

COMP.CS.300 Tietorakenteet ja algoritmit 1

Lähteet

Luentomoniste pohjautuu vahvasti prof. Antti Valmarin vanhaan luentomonisteeseen Tietorakenteet ja algoritmit.

Useimmat algoritmit ovat peräisin kirjasta Introduction to Algorithms; Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein.

Lisäksi luentomonistetta koottaessa on käytetty seuraavia kirjoja:

- Introduction to The Design & Analysis of Algorithms; Anany Levitin
- Olioiden ohjelmointi C++:lla; Matti Rintala, Jyke Jokinen
- Tietorakenteet ja Algoritmit; Ilkka Kokkarinen, Kirsti Ala-Mutka
- The C++ Standard Library; Nicolai M. Josuttis

1 Johdanto

Mietitään ensin hiukan syitä tietorakenteiden ja algoritmien opiskelulle

Algorithms in the world

1.1 Miksi?

Mitkä ovat sinun elämääsi eniten vaikuttavat algoritmit?

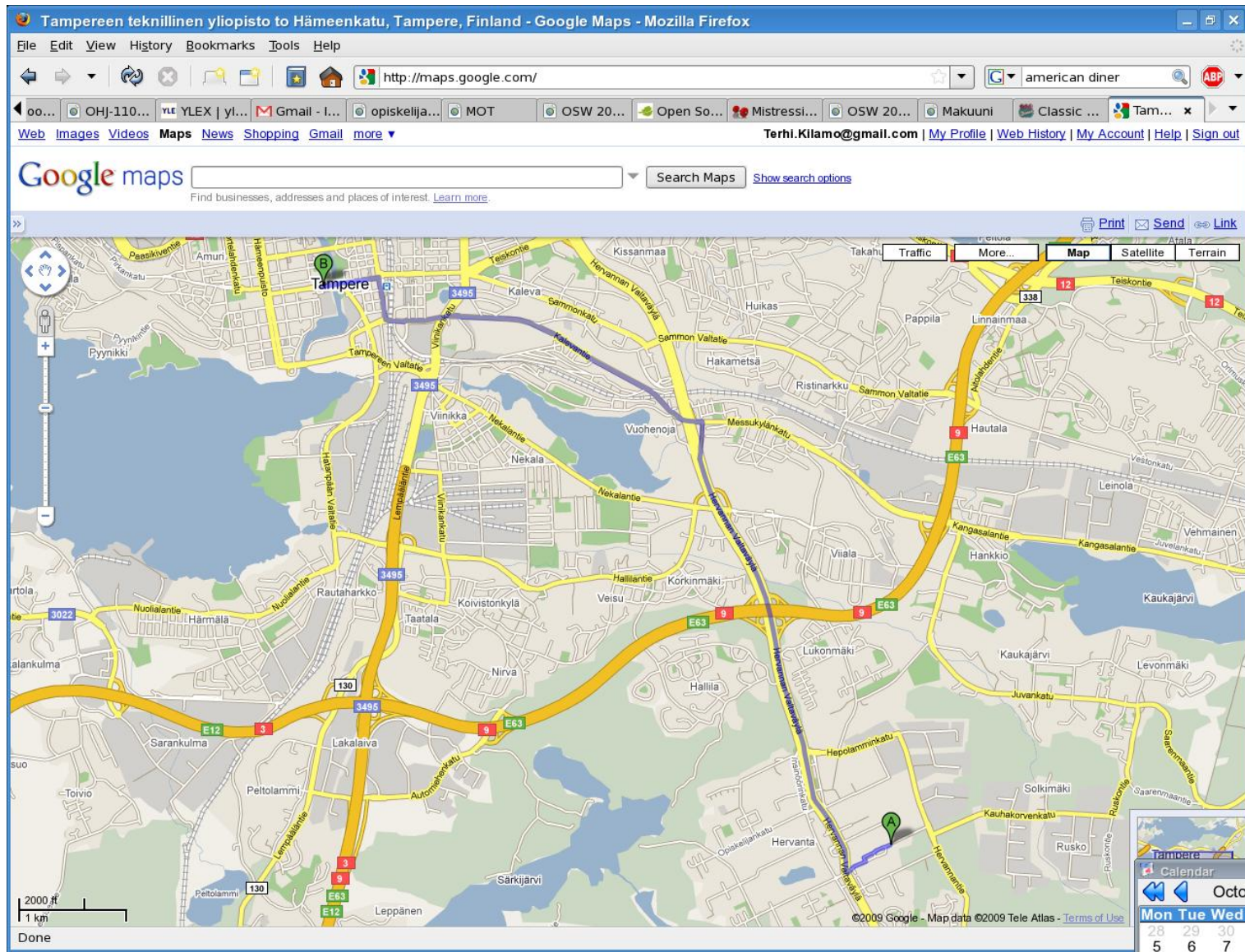


Picture: Chris Watt

Tietokoneohjelmia ei ole olemassa ilman algoritmeja

- algoritmeihin törmää esimerkiksi seuraavissa sovelluksissa:





momondo

flights hotel car rental holiday rentals

