Algoritmy a datové struktury I

11. cvičení

čtvrtek 5. 5. 2016 9:00

Nejlehčí kostry

Řezové lemma: Buď $G = (V, E)$ neorientovaný graf s unikátními váhami na hranách, $E' \subset E$ elementární řez[1] a $e \in E'$ nejlehčí hrana tohoto řezu. Pak se e nachází ve všech nejlehčích kostrách.

Důsledek: Každý graf s unikátními váhami hran má minimální kostru jednoznačně určenou uspořádáním vah hran, na hodnotách vah nezáleží.

Algoritmy:

1. **Jarníkův** (též Primův či Dijkstrův): Pěstujeme souvislý podgraf kostry. Začneme s jedním vrcholem a vždy přidáváme nejlehčí hranu vedoucí z aktuálního podgrafu do zbytku grafu.
2. **Borůvkův**: Začneme s lesem tvořeným všemi vrcholy. V každé fázi rozložíme les na komponenty souvislosti a pro každou komponentu přidáme nejlehčí incidentní hranu. Potřebuje unikátní váhy hran, nejvýše $O(\log n)$ fází.
3. **Kruskalův** (též Hladový): Začneme s lesem bez hran. Probíráme hrany grafu od nejlehčí a pokud přidání hrany nevytvoří v lese cyklus, tak ji přidáme.

Složitost algoritmů

- Borůvkův algoritmus trvá $\log n$ fází a každá fáze je lineární, takže celkem $\mathcal{O}(m \log n)$.
- Jarníkův algoritmus: záleží jak reprezentujeme hrany vedoucí ven. Nejlepší je pamatovat si v haldě sousední vrcholy ohodnocené délkou nejkratší hrany, kterou do nich můžeme přijít. Toto v kombinaci s Fibonaciho haldou dává $\mathcal{O}(n \log n + m)$.
- Kruskalův algoritmus: čas na setřídění hran plus n unionů a m findů na struktuře velikosti n.

Struktura Union-Find

Reprezentuje komponenty souvislosti grafu a umí operace:

- FIND(u, v): zjistí, zda u a v jsou v téže komponentě.
- UNION(u, v): spojí komponenty obsahující u a v.

Implementace:

- **Polem**: pole celých čísel velikosti n. Číslo určuje komponentu. Čas na Kruskalův algoritmus: naivně $\mathcal{O}(n^2)$, chytřeji $\mathcal{O}(m + n \log n)$ (viz cvičení 7).

[1] Množina hran E' je elementární řez, pokud $E' = e(V', V \setminus V')$ pro nějakou $\emptyset \neq V' \subsetneq V$.

- **Keříky**: každá komponenta je reprezentována stromem orientovaným do kořene. Složitos Kruskalova algoritmu (vyjma třídění):
  - naivně $\mathcal{O}(mn)$
  - UNION dle ranků či komprese cest $\mathcal{O}(m \log n)$
  - UNION dle ranků s kompresí cest $\mathcal{O}(m\alpha(n))$

Příklady

1. V rozboru implementace jsme navrhovali uložit všechny hrany řezu do haldy. Rozmyslete si všechny detaily tak, aby váš algoritmus běžel v čase $\mathcal{O}(m \log n)$.
2. Dokažte, že Jarníkův algoritmus funguje i pro grafy, jejichž váhy nejsou unikátní.
3. Rozmyslete si, jak v případě, kdy váhy nejsou unikátní, najít všechny minimální kostry. Jelikož koster může být mnoho (pro úplný graf s jednotkovými vahami jich je n^{n-2}), snažte se o co nejlepší složitost v závislosti na velikosti grafu a počtu minimálních koster.
4. Unikátnost vah je u Borůvkova algoritmu důležitá, protože by v kostře mohl vzniknout cyklus. Najděte příklad grafu, kde se to stane. Jak přesně pro takové grafy selže náš důkaz správnosti?
5. **Kontrahující Borůvkův algoritmus**: Borůvkův algoritmus můžeme přeformulovat, aby každý strom lesa udržoval zkontrahovaný do jednoho vrcholu. Iterace pak vypadá tak, že si každý vrchol vybere nejlehčí incidentní hranu, tyto hrany zkontrahujeme a zapamatujeme si, že patří do minimální kostry. Ukažte, jak tento algoritmus implementovat tak, aby běžel v čase $\mathcal{O}(m \log n)$. Jak si poradit s násobnými hranami a smyčkami, které vznikají při kontrakci?
6. Ukažte, že pokud algoritmus ze cvičení 5 používáme pro rovinné grafy, běží v čase $\Theta(n)$. Opět je třeba správně ošetřit násobné hrany.
7. Pečlivě ukažte, že Union-Find s polem lze zrychlit přečíslováváním menších komponent.

Domácí úkoly

Úkoly jsou za plný počet bodů **3 týdny** od zadání (deadline je počátek cvičení), poté za polovinu bodů. Úkoly mi posílejte na `husek+ads@iuuk.mff.cuni.cz`.

1. Mějme neorientovaný graf s hranami ohodnocenými celými čísly od 1 do L. Vymyslete co nejefektivnější algoritmus pro nalezení minimální kostry. Jeho složitost analyzujte vzhledem k velikosti grafu a číslu L. Devět bodů za algoritmus efektivní pro malé L (řekněme desítky), více, pokud bude efektivní i pro řádově větší L. **[lmst, 9]**
2. Sestrojte graf, na kterém algoritmus ze cvičení 5 potřebuje čas $\Omega(m \log n)$. (Formálně nekonstruujete jeden graf, ale nekonečnou posloupnost grafů, protože pro jeden graf asymptotiocká notace nedává smysl.) **[graf, 9]**
3. Nalezněte druhou nejlehčí kostru. **[mst2, 13]**

Informace o cvičení viz `http://iuuk.mff.cuni.cz/~husek/vyuka/ads1516.html`