Tehokkuus

Ohjelma ratkaisee pääongelemansa elikkä algoritmien ajallisen vertailun määrittelydokumentin määrittämällä tavalla. Algoritmit on toteutettu mallin mukaan. Ohjelman pahin tapaus suorittaa jokaisen algoritmin kerran ja kopioi jokaiselle algoritmille oman taulukon syötteenä annetusta taulukosta. Algoritmien omien toimintojen lisäksi oikeastaan vain käytetään muutamaa if lauseketta tarkistaakseen mitkä algoritmit on valittu vertailun kohteeksi ja tarkistetaan järjestelmän aika kahdesti. Ajantarkistuksista otetaan ylös kaksi long arvoa, joista lasketaan yksinkertaisella miinus ja jakolaskulla algoritmin suorittamiseen kulunut aika millisekuntteina. Ohjelmassa on mahdollisuus myös tulostaa käytettävät taulukot, mutta tähän kulunutta aikaa ei oteta huomioon vaativuus analysissä sillä toiminnon voi myös ottaa pois päältä.

Perustelu

Ohjelman suorituksen perusidea pseudokoodina:

startComparison ()

1. if mergeSortSelected == true
2. timedSort(mergesort)
3. if quickSortSelected == true
4. timedSort(quickSort)
5. if heapSortSelected == true
6. timedSort(heapSort)

If lauseeseen kuluva aika on vakio ja niitä on kolme. Oleellisinta aikavaativuuden kannalta on se, että jokainen algoritmi suoritetaan enintään kerran, joista pahimmassa tapauksessa hitain on quickSort $O(n^2)$.

timedSort(algorithmClass)

1. algorithmClass.clone(A) // luokkaa kloonaa taulukon omaa käyttöönsä
2. start = currentTimeInNanoseconds
3. algorithmClass.sort()
4. elapsedTime = currentTimeInNanoseconds – start
5. print(elapsedTime/10000000)

Algoritmin suorittamisen lisäksi siis tarkistetaan tämän hetkinen aika kahdesti ja lasketaan yksinkertaisilla laskutoimituksilla kulunut aika. Aikaan liittyvät operaatiot ovat kaikki vakioaikaisia ja syötteestä riippumattomia. Syötteen mukaisia tauluja luodaan yhteensä enintään neljä joten tähän käytetty tila on O(4n). Jokaisella algoritmilla on siis käytössä yksi näistä tauluista ja ohjelma itse pitää muistissa omaa kopiotaan tauluista. Itse algoritmeista eniten tilaa vie mergeSort eli O(n). tilavaativuus on siisk oko testin osalta O(n) eikä se ylitä sitä.

Ohjelman puutteet

Valitettavasti itse ohjelman demoamisosuus jää vajaaksi. Demoaminen oli hankala toteuttaa visuaalisesti järkevästi haluamallani tavalla algoritmien rekursiivisuuden takia. Joten keskityin täysin ohjelman oleelliseen osioon eli aikavertailuun. Itse aikavertailuun olen tyytyväinen. Ohjelmaan voisi lisätä enemmän tapoja luoda vertailutauluja, mutta tämän on enemmänkin laajennusmahdollisuus, kuin puute.