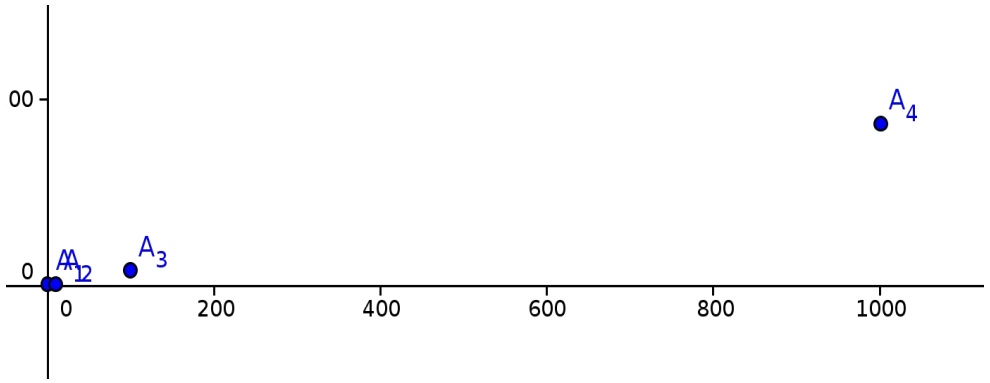
Testausdokumentti

Riikka Huusari

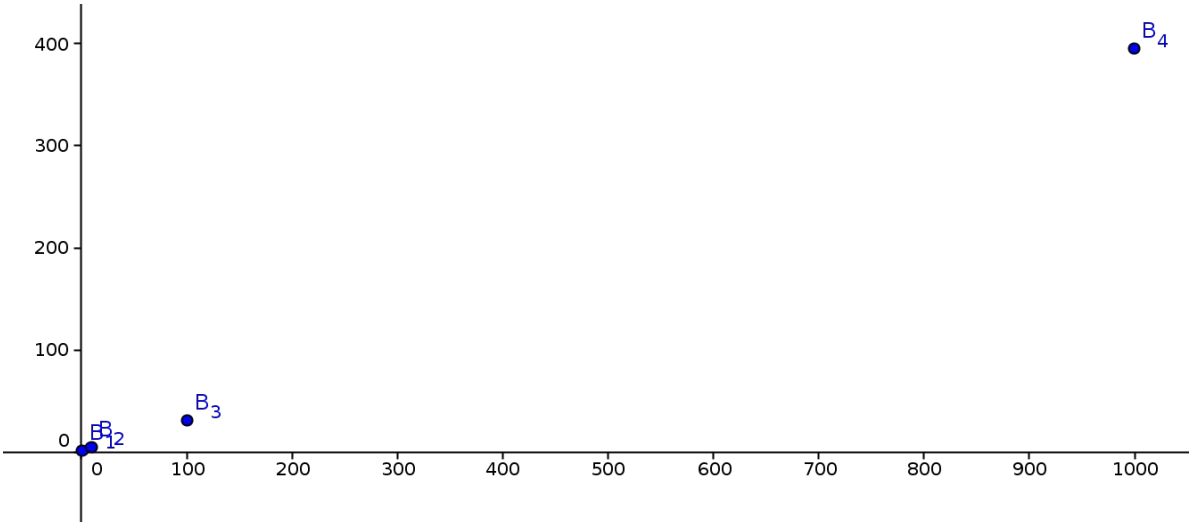
Tein kaikkiin kekoluokkiin omat JUnit testit, kuten myös Kekojärjestämiseen ja Dijkstraan, varmistaakseni koodin toimivuuden. Aikavaativuuksien toteutumista testasin lähinnä ajamalla ohjelmaa eri syötteillä.

