

Johdanto

Ari Korhonen

1. JOHDANTO

- 1.1 Määritelmiä
- 1.2 Tietorakenteen ja algoritmin valinta
- 1.3 Algoritmit ja tiedon määrä
- 1.4 Tietorakenteet ja toiminnot
- 1.5 Esimerkki: lineaarinen vs. eksponentiaalinen toteutus

1.1 Määritelmiä

Algoritmi:

- **Epäformaalisti:** täsmällinen menettelytapa, joka ratkaisee jonkin hyvin määritellyn laskennallisen ongelman.

Määritelmä:

Algoritmi on äärellinen jono yksikäsitteisiä, äärellisellä työllä suoritettavissa olevia käskyjä, jotka laskevat funktion $f: I \rightarrow O$

- I on syötejoukko
- O on tulosjoukko
- kaikilla $i \in I$, algoritmi pysähtyy s.e., $o = f(i) \in O$

1.1 Määritelmiä

Tietorakenne:

- Funktion f tehokas laskenta edellyttää, että syötejoukko I on hyvin organisoitu
- Päteekö sama tulosjoukolle O ? (Vrt. $g(f(n))$)
- Voidaan kuvata tietotyyppinä
- Alkeistyyppit, perustietotyyppit, abstraktit tietotyyppit
- Määritelmä:

Tietorakenne on joukko muuttujia, joiden keskinäiset suhteet on määritelty täsmällisesti.

- Huomaa, että em. määrittelyssä yksittäisen muuttujan tyyppi voi olla toinen tietorakenne.

1.2 Tietorakenteen ja algoritmin valinta

- Sama asia tai toiminto voidaan toteuttaa lukemattoman monella eri tavalla tietokoneohjelmassa
- Eri toteutusten paremmuutta voidaan verrata monessa suhteessa, mm. :
 - toimintojen nopeus (algoritmit)
 - vaadittu muistitila (tietorakenteet)
 - ohjelmakoodin laatu ja ylläpidettävyys
 - toteutuksen avoimuus laajennuksille
- Tällä kurssilla keskitytään kahteen ensimmäiseen näkökulmaan

ÄLÄ OPTIMOI KOODIA - VALITSE PAREMPI ALGORITMI

- Usein valitsemalla sopiva tiedon esitysrakenne voidaan merkittävästi vaikuttaa jonkin algoritmin tehokkuuteen

Esimerkki 1: Haku linkitetystä listasta tai haku tasapainotetusta binäärihakupuusta

- Edellisessä hakuaika on suoraan suhteessa talletettujen alkioden lukumäärään
- Jälkimmäisessä hakuaika on suhteessa talletettujen alkioden lukumäärän *logaritmiin*

N	$\log_2 N$
1000	10
1000000	20

Esimerkki 2: Miten tallentaa maantieverkko?

- Seuraajalistana, eli luetellaan jokaiselle solmulle ne solmut, joihin kyseisestä solmusta pääsee
- Vierusmatriisina (2-ulotteisena taulukkona), jolloin jokaiselle solmuparille merkitään, onko niiden välillä yhteyttä (1) vai ei (0)
- Seuraajalista vaatii tilaa sen verran kuin yhteyksiä on olemassa
- Vierusmatriisi vaatii tilaa suhteessa solmujen lukumäärän neliöön riippumatta yhteyksien määrästä
- Vierusmatriisi voi olla tehokkaampi, jos verkko on tiheä

1.3 Algoritmit ja tiedon määrä

Algoritmin nopeus riippuu tiedon määrästä:

Esimerkki 3: Taulukon alkioden järjestäminen

- Valintalajittelu on hyvin helppo ymmärtää ja toteuttaa
- Quicksort on paljon mutkikkaampi
- Valintalajittelun tehokkuus on suhteessa alkioden lukumäärän N neliöön
- Quicksortin tehokkuus vastaavasti lukuun $N \log N$

