T–106.4100 Algoritmien suunnittelu ja analyysi, syksy 2014

Tentti, 9.12.2014 **Ei apuvälineitä eikä laskimia**

Kirjoita jokaisen palauttamasi paperin yläreunaan selvästi "T-106.4100 Algoritmien suunnittelu ja analyysi, 9.12.2014", nimesi, opiskelijanumerosi ja koulutusohjelmasi sekä palauttamiesi paperien kokonaismäärä.

1. a) (3p) Mitkä seuraavista väittämistä pitävät paikkansa ja mitkä eivät? Perustele lyhyesti! (Lyhyt sanallinen perustelu riittää, tarkkoja matemaattisia todistuksia ei vaadita.)
   i. $2n^5 + 2n \in \Omega(n^4)$
   ii. $3n^6 + 7n \log n \in O(n^3)$
   iii. $2n + 16n \log\log n \in \Theta(n^3)$

   b) (3p) Verkon syvyyssuuntainen läpikäynti luokittelee verkon kaaret puukaariin, eteneviin kaariin, takautuviin kaariin ja poikittaiskaariin. Mitä nämä neljä tarkoittavat?

2. a) (3p) Ratkaise seuraava rekursioyhtälö, kun n on kolmen potenssi. Anna täsmällinen ratkaisu (suuruusluokka ei riitä).

$$T(n) = \begin{cases} 1, & \text{kun } n = 1 \\ 2T(n/3) + 3n & \text{kun } n > 1 \end{cases}$$

   b) (3p) Arvaa hyvä ratkaisu seuraavalle palautuskaavalle ja todista ratkaisusi oikeaksi induktiolla (c_1 ja c_2 ovat vakioita ja n on viiden potenssi).

$$T(n) \leq \begin{cases} c_1, & \text{kun } n = 1 \\ 5T(n/5) + c_2 n & \text{kun } n > 1 \end{cases}$$

3. a) (3p) Selitä, mitä tasoitettu vaativuus (amortized complexity) tarkoittaa ja miten se voidaan laskea pankkitili- eli potentiaalimenetelmällä.

   b) (3p) Esitä kaksi Fibonacci-keon operaatiota, joilla tasoitettu aikavaativuus on eri kuin pahimman tapauksen aikavaativuus. Anna näiden operaatioiden molemmat aikavaativuudet ja kerro, mihin nämä aikavaativuustulokset perustuvat (tarkkoja matemaattisia johtoja ei tarvitse esittää).

4. a) (2p) Selitä lyhyesti pyyhkäisyviiva-algoritmien (engl. sweeping) periaate.

   b) (4p) Union-find-rakenne (toiselta nimeltään disjoint-set forest) pitää yllä tietoa siitä, mihin pistevieraisiin joukkoihin tietyt alkiot kuuluvat, siten että joukkoja on tehokasta yhdistää (union).

   Käsitellään tilannetta, jossa jokaisella alkiolla on myös jokin arvo (positiivinen kokonaisluku), ja jokaisesta joukosta pitäisi milloin tahansa tehokkaasti saada selville, mikä on suurin joukkoon kuuluvien alkioiden arvoista. Miten muuttaisit union-find-rakenteen toteutusta pitämään kirjaa näistä maksimiarvoista tehokkaasti? Ongelman voi ratkaista niin, että operaatioiden aikavaativuudet eivät kasva vakiokerrointa enempää. Kerro joko tarvittavat muutokset tarkasti tai anna muuntamiesi operaatioiden (Makeset, Union ja Find) toteutukset pseudookoodina. Voit käyttää pohjana mitä tahansa Union-find-rakenteen toteutuksen versiota.

5. (6p) Selkäreppuongelmassa (knapsack problem) on annettu joukko esineitä, joilla kullakin on koko s_i ja arvo v_i ($1 \leq i \leq n$, missä n on esineiden määrä). Näistä pitäisi valita joukko esineitä, joiden yhteinen arvo on mahdollisimman suuri, mutta joiden yhteinen koko on enintään annettu raja V (eli valitse arvokkain osajoukko esineitä, joka vielä mahtuu V:n kokoiseen "selkäreppuun"). Keksi branch and bound -menetelmää käyttävä algoritmi, joka ratkaisee tämän ongelman. Kirjoita ratkaisustasi pseudokoodi. (Halutaan siis täydellistä ratkaisua eikä approksimointialgoritmia.)

Scanned by CamScanner

Summakaavoja:

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

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

$$\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$$

$$\sum_{i=0}^{n} a^i = \frac{a^{n+1} - 1}{a - 1}$$

$$\sum_{i=0}^{n} i\,a^i = \frac{n\,a^{n+2} - (n+1)\,a^{n+1} + a}{(a-1)^2}, \text{ jossa } a \neq 1$$

Logaritmi:

$$x_1 > x_2 \Rightarrow \log x_1 > \log x_2$$

$$\log x_1 = \log x_2 \Rightarrow x_1 = x_2$$

$$\log 1 = 0$$

$$\log(x_1 x_2) = \log x_1 + \log x_2$$

$$\log x^b = b \log x$$

$$x_1^{\log x_2} = x_2^{\log x_1} \quad (\log x_2 \log x_1 = \log x_1 \log x_2)$$

Scanned by CamScanner