

Skriftlig Eksamen  
Algoritmer og Datastrukturer (DM02)

Institut for Matematik og Datalogi  
Syddansk Universitet, Odense

Mandag den 17. januar 2005, kl. 9–13

Løsningsforslag



## Opgave 1 (15%)

**Spørgsmål a:** Kun  $T_1$  er et rødsort træ.

Alle fire træer har sort rod og sorte blade.

For  $T_1$  gælder desuden:

— Ingen rød knude har et rødt barn.

—  $T_1$  er et søgetræ.

— For enhver knude  $v$  i  $T_1$  gælder, at alle stier fra  $v$  til et blad har det samme antal sorte knuder.

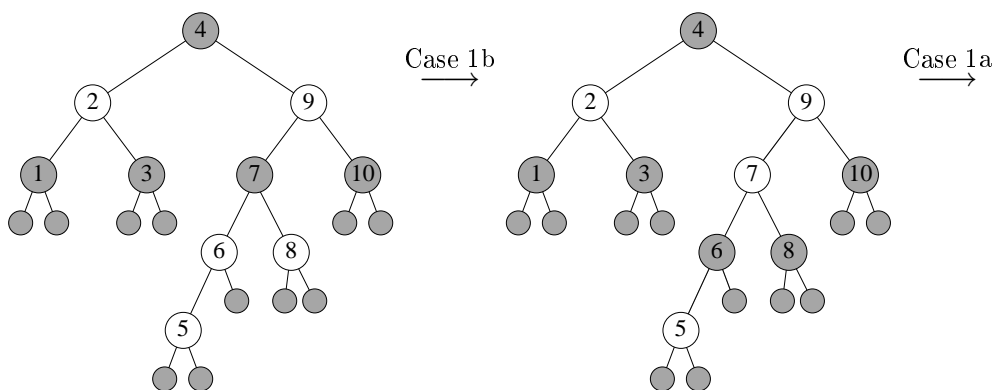
Ikke alle stier fra roden i  $T_2$  har det samme antal sorte knuder.

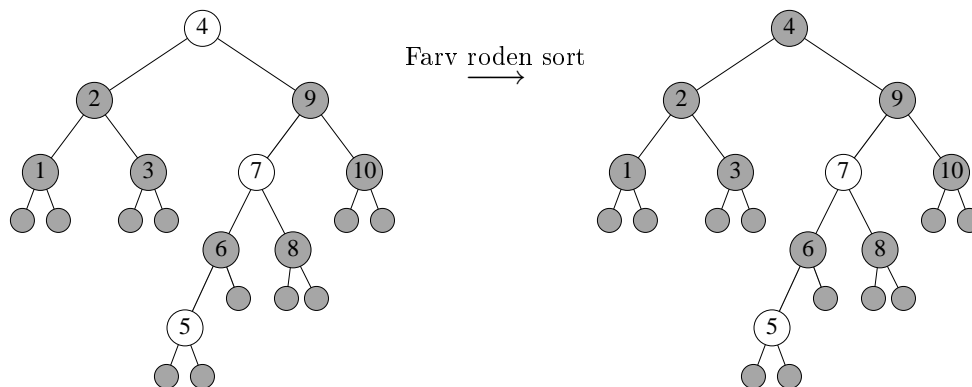
$T_3$  er ikke et søgetræ, derimod er det hob-ordnet.

I  $T_4$  er der en rød knude, som har et rødt barn. □

**Spørgsmål b:**

Vi får kun brug for omfarvninger, ingen rotationer:





□

## Opgave 2 (15%)

### Spørgsmål a:

*Basis:*  $a = 1$

$ab = 1 \cdot b = b$ , og algoritmen returnerer netop  $b$ .

*Induktionsskridt:*  $a \geq 2$

Iflg. induktionsantagelsen returnerer  $\text{MULT}(\lfloor \frac{a}{2} \rfloor, 2b)$  produktet  $\lfloor \frac{a}{2} \rfloor \cdot 2b$ .

$a$  lige:  $\lfloor \frac{a}{2} \rfloor \cdot 2b + b \cdot (a \bmod 2) = \frac{a}{2} \cdot 2b + b \cdot 0 = ab$

$a$  ulige:  $\lfloor \frac{a}{2} \rfloor \cdot 2b + b \cdot (a \bmod 2) = \frac{a-1}{2} \cdot 2b + b \cdot 1 = ab - b + b = ab$

□

### Spørgsmål b:

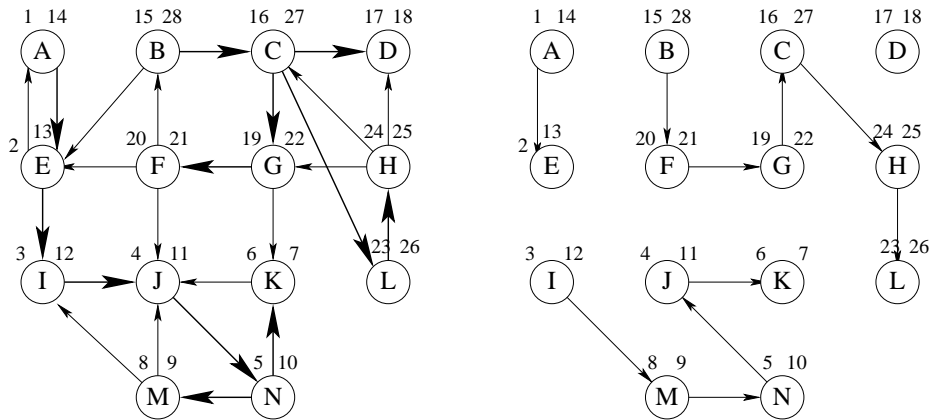
$$T(a) = \begin{cases} \Theta(1), & a = 1 \\ T(\lfloor \frac{a}{2} \rfloor) + \Theta(1), & a \geq 2 \end{cases}$$

Master-sætningen, tilfælde 2 giver  $T(a) \in \Theta(\log a)$ .

