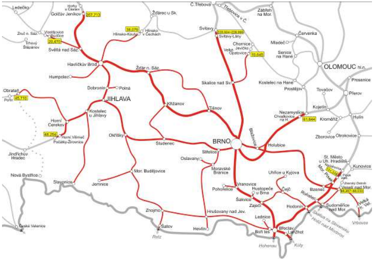
# 8 Procházení grafu a odvozené úlohy

Nyní se hlouběji podíváme na grafy z programátorské perspektivy: podíváme se na obecné schéma procházení grafu, které je základem mnoha užitečných algoritmů na grafech. Poté se hlouběji zaměříme na dvě specifické grafové úlohy – hledání nejkratší cesty a minimální kostry.




## Stručný přehled lekce

* Obecné schéma procházení grafem a jeho varianty.
* Nejkratší cesta v grafu a Dijkstrův algoritmus.
* Minimální kostra grafu a její základní algoritmy.

## 8.1 Jak obecně projít souvislý graf

**Metoda 8.1. Schéma algoritmu pro procházení grafem**

*Pro vytvoření co nejobecnějšího schématu si pomůžeme následujícími:*

- **Vrchol** grafu: má stavy ... □
  - ∗ iniciační – dostane na začátku,
  - ∗ nalezený – implicitní stav poté, co jsme jej přes některou hranu nalezli (a odložili ke zpracování později), □
  - ∗ zpracovaný – poté, co jsme už probrali všechny hrany z něj vycházející,
  - ∗ (příp. ještě stav „post-zpracovaný“, po dokončení všech následníků).□
- **Úschovna**: je pomocná datová struktura (množina s dodatečnými atributy),
  - ∗ udržuje odložené, tj. nalezené a ještě nezpracované vrcholy, spolu s dodatečnou specifickou informací. □
- Způsob, kterým se vybírají vrcholy z úschovny ke zpracování, určuje variantu algoritmu procházení grafu.
- V prohledávaných vrcholech a hranách se volitelně provádějí *dodatečné programové akce pro prohledání a zpracování* našeho grafu.

## Algoritmus 8.2. Generické procházení souvislé komponenty $G$ grafu

- Vstup: Souvislý graf $G$, daný seznamem vrcholů a seznamy vycházejících hran z každého vrcholu, plus případné ohodnocení. □
- Vybereme lib. počátek prohledávání $u \in V(G)$; úschovna $U \leftarrow \{u\}$. □
- Dokud $U \neq \emptyset$, opakujeme:
  - Zvolíme libovolně $v \in U$; odebereme $U \leftarrow U \setminus \{v\}$. (!)□
  - Pokud stav$(v) =$ zpracovaný, jdeme zpět na start cyklu. (*)
  - Případně provedeme libovolnou akci ZPRACUJ(v). □
  - Pro všechny hrany $f \in E(G)$ vycházející z $v$ provedeme:
    - Nechť $w$ je druhý konec hrany $f = vw$;
    - pokud stav$(w) \neq$ zpracovaný, odložíme $U \leftarrow U \cup \{w\}$. (**)□
  - stav$(v) \leftarrow$ zpracovaný; na start cyklu. □
- Souvislý $G$ je celý prohledaný a zpracovaný.

*Pozor, všimněte se, že v bodě (**) obecně dochází k násobnému ukládání, což v praktické implementaci často obejdeme pouhou změnou „odloženého stavu“.*

## Některé implementace procházení grafu

Jak je vlastně proveden krok (!); „zvolíme libovolně $v \in U$“ ? Právě tato volba je klíčová pro výslednou podobu projití grafu $G$:

- *Procházení „do šířky“, BFS* – úschovna $U$ je implementovaná jako fronta, neboli je voleno $v \in U$ od prvních vrcholů vložených do úschovny. □
- *Procházení „do hloubky“, DFS* – úschovna $U$ je implementovaná jako zásobník, neboli je voleno $v \in U$ od později vložených do úschovny. (Opakované vložení vrcholu $v$ do $U$ jej posune na vršek zásobníku.) □

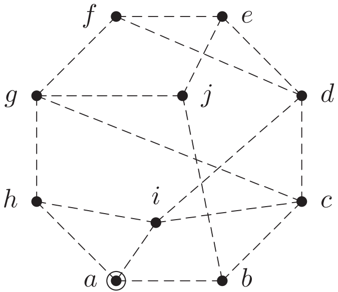
Dále zmíníme i tyto dva konkrétní, staré a dobře známé algoritmy přímo založené na prohledávání grafu:

- *Dijkstrův algoritmus* pro nejkratší cestu – z úschovny vybíráme vždy vrchol nejbližší (dosud určenou celkovou vzáleností) k počátečnímu $u$. □
- *Jarníkův algoritmus* pro minimální kostru – z úschovny vybíráme vždy vrchol nejbližší (délkou hrany) ke kterémukoliv již zpracovanému vrcholu.

