Testaus

Testaus tehtiin Netbeansilla käyttämällä Javan System-luokan nanoTime-metodia ottamalla aika ennen ja jälkeen algoritmin suorituksen. Testeissä ei käytetty muuta koodia kuin itse algoritmia, joten esimerkiski käyttöliittymää ei ollut. Lisäksi kaikki muut ohjelmat tietokoneelta oli suljettu testauksen ajaksi. Testissä käytetyt taulukot olivat kokoa 10, 100, 1000, 10000, 100000, 1000000, 10000000 ja 1000000000. Valmiiksi järjestettyä ja käänteisessä järjestyksessä olevaa taulukkoa testattaessa suurimmat taulukot jätettiin pois, koska erot tulivat selvästi ilmi jo pienemmilläkin.

Testit voi toistaa kuka vain ottamalla ajan ennen algoritmin suoritusta ajan System-luokan nanoTime-metodilla ja toisen kerran suorituksen jälkeen. Algoritmin suoritukseen kulunut aika on jälkimmäisen ajan ja ensimmäisen ajan erotus. Tätä metodia käytettäessä suoritusaikaan vaikuttaa esimerksiki muiden ohjelmien aiheuttama rasitus prosessorille sillä se mittaa vain kuinka kauan oikeaa aikaa kului. Kokeilin myös mitata ThreadMXBean-luokalla kuinka kauan algoritmi oli suorittimessa suoritettavana, mutta en tuntunut saavan järkeviä tuloksia. Itse järjestämisohjelma kuitenkin vielä näyttää tämänkin ajan.

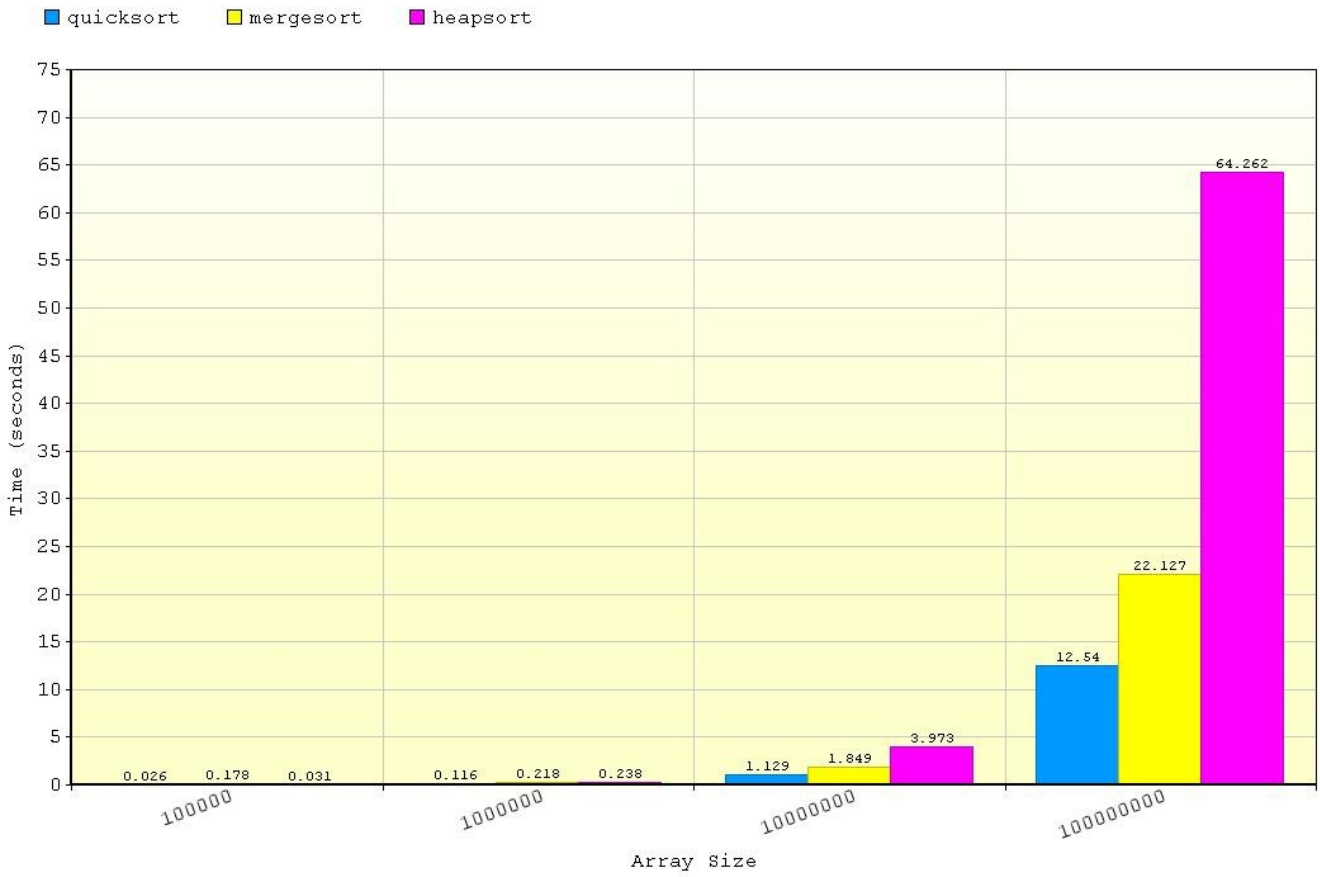
Näiden testien lisäksi ohjelmassa on lisäksi Junit-testit, jotka testaavat jokaisen algoritmin toimivuuden erilaisilla syötteillä, esimerkiksi valmiiksi järjestetyllä taulukolla, käänteisesti järjestetyllä taulukolla ja tyhjällä taulukolla. Junit-testit eivät kuitenkaan testaa algoritmien suorituskykyä.

