsou již zpracované a červené drobné uzly jsou ty aktivní (poznají se také podle toho, že do nich již vede některá zpracovaná hrana). Hrany zpracováváme v pořadí orientace proti hodinovým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

Ilustrace prohledávání do šířky:



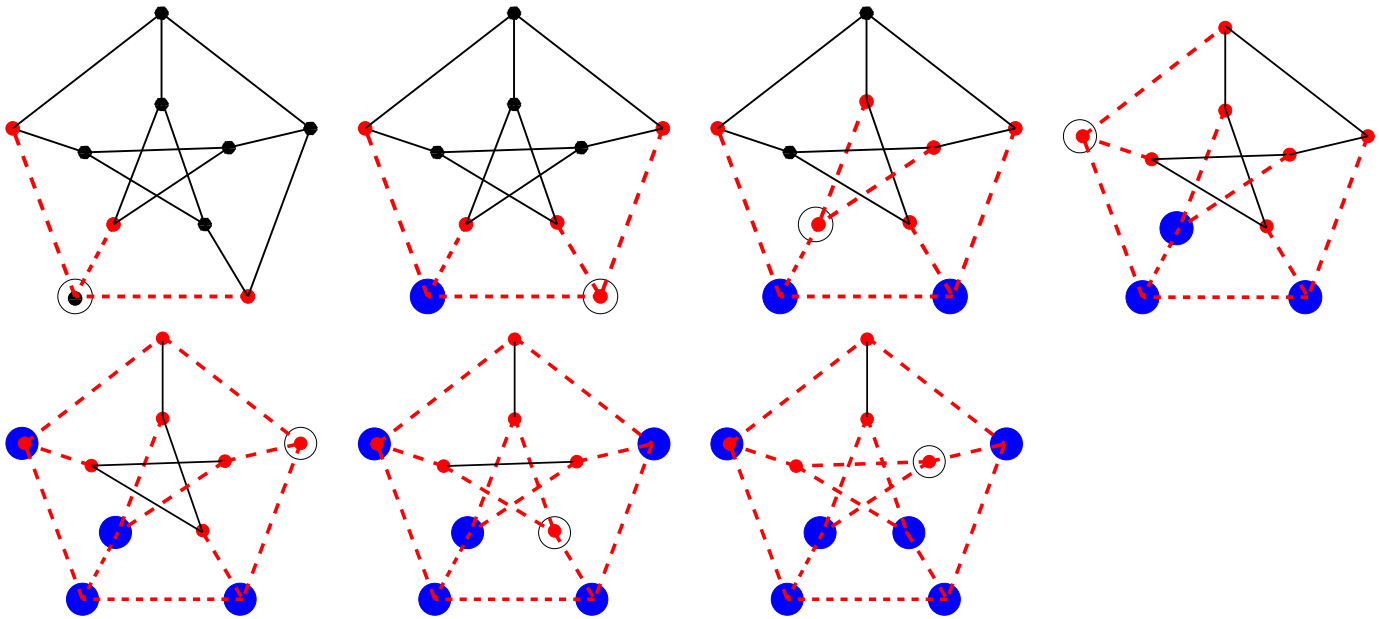
Zakroužkovaný vrchol je ten právě zpracovávaný, modré velké puntíky jsou již zpracované uzly, čárkované červené hrany jsou již zpracované a červené drobné uzly jsou ty aktivní (poznají se také podle toho, že do nich již vede některá zpracovaná hrana). Hrany zpracováváme v pořadí orientace proti hodinovým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

Ilustrace prohledávání do šířky:



