Skriftlig Eksamen
Algoritmer og Datastrukturer (DM507)

Institut for Matematik og Datalogi
Syddansk Universitet, Odense

Mandag den 7. juni 2010, kl. 9–13

LØSNINGSFORSLAG

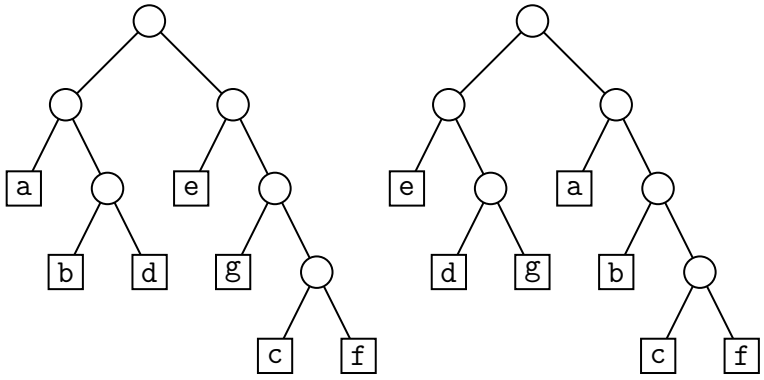
Opgave 1

Spørgsmål a: I master theorem er $a = 16$, $b = 2$ og $f(n) = n^4 + n^2$. Da $\log_b a = 4$ og $n^4 + n^2 = \Theta(n^4)$, er vi i Case 2 i master theorem, og løsningen er derfor $T(n) = \Theta(n^4 \log n)$.

□

Spørgsmål b:

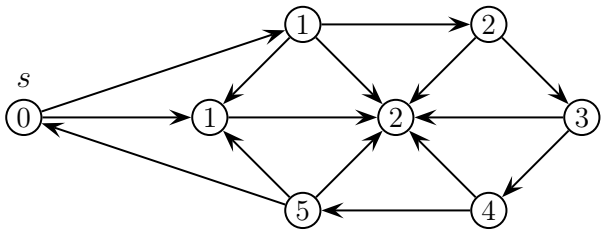
Der er følgende to forskellige løsninger (én er nok i besvarelsen), da der for dette input i et skridt i algoritmen er to lovlige valg. (Og alle træer som kan laves ved at bytte rundt på højre og venstre barn i een eller flere knuder er også løsninger.)



□

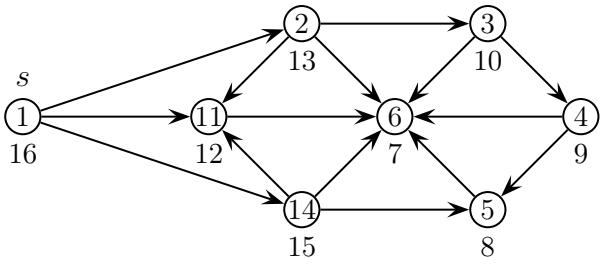
Opgave 2

Spørgsmål a: Distanceværdierne er angivet i knuderne.



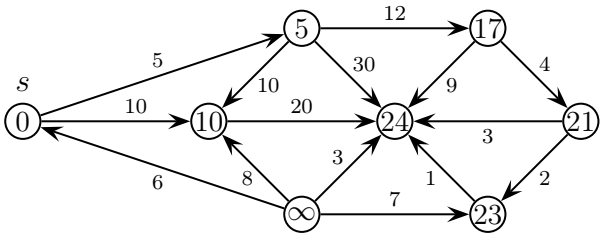
□

Spørgsmål b: Discovery times er angivet inden i knuderne, finishing times nedenunder knuderne.



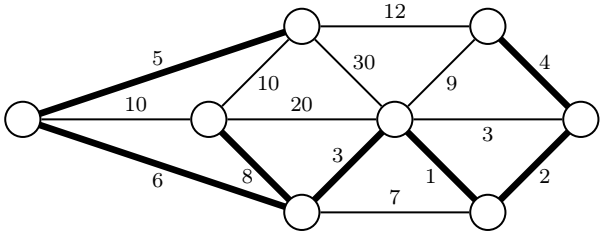
□

Spørgsmål c: Distanceværdierne er angivet inden i knuderne.



