Aineopintojen harjoitustyö: Tietorakenteet ja algoritmit

Vertaileva analyysi viiden eri järjestämisalgoritmin nopeudesta

Tekijä: Marko Mäkinen, markoma@iki.fi

Opiskelijanumero: 014248188

Palautettu: 7.9.2014

Tiivistelmä

Tämän harjoitustyön tavoitteena on testata yleisimpiä, mutta eri menetelmillä toimivia järjestämisalgoritmeja ja analysoida näiden toimintaa erilaisissa järjestämisskenaarioissa. Järjestämisalgoritmien toiminnasta on määritelty aika- ja tilavaativuus, sekä suoritusmitattu aikaa useilla eri järjestysskenaariolla, joita ovat muun muassa; epäjärjestys, melkein järjestetty, päinvastainen järjestys ja järjestetty.

Testattavia algoritmeja ovat lisäysjärjestäminen (insertion sort), kuplajärjestäminen (bubble sort), pikajärjestäminen (quick sort), lomitusjärjestäminen (merge sort) sekä valintajärjestäminen (selection sort).

Johdanto

Harjoitustyössä testataan yleisimpien järjestämisalgoritmien suoritusnopeutta sekä analysoidaan näiden käyttäytymistä erilaisissa järjestämistilanteissa. Tässä dokumentissa käsitellään järjestämisalgoritmeja yleisesti ja miksi on valittu juuri nämä algoritmit. Jokaista valittua algoritmia tutkaillaan lähemmin kysyen: miksi tämä on hyvä tai huono ja minkälaisissa sovelluksissa tätä voidaan hyödyntää.

Harjoitustyössä testattaviksi järjestämisalgoritmeiksi on valittu kuplajärjestäminen (bubble sort), lisäysjärjestäminen (insertion sort), Pikajärjestäminen (quicksort), lomitusjärjestäminen (merge sort) ja valintajärjestäminen (selection sort). Valituista algoritmeista kahden aikavaativuus on O(n log n) näin erottuen selkeästi muista testattavista algoritmeista, joiden meridiaani aikavaativuudet ovat luokkaa O(n2).

Järjestämisskenaarioiksi on valittu hyvin monipuolinen joukko taulukoita. Monipuolisuuden tavoitteena on löytää kullekin algoritmille ne huonoimmat ja parhaimmat tilanteet. Erilaisia taulukkoja on viisi kappaletta: satunnainen (ei-duplikaatteja), satunnainen (vain numerot n = 0..4), melkein järjestetty (5 – 10 % epäjärjestyksessä), järjestetty ja käänteinen järjestys.

Tässä harjoitustyössä käsitellään ensisijaisesti järjestämisnopeutta. Tilan tarvetta ei ole mitattu sen monimutkaisen mittaustavan vuoksi, mutta jokaisesta järjestämisalgoritmista on tilavaativuus määritelty yksityiskohtaisemmassa tarkastelussa.

Testausohjelma ja algoritmit on kokonaisuudessaan ohjelmoitu Javalla.

Järjestämisalgoritmit

Järjestämisalgoritmit voidaan jakaa kahteen eri yläluokkaan: Vertailevat ja ei-vertailevat. Vertailevat algoritmit ovat järjestämisalgoritmeja, jotka vertailevat kahta elementtiä keskenään (Yleensä ”pienempi tai yhtä suuri kuin” operaattorilla) (wikipedia: Comparison sort). Ei-vertailevien algoritmien toiminta perustuu osittaiseen esitietoon järjestettävistä alkiosta, kuten tietoon järjestettävän alueen koosta.

Järjestämisalgoritmit voidaan jakaa luokkiin käytetyn järjestämismenetelmän mukaisesti. Tässä harjoitustyössä käsiteltyjä järjestämismenetelmiä ovat jakojärjestäjät (exchange sorts), lisäysjärjestäjät (insertion sorts), valintajärjestäjät (selection sorts) ja lomitusjärjestäjät (merge sorts). Näiden lisäksi on luokiteltu muun muassa distribution sorts, concurrent sorts ja hybrid sorts. Niin ikään voidaan järjestämisalgoritmit jakaa rekursiivisiin ja ei-rekursiivisiin eli iteratiivisiin algoritmeihin.

