Skriftlig Eksamen
Algoritmer og Datastrukturer (DM507)

Institut for Matematik og Datalogi
Syddansk Universitet, Odense

Torsdag den 11. januar 2007, kl. 9–13

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

Eksamenssættet består af 5 opgaver på 9 nummererede sider (1–9).
Fuld besvarelse er besvarelse af alle 5 opgaver.
De enkelte opgavers vægt ved bedømmelsen er angivet i procent.

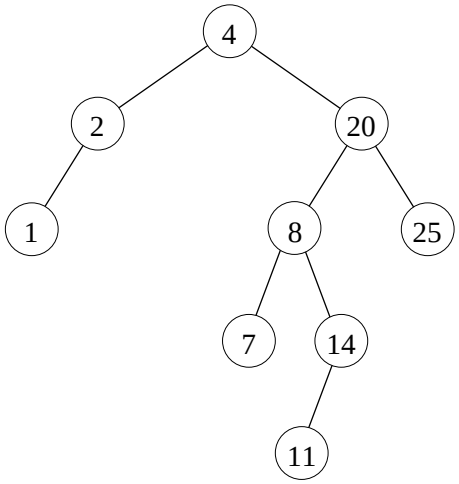
Der må gerne refereres til algoritmer og resultater fra lærebogen inklusive øvelsesopgaverne. Henvisninger til andre bøger accepteres ikke som besvarelse af et spørgsmål.

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!

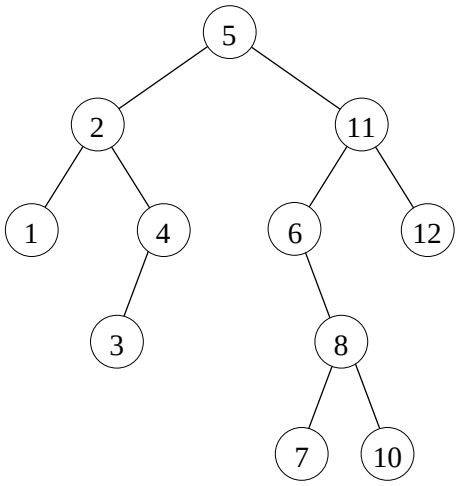
Opgave 1 (20%)

Spørgsmål a (5%): Er nedenstående et binært søgetræ?



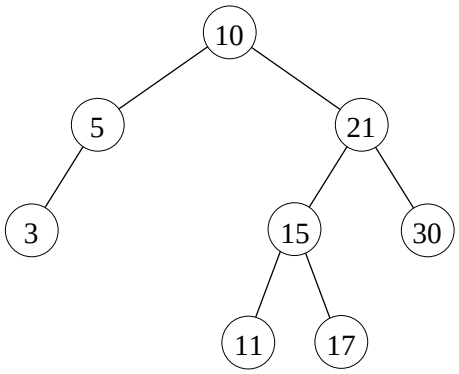
□

Spørgsmål b (8%): Betragt nedenstående binære søgetræ. Tegn træet, som det ser ud, efter at knuden med nøgle 5 er slettet. Du skal bruge algoritmen fra lærebogen (s. 261).



□

Spørgsmål c (7%): Betragt nedenstående binære søgetræ. Tegn træet, som det ser ud, efter at en knude med nøgle 16 er blevet indsat. Du skal bruge algoritmen fra lærebogen (s. 262).



□

Opgave 2 (15%)

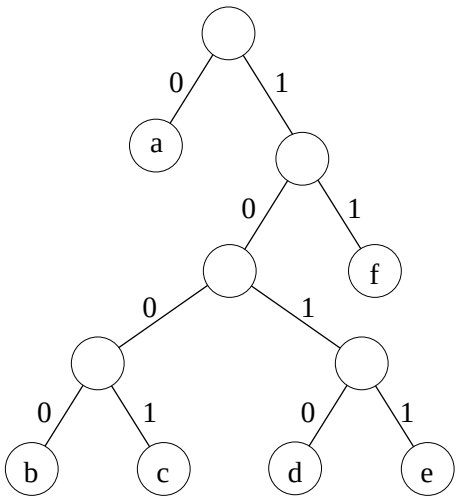
Denne opgave handler om Huffman-koder.