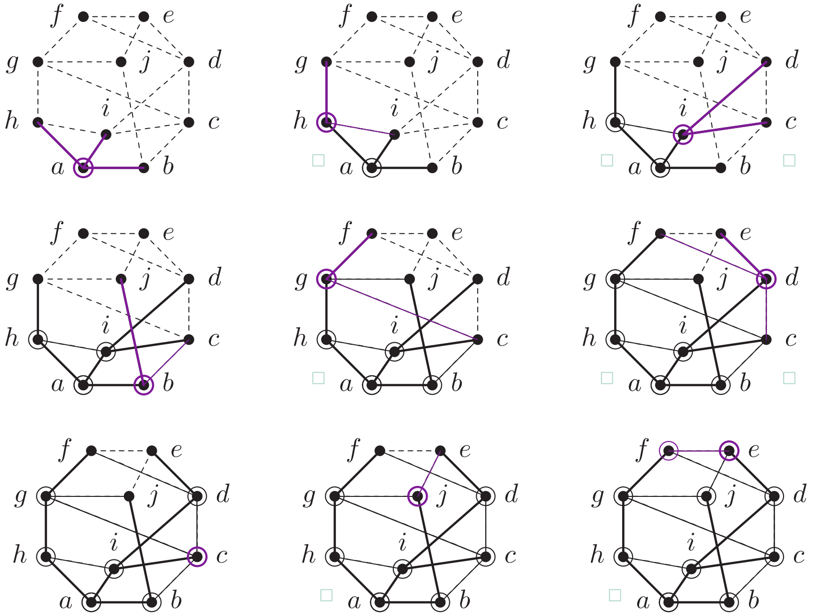
**Poznámka**: Jarníkův algoritmus se ve světové literatuře se obvykle připisuje Američanu Primovi, který jej však objevil a publikoval až skoro 30 let po Jarníkovi.

# Konkrétní ukázky BFS a DFS

**Příklad 8.3.** *Ukázka průchodu následujícím grafem do šířky z vrcholu* $a$.

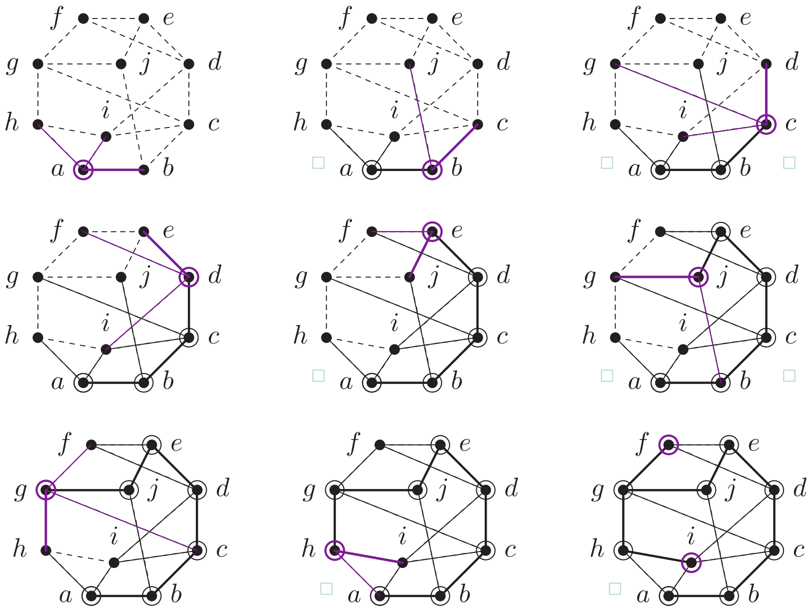


Značení v prohledávaném grafu: barevně aktuálně zpracovávaný vrchol a jeho hrany objevující nové vrcholy, kroužkem a plnou čarou již zpracované vrcholy a hrany.