Taulukko 1: Järjestämisalgoritmien luokittelu järjestämismenetelmien mukaan. (Ocampo 2008) Valitut järjestämisalgoritmit on lihavoitu.

	Vertailevat				**Ei-vertailevat**
	Vaihtojärjestäjät (jako)	**Lisäysjärjestäjät**	**Valintajärjestäjät**	**Lomitusjärjestäjät**	
Iteratiiviset	**Kuplajärjestäminen,** Cocktail/shaker sort, Comb sort, Gnome sort	Linear insertion sort, Binary sort, **Lisäysjärjestäminen** Shell sort	**Valintajärjestäminen,** Heap sort	In-place merge sort	
Rekursiiviset	**Pikajärjestäminen**	Library sort	Tree sort	**Klassinen Lomitusjärjestäminen**	Radix sort, Bucket sort, Laskemisjärjestäminen

Testattavat järjestämisalgoritmit

Tähän analyysiin valitut järjestämisalgoritmit on valittu mukaan, koska ne ovat tunnetuimpia ja yleisimmin käytetyimpiä algoritmeja. Ne kaikki edustavat eri järjestämismenetelmiä (ks. Taulukko 1), sekä ne kuuluvat vertailevien järjestämisalgoritmien joukkoon.

Taulukko 2: Testattavien järjestämisalgoritmien vertailu (Floréen 2014)

Järjestämisalgoritmi	**Keskiarvo**	**Huonoin tapaus**	**Menetelmä**	**Vakaa**
Kuplajärjestäminen	$O(n^2)$	$O(n^2)$	Vaihtojärjestäjä	on
Lisäysjärjestäminen	$O(n^2)$	$O(n^2)$	Lisäysjärjestäjä	on
Pikajärjestäminen	O(n log n)	$O(n^2)$	Jakojärjestäjä	ei
Lomitusjärjestäminen	O(n log n)	O(n log n)	Lomitusjärjestäjä	on
Valintajärjestäminen	$O(n^2)$	$O(n^2)$	Valintajärjestäjä	Riippuu toteutuksesta

Testausohjelmassa jokainen algoritmi laajentaa abstraktin yläluokan Algo:n ja suorittaa sort(int[] a) metodin, jolle annetaan parametrina järjestettävän taulukko.

```
public abstract void sort(int[] a);
```

Kuplajärjestäminen (Bubble sort)

Kuplajärjestäminen yksinkertainen ja tunnettu, mutta tehoton järjestämisalgoritmi. Sen toiminta perustuu toistuvaan epäjärjestyksessä olevien elementtien vertailuun ja vaihtamiseen jos ne ovat väärässä järjestyksessä. (Leiserson et al. 2009) Sen aikavaativuus on huonoimmassa ja keskiarvotapauksessa luokkaa O(n2). Täten suurilla taulukoilla on kuplajärjestäminen hitautensa vuoksi hyödytön. Kuplajärjestämisen tilavaativuus on O(1).

Kuplajärjestämisen paras puoli on sen kyky ”huomata” listan olevan järjestyksessä. Tällaisessa tilanteessa sen aikavaativuus on vain O(n). Se on silti hidas verrattuna lisäysjärjestämiseen vastaavassa tilanteessa, sillä sen on käytävä tietoaluetta läpi paljon ”syvällisemmin”.

Valintajärjestäminen (Selection sort)

Valintajärjestäminen on kuten kuplajärjestäminen mutta vähemmän kuin se. Valintajärjestäminen eroaa hyvin vähän kuplajärjestämisestä. Idealtaan niin, että sillä on ”valittuna” taulukon indeksi, jonka alla luvut ovat järjestyksessä. Vaihtaminen tapahtuu muuten samaan tapaan kuin kuplajärjestämisessä. Valintajärjestämisen aikavaativuus on lähes $O(n^2/4)$. (Edjlal et al. 2011) Huonoimmassa tapauksessa sen aikavaativuus on luokkaa $O(n^2)$.