N	$N * \log_2(N)$	N^2
10	30	100
10^3	$1 * 10^4$	10^6
10^6	$2 * 10^7$	10^{12}

Esimerkki 4: Alkioiden (N kpl) talletus listaan

- Jos kukin alkio talletetaan listan alkuun, eikä listaa pidetä järjestyksessä, alkioiden talletus käy lineaarisessa ajassa (yksittäisen alkion talletus vie vakioajan)
- Jos kukin alkio talletetaan listan loppuun ja paikka haetaan aina listan alusta lähtien, alkioiden talletus käy ajassa, joka on suhteessa listan pituuden neliöön
- Jos kukin alkio talletetaan listan loppuun ja samalla ylläpidetään osoitinta listan viimeiseen alkioon, alkioiden talletus käy taas lineaarisessa ajassa

18.1.2005

9

1.4 Tietorakenteet ja toiminnot

Tietorakenteen tehokkuus riippuu siihen kohdistuvista toiminnoista (oikea abstraktio, frekvenssit, jne.)

Esimerkki 5: Haetaan tiettyä tietuetta suuresta määrästä tietueita

- Jos tiedot ovat staattisia, ne voidaan tallentaa taulukkoon suuruusjärjestyksessä ja hakea sieltä vaikka binääri- eli puolitushaulla
- Haku-aika on silloin suhteessa taulukon koon logaritmiin
- Jos lisätään uusia tietueita tai poistetaan vanhoja, taulukko pitää järjestää kokonaan uudelleen. Tässä vaiheessa aika kuluu lineaarisesti suhteessa taulukon kokoon.
- Jos tiedot talletetaan tasapainotettuun binääriiseen hakupuuhun, tieto voidaan edelleen hakea logaritmisessa ajassa, mutta myös lisäykset ja poistot voidaan tehdä logaritmisessa ajassa

18.1.2005

10

1.5 Esimerkki: lineaarinen vs. eksponentiaalinen toteutus

Esimerkki 6: Fibonacci-luvut: 1, 1, 2, 3, 5, 8, 13, 21, 34,...

$$F(0) = F(1) = 1$$

$$F(N) = F(N-1) + F(N-2)$$

- Lineaarinen menetelmä, käytetään muuttujia x , y ja z ja pyöritetään niitä seuraavassa luopissa

$$z = y + x;$$

$$x = y;$$

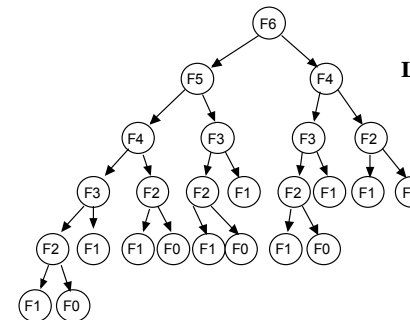
$$y = z$$

- suoritus-aika verrannollinen N :ään

18.1.2005

11

- Exponentiaalinen menetelmä, sovelletaan suoraan määritelmää rekursiivisesti



Laskettaessa F_6 :n arvoa, lasketaan:

F_5	1 kertaa
F_4	2 kertaa
F_3	3 kertaa
F_2	5 kertaa
F_1	8 kertaa

18.1.2005

12

- **Exponentiaalinen menetelmä, sovelletaan suoraan määritelmää rekursiivisesti....**

Koska pätee, että $F_{100} \approx F^{100}$, missä $F \approx 1.618$, seuraa, että F_{100} :n laskemiseen tarvitaan luokkaa 10^{18} operaatiota...

Eli algoritmien valinnalla on merkitystä...

Ensi kerraksi...

- **Muista ilmoittautua kurssille (Webtopissa)!**
- **Muista valita laskuharjoitusryhmä (vain T-kurssi)**
- **Tutustu oppikirjan/oppimateriaalin avulla käsitteisiin**
 - Rekursio
 - Abstrakti tietotyyppi