Compare cheap flights and hotels

Tell us where you're going and we find the best prices on flights and hotels. Enjoy your trip! Psst ... we're a free service, not a travel agency and we don't add any booking fees.

We come recommended by **CNN TRAVEL+LEISURE**

Flights ✈️ **Hotel** 🏨

☐ One-way ☒ Return Trip ☐ Multiple destinations

From: Helsinki (HEL), Finland

Departure date: 03/20/2014

To: Madrid (MAD), Spain

Return date: 03/24/2014

Adults: 1 Children: 0 Ticket Class: Economy

SELECT YOUR END DATE

FEBRUARY 2014

Wk	Su	Mo	Tu	We	Th	Fr	Sa
5							1
6	2	3	4	5	6	7	8
7	9	10	11	12	13	14	15
8	16	17	18	19	20	21	22
9	23	24	25	26	27	28	
10							

MARCH 2014

Wk	Su	Mo	Tu
9			
10	2	3	4
11	9	10	11
12	16	17	18
13	23	24	25
14	30	31	

aina, kun käytät tietokonetta, käytät myös algoritmeja.

1.2 Kaikki pennillä?



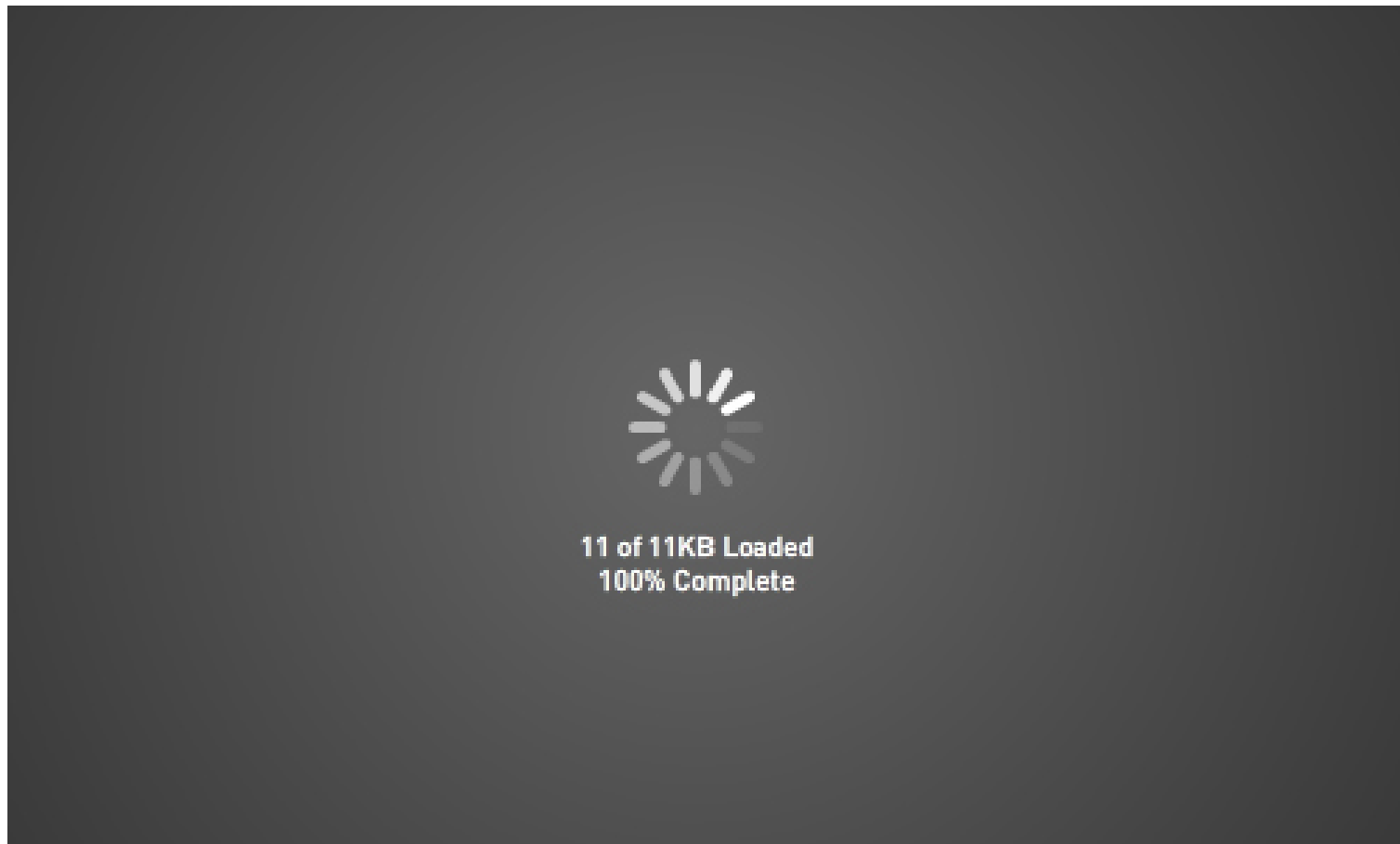
Due to an outside glitch, several British merchants on Amazon found their good were selling for just a penny last week. (Stephen Hilger/Bloomberg News)

