Johanna Keto

Dokumentaatio

Määrittely

Projektissani toteutan kolme kekoa: binomi-, binääri- ja fibonaccikeon. Pyrin toteuttamaan kaikkien kekojen seuraavat toiminnallisuudet:
-uuden alkion lisääminen kekoon
-huipun (minimin) palauttaminen keosta
-huipun (minimin) poistaminen keosta

Jos saan kekojen tärkeimmät ominaisuudet tehtyä, vertailen seuraavaksi niiden suoritusnopeuksia.
Binäärikekoon ja binomikekoon uuden alkion lisäämisen tulisi tapahtua O(log n)-ajassa ja fibonaccikekoon O(1)-ajassa.

Keon minimiarvon löytämisen aikavaativuus tulisi Binäärikeossa olla O(1), binomikeossa O(log n) ja fibonaccikeossa 0(1). Binomikeossa aikavaativuus voi olla myös 0(1), jos osoittimen pienimmän arvon sisältävään alkioon voi jotenkin tallentaa.

Pienimmän arvon poistamisen tulisi tapahtua binäärikeossa ja binomikeossa O(log n)-ajassa. Fibonaccikeossa aikavaatimus vaihtelee sen mukaan, onko keosta poistettu mitään ensimmäisten alkiolisäysten jälkeen. Keskimääräinen aikavaatimus on O(log n).

Ohjelmointikielenä käytän Javaa ja ohjelmointiympäristönä NetBeansia (versio 7.3.1).
Lähteinäni on internet (lähinnä kekojen wikipediasivut), sekä kirja Introduction to Algorithms(2.painos).

Toteutus

Seuraavissa taulukoissa on listattuna ohjelmallani saatuja tuloksia eri kekojen operaatioista. Ohjelmani laskee ja tulostaa samat taulukot, mutta niissä arvot heittelevät roskienkerääjän vuoksi (näissäkin se saattaa hieman vaikuttaa). Allaolevien taulukoiden arvot on laskettu kaikki erikseen.

Binäärikeko

Alkioita kpl	Insert /ns	insert/n*1000	GetMIn /ns	Pop /ns	Pop/n*1000
100	1	10.0	0	3	30.0
1000	5	5.0	0	22	22.0
10000	30	3.0	0	48	4.8
100000	48	0.43	0	147	1.47
500000	103	2.06	0	1264	25.28
1000000	948	0.948	0	2646	2.646
5000000	968	0.1932	0	18856	3.7712
10000000	10714	0.1.0714	0	78125	7.8125

Binomikeko

Alkioita kpl	Insert /ns	N/insert*1000	GetMIn /ns	Pop /ns	Pop/n*1000
100	1	10.0	0	2	20.0
1000	7	7.0	0	18	18.0
10000	32	3.2	0	67	6.7
100000	41	0.41		175	1.75
1000000	1200	1.2	0	1543	1.543
5000000	782	0.1936	0	10616	2.1232
10000000	6896	1.0714	0	54844	5.4844

Fibonaccikeko

Alkioita kpl	Insert /ms	N/insert*1000	GetMin /ms	Pop /ms	Pop/n*1000
100	1	10.0	0	3	30.0
1000	3	3.0	0	33	33.0
10000	25	2.5	0	66	6.6
100000	41	0.41	0	181	1.81
500000	93	0.186	0	1248	2.696
1000000	1039	1.039	0	2127	2.127
5000000	2544	0.5088	0	20596	4.1192
10000000	8786	0.8786	0	87690	8.769

Taulukoihin on laskettu myös yksittäisten operaatioiden keskimääräiset kestot

tuhannella kerrottuna. Näistä tuloksista voimme päätellä, että halutut aikavaativuudet saatiin toteutettua. Käytännössä tällaisista tuloksista on kuitenkin vaikea hahmottaa, onko saatu tulos O(log n) vai O(1)-aikainen.

Testaus

Testaus oli ohjelmointini kannalta elintärkeää, ja testailinkin koodiani läpi projektin. Projektistani löytyy automaattiset JUnit-testit, mutta testasin koodia myös debuggaamalla, väliaikaisilla maineilla sekä erilaisilla testitulosteilla.

Koodin ajaminen

Voit testata yksittäisen keon ominaisuuksia tai ajaa vertailuoperaation, joka tulostaa taulukon kaikkien kekojen toiminnallisuuksien ajoajoista.
Kirjoita käskyt komentorivillä siinä kansiossa, missä kekoilua.jar sijaitsee.
Jos haluat testata tiettyä kekoa esimerkiksi tuhannella alkiolla, kirjoita haluamastasi keosta riippuen joko
java -jar kekoilua.jar binääri 1000
tai
java -jar kekoilua.jar binomi 1000
tai
java -jar kekoilua.jar fibonacci 1000

Jos haluat nähdä vertailutaulukon, kirjoita
java -jar kekoilua.jar vertailu
Epäilen roskienkerääjää aikaheittojen syyksi... Toisella testaustavalla niitä ei esiinny.

Huom! Maini on aika hirveää luettavaa, mutta yksikkötestien ja itse kekojen pitäisi olla aika siistejä.