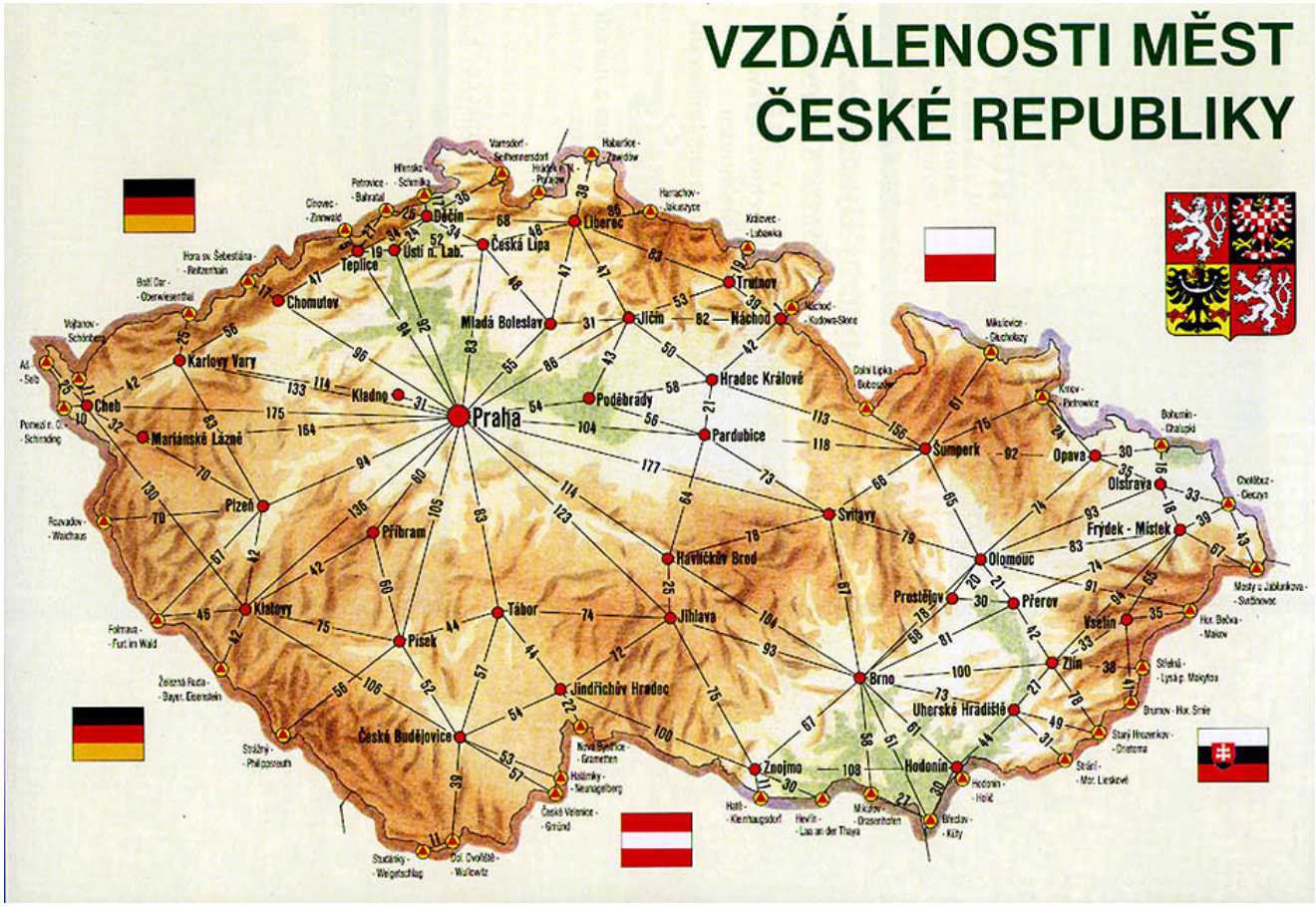
Dva algoritmy související s vzdáleností

- Jak se nejlépe dostat z jednoho vrcholu do druhého?
- Jak nejlépe pospojovat vrcholy dohromady?

