Skriftlig Eksamen
DM507 Algoritmer og Datastrukturer

Institut for Matematik og Datalogi
Syddansk Universitet, Odense

Mandag den 6. juni 2016, kl. 15:00–19:00

Besvarelsen skal afleveres elektronisk. Se vejledning udsendt i kurset.

Alle skriftlige hjælpemidler (lærebøger, notater, osv.) samt brug af computer er tilladt. Det er ikke tilladt at bruge internettet, undtagen til den elektroniske aflevering.

Eksamenssættet består af 10 opgaver på 8 nummererede sider (1–8). Fuld besvarelse er besvarelse af alle 10 opgaver. De enkelte opgavers vægt ved bedømmelsen er angivet i procent.

Der må gerne refereres til algoritmer og resultater fra lærebogen (Cormen et al., *Introduction to Algorithms*, tredie udgave), samt andre materialer fra kurset (f.eks. opgavesedler og slides). Henvisninger til andre kilder kan ikke bruges i besvarelsen af et spørgsmål.

Bemærk, at hvis der er et spørgsmål, man ikke kan besvare, må man gerne besvare de efterfølgende spørgsmål og blot antage, at man har en løsning til de foregående spørgsmål.

Opgave 1 (10%)

Denne opgaver handler om følgende rekursionsligninger:

i) $T(n) = 2 \cdot T(n/2) + 1$
ii) $T(n) = 3 \cdot T(n/3) + n$
iii) $T(n) = 4 \cdot T(n/4) + n \log n$
iv) $T(n) = 5 \cdot T(n/5) + n^2$

Spørgsmål a (5%):

Angiv for hver rekursionsligning *om* den kan løses med Master Theorem fra lærebogen, og i givet fald hvilken case det er.

Spørgsmål b (5%):

Angiv løsningen $T(n)$ for hver rekursionsligning som kan løses med Master Theorem.

Opgave 2 (10%)

Angiv for hvert af nedenstående udsagn, om de er sande eller falske.

i) 1 er $O(2)$
ii) 1 er $\Omega(2)$
iii) n er $O(n^2)$
iv) n er $\Omega(n^2)$
v) $3x + 2x^2 + x^3$ er $\Theta(x + 2x^2 + 3x^3)$
vi) $\log n$ er $o(n/\log n)$
vii) $n^{1/2}$ er $o(n/2^n)$
viii) $\log n$ er $\omega(\log n)$
ix) $2^n \cdot \log n$ er $\omega(2^n)$
x) $n^2/\log n$ er $O(n(\log n)^2)$

Opgave 3 (7%)

Udfør BUILD-MAX-HEAP(A) på nedenstående array A.

Angiv arrayets udseende bagefter ved at skrive elementerne i rækkefølge fra venstre mod højre.

	1	2	3	4	5	6	7	8	9
A:	2	1	5	4	8	6	7	9	3

Opgave 4 (6%)

Nedenstående er en hashtabel H som bruger hashfunktionen

$$h'(x) = (7x + 4) \bmod 11$$

samt linear probing (i lærebogen kaldes $h'(x)$ for "auxiliary hash function" i denne sammenhæng).

	0	1	2	3	4	5	6	7	8	9	10
H:	67	20	17		33		16	2			15

Indsæt værdierne 18 og 26 (i den rækkefølge) Angiv udseendet af hashtabellen efter hver af de to indsættelser.

Svar ved at skrive indholdet af H i rækkefølge fra venstre mod højre, med tomme pladser angivet som x.

Opgave 5 (7%)

En fil indeholder nedenstående tegn med de angivne hyppigheder. Der er 3300 tegn i alt.

Tegn	`x`	`y`	`z`	`æ`	`ø`	`å`
Hyppighed	1000	200	600	800	300	400

Lav et Huffman-træ på dette input. Angiv det resulterende kodeord for hvert af tegnene `x`, `y`, `z`, `æ`, `ø` og `å`, og angiv også hvor mange bits den kodede fil fylder (dvs. angiv den samlede længde i bits af de 3300 kodede tegn).