□

Spørgsmål d: Følgende minimum spanning tree, angivet med fede kanter, kan findes med f.eks. Kruskals algoritme. Det har vægt 29.



□

Opgave 3

Spørgsmål a:

For $n = 55$ er $k = 5$ og $b_5 b_4 \ldots b_1 b_0 = 110111$.

□

Spørgsmål b: Det vises ved induktion på antal gange testen i **while**-løkken er blevet foretaget.

Basis: Testen er blevet foretaget nul gange. Her følger invarianten del i) af initialiseringen $i = 0$ og $d = n$, da $n \cdot 2^0 + \sum_{j=0}^{-1} b_j \cdot 2^j = n \cdot 1 + 0 = n$. Del ii) følger af at $d = n$, og n er et positivt tal.

Induktionsskridt: For del i), antag udsagnet er sandt før et gennemløb af løkken. Vi skal så vise at det gælder efter gennemløbet. Kald de nye værdier af i og d efter gennemløbet for i' og d'. Vi har $i' = i + 1$, $d' = d \text{ div } 2$, samt $b_i = d \bmod 2$.

Af induktionsantagelsen (at invarianten gælder før gennemløbet) samt formlen (1) med $x = d$ og $y = 2$ fås

$$\begin{aligned}
n &= d \cdot 2^i + \sum_{j=0}^{i-1} b_j \cdot 2^j \\
&= (2 \cdot (d \text{ div } 2) + (d \bmod 2)) \cdot 2^i + \sum_{j=0}^{i-1} b_j \cdot 2^j \\
&= (2 \cdot d' + b_i) \cdot 2^i + \sum_{j=0}^{i-1} b_j \cdot 2^j \\
&= d' \cdot 2^{i+1} + \sum_{j=0}^{i} b_j \cdot 2^j \\
&= d' \cdot 2^{i'} + \sum_{j=0}^{i'-1} b_j \cdot 2^j,
\end{aligned}$$

hvilket var, hvad vi skulle vise.

For del ii) bruger vi at division (også heltalsdivision) med to ikke-negative tal altid giver ikke-negative tal, så af induktionsantagelsen $d \geq 0$ fås $d' = d \text{ div } 2 \geq 0$.

□

Spørgsmål c:

Vi viser først at algoritmen terminerer. I starten er $d = n$, og n er et heltal. Alle senere værdier af d er resultatet af en heltalsdivision, og derfor heltal, så d er altid et heltal. Hvis $d > 0$ gælder derfor $d \geq 1$, så for ethvert løkkegennemløb har vi $d' = \lfloor d/2 \rfloor \leq d - 1$. Da løkken stopper når $d \leq 0$, kan løkken ikke køre uendeligt.

Derefter viser vi at output er korrekt når den stopper. Når løkken stopper ved vi at $d \leq 0$. Af invarianten del ii) haves $d \geq 0$. Altså gælder $d = 0$ når løkken stopper. Indsættes dette sammen med $k = i - 1$ i invarianten del i) fås

$$n = \sum_{j=0}^{k} b_j \cdot 2^j,$$

hvilket vi skulle vise.

□

Spørgsmål d:

Vi har for ethvert løkkegennemløb $d' = \lfloor d/2 \rfloor \leq d/2$. Da $d = n$ før første gennemløb, gælder efter t gennemløb at $d \leq n/2^t$. Før sidste gennemløb af løkken har vi $d \geq 1$ (da $d > 0$ og da d som argumenteret ovenfor altid er et heltal), så før sidste gennemløb har vi $1 \leq d \leq n/2^t$, hvoraf følger $t \leq \log n$. Så det samlede antal gennemløb er højst $1 + \log n$. Hvert gennemløb tager $O(1)$ tid, og derudover udføres $O(1)$ yderligere arbejde før og efter løkken. Alt i alt er køretiden $O(\log n)$.