Aplikace: doprava, budování sítí, počítačové grafice, taxonomii...

... pro tyto účely je vhodné doplnit informaci o délkách hran.



Formulace obou problémů

Definice: *Hranové ohodnocení* grafu G je zobrazení $w : E \to \mathbb{R}^+$.

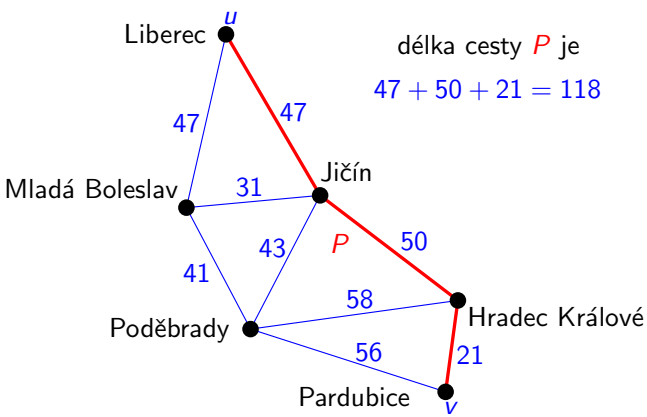
Definice: *Délka* cesty P v hranově ohodnoceném grafu je součet ohodnocení hran použitých v P.

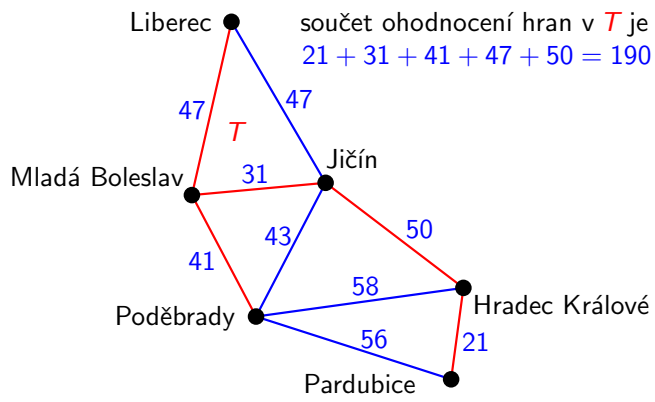
Problém: *Nejkratší cesta* z u do v, určí v grafu G délku nejkratší možné cesty P mezi těmito dvěma vrcholy.

Definice: *Kostra* grafu G je strom T, který je podgrafem grafu G a přitom obsahuje všechny vrcholy G.

Problém: *Minimální kostra*, neboli kostra T grafu G taková, že součet ohodnocení hran v T je nejmenší možný.

Ukázky:





Nejkratší cesta — Dijkstrův algoritmus

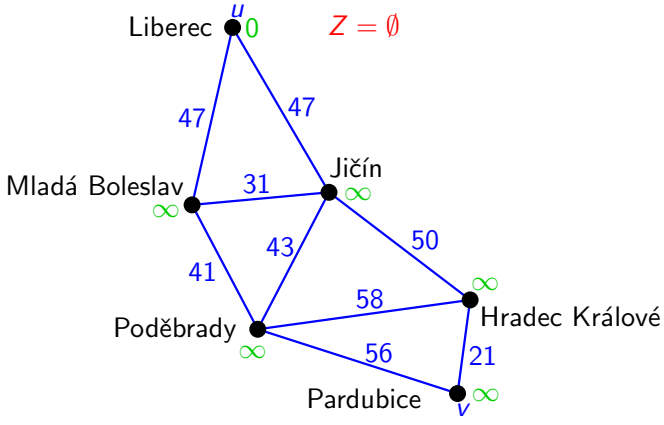
- Po celou dobu algoritmu si u každého vrcholu udržujeme horní odhad d na vzdálenost do u.
- Postupně budujeme množinu „zpracovaných“ vrcholů Z, v níž jsou ty, u nichž už známe přesnou vzdálenost z u.
- Na začátku je u všech vrcholů nastaven odhad d na ∞ kromě výchozího vrcholu $d(u) = 0$, a také $Z = \emptyset$.
- V každé iteraci vybereme mezi doposud nezpracovanými vrcholy $V \setminus Z$ vrchol x s minimální hodnotou $d(x)$.
- Tento x zpracujeme tak, že pro každý jeho nezpracovaný soused $y \notin Z$ spočítáme odhad do y cesty přes x, a pokud je tento odhad lepší, tak jej aktualizujeme.
  Formálně $d(y) := \min\{d(y), d(x) + w(x, y)\}$
- Opakujeme, dokud není zpracován i vrchol v.

