Pro první tři příklady se nám bude hodit následující lemma, které zjednodušeně říká, že pokud máme neunikátní váhy hran w, můžeme je libovolně doseřadit (tj. určit pořadí stejně těžkých hran), a výsledná minimální kostra je jednou z minimálních koster původního grafu.

Lemma. *Mějme ohodnocený graf $G = (V, E, w)$, kde $w : E \to \mathbb{R}^+$ je neunikátní ohodnocení hran. Buď $w' : E \to \mathbb{R}^+$ **unikátní** ohodnocení hran takové, že pro různé hrany e, f platí:*

- *pokud $w(e) < w(f)$ pak $w'(e) < w'(f)$, a*
- *pokud $w(e) = w(f)$ pak $|w'(e) - w'(f)| \leq \epsilon$.*

Potom pro dostatečně malé ϵ platí, že minimální kostra T v ohodnocení w' je jednou z minimálních koster v původním ohodnocení w.

Důkaz. Klíčové je, že váha libovolné kostry T v obou ohodnocení se příliš neliší. Přesně pro libovolnou kostru T platí $w(T) + \epsilon n \geq w'(T) \geq w(T) - \epsilon n$ jelikož T má méně než n hran.

Nechť T je libovolná minimální kostra v ohodnocení w a T' je unikátní minimální kostra v ohodnocení w'. Proč platí nerovnosti $w(T) \leq w(T')$ a $w'(T') \leq w'(T)$? Teď už jen tyto nerovnosti poskládejte s nerovností z předchozího odstavce a dostanete $w(T') \leq w(T) + 2\epsilon n$. Jaká volba ϵ postačí aby tato nerovnost implikovala $w(T') = w(T)$. □

Ve skutečnosti budeme lemma aplikovat pouze „virtuálně“. Uvažme libovolné uspořádání hran $e_1 \prec e_2 \prec \ldots \prec e_m$, které respektuje w, tj. pokud $w(e_i) < w(e_j)$ pak $e_i \prec e_j$. Pro takové uspořádání umíme vyrobit ohodnocení w' splňující podmínky lemmatu a tedy pokud se nějaký algoritmus bude řídit pouze uspořádáním $\prec$ bude se chovat stejně jako kdyby porovnával pomocí w', a tedy speciálně díky lemmatu najde nějakou minimální kostru původního grafu.

1. *Neunikátní Jarník: Fungoval by Jarníkův algoritmus pro neunikátní váhy hran?*

   Jarníkův algoritmus nemůže vytvořit cyklus. Potřebujeme tedy pouze ukázat, že i pro neunikátní váhy najde vždy nějakou minimální kostru. Rozmyslete si, že můžeme hrany lineárně setřídit tak aby uspořádání odpovídalo rozhodnutím Jarníkova algoritmu. Potom už jenom aplikujeme lemma.

2. *Neunikátní Borůvka: Fungoval by Borůvkův algoritmus pro neunikátní váhy hran?*

   Borůvkův algoritmus může pro neunikátní váhy vyrobit graf, který vůbec není strom. Mějme dvě komponenty C_1 a C_2 mezi kterými vedou dvě hrany e_1 a e_2 stejné minimální váhy. Pak Borůvkův algoritmus může vybrat jako nejlehčí hranu z komponenty C_i hranu e_i a tím vytvoří v množině vybraných hran cyklus.

   Řešením je jednoznačně douspořádat hrany před spuštěním Borůvkova algoritmu. Druhým porovnávacím kritériem pro dvě hrany stejné váhy může být například index hrany, nebo lexikografické uspořádání podle jejich koncových vrcholů. Poté už se opět odvoláme na lemma nahoře.

3. *Neunikátní Kruskal: Fungoval by Kruskalův algoritmus pro neunikátní váhy hran?*

   Viz. Jarníkův algoritmus.

4. *Změna hrany: Máte nalezenou minimální kostru. Jak najít novou minimální kostru pokud:*

   (a) Z grafu odstraníme jednu hranu.
   (b) Zvýšíme váhu jedné hrany.
   (c) Snížíme váhu jedné hrany.

V případě (a) se mohlo něco změnit pouze pokud jsme odstranili hranu ležící v minimální kostře. Odstraněním hrany se kostra rozpadne na 2 disjunktní stromy T_1 a T_2, které přirozeně definují řez v grafu. Novou minimální kostru získáte výběrem nejlehčí hrany v tomto řezu.

Případ (b) je podobný. Pokud zvýšíme váhu hrany e v minimální kostře, může se nám vyplatit nepoužít ji. Opět tedy vezmeme kostry na T_1 a T_2 určené e a v řezu mezi nimi hledáme nejlevnější hranu. Tentokrát ale tou hranu může být klidně znovu e.

V případě (c) nás naopak zajímá když snížíme váhu hrany e mimo minimální kostru, jestli se nám vyplatí ji použít. Tentokrát naopak přidáme e ke stávající kostře, čímž dostaneme graf s právně jedním cyklem a z tohoto cyklu odstraníme nejtěžší hranu (opět mohlo jít klidně znovu o původní e).

Rozmyslete si proč se minimální kostra nemohla změnit nějak výrazněji. Pro (a), (b) použijte řezové lemma[1] na řezy definované původní minimální kostrou. Pro (c) si nejprve rozmyslete, že nejtěžší hrana z libovolného cyklu nemůže nikdy být součástí minimální kostry. A poté aplikujte řezové lemma na řezy definované novou minimální kostrou. Větší pozornost věnujte hranám na cyklu utvořeném přidáním upravené hrany do původní kostry.

5. *L-kostry: Vymyslete algoritmus na hledání kostry grafu, v němž jsou váhy hran přirozená čísla z množiny* $\{1, \ldots, L\}$.

   Téměř ekvivalentní problému Dijkstra III. z 5. cvičení. Použijeme Jarníkův algoritmus, ale haldu pro hledání nejlehčí hrany ven nahradíme polem `Q` velikosti L, kde `Q[i]` obsahuje spojový seznam vrcholů do nichž vede minimální hrana váhy právě i. Přidávat nové vrcholy do `Q` umíme v $O(1)$, aktualizaci jejich ohodnocení také v $O(1)$ pokud si u každého vrcholu držíme ukazatel na jeho místo v seznamu, a hledání minima zvládneme pomocí nalezení prvního neprázdného seznamu v čase $O(L)$. Časová složitost tedy bude $O(nL + m)$, paměťová pouze $O(n + m + L)$.

[1] Průvodce labyrintem algoritmů s. 162