Via: The Washington Post

Tietorakenteita tarvitaan ohjelmissa käsiteltävän tiedon tallettamiseen ja sen käsittelyn mahdollistamiseen ja helpottamiseen

- tietorakenteita on monia eivätkä ne kaikki sovi kaikkiin tilanteisiin
 - ⇒ ohjelmoijan pitää osata valita tilanteeseen sopivin
 - ⇒ vaihtoehtojen käyttäytyminen, vahvuudet ja heikkoudet on tunnettava

Modernit ohjelmointikielet tarjoavat valmiina helppokäyttöisiä tietorakenteita. Näiden ominaisuuksien sekä käyttöön vaikuttavien rajoitteiden tuntemiseksi tarvitaan perustietorakenneosaamista



Kuinka moni on turhautunut ohjelman tai esimerkiksi kännykän hitauteen?

- toiminnallisuus on toki ensisijaisen tärkeää kaikille ohjelmille, mutta tehokkuus ei ole merkityksetön sivuseikka
- on tärkeää huomioida ja miettiä ratkaisujaan myös ajan- ja muistinkäytön kannalta
- valmiin kirjaston käyttö näyttää usein suoraviivaisemmalta kuin onkaan

Näitä asioita käsitellään tällä kurssilla

2 Käsitteitä ja merkintöjä

Tässä luvussa esitellään kurssilla käytettävää käsitteistöä ja merkintätapoja.