Pseudokód Dijkstrova algoritmu

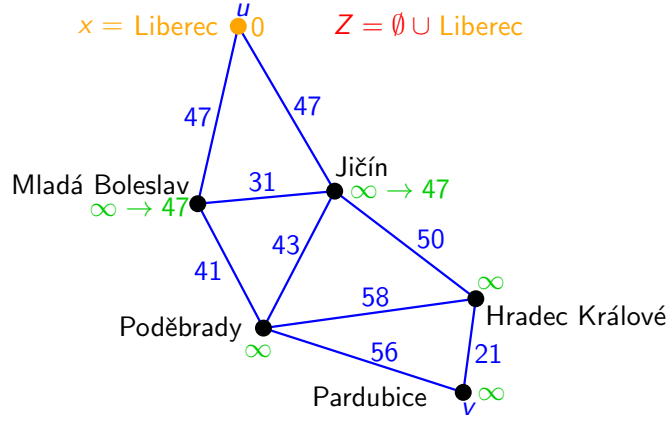
Input: Souvislý G, hranové ohodnocení w, vrcholy $u, v \in V$
Output: Délka nejkratší cesty z u do v.
begin
  $Z := \emptyset$; $d(u) := 0$;
  foreach $z \neq u$ **do** $d(z) := \infty$ `/* inicializace */`
  repeat
    zvol $x \in V \setminus Z$, že $d(x) = \min\{d(z) : z \in V \setminus Z\}$;
    `/* volba vrcholu` x `ke zpracování */`
    foreach $y \in \{z : (x, z) \in E\} \setminus Z$ **do**
      `/*` y `je nezpracovaný soused` x `*/`
      $d(y) := \min\{d(y), d(x) + w(x, y)\}$
      `/* případné zkrácení odhadu` d `pro` y `*/`
    end
    $Z := Z \cup x$ `/*` x `je nyní zpracované */`
  until $v \in Z$;
  return „*Délka nejkratší cesty z* u *do* v *je* $d(v)$.“
end

Ukázka (část)