Spørgsmål a (8%): Betragt alfabetet med de seks tegn a, b, c, d, e, f. Nedenstående tabel viser, hvor tit hvert enkelt tegn optræder i en given tekst.

a	b	c	d	e	f
33	28	52	20	10	12

Tegn Huffman-træet, som repræsenterer Huffman-koderne for dette eksempel. □

Spørgsmål b (7%): Brug følgende Huffman-træ



til at dekode 1101001010101011 □

Opgave 3 (15%)

Spørgsmål a (15%): Hvilke af følgende fem udsagn er sande?

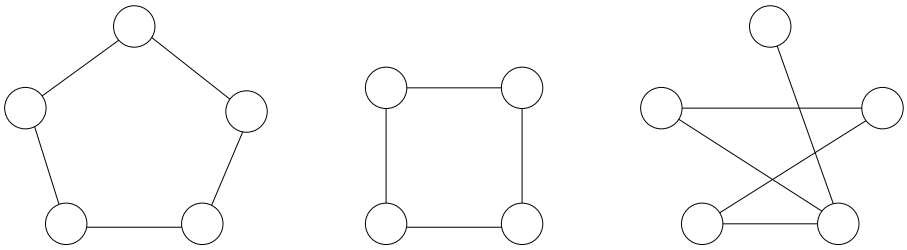
1. $n^2 \in \Omega(n)$
2. $n \in \Theta(n^2)$
3. $n \log n \in o(n^2)$
4. $\log n \in O(\sqrt{n})$
5. $n! \in \omega(2^n)$

□

Opgave 4 (20%)

Denne opgave handler om todelte grafer. En *todelt graf* er en graf, hvis knudemængde kan deles i to disjunkte mængder X og Y, sådan at hver kant har det ene endepunkt i X og det andet endepunkt i Y.

Spørgsmål a (6%): Hvilke af de tre grafer nedenfor er todelte? For de grafer, der er todelte, skal du angive en todeling.



(a) (b) (c)

□

Spørgsmål b (7%): Modificer BFS-algoritmen, så den afgør, om inputgrafen er todelt. Din algoritme skal have samme asymptotiske køretid som BFS.

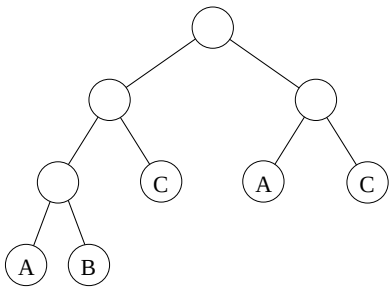
Argumenter for, at din algoritme er korrekt. Du skal ikke give et formelt bevis. □

Spørgsmål c (7%): En ulige kreds er en kreds, som indeholder et ulige antal kanter. Argumenter for, at en graf er todelt, hvis og kun hvis den ikke indeholder en ulige kreds. □

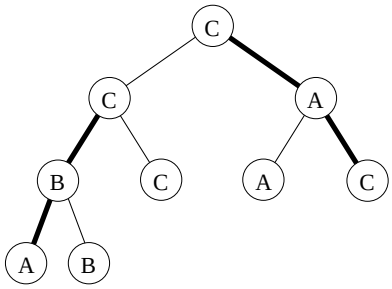
Opgave 5 (30%)

Denne opgave handler om et problem, som kaldes parsimony-problemet. Input er et binært træ, hvor alle knuder har præcis 0 eller 2 børn, og hvor hvert blad indeholder et tegn fra et givet alfabet Σ. Output skal være en tildeling af tegn fra alfabetet til de interne knuder, sådan at der bliver færrest muligt kanter, hvis endepunkter indeholder forskellige tegn. Sådanne kanter kaldes *overgangskanter*.