□

**Příklad 8.4.** *Ukázka průchodu předchozím grafem do hloubky z vrcholu* $a$.

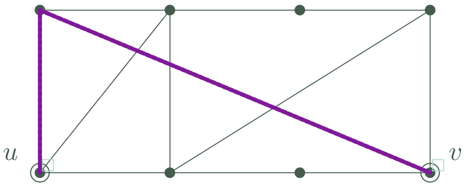



$\square$

## 8.2 Vzdálenost v grafu

**Definice 8.5.** **Vzdálenost** $d_G(u,v)$ dvou vrcholů $u,v$ v grafu $G$ je dána délkou nejkratší cesty mezi $u$ a $v$ v $G$.
Pokud cesta mezi $u,v$ neexistuje, je vzdálenost definována $d_G(u,v)=\infty$. □

Neformálně řečeno, vzdálenost mezi $u,v$ je rovna *nejmenšímu počtu hran*, které musíme překonat, pokud se chceme dostat z $u$ do $v$. Speciálně $d_G(u,u)=0$.




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

**Lema 8.6.** *Vzdálenost v grafech splňuje trojúhelníkovou nerovnost:*

$$\forall u,v,w\in V(G)\ :\quad d_G(u,v)+d_G(v,w)\geq d_G(u,w)\,.$$

## BFS a zjištění vzdálenosti

Jak nejsnadněji určíme vzdálenost v grafu? Stačí si povšimnout hezkých vlastností procházení grafu do šířky.

**Věta 8.7.** *Algoritmus procházení grafu do šířky lze použít pro výpočet grafové vzdálenosti z daného vrcholu $u$.* □

- Toto je poměrně jednoduchá aplikace, kdy počátečnímu vrcholu $u$ přiřadíme vzdálenost $0$, a pak vždy každému dalšímu nalezenému vrcholu $v$ přiřadíme vzdálenost o $1$ větší než byla vzdálenost vrcholu, ze kterého byl nalezen. □

**Důkaz** se opírá o následující tvrzení:

* Nechť $u, v, w$ jsou vrcholy souvislého grafu $G$ takové, že $d_G(u, v) < d_G(u, w)$. Pak při algoritmu procházení grafu $G$ do šířky z vrcholu $u$ je vrchol $v$ nalezen dříve než vrchol $w$. □

V důkaze postupujeme indukcí podle vzdálenosti $d_G(u, v)$... □

## 8.3 Hledání nejkratší cesty

**Definice**: Vážený graf je graf $G$ spolu s ohodnocením $w$ hran reálnými čísly

$$w : E(G) \to \mathbb{R}\,.$$

*Kladně vážený graf* $(G, w)$ je takový, že $w(e) > 0$ pro všechny hrany $e$. □

**Definice 8.8. (vážená vzdálenost)** Mějme (kladně) vážený graf $(G, w)$. Váženou délkou cesty $P$ je

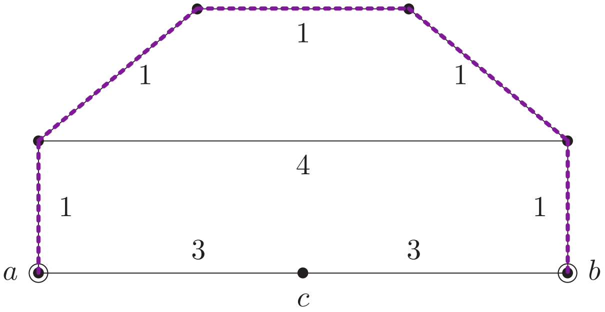
$$d_G^w(P) = \sum\nolimits_{e \in E(P)} w(e)\,.$$

*Váženou vzdáleností* v $(G, w)$ mezi dvěma vrcholy $u, v$ pak je

$$d_G^w(u, v) = \min\{d_G^w(P) : P \text{ je cesta s konci } u, v\}\,.$$ □

**Lema 8.9.** *Vážená vzdálenost v nezáporně vážených grafech (i orientovaných grafech) splňuje trojúhelníkovou nerovnost.*

