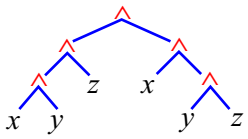
LÖSNINGAR, Algoritmik 16 OKT 1998

1 a) Rita ett binärt träd för $((x^y)^z)^{x^{(y^z)}}$

b) Räkna upp det binära trädets noder i *pre–*, *post–* och *inordning*.

LÖSNING

a)



b) *pre* $\wedge\wedge\wedge xyz\wedge x\wedge yz$

post $xy\wedge z\wedge xyz\wedge\wedge\wedge$

in $x\wedge y\wedge z\wedge x\wedge y\wedge z$

2 Vad finns i resultatshögen `r` när vidstående program har stannat, om det körs igång på

```
r ← x
Subtrahera r från y i u
Sålänge u ≠ 0{Öka r
              Minska u}
Subtrahera r från z i u
Sålänge u ≠ 0{Öka r
              Minska u}
```

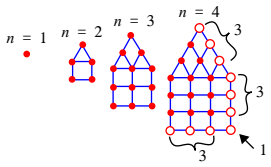
a) `x=1,y=2,z=3`,

b) `x,y,z` med godtyckligt innehåll.

LÖSNING a) `x=1,y=2,z=3` ger `r=3`.

b) Det största av `x`, `y`, `z`. MOTIVERING: Om `x` är störst kommer ingen av slingorna att köras, varför `r` (som från början innehåller detsamma som `x`) kommer att passera oförändrad genom resten av programmet. Om istället `y` är störst kommer den första av slingorna att köras, och därvid kommer `r` (som från början innehåller detsamma som `x`) att fyllas på med skillnaden mellan `y` och `x`. Något som gör att `r` blir detsamma som `y`. (Den andra slingan kommer inte att köras.) Till sist, om `z` är störst kommer den senare slingan eller båda slingorna att köras, vilket gör att `r` (som från början innehåller detsamma som `x`) kommer att fyllas på först med skillnaden mellan `y` och `x`, sedan med skillnaden mellan `z` och `x`. Dessa två påfyllningar gör att `r` blir detsamma som `z`.

3 Konstruera en rekursiv funktion som returnerar antalet noder i den n:te femhörningsgrafen byggd enligt skissen.



LÖSNING Skillnaden mellan den n:te och den $n+1$:ta femhörningen är $3n+1$ stycken noder (se skissen till vänster). Härav,

$$\begin{cases} AntalNoder(0) = 0 \\ AntalNoder(Öka(n)) = Add(AntalNoder(n), Öka(Mult(Tre(n), n))) \end{cases}$$

4 Konstruera en rekursiv funktion $TreProdukt?(x)$ som returnerar 1 om x är en produkt av tre på varandra följande naturliga tal (som t ex $7 \cdot 8 \cdot 9$), och som returnerar 0 annars.

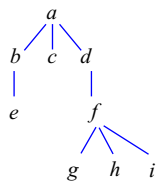
LÖSNING

$$TreProdukt?(x) = h(x, 0)$$

$$h(x, y) = Om(Mindre(TreProdukt(y), x), h(x, Öka(y)), Lika(x, TreProdukt(y)))$$

där $TreProdukt(y) = Mult(y, Mult(Öka(y), Öka(Öka(y))))$.

5 Konstruera en funktion som för ett godtyckligt träd returnerar en atomär lista innehållande varje nod från trädet som är äldst i en syskonskara. T ex skall funktionen returnera en lista som innehåller b, e, f, g och inget annat för trädet bredvid.



