
Tietorakenteet ja algoritmit

Algoritmiteorian perusajatuksia

Ari Korhonen

3. ALGORITMITEORIAN PERUSAJATUKSIA

[3.1 Tietorakenteiden käsitteellistä luokittelua](#)

[3.2 Talletusrakenteet](#)

[3.3 Abstraktit tietotyypit](#)

[3.4 Algoritmit suhteessa tietorakenteisiin](#)

[3.4 Algoritmien vertailu](#)

[3.5 Algoritmien toteutus ja esittäminen](#)

3.1 Tietorakenteiden käsitteellistä luokittelua

Luokitteluun on erilaisia lähestymistapoja:

1) Mitkä seuraavista ovat tietorakenteita?

- Bitti
- Boolean
- Integer
- Taulukko
- Tuple
- Double
- Verkko

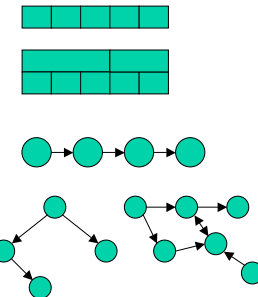
Atominen tieto vs. rakenteinen tieto

- Riippuu abstraktiotasosta

Staattinen vs. dynaaminen tietorakenne


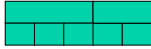
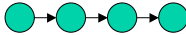
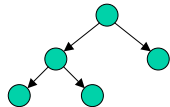
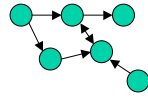
2) Mikä seuraavista on staattinen, mikä dynaaminen tietorakenne?

- Taulukko
- Tietue
- Dynaaminen taulukko
- Linkitetty lista
- Verkko
- Binääripuu



3) Alkion saavutettavuus:

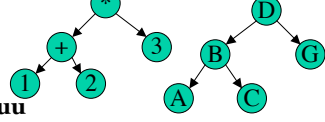
Mitä eroa?

- Taulukko 
- Tietue 
- Linkitetty lista 
- Binääripuu 
- Verkko 

Alkion paikka voidaan laskea vakioajassa tai sitten ei.

4) Abstraktiotaso

Mitä eroa?

- Binääripuu 
- Jäsennyspuu
- Binäärinen hakupuu
- Taulukko
- Hajautustaulukko

Talletusrakenne vai semanttinen rakenne:

- Rakenne voi olla puhdas *talletusrakenne* tai *abstrakti tietotyyppi*, jolla on jokin semantiikka

3.2 Talletusrakenteet (fundamental data type, FDT)

- Tietorakenteita, jotka määrittelevät vain tietoalkioiden sijainnin suhteessa toisiinsa (topologia)
- Eivät ota kantaa tietosisältöön tai sen merkitykseen
- Esimerkiksi:
 - Taulukko
 - Linkitetty lista
 - Binääripuu
 - Yleinen puu
 - Isälinkitetty puu
 - Verkko

3.3 Abstraktit tietotyypit (abstract data type, ADT)

- Kun talletetulle tiedolle annetaan semanttinen merkitys (mitä tieto tarkoittaa) ja määritellään, mitä tiedolle saa tehdä, saadaan abstrakti tietotyyppi.
- Määritellään tietorakenteeseen kohdistuvat operaatiot
- Ei oteta kantaa, miten operaatiot toteutetaan
- Otetaan kantaa, mitä operaatiot tekevät (riittävä tarkkuustaso!)
- Mahdollistaa toteutuksen muuttamisen joustavasti
- Sama abstrakti tietotyyppi voidaan toteuttaa usean eri talletusrakenteen avulla.

Abstraktit tietotyypit / Esimerkkejä

A) Joukko abstraktina tietotyyppinä

▪ Määritellään vaikkapa operaatiot:

- \cap - Leikkaus; Set intersection(Set A, Set B)
- \cup - Unioni; Set union(Set A, Set B)
- $\bar{}$ - Komplementti; Set compl(Set A, Set superSet)
- $||$ - Koko (= alkioiden määrä); int size(Set A)
- \in - Kuuluu joukkoon?; boolean in(Element E, Set A)

▪ Operaatiot toteutetaan aliohjelmina

- Muualla ohjelmassa voidaan kutsua y.o. operaatioita tarvitsematta tietää, miten alkiot on talletettu (taulukko, lista, puu, ...)