Alempana testaustuloksiani.

	10	100	1000	10000	100000	1000000	10000000	100000000
Quicksort	0.08	0.12	0.76	7.86	25.95	121.99	1 119.99	12 539.57
Mergesort	0.08	0.23	1.88	14.93	30.82	217.62	1 847.68	20 418.62
Heapsort	0.09	0.30	3.31	7.84	30.75	230.88	3 963.80	63 944.98
Insertion sort	0.07	0.20	4.81	20.44	1 648.08	172 609.62		
Bubble sort	0.08	0.72	6.20	262.19	26 422.00			
Bogo sort	88.21							

Täysin satunnaisen taulukon järjestäminen, jossa suurimpana lukuna oli taulukon koko (ajat millisekunteina)

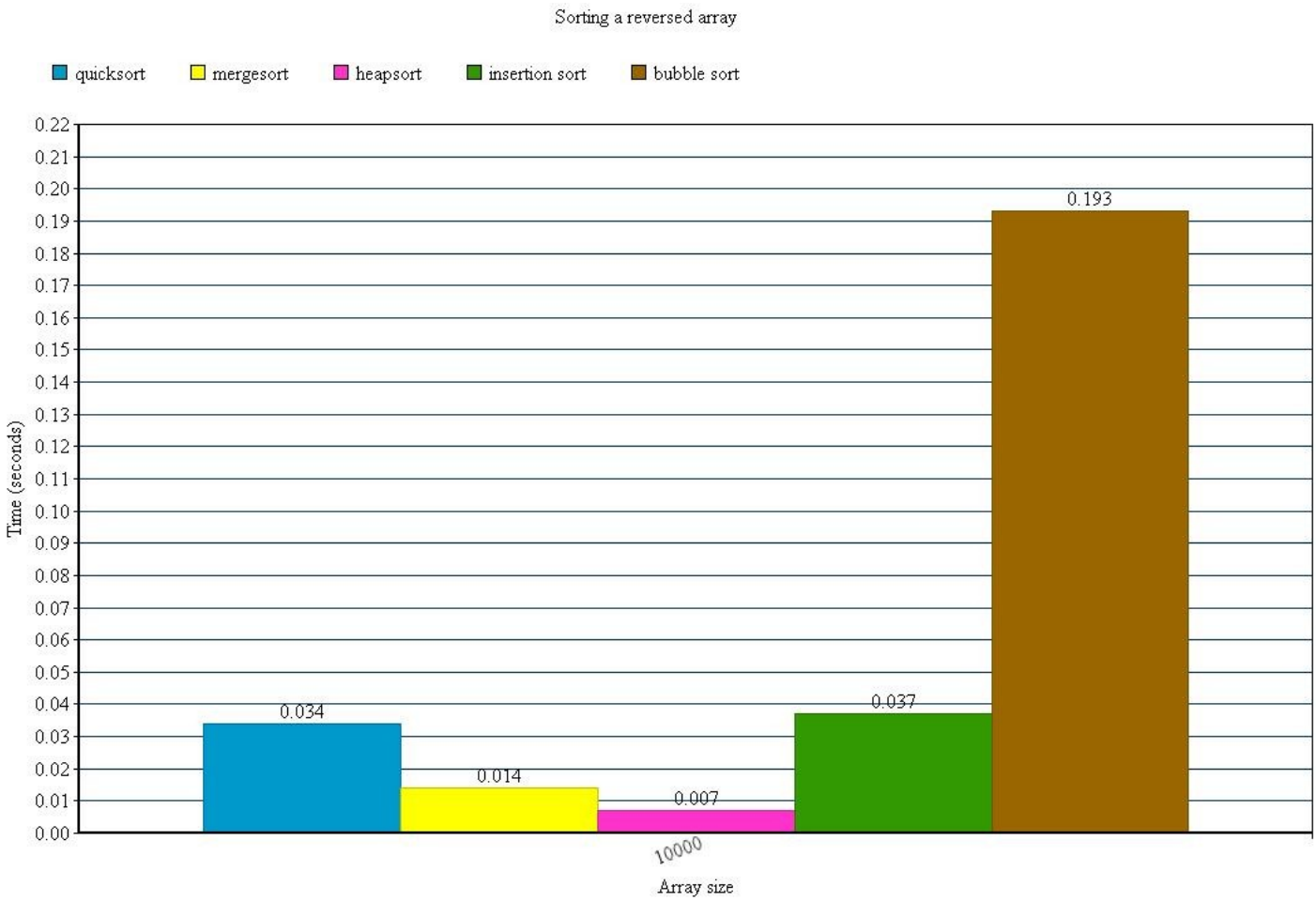
Täysin satunnaisen taulukon järjestämisestä nopeiten suoriutui pikajärjestäminen. Pienellä taulukolla algoritmilla ei ollut juurikaan väliä, mutta erot tulivat ilmi nopeasti. Kuplajärjestäminen oli isommilla taulukoilla jo sen verran hidas, etten saanut sitä edes testattua Bogosortista puhumattakaan.



	1000	10000	100000
Quicksort	6.01	34.58	2 581.36
Mergesort	1.84	14.93	26.81
Heapsort	3.18	7.70	28.97
Insertion sort	5.02	37.88	3 354.18