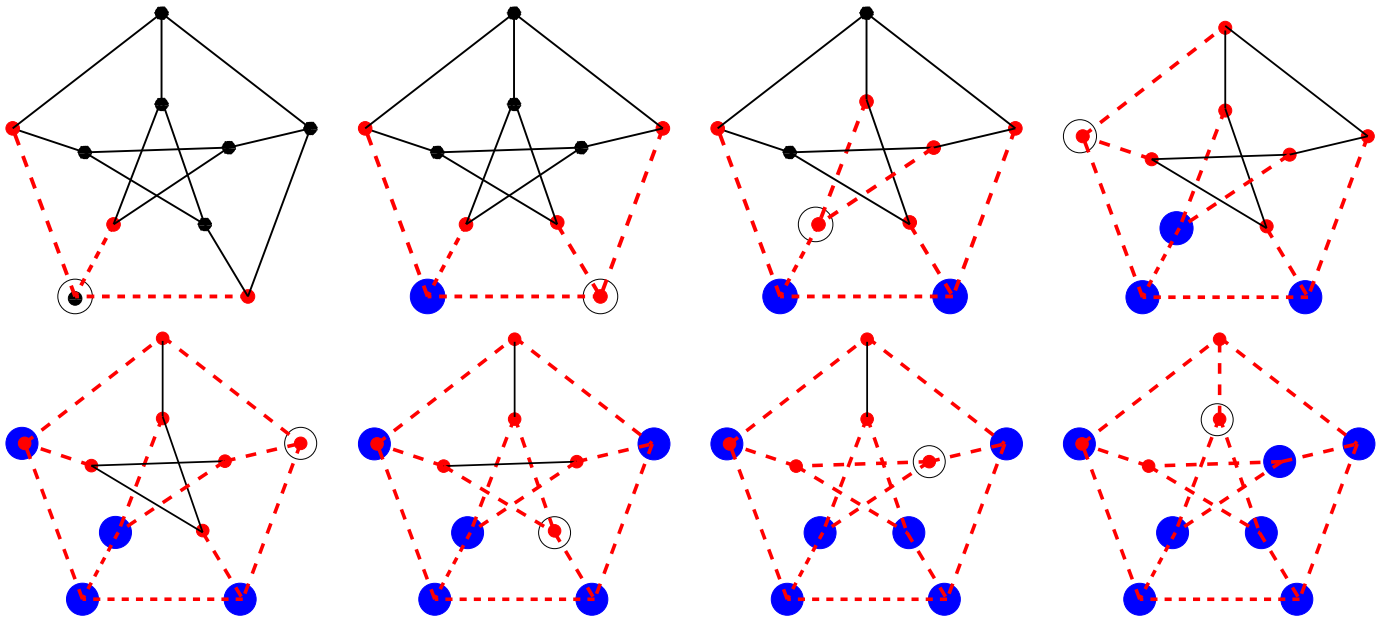
Zakroužkovaný vrchol je ten právě zpracovávaný, modré velké puntíky jsou již zpracované uzly, čárkované červené hrany jsou již zpracované a červené drobné uzly jsou ty aktivní (poznají se také podle toho, že do nich již vede některá zpracovaná hrana). Hrany zpracováváme v pořadí orientace proti hodinovým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

Ilustrace prohledávání do šířky:



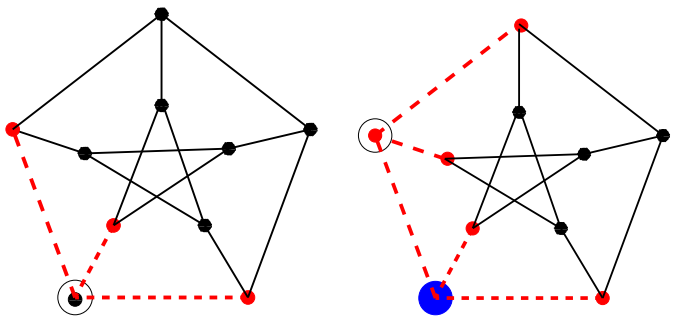
Zakroužkovaný vrchol je ten právě zpracovávaný, modré velké puntíky jsou již zpracované uzly, čárkované červené hrany jsou již zpracované a červené drobné uzly jsou ty aktivní (poznají se také podle toho, že do nich již vede některá zpracovaná hrana). Hrany zpracováváme v pořadí orientace proti hodinovým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

Ilustrace prohledávání do šířky:

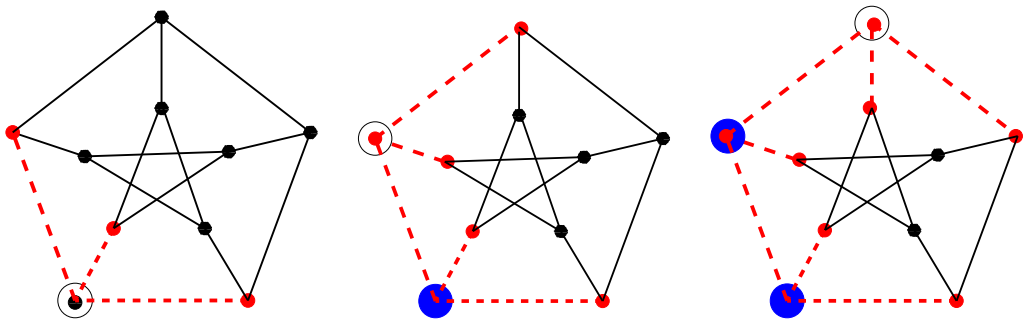


Zakroužkovaný vrchol je ten právě zpracovávaný, modré velké puntíky jsou již zpracované uzly, čárkované červené hrany jsou již zpracované a červené drobné uzly jsou ty aktivní (poznají se také podle toho, že do nich již vede některá zpracovaná hrana). Hrany zpracováváme v pořadí orientace proti hodinovým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

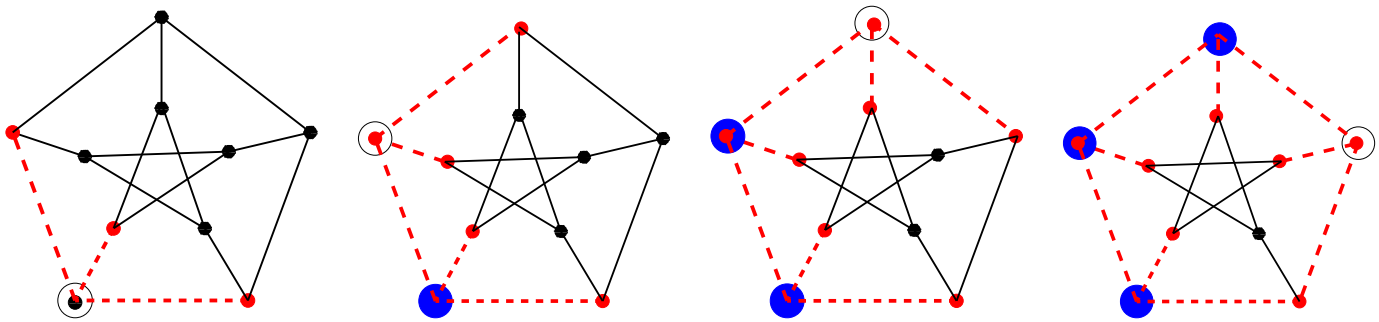
Totéž postupem „do hloubky“. Všimněte si, že první krok je stejný jako v předchozím případě.



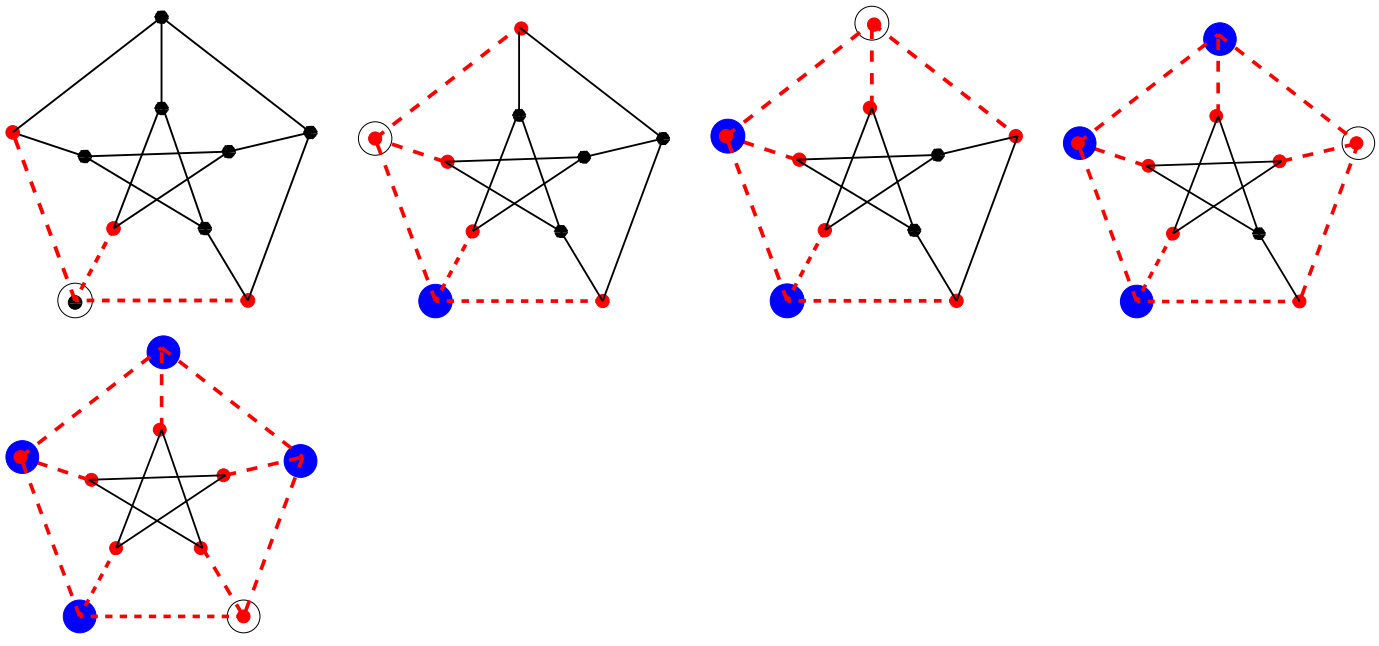
Totéž postupem „do hloubky“. Všimněte si, že první krok je stejný jako v předchozím případě.



