Några svar till
TDDC70/91 Datastrukturer och algoritmer

2012-01-11

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) **Möjligt**. Använd insertion sort. 1000 är en konstant, så detta är "nästan sorterat data".
   (b) **Omöjligt**. Måste titta på allt data $\Rightarrow \Omega(n)$ tid.
   (c) **Omöjligt**. Jämförelsebaserad sortering gör $\Omega(n \log n)$ jämförelser i värsta fallet.

2. $O(mn)$ resp. $O(m+n)$.

3. (a)
```
tl (root r)
  if (r == null)
    return 0
  else
    return tl(r, 0)
}

tl (node r, len x) {
  s1 = 0
  s2 = 0
  if (r.hasLeft)
    s1 = tl(r.left, x+1)
  if (r.hasRight)
    s2 = tl(r.right, x+1)
  return x + s1 + s2
}
```
   Algoritmen har linjär tidskomplexitet.

   (b)
```
     20                    20
    /   \                 /   \
   5     30              10     30
    \   /  \            /  \   /  \
     15 25   35         5   15 25   35
    /
  10
```

   (c) Om a eller b finns är svaret ja. Annars? Att söka efter $a+1, a+2, \ldots, b-1$ blir väldigt långsamt. (Beror på värdena, inte *antalet* noder i trädet.) Om sökning efter a och b terminerar i a_0 och b_0 ($a_0 \neq b_0$), så finns specifik vikt mellan a och b: förfadern till a_0 och b_0 är en av dem. Om båda misslyckade sökningarna terminerade i samma löv är svaret nej. Tidskomplexiteten blir $O(\log n)$.

4. (a)

0	1	2	3	4	5	6
7	21	42	10	22		

(b)

0	1	2	3	4	5	6
7	10		42	21	22	

5. Algoritmen liknar binärsökning.

```
public static int max (int[] a, int lo, int hi) {
   if (hi == lo) return a[hi];
   int mid = lo + (hi - lo) / 2;
   if      (a[mid] < a[mid + 1]) return max(a, mid + 1, hi);
   else if (a[mid] > a[mid + 1]) return max(a, lo, mid);
   else                          return a[mid];
}
```

6. (a) a d f i h j b c e g
   (b) a d f h j i b c e g
   (c) g e c b a d i f h j
   (d) Om G är en oriktad enkel graf med $|V(G)| \geq 2$ så finns det två noder med samma antal grannar i grafen.

   Antag att G ej har isolerad nod. Detta medför att varje nod har $[1, n-1]$ grannar. Men det finns n noder, så då måste två noder ha samma antal grannar enligt duvslagsprincipen. Om G har en isolerad nod finns det $n-1$ noder med $[1, n-2]$ grannar...