

Tietorakennevertailun toteutusdokumentti

Marko Haanranta

16. kesäkuuta 2013

1 Ohjelman yleisrakenne

Tietorakenteiden harjoitustyönä toteutin neljä eri tietorakennetta ja vertailin niiden lisäys ja poisto operaatioiden ajankäyttöä suhteessa javan valmiiseen priority queue tietorakenteeseen. Tietorakenteina toteutin binäärikeon, binomikeon, fibonaccin keon ja AVL-puun. Kaikki keot on toteutettu minimekeoina.

Ohjelman aikavaativuuden testaamisen käytiin Vertailija.java luokkaa.

1.1 Binäärikeko

Binäärikeko on toteutukseltaan hieman yksinkertaisempi kuin muut toteutetut tietorakenteet. Kekoon lisättävä avain tallennetaan taulukkoon ja sen jälkeen ajetaan heapify metodi joka vie avaimen sen oikealle paikalle keossa. Jos käytetään int-muuttujaa avaimen tallentamiseen niin binäärikeko on selkeästi nopeampi kuin muut toteuttamani tietorakenteet. Muutin toteutustani sen verran, että nyt tallennan taulukkoon Integer-olioita ja näin toteutettuna binäärikeko on hitain toteuttamani tietorakenne. Binäärikeon aikavaativuus lisäys ja poisto operaatioille on $O(\log n)$ [1].

1.2 Binomikeko

Binomikeon solmut tallennetaan Node-olioina. Jokaisella solmu-oliolla on seuraavat kentät: key, parent, child, sibling ja degree. Binomikeko rakentuu linkitetystä listasta, johon on tallennettu binomipuita. Puiden solmuilla on aste eli degree ja jotta keko olisi binomikeko ehdon mukainen juurilistalla ei voi olla kahta puuta joilla on sama aste. Binomikeon aikavaativuus lisäys ja poisto operaatioille on $O(\log n)$ [1].

1.3 Fibonaccin keko

Fibonaccin kekoon lisättävä avain tallennetaan FibNode-oliona. FibNode oliolla on seuraavat kentät key, parent, child, left, right, degree ja mark[2].

Fibonaccin keko rakentuu puista fibonaccin puista. Fibonaccin keossa kaikki avaimet lisätään juurisolmulistaan ja varsinainen keko rakennetaan vasta poisto operaation yhteydessä. Fibonaccin keon aikavaativuus lisäys operaatiolle on $O(1)$ ja poisto operaatiolle $O(\log n)$ [3].

1.4 AVL-puu

AVL-puu rakentuu `AvlNode`-olioista. Jokaisella solmulla on `key`, `left`, `right`, `parent` ja `height` kentät[1]. AVL-puu on tasapainoinen binäärihakupuu eli jokaisella solmulla on kaksi lasta(`child`) ja vanhempi(`parent`). Paitsi tietysti koko puun juurisolmulla jonka `parent` kentän arvo on `null`. Puun solmun vasemmassa alipuussa on vain solmua pienemmän avaimen omaavia solmuja. Puun solmun oikeassa alipuussa on vain puun solmua suuremman avaimen omaavia solmuja. AVL-puu pidetään tasapainossa suorittamalla kierto operaatioita, kun puuhun lisätään tai siitä poistetaan solmuja. AVL puun aikavaativuus lisäys ja poisto operaatioille on $O(\log n)$ [4].

2 Saavutetut aika- ja tilavaativuudet

Kaikki toteuttamani tietorakenteet saavuttivat lisäys ja poisto operaatioiden aikana $O(\log n)$ aikavaativuuden. Fibonaccin keon lisäys ei syystä tai toisesta toiminut odotetun $O(1)$ aikavaativuuden mukaisesti, mistä olen kirjoittanut enemmän pohdintaa testausdokumentin kritiikkiä osioon.

Kaikki toteuttamani algoritmit myös toimivat $O(1)$ tilassa, koska missään operaatiossa tarvita suuria määriä apumuuttujia.

3 Työn puutteet ja parannusehdotukset

Alkuperäisenä tarkoituksenani oli testata avainten lisäämistä ja poistamista myös käänteisessä ja satunnaisessa järjestetyksessä syötetyillä avaimen arvoilla. Tämä jäi nyt toteuttamatta, koska jouduin käyttämään huomattavan paljon aikaa fibonaccin keon ja avl-puun delete operaatioiden debuggaamiseen.

Testatessani huomasin ettei toteuttamani Vertailija toiminut ihan odotetusti. Jos ajoin kaikki testit kerralla niin sain huomattavasti parempia tuloksia kuin jos ajoin vain yhden testin kerrallaan. Tästä johtuen ajoin kaikki testit erikseen muutamia kertoja ja otin tuloksista keskiarvot.

Fibonaccin keko ja binomikeko tarjoavat operaation kahden keon yhdistämiseen. En toteuttanut Fibonaccin keon union operaatiota ja tästä johtuen en myöskään erikseen testannut binomikeon merge operaation nopeutta.

Viitteet

- [1] Patrick Floreen, *Tietorakenteet ja algoritmit*,
<http://www.cs.helsinki.fi/u/floreen/tira2013/tira.pdf>,
Luentomateriaali 2013.
- [2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest ja Clifford Stein: *Introduction to Algorithms, third edition*. The MIT Press 2009.
- [3] Heap (data structure),
http://en.wikipedia.org/wiki/Heap_data_structure/
- [4] AVL tree,
http://en.wikipedia.org/wiki/AVL_tree