27.1.2004

9

B) Hakurakenne abstraktina tietotyyppinä

- Määritellään, että hakurakenne on kokoelma *tietueita*, jotka voidaan yksiselitteisesti tunnistaa niihin talletetun *hakuavaimen* arvon perusteella

▪ Määritellään esimerkiksi operaatiot

- Element Search(Key K)
- Element Insert(Key K, Data D)
- Element Delete(Key K)
- Element Next(Key K) [Previous]
 - Etsi annettua tietuetta lähinnä seuraava [edellinen] tietue
- Element[] Range(Key From, Key To)
 - Etsi tietueet, joiden avainarvot ovat avainarvojen [From,To] välissä

27.1.2004

10

C) Prioriteettijono abstraktina tietotyyppinä

- Määritellään, että rakenne on kokoelma alkioita, joihin jokaiseen liittyy alkion suhdetta muihin alkioihin kuvaava *prioriteetti*

▪ Määritellään esimerkiksi operaatiot

- Insert(Element E, int P)
 - Lisää alkio ja aseta sen prioriteetiksi P
- Element DeleteMax()
 - Poista ja palauta alkio, jonka prioriteetti-arvo on suurin (tai deleteMin - pienin)
- DecreaseKey(Element E, int delta)
 - pienennä (tai IncreaseKey -suurena) alkion prioriteettiä deltalla

27.1.2004

11

D) Ohjelmointikielet ja abstraktit tietotyypit

- Esim. C-kielen standardityypit on toteutettu ADT:nä:

- int (16-32 bit)
- float, double (IEEE, ..)
- char (8 bit)
- Tiedostojen käsittely, FILE *

27.1.2004

12

- **Standardi-C ei sisällä mekanisme, jolla voitaisiin määritellä abstrakteja tyyppjä**
- **Ohjelmoija voi tosin noudattaa samaa periaatetta:**
 - Määritellään aliohjelmat, joilla käsitellään tietyn tyyppisiä objekteja
 - Ei käytetä tyyppin rakennetta näiden aliohjelmien ulkopuolella
 - Käsiteltävä objekti välitetään aliohjelmille muuttujaparametrina
 - Käsitellyaliohjelmat kootaan yhteen paikkaan ja nimetään loogisesti

⇒ **VASTUU EHEYDEN NOUDATTAMISESTA JÄÄ OHJELMOIJALLE**
- **Oliokielissä abstraktit tietotyypit määritellään yleensä rajapintoina ja toteutetaan luokkina (Java, C++)**

27.1.2004

13

3.4 Algoritmit suhteessa tietorakenteisiin

- **Useimmat algoritmit käsittelevät jotain tietorakennetta, eikä atomista dataa**
- **Käsiteltävä data on siten joukko data-alkioita, jotka muodostavat jonkin kokonaisuuden. Algoritmi käsittelee tätä kokonaisuutta eli tietorakennetta. Esimerkiksi:**
 - Lajitellaan tieto suuruusjärjestykseen
 - Etsitään jotain yksittäistä tietoa suuresta tietojoukosta
 - Käsitellään rakennetta, jossa tietojoukon yksittäisten alkoiden välillä voi olla yhteyksiä, jotka yksilöivät alkion suhteen muihin alkioihin (verkot)
 - Käsitellään koordinaattiavaruudessa olevaa kokonaisuutta

27.1.2004

14

3.5 Algoritmien vertailu

- **Jos sama asia voidaan toteuttaa eri tavoin, täytyy jotenkin pystyä arvioimaan, mikä toteutus on kulloinkin hyvä**
- **Vertailukriteereitä on monia:**
 1. **Algoritmin askelmäärä suhteessa käsiteltävän tiedon määrään**
 - ✓ Pahin mahdollinen askelmäärä
 - ✓ Keskimääräinen askelmäärä
 - ✓ Tasattu askelmäärä

=> **ALGORITMIANALYYSI**

27.1.2004

15

Algoritmien vertailu / kriteereitä

- **Algoritmin kuluttama todellinen cpu-aika (usein ongelmallinen arvioitava)**
- **Algoritmin/tietorakenteen vaatima muistitila suhteessa käsiteltävän tiedon määrään**
- **Algoritmin toteutuksen monimutkaisuus (kannattaako optimointi?)**

27.1.2004

16

3.6 Algoritmien toteutus ja esittäminen

- **Ohjelman ylläpidettävyyden kannalta on ehdottomasti järkevää toteuttaa tärkeä algoritmi siten, että sen mahdollinen vaihtaminen tehokkaampaan vaikuttaa mahdollisimman vähän muuhun ohjelmaan**
 - ⇒ Käytetään abstrakteja tietotyyppejä
- **Usein algoritmit esitetään pseudokielellä (á la Pascal) ja jätetään toteutuksen pienimmät yksityiskohdat avoimiksi**
 - ⇒ Algoritmin perusajatusta on helpompi ymmärtää, kun ei tarvitse kiinnittää huomiota ohjelmointikielen syntaksiin
 - ⇒ Kuvaus on kieliriippumaton