

Kuinka saada ASA-kurssin tenttivastaus näyttämään järkevältä?

Pekka Kilpeläinen
Tietojenkäsittelytieteen laitos, Kuopion yliopisto

11. kesäkuuta 2003

Algoritmien suunnittelu ja analysointi (ASA) -kurssin tenttipapereissa näkee monenlaisia vastauksia. Alla muutamia vinkkejä pahimpien rیمانalitusten välttämiseksi. Seuraavanlainen tehtävätyyppi esiintyy usein ASA-tentissä¹:

Esitä algoritmi, joka laskee syötteenä annetusta kokonaisluvusta n arvon $\sum_{i=1}^n 2i$. Arvioi algoritmin aikakompleksisuus.

Pyydetyn algoritmin olisi siis palautettava seuraavantapaisia arvoja:

$$\begin{aligned}n = 1 &\longrightarrow 2 \cdot 1 = 2 \\n = 2 &\longrightarrow 2 \cdot 1 + 2 \cdot 2 = 6 \\n = 3 &\longrightarrow 2 \cdot 1 + 2 \cdot 2 + 2 \cdot 3 = 12\end{aligned}$$

Yleisesti palautettava arvo on tietenkin

$$2 \cdot \sum_{i=1}^n i = n(n+1) \tag{1}$$

Huom: Ylläolevan kaltaiset laskelmat eivät vielä esitä tehtävässä pyydettyä algoritmia. Algoritmi on yleinen jonkin laskentaongelman ratkaisumenetelmä. Soveltuvien esitystapa on usein kurssillakin käytetty toteutuskieliriippumaton pseudokoodi, esim. seuraavaan tapaan:

```
A1:  read n;
      sum:= 0;
      for i:= 1 to n do sum:= sum + 2*i;
      return sum;
```

Havainnon (1) perusteella seuraava olisi vaihtoehtoinen (ja parempi) algoritmi:

¹Tämä on esimerkki tehtävän tyypillisestä muodosta muttei vaativuudeltaan vastaa syventävän kurssin koetehtävän tasoa.

```
A2:   read n;
      return n*(n+1);
```

Entä tehtävän jälkipuoli, algoritmin aika-analyysi?

Aikakompleksisuus tarkoittaa algoritmin ajantarpeen tai työmäärän riippuvuutta syötteen koosta. Se ilmaistaan syötteen kokoa ilmaisevasta parametrissa riippuvana lausekkeena ja esitetään usein yksinkertaisemmin asymptoottisesti kertaluokkana notaatioin $O()$ (yläraja), $\Theta()$ (tarkka kertaluokka) tai $\Omega()$ (aläraja).

Esitetään tässä algoritmin aikakompleksisuus riippuvuutena syöteluvun arvosta n (vaikka varsinaisesti luvun n pituus normaaleilla lukuesityksillä onkin vain $\Theta(\log n)$.)

Huom: Algoritmin palauttama arvo (1) on eri asia kuin algoritmin kompleksisuus.

Oletaan yksinkertaisuuden takia, että käsiteltävät luvut mahtuvat yhteen muistiinpaikkaan, jolloin niihin kohdistuvat perusoperaatiot onnistuvat vakioajassa. (Jos tämä oletus ei ole perusteltu, lukujen pituuden niiden käsittelyn työläytenä huomioiva ns. *logaritminen kustannus* on realistisempi.)

Algoritmi A1 suorittaa n kertaa rungoltaan vakioaikaisen silmukan (ynnä muutaman vakioajassa suoritettavan operaation), joten sen aikakompleksisuus on $\Theta(n)$. Myös ylärajanotaatio käyttävä $O(n)$ olisi oikea ilmaisu. Periaatteessa oikeita mutta epätärkkuutensa takia hylättäviä arvioita olisivat esimerkiksi $O(n \log n)$, $O(n^2)$, $O(n^3)$, $O(2^n)$ jne.

Algoritmi A2 toimii vakioajassa, eli sen aikakompleksisuus on $\Theta(1)$ tai $O(1)$.

Huom: Asymptoottinen merkintätapa on esimerkiksi

$$T(n) = O(n^2) ,$$

missä vasempana puolena on arvioitava lauseke, esimerkiksi aikakompleksisuus, ja oikeana puolena sen kertaluokka. Vastauksissa esiintyy usein merkintöjä $O(n) = n^2$, $n \leq O(n^2)$ tms, jotka ovat *merkitykseltään epäselviä*.

Yllämainittuja asioita opetetaan tietojenkäsittelytieteen peruskursseilla, kerrataan ja harjoitellaan ASA-kurssilla, ja niistä kerrotaan yleisesti saatavilla olevassa algoritmikirjallisuudessa. Pahimmat puutteellisuudet johtunevat siis yrityksistä suoriutua tentistä opiskelematta kurssin tai sen esitietoina edellytettyjen kurssien sisältöä.