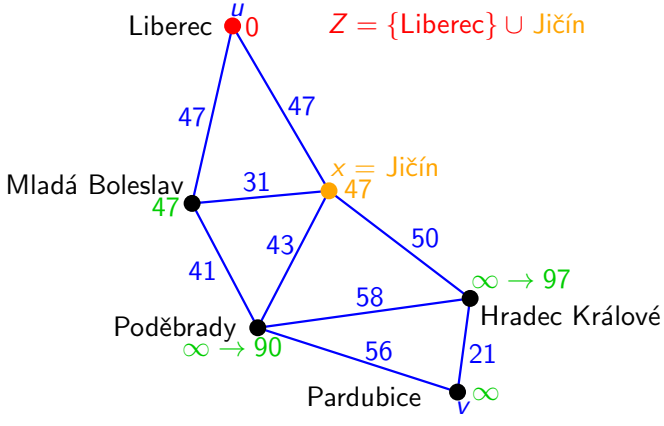
Inicializace



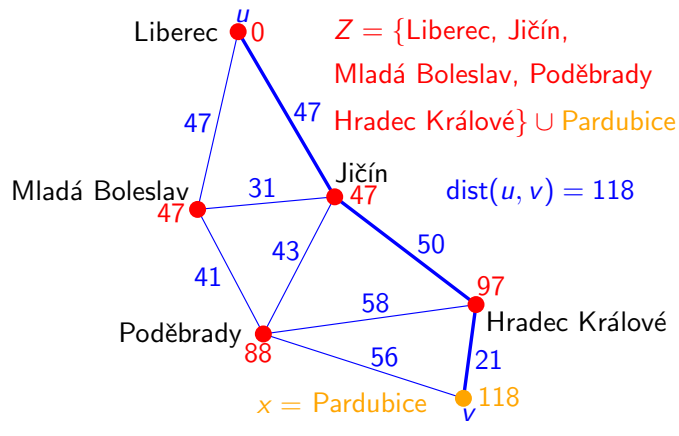
1. iterace



2. iterace

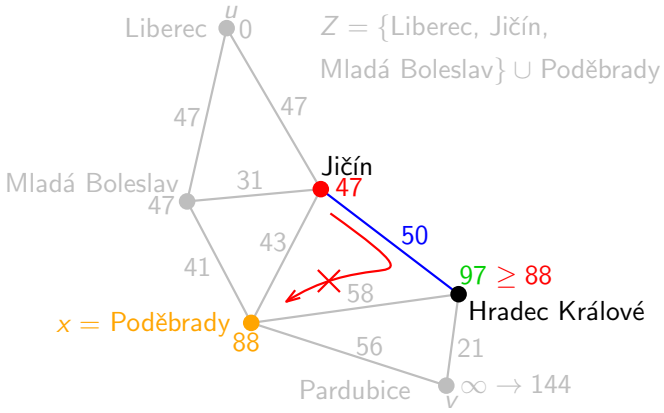


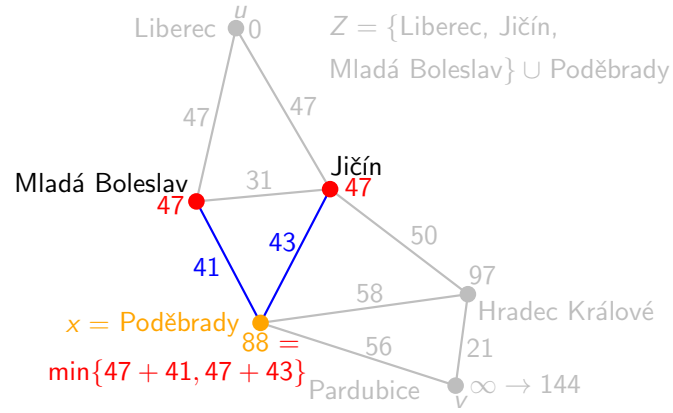
. . . poslední iterace



Korektnost Dijkstrova algoritmu

- Množina Z indukuje souvislý podgraf a zpracovávaný vrchol je v sousedství Z, protože v sousedství Z má d konečné hodnoty.
- Algoritmus je konečný, protože v každém kroku zpracujeme alespoň jeden vrchol.
- Do zpracovávaného vrcholu x už nemůže vést kratší cesta přes doposud nezpracované vrcholy, protože už podél hrany, kdy cesta opustí množinu Z, nemůže být její délka kratší.
- Ani jen přes dříve zpracované nemůže vést do x kratší cesta, protože v době zpracování x platí:
  $d(x) = \min\{d(y) + w(x, y) \colon y \in Z \wedge (x, y) \in E\}$.
- Hodnota $d(x)$ pro $x \in Z$ je délka nejkratší cesty z u do x.





