Några svar till
TDDC70/91 Datastrukturer och algoritmer

2013-08-27

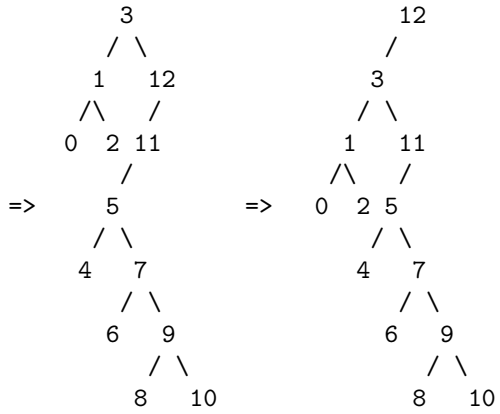
Följande är lösningsskisser och svar till uppgifterna på tentan. Lösningarna som ges här ska bara ses som vägledning och är oftast inte tillräckliga som svar på tentan.

1. (a) **Falskt**. Ett motexempel: $f(n) = g(n) = n^2$ och $h(n) = n^3$.

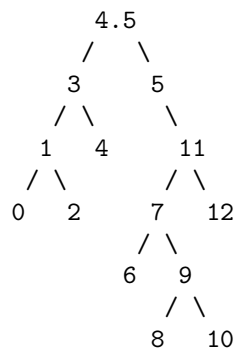
   (b) **Sant**. Enligt definitionen av Ω behöver vi hitta en reell konstant $c > 0$ och en heltalskonstant $n_0 \geq 1$ sådana att $n \log_2(n) \geq cn$ för $n \geq n_0$. Valet $c = 1$ och $n_0 = 2$ visar att $n \log_2(n) \geq cn$ för $n \geq n_0$ eftersom $\log_2(n) \geq 1$ i det intervallet.

   (c) **Sant**. $3^{\log(n^2)} = 3^{2\log(n)} = 9^{\log(n)} = n^{\log 9}$ som är polynomiskt.

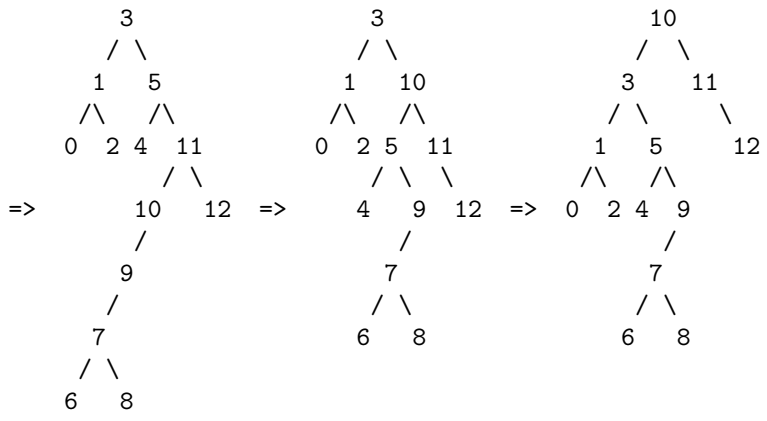
2. (a)



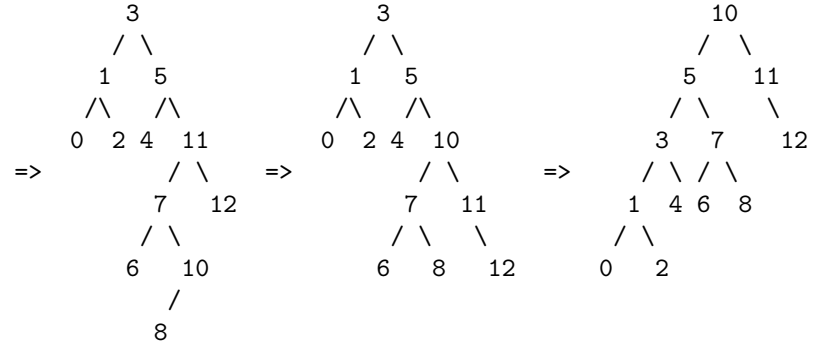
   (b)



(c)



(d)



3. (a)
```
5
5 3
5
5 2
5 2 8
5 2
5
5 9
5 9 1
5 9
5 9 7
5 9 7 6
5 9 7
5 9
5 9 4
5 9
5
```

(b)
```
5
5 3
3
3 2
3 2 8
2 8
8
8 9
8 9 1
9 1
9 1 7
9 1 7 6
```

```
1 7 6
7 6
7 6 4
6 4
4
```

Algorithm 0.1: REVERSE(Q)

(c)
```
S ← en tom stack
while ! Q.isEmpty()
  do S.push(Q.dequeue())
while !S.isEmpty()
  do Q.enqueue(S.pop())
```

4. Använd en skiplista för att implementera ADT DICTIONARY. Varje nod på bottennivån pekar på en länkad lista med element med samma nyckel. På så sätt fungerar alla uppdateringsoperationer i $O(\log r)$ tid, medan `getAll` exekverar i tid $O(\log r + s)$.

5. (a) Om inexakta sökningar behövs. Om vi, till exempel, vill hitta det element med nyckel mindre än eller lika med 5 som är närmast 5. Hashtabeller kan bara utföra exakta sökningar.

   (b) Om varje enskild operation måste ha körtid $O(\log n)$. Eller om de flesta operationerna är `find()` och accessmönstret är likformigt fördelat.

   (c) Om minnesanvändning är absolut högst prioriterat.

6. (a)

0	1	2	3	4	5	6	7	8	9	10	11	12
-	Y	X	H	G	T	C	A	F	B	Q	R	-

   (b) Endast den den slutliga arrayrepresentation redovisas här.

0	1	2	3	4	5	6	7	8	9	10	11	12
-	Y	X	*P	G	T	*H	A	F	B	Q	R	*C

   (c) `H I J K L M N`
   Nyckeln är $\leq$ `N` eftersom den är ett barn till `N` i ursprungsheapen och $\geq$ `H` eftersom den är förälder till `H` i heapen efer anropet till `removeMax`.

7. (a)

0	1	2	3	4	5	6
G	B	D	A	C	E	F

   (b) Endast I är möjlig.

   - I är resultatet av insättning av nycklarna i ordning B D G A C E G.
   - II är inte möjlig. Första nyckeln som sätts in hamnar på en plats i tabellen som motsvarar dess hashvärde. Men ingen nyckel har den egenskapen.
   - III är inte möjlig. Både A och F hamnar på platser i tabellen som motsvarar deras hashvärden, så vi kan anta att de sattes in som första och andra element. Det betyder att den tredje nyckeln som sattes in också måste ha hamnat på en plats som motsvaras av dess hashvärde. Men inga nycklar (förutom A och F) har denna egenskap.

8. En möjlig lösning: Använd DFS för att identifiera bågarna som behöver tas bort för att göra grafen acyklisk. Det totala antalet "back"-bågar är det minsta antalet bågar som behöver tas bort för att göra grafen acyklisk. Algoritmen använder $O(|V| + |E|)$ tid.