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

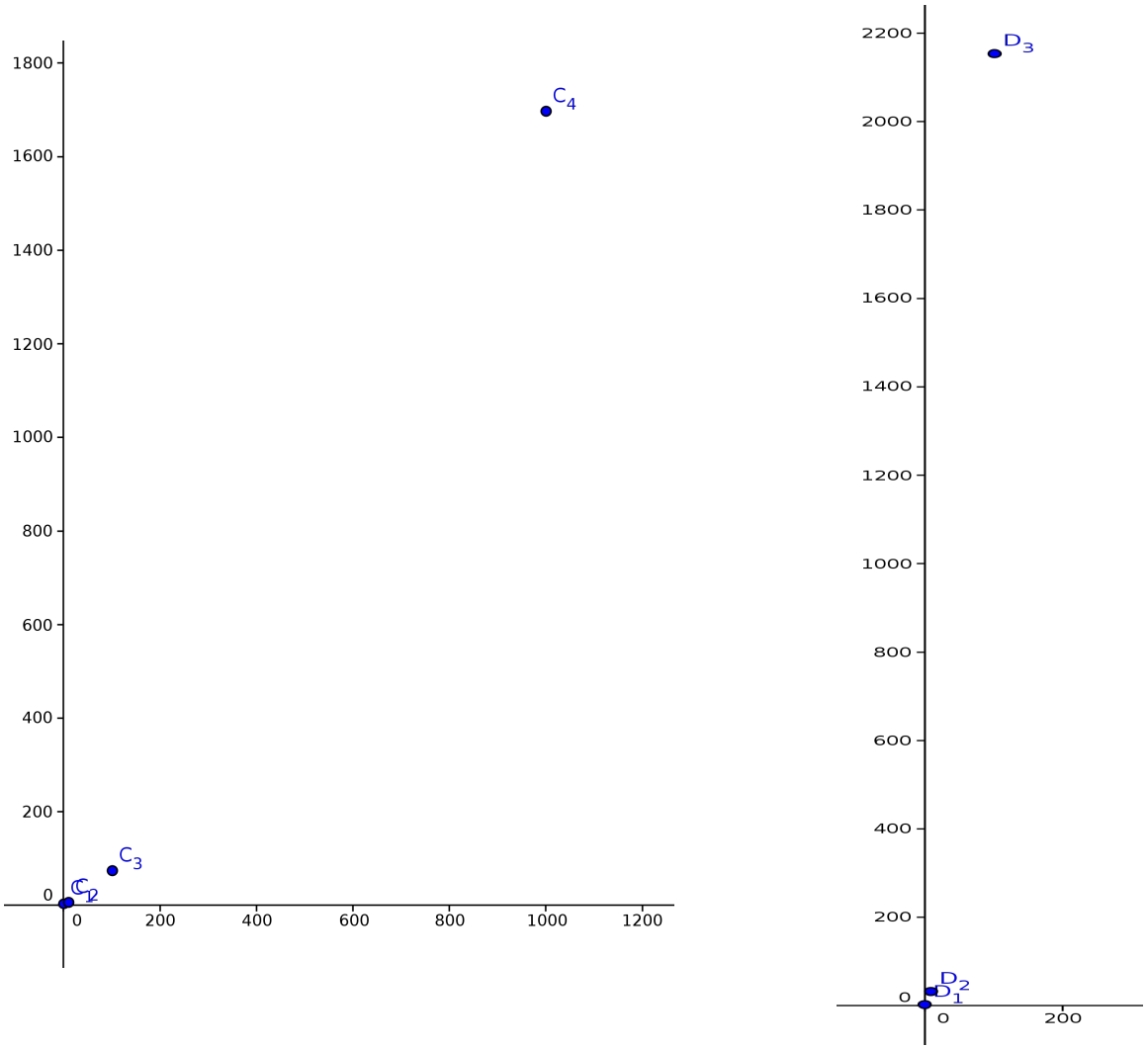


Binäärikeolla järjestäminen näyttäisi kuvan mukaan olevan aikavaativuudeltaan luokkaa O(n).

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).

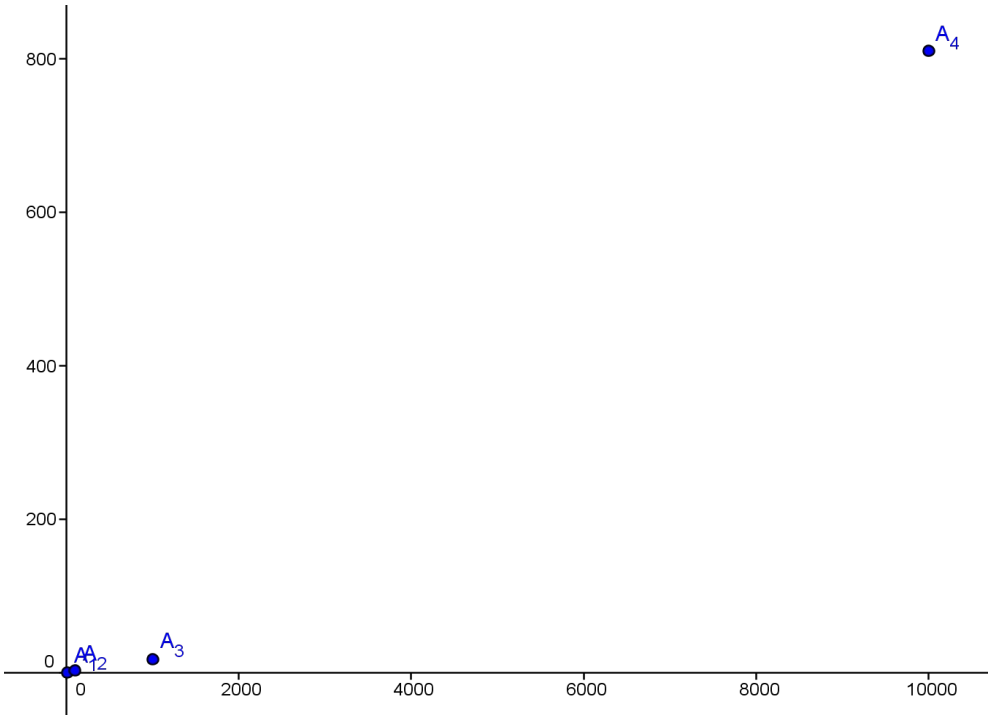
Fibonaccikeko (oikealla). Kuvassa ei ole arvoa taulukolle, jonka pituus olisi 1 000 000, sillä sen järjestäminen kesti liian kauan. Tämän aikavaativuudet näyttäisivät kuvan mukaan suunnilleen eksponentiaalisilta. Saattaa toki myös johtua hyvin suurista vakiokertoimista.

