Domácí úkol 4:

Mějme na vstupu souvislý graf G s unikátními váhami hran w a předpočítanou minimální kostru T.

Navrhněte algoritmus, který najde (nějakou) druhou nejlehčí kostru.

V tomto úkolu nás bude zajímat především myšlenka návrhu. Chceme tedy hlavně popis postupu, a argumentaci jeho korektnosti. Pseudokód není nutný. Algortitmus by se měl vejít do časové složitosti $\mathcal{O}(m^2)$ (lze přímočaře dosáhnout $\mathcal{O}(nm)$).

Hint: Přímočarý přístup ke korektnosti já třeba následující. Jakákoliv ne-minimální kostra se musí lišit od té minimální alespoň jednou hranou. Každá odlišnost zvyšuje váhu ne-minimální kostry. Rozšířením této úvahy je možné i odvodit celý postup.

Popis: Vyzkoušíme všechny kostry, které vzniknou z minimální výměnou jedné hrany za jinou (se zachováním souvislosti). Ze všech koster vezmeme tu s nejmenší vahou, tedy takovou kde rozdíl vah vyměněných hran je nejmenší.

pseudokód

G je vstupní graf, T je minimální kostra
Pro každou hranu $e \in T$ proveď následující:
  Detekuj komponenty $T - e$ (DFS, ulož id komponenty do vrcholů)
  Projdi všechny hrany f v G spojující (různé) komponenty $T - e$
  Průběžně udržuj pár e, f s minimálním rozdílem $w(f) - w(e)$
Vydej $T - e + f$ nebo chybu pokud žádný pár e, f neexistuje

Korektnost:

Tvrzení: druhá nejlehčí kostra S se liší od T výměnou právě jedné hrany za jinou.

Důkaz1: Zjevně se liší v alespoň jedné hraně. Nechť e je hrana v T, která není v S. Zřejmě S je kostra $G-e$, a navíc minimální, jinak máme spor s volbou S (existovala by lehčí kostra lišící se od T). Uvažme modré lemma, a podívejme se jak se liší modré hrany G a $G - e$. Protože modré hrany jsou minimální na nějakém řezu, odebráním hrany e se všechna minima různá od e nezmění. Všechny modré hrany kromě e v G jsou tedy modré hrany i v $G - e$. Protože S jako minimální kostra $T - e$ musí obsahovat všechny tyto modré hrany, liší se od T maximálně jednou hranou.

Důkaz2: Zjevně se liší v alespoň jedné hraně. Pro spor nechť e, f jsou hrany T, které nejsou v S. $T - e - f$ má tři komponenty a dva řezy mezi nimi na kterých jsou e a f nejlehčí (z modrého lemma), označme e', f' libovolné hrany z S na příslušných řezech (tedy e', f' jsou různé hrany nahrazující e, f v S). Musí platit $w(T-e-f) \leq w(S-e'-f')$, kdyby ne, potom $w(T) > w(S-e'-f'+e+f)$ což by byl spor s minimalitou T. Potom ale z unikátnosti vah $w(e) < w(e')$ a $w(f) < w(f')$. Dostáme následující vztah:

$$w(S) = w(s-e'-f')+w(e')+w(f') > w(S-e'-f')+w(e)+w(f') > w(T-e-f)+w(e)+w(f) = w(T)$$

Tedy $S - e' + e$ je lehčí než S.

Tvrzení: Výstup je korektní. Na výstupu je zjevně kostra, protože uvažujeme pouze páry jejichž výměna zachovává souvislost. Algoritmus uváží všechny páry hran z nichž jedna je uvnitř kostry a druhá vně kostry, zahozeny jsou pouze páry, jejichž výměna nezachovává souvislost. Z tvrzení výše tedy algoritmus musí uvážit i druhou nejlehčí kostru (resp. pár jehož výměnou vznikne). Pár jehož výměnou vznikne bude minimalizovat rozdíl, protože je roven rozdílu váhy T a uvažované kostry. Poznamenejme, že druhá nejlehčí kostra nemusí být unikátní; i pokud jsou váhy hran unikátní, rozdíly už být nemusejí.

Speciální případy nastanou pouze pár s minimálním rozdílem vybíráme z prázdné množiny. T.j. pokud neexistují hrany mimo kostru (G je strom) nebo neexistují hrany uvnitř kostry (G je jednovrcholový). V těchto případech druhá nejlehčí kostra zřejmě neexistuje.

Složitost:

Hlavní cyklus proběhne $\mathcal{O}(n)$-krát. Detekování komponent $T-e$ zvládneme v čase $\mathcal{O}(n+m)$ (dokonce $\mathcal{O}(n)$ pokud máme T na vstupu jako graf, nebo si ho na začátku algoritmu vytvoříme). Na projití všech hran, zbylé výpočty a udržování minim stačí $\mathcal{O}(m)$. Celková složitost $\mathcal{O}(nm)$.

Jiné řešení:

Setřídíme všechny hrany G, a pro každou $e \in T$ najdeme minimální kostru $G-e$ pomocí Kruskalova algoritmu. Vrátíme nejlehčí z nalezených koster.

Korektnost lze dokázat analogicky řešení výše, v důkazu 1 jsme jako mezikrok nahlédli, že hledaná kostra je minimální kostra $G-e$ pro nějaké $e \in T$. Můžeme tedy stejně argumentovat, že

Pro analýzu časové složitosti si všimneme, že stačí hrany setřídit pouze jednou, $\mathcal{O}(mlogn)$ celkem. Jeden Kruskalův algoritmus tedy poběží v čase $\mathcal{O}(n{*}Union{+}m{*}Find)$, spuštěn bude $\mathcal{O}(n)$-krát. Pro přímočarou implementaci a jednoduchý odhad tedy dostáváme celkem $\mathcal{O}(nm\log n)$. Použijeme-li černou magii sofistokované analýzy, můžeme tvrdit $\mathcal{O}(nm\log *n)$ nebo $\mathcal{O}(nmInvAckn)$ (složitost třídění lze zanedbat, protože $m\log n = \mathcal{O}(nm)$).

Podotkněme, že tímto řešením nelze dosáhnout složitosti $\mathcal{O}(nm)$, a řešení je dokonce neporovnatelné s požadavkem zadání $\mathcal{O}(m^2)$; pokud je graf na vstupu řídký (třeba rovinný, s $m = \mathcal{O}(n)$), bude tato složitost asymptoticky vždy mírně větší než $\mathcal{O}(m^2) = \mathcal{O}(n^2)$ (i když pro inverzní Ackermannovu funkci je rozdíl nedetekovatelně malý).