DMA uge 6, løsninger til nogle af mandagens opgaver

Nogle af disse løsninger er skrevet ned meget kortfattet. Her må man selv lave "mellemregningerne". Hensigten er at man kan tjekke at man ikke er helt ved siden af.

Der kan også have sneget sig fejl ind. Hvis du mener der er en fejl, så rapportér det på Discussions på kursussiden, så det kan blive afklaret og andre kan få glæde af det, hvis du har ret.

1 (a) Det er kun træet (c) der er et binært søgetræ. De andre overholder ikke søgetræsinvarianten.

2 (a) Her ses hukommelsen:

0	1	2	3	4	5					10						15							20				
			20	/	7	13	21	/	/	3			16	37	3	/							3	/				

25					30					35					40					45						
/	27	1	/	23	37							8	27	41	13	14	/	/	37							

2 (b) Her ses hukommelsen efter at en knude med nøgleværdi 12 er blevet indsat:

0	1	2	3	4	5					10						15							20				
			20	/	7	13	21	/	/	3			16	37	3	/							3	/				

25					30					35					40					45						
/	27	1	/	23	37							8	27	41	13	14	41	/	37		12	/	/	41		

3 (a) En længste er A-B-E-K-J.

En korteste er A-B-D.

3 (b)
```
KortesteSti(x)
    if x.left == NIL or x.right == NIL
      return 0
    return 1 + minimum(KortesteSti(x.left), KortesteSti(x.right) )
```

Køretiden er $\Theta(n)$, hvor n er antallet af knuder, idet algoritmen besøger hver knude én gang.

Vi kan bevise at algoritmen er korrekt vha. stærk induktion over n. Antag at hvis deltræet T_x som hænger under x (dvs. deltræet som består af knuden x og alle efterkommere af x) har størrelse højst $n-1$, så returnerer `KortesteSti(x)` længden af den korteste vej fra x til et blad som er en efterkommer til x. Betragt nu et tilfælde hvor T_x har n knuder. Hvis x ikke har et venstre barn eller ikke har et højre barn, så virker algoritmen, fordi den returnerer 0 som den skal. Ellers kaldes den rekursivt på højre og venstre barn, og der returneres én mere end minimum af de to værdier. Induktionsantagelsen giver at algoritmen finder længderne af de korteste veje fra det venstre og højre barn og til blade som er deres efterkommere. Derfor fås det korrekte resultat når vi tager minimum af disse to værdier og lægger 1 til.

4 (a) Nedenstående algoritme løser problemet:

```
Blade(x)
  if x == NIL
    return 0
  if x.left == NIL and x.right == NIL
    return 1
  return Blade(x.left) + Blade(x.right)
```

4 (b) Nedenstående algoritme løser problemet:

```
Height(x)
  if x == NIL or (x.left == NIL and x.right == NIL)
    return 0
  return 1 + maximum(Height(x.left), Height(x.right))
```

5 Nedenstående algoritme løser problemet:

```
SearchTreeInvariant(x)
  if x.left != NIL and (x.left > x.key or not SearchTreeInvariant(x.left))
    return false
  if x.right != NIL and (x.right < x.key or not SearchTreeInvariant(x.right))
    return false
  return true
```