LÖSNING

$$Äldst\_i\_trädet([nod|S]) = Äldst\_i\_skogen(S)$$

$$Äldst\_i\_skogen([\,]) = [\,]$$

$$Äldst\_i\_skogen([T|S]) = FogaSamman(FogaIn(Första(T), Äldst\_i\_trädet(T)), Äldst\_i\_resten\_av\_skogen(S))$$

$$Äldst\_i\_resten\_av\_skogen([\,]) = [\,]$$

$$Äldst\_i\_resten\_av\_skogen([T|S]) = FogaSamman(Äldst\_i\_trädet(T), Äldst\_i\_resten\_av\_skogen(S))$$

6 Tillverka en listfunktion vars input är en lista av listor av tal och som

a) sorterar inputlistan m.a.p. dess noders listlängd, från kortare till längre.

b) sorterar inputlistan efter dess noders listlängd (från kortare till längre), och dessutom sorterar inputlistans varje nod (som ju också är en lista) från mindre till större.

LÖSNING Idé: Om hela listan förutom dess inledande nod är sorterad, så behöver man bara sticka in den nämnda noden på rätt plats!

a)
$$\begin{cases} NodLängdSortera([\,]) = [\,] \\ NodLängdSortera([nod|L]) = StickIn(nod, NodLängdSortera(L)) \end{cases}$$

$$\begin{cases} StickIn(X, [\,]) = [X] \\ StickIn(X, [Y|Z]) = Om(Mindre(Längd(X), Längd(Y)), \\ \qquad\qquad\qquad\qquad [X|[Y|Z]], \\ \qquad\qquad\qquad\qquad [Y|StickIn(X, Z)]) \end{cases}$$

b)
$$\begin{cases} TotalSortera([\,]) = [\,] \\ TotalSortera([nod|L]) = StickIn(Sortera(nod), TotalSortera(L)) \end{cases}$$

7 S.k. binär sökning efter ett visst element x i en sorterad lista L går till så att man undersöker om L:s mittersta elementet är lika med eller större än x. Om likhet råder, så avslutas sökningen. Annars fortsätter man att leta efter det sökta elementet i ena halvan av L. Man kan nämligen utesluta att det sökta elementet finns i vänster (höger) halva om L är sorterad från mindre (större) till större (mindre).

a) Konstruera funktionen *HögerHalva*(L) som returnerar höger "halva" av listan L, så att t ex
HögerHalva([2 3 5 6]) = [5 6], och
HögerHalva([2 3 5 6 9]) = [5 6 9].

b) Implementera binär sökning i form av en listfunktion $Sök(x, L)$ som returnerar 1 om x finns i listan L, och 0 annars. Listan L antas vara sorterad från mindre till större.

LÖSNING Det är naturligt att beräkna vänster- och höger halva i samma veva. (För övrigt behövs ju vänster halva i b)-uppgiften.) I min lösning nedanför använder jag en hjälpfunktion h som i starten (dvs när någon av funktionerna *VänsterHalva* eller *HögerHalva* anropar h) har den givna listan i sitt vänstra argument och tom lista i sitt högra argumentet. Under rekursionen flyttas den ena noden efter den andra från h:s vänstra argument över till h:s högra. Och när rekursionen bottnar returneras en lista med de två argumenten såsom listans noder. Här är en provkörning av h:
$h([a\ b\ c\ d], [\,]) = h([a\ b\ c], [d]) = h([a\ b], [c\ d]) = [[a\ b]\ [c\ d]]$

a)
$$VänsterHalva(L) = Första(h(L, [\,]))$$
$$HögerHalva(L) = Sista(h(L, [\,]))$$
$$h(X, Y) = Om(Större(Längd(X), Längd(Y)), \\ \qquad h(UtomSista(X), [Sista(X)|Y]), \\ \qquad [X\ \ Y])$$

b)
$$Sök(x, [\,]) = 0$$
$$Sök(x, L) = Om(Lika(x, Första(HögerHalva(L))), \\ \qquad 1, \\ \qquad Om(Större(x, Första(HögerHalva(L))), \\ \qquad\quad Sök(x, UtomFörsta(HögerHalva(L))), \\ \qquad\quad Sök(x, VänsterHalva(L))))$$