

Tik–106.4100 Algoritmien suunnittelu ja analyysi, syksy 2011

Tentti 14.12.2011

Ei apuvälineitä eikä laskimia

Kirjoita jokaisen palauttamasi paperin yläreunaan selvästi kurssin koodi ja nimi sekä tentin päivämäärä, nimesi, opiskelijanumerosi ja tutkinto-ohjelmasi sekä palauttamiesi paperien kokonaismäärä.

1. a) (3p) Mitkä seuraavista väittämistä pitäävät paikkansa ja mitkä eivät? Perustele lyhyesti! (Lyhyt sanallinen perustelu riittää, tarkkoja matemaattisia todistuksia ei vaadita.)
 - i. $4n^3 + 5n \log n \in O(n^4)$
 - ii. $3n \log n + 5n \in \Omega(n)$
 - iii. $2n \log n + 5n + 16 \in \Theta(n^2)$b) (3p) Selitä, mitä tarkoitetaan algoritmin keskimääräisellä aikaavaativuudella (engl. average case time complexity), mitä tarkoitetaan algoritmin tasoitetulla aikaavaativuudella (engl. amortized time complexity) ja mitä eroa näillä kahdella on.
2. a) (3p) Ratkaise seuraava rekursioyhtälö, kun n on viiden potenssi. Anna täsmällinen ratkaisu (suuruusluokka ei riitä).

$$T(n) = \begin{cases} 1, & \text{kun } n = 1 \\ 3T(n/5) + 2n & \text{kun } n > 1 \end{cases}$$

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

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

3. a) (2p) Kerro, mihin kekoja (esim. binomikekoja) käytetään. Anna esimerkki jostain algoritmista, joka käyttää tärkeimpää keko-operaatioita. Itse algoritmia ei tarvitse tässä kuvata perusteellisesti. Riittää, että kerrot lyhyesti, mihin algoritmia käytetään ja mihin tarkoitukseen algoritmi käyttää tärkeitä keko-operaatioita.b) (4p) Miten binomi- ja Fibonacci-keot poikkeavat toisistaan? Minkä keko-operaatioiden aikaavaatimukset ovat näissä rakenteissa erilaiset? Esitä lyhyesti eri keko-operaatioiden aikaavaatimukset kummassakin rakenteessa ja perustele lyhyesti, mistä nämä aikaavaatimukset johtuvat (tarkkoja matemaattisia todistuksia ei tarvitse esittää).
4. (6 p) Tarkastellaan *rahanvaihto-ongelmaa*: Olkoon annettu rahasumma n ja m erilaista kolikon arvoa $\{d_1, d_2, \dots, d_m\}$. Kunkin arvoisia kolikoita on käytettäväissä rajaton määrä. Rahasumma n halutaan muodostaa kolikoiden avulla niin, että tarvittavien kolikoiden määrä on mahdollisimman pieni. (Määrää tarkastellessa vain kolikoiden yhteismäärä merkitsee. Kolikoiden arvoilla ei ole tässä merkitystä.)

Laadi dynaamista ohjelmointia käyttävä algoritmi, joka selvittää, miten käytettäväissä olevista kolikoista saadaan muodostettua haluttu rahasumma niin, että tarvittavien kolikoiden määrä on mahdollisimman pieni. Älä kirjoita algoritmissa pseudokoodina, vaan esitä laskennassa käytettävä lausekkeet, kerro kuinka ja missä järjestysessä niiden arvot lasketaan ja miten voidaan lopulta päättää, montako kappaletta kutakin kolikkoa tarvitaan (tai että rahasumman muodostaminen käytettäväissä olevista kolikoista ei ole lainkaan mahdollista).

Ratkaisusta ei saa täysiä pisteitä, jos se ei käytä dynaamista ohjelmointia.

Vinkki: tarkista, että ratkaisusi toimii oikein esimerkiksi silloin, kun $n = 110$ ja käytettäväissä olevat kolikoiden arvot ovat $\{20, 50\}$.

5. (6p.) Tämä tehtävä liittyy verkon maksimaalisen virtauksen laskemiseen.