**Příklad 8.10.** *Podívejme se na následující ohodnocený graf (čísla u hran udávají jejich váhy–délky.)*

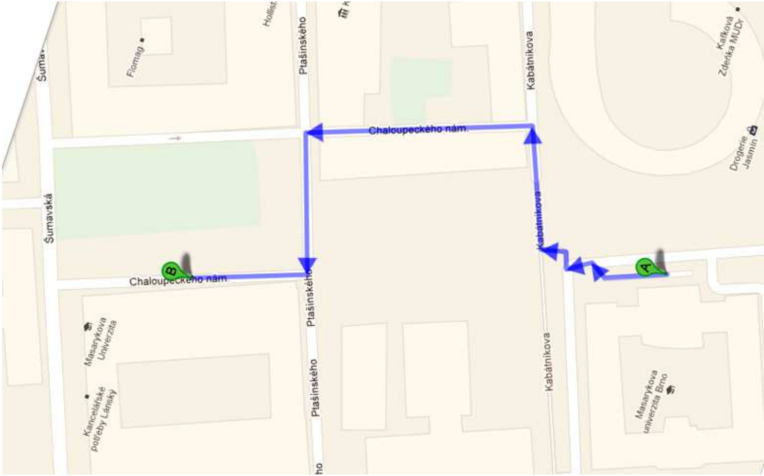



Vzdálenost mezi vrcholy $a, c$ je $3$, stejně tak mezi $b, c$. Co ale mezi $a, b$? Je jejich vzdálenost $6$? Kdepak, vzdálenost $a, b$ je $5$, její cesta vede po „horních" vrcholech, jak je vyznačeno.

Povšimněte si, že tento příklad zároveň ukazuje, že postup prohledáváním do šířky není korektní pro hledání vzdáleností ve váženém grafu. □

## Problém nejkratší cesty

Jedná se patrně o nejznámější „grafový“ problém v praktických aplikacích, jenž nalezneme od vyhledávání dopravních spojení, GPS navigací, plánování pohybů robota, až po třeba rozhodovací systémy.



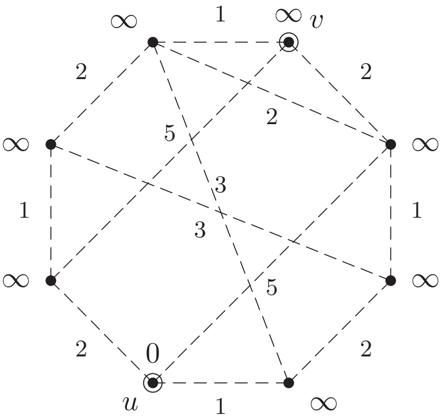
## Dijkstrův algoritmus

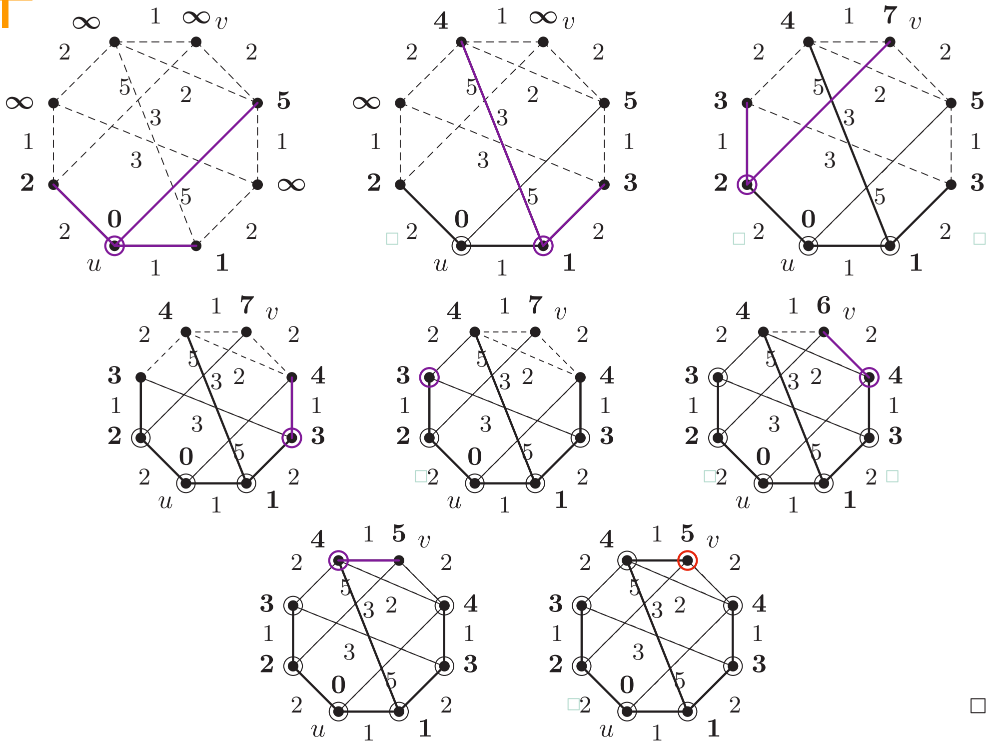
**Algoritmus 8.11. Hledání nejkratší cesty mezi $u$ a $v$ v kladně váž. grafu.**

