Riikka Huusari

Määrittelydokumentti

Toteutan työssäni binäärikeon, binomikeon, fibonaccikeon sekä d-ary –keon. Vertailen kekojen toimivuutta ja eri operaatioiden aikavaativuuksia ainakin jonkun kekoa käyttävän algoritmin avulla.

Binomikeko muodostuu useista puista, joista jokainen toteuttaa kekoehdon eli lapsi on aina arvoltaan joko pienempi tai yhtä suuri kuin vanhempansa. Jokaisella puulla on tarkka rakenne asteensa mukaan. Nollannen asteen puu on yksi solmu, ja k:nnen asteen puu muodostetaan lisäämällä k-1:n asteen puu lapseksi toiselle k-1:n asteen puulle. Keossa voi olla korkeintaan yksi tiettyä astetta oleva puu.

Fibonaccikeko muodostuu binomikeon tapaan useista kekoehdon toteuttavista puista, mutta sen rakenne ei ole yhtä jäykkä kuin binomikeon. Kekoon lisäys tapahtuu vain tekemällä uusi solmu ja lisäämällä se yhdeksi keon puista. Puita yhdistellään vasta deleteMin-operaatiossa, jonka jäljiltä keossa ei ole kahta samanasteista puuta. Keon litteydestä huolehditaan operaatiossa decreaseKey: jos solmulta poistetaan jo toinen lapsi, myös kyseinen solmu poistetaan puusta ja lisätään uudeksi puuksi.

Taulukossa ovat tavoittelemani aikavaativuudet, missä n tarkoittaa keon solmujen lukumäärää ja d on solmun lapsien maksimimäärä.

	binäärikeko	binomikeko	fibonaccikeko	d-ary keko
insert	O(log n)	O(log n)	O(1)	O(log n /log d)
findMin	O(1)	O(log n)	O(1)	O(1)
deleteMin	O(log n)	O(log n)	O(log n)	O(d log n/log d)
decreaseKey	O(log n)	O(log n)	O(log n)	O(log n/log d)

Lähteet:

www.cs.princeton.edu/~wayne/teaching/fibonacci-heap.pdf
http://www.cs.princeton.edu/courses/archive/spring13/cos423/lectures/BinomialHeaps.pdf
http://en.wikipedia.org/wiki/Heap_(data_structure)
http://en.wikipedia.org/wiki/Binary_heap
http://en.wikipedia.org/wiki/D-ary_heap
http://en.wikipedia.org/wiki/Binomial_heap
http://en.wikipedia.org/wiki/Fibonacci_heap