□

Opgave 4

Spørgsmål a:

$i \setminus j$	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	2	2	2	2
2	0	0	2	3	3	3
3	0	4	4	4	4	4
4	0	4	4	4	7	7

□

Spørgsmål b:

Vi udfylder en tabel som ovenstående på struktureret måde, baseret på de afhængigheder som findes i rekursionsformlen. Her er afhængigheden at man under beregning af feltet $W(i, j)$ skal kende $W(i-1, j)$, $W(i, j-1)$ og $W(i-1, j-1)$, dvs. de tre nabofelter til venstre og opad. Denne afhængighed følger af rekursionsformlens to sidste dele. Rekursionsformlens første del giver direkte værdier at indsætte i tabellens venstre søjle og øverste række, til initialisering af processen. Til sidst aflæses resultatet $W(n, m)$.

Der er flere måder, hvorpå man struktureret kan udfylde tabellen under iagttagelse af afhængighederne. Én af dem er rækkevis fra oven og nedad, hvor hver række udfyldes fra venstre mod højre. I pseudo-kode bliver det:

```
LCWS(X,Y)
    n = X.length
    m = Y.length
    for i = 0 to n
        W(i,0) = 0
    for j = 0 to m
        W(0,j) = 0
    for i = 1 to n
        for j = 1 to m
            if x_i ≠ y_j
                W(i,j) = max{W(i-1,j), W(i,j-1)}
            else
                W(i,j) = max{W(i-1,j), W(i,j-1),
                             W(i-1,j-1) + w(x_i)}
    return W(n,m)
```

Da hver indgang udfyldes under læsning af højst tre andre indgange (som alle *er* udfyldt på det givne tidspunkt), udfyldes hver indgang i $O(1)$ tid. Den samlede tid bliver derfor propertional med antal indgange, hvilket er $O(nm)$. Man kan her også argumentere ud fra pseudo-koden, hvis løkker kører henholdsvis $n+1$, $m+1$, og $n \cdot m$ gange, og alle laver $O(1)$ arbejde i hvert gennemløb.

□

Spørgsmål c:

Vi bruger notationen $X_i = x_1 \dots x_i$ og $Y_j = x_1 \dots x_j$. Se på en optimal løsning L for strengparret X_i og Y_j, dvs. en fælles delsekvens for disse med vægt $W(i,j)$. Som enhver anden fælles delsekvens kan den ses som en parring af ens tegn fra X_i og Y_j.

Hvis $x_i \neq y_j$, kan parret (x_i, y_j) ikke være en del af L, og L ligger derfor helt inden for enten X_{i-1} og Y_j eller X_i og Y_{j-1}. Den må nødvendigvis være en optimal fælles delsekvens i disse (ellers er der en bedre sekvens end L i et af disse strengepar, og derfor også i strengparret X_i og Y_j, hvilket er i modstrid med at L er optimal). Dette viser $W(i,j) = \max\{W(i-1,j), W(i,j-1)\}$, og dermed anden linie i rekursionsformlen.

Hvis $x_i = y_j$, kan parret (x_i, y_j) godt være en del af løsningen L (og er så det sidste par i L). Hvis det er, ligger resten af L helt inden for X_{i-1} og Y_{j-1}, og

må være en optimal løsning der (findes en bedre løsning der, kan en bedre løsning (end L) findes for X_i og Y_j ved at tilføje parret (x_i, y_j), hvilket er i modstrid med at L er optimal). Værdien af L er derfor $W(i-1, j-1)+w(x_i)$. Hvis derimod parret (x_i, y_j) ikke er en del af L, kan der argumenteres som ovenfor for at $W(i, j) = \max\{W(i-1, j), W(i, j-1)\}$.

Vi ved ikke om parret (x_i, y_j) er med i L eller ej, men den er én af delene, så udtrykket $\max\{W(i-1, j), W(i, j-1), W(i-1, j-1) + w(x_i)\}$ må være en overgrænse for $W(i, j)$.