Lisäysjärjestäminen (Insertion sort)

Lisäysjärjestäminen on tehokas algoritmi pienien taulukoiden järjestämiseksi. Lisäysjärjestäminen toimii samalla tavalla kuten korttien järjestäminen kädessä. Aloitetaan tyhjästä kädestä ja aina seuraava nostettu kortti laitetaan oikeaan paikkaan järjestetyssä kädessä. (Leiserson et al. 2009)

Lisäysjärjestäminen toimii vakaasti. Se on keskiarvoiselta aikavaativuudeltaan luokkaa O(n2), vaikka se on nopea jo valmiiksi järjestyksessä olevalla taulukolla, jolloin sen aikavaativuus luokkaa O(n). Ongelmana on, että ”keskiarvoinen tapaus” on usein yhtä hidas kuin huonoin tapaus. (Leiserson et al. 2009) Huonoin tapaus syntyy kun taulukko on käänteisessä järjestyksessä, tällöin sen aikavaativuus on luokkaa O(n2).

Pikajärjestäminen (Quicksort)

Nopea, rekursiivinen ja ei-vakaa järjestämisalgoritmi, joka toimii *divide and conquer* periaatteella. Sen aikavaativuus on keskiarvoltaan luokkaa O(n log n) ja huonoimmassa tapauksessa se on luokkaa O(n2).

Pikajärjestäminen toimii pivot -idealla, johon perustuen tapahtuu taulukon jako kahteen (divide). Jaetuissa taulukoissa on jää vasempaan taulukkoon pivottia pienemmän luvut ja oikealle sitä suuremmat. Tämän jälkeen kutsutaan vasenta ja oikeaa listaa rekursiivisesti.
Quicksort ei kopioi tietoa. Se vain vaihtaa vaihtaa arvojen sijaintia taulukossa, joten sen tilavaativuus on pieni.

Pikajärjestämisen huonoin puoli on jo valmiiksi järjestyksessä olevan taulukon läpikäynnin hitaus. Päinvastoin kuin lomitusjärjestämisessä on pikajärjestämisen aikavaativuus järjestyksessä olevan listan läpikäymiseksi lineaarisessa suhteessa puunkorkeuteen. (Goodrich 2002) Lineaarinen on hitaampi kuin logaritminen.

Pikajärjestäminen ylivoimaisena monen tilanteen järjestämisalgoritmina on suositeltu vakiojärjestämisalgoritmi. (Hoare 1962) Esimerkiksi Javan vakiometodi java.util.Arrays.sort() on modifioitu kahden pivotin versio pikajärjestämisestä. (http://docs.oracle.com/javase/7/docs/api/java/util/Arrays.html#sort(int[]))

Lomitusjärjestäminen (Merge sort)

Nopea, rekursiivinen ja vakaa järjestämisalgoritmi, joka toimii *divide and conquer* periaatteella. Sen aikavaativuus on keskiarvoltaan luokkaa O(n log n).

Lomitusjärjestäminen toimii samaan tapaan kuin pikajärjestäminen, jossa taulukko jaetaan aina kahteen. Nämä listat järjestetään ja yhdistetään. Yhdistämisessä taulukkojen arvot liitetään ”lomittain” (merging) samaan

taulukkoon.
Lomitusjärjestäminen on hyvin ennustettavissa oleva järjestysalgoritmi. Se suorittaa keskimäärin 0,5 * log (n) ja log (n) vertailua per alkio ja väliltä log (n) ja 1,5 * log (n) vaihtoa per vertailu. (http://www.sorting-algorithms.com/merge-sort)

Lomitusjärjestäminen sopii moneen järjestämistilanteeseen, esimerkiksi: Kun vaaditaan vakautta tai kun järjestetään paljon pieniä taulukoita.

Testiskenaariot

Erilaisia testiskenaarioita luotiin kattavasti, jotta algoritmien meridiaaninopeutta pystyttäisiin määrittelemään mahdollisimman tarkasti ja jotta saataisiin jokaisen järjestämisalgoritmin toimintaa ääritilanteissa parhaiten hahmotettua sekä löydettyä kullekin algoritmille paras ja huonoin järjestämistapaus.

