Reitinhaku verkossa

Toteutus

Harjoitustyön toteutuksessa on käytetty Dijkstran algoritmia. Kuten määrittelydokumentissa on todettu, Dijkstran algoritmin aikavaativuus on $O((|V|+|E|) \log|V|)$ ja tilavaativuus $O(|V|)$, missä $|V|$ on verkon G solmujen ja $|E|$ kaarien lukumäärä. Nämä vaativuusluokat on johdettu monisteen[1] sivulla 526. Harjoitustyön toteutuksessa päästään samoihin luokkiin, minkä perustelemiseksi tarkastelemme seuraavaksi rinnakkain monisteen pseudokoodia ja harjoitustyön lähdekoodia.

Monisteen sivulla 522 esitetty algoritmi Dijkstra-With-Heap alkaa operaatioilla Initialize-Single-Source ja S = ∅, jotka vievät aikaa $O(|V|)$. Monisteen Initialize-Single-Source:ssa asetetaan distance[v] = 0 ja path[v] = NIL kaikilla $v \in V \setminus \{s\}$. Lähdekoodissa tämä on toteutettu suoraviivaisesti sillä erolla, että lisäksi kutsutaan vakioaikaista operaatiota locationToNode.

Seuraavaksi monisteen pseudokoodissa lisätään kaikki solmut kekoon. Lähdekoodissa tämä tehdään operaatiolla heapInsertAll, jossa käydään läpi kaikki kartan ruudut. Kutakin ruutua vastaa yksi verkon solmuista, joten insert-operaatiota toistetaan yhteensä $|V|$ kertaa. Insert on vaativuudeltaan $O(\log|V|)$: pahimmassa tapauksessa uusi solmu on kuljetettava binääripuun lehdestä juureen asti, ja lähes täydellisen binääripuun korkeus on tunnetusti logaritminen puussa olevien solmujen määrään nähden. Siispä heapInsertAll on vaativuudeltaan $O(|V| \log|V|)$.

Seuraavana vuorossa olevaa while-silmukkaa toistetaan $|V|$ kertaa. Joka kierroksella kutsutaan $O(\log|V|)$ -aikaista operaatiota heap-del-min. Relax- ja heap-dec-key-operaatiot tehdään enintään $|E|$ kertaa. Relax toimii ajassa $O(1)$ ja heap-dec-key ajassa $O(\log|V|)$, joten while-silmukan aikavaativuus on $|V|\ O(\log|V|) + |E|\ O(\log|V|) = O((|V|+|E|) \log|V|)$. □

1 Floréen, P. 2013: Tietorakenteet ja algoritmit. Helsingin yliopiston Tietojenkäsittelytieteen laitos.