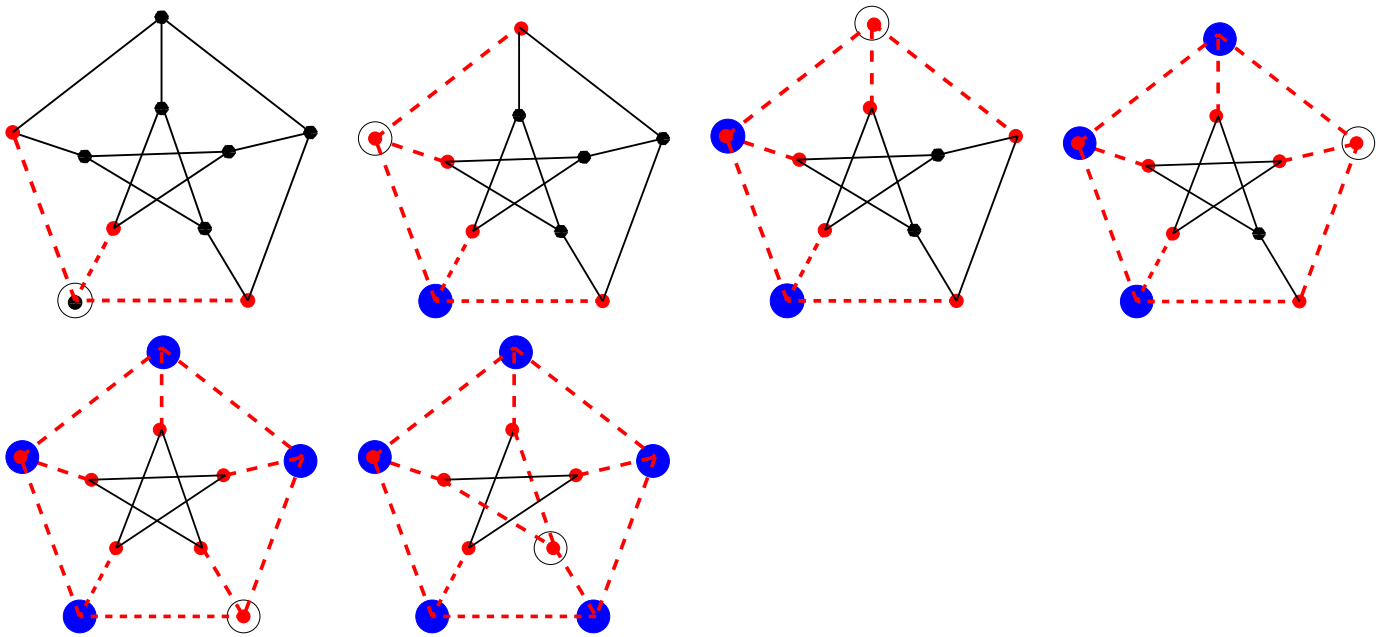
Totéž postupem „do hloubky“. Všimněte si, že první krok je stejný jako v předchozím případě.