- Vstup: Souvislý graf $G$, daný seznamem vrcholů a seznamy vycházejících hran z každého vrcholu, plus váhy $w$ hran. Počáteční vrchol $u$ a koncový $v$.
- Úschovna $U \leftarrow \{(u, d(u) = 0)\}$.
- Dokud $U \neq \emptyset$, opakujeme:
  - ∗ Zvolíme $(x, d(x)) \in U$ takové, že $d(x)$ je *minimální*. Odebereme $U \leftarrow U \setminus \{(x, d(x))\}$.
  - ∗ Pokud $x = v$, algoritmus může skončit.
  - ∗ Pro všechny hrany $f \in E(G)$ vycházející z $x$ provedeme:
    - Nechť $y$ je druhý konec hrany $f = xy$; a nechť $d'(y) = d(x) + w(f)$ (nová cesta do $y$ přes $x$).
    - Pokud $(y, d(y)) \notin U$, nebo $(y, d(y)) \in U$ pro $d(y) > d'(y)$, odložíme $U \leftarrow \big(U \setminus \{(y, d(y))\}\big) \cup \{(y, d'(y))\}$ (výměna za novou, lepší dočasnou vzdálenost do $y$).
- Výstup: $d(v)$ udává váženou vzdálenost z $u$ do $v$.

Klíčem k pochopení činnosti Dijkstrova algoritmu je „uvidět“, že v každé jeho fázi jsou správně nalezeny všechny nejkratší cesty z $u$ vedoucí po zpracovaných vrcholech. Postupem prohledávání grafu se tak jednou dostaneme až k určení správné vzdálenosti cíle $v$. □

**Příklad 8.12.** *Ukázka běhu Dijkstrova Algoritmu 8.11 pro nalezení nejkratší cesty mezi vrcholy $u, v$ v následujícím váženém grafu.*




∞ ∞ v ∞ 5 2 ∞ 0 u 1
4 ∞ v ∞ 5 2 3 0 u 1
4 7 v 3 5 2 3 0 u 1
4 7 v 3 4 2 3 0 u 1
4 7 v 3 4 2 3 0 u 1
4 6 v 3 4 2 3 0 u 1
4 5 v 3 4 2 3 0 u 1
4 5 v 3 4 2 3 0 u 1

## Důkaz správnosti

**Věta 8.13.** *Dijkstrův Algoritmus 8.11 pro kladně vážený graf* $(G, w)$ *vždy správně najde nejkratší cestu mezi danými vrcholy* $u, v$. □

**Důkaz** vede přes následující zesílené tvrzení indukcí:

- V každé iteraci Algoritmu 8.11 hodnota $d(x)$ udává nejkratší vzdálenost z vrcholu $u$ do $x$ při cestě pouze po už zpracovaných vnitřních vrcholech.

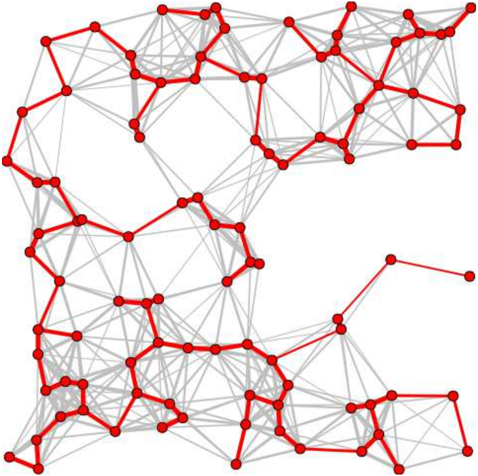
V bázi indukce dovolujeme pouze cesty používající $u$ a $x$, tj. jen hrany vycházející z $u$. Ty jsou v první iteraci algoritmu probrány a jejich délky uloženy do $U$. □

V každém dalším kroku je vybrán jako vrchol $x$ ke zpracování ten, který má ze všech nezpracovaných vrcholů nejkratší nalezenou vzdálenost od počátku $u$. V tom okamžiku je $d(x)$ platnou vzdáleností do $x$, neboť jakákoliv cesta přes jiný nezpracovaný vrchol nemůže být kratší díky nezápornosti vah $w$.