Omvendt findes der fælles delsekvenser af X_i og Y_j hvis vægt er hver af de tre værdier, der tages maksimum over (disse fælles delsekvenser er henholdsvis den optimale fælles delsekvens for X_{i-1} og Y_j, den optimale fælles delsekvens for X_i og Y_{j-1}, og den optimale fælles delsekvens for X_{i-1} og Y_{j-1} med parret (x_i, y_j) tilføjet). Udtrykket er derfor også en undergrænse for $W(i, j)$, som jo er den største vægt en fælles delsekvens af X_i og Y_j har. Derfor er udtrykket præcist lig med $W(i, j)$. Dette viser den sidste linie i rekursionsformlen.

Hvis én af strengen er tomme, er den optimale delsekvens nødvendigvis tom, og har derfor vægten nul, hvilket viser første linie i rekursionsformlen.

□

Opgave 5

Spørgsmål a:

Vi ser på en knude v med et venstrebarn u og højrebarn w, begge forskellige fra NIL. Pr. definition af in-order i søgetræer er alle de søgte afstande (dvs. afstande mellem knuder og deres predecessor) i v's undertræ enten i u's undertræ, eller i w's undertræ, eller er afstanden mellem $v.key$ og $u.max$, eller er afstanden mellem $w.min$ og $v.key$. Derfor kan vi bestemme $v.maxGap$ i $O(1)$ tid som:

$$v.maxGap = \max\{u.maxGap, w.maxGap, (v.key - u.max), (w.min - v.key)\}.$$

Vi bemærker at dette stadig er korrekt hvis u's eller v's undertræ kun indeholder én nøgle, fordi deres $maxGap$ da pr. definition er sat til nul (og derfor ikke kan dominere over de andre værdier der maksimeres over, eftersom afstande altid er ikke-negative).

Vi bestemmer $v.max$ og $v.min$ i $O(1)$ tid som henholdsvis $w.max$ og $u.min$.

Hvis u eller w er NIL, fås simplere versioner af ovenstående udregning. Hvis f.eks. u er NIL, mens w ikke er, skal vi i stedet bruge

$$v.maxGap = \max\{w.maxGap, (w.min - v.key)\},$$

og $v.max$ og $v.min$ sættes til henholdsvis $w.max$ og $v.key$.

Hvis begge er NIL, skal $v.maxGap$ sættes til nul, og $v.max$ og $v.min$ sættes begge til $v.key$.

□

Spørgsmål b:

Dette følger af spørgsmål a), samt sætning 14.1 (side 346) i lærebogen.

□

Spørgsmål c:

Dette kan laves ved en rekursiv søgealgoritme, der starter i roden, og som for en besøgt knude v med venstrebarn u og højrebarn w (begge forskellige fra NIL) undersøger, hvilken af de fire værdier $(v.key - u.max)$, $(w.min - v.key)$, $u.maxGap$ og $w.maxGap$, der er lig $v.maxGap$ (mindst én af dem skal være det).

I det første tilfælde returneres $v.key$. I det næste tilfælde returneres $w.min$. I de to sidste tilfælde kalder algoritmen rekursivt sig selv på henholdsvis u og w.

Hvis enten u eller w er NIL, er der blot to af de fire cases at betragte. (Hvis både u og w er NIL, bør der returneres en fejlkode—dette sker kun når hele træet har størrelse én, og da der ikke er nogle knuder som har en predecessor, er problemet ikke veldefineret).

Der bruges $O(1)$ tid i hver knude, og alle besøgte knuder befinder sig på én sti fra roden til et blad. Da et rød-sort træ har højde $O(\log n)$, er den samlede tid $O(\log n)$.

Korrekthed (ikke krævet for fuld besvarelse) følger ved at vise (ved induktion på antal rekursive kald) en invariant som siger, at når den rekursive algoritme kaldes på en knude v, indeholder v's undertræ en knude og dennes predecessor, mellem hvilke der er den maksimale afstand i hele træet.

□