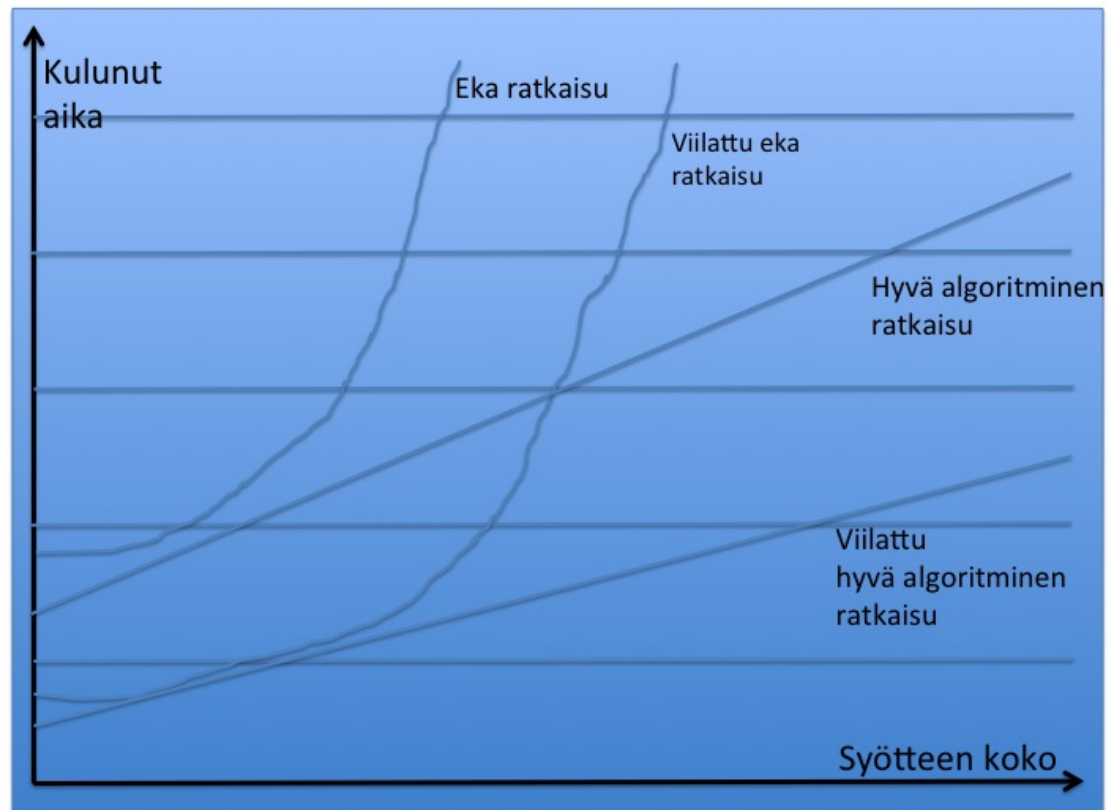
Luvussa käsitellään myös pseudokoodiesityksen ja ohjelmointikielisen koodin eroja. Esimerkkinä käytetään järjestämisalgoritmia INSERTION-SORT.

2.1 Tavoitteet

Kurssin keskeisenä tavoitteena on antaa opiskelijalle käyttöön peruskoneisto kuhunkin ohjelmointitehtävään sopivan ratkaisun valitsemiseen ja omien ratkaisujen tehokkuuden arvioimiseen karkealla tasolla.

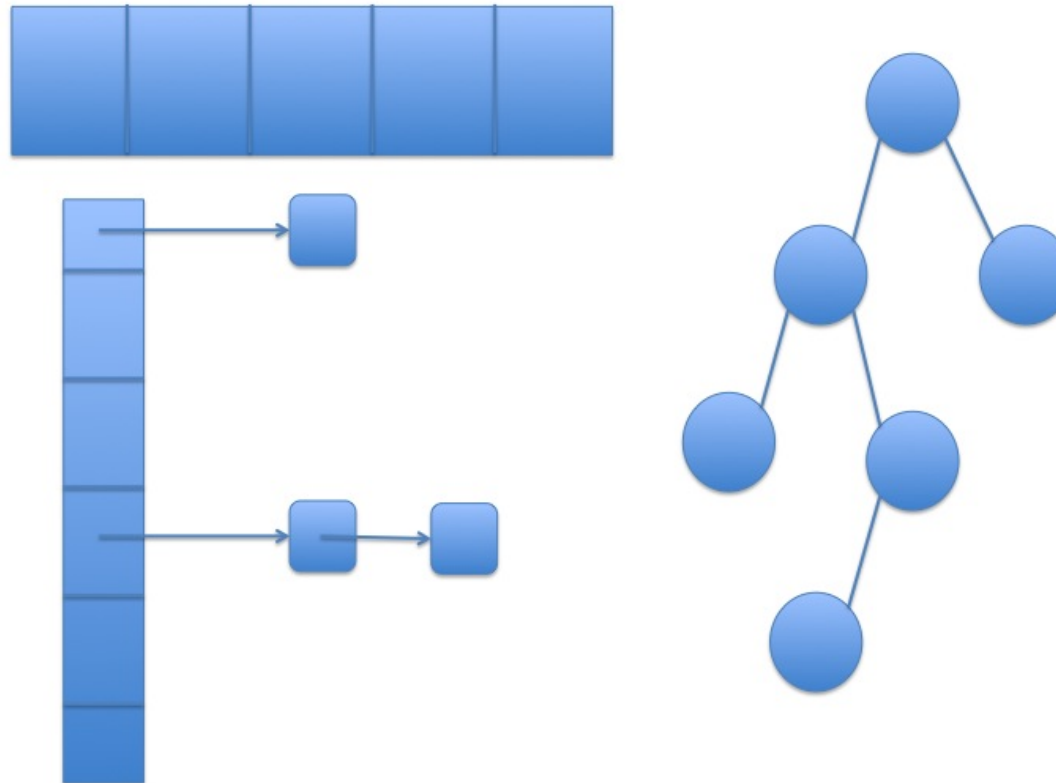
- Kurssilla keskitytään tilanteeseen sopivan tietorakenteen valintaan.
- Lisäksi käsitellään käytännön tilanteissa usein vastaan tulevia ongelmatyyppejä ja algoritmeja, joilla ne voi ratkaista.