Z toho pak vyplývá, že zpracování vrcholu $x$ správně upraví dočasné vzdálenosti odložené do $U$. Důkaz indukcí je hotov. □

## 8.4 Problém minimální kostry

V tomto případě nebudeme hledat nejkratší spojení mezi dvojicí vrcholů, ale mezi všemi vrcholy najednou – této úloze se říká minimální kostra neboli MST („minimum spanning tree“).



V návaznosti na Oddíl 7.4 definujeme toto:

**Definice**: Podgraf $T \subseteq G$ souvislého grafu $G$ se nazývá *kostrou*, pokud

* $T$ je stromem a
* $V(T) = V(G)$, neboli $T$ propojuje všechny vrcholy $G$. □

*Váhou* (délkou) kostry $T \subseteq G$ váženého souvislého grafu $(G, w)$ rozumíme

$$d_G^w(T) = \sum_{e \in E(T)} w(e)\,.\square$$

**Definice 8.14. Problém minimální kostry (MST)** ve váž. grafu $(G, w)$ hledá kostru $T \subseteq G$ s nejmenší možnou vahou (přes všechny kostry grafu $G$).□

Problém minimální kostry je ve skutečnosti historicky úzce svázán s jižní Moravou a Brnem, konkrétně s elektrifikací jihomoravských vesnic ve dvacátých letech! Právě na základě tohoto praktického optimalizačního problému brněnský matematik Otakar Borůvka jako první podal řešení problému minimální kostry v roce 1928.

## Hladové řešení minimální kostry

**Metoda 8.15. Hladový postup pro minimální kostru grafu $(G, w)$.**
*Mějme dán souvislý vážený graf $G$ s ohodnocením hran $w$.*

- Seřadíme všechny hrany $G$ jako $E(G) = (e_1, e_2, \ldots, e_m)$ tak, že $w(e_1) \leq w(e_2) \leq \cdots \leq w(e_m)$. □
- Inicializujeme prázdnou kostru $T = (V(G), \emptyset)$.
- Po řadě pro $i = 1, 2, \ldots, m$ provedeme následující:
  - ∗ Pokud $T + e_i$ nevytváří kružnici, tak $E(T) \leftarrow E(T) \cup \{e_i\}$.
    (Neboli pokud $e_i$ spojuje různé komponenty souvislosti dosavadního $T$.) □
- Na konci $T$ obsahuje minimální kostru grafu $G$.

**Věta 8.16.** *Hladový postup korektně spočítá minimální kostru grafu $(G, w)$.* □

**Důkaz** (náznak): Pro spor předpokládejme, že $T_1$ je kostra spočítaná Metodou 8.15 a $T_2$ nějaká minimální kostra „blízká“ $T_1$, kde $d_G^w(T_2) < d_G^w(T_1)$. Nyní najdeme jinou min. kostru $T_3$ „bližší“ $T_1$, což je spor s volbou $T_2$. □

## Jarníkův (Primův) algoritmus

**Algoritmus 8.17. Hledání minimální kostry ve váž. grafu $(G, w)$.**
*Níže uvedená specifická implementace procházení grafu využívá úschovnu rozšířeným způsobem, kdy ukládá i příchozí hranu do vrcholu.*

- Vstup: Souvislý graf $G$, daný seznamem vrcholů a seznamy vycházejících hran z každého vrcholu, plus váhy $w$ hran. $\Box$
- Vybereme lib. počátek prohledávání $u \in V(G)$; úschovna $U \leftarrow \big\{(u, \emptyset)\big\}$. Kostra $T = (V(G), \emptyset)$. $\Box$
- Dokud $U \neq \emptyset$, opakujeme:
  - ∗ Zvolíme $(x, e) \in U$ takové, že $w(e)$ je minimální (kde $w(\emptyset) = 0$). Odebereme $U \leftarrow U \setminus \big\{(x, e)\big\}$. $\Box$
  - ∗ Přidáme $E(T) \leftarrow E(T) \cup \{e\}$ (nová hrana do budoucí kostry). $\Box$
  - ∗ Pro všechny hrany $f \in E(G)$ vycházející z $x$ provedeme:
    - – Nechť $y$ je druhý konec hrany $f = xy$.
    - – Pokud $(y, f') \notin U$, nebo $(y, f') \in U$ pro nějaké $w(f') > w(f)$, odložíme $U \leftarrow \big(U \setminus \{(y, f')\}\big) \cup \big\{(y, f)\big\}$ $\Box$
- Výstup: $T$ udává výslednou minimální kostru.