Olkoon annettu suunnattu verkko $G = (V, E)$ siten, että jokaiseen kaareen $(u, v) \in E$ on liitetty sen kapasiteetti $c(u, v)$ ja kaarta pitkin tällä hetkellä menevä virtaus $f(u, v)$. Olkoon edelleen jo laskettu verkon tämänhetkistä virtausta vastaava jäännösverkko (engl. residual network) $G(f)$. Kirjoita pseudokoodi algoritmile, joka laskee jäännösverkon avulla yhden lyhimmän täydennyspolun (engl. augmenting path) solmusta s solmuun t ja täydennyspolku vastaavan täydennyksen. Täydennyspolun ei tarvitse olla paras mahdollinen, mutta sen on oltava mahdollisimman lyhyt.

Algoritmi saa siis syötteenä jäännösverkon sekä solmut s ja t ja sen tulee tulostaa täydennyspolkuun kuuluvat kaaret sekä polku vastaava täydennys.

Tik–106.4100 Design and Analysis of Algorithms, autumn 2011

Exam, December 14th, 2011

No calculators or extra material allowed

Write the following clearly on top of each paper you submit: "T-106.4100 Design and Analysis of Algorithms, Dec 14th, 2011", your full name, student ID and study programme, and the total number of papers you submit.

1. a) (3p) Which of the following conjectures are correct and which are incorrect? Give a short justification for each answer (exact mathematical proofs are not required)!
 - i. $4n^3 + 5n \log n \in O(n^4)$
 - ii. $3n \log n + 5n \in \Omega(n)$
 - iii. $2n \log n + 5n + 16 \in \Theta(n^2)$
- b) (3p) Explain what the average case time complexity of an algorithm means and what the amortized time complexity of an algorithm means. What is the difference between them?
2. a) (3p) Solve the following recurrence, when n is a power of five. An exact answer is required (an answer in Θ or O notation is not enough).

$$T(n) = \begin{cases} 1, & \text{when } n = 1 \\ 3T(n/5) + 2n & \text{when } n > 1 \end{cases}$$

- b) (3p) Make a good guess to solve the following recurrence and check your result by using induction (c_1 and c_2 are constants).

$$T(n) \leq \begin{cases} c_1, & \text{when } n = 1 \\ T(n-1) + c_2 \log n & \text{when } n > 1 \end{cases}$$

3. a) (2p) For which purposes are heaps (for example a binary heap, a binomial heap or a Fibonacci heap) used? Give an example of an algorithm which uses the most important heap operations. You do not have to explain the whole algorithm completely. It is enough to explain briefly the purpose of the algorithm and how the algorithm uses heap operations.
- b) (4p) Explain the main differences between binomial heaps and Fibonacci heaps. Which heap operations have different time complexities in these structures? Give the time complexities of those operations (in both binomial and Fibonacci heaps) and give short justifications for the complexity results (exact mathematical analysis is not required).
4. (6p) Design a dynamic programming algorithm for the *change-making problem*: given an amount n and unlimited quantities of coins of each of the denominations $\{d_1, d_2, \dots, d_m\}$, find the smallest number of coins that add up to n or indicate that the problem does not have a solution. In addition of the total number of coins needed, your algorithm has also to find out the number of coins of each denomination needed. (Note that you should minimize the total number of coins needed, not the number of different denominations.)

Do not write the code of the algorithm, but present the expressions needed by the dynamic programming and tell how and in which order the values of these expressions are calculated. Explain also how the number of coins of each denomination needed can be found out.

If your algorithm is not based on dynamic programming, you cannot obtain full points from your solution.

Hint: Check that your solution works properly if $n = 110$ and the denominations are $\{20, 50\}$.

5. (6p) Consider the problem of calculating the maximum flow in a flow network (graph).

Suppose we have a flow network $G = (V, E)$ (G is a directed graph) in which each edge $(u, v) \in E$ has a nonnegative capacity $c(u, v)$ and a current flow $f(u, v)$. Assume that the residual network of G induced by the current flow f has already been calculated. The residual network is denoted by $G(f)$. Write a pseudocode for an algorithm which calculates one shortest augmenting path from node s to node t by using the residual network $G(f)$. In addition to the augmenting path, your algorithm must calculate the maximal increase of flow in this augmenting path. The augmenting path calculated does not have to be the best of all augmenting paths, but it must be as short as possible.

The input of the algorithm are the residual network $G(f)$ and the nodes s and t . The output of the algorithm are the edges belonging to the augmenting path and the maximal increase of flow in this augmenting path.

Please, fill the course feedback form. The link is in the Noppa page of the course!