SI-möte #6, Programkonstruktion och datastrukturer

Elias Castegren & Kristiina Ausmees
elca7381@student.uu.se ‖ krau6498@student.uu.se

7 december 2011

Begrepp

i) Vad menas med att en funktion är asymptotiskt större än en annan? Vad menas med att den polynomiellt större?

ii) Vad är Master Theorem och när och hur använder man det? Vad är de tre olika fallen i satsen?

iii) Vad menas med ett *binärt sökträd*? Hur sätter man in, respektive tar bort noder ur ett sådant träd?

iv) Vad är höjden av ett binärt sökträd i sämsta och bästa fall? Hur ser trädet ut i de olika fallen?

v) Vad menas med en traversering? Vad är skillnaden på pre order-, in order- och post order-traverseringar? Vad är resultatet av en in order-traversering av ett binärt sökträd?

vi) Vad är ett *quadträd*? När kan man vilja använda ett "kvadrärt" sökträd?

Övningar

1.
Lös följande rekursioner med hjälp av Master Theorem. Ange först värdena på a, b och $f(n)$. Om MT inte är applicerbar för någon rekursion, ange varför (du behöver då inte heller lösa rekursionen).

i) $T(n) = 2T(\frac{n}{2}) + n$ *ii*) $T(n) = 4T(\frac{n}{2}) + n^2 \cdot \lg n$ *iii*) $T(n) = 10T(\frac{n}{3}) + 2n^2$

iv) $T(n) = T(\frac{n}{2}) + n$ *v*) $T(n) = 2^n \cdot T(\frac{n}{2}) + n^4$

2.
Sätt in följande element (i given ordning) i ett tomt binärt sökträd:

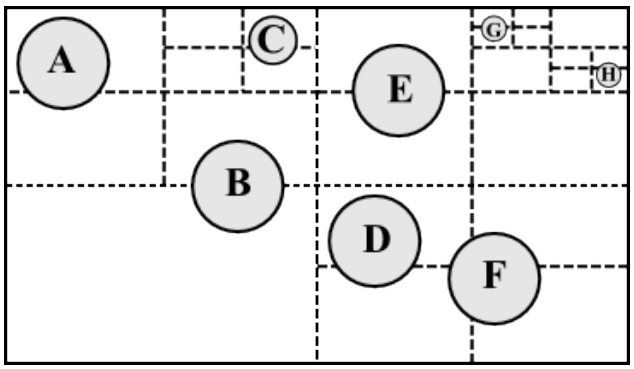
$5, 3, 8, 2, 4, 1, 7, 9, 6$

Rita upp hur hela trädet ser ut efter varje insättning.

3.
Utgå