KTH CSC, Magnus Rosell

2D1320 Tilda: lösningsskiss till tenta 23 oktober 2006

1. *Sorterade CD:s.*

a. Kan ej ta ut annat än i toppen. Kan ej flytta saker innuti.

b. Quicksort.

c. linjärsökning, binärsökning
binärsökning, $O(log(n))$, 15, dela upprepade gånger i två tills endast en återstår

d. Distributionsräkning. Se kurslitteraturen.

e. *Rekursion*: Totala antalet låtar är antalet låtar på den översta CD:n plus antalet låtar i resten av stacken.
Basfall: I en tom stack finns inga låtar.

f. Se kurslitteraturen. Tex:

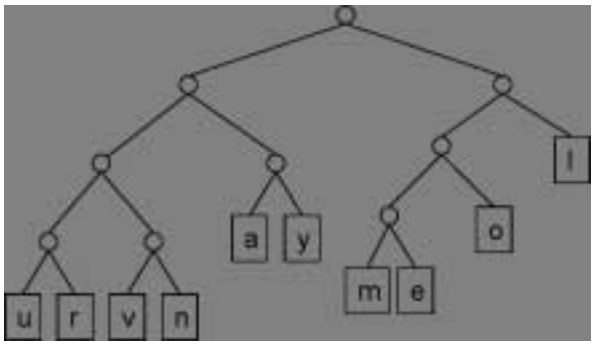
```
def compare(annancd):
    # returnerar 1, 0, -1
    # om denna cd är större än, lika med, mindre än
    # annancd (definerat tex som i a-uppgiften)

def sattutgivningsar(ar):
    # sätter utgivningsåret på CD:n

def latlista():
    # returnerar en kö med CD:ns låtar
```

2. *"lay all your love on me"* Här är en av alla tänkbara huffmankoder:

bokstav	antal	kod
l	4	11
a	2	010
y	2	011
o	3	101
u	1	0000
r	1	0001
v	1	0010
e	2	1001
n	1	0011
m	1	1000



Och här är kodningen av önskan med mellanslag mellan orden för tyglighets skull:

```
11010011 0101111 01110100000001
1110100101001 1010011 10001001
```

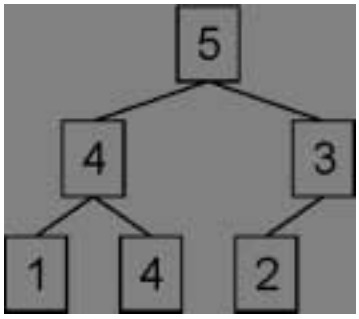
3. *Greatest Hits.*

```
[Gg]reatest_[Hh]its?
```

4. *Bästa spellistan.*

a. En *trappa* (*heap*) kan visualiseras som ett binärt träd och implementeras som en vektor. Här är vektorn och motsvarande träd efter det att man stoppat in alla talen i en max-heap:

-	5	4	3	1	4	2	...



En min-heap med dessa tal blir mycket enklare.

b. Låten hamnar först igen. Finns det fler med samma betyg hamnar den "sist" bland dessa. Detta leder till att bara låtar med högsta betyget spelas om och om igen.

c.
```
def f(betyg, antal):
    return betyg - antal
```

En låt med högt betyg ska ha hög prioritet. Låtars prioritet ska sjunka med antalet gånger de spelas eftersom man annars tröttnar på dem. Detta löser också problemet i b-uppgiften.

5. *Influenser.*

a. Skapa en hashtabell med artist/band som nyckel. För varje artist sparas en tom kö. I den ska de artister som blivit inspirerade av den sparas.

Konstruktionsalgoritm:

För alla artister i den ursprungliga datastrukturen:

* Kalla artisten *influerad*. (Tänk "Rolling Stones".)
* Gör `influenser`(*influerad*). Kalla den returnerade kön k1.
* För alla artister i k1:
  - Kalla artisten *influens*. (Tänk "Muddy Waters".)
  - Hämta kön från hashtabellen med nyckeln *influens*. Kalla den *k2*.
  - Lägg in *influerad* i k2.

Användning: När `infuerat`-funktionen körs returnerar den (en kopia av) kön för den artisten i hashtabellen.

b. Några kommentarer:

* Man kan tänka sig ett problemträd där Muddy Waters är stamfar. *influerat*-funktionen kan användas som *makechildren*-funktion.
* Djupet-förstsökning (DFS) eller bredden-förstsökning (BFS). Spara artister i noder med förälderpekare. Dessa noder läggs i stacken respektive kön.
  - DFS - spar den just nu längsta kedjan. När stacken är tom har man längsta kedjan.
  - BFS - den sista artisten i kön är den som ligger längst från Muddy Waters.
* Det får inte vara några cirklar i influenskedjan, dvs inga dumbarn, men det fungerar lite annorlunda i denna uppgift. Man kan inte ha *ett* dumbarnsträd eller *en* dumbarnshashtabell eftersom artisterna kan förekomma på flera ställen i problemträdet - flera artister kan ju ha Muddy Waters som tex förälders förälder i problemträdet. Lösningen är att kontrollera nya barn mot alla i kedjan man förlänger.
* Skriv ut kedjan - följ förälderpekarna.

6. *Poptextssyntax.*

a. `<pop>::=<ord><pop>|<yeah><pop>|<ord>|<yeah>`

b.
```
<komppop>::=<del><komppop>|<del>
<del>::=<pop>|<tal>[<pop>]
<pop>::=<ord><pop>|<yeah><pop>|<ord>|<yeah>
<tal>::=2|3|...
```

7. *Sorterade CD:s.*

a. $O(n)$.

En tillräcklig analys är följande: Man passerar alla tal med "ett pekfinger" en gång och gör ett begränsat antal operationer med varje tal. Varje tal jämförs med skiljevärdet (pivotelementet) - n jämförelser.

Noggrannare gäller också: I värsta fall måste alla element byta plats (alla damer finns till höger och alla herrar till vänster). Då måste $n/2$ byten göras.

b. 5, $\sqrt{n}$, $3n$, $0.004n^2$, 7^{n^2}, snabbast växande sist.

c-e. Ja.