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.

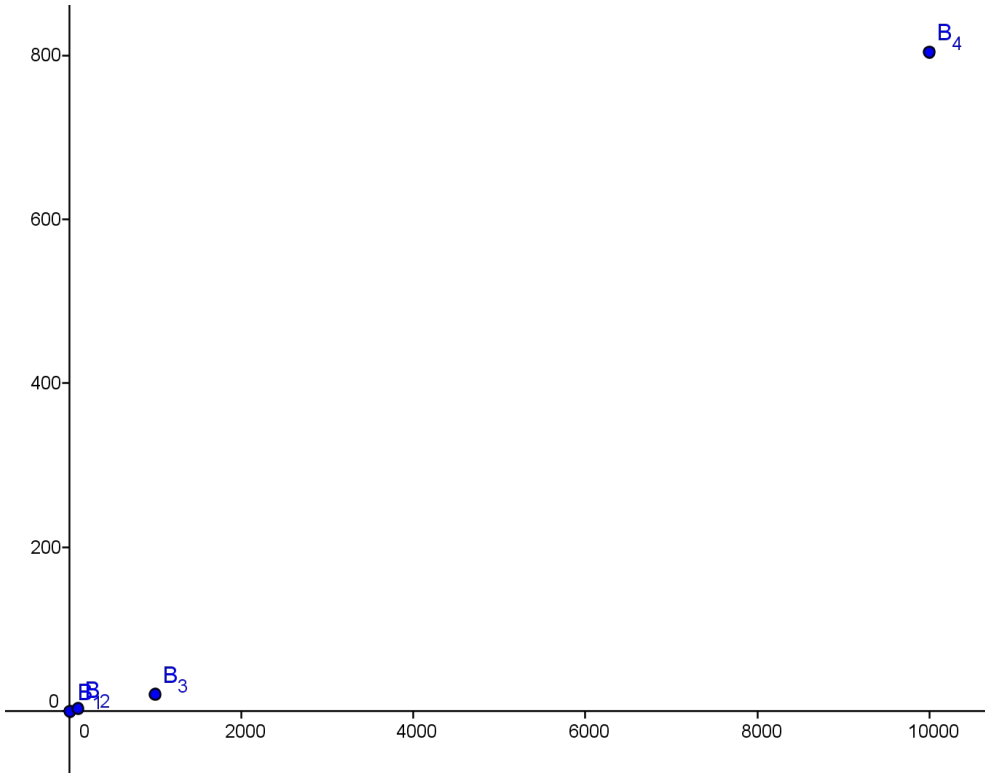
	1000	**10000**	**100000**	**1000000**
Binäärikeko	1,1 ms	2,8 ms = **2,5***1,1 ms	16 ms = **5,7***2,8 ms	175 ms = **10,9***16 ms
Binomikeko	2,7 ms	6 ms= **2,2** * 2,7 ms	76 ms = **12,7** *6 ms	1698 ms = **22,3** * 76 ms
D-ary -keko	0,9 ms	4,2 ms= **4,6** * 0,9 ms	32 ms = **7,6** * 4,2 ms	395 ms = **12,3**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