Minimální kostra — Kruskalův algoritmus

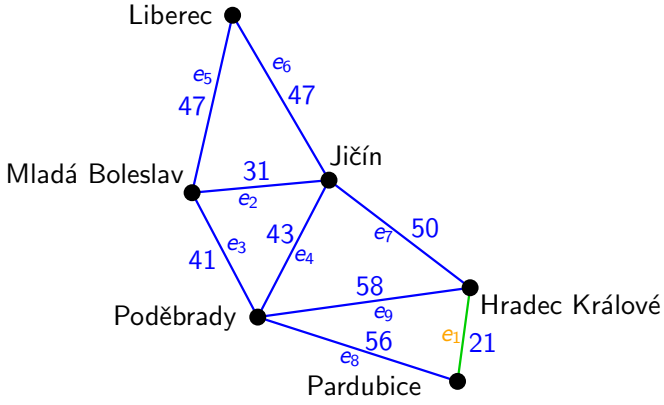
- Začínáme s prázdným podgrafem a postupně do něj přidáváme hrany.
- Hrany probíráme od těch s nejmenší vahou k těm s největší.
- Pokud by přidávaná hrana s doposud vybranými hranami vytvořila cyklus, tak ji do kostry nezahrneme, jinak ano.
- Opakujeme tak dlouho, dokud neprobereme všechny hrany. Cyklus lze ukončit i dříve, jakmile je vybraný podgraf souvislý.

Pseudokód Kruskalova algoritmu

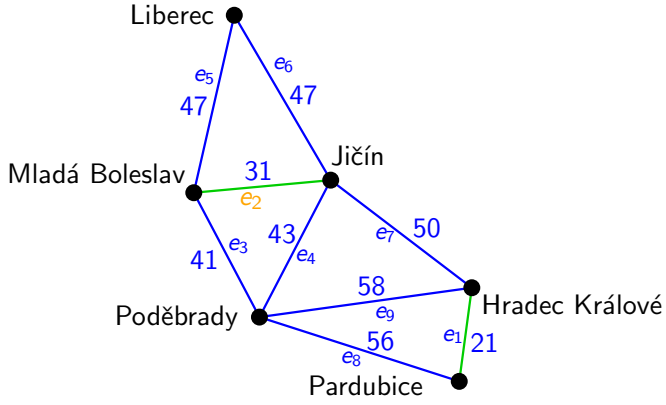
Input: Souvislý graf G, hranové ohodnocení w
Output: Minimální kostra T.
begin
    $V_T := V_G$; $E_T := \emptyset$; $k := |E_G|$; `/* inicializace */`
    Setřiď hrany E_G tak, že $w(e_1) \leq w(e_2) \leq \cdots \leq w(e_k)$;
    for $i = 1$ **to** k **do**
        if *podgraf daný* $E_T \cup e_i$ *neobsahuje cyklus* **then**
            $E_T := E_T \cup e_i$
                `/* přidáme hranu` e_i`, je-li to možné */`
        end
    end
    return „*Minimální kostra je* T.“
end

Ukázka (část)

