Skriftlig Eksamen
Algoritmer og Datastrukturer (DM507)

Institut for Matematik og Datalogi
Syddansk Universitet, Odense

Tirsdag den 14. juni 2011, kl. 9–13

Løsningsforslag

Opgave 1 (18%)

Spørgsmål a (8%):

- ÷ T_1 er ikke et søgetræ (2 står til venstre for 1).
- ÷ T_2 har en rød rod.
- √ T_3, T_4 og T_8 har en sort rod, sorte blade og ikke to røde knuder i træk. Alle stier fra roden til et blad har det samme antal sorte knuder. Desuden står nøglerne rigtigt i forhold til hinanden.
- ÷ T_5 har to røde knuder i træk.
- ÷ T_6's sorte højde er ikke veldefineret.
- ÷ T_7's sorte højde er ikke veldefineret.

□

Spørgsmål b (10%): 3 indsættes som venstrebarn til 4.

Tilfælde 1: 1, 2 og 4 omfarves.

Tilfælde 3: Højre-rotation om 5.

Resultat: T_7 □

Opgave 2 (15%)

(a) er sand.

$f_1(n) \leq c_1 \cdot g_1(n)$, for $n \geq n_1$ og

$f_2(n) \leq c_2 \cdot g_2(n)$, for $n \geq n_2$

medfører, at

$f_1(n) + f_2(n) \leq \max(c_1, c_2) \cdot (g_1(n) + g_2(n))$, for $n \geq \max(n_1, n_2)$.

(b) er sand.

$f_1(n) \leq c_1 \cdot g_1(n)$, for $n \geq n_1$ og

$f_2(n) \leq c_2 \cdot g_2(n)$, for $n \geq n_2$

medfører, at

$g_1(n) + g_2(n) \geq \frac{1}{\max(c_1, c_2)} \cdot (f_1(n) + f_2(n))$, for $n \geq \max(n_1, n_2)$.

(c) er ikke nødvendigvis sand.

Eks:

$n^2 \in O(n^2)$, og

$n \in O(n^2)$, men

$\frac{n^2}{n} \notin O\left(\frac{n^2}{n^2}\right)$.

Opgave 3 (24%)

	Tilfælde 1 forskellige nøgler omvendt sorteret	Tilfælde 2 ens nøgler
Insertion Sort	$\Theta(n^2)$	$\Theta(n)$
Quicksort	$\Theta(n^2)$	$\Theta(n^2)$
Heapsort	$\Theta(n \log n)$	$\Theta(n)$
Radix Sort	$\Theta(n)$	$\Theta(n)$

Insertion Sort:

Tilfælde 1: Alle elementer flyttes frem til starten af listen. I alt $\Theta(n^2)$.

Tilfælde 2: Ingen elementer flyttes. D.v.s. der bruges $\Theta(1)$ tid på hvert element. Ialt $\Theta(n)$.

Quicksort: Ved hver opdeling bliver det ene del-array tomt. Det giver $\Theta(n^2)$.

Heapsort: Det tager tid $\Theta(n)$ at bygge hoben.

Tilfælde 1: Input udgør fra starten en hob. D.v.s. Build-Max-Heap ændrer ikke noget. Dermed er det hele tiden det mindste tal i hoben, som flyttes op i roden, og det skal derfor bobles hele vejen ned til bunden af hoben. D.v.s. de første $n/2$ elementer skal bobles ca. $\log n$ niveauer ned. Det giver i alt $\Theta(n \log n)$

Tilfælde 2: Ingen nøgler bobles ned. D.v.s. efter Build-Max-Heap bruges der $\Theta(1)$ tid på hvert element. I alt $\Theta(n)$.

Radix Sort: Ved at bruge radix n bliver der højst 6 iterationer, og hver iteration tager $\Theta(n)$ tid. I alt $\Theta(n)$.

Opgave 4 (18%)

	G_1	G_2	G_3
BFS	$\surd$	$\div$	$\div$
D-S-P	$\surd$	$\div$	$\surd$
Dijkstra	$\surd$	$\surd$	$\div$

Spørgsmål a (6%): BFS tæller blot kanter. D.v.s. den finder den korrekte korteste vej, når alle vægte er ens, og beregner den korrekte afstand, når alle vægte er 1. Altså giver BFS det korrekte svar for G_1.

På G_2 giver BFS 2, men det korrekte svar er 3.

På G_3 giver BFS 2 i stedet for 4. □

Spørgsmål b (6%): DAG-SHORTEST-PATHS fungerer på acykliske grafer, d.v.s. G_1 og G_3.

På cykliske grafer er man ikke garanteret et korrekt svar, da der ikke findes en topologisk ordning. Hvis man kører TOPOLOGICAL-SORT på G_2, kan man få ordningen s, a, c, b, t (alt efter i hvilken rækkefølge knuderne optræder i adjacens-listerne), og da fås afstanden 11 (den rigtige er 3). □

Spørgsmål c (6%): Dijkstras algoritme virker korrekt på grafer uden negative vægte, d.v.s. G_1 og G_2.

På G_3 inkluderer Dijkstras Algoritme t i S, inden den inkluderer a. Dermed returnerer den 5 i stedet for 4. □

Opgave 5 (25%)

Spørgsmål a (10%):

x	a	b	c	d	e	f	g	h
Med	10	6	5	4	2	3	2	4
Uden	16	7	0	2	4	0	0	0

□

Spørgsmål b (5%): Den største samlede vægt fås ved at udelade a.

Dermed skal vi for hvert af a's børn vælge den mulighed (tag barnet med eller udelad det), som giver den bedste løsning for barnets undertræ. D.v.s. b skal ikke med, men det skal c og d. Dermed kan g ikke komme med.

Da b ikke er med, er det lovligt at tage e med, men den samlede vægt i e's undertræ bliver størst ved at udelade e og tage h med.

Resultat: c, d, f og h.
Check: de fire knuder har samlet vægt $5 + 4 + 3 + 4 = 16$. □

Spørgsmål c (10%): Hver knude x har et tal x.vægt, som angiver knudens vægt, og en liste x.børn, som indeholder dens børn.

VægtMaxUafhMgd(T)

Hvis $r = \text{NIL}$
    return 0
Ellers
    For hver knude x i T
        $V[x, \text{med}] \leftarrow -1$
        $V[x, \text{uden}] \leftarrow -1$
    $r \leftarrow T.\text{root}$
    return $\max\big(\text{Med}(r), \text{Uden}(r)\big)$

Med(r)

Hvis $V[r, \text{med}] = -1$
    $V[r, \text{med}] \leftarrow r.\text{vægt}$
    For hvert $x \in r.\text{børn}$
        $V[r, \text{med}] \leftarrow V[r, \text{med}] + \text{Uden}(x)$
return $V[r, \text{med}]$

Uden(r)

Hvis $V[r, \text{uden}] = -1$
    $V[r, \text{uden}] \leftarrow 0$
    For hvert $x \in r.\text{børn}$
        $V[r, \text{uden}] \leftarrow V[r, \text{uden}] + \max\big(\text{Med}(x), \text{Uden}(x)\big)$
return $V[r, \text{uden}]$

For at beregne med- og uden-værdierne for en knude x bruges konstant tid for hvert af x's børn. Det giver i alt $\Theta(n)$ tid, hvor n er antallet af knuder i træet. Der gemmes en konstant mængde information for hver knude. D.v.s. pladsforbruget er også $\Theta(n)$. □