Kontrakce

Příklad 1: Obecná kontrakce: Pro graf G s vyznačenou množinou hran F (hrany na sobě mají značky) zkonstruujte kontrakci G/F. Chceme dosáhnout lineárního času.

Příklad 2: Kontrahující Borůvka: Navrhněte verzi Borůvkova algoritmu, která po každé fázi zkontrahuje hrany vybrané do kostry. Bude třeba ošetřit smyčky a násobné hrany. Chceme složitost $\mathcal{O}(m \log(n))$

Příklad 3: Rovinné grafy: Ukažte, že pro rovinné grafy kontrahující Borůvka najde minimální kostru v lineárním čase.

Hint: výsledkem kontrakce na rovinném grafu je opět rovinný graf

Vyhledávací Stromy

Pro jednoduchost předpokládáme, že všechny hodnoty jsou uloženy v listech (podobně jako v (a, b)-stromech).

Příklad 4: Mějme obecný VS, navrhněte následující operace v $\mathcal{O}(n)$:

- výpis všech hodnot v pořadí a vyvážení libovolně zneváženého stromu
- slití dvou stromů s menšími a většími hodnotami do jednoho
- rozštípnutí jednoho stromu na dva podle pivotní hodnoty

Příklad 5: Pro obecný VS s operacemi který drží přímo hodnoty (bez klíčů), s operacemi Insert, Delete a Find v čase $\mathcal{O}(\log n)$. Se zachováním asymptotické složitosti všech operací naučte VS následující operace v co nejlepší časové složitosti.

- nalezení předchůdce a následníka hodnoty (dané odkazem do stromu)
- Min a Max ze všech hodnot

Implementace v $\mathcal{O}(\log n)$ jsou snadné, chceme najít vylepšení stromu, aby operace probíhaly v $\mathcal{O}(1)$

Příklad 6: Pro obecný VS s operacemi který drží hodnoty řazené dle klíčů, s operacemi Insert, Delete a Find v čase $\mathcal{O}(\log n)$. Se zachováním asymptotické složitosti všech operací naučte VS následující operace v co nejlepší časové složitosti.

- počet prvků uložených pod klíči z intervalu $< a, b >$
- Min a Max hodnot z intervalu klíčů $< a, b >$

Implementace v čase závislém na počtu prvků v intervalu jsou snadné, chceme najít vylepšení stromu, aby operace probíhaly v $\mathcal{O}(\log n)$

Domácí úkol 5:

Při porovnávání třech základních algoritmů na kostry si můžeme všimnout, různých detailů chování algoritmů. Kontrahující verze Borůvkova algoritmu umí velmi rychle redukovat počet vrcholů vstupního grafu, ale v obecném případě se jeho pracovní graf zahustí a hrany ho zpomalí. Na druhou stranu Jarníkův algoritmus s vhodnou haldou (třeba Fibbonacciho) běží nejrychleji na hustých grafech, protože jeho složitost závisí na počtu hran pouze lineárně.

Jak těchto nuancí zneužít? Můžeme zkusit použít Borůvku na snížení počtu vrcholů vstupního grafu, a než se příliš zpomalí předat slovo Jarníkovi. Jarníkovi nebude vadit velké množství hran, ale menší množství vrcholů ho potěší.

Analyzujte časovou složitost následujícího algoritmu a ukažte, že je asymptoticky rychlejší než $\mathcal{O}(m \log(n))$

Function *Chimera(G, w váhy hran)*
  Proveď $\log \log n$ fází Borůvkova algoritmu
  $F_B \leftarrow$ hrany vybrané Borůvkou
  $G' \leftarrow G/F_B$ (kontrakce lesa nalezeného Borůvkou)
  $T' \leftarrow$ JarnikSFibHaldou(G')
  return $T' + F_B$
end

Upřesnění: Hrany chápeme jako objekty, které si zachovávají svou identitu i poté co jsou jejich koncové vrcholy změněny (např. každá hrana má svoje id). Všechny hrany G' tedy vědí ze kterých hran vstupního G vznikly. (a tedy dává smysl kostru G stavět z hran G')

Jak je zvykem u minimálních koster, předpokládáme, že G je souvislý a váhy hran jsou unikátní.

Pro správnou časovou analýzu je potřeba znát časovou složitost Jarníkova alg. s Fib. haldou, složitost fáze Borůvkova algoritmu, a správně odhadnout počet vrcholů v grafu G' (počet hran stačí odhadnout triviálně jako $\mathcal{O}(m)$). Všechny potřebné ingredience se řešily na cvičení, případně najdete spoustu detailů v sekcích 7.2 a 7.3 průvodce.

Pro úspěšné řešení je vyžadována pouze časová analýza časové složitosti se zdůvodněním.