Linköpings Tekniska Högskola
Institutionen för Datavetenskap
Tommy Färnqvist

24 augusti 2011

TDDC70/TDDC91 Datastrukturer och algoritmer
Tentamen 2011-08-24, 14–19 (TER1)

Examinator: Tommy Färnqvist
Jour: Tommy Färnqvist (telefon 070 4547668).
Max poäng: 27 poäng (betyg 5 = 23p, 4 = 18p, 3 = 13p)
Hjälpmedel: INGA HJÄLPMEDEL TILLÅTNA!!!

VÄNLIGEN IAKTTAG FÖLJANDE

- Lösningar till olika problem skall placeras enkelsidigt på separata blad. Skriv inte två lösningar på samma papper.
- Sortera lösningarna innan de lämnas in.
- MOTIVERA DINA SVAR ORDENTLIGT: avsaknad av, eller otillräckliga, förklaringar resulterar i poängavdrag. Även felaktiga svar kan ge poäng om de är korrekt motiverade.
- Om ett problem medger flera olika lösningar, t.ex. algoritmer med olika tidskomplexitet, ger endast optimala lösningar maximalt antal poäng.
- SE TILL ATT DINA LÖSNINGAR/SVAR ÄR LÄSBARA.
- Lämna plats för kommentarer.

Lycka till!

1. Komplexitetsanalys (5 p)

   (a) Följande åtta funktioner kan jämföras med avseende på deras asymptotiska tillväxt. (2)

   $$n, 2^n, n \log n, n - n^3 + 7n^5, n^2 + \log n, n^2, \log n, n!$$

   Ordna dem i en sekvens $f_1(n), ..., f_8(n)$ sådan att $f_i(n)$ tillhör $\mathcal{O}(f_{i+1}(n))$ för $i = 1, ..., 7$. Motivera ditt svar.

   (b) En funktion på de naturliga talen har följande rekursiva definition: (1.5)

   $$F(0) = 0$$
   $$F(1) = 1$$
   $$F(n) = F(n-2) + F(n-1) \text{ för } n > 1$$

   Visa att en rekursiv procedur/metod baserad direkt på denna definition har exekveringstid $\Omega(2^{n/2})$.

   (c) Skriv pseudokod för en algoritm som beräknar funktionen $F(n)$ ovan i tid $O(n)$. Motivera att din algoritm har just denna tidskomplexitet. (1.5)

2. Algoritmiska paradigm (3 p)
   Av de fyra viktigaste metoderna för algoritmkonstruktion som nämnts i kursen är totalsökning en. Vilka är de andra tre? Ge för varje konstruktionsmetod exempel på en algoritm från kursen som bygger på den metoden. Skriv antingen namnet på algoritmen eller ange både problemet som algoritmen löser och ett $\mathcal{O}$-uttryck för tidskomplexiteten.

3. Datakompression (2 p)
   X, som precis har blivit färdig med sin doktorsavhandling om komprimeringsalgoritmer, har föreslagit en algoritm som, hävdar hon, komprimerar vilken fil som helst med åtminstone 10% av originalstorleken. (D.v.s. om en fil var x byte lång före komprimeringen, är den inte större än $0.9 \cdot x$ efter komprimeringen.) Tycker du att X verkar ha lärt sig mycket om komprimering under sin doktorandtid? Motivera din slutsats.

4. Stackar (2 p)
   Antag att Tommy har valt tre heltal och placerat dem på en stack i slumpmässig ordning. Skriv en snutt pseudokod (utan loopar eller rekursion) som bara använder en jämförelse och en variabel x, men ändå kan garantera att sannolikheten att variabeln x innehåller den största av Tommys tre heltal är 2/3. Motivera varför din metod fungerar.

5. Sökträd (3 p)

   (a) Betänk det binära sökträdet vars traversering i preorder ger följande sekvens av nycklar: (2)
   41 16 39 67 49 47 61 74.