Bubble sort	5.67	193.50	19 356.72

Käänteisessä järjestyksessä olevan taulukon järjestäminen (ajat millisekunteina)

Käänteisen taulukon järjestämisessä pikajärjestäminen hidastui huomattavasti ja oli jo lähes yhtä hidas kuin lisäysjärjestäminen. Tätä olisi voinut parantaa valitsemalla erilaisen jakoalkion käytettäväksi pikajärjestämisessä. Testit tehtiin käyttämällä taulukon viimeistä eli pienintä lukua.



	1000	10000	100000
Quicksort	4.18	35.94	3 268.18
Mergesort	1.85	14.94	27.05
Heapsort	3.18	7.91	22.20
Insertion sort	0.13	0.88	3.98
Bubble sort	4.23	179.08	17 985.97

Valmiiksi järjestetyn taulukon järjestäminen (ajat millisekunteina)

Valmiiksi järjestettyä taulukkoa järjestettäessä pikajärjestämisellä törmätään samaan ongelmaan kuin käänteisessä järjestyksessä olevaa taulukkoa järjestettäessä. Myös muut algoritmit suoriutuvat tästä samoin lisäysjärjestämistä lukuun ottamatta, joka käy valmiiksi järjestetyn taulukon läpi lineaarisessa ajassa.