□

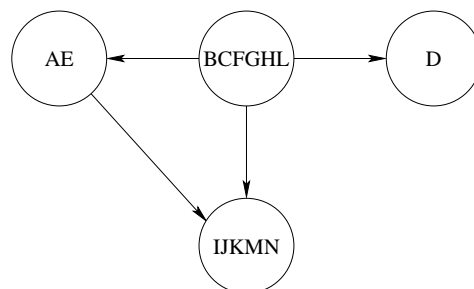
### Opgave 3 (25%)

Spørgsmål a:



□

Spørgsmål b:



□

**Spørgsmål c:** Knuderne BCFGHL, AE og IJKLMN skal optræde i denne indbyrdes rækkefølge. Knuden D skal blot optræde efter BCFGHL — dvs. der er følgende tre mulige sorteringer:

- BCFGHL, D, AE, IJKLMN
- BCFGHL, AE, D, IJKLMN
- BCFGHL, AE, IJKLMN, D

□

## Opgave 4 (25%)

Spørgsmål a:

		$j$						
		0	1	2	3	4	5	6
$i$	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0
	2	0	0	0	0	0	1	1
	3	0	0	1	1	1	1	1
	4	0	0	1	1	1	1	1
	5	0	0	1	1	1	1	2

□

**Spørgsmål b:** Hvis den ene streng er tom, har længste fælles delsekvens længde 0, og dermed er blokscoren 0.

Hvis  $s_i \neq t_j$ , må det sidste tegn i en fælles delsekvens for  $S_i$  og  $T_j$  være forskellig fra mindst en af  $s_i$  og  $t_j$ . D.v.s. vi kan se bort fra en af dem.

Hvis  $s_i = t_j$ , findes der en fælles delsekvens for  $S_i$  og  $T_j$ , som slutter med  $s_i$ . Det er dog ikke sikkert, at det giver den størst mulige blokscore at inkludere  $s_i$  i løsningen, men hvis  $s_i$  skal med, så skal det længste suffix af  $S_i$ , som også er et suffix af  $T_j$  med. Vi beregner den bedst mulige blokscore for både denne mulighed og de to muligheder, hvor vi udelukker enten  $s_i$  eller  $t_j$  fra løsningen — og vælger den bedste af de tre muligheder. På den måde får vi undersøgt alle muligheder. Hvis vi vælger muligheden, hvor  $s_i$  skal med i delsekvensen, tilføjer vi  $m_{ij}$  tegn til delsekvensen (derfor  $+m_{ij}$ ), men vi tilføjer også en ny blok (derfor  $-1$ ). □

**Spørgsmål c:** Man kan lave en rekursiv eller en iterativ algoritme. Vælger man en iterativ løsning, kunne det se sådan ud:

```

MAXBLOKSCORE( $S, T$ )
for  $i \leftarrow 0$  til  $|S|$ 
     $B(i, 0) \leftarrow 0$ 
for  $j \leftarrow 1$  til  $|T|$ 
     $B(0, j) \leftarrow 0$ 
for  $i \leftarrow 0$  til  $|S|$ 
    for  $j \leftarrow 0$  til  $|T|$ 
         $M(i, j) \leftarrow 0$ 
for  $i \leftarrow 1$  til  $|S|$ 
    for  $j \leftarrow 1$  til  $|T|$ 
         $B(i, j) \leftarrow \max \{B(i, j - 1), B(i - 1, j)\}$ 
        Hvis  $S[i] = T[j]$ 
             $M(i, j) \leftarrow M(i - 1, j - 1) + 1$ 
             $B(i, j) \leftarrow \max \{B(i, j), B(i - M(i, j), j - M(i, j)) + M(i, j) - 1\}$ 
returner  $B(|S|, |T|)$ 

```

Lad  $m = |S|$  og  $n = |T|$ . Ved at bruge tabellen  $M$  opnår vi en køretid på  $\Theta(mn)$ . Hvis man i stedet beregner  $m_{ij}$  fra bunden, hver gang man har brug for den, får man en worst-case køretid på  $\Theta(mn \min(m, n))$ .  $\square$

## Opgave 5 (20%)

**Spørgsmål a:** Man indsætter tallene et for et i et rød-sort træ. I hver knude i træet har man en tæller, som fortæller, hvor mange gange knudens element er fundet i input. På den måde kan man nøjes med  $O(\log n)$  knuder.

Når man skal indsætte et tal, checker man først, om der allerede er en knude med det tal. Hvis ja, inkrementerer man blot tælleren i den knude. Hvis nej, opretter man en knude med dette tal og en tæller, som man sætter lig 1.

Da et rød-sort træs højde er logaritmisk i antallet af knuder, giver det en højde på  $O(\log(\log n))$ . D.v.s. de  $n$  indsættelser tager sammenlagt  $O(n \log(\log n))$  tid.

Derefter laver man blot et inorder-gennemløb af træet og udskriver tallene, efterhånden som man møder dem. I hver knude aflæser man tælleren for at vide, hvor mange gange tallet i knuden skal udskrives, inden man går videre til næste knude. Et inorder-gennemløb tager tid  $\Theta(V)$ , hvor  $V$  er antallet af knuder i træet, plus den tid, der bruges i knuderne. Da der skal udskrives  $n$  tal, bruges der sammenlagt  $\Theta(n)$  tid i knuderne. Da  $V \in O(\log n)$  bruges der alt i alt  $\Theta(n)$  tid på at udskrive tallene.  $\square$