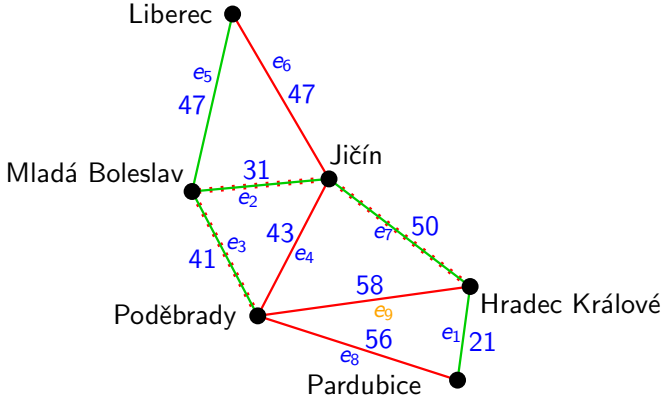
1. iterace — cyklus nevzniká



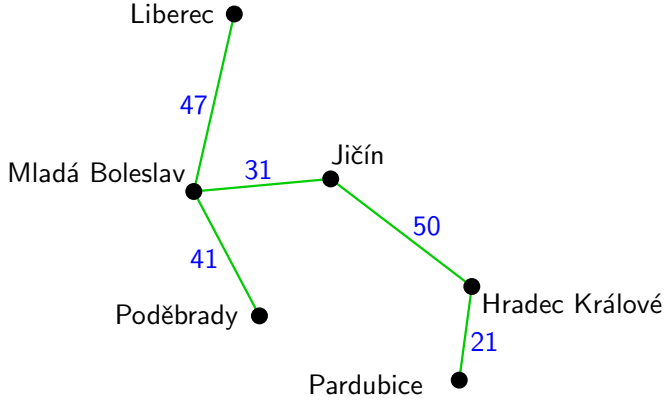
2. iterace — cyklus nevzniká



. . . 9. iterace — cyklus vzniká

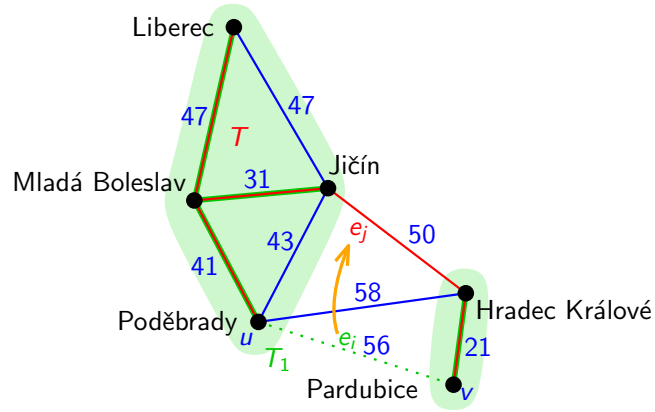


Výsledná minimální kostra



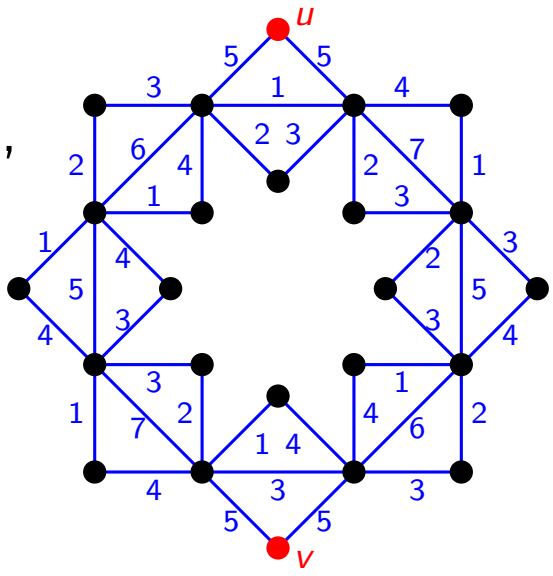
Korektnost Kruskalova algoritmu

- Výsledný podgraf T nemá cyklus.
- Také je souvislý. Kdyby nebyl, pak by hrana nejmenší váhy mezi dvěma komponentami nevytvořila cyklus a měla by být vybrána.



- Minimalita součtu ohodnocení hran v T:
  - Je-li T_1 jiná kostra než T, uvažme hranu e_i s nejmenším indexem, náležící T_1, ale nikoli T. Označme ji $e_i = (u, v)$.
  - V okamžiku testování e_i v T vznikne cyklus, čili z u do v vede v T cesta, jejíž hrany jsou ohodnoceny nanejvýš $w(e_i)$.
  - Z této cesty alespoň jedna hrana e_j s $j < i$ ovšem spojuje také dvě komponenty, které vzniknou odebráním e_i z T_1.
  - Nahrazením hrany e_i v T_1 za e_j získáme kostru T_2. Díky $w(e_j) \leq w(e_i)$ nevzroste v T_2 součet ohodnocení hran.
  - Kostra T_2 má více hran společných s T než měla T_1. Opakováním tohoto postupu dostaneme po nejvýše $|V| - 1$ krocích kostru T, což zaručuje její minimalitu.