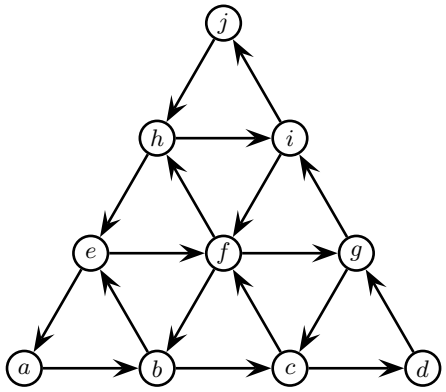
Opgave 6 (26%)

Spørgsmål a (7%):

Udfør BFS på grafen G_1 nedenfor, med start i knuden a. Du skal antage at en knudes naboliste er sorteret i alfabetisk orden efter naboknudernes navne.

Som svar, angiv knuderne i den rækkefølge de udtages fra køen (med operationen DEQUEUE) under algoritmens kørsel.

Angiv også for hver knude v slutværdien af $v.d$, dvs. afstanden fra a til v.



G_1

Spørgsmål b (7%):

Udfør DFS på grafen G_1 ovenfor, med start i knuden a. Du skal antage at en knudes naboliste er sorteret i alfabetisk orden efter naboknudernes navne.

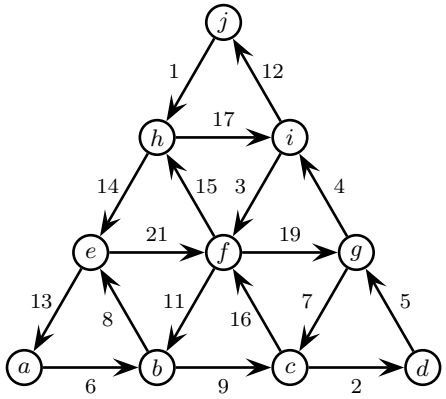
Som svar, angiv for alle knuder v starttiden (discovery time) $v.d$ og sluttiden (finishing time) $v.f$.

Spørgsmål c (7%):

Udfør Dijkstras algoritme på grafen G_2 nedenfor, med start i knuden a.

Som svar, angiv knuderne i den rækkefølge de udtages fra prioritetskøen (med operationen EXTRACT-MIN) under algoritmens kørsel.

Angiv også for hver knude v slutværdien af $v.d$, dvs. afstanden fra a til v.

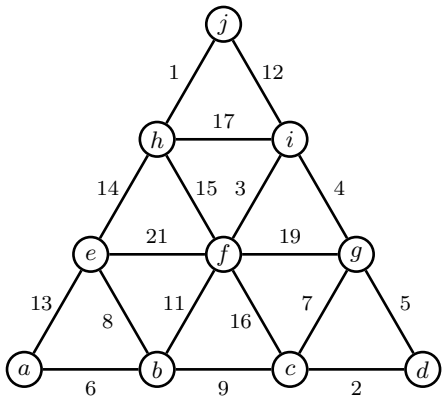


G_2

Spørgsmål d (5%):

Udfør Kruskals algoritme på grafen G_3 nedenfor.

Som svar, angiv kanterne i det resulterende minimum spanning tree (MST). En kant med endepunkter u og v skrives som sædvanligt (u, v).



G_3

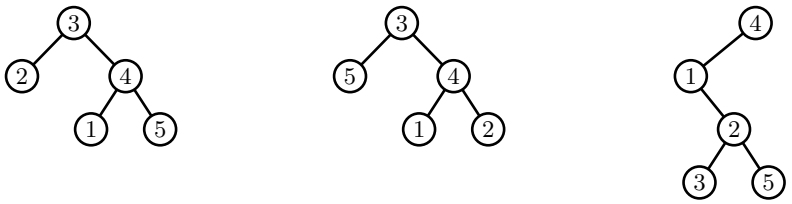
Opgave 7 (9%)

Binære træer er forskellige hvis de har forskellig facon. Figuren herunder viser to forskellige binære træer med fem knuder.

Spørgsmål a (3%):

Hvor mange forskellige binære træer med tre (3) knuder findes der? Svar ved at angive antallet.

I resten af opgaven ser vi på binære træer med nøgler i knuderne. To træer er forskellige hvis de har forskellig facon, eller hvis de har samme facon men forskellig nøgle i mindst én knude. Figuren herunder viser tre forskellige binære træer med fem knuder og nøglerne 1, 2, 3, 4, 5.



Spørgsmål b (2%):

Hvor mange forskellige binære træer med tre knuder og nøglerne 1, 2, 3 findes der som overholder max-heaporder? Svar ved at angive antallet.

Spørgsmål c (2%):

Hvor mange forskellige binære træer med tre knuder og nøglerne 1, 2, 3 findes der som overholder inorder? Svar ved at angive antallet.

Spørgsmål d (2%):

Hvor mange forskellige binære træer med tre knuder og nøglerne 1, 2, 3 findes der som overholder både inorder og max-heaporder? Svar ved at angive antallet.

Opgave 8 (5%)

Følgende kode har til formål at beregne $7 \cdot n$ for heltal $n \geq 0$.

```
GangeSyv(n)
    x = n
    r = 0
    while x > 0
        x = x − 1
        r = r + 7
    return r
```

Angiv for hvert af nedenstående udsagn om det er en løkke-invariant for algoritmen (dvs. altid er sandt når testen i starten af **while**-løkken udføres) for alle input der er heltal $n \geq 0$. (Du behøver ikke argumentere for svarene.)

i) $r = 7n$

ii) $r < 7n$

iii) $n - x = 7r$

iv) $7x + r = 7n$

v) $x \geq 0$

Opgave 9 (8%)

Angiv for hver af følgende algoritmer deres asymptotiske køretid i Θ-notation som funktion af n. Input n er for alle algoritmerne et positivt heltal.

```
ALGORITME1(n)
    for i = 1 to n
        j = i
        while j > 0
            j = j − 1
```

```
ALGORITME2(n)
    i = 1
    while i < n
        i = i ∗ 2
```

```
ALGORITME3(n)
    i = 1
    while i < n
        i = i ∗ n
```

```
ALGORITME4(n)
    i = 1
    while i < n
        s = 0
        while s < n
            s = s + i
        i = i ∗ 2
```

Opgave 10 (12%)

Denne opgave handler om hvordan heltal kan skrives som summer af kvadrater. Nogle eksempler på dette er:

$$
\begin{aligned}
7 &= 2^2 + 1^2 + 1^2 + 1^2 \\
25 &= 5^2 \\
25 &= 4^2 + 3^2 \\
33 &= 4^2 + 4^2 + 1^2 \\
33 &= 4^2 + 3^2 + 2^2 + 2^2
\end{aligned}
$$

For et ikke-negativt heltal n lader vi $K(n)$ være det mindste antal heltal hvis sum af kvadrater giver n. Dvs. for $k = K(n)$ findes der k heltal $a_1, \ldots, a_k$ således at $n = a_1^2 + \cdots + a_k^2$, mens dette ikke gælder for $k < K(n)$. Som eksempel er $K(25) = 1$ og $K(33) = 3$.

$K(n)$ kan bestemmes ved følgende rekursive formel:

$$K(n) = \begin{cases} 0 & \text{hvis } n = 0 \\ 1 + \min\{K(n - a^2) \mid a \text{ er et positivt heltal hvor } a^2 \leq n\} & \text{hvis } n \geq 1 \end{cases}$$

Spørgsmål a (3%):

Angiv $K(19)$, samt en måde at skrive 19 som summen af dette antal kvadrater.

Spørgsmål b (3%):

Beskriv en algoritme baseret på dynamisk programmering, der givet n beregner $K(n)$. Angiv algoritmens asymptotiske køretid og pladsforbrug som funktion af n.

Spørgsmål c (3%):

Beskriv en udvidelse af algoritmen som også finder k heltal $a_1, \ldots, a_k$ som opfylder $n = a_1^2 + \cdots + a_k^2$, med $k = K(n)$.

Spørgsmål d (3%):

Argumenter for at den rekursive formel for $K(n)$ er korrekt.