Några svar till
TDDC70/91 Datastrukturer och algoritmer

2011-08-24

Följande är lösningsskisser och svar till uppgifterna på tentan. Lösningarna som ges här ska bara ses som vägledning och är oftast inte tillräckliga som svar på tentan.

1. (a) $\log n, n, n \log n, n^2, n^2 + \log n, n - n^3 + 7n^5, 2^n, n!$

   (b) Låt $T(n)$ vara tiden för att beräkna $F(n)$. $F(0)$ och $F(1)$ beräknas i konstant tid, t_0 resp. t_1. För $n > 1$ får vi $T(n) = T(n-1) + T(n-2) + c$ där c är tiden för att addera två naturliga tal. Alltså, för alla $n \geq 2$ har vi: $T(n) > 2T(n-2)$. För udda $n(= 2m+1)$ har vi alltså $T(n) > min(t_0, t_1)2^m = min(t_0, t_1)2^{(n-1)/2}$. På samma sätt, för jämna $n > 1$, får vi $T(n) > min(t_0, t_1)2^{(n-2)/2}$. Alltså gäller $T(n) \in \Omega(2^{(n/2)})$.

   (c) Huvudloopen för att beräkna $F(n)$ för $n > 1$ skulle kunna ha formen

   for i **from** 2 **to** n
   $temp \leftarrow prev$;
   $prev \leftarrow curr$;
   $curr \leftarrow curr + temp$;

   där *curr* har värdet $F(n)$ när loopen terminerar. Före loopen sätts *curr* till 1 och *prev* till 0. Loopen använder tre operationer som tar konstant tid vilka exekveras $n-1$ gånger vilket ger tidskomplexitet $\Theta(n)$.

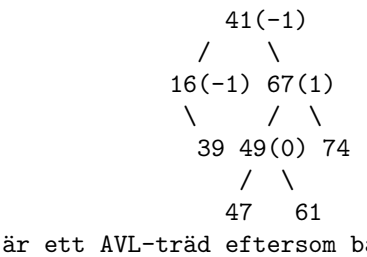
2. - Söndra och härska. Merge sort.
   - Dynamisk programmering. Floyd-Warshall.
   - Giriga algoritmer. Dijkstras algoritm.

3. Vi skulle kunna använda den föreslagna algoritmen iterativt på en given fil för att komprimera den ner till att bara använda en bit, vilket faller på sin egen orimlighet.

4. ```
   x <- S.pop()
   if x < S.top() then
     x <- S.pop()
   ```

   Notera att om det största heltalet är det första eller andra elementet i S så lagras det i x. Alltså lagrar x det största elementet med sannolikhet 2/3.

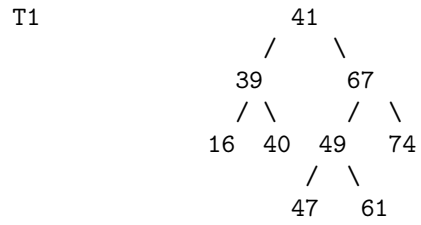
5. (a) - `Trädet`



   `är ett AVL-träd eftersom balansfaktorn för varje nod är 0, 1 eller -1.`
   ```

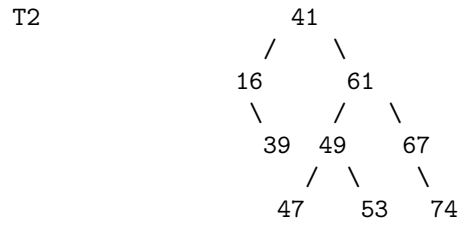
- Insert(40) slutresultat:

T1



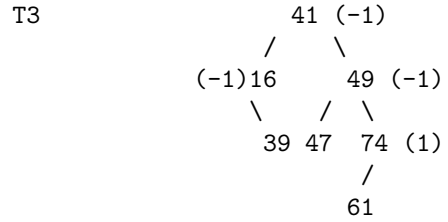
Insert(53) slutresult:

T2



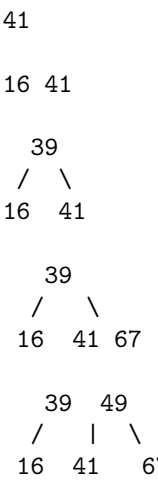
Delete(67) slutresultat om efterföljaren i inorder används som ersättare:

T3



En alternativ lösning är att ersätta 67 med dess föregångare i inorder.
Då behövs ingen ombalansering.