- Kurssilla keskitytään lähinnä ns. hyviin algoritmeihin.
- Painotus on siis siinä, miten algoritmin ajankulutus kasvaa syötekoon kasvaessa, eikä niinkään yksityiskohtien optimoinnissa.



2.2 Peruskäsitteistöä

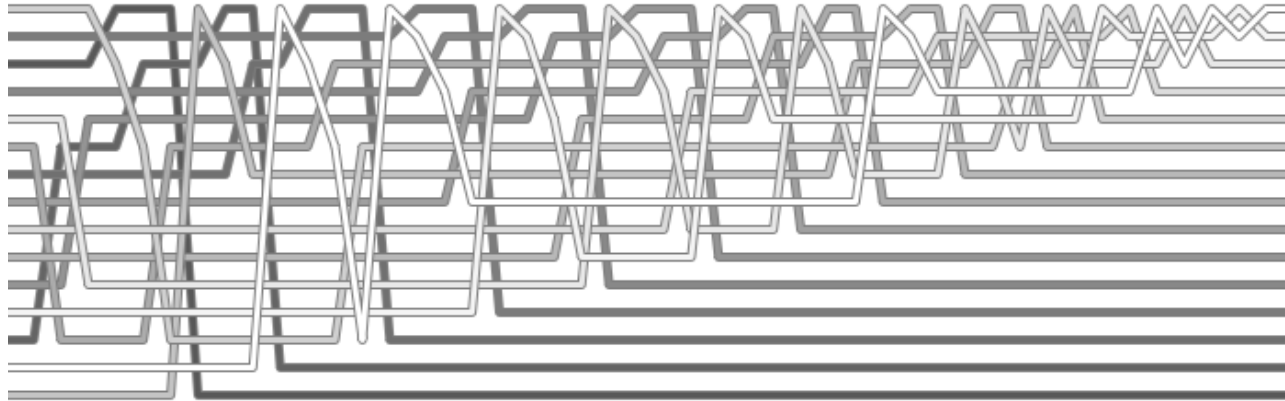
Tietorakenne



Tapa tallettaa ja järjestää tietoa:

- tietoa pystytään lisäämään ja hakemaan algoritmien avulla.
- talletettua tietoa voidaan muokata
- tietorakenteita on useamman tasoisia: tietorakenne voi koostua toisista tietorakenteista

Algoritmi:



Kuva 1: kuva: Aldo Cortesi

- Joukko ohjeita tai askeleita jonkin ongelman ratkaisemiseksi
- Hyvin määritelty laskentamenetelmä, joka saa **syötteenään** alkion tai joukon alkioita ja tuottaa **tuloksenaan** alkion tai joukon alkioita
- Tapa tuottaa syöttestä tulos