Kvíz

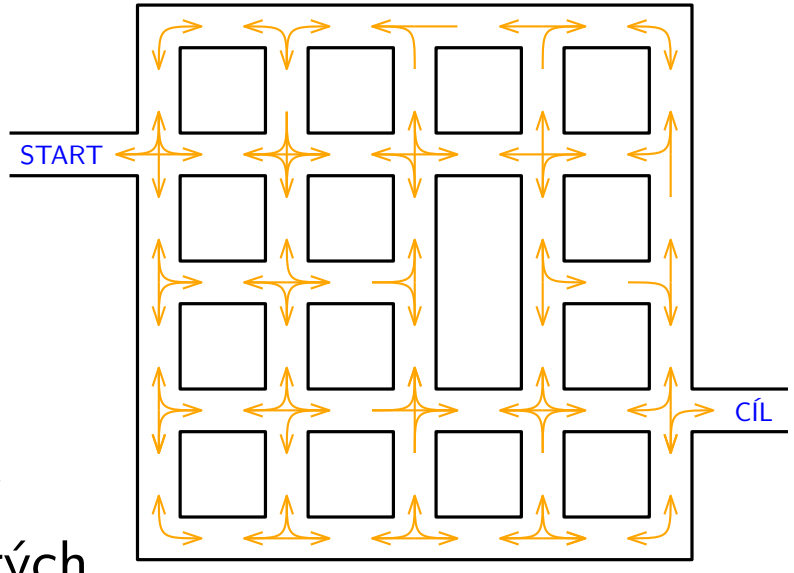


Je-li u některých otázek více možností správných, vyberte všechny.

1. Kolik nejkratších cest vede mezi vrcholy u a v v následujícím grafu?
   a) 6, b) 8, c) 12, d) 16, e) 24, f) 36, g) 64.
2. Kdybychom v Dijkstrově algoritmu počítali $d(y) := \min\{d(y), d(x) + w(x, y)\}$ pro *všechny* sousedy x, pak by algoritmus: a) stejně rychle dal správný výsledek, b) dal správný výsledek dřív, c) dal správný výsledek později, d) doběhl, ale s nesprávnými výsledky, e) se mohl zacyklit.
3. Pravda nebo lež? Minimální kostra je určena jednoznačně právě když je ohodnocení hran w prosté zobrazení.
4. Nejvýše ohodnocená hrana grafu G do minimální kostry T
   a) nikdy nenáleží, b) náleží, jen nemá-li G cykly,
   c) může náležet i v jiných případech.

Otázky k porozumění tématu přednášky

- Jak z délek nejkratších cest z u rekonstruovat, o které cesty jde?
- Je řešení vždy jednoznačné?
- Jak souvisí algoritmus na hledání nejkratší cesty s hledáním cesty v bludištích?
- Fungovaly by algoritmy, pokud by ohodnocení w nabývalo na některých hranách záporných hodnot?



- A jak je mohou ovlivni trojice hran, u nichž ohodnocení nesplňuje trojúhelníkovou nerovnost, čili $w(u, v) > w(u, z) + w(z, v)$?

Poznámky k historii

Oba algoritmy, Kruskalův i Dijkstrův pocházejí shodou okolností z roku 1956.

První algoritmus pro minimální kostru sestavil brněnský profesor O. Borůvka v roce 1926 pro efektivní elektrifikaci Moravy.

V reakci na Borůvkův článek navrhl svou variantu i V. Jarník (1930).

Otakar Borůvka
1899 – 1995

Dlužno dodat, že uvedené algorimy byly několikrát znovuobjeveny.