Toteutusdokumentti

Ohjelman rakenne

Ohjelma koostuu geneerisistä luokista `AVLTree`, `Treap` ja `ScapegoatTree`, jotka perivät abstraktin geneerisen luokan `BinaryTree`. Kukin näistä toteuttaa tasapainotetun binääripuun omalla tavallaan.

Lisäksi ohjelmassa on mukana luokka `Tester`, joka ottaa parametreikseen abstraktin luokan `Test` periviä testiluokkia, jotka generoivat syötteen ja antavat sen annetulle puulle testattavaksi. Kunkin testin jälkeen `Tester` tulostaa raportin testin onnistuneisuudesta ja siihen kuluneesta ajasta. Testaaja käyttää hajautustaulua pitääkseen kirjaa siitä, mitä lukuja puussa kuuluisi olla sillä hetkellä.

Lisäksi luokat hyödyntävät `util`-paketissa olevia tietorakenteita, kuten `Pair` ja `List`. Hajautustaulu vaatii vielä oman implementaationsa.

Aika- ja tilavaativuudet

Aikavaativuksista yksinkertaiset todistusideat tähän.

AVL-puu

AVL-ehto sanoo, että kunkin solmun lapsien syvyyksien ero saa olla enintään yksi. Todistetaan puun syvyyden olevan $O(\log n)$, mikäli AVL-ehto pätee.

Olkoon A_n pienin määrä solmuja, jotka tarvitaan, että puun syvyys on n ja AVL-ehto pätee. Selvästi $A_0 = 1$, $A_1 = 2$ ja $A_n = 1 + A_{n-1} + A_{n-2}$, sillä konstruktio tapauksesta A_{n-1} tapaukseen A_n tarkoittaa, että A_{n-1}:n juureen kiinnitetään uusi juuri, ja nyt uusi juuri vaatii AVL-ehdon säilymiseksi toiseksi lapsekseen vähintään $n-2$-syvyisen puun.

Koska A on aidosti kasvava, niin $A_n \geq 2A_{n-2}$. Puun syvyyden on siis oltava logaritminen AVL-ehdon pätiessä.

Treap

Treapin jokainen solmu sisältää prioriteetin ja säilytettävän arvon. Arvoja säilytetään puussa siten, että pienemmät arvot ovat aina vasemmassa alipuussa ja suurin prioriteetti ylimpänä. Tämä määrää puun rakenteen yksiselitteisesti,

olettaen että prioriteetit ovat keskenään eriäviä. Koska törmäyksiä tulee vain harvoin, voimme käytännössä olettaa näin olevan.

Hakeminen

Sekä AVL-puu että syntipukkipuu ovat syvyydeltään $O(\log n)$, joten tietyn solmun hakeminen puusta vie vain $O(\log n)$ aikaa. Treap on odotusarvoisesti ja lähes varmasti myös syvyydeltään logaritminen suhteessa puun kokoon, joten myös sille voidaan olettaa haun vievän $O(\log n)$ aikaa.

Muistivaativuus

Kukin solmu vaatii vain vakiomäärän tilaa, joten kaikkien puiden tilavaativuus on $\Theta(n)$.

Suorituskykyvertailu

Nykyisien tuloksien perusteella treap on vakiokertoimiensa suhteen optimaalisempi kuin AVL-puu tai syntipukkipuu. Tämä johtunee osittain siitä, ettei treapin tarvitse päivittää pituuttaan tai kokoaan tai uudelleenrakentaa itseään. AVL-puu on aavistuksen huonompi vakiokertoimiltaan, mutta päihittää syntipukkipuun yli kaksinkertaisella nopeudella suuremmissa tapauksissa. Ilmeisesti syntipukkipuuta hidastaa erityisesti puun uudelleenrakennus, vaikka sen operaatiot ovatkin asymptoottisesti $O(\log n)$.

On kuitenkin tärkeää muistaa, että tulokset eivät ole aivan vertailukelpoisia, sillä implementaatiota voisi varmasti optimoida kaikille luokille.

Parannusehdotukset

Lisättäköön kun saadaan työ loppuun.

Lähteet

• Wikipedia. AVL Tree. Viitattu 28.9.2018
https://en.wikipedia.org/wiki/AVL_tree
• Wikipedia. Treap. Viitattu 28.9.2018
https://en.wikipedia.org/wiki/Treap
• Wikipedia. AVL Tree. Viitattu 28.9.2018

https://en.wikipedia.org/wiki/Scapegoat_tree