Flight	From	Time	Duration	To	Class	Price
1	HEL	14:00	35min	TKU	Economy	328 EUR
2	TKU	22:00	35min	HEL	Economy	328 EUR

328 EUR

[Siirry sivulle](#)

- hyvin määritelty =
 - jokainen askel on kuvattu niin tarkasti, että lukija (ihminen tai kone) osaa suorittaa sen
 - jokainen askel on määritelty yksikäsitteisesti
 - samat vaatimukset pätevät askelten suoritussjärjestykselle
 - suorituksen tulee päättyä äärellisen askelmäärän jälkeen






Algoritmi ratkaisee jonkin hyvin määritellyn (laskenta)tehtävän.






- laskentatehtävä määrittelee, missä suhteessa tulosten tulee olla annettuihin syötteisiin
- esimerkiksi:
 - taulukon järjestäminen
 - syötteet:** jono lukuja a_1, a_2, \dots, a_n
 - tulokset:** luvut a_1, a_2, \dots, a_n suuruusjärjestyksessä pienin ensin
 - lentoyhteyksien etsiminen
 - syötteet:** lentoreittiverkosto eli kaupunkeja joiden välillä lentoyhteyksiä
 - tulokset:** Lentojen numerot, yhteyden tiedot ja hinta.

- laskentatehtävän *esiintymä* eli *instanssi* saadaan antamalla tehtävän syötteille lailliset arvot
 - järjestämistehtävän instanssiesimerkki: 31, 41, 59, 26, 41, 58

Algoritmi on *oikea* (*correct*), jos se pysähtyy ja antaa oikeat tulokset aina kun sille on annettu laillinen syöte.

- algoritmin tai laskentatehtävän määritelmä saa kieltää osan muodollisesti mahdollisista syötteistä

	HEL 13:55 Helsinki	5t55m 2 VAIHT.	19:50 TKU Turku	Economy	
	TKU 11:00 Turku	23t30m 1 PYS.	10:30 HEL Helsinki	 1	2 590 EUR
 Lennon tiedot				Eticket.fi 2 590 EUR	
Siirry sivulle					

	HEL 10:25 Helsinki	6t10m 2 VAIHT.	16:35 TKU Turku	Economy	
	TKU 11:00 Turku	23t30m 1 PYS.	10:30 HEL Helsinki	 1	2 590 EUR
 Lennon tiedot				Eticket.fi 2 590 EUR	
Siirry sivulle					

algoritmi voi olla virheellinen kolmella tavalla

- antaa väärän lopputuloksen
- kaatuu kesken suorituksen
- ei koskaan lopeta

virheellisenkin algoritmi on joskus hyvin käyttökelpoinen, jos virhetiheys hallitaan!

- esim. luvun testaus alkuluvuksi

Periaatteessa mikä tahansa menetelmä kelpaa algoritmien esittämiseen, kunhan tulos on tarkka ja yksikäsitteinen.

- yleensä algoritmit toteutetaan tietokoneohjelmina tai laitteistoina
 - käytännön toteutuksessa on otettava huomioon monia insinöörinäkökohtia
 - sopeuttaminen käyttötilanteeseen
 - syötteiden laillisuuden tarkistukset
 - virhetilanteiden käsittely
 - ohjelmointikielen rajoitukset
 - laitteiston ja kielen aiheuttamat nopeus- ja tarkoituksenmukaisuusnäkökohdat
 - ylläpidettävyys \Rightarrow modulaarisuus jne.
- \Rightarrow algoritmin idea hukkuu helposti toteutusyksityiskohtien alle

