Testaus

Testasin bubble sort, mergesort ja quicksort algoritmeja lähinnä satunnaisesti järjestetyillä taulukoilla joiden koko oli 0 – 10 000 000 alkiota. Tätä suuremmat taulukot aiheuttavat ainakin luomassani pääohjelmassa virheilmoituksen mutta silloin mennään muutenkin jo melkein yli int muuttujien kapasiteetin. Arvot olivat suhteellisen pieniä, eli itseisarvoltaan alle 100000, positiivisia kokonaislukuja. Testasin tosin myös että negatiiviset luvut eivät aiheuta virhettä. Järjestäminen tapahtuu oikein. Kaikille kolmelle algoritmille löytyy automaattiset junit testit. Lisäksi pääohjelma on varsinaisesti vain näiden algoritmien testaustyökalu.

Aikavaatimukset olivat testien perusteella odotetut. Alla on testien tuloksia, jossa rivin viimeinen numero on aina käytetty aika millisekuntteina:

pika 1000 alkioinen taulukko 1
pika 10000 alkioinen taulukko 1
pika 100000 alkioinen taulukko 14
pika 1000000 alkioinen taulukko 145
pika 10000000 alkioinen taulukko 1484

msort 1000 alkioinen taulukko 1
msort 10000 alkioinen taulukko 2
msort 100000 alkioinen taulukko 23
msort 1000000 alkioinen taulukko 268
msort 10000000 alkioinen taulukko 2978

kupla 1000 alkioinen taulukko 5
kupla 10000 alkioinen taulukko 322
kupla 100000 alkioinen taulukko 32379

Yllä olevista luvuista on helppo nähdä että bubble sort oli hyvin hidas, ja hyvin tarkkaan n^2 ajassa toimiva koska aineiston kymmenkertaistuminen johtaa käytetyn ajan satakertaistumiseen. 10000000 alkioisen taulukon järjestäminen olisi kestänyt noin 3 200 000 sekunttia eli useita vuorokausia. On myös selvää että mergesort ja quicksort algoritmeissa aika kasvaa hiukan nopeammin kuin n, eli ilmeisesti n log n. Quicksort käyttää aikaa noin puolet siitä mitä mergesort vastaavassa tapauksessa.

Yllä olevissa testeissä alkiot olivat siis satunnaisia, mutta testasin myös toimintaa kun taulukko oli valmiiksi jo melkein järjestyksessä. Tällöin bubble sort oli ylivoimaisesti nopein ja mergesort hitain. Quicksort vei tässäkin aikaa noi puolet mergesortin ajasta.

Quicksortin toimintaa testasin muutamalla erilaisella pivot arvoilla. Kun pivot arvoksi valittiin yksinkertaisesti ensimmäisen alkion arvo, 10000000 alkioisen taulukon järjestäminen kesti 1404 millisekunttia. Kun pivot arvoksi valittiin ensimmäisen, keskimmäisen ja viimeisen alkion arvojen mediaani 10000000 alkioisen taulukon järjestäminen kesti 1484 millisekunttia. Mediaanin valitseminen siis hiukan hidastaa algortimia. Se kuitenkin kannattaa koska kun kokeilin järjestää melkein järjestettyä taulukkoa ensimmäisen alkion ollessa pivot arvo, koko ohjelma kaatui liian syvän rekursion seurauksena vähänkään suuremmalla taulukolla. Teoriassa pivot arvoa ei voi ilmeisesti valita niin että voisi olla täysin varma ettei tällaisia ongelmia tule. Mediaani tekee ongelmat kuitenkin hyvin epätodennäköisiksi.

Kuvassa vertailtu mergesortia (punainen) ja quicksorttia (vihreä):