Tietorakenteet ja Algoritmit

Lajittelualgoritmin suorituskykykilpailu

Karusalmi Joona  
Paakki Iikka-Matti  
Venäläinen Waltteri

Harjoitustyö

Huhtikuu 2017

Tieto- ja viestintätekniikka

Kyberturvallisuus

Sisällys

[1 Johdanto 2](#_Toc479368975)

[2 Lajittelualgoritmit 2](#_Toc479368976)

[3 Kuplalajittelu, Bubble sort 2](#_Toc479368977)

[4 Kompleksisuus 3](#_Toc479368978)

[4.1 Theta, θ 3](#_Toc479368979)

[4.2 ”O” (Omikron) 4](#_Toc479368980)

[4.3 Omega, Ω 4](#_Toc479368981)

[5 Vertailua 4](#_Toc479368982)

[6 Yhteenveto 6](#_Toc479368983)

[7 Lähteet 6](#_Toc479368984)

[Kuvio 1. Algoritmin koodi 5](#_Toc479366507)

[Kuvio 2. Luokka D331 5](#_Toc479366508)

[Kuvio 3. Luokka 422 5](#_Toc479366509)

# Johdanto

Aiheenamme on lajittelualgoritmin suorituskykykilpailu. Toteutustapana suunnittelimme käyttää jotain koodattua lajittelu algoritmia, kuten esimerkiksi kuplalajittelua, antaa tälle jokin tietty määrä alkioita lajiteltavaksi, mittauttaa lajitteluun käytetty aika ja verrata tämän suhdetta kahdella eri koneella suoritetun kokeen prosessorien tehoihin. Tarkoituksena on myös avata lajittelualgoritmien teoriaa. Algoritmin ja lajiteltavan datan täytyy olla identtiset molemmilla alustoilla.

# Lajittelualgoritmit

Lajittelualgoritmit, tunnetaan myös järjestämisalgoritmeina, ovat keskeisin osa mitä ohjelmistotekniikkaan tulee. Päämääräinen tarkoitus lajittelualgoritmilla on järjestää ja listata annetut alkiot ja listat etukäteen sovittuun järjestykseen, esimerkiksi luvut suuruus tai pienuus järjestykseen tai kirjaimet aakkostetusti. Olennaisin käyttötarkoitus on sovelluksissa, joiden käsittelyssä on valtavia määriä tietoa.

# Kuplalajittelu, Bubble sort

Testikohteenamme toimii kuplalajittelualgoritmi. Tämä on tutkitusti äärimmäisen, ellei jopa kaikkein hitain kaikista saatavilla olevista lajittelumenetelmistä. Sen toiminta perustuu siihen, että ensimmäisenä käydään koko lista läpi ja verrataan kahden peräkkäisen alkion arvoa keskenään. Mikäli näiden alkioiden järjestys ei ole lopullisessa toivotussa asemassa (Suuruus-, Pienuus-, aakkosjärjestyksessä), niiden paikkaa siirretään päittäin. Kun ensimmäinen vaihto on tehty, alkaa listan läpikäyminen uudelleen, kunnes saavutetaan seuraava epätosi tapaus, ja vaihto suoritetaan jälleen. Kun algoritmi huomaa, ettei yksikään merkki ole väärällä paikalla, on lajittelu suoritettu ja lista on nyt halutussa järjestyksessä.

Esimerkkinä tästä toimii lista, joka pitää sisällään luvut [3,5,2,7].

Mikäli oletetaan, että lista halutaan järjestykseen pienimmästä suurimpaan, verrataan ensin kahta ensimmäistä alkiota, ”kuplaa”, ja asetetaan ne haluttuun järjestykseen.  
[3,5,2,7]  
Ne ovat jo järjestyksessä, joten siirrytään seuraavaan parin.  
[3,5,2,7]  
Koska alkiot eivät ole järjestyksessä, niiden paikka vaihdetaan haluttuun.  
[3,2,5,7]  
Tämän jälkeen kierros alkaa jälleen alusta, ja verrataan taas kahta perättäistä alkiota.

[3,2,5,7]

Vaihto.  
[2,3,5,7]

Lista käydään vielä läpi ja todetaan, että se on nyt halutussa järjestyksessä.  
[2,3,5,7]

# Kompleksisuus

Kompleksisuudella tarkoitetaan algoritmien varsinaiseen mittaukseen käytettyä ilmaisua. Se ilmaisee sitä, kuinka nopeasti jonkin algoritmin suoritusaika kasvaa suhteessa sille selvitettäväksi annettujen alkioiden lukumäärään nähden. Algoritmien kompleksisuutta voidaan määrittää tarkasti. Sen tehokkuudelle pystytään laskemaan minimi- keskimääräinen ja maksimiarvot muun muassa seuraavien kaavojen avulla.

## Theta, θ

Theta, tai tarkemmin funktio Theta alkioiden lukumäärän suhteen **(θ(n)),** ilmaisua käytettään silloin, kun algoritmin suorittamiseen kuluva aika on äärimmäisen lähellä, ellei täysin verrannollinen annettujen alkioiden neliön suhteen.

## ”O” (Omikron)

O-merkinnän käyttö tapahtuu seuraavasti. Jollei täsmällistä ylärajaa ole mahdollista selvittää, mutta silti tälle on saatavilla jokin arvo, voidaan käyttää merkintää ”*O”*. Tässä on kysymys siis jostakin ylärajasta. Esimerkiksi funktio **O(n²)** tarkoittaa algoritmin maksimisuoritusaikaa alkioiden neliöön nähden Luku voisi myös siis olla **O(n3)**, **O(n4)** tai vaikka **O(n25)**.