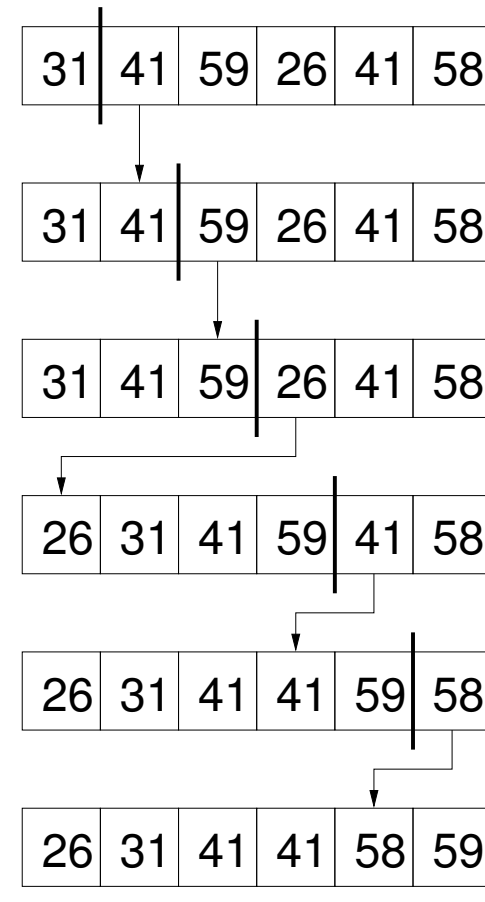
Tällä kurssilla keskitytään algoritmien ideoihin ja algoritmit esitetään useimmiten pseudokoodina ilman laillisuustarkistuksia, virheiden käsittelyä yms.

Otetaan esimerkiksi pienten taulukoiden järjestämiseen soveltuva algoritmi INSERTION-SORT:



Kuva 2: kuva: Wikipedia

- periaate:
 - toiminnan aikana taulukon alkuosa on järjestyksessä ja loppuosa ei
 - osien raja lähtee liikkeelle paikkojen 1 ja 2 välistä ja etenee askel kerrallaan taulukon loppuun
- kullakin siirtoaskeleella etsitään taulukon alkuosasta kohta, johon loppuosan ensimmäinen alkio kuuluu
 - uudelle alkiolle raivataan tilaa siirtämällä isompia alkioita askel eteenpäin
 - lopuksi alkio sijoitetaan paikalleen ja alkuosaa kasvatetaan pykälällä



Kurssilla käytetyllä pseudokoodiesityksellä INSERTION-SORT näyttää tältä:

INSERTION-SORT(A)	<i>(syöte saadaan taulukossa A)</i>
1 for $j := 2$ to $A.length$ do	<i>(siirretään osien välistä rajaa)</i>
2 $key := A[j]$	<i>(otetaan alkuosan uusi alkio käsittelyyn)</i>
3 $i := j - 1$	
4 while $i > 0$ and $A[i] > key$ do	<i>(etsitään uudelle alkiolle oikea paikka)</i>
5 $A[i + 1] := A[i]$	<i>(raivataan uudelle alkiolle tilaa)</i>
6 $i := i - 1$	
7 $A[i + 1] := key$	<i>(asetetaan uusi alkio oikealle paikalleen)</i>

- **for**- yms. rakenteellisten lauseiden rajausta osoitetaan sisennyksillä
- *(kommentit)* kirjoitetaan sulkuihin kursiivilla
- sijoitusoperaattorina on “:=” (“=” on yhtäsuuruuden vertaaminen)
- ▷ -merkillä varustettu rivi antaa ohjeet vapaamuotoisesti

- tietueen (tai olion) kenttiä osoitetaan pisteen avulla
 - esim. *opiskelija.nimi*, *opiskelija.numero*
- osoittimen x osoittaman tietueen kenttiä osoitetaan merkin \rightarrow avulla
 - esim. $x \rightarrow \textit{nimi}$, $x \rightarrow \textit{numero}$
- ellei toisin sanota, kaikki muuttujat ovat paikallisia
- taulukoilla ja / tai osoittimilla kootun kokonaisuuden nimi tarkoittaa **viitettä** ko. kokonaisuuteen
 - tuollaiset isommat tietorakenteethan aina käytännössä kannattaa välittää viiteparametreina
- yksittäisten muuttujien osalta aliohjelmat käyttävät arvonvälitystä (kuten C++-ohjelmatkin oletuksena)
- osoitin tai viite voi kohdistua myös ei minnekään: NIL

2.3 Algoritmien toteutuksesta

Käytännön toteutuksissa teoriaa tulee osata soveltaa.

Esimerkki: järjestämisalgoritmin sopeuttaminen käyttötilanteeseen.