Skenaariot:

- **Epäjärjestys:** Javan Random generaattoria hyödyntämällä luodaan taulukkoon 32-bittisiä lukuja. Ei samoja lukuja.
- **Epäjärjestys muutamalla luvulla:** Javan Random generaattoria hyödyntämällä luodaan taulukko, jossa luvut ovat väliltä 0-4.
- **Melkein järjestetty:** Järjestetty lukujono, josta 5-10% on sekoitettu.
- **Järjestetty:** Täysin järjestyksessä oleva lukujono.
- **Päinvastainen:** Laskevassa järjestyksessä oleva taulukko.

Testitaulukot on jaettu kahteen koko luokkaan: isot ja pienet. Pienet taulukot ovat kokoluokkaa 5 - 500 alkiota. Isot ovat kokoluokkaa 500 – 10 000 alkiota. Jokainen taulukko luodaan aina uudelleen ennen ajanmittausta.

Nopeuden mittaus

Tässä harjoitustyössä esitetyt tulokset on testeistä jotka on pienillä taulukoilla suoritettu 10000 kertaa ja suurilla taulukoilla 1000 kertaa. Taulukot alustetaan aina ennen ajanmittausta. Ajanmittauksessa on käytetty Javan vakiokirjaston työkalua System.nanoTime(), joka antaa nopeuden nanosekunteina. Javan long -muotoinen tulos jaetaan suorituskertojen määrällä. Tämä aiheuttaa pienen pyöristysvirheen, joka tässä tilanteessa on merkityksetön. Kaikki testitulokset tallennetaan taulukkoon .csv yhteensopivassa muodossa, jossa sarakkeet on erotettu kaksoispisteillä.

Testituloksien ajanmittausmetodi

```
private static long timing(Arr array, Algo algo) {
    int[] a = array.get(); // reproduce of an array
    long start = System.nanoTime();
    algo.sort(a);
    long end = System.nanoTime();
    return end - start;
}
```

Testiohjelmassa taulukot suorittavat abstraktin yläkuokan ”Arr” get() metodin, joka palauttaa uudelleen luodun taulukon. Tällä ehkäistään taulukon jäämistä muistiin ja täten virheellistä mittaustulosta.

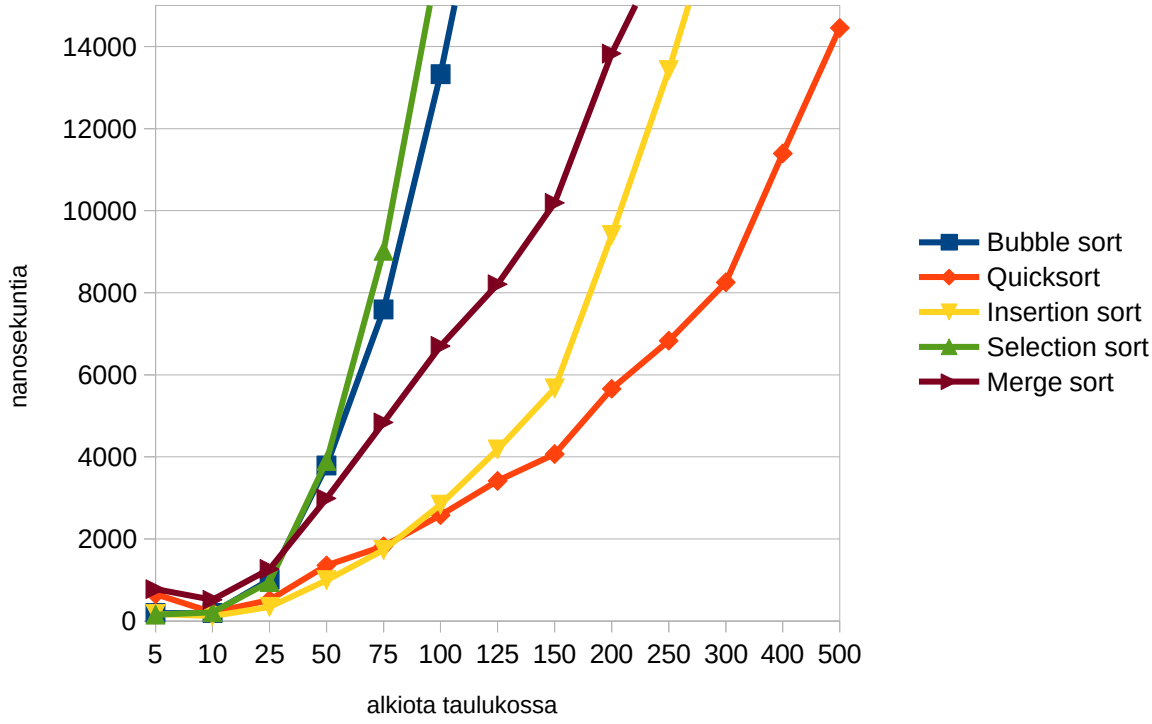
Testikoneena on Macbook Air 1.8Gh Intel core i5, 4Gt 1600mhz DDR3, OSX 10.9. Javan versio on 1.7.