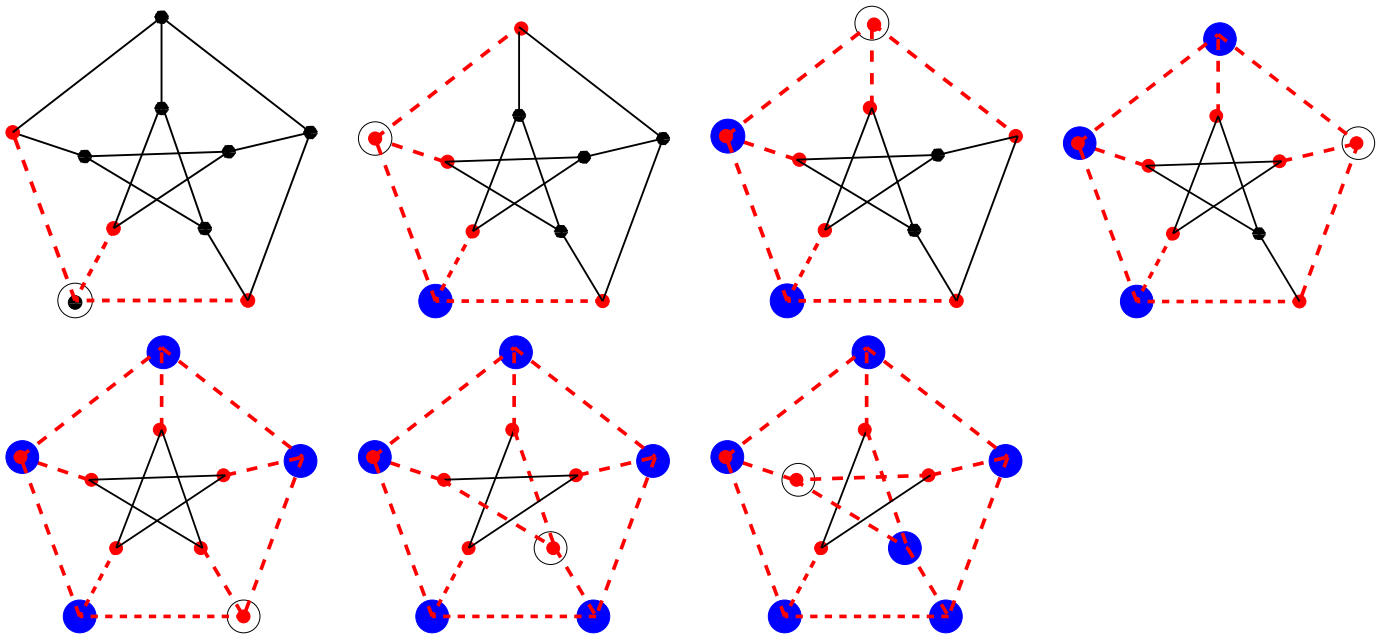
Totéž postupem „do hloubky“. Všimněte si, že první krok je stejný jako v předchozím případě.