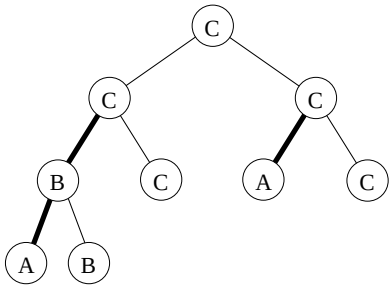
Eksempel: Betragt nedenstående input-træ:



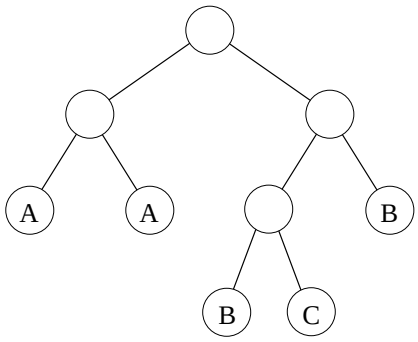
Man kunne vælge at tildele tegn til de indre knuder på følgende måde:



Men dette er ikke en optimal løsning, idet der er 4 overgangskanter (tegnet med tykke streger), og man kan nøjes med 3 overgangskanter:



Spørgsmål a (5%): Betragt nedenstående træ. Tildel tegn til de indre knuder, så der bliver så få overgangskanter som muligt.



□

Lad T være et binært træ, hvor alle knuder har 0 eller 2 børn, og hvert blad indeholder et tegn fra Σ. Hver knude k har en reference til sit venstrebarn k.left og sit højrebarn k.right. For hvert blad k angiver k.tegn tegnet indeholdt i k.

Betragt nedenstående pseudokode. Hvis TotalCost kaldes med roden i T som argument, beregner den det mindste antal overgangskanter, man kan nøjes med, når man tildeler tegn til de indre knuder i T.

```
    TotalCost(r)
1   Returner min_{a∈Σ} {Cost(r, a)}
```

(Koden fortsættes på næste side.)

```
    COST(k, a)
2   Hvis k er et blad
3       Hvis k.tegn = a
4           returner 0
5       Ellers
6           returner ∞
7   Ellers
8       minLeft ← min_{b∈Σ} {OVERGANG(a, b) + COST(k.left, b)}
9       minRight ← min_{b∈Σ} {OVERGANG(a, b) + COST(k.right, b)}
10      Returner (minLeft + minRight)

    OVERGANG(a, b)
11  Hvis a = b
12      Returner 0
13  Ellers
14      Returner 1
```

Linie 1 realiseres vha. kaldet $\textsc{Min}_1(r)$ til følgende algoritme:

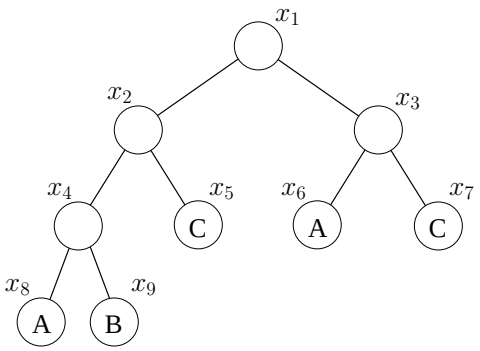
```
    MIN_1(k)
15  min ← ∞
16  For hvert a ∈ Σ
17      temp ← COST(k, a)
18      Hvis temp < min
19          min ← temp
20  Returner min
```

Linie 8 realiseres vha. kaldet $\textsc{Min}_2(k.\text{left}, a)$ til følgende algoritme:

```
    MIN_2(k, a)
21  min ← ∞
22  For hvert b ∈ Σ
23      temp ← OVERGANG(a, b) + COST(k, b)
24      Hvis temp < min
25          min ← temp
26  Returner min
```

Tilsvarende realiseres linie 9 vha. kaldet $\textsc{Min}_2(k.\text{right}, a)$.

Spørgsmål b (7%): Betragt igen træet fra side 6. Antag, at knuderne er navngivet $x_1, x_2, \ldots, x_9$:



Antag, at $\Sigma = \{A, B, C\}$. Hvor mange gange kaldes Cost med x_4 som det ene argument? □

Spørgsmål c (8%): Argumentér for, at TotalCost har køretid $\Omega(|\Sigma|^{\log_2 n})$, hvor n er antallet af knuder i træet, og $|\Sigma|$ er størrelsen af alfabetet. □

Spørgsmål d (10%): Lav en ny version af algoritmen TotalCost vha. dynamisk programmering. Du må gerne antage, at hver knude k har et unikt nummer k.id mellem 1 og n, og at Σ blot består af heltallene mellem 1 og $|\Sigma|$.

Hvad er worst-case asymptotisk køretid og pladsforbrug for din algoritme? □