Testitulokset

Pienet taulukot

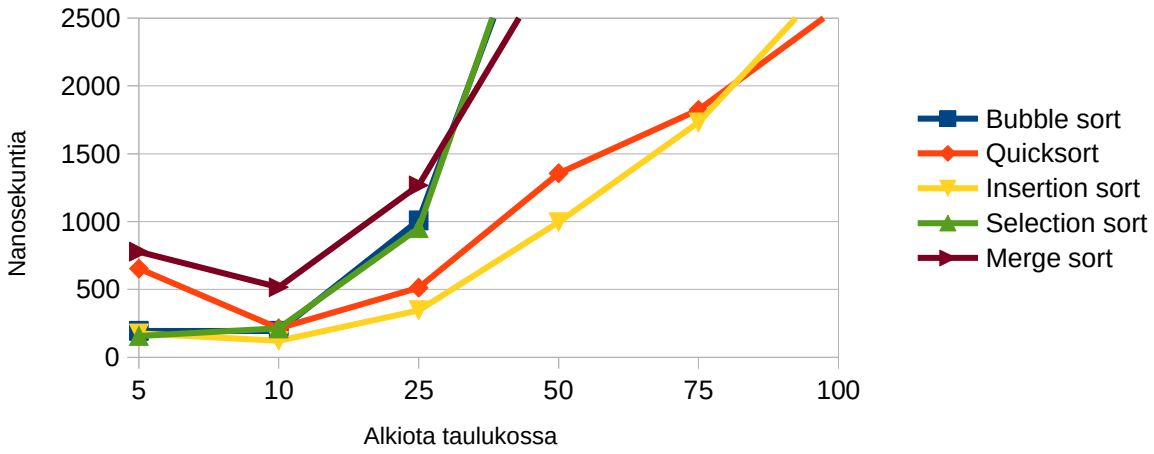
Testit suoritettiin taulukoilla, joiden koot olivat 5, 10, 25, 50, 75, 100, 125, 150, 200, 250, 300, 400 ja 500 alkiota. Testi suoritettiin jokaisella skenaariolla ja algoritmilla kymmenen tuhatta kertaa. Suoritusaika koko testin suorittamiseksi testiympäristössä oli noin viisi minuuttia.