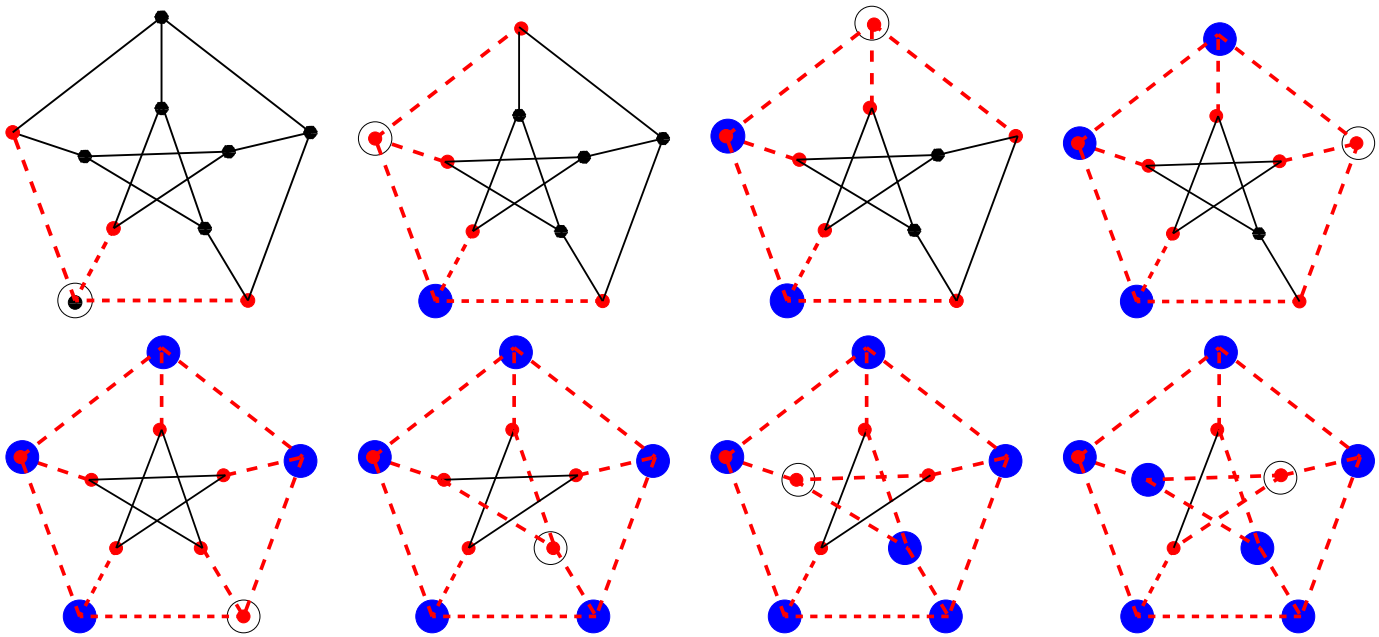
Totéž postupem „do hloubky“. Všimněte si, že první krok je stejný jako v předchozím případě.



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 disjunktní 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 disjunktní 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 disjunktní 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 disjunktní 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

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 $d_G : V \times V \to \mathbb{N}$ 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 $d_G : V \times V \to \mathbb{N}$ 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 $d_G : V \times V \to \mathbb{N}$ 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 celočíslené 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

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 čísly $d_w(v)$, 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 čísly $d_w(v)$, 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 algoristmus 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 algoristmus 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 algoristmus 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 Dijsktrův 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.

Plán přednášky

Jak vypadá dětská hříčka „nakresli obrázek jedním tahem“ v grafech?
V řeči grafů to znamená *najděte sled, který projde všechny hrany a každý vrchol alespoň jednou*.

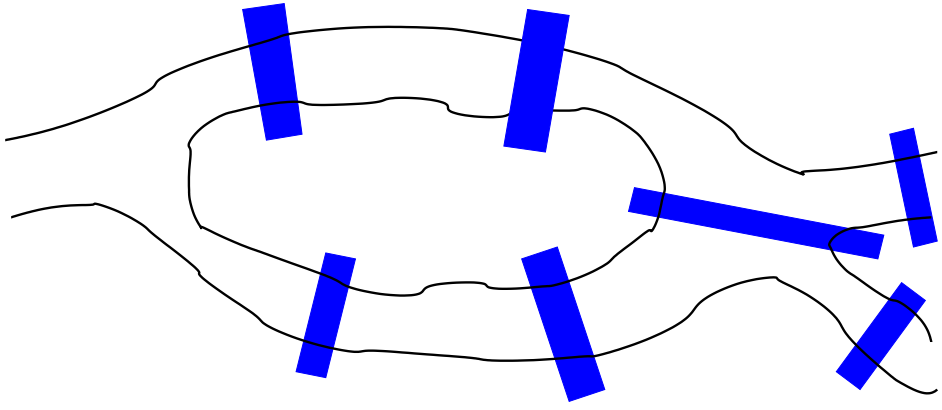
Jak vypadá dětská hříčka „nakresli obrázek jedním tahem“ v grafech?
V řeči grafů to znamená *najděte sled, který projde všechny hrany a každý vrchol alespoň jednou*.

