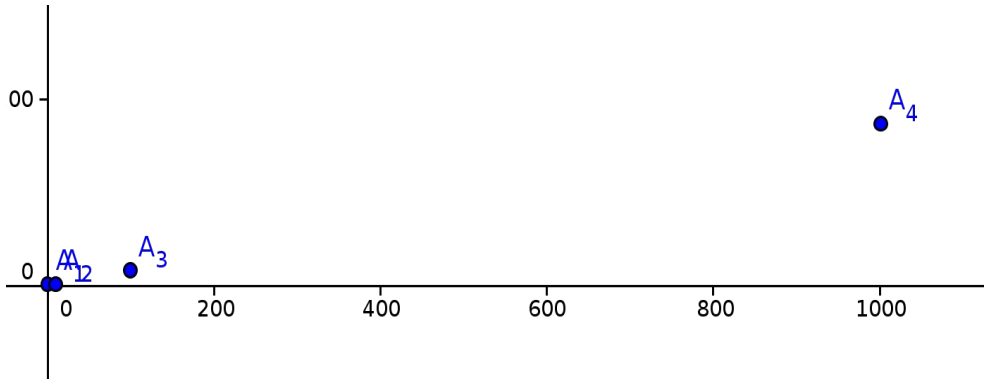
Testausdokumentti

Riikka Huusari

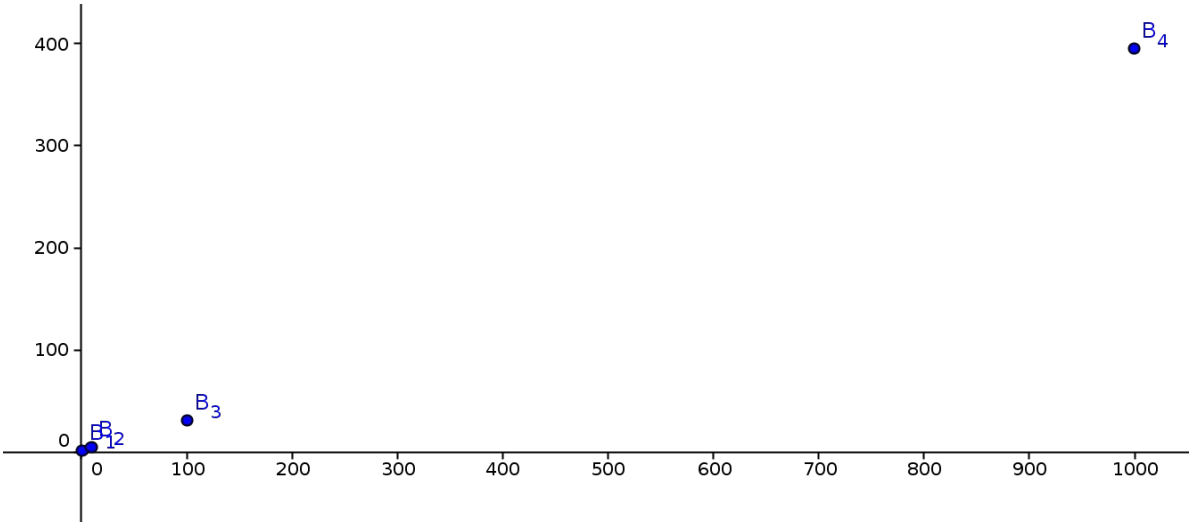
Tein kaikkiin kekoluokkiin omat JUnit testit, kuten myös Kekojärjestämiseen ja Dijkstraan, varmistaakseni koodin toimivuuden. Aikavaativuuksien toteutumista testasin lähinnä ajamalla ohjelmaa eri syötteillä.

Suoritin ohjelmani ja piirsin kuvat eri kekojen taulukon järjestämisestä niin, että pystyakselilla on järjestämiseen kulunut aika millisekunteina, ja vaaka-akselilla on taulukon koko/1000.
Binäärikeko:

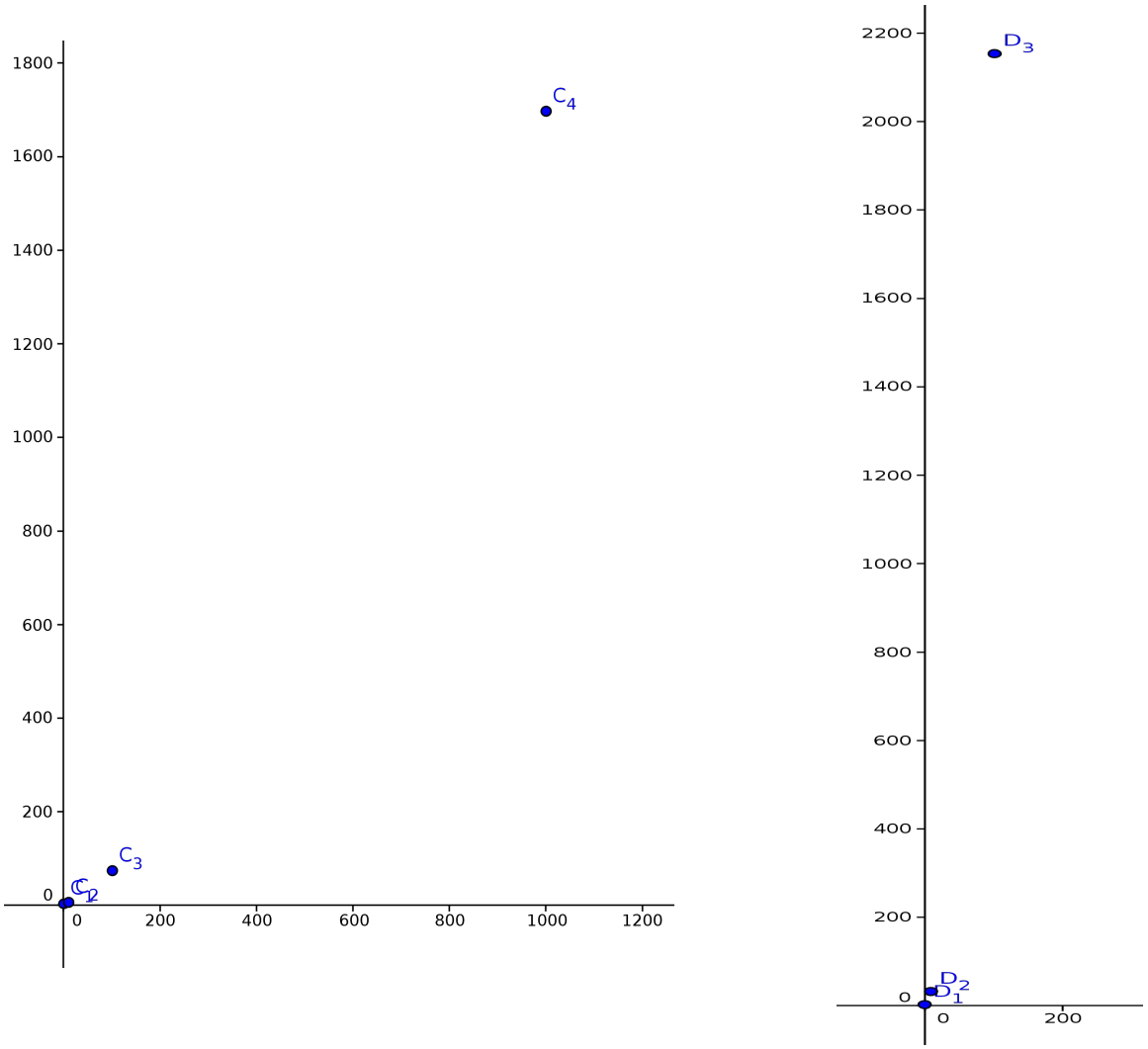


Binäärikeolla järjestäminen näyttäisi kuvan mukaan olevan aikavaativuudeltaan luokkaa O(n).

D-ary -keko:



D-ary -keon kuvaaja näyttää hyvin samanlaiselta kuin binäärikeon.



Binomikeon kuvaaja (vasemmalla) näyttää nousevan jyrkemmin kuin edeltäjänsä. Aikavaativuus olisi kuvan perusteella ehkä luokkaa O(n).

Fibonaccikeko (oikealla). Kuvassa ei ole arvoa taulukolle, jonka pituus olisi 1 000 000, sillä sen järjestäminen kesti liian kauan. Tämän aikavaativuudet näyttäisivät kuvan mukaan suunnilleen eksponentiaalisilta.

Seuraavassa olen laittanut taulukkomuotoon saamani kekojärjestämisen ajat. Ylhäällä on taulukon pituus, sivussa lukee millä keolla taulukko järjestettiin. Muissa kuin pienimmässä järjestämisessä olen myös kirjoittanut millä kertoimella seuraava aika on saatu edellisestä.

	1000	**10000**	**100000**	**1000000**
Binäärikeko	1,1 ms	2,8 ms = **2,5***1,1 ms	16 ms = **5,7***2,8 ms	175 ms = **10,9***16 ms
Binomikeko	2,7 ms	6 ms= **2,2** * 2,7 ms	76 ms = **12,7** *6 ms	1698 ms = **22,3** * 76 ms
D-ary -keko	0,9 ms	4,2 ms= **4,6** * 0,9 ms	32 ms = **7,6** * 4,2 ms	395 ms = **12,3** * 32 ms
Fibonaccikeko	3,3 ms	31,1 ms = **9,4** * 3,3ms	2155 ms= **69,3** * 31,1 ms	-

Oikealle mennessä taulukon pituus kymmenkertaistuu, joten kertoimien perusteella eivät järjestämiset mielestäni toimi O(log n) ajassa. Varsinkin Fibonaccikeolla järjestämiseen kuluva aika kasvaa hyvin nopeasti. Binäärikeko ja d-ary -keko ovat parhaita.