Kuva 1. Keskiarvonopeudet pienillä taulukoilla



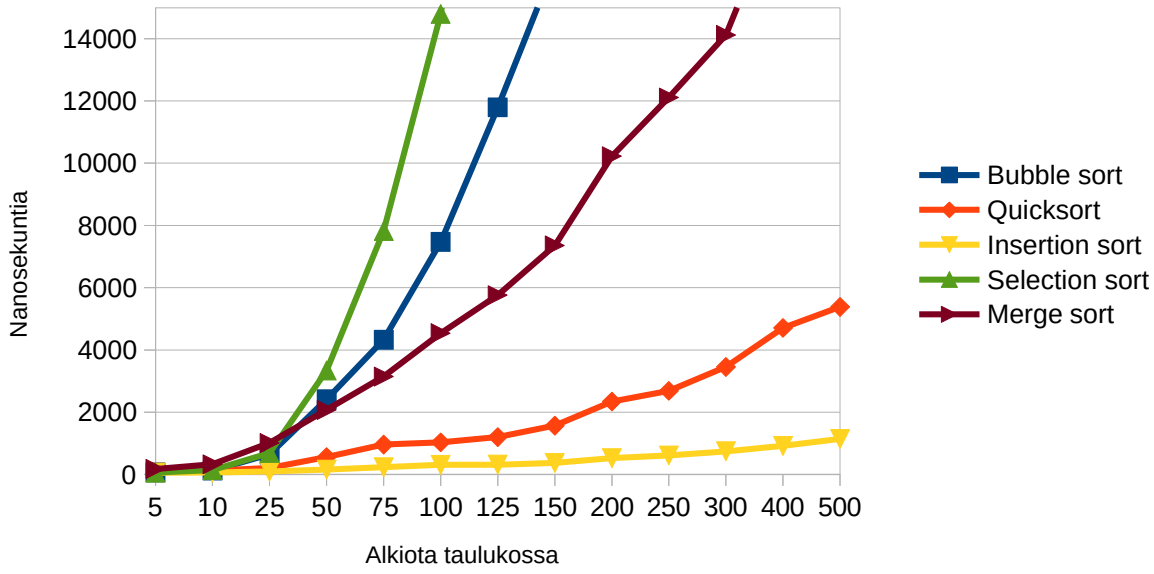
Pienillä alkioilla mitatut nopeudet ovat juuri odotetunlaiset. Aikavaativuudeltaan $O(n^2)$ järjestämisalgoritmit hidastuvat hyvin nopeasti taulukon kasvaessa, verrattuna aikavaativuudeltaan O(n log n) algoritmeihin. Tarkastellaan vielä keskiarvonopeuksia lähemmin.

Kuva 2. Keskiarvonopeudet lähikuva



Kuvasta 2. havaitsemme lisäysjärjestämisen odotetusti olevan pikajärjestämistä nopeampi pienillä taulukoilla. Taulukon koon kasvaessa sataan ja sitä suuremmaksi kääntyy pikajärjestäminen lisäysjärjestämistä nopeammaksi. Kaikki testatut järjestämisalgoritmit toimivat keskimääräisesti nopeasti pienillä taulukoilla. Kiinnostavaa on myös lomitusjärjestämisen ”hitaus” suhteessa lisäysjärjestämiseen, lomitusjärjestämisen aikavaativuuden ollessa luokkaa O(n log n) ja lisäysjärjestämisen ollessa $O(n^2)$. Tämä ero johtuu lisäysjärjestämisen ylivoimaisesta nopeudesta jo järjestyksessä olevilla taulukoilla (ks. Kuva 3.).