- harvoin järjestetään pelkkiä lukuja; yleensä järjestetään tietueita, joissa on
 - *avain (key)*
 - *oheisdataa (satellite data)*
- avain määrää järjestyksen
 - ⇒ sitä käytetään vertailuissa
- oheisdataa ei käytetä vertailuissa, mutta sitä on siirreltävä samalla kuin avaintakin

Edellisessä kappaleessa esitelty INSERTION-SORTissa muuttuisi seuraavalla tavalla, jos siihen lisättäisi oheisdata:

```
1  for  $j := 2$  to  $A.length$  do  
2       $temp := A[j]$   
3       $i := j - 1$   
4      while  $i > 0$  and  $A[i].key > temp.key$  do  
5           $A[i + 1] := A[i]$   
6           $i := i - 1$   
7       $A[i + 1] := temp$ 
```

- jos oheisdataa on paljon, kannattaa järjestää taulukollinen osoittimia tietueisiin ja siirtää lopuksi tietueet suoraan paikoilleen

Jotta tulokseksi saataisi ajokelpoinen ohjelma, joka toteuttaa INSERTION-SORT:n tarvitaan vielä paljon enemmän.

- täytyy ottaa käyttöön oikea ohjelmointikieli muuttujien määrittelyineen ja aliohjelmineen
- tarvitaan pääohjelma, joka hoitaa syötteenluvun ja sen laillisuuden tutkimisen ja vastauksen tulostamisen
 - on tavallista, että pääohjelma on selvästi algoritmia pidempi

Ohjelmointikieli määrää usein myös muita asioita, esim:

- Indeksointi alkaa 0:sta (pseudokoodissa usein 1:stä)
- Käytetäänkö edes indeksointia (tai taulukoita, tai...)
- (C++) Onko data oikeasti tietorakenteen sisässä, vai osoittimen päässä (jolloin dataa ei tarvitse siirtää ja sen jakaminen on helpompaa)
- Jos dataan viitataan epäsuorasti muualta, tapahtuuko se
 - Osoittimella
 - Älyosoittimella (esim. `shared_ptr`)
 - Iteraattorilla (jos data tietorakenteessa)
 - Indeksillä (jos data vektorissa tms.)
 - Hakuavaimella (jos data tietorakenteessa, josta haku nopeaa)
- Toteutetaanko rekursio iteroinnilla vai ei (riippuu myös ongelmasta)
- Ovatko algoritmin "parametrit" oikeasti parametreja, vai vain muuttujia tms.

Otetaan esimerkiksi edellä kuvatun ohjelman toteutus C++:lla:

```
#include <iostream>
#include <vector>
typedef std::vector<int> Taulukko;

void insertionSort( Taulukko & A ) {
    int key = 0; int i = 0;
    for( Taulukko::size_type j = 1; j < A.size(); ++j ) {
        key = A.at(j); i = j-1;
        while( i >= 0 && A.at(i) > key ) {
            A.at(i+1) = A.at(i); --i;
        }
        A.at(i+1) = key;
    }
}

int main() {
    unsigned int i;
    // haetaan järjestettävien määrä
    std::cout << "Anna taulukon koko 0...: "; std::cin >> i;
```

```
Taulukko A(i); // luodaan taulukko
// luetaan järjestettävät
for( i = 0; i < A.size(); ++i ) {
    std::cout << "Anna A[" << i+1 << "]: ";
    std::cin >> A.at(i);
}
insertionSort( A );    // järjestetään

// tulostetaan siististi
for( i = 0; i < A.size(); ++i ) {
    if( i % 5 == 0 ) {
        std::cout << std::endl;
    }
    else {
        std::cout << " ";
    }
    std::cout << A.at(i);
}
std::cout << std::endl;
}
```

Ohjelmakoodi on huomattavasti pseudokoodia pidempi ja algoritmille ominaisten asioiden hahmottaminen on siitä paljon vaikeampaa.

Tämä kurssi keskittyy algoritmien ja tietorakenteiden periaatteisiin, joten ohjelmakoodi ei palvele tarkoituksiamme.

⇒ Tästä eteenpäin toteutuksia ohjelmointikielillä ei juurikaan esitetä.