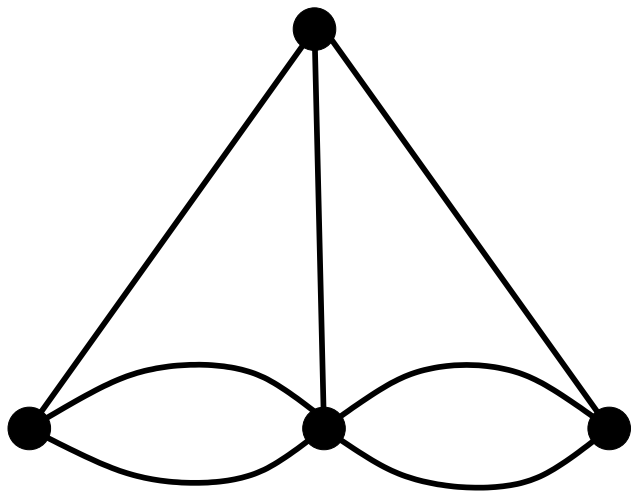
Definition

Sledům, který prochází právě jednou všemi hranami a začíná a končí v jednom vrcholu, se nazývá **uzavřený eulerovský sled**.
Grafům, které takový sled připouští říkáme **eulerovské**.

Terminologie odkazuje na klasický příběh o sedmi mostech ve městě Královec (Königsberg, tj. Kaliningrad), které se měly projít na procházce každý právě jednou a důkaz nemožnosti takové procházky od Leonharta Eulera z roku 1736.

Terminologie odkazuje na klasický příběh o sedmi mostech ve městě Královec (Königsberg, tj. Kaliningrad), které se měly projít na procházce každý právě jednou a důkaz nemožnosti takové procházky od Leonharta Eulera z roku 1736.
Situace je znázorněna na obrázku. Nalevo neumělý náčrt řeky s ostrovy a mosty, napravo odpovídající (multi)graf. Vrcholy tohoto grafu odpovídají „souvislé pevnině“, hrany mostům. Pokud by nám vadily násobné hrany mezi vrcholy, stačí rozdělit hrany pomocí nových vrcholů.





Kupodivu je obecné řešení takového problému dosti snadné, jak ukazuje následující věta. Samozřejmě také ukazuje, že se Euler zamýšleným způsobem procházet nemohl.

Kupodivu je obecné řešení takového problému dosti snadné, jak ukazuje následující věta. Samozřejmě také ukazuje, že se Euler zamýšleným způsobem procházet nemohl.

Theorem

Graf G je eulerovský tehdy a jen tehdy, když je souvislý a všechny vrcholy v G mají sudé stupně.

Kupodivu je obecné řešení takového problému dosti snadné, jak ukazuje následující věta. Samozřejmě také ukazuje, že se Euler zamýšleným způsobem procházet nemohl.

Theorem

Graf G je eulerovský tehdy a jen tehdy, když je souvislý a všechny vrcholy v G mají sudé stupně.

Corollary

Graf lze nakreslit jedním tahem právě, když má všechny stupně vrcholů sudé nebo když existují pravě dva vrcholy se stupněm lichým.

Obdobný požadavek na průchod grafem, ovšem tak, abychom prošli právě jednou každým vrcholem (tj. zároveň nejvýše jednou každou hranou), vede na obtížné problémy. Takový průchod grafem je realizován kružnicí, která obsahuje všechny vrcholy grafu G, hovoříme o **hamiltonovských kružnicích** v grafu G. Graf se nazývá hamiltonovský, jestliže má hamiltonovskou kružnici. Lze ukázat, že neexistuje algoristmus, který by v polynomiálním čase rozhodnul, zda je graf hamiltonovský.

Obdobný požadavek na průchod grafem, ovšem tak, abychom prošli právě jednou každým vrcholem (tj. zároveň nejvýše jednou každou hranou), vede na obtížné problémy. Takový průchod grafem je realizován kružnicí, která obsahuje všechny vrcholy grafu G, hovoříme o **hamiltonovských kružnicích** v grafu G. Graf se nazývá hamiltonovský, jestliže má hamiltonovskou kružnici. Lze ukázat, že neexistuje algoristmus, který by v polynomiálním čase rozhodnul, zda je graf hamiltonovský.
Problém nalezení hamiltonovské kružnice je podstatou mnoha problémů v logistice, tj. například když řešíme optimální cesty při dodávkách zboží.