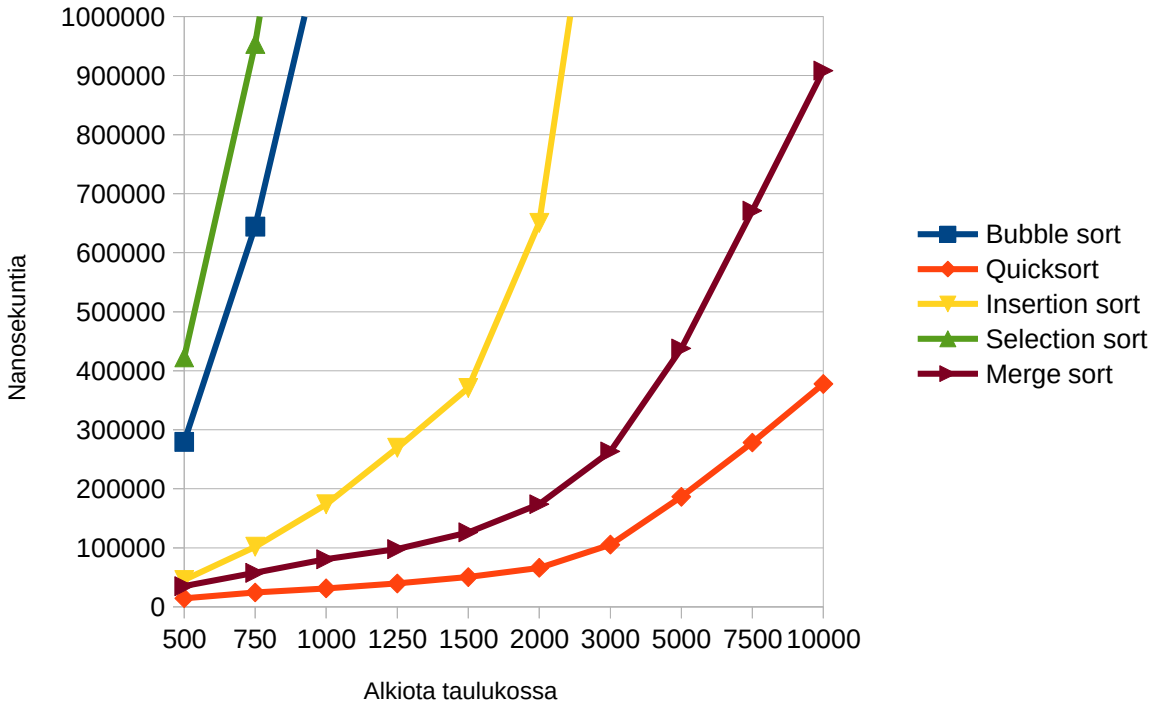
Kuva 3. Järjestetty taulukko



Suuret taulukot

Suurien taulukoiden testi suoritettiin taulukoilla, joiden koot olivat 500, 750, 1000, 1250, 1500, 2000, 3000, 5000, 7500 ja 10000 alkiota. Testi suoritettiin jokaisella skenaariolla ja algoritmilla tuhat kertaa. Suoritusaika koko testin suorittamiseksi testiympäristössä oli noin tunnin.

Kuva 4. Keskiarvonopeudet suurilla taulukoilla

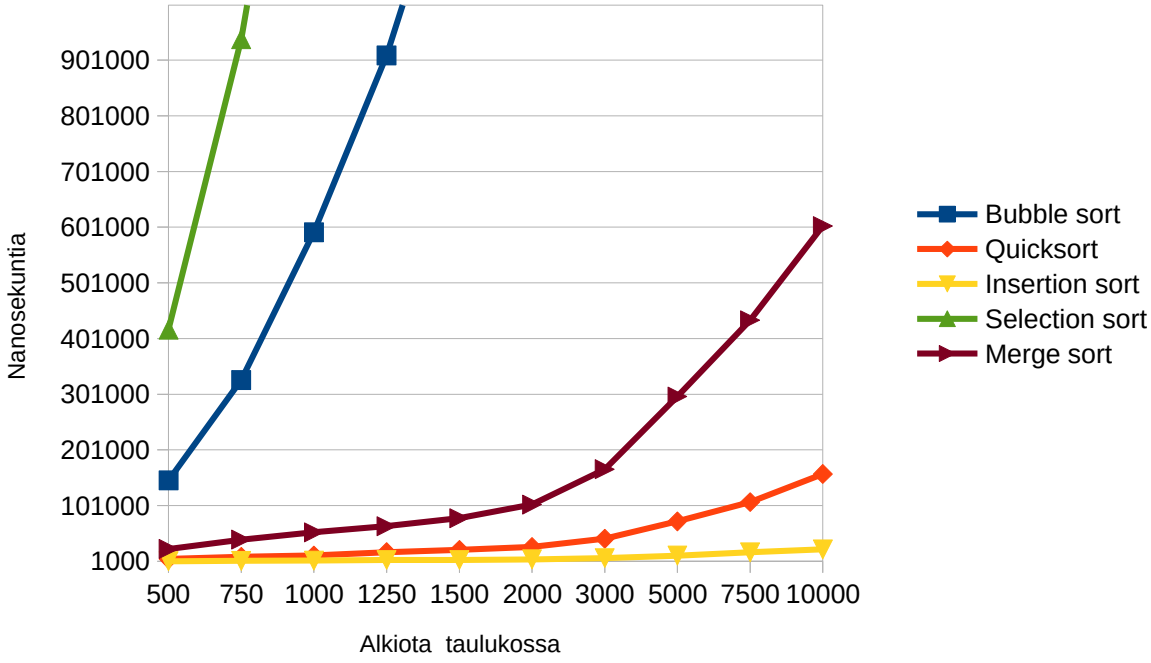


Suurilla taulukoilla huomaamme lisäysjärjestämisen hitauden näkyvän ja nopeuden muuttuvan lähemmäksi aikavaativuutta $O(n^2)$. Suuremmilla taulukoilla lomitusjärjestämisen O(n log n) nopeus tulee esiin. Kupla- ja valintajärjestäminen $O(n^2)$ ei pärjää suuremmin taulukoilla O(n log n) järjestämisalgoritmeille, ja hitaus tekeekin näistä lähes täysin käyttökelvottomia suurien taulukoiden järjestämisessä.