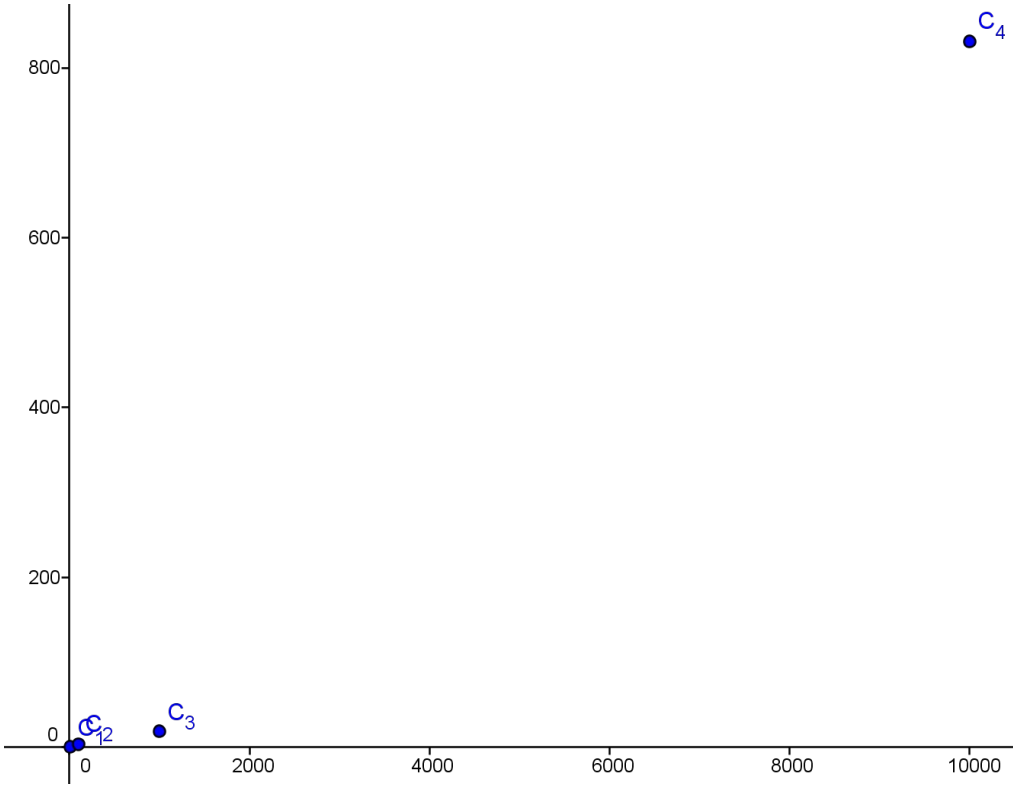
Mittasin myös eri kekojen suoriutumista Dijkstran algoritmista. Kuvassa binäärikeolla kuluneet ajat, pystyakselilla millisekunnit ja vaaka-akselilla verkon solmujen lukumäärä.



D-ary -keko:



Binomikeko:



Kaikki keot näyttävät suoriutuvan samalla tavalla Dijkstran algoritmista. Vielä taulukkona, jossa olen korostanut kertoimia edellisen ja seuraavan ajan välillä:

	10	**100**	**1000**	**10000**
Binäärikeko	0,33 ms	3,2 ms = **9,7** * 0,33 ms	17,4 ms = **5,4** * 3,2 ms	810 ms = **47** * 17,4 ms
Binomikeko	0,26 ms	2,4 ms= **9,2** * 0,26 ms	20,7 ms = **8,6** *2,4 ms	804 ms = **39** * 20,7 ms
D-ary -keko	0,25 ms	3,5 ms= **14** * 0,25 ms	18,6 ms = **5,3** * 3,5 ms	831 ms = **44,7** * 18,6 ms
Fibonaccikeko				

Kerroin viimeisen ja toisiksiviimeisen pituuden välillä on valtava. Oletan että se johtuu ainakin osittain siitä, että Dijkstran verkko on vierusmatriisina, joten sivun pituuden kasvaessa alkioiden määrä matriisissa kasvaa todella paljon.

Dijkstra ei toimi kunnolla fibonaccikeolla. En ole vielä paikantanut virhettä mutta luulen, että se liittyy jotenkin decreaseKey-metodiin, koska se on ainoa 'uusi' metodi verrattuna kekojärjestämiseen. Testi testaaUudestaanPienellaVerkollaJaFibonaccikeolla jää vain junnaamaan, eikä anna mitään tulosta (se on nyt kommentoitu pois). Sama käy kun yritän ajaa ohjelmani. Jätin sen kerran taustalle pyörimään, eikä 28 minuutin kuluttua ollut vielä tapahtunut mitään.