   - Rita trädet och visa att det är ett AVL-träd. (0.5)
   - Visa steg för steg hur AVL-trädsoperationerna $Insert(40)$, $Insert(53)$, $Delete(67)$ (1.5)
     utförs på detta träd (varje operation ska appliceras på originalträdet).

   (b) Visa (2,3)-trädet som fås genom att sätta in följande nycklar i ett från början tomt träd: 41, 16, 39, 67, 49, 47 (1)

6. Implementationer av ADT Map (3 p)
På en initialt tom Map M utförs följande sekvens av operationer: $put(15, c), put(8, a), put(14, b), remove(15), put(32, d), put(4, e), put(7, f)$. Visa hur denna sekvens av operationer utförs på M implementerad som:

(a) en 7-elements *hashtabell* (indexerad från 0 till 6) med öppen adressering och dubbel hashning, där $h_1(k) = k \ mod \ 7$, $h_2(k) = 5 - (k \ mod \ 5)$, och borttagning implementeras med *deleted bit*-tekniken. (1.5)

(b) en *skiplista* där slumpsekvensen som används för att kontrollera *put*-operationerna är 1001110101100, där 1 indikerar höjning av insättningsnivån. (1.5)

7. Sortering (4 p)
Illustrera de konsekutiva stegen vid sortering av heltalsarrayen $[7, 5, 9, 6, 8]$ med hjälp av *in-place*-varianterna av följande algoritmer:

(a) Insertion Sort. Är algoritmen stabil? Motivera ditt svar. (1)

(b) Quick Sort där pivotelementet alltid väljs som sista elementet i sekvensen. (1)

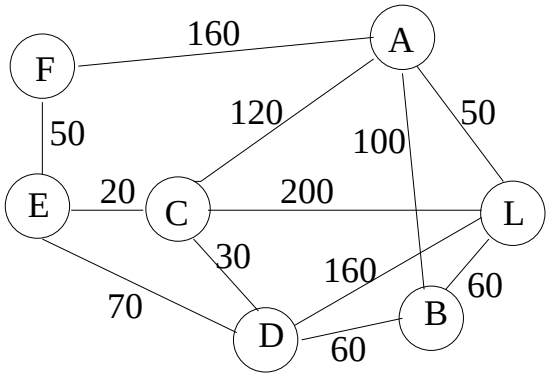
(c) Heap Sort med max-heap. Visa både array- och trädrepresentationerna för att illustera stegen. (2)

8. Algoritmkonstruktion (3 p)
Lisa och Sluggo bygger en robot i en projektkurs de läser och har upptäckt att de behöver passa in två legobitar sida vid sida i en öppning. Öppningen är x centimeter bred och summan av de två bitarnas bredder måste vara precis lika med öppningens bredd, annars faller roboten sönder under demonstrationen. De två konstruktörerna har tillgång till en enorm mängd legobitar med varierande bredder och måste välja två bitar som uppfyller ovanstående villkor.

Eftersom vi på IDA tycker att allt ska göras enligt väldefinierade algoritmer är din uppgift att förse Lisa och Sluggo med en asymptotiskt effektiv algoritm för att lösa deras problem!

Mer specifikt får du en mängd S bestående av n reella tal och ytterligare ett reellt tal x. Beskriv en algoritm, med exekveringstid $\mathcal{O}(n \log(n))$, som avgör huruvida det finns två element i S vars summa är exakt x eller ej.

9. Grafer (2 p)
Betrakta följande graf



(a) Visa sekvensen av noder som fås vid en djupetförstsökning av grafen med start i nod A där bågarna utforskas med stigande bågvikter. Visa det spännande träd DFS ger för grafen. (1)

(b) Visa sekvensen av noder som fås vid en breddenförstsökning av grafen med start i nod A där bågarna utforskas med stigande bågvikter. Visa det spännande träd BFS ger för grafen. (1)