(b)

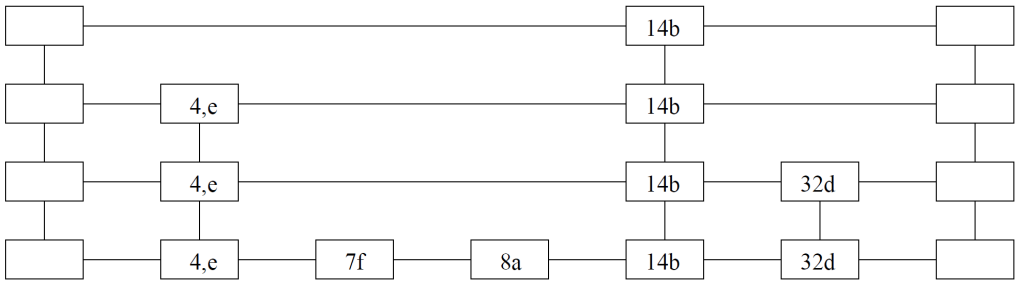


```
    39   49
   /   |    \
 16 41 47   67
```

6. (a) Hashtabellen:

0	0	0	14b 0	14b 0	14b 0	14b 0	14b 0
1	15c 0	15c 0	15c 0	15c 1	15c 1	15c 1	15c 1
2	0	0	0	0	0	0	0
3	0	8a 0	8a 0	8a 0	8a 0	8a 0	8a 0
4	0	0	0	0	32d 0	32d 0	32d 0
5	0	0	0	0	0	4e 0	4e 0
6	0	0	0	0	0	0	7f 0

(b) Skiplistan:



7. (a) Insertion Sort

```
7 5 9 6 8
5 7 9 6 8
5 6 7 9 8
5 6 7 8 9
```

Insertion Sort är stabil.

(b) Quick Sort

```
7 5 9 6 8        partitionering startar: pivot 8
7 5 6 9 8
7 5 6 8 9        partitionering slutar
7 5 6 8 9
7 5 6 8 9        partitionering av 7 5 6 startar:  pivot 6
5 7 6 8 9
5 6 7 8 9
```

(c) Heap Sort; träden visas inte

```
7 5 9 6 8      Heapification startar
9 5 7 6 8
9 8 7 6 5      max-heap skapad
5 8 7 6 9      maximalt element flyttat från heapen
8 5 7 6 9      re-heapification
8 6 7 5 9
5 6 7 8 9      maximalt element flyttat från heapen
7 6 5 8 9      re-heapification
5 6 7 8 9      maximalt element flyttat från heapen
6 5 7 8 9      re-heapification
5 6 7 8 9
```

8. Lösningen använder en teknik som påminner om merge sort.

   Find-2-Numbers(S, x, n)
   S is a set of n numbers.
   x is the sum we are aiming for.

   (a) $A \leftarrow MergeSort(S)$
   (b) $left \leftarrow 0$
   (c) $right \leftarrow (n-1)$
   (d) while $(left < right)$
   (e) if $((A[left] + A[right]) = x)$
   (f) then return TRUE.
   (g) else if $((A[left] + A[right]) < x)$
   (h) $left \leftarrow (left + 1)$
   (i) else
   (j) $right \leftarrow (right - 1)$
   (k) return FALSE.

   Steg (a) tar $O(n \log n)$ tid och resten av stegen tar $O(n)$ tid. Algoritmen evekverar i tid $O(n \log n)$.

   Ett alternativt sätta att göra detta skulle kunna vara att sortera arrayen och att sedan, för varje y i arrayen, söka efter $(x - y)$. Varje sökningen skulle ta tid $O((\log n)$ och algoritmen totalt $O(n \log n)$ tid.

9. (a) A,L,B,D,C,E,F. Det spännande trädet har bågarna: (A,L), (L,B), (B,D), (D,C), (C,E), (E,F).
   (b) A,L,B,C,F,D,E. Det spännande trädet har bågarna: (A,L), (A,B), (A,C), (A,F), (L,D), (C,E).