Några svar till
TDDC70/91 Datastrukturer och algoritmer

2012-10-23

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) **Sant**. Låt $f(n) = 1$ när n är jämnt och n när n är udda. Låt också $g(n) = 1$ när n är udda och n när n är jämnt. Då oscillerar kvoten $f(n)/g(n)$ mellan 0 och ∞ när n blir stort.

   (b) **Falskt**.

$$\frac{n!}{2^n} = \frac{n \cdot (n-1) \cdot \ldots \cdot 2 \cdot 1}{2 \cdot 2 \cdot \ldots \cdot 2 \cdot 2} \geq \frac{3 \cdot 3 \cdot 3 \cdot \ldots \cdot 3 \cdot 3}{2 \cdot 2 \cdot 2 \cdot \ldots \cdot 2 \cdot 2} \cdot 1 \cdot \frac{1}{2} = \frac{1}{2} \left(\frac{3}{2}\right)^{n-2}$$

   (c) **Falskt**. I den kompletta grafen K_n finns $n(n-1)/2 \in \Theta(n^2)$ bågar, vilket inte ryms i $O(n \log n)$.

2. 1:a, 2:d, 3:e, 4:c, 5:f, 6:b

3. (a)

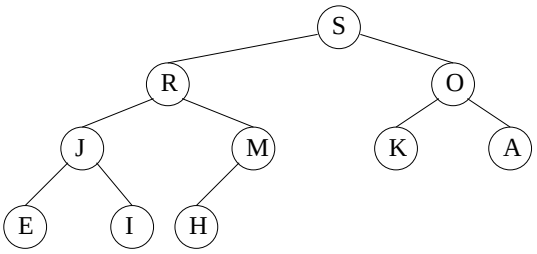
0	1	2	3	4	5	6
G	D	B	F	E	A	C

   (b)

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

4. (a) K M R

   (b)



5. - `sample()`: Välj ett slumpmässigt arrayindex `r` (mellan 1 och N) och returnera nyckeln `a[r]`.

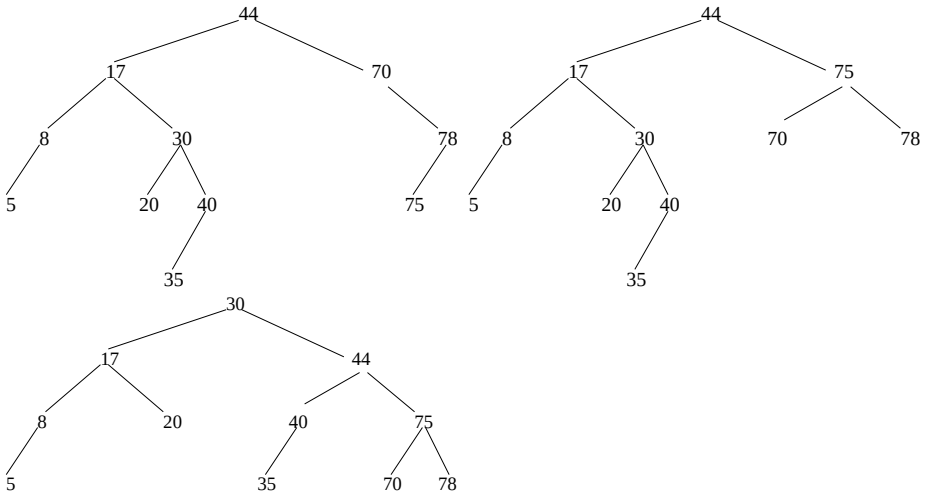
```
public Key sample() {
   int r = 1 + Random.uniform(N);
   return a[r];
}
```

   - `removeRandom()`:
     - Välj ett slumpmässigt arrayindex `r` (mellan 1 och N) och spara undan nyckeln `a[r]` som ska returneras.

– Byt plats på `a[r]` och `a[N]` och minska `N` med ett.
– Återställ heapordningen genom att utföra anropen `downHeap(r)` och `upHeap(R)` för att fixa till att heapordningen ev. inte är uppfylld kring `r`. Notera att `a[N]` i ursprungsheapen inte behöver ha varit den största nyckeln i heapen, så anropet till `upHeap(R)` är nödvändigt.

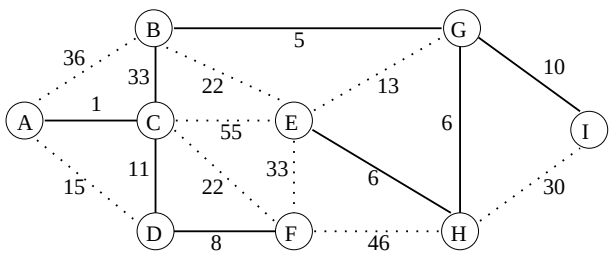
```
public Key removeRandom() {
   int r = 1 + Random.uniform(N);
   Key key = a[r];
   swap(r, N--);
   downHeap(r);    // om a[N] var för stor
   upHeap(r);      // om a[N] var för liten
   a[N+1] = null;  // städa upp
   return key;
}
```

6. (a) T är ett binärt sökträd och uppfyller balanskriterierna ett AVL-träd måste uppfylla.
   (b) Endast de mellanliggande träden samt slutresultatet visas.



7. Idén är att om det finns *någon* trippel $x < y < z$ som i uppgiftslydelsen så är det också sant att $\min_{T_B} < y < \max_{T_B}$. Alltså letar vi först upp det minsta och det största elementet i T_B och söker sedan efter dessa i T_A. Beroende på resultatet av dessa sökningar kan vi avgöra om vi ska returnera `true` eller `false`. Det blir en del specialfall att hålla reda på, men om vi använder t.ex. AVL-träd klarar vi oss med $O(\log n)$ tid för att leta reda på min, max och de två modifierade sökningarna.

8. (a) A B G C D F E H I
   (b) Endast I och II. Anropsstacken innehåller en sekvens av noder på en riktad stig från s till nuvarande nod (med s på botten och nuvarande nod på toppen).
   (c)



Nästa nod	Uppskattat avst. från startnod								Båge från molnet till nästa nod
	B	C	D	E	F	G	H	I	
A	36	1	15	∞	∞	∞	∞	∞	—
C	34	1	12	56	23	∞	∞	∞	A-C
D	34	1	12	56	20	∞	∞	∞	C-D
F	34	1	12	53	20	∞	66	∞	D-F
B	34	1	12	53	20	39	66	∞	C-B
G	34	1	12	52	20	39	45	49	B-G
H	34	1	12	51	20	39	45	49	G-H
I	34	1	12	51	20	39	45	49	G-I
E	34	1	12	51	20	39	45	49	H-E