Niin ikään lisäysjärjestäminen järjestää suurillakin taulukoilla jo järjestyksessä olevan taulukon ylivoimaisella nopeudella (ks kuva 5.). Tämä ylivoimaisuus kumminkin pahoin tasoittuu muissa järjestämistilanteissa, joiden aikavaativuus on kaikissa muissa tilanteissa luokkaa $O(n^2)$.

Odotetusti pikajärjestäminen suoriutuu parhaiten suurien taulukoiden järjestämisessä, ja on näin ollen suositelluin vaihtoehto niille.

Kuva 5. Järjestetyn taulukon järjestämisnopeus



Yhteenveto

Kupla- ja valintajärjestäminen eivät poikkea toisistaan lähes ollenkaan, vaikka kuplajärjestäminen on hitusen hitaampaa suurilla taulukoilla kuin valintajärjestäminen. Molemmat ovat joka tapauksessa hyvin hitaita suurilla taulukoilla.

Lisäysjärjestäminen on hyvin nopea pienillä taulukoilla, joilla ylivertainen nopeus jo järjestyksessä olevan taulukon läpikäynnissä kääntää keskiarvonopeuden kilpailukykyiseksi. Lisäysjärjestäminen onkin täten suositeltu menetelmä pienien ja lähes järjestyksessä olevien taulukoiden järjestämiseksi. Suurilla taulukoilla nopeus ei ole kilpailukykyinen O(n log n) järjestämisalgoritmien kanssa.

Lomitusjärjestäminen paljastui hyvin nopeaksi järjestämisalgoritmiksi, ja pärjää hyvin sen kilpailijalle pikajärjestämiselle. Lomitusjärjestäminen tulee kysymykseen kun tarvitaan järjestämisalgoritmia suurille taulukoille jolta vaaditaan vakautta ja hyvää ennustettavuutta. Pikajärjestäminen oli odotetusti nopein all-around järjestämisalgoritmi. Lisäysjärjestäminen on nopeampi pienillä taulukoilla kuin pikajärjestäminen, mutta pikajärjestäminen tasaisesti hyvin nopea jokaisessa järjestämistilanteessa, vaikkakin sen pienenä heikkoutena on jo järjestyksessä olevan taulukon läpikäynnin hitaus. Pikajärjestäminen soveltuukin hyväksi perusjärjestäjäksi, jos vakautta ei tarvita.

Lähteet

Edjlal, R., Edjlal, A. & Moradi, T., 2011. A sort implementation comparing with Bubble sort and Selection sort. In *ICCRD2011 - 2011 3rd International Conference on Computer Research and Development.* pp. 380–381.

Floréen, P. (HIT), 2014. 58131 Tietorakenteet ja algoritmit.

Goodrich, M.T., 2002. Data Structures and Algorithms in Java. *Science*, p.800.

Hoare, C.A.R., 1962. Quicksort. *The Computer Journal*, 5, pp.10–16. Available at: http://comjnl.oxfordjournals.org/cgi/content/abstract/5/1/10\nhttp://comjnl.oxfordjournals.org/content/5/1/10.full.pdf+html.

Leiserson, C.C.E. et al., 2009. *Introduction to Algorithms, Third Edition*, Available at: http://www.amazon.com/Introduction-Algorithms-Third-Thomas-Cormen/dp/0262033844\nhttp://books.google.com/books?hl=en&lr=&id=NLngYyWFl_YC&oi=fnd&pg=PR13&dq=Introduction+to+algorithms&ots=BxVtEG-hJc&sig=LyOyin9nLdYbGCwRU5l3Xu5IScI.

Ocampo, J., 2008. An empirical comparison of the runtime of five sorting algorithms. *International Baccalaureate Extended* …. Available at: http://uploads.julianapena.com/ib-ee.pdf [Accessed August 23, 2014].