Toteutus

Toteutin Javalla buble sort, mergesort ja quicksort algoritmit. Toteutin lisäksi ohjelman jolla voi vertailla näiden algoritmien käyttämää aikaa eri kokoisilla taulukoilla antamalla käytettävän algoritmin ensimmäisen kirjaimen ja taulukon koon, esimerkiksi 'q 100000'. Tämä on ihan eri asia kuin alunperin ajattelin, jeli tekstitiedostojen järjestäminen, mutta tämä on selkeästi parempi idea ainakin jos uskoo että järjestäminen tosiaan tapahtuu. Pääohjelmassa ei ole varauduttu käyttäjän virheisiin joten se kaatuu helposti. Itse algoritmit toimivat empiiristen testien perusteella oikeilla aikavaatimuksilla.

Lähteinä käytin Wikipediaa ja vogella.com sivuston tutoriaaleja.

Bubble sort algoritmin aikavaatimus on helppo laskea. Algoritmi on muotoa:

```
for (n kertaa) {
        for (n kertaa) {...}
}
```

Eli yksi for lause toisen for lauseen sisällä. Selvästi aikavaatimus on huonoimmillaan O(n^2). Koska joka kierroksen jälkeen kuitenkin tarkastetaan oliko taulukko järjestyksessä, voi algoritmi olla parhaassa tapauksessa, eli kun taulukko on valmiiksi melkein järjestetty, hyvin nopea aikavaatimuksella O(n). Tilavaatimus on O(1) koska algoritmi luo vain muutaman muuttujan, joita siis on vakio määrä.

Mergesort algortimi toimii seuraavasti:

1) Jaa taulukko osiin kunnes jokaisen osan koko on yksi.
2) Yhdistä näitä taulukoita samalla järjestäen kunnes jäljellä on enää yksi taulukko.

Kohta 1 vie aikaa n. Kohta 2 vie aikaa n + n/2 + n/2 + n/4 ja niin edelleen. Yhdistämisiä tapahtuu logaritminen määrä kertoja. O notaatiossa aika vaatimukseksi pöyristyy O(n log n). Tilavaatimus on O(n) koska algoritmi käyttää n kokoisia aputaulukoita.

Quicksort on hiukan vaikeampi analysoida koska siinä oleellista on keskimääräinen aikavaatimus joka eroaa selvästi huonoimman tapauksen aikavaatimuksesta. Quicksort toimii seuraavasti:

1) Valitaan joku alkiosta vertailuarvoksi eli pivot-arvoksi.
2) Järjestetään taulukko siten että pivot arvoa pienemmät arvot tulevat sitä ennen ja suuremmat sen jälkeen. Pivot-arvo on nyt lopullisella paikallaan.
3) Seuraavaksi sovelletaan rekursiivisesti edellistä erikseen pivotia pienempään taulukon osaan ja suurempaan taulukon osaan, kunnes päästään korkeintaan yhden alkion kokoisiin alilistoihin.

Pahimmassa tapauksessa pivot-arvo valitaan aina siten että toiselle puolelle tulee vain yksi alkio. Silloin aika vaatimus on n + n-1 + n-2 ja niin edelleen, mikä pyöristyy muotoon O(n^2). Jos pivot kuitenkin valitaan siten että keskimäärin jaossa molemmille puolille päätyy noin puolet on aikavaatimus n + n/2 + n/2 + n/4 ja niin edelleen, jossa jakojen määrä on logaritminen. Eli Aikavaatimus on muotoa O(n log n). Huonoimmillaa tilavaatimus on O(n) koska rekursion syvyydeksi tulee n. Käytin pivot-arvona ensimmäisen, keskimmäisen ja viimeisen arvon mediaania, jolloin huonoimman tapauksen pitäisi olla hyvin harvinainen. Tilavaatimus olisi ollut teoriassa mahdollista toteutttaa parempana: O(log n).