## Omega, Ω

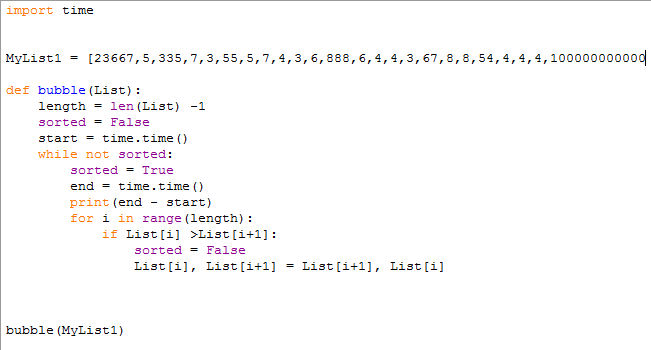
Kuten aikaisempi ”*O*” merkintä, on olemassa myös merkintätapa Oomega, ”Ω”. Tämä merkintä tarkoittaa algoritmin suoritusajan alarajaa, missä kompleksisuudeltaan Ω(n2) algoritmi on suoritusajaltaan verrannollinen vähintään sille annettujen alkioiden neliöön.

# Vertailua

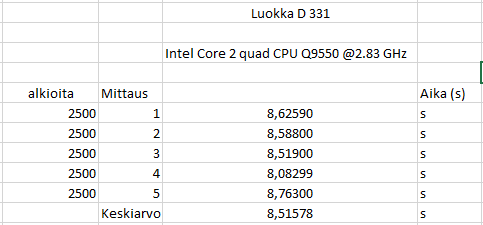
Kirjoitimme yksinkertaisen kuplalajittelualgoritmin, jonka tarkoituksena on lajitella 2500 annettua alkiota mahdollisimman nopeasti ja jonka jälkeen tulostaa suoritukseen kuluneen ajan ruudulle. Ajoimme kokeen muutamalla eri koneella, kuten JAMK:n työtilan D331 ja työskentelytilan D422 tietokoneilla. Suoritimme myös muutamia kokeita opiskelijoiden omilla kotitietokoneilla.

Kokeen suorittaminen tapahtui siten, että ajoimme koodin koneella, otimme ylös siihen kuluneen ajan ja toistimme tämän viisi kertaa, jonka jälkeen saimme selville lajitteluun käytetyn keskimääräisen ajan.

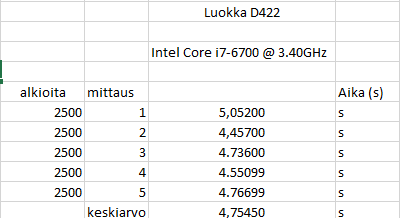
Algoritmimme koodi on ja mittaustulokset olivat seuraavan laiset:



Kuvio . Algoritmin koodi



Kuvio . Luokka D331



Kuvio . Luokka 422

# Yhteenveto

Tutkittuamme aikamme lajittelualgoritmeja ja erityisesti keskityttyämme kuplalajitteluun, havaitsimme sen olevan erittäin kömpelö tehokkuudeltaan. Kuplalajittelualgoritmi tosin havainnoi erittäin hyvin varsinkin oppivaiheessa algoritmien päämääräistä tarkoitusta ja toimintatapaa.

Algoritmit ovat yksi tietokoneiden ja ohjelmistojen kulmakivistä, varsinkin kun käsittämättömiä määriä tietoa tulee saada haluttuun muotoon. Näiden algoritmien toimitatarkoitus ja käyttötapaukset vaihtelevat tarpeen mukaan, siinä missä jokin raskaampi algoritmi kuluttaa pieneen määrään dataa yltiöpäisesti aikaa, jokin pieni yksinkertainen lajittelumenetelmä hoitaa työn ongelmitta. Tämä pätee myös päinvastoin, jossa suurten tietomäärien lajittelu vaatii jo monimutkaisempia algoritmeja joilla tieto pilkotaan pienempiin, helpommin hallittaviin osiin ennen varsinaista lajittelua.

Tietokoneen teho ja tämän suoritusnopeus on suuntaa-antava esimerkki varsinaisen työtaakan suoritusnopeudesta. Hyvänä muistisääntönä on se, mitä tehokkaampi suoritin, sitä nopeammin algoritmi suoriutuu tehtävästä

Algoritmien nopeuden määritelmästä käytetään ilmaisua kompleksisuus. Kompleksisuus on suoraan suhteessa siihen, kuinka kauan jonkin tehtävän suoritus kestää suhteessa annettujen alkioiden määrään.

# Lähteet

Logaritmi, Wikipedia 7.4.2017. <https://fi.wikipedia.org/wiki/Logaritmi>

Luento 3. Funktioiden kasvu 7.4.2017. <https://www.raippa.fi/AlgoritmitJaTietorakenteet/3.%20Luento>

Sorting algorithms, artikkeli lajittelualgoritmeista Wikipediassa 7.4.2017. <https://en.wikipedia.org/wiki/Sorting_algorithm>

Tietokoneohjelmien nopeus ja kompleksisuuden käsite 7.4.2017. <http://people.uta.fi/~jm58660/jutut/ohjelmointi/kompleksisuus.html>