Minimální Kostry

• Při hledání minimálních koster obvykle přepokládáme unikátnost hran. Jak tento předpoklad zajistit, když není splněn na vstupu?
• Jak se vypořádáme se zápornými vahami a multihranami?

Kruskal

• Uvažujme neunikátní váhy. Ukažte, že přijmutí hrany není závislé na konkrétním výběru hran s nižší vahou. Dokonce umíme v lineárním čase říct kolik hran dané váhy bude v každé minimální kostře, jak?
• Jakou časovou složitost má Kruskal, pokud dostane hrany na vstupu již seřazené?

Jarník/Prim

• Navrhněte vhodnout implementaci s použitím haldy. Co chceme v haldě uchovávat?
• Jakou časovou složitost má Jarníkův algoritmus, pokud ho vyzbrojíme vhodnou datovou strukturou?

Borůvka

• Na jaké problémy narazíme při implementaci s haldami?
• Zastavme Borůvkův algoritmus po jedné fázi/kroku a vezměme nalezený les F a kontrahujme ho. Z výsledku chceme také odstranit násobné hrany a izolované vrcholy. Jak toho docílit v lineárním čase? Jak velký je výsledný graf?
• Iterujme předchozí kontrahující Borůvkův krok. Jako výstup vydejme sjednocení nalezených lesů. Je tento algoritmus korektní? Jaká je jeho časová složitost?

Úložky

Neunikátnost Co se rozbije pokud váhy nejsou unikátní?
Diskrétní váhy Pokud naopak máme pouze váhy $1, \ldots, k$, jaké můžeme dosáhnout časové složitosti?
Rovinná kostra Vypusťme naše tři algoritmy na rovinné grafy. Jaká bude jejich časová složitost? Poznámka: stejná úvaha platí pro překvapivě mnoho grafových tříd.
Inverzní Kruskal Mějmě hrany seřazené sestupně. Každou hranu ponecháme pokud neleží na cyklu a jinak ji smažeme. Na výstup dáme vše co zbylo. Je to korektní algoritmus?
Stříbrná kostra Pokud známe minimální kostru, jak najít druhou nejmenší? Dovolme si kvadratický čas, ale jde to i lineárně (velmi těžké).

Domácí úkol

První část

Mějme v grafu vyznačené speciální vrcholy. Jak najdeme nejlehčí kostru ve které jsou všechny tyto vrcholy listy? Dokažte korektnost (ideálně sporem) a určete časovou složitost.

Druhá část

Kontrahující Borůvka je rychlý na řídkých grafech, ale časem je zahušťuje. Jarník může být naopak rychlý na hustých grafech. Zkusme toho zneužít.

Pusťme $loglog(n)$ kontrahujících borůvkových kroků (fází). Na zkontrahovaný zbytek (resp. na každou komponentu) pusťme Jarníkův algoritmus vyzbrojený Fibonacciho haldou. Jako výsledek prohlásíme sjednocení hran vybraných oběma algoritmy. Určete časovou složitost tohoto algoritmu.

Hint: musíme určit velikost vstupu Jarníka. Jeho složitost (s Fib. haldou a na souvislém grafu) je pak $\mathcal{O}(m' + n'logn')$. Počet hran ale nepůjde odhadnout jinak než hrubě.

Nezapomeňte argumentovat, proč můžeme složitost Jarníka spuštěného postupně na několik komponent nesouvislého grafu asymptoticky odhadnout stejně jako jediný běh na souvislém grafu (se stejným počtem vrcholů a hran).

Bonusová část

Pro graf G a jeho (nějakou) fixní kostru $\bar{T}$, hrana $e \notin \bar{T}$ je $\bar{T}$-těžká pokud na jediné kružnici v $\bar{T} \cup e$ je nejtěžší (a $\bar{T}$-lehká jinak, zbytek hran jsou právě hrany $\bar{T}$). Pozorování: pokud je e $\bar{T}$-těžká, pak nepatří do minimální kostry. Mějme přislíbený algoritmus, který pro G a $\bar{T}$ najde v lineárním čase ($\mathcal{O}(m+n)$) všechny $\bar{T}$-těžké hrany.

Definujme rekurzivní algoritmus následovně. Nejprve pustíme dva kontrahující Borůvkovy kroky, čímž dostaneme les F_1 a kontrahovaný graf $\bar{G}$. Pokud $\bar{G}$ je prázdný, vrátíme F_1. Zahodíme z $\bar{G}$ každou hranu s pravděpodobností $1/2$ a na zbytku rekurzivně najdeme kostru (les) $\bar{T}$. Vezmeme opět $\bar{G}$, smažeme z něj všechny $\bar{T}$-těžké hrany, rekurzivně najdeme kostru (les) F_2 a vrátíme $F_1 \cup F_2$.

Tento algoritmus je randomizovaný a v průměru běží lineárně - to je těžké dokázat a zatím ani neumíme formálně říct, co to znamená. Dokažte, že v nejhorším případě algoritmus běží v čase $\mathcal{O}(mlog(n))$, to lze udělat i zcela bez pravděpodobnostních argumentů.

Jak to udělat? Zapomeňme nejprve na Borůvkovy kroky. Všimněme si, že uvnitř každého volání provádíme lineárně práce (nepočítáme rekurze). Dále odhadneme velikosti vstupů obou rekurzí - to uděláme dohromady. Rozdělením hran na tři typy podle vztahu k $\bar{T}$ můžeme nahlédnout, že součet velikostí vstupů je hezky omezen. Nastává čas zazáplatovat algoritmus vložením Borůvkových kroků; a pro kontrolu si můžeme všimnout, že jeden je málo. Zbývá jenom sečíst celkovou práci přes všechny úrovně rekurze a máme worst-case odhad.