Skriftlig Eksamen
Algoritmer og Datastrukturer (DM507)

Institut for Matematik og Datalogi
Syddansk Universitet, Odense

Onsdag den 10. juni 2009, kl. 9–13

Alle sædvanlige hjælpemidler (lærebøger, notater, osv.) samt brug af lommeregner er tilladt.

Eksamenssættet består af 5 opgaver på 7 nummererede sider (1–7).
Fuld besvarelse er besvarelse af alle 5 opgaver.
De enkelte opgavers vægt ved bedømmelsen er angivet i procent.

Der må gerne refereres til algoritmer og resultater fra lærebogen inklusive øvelsesopgaverne. Henvisninger til andre bøger accepteres ikke som besvarelse 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.

Husk at begrunde dine svar!

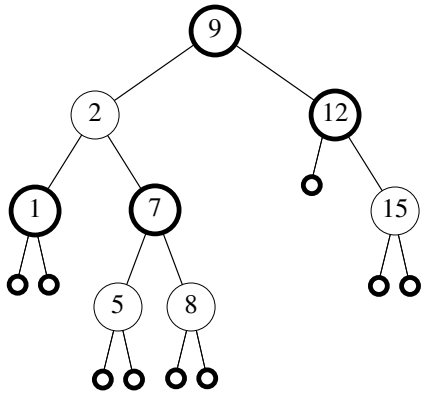
Opgave 1 (20%)

Spørgsmål a (7%): Udfør HEAP-EXTRACT-MAX på den binære hob repræsenteret ved nedenstående array.

10	7	6	5	4	2	3	1	2	3	1	1

Vis hvert skridt. Det kan være en god ide at tegne træ-repræsentationen fremfor array-repræsentationen. □

Spørgsmål b (6%): Betragt nedenstående rød-sorte træ, hvor sorte knuder er tegnet med fed. Tegn træet, som det ser ud, efter at knuden med nøgle 2 er slettet.



□

Spørgsmål c (7%): Betragt følgende hash-tabel med 16 pladser.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
32				20			23	7		39					

Der bruges åben adressering og kvadratisk probing med

$$h'(k) = k \bmod 16 \quad \text{og} \quad c_1 = c_2 = \frac{1}{2}\,.$$

Nu indsættes et element med nøgle 71. På hvilken plads havner elementet? □

Opgave 2 (15%)

Betragt følgende algoritme, som beregner et heltal tæt på $\sqrt{n}$.

```
KvadratRod(n)
i ← 0
s ← 0
while s ≤ n
    s ← s + 2i + 1
    i ← i + 1
r ← i − 1
return r
```

Spørgsmål a (8%): Bevis følgende invariant for while-løkken:
I starten af while-løkken er $s = i^2$

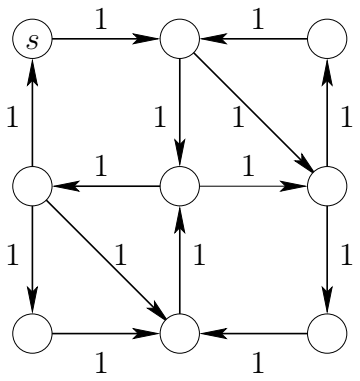
□

Spørgsmål b (7%): Brug invarianten fra spørgsmål a til at argumentere for, at algoritmen KvadratRod returnerer det største heltal, som er mindre end eller lig med $\sqrt{n}$. □

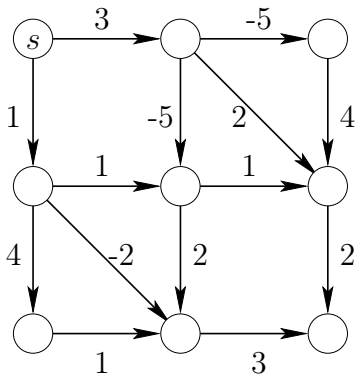
Opgave 3 (20%)

For hver af de tre grafer G_1, G_2 og G_3 skal du angive en simplest mulig algoritme, som kan bruges til at finde de korteste veje fra s til hver af de andre knuder.
Begrund dit valg af algoritme.

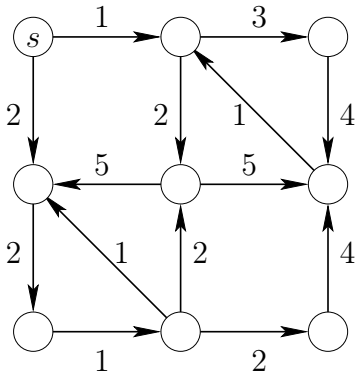
Angiv desuden afstandene fra s til hver af de andre knuder.
Graferne er også gengivet på næst-sidste side i dette sæt. Du kan evt. skrive afstandene ved knuderne på denne side (husk i så fald at aflevere siden sammen med resten af besvarelsen).



(a) G_1



(b) G_2



(c) G_3

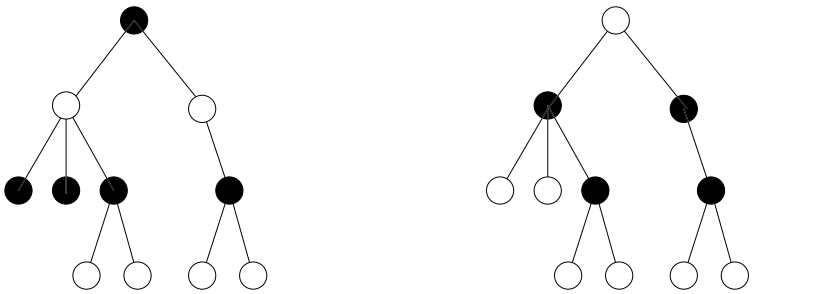
Figur 1: De tre grafer

Opgave 4 (20%)

En *knude-dækning* af en graf $G = (V, E)$ er en delmængde $V' \subseteq V$ af knuderne, sådan at alle kanter i E er incidente til mindst en knude i V'.

Denne opgave handler om at finde en mindst mulig knude-dækning af et træ.

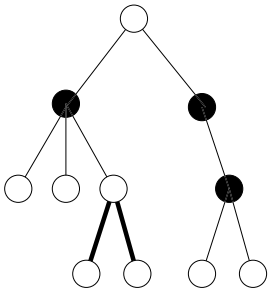
Knuderne, som er farvet sorte i Figur 2, er eksempler på knude-dækninger. De sorte knuder i Figur 3 udgør *ikke* en knude-dækning, da de to kanter tegnet med fed ikke er dækkede.



(a) En knude-dækning

(b) En mindst mulig knude-dækning

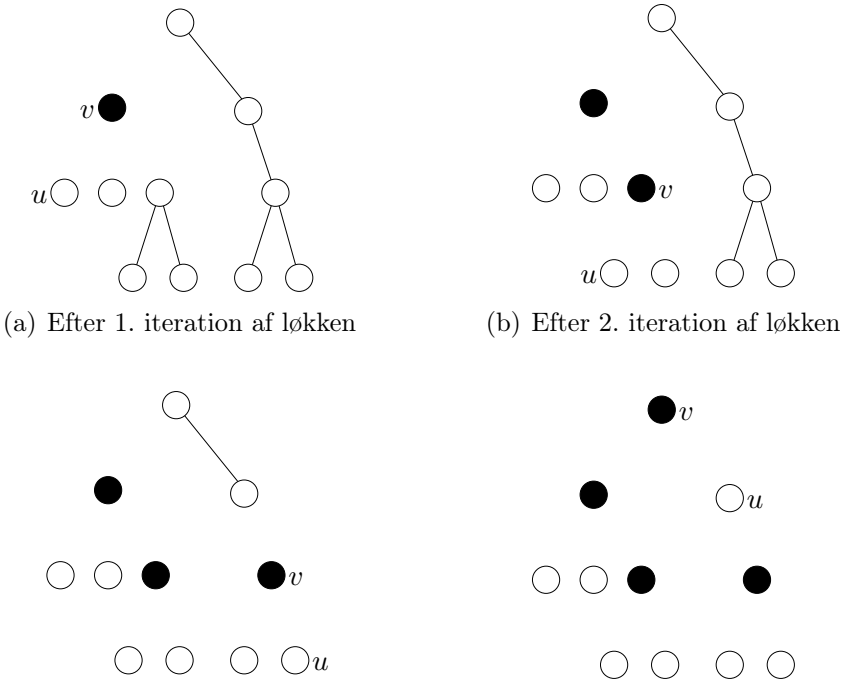
Figur 2: To forskellige knude-dækninger af det samme træ



Figur 3: *Ikke* en knude-dækning

Betragt følgende grådige algoritme til at finde en knude-dækning af et træ $T = (V, E)$, hvor $e(v)$ betyder mængden af kanter, som er incidente til knuden v.

KNUDE-DÆKNING(V, E)
$V' \leftarrow \emptyset$
while $E \neq \emptyset$
    Vælg en knude v, som har et barn u, der ikke (længere) har nogen børn
    $V' \leftarrow V' \cup \{v\}$
    $E \leftarrow E - e(v)$
Return V'



(a) Efter 1. iteration af løkken

(b) Efter 2. iteration af løkken

(c) Efter 3. iteration af løkken

(d) Efter 4. iteration af løkken

Figur 4: Muligt forløb af algoritmen KNUDE-DÆKNING på træet fra Figur 2

Spørgsmål a (10%): Bevis, at algoritmen KNUDE-DÆKNING altid finder en mindst mulig knude-dækning af input-træet. □

Spørgsmål b (10%): Forklar, hvordan man kan finde en mindst mulig knude-dækning af et træ i lineær tid; dvs. i $O(n)$ tid, hvor n er antallet af knuder i træet. □

Opgave 5 (25%)

Denne opgave handler om dynamisk programmering.

Et *palindrom* er et ord, der læses ens forfra og bagfra, f.eks. "bob", "abba" og "kajak". Lad $S = x_1, x_2, \ldots, x_n$ være en sekvens af længde n. I denne opgave ønsker vi at finde en længst mulig delsekvens af S, som er et palindrom.

> *Eksempel:*
> `ADA` er en længst mulig delsekvens af `ANDREAS`, som er et palindrom,
> og
> `ABBA` er en længst mulig delsekvens af `SABBAT`, som er et palindrom.

Hvis følgende rekursive algoritme kaldes med $\mathrm{LP}(S, 1, n)$, finder den længden af en længst mulig delsekvens af S, som er et palindrom.

$\underline{\mathrm{LP}}(S, i, j)$
if $i > j$
    return 0
if $i = j$
    return 1
if $i < j$ and $x_i = x_j$
    return $2 + \mathrm{LP}(i + 1, j - 1)$
if $i < j$ and $x_i \neq x_j$
    return $\max\{\mathrm{LP}(i + 1, j), \mathrm{LP}(i, j - 1)\}$

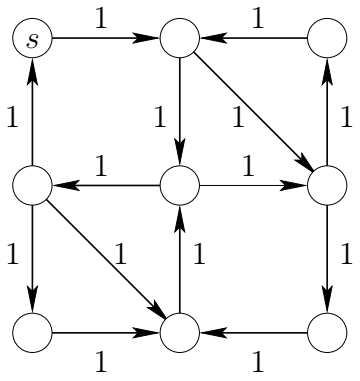
Spørgsmål a (9%): Indsæt værdierne for LP(`ALBA`,i, j) i nedenstående tabel. På sidste side af sættet er der en kopi af tabellen, som du evt. kan bruge (husk i så fald at aflevere siden sammen med resten af din besvarelse).

		j			
		1	2	3	4
i	1				
	2				
	3				
	4				

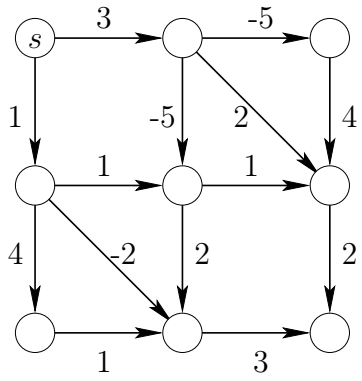
□

Spørgsmål b (11%): Skriv en *iterativ* algoritme, som vha. *dynamisk programmering* finder længden af en længste delsekvens af S, som er et palindrom. □

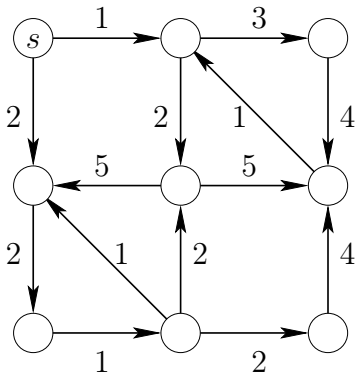
Spørgsmål c (5%): Hvad er køretid og pladsforbrug for algoritmen fra spørgsmål b? □



(a) G_1



(b) G_2



(c) G_3

Figur 5: De tre grafer fra opgave 3

		j			
		1	2	3	4
i	1				
	2				
	3				
	4				

Figur 6: Tabellen fra opgave 5