Diskstra a relaxační algoritmy

Pojmy: prefixová vlastnost nejkratších cest, strom nejkratších cest, záporný cyklus, Disjkstrův algoritmus

Příklad 1: Haldy: Vzpomeňte na definici obecné haldy.

Analyzujte složitost Dijkstrova algoritmu s použitím různých typů haldovitých struktur. Chceme použít pole/seznam, binární strom (nebo binární haldu), a Fibonacciho haldu.

(+) Ukažte, že nemůže existovat obecná halda t.ž. maximum ze složitosti insert a extract min je menší než $\mathcal{O}(\log(n))$ (kde n je počet prvků v haldě) Hint: taková halda by rozbila jeden z dolních odhadů na časovou složitost konkrétního problému

Příklad 2: Diskrétní délky: Mějme graf s délkami hran pouze $1, \ldots, k$ pro konstantu k. Najděte způsob jak zneužít toto omezení pro zrychlení Dijkstry. Hledáme návrh efektivní haldy s operacemi v konstatním čase $\mathcal{O}(k)$.

Hint: jaké hodnoty mohou být v haldě?

Příklad 3: Zápornost: Najděte graf s jedinou zápornou hranou na které Dijkstra selže. Navrhněte úpravu algoritmu, aby neselhal (za cenu horší časové složitosti).

Příklad 4: Spolehlivost: Máme graf komunikační sítě kde každá hrana má spolehlivost (pravděpodobnost, že data při přenosu nepoškodí). Najděte nejspolehlivější spoje ze zdroje do všech vrcholů sítě.

Příklad 5: Záporné cykly: Upravte relaxační algoritmus tak, aby v grafu se záporným cyklem našel cyklus a vydal ho na výstup jako certifikát selhání.

All-Pairs algoritmy

Příklad 6: Floyd-Warshall: Zopakujte si princip F.-W. algoritmu. Upravte algoritmus tak, aby měl co nejmenší prostorovou složitost.

Příklad 7: Složitost: Porovnejte F.-W. algoritmus s aplikací relaxačních algoritmů na all-pairs problém. V jakých speciálních případech budou relaxační algoritmy rychlejší?

Příklad 8: Záporný cyklus: Pokud F.-W. narazí na záporný cyklus, chceme ho vypsat, jak toho docílit?

Domácí úkol 3:

Mějme mapu malebného městečka se spoustou úzkých uliček a kaváren s předzahrádkami. Chceme z našeho distrubučního skladu rozvážet do kaváren suroviny. Některé ulice jsou ale příliš úzké pro nákladďáky, dodávky, mnohdy i rozumně velké automobily. Chceme pro každou kavárnu najít cestu po které se k ní dostane co nejširší dopravní prostředek.

Na vstupu máme (orientovaný) graf s hranami ohodnocenými šířkou $1, \ldots, \infty$. Je dán výchozí vrchol v. Pro všechny ostatní vrcholy chceme spočítat šířku největšího dopravního prostředku, který by se do daného vrcholu z v mohl dostat. Budeme také chtít spočítat informace, ze kterých můžeme v případě potřeby pro libovolný vrchol w zrekonstruovat celou cestu mezi v a w v rozumném čase (úměrný délce cesty).

Předpokládáme, že vstup je korektně zadaný graf s vrcholem a ohodnocením. Můžeme předpokládat, že všechny hrany, které v grafu jsou mají nenulovou šířku. Pro každý vrchol chceme cestu maximální šířky, neuvažujeme žádnou množinu dostupných dopravních prostředků nebo další omezení.

(+) [tuto část nemusíte řešit ani odevzdávat, nebude hodnocena] ... ale můžete se zkusit samostatně zamyslet co by se na úloze změnilo pokud bychom měli danou množinu aut s šířkami, a chtěli najít cestu pro nejšírší z dostupných aut. Úloha bude téměř totožná, pouze s jedním obskurujícím krokem navíc. Přístupů je ale hned několik.

Řešení by tradičně mělo obsahovat stručný popis metody, pseudokód, komentář ke speciálním případům, pořádný důkaz korektnosti (můžete odkazovat na vlastnosti algoritmů z přednášek, cvičení a průvodce), analýzu časové složitosti, a stručný komentář jak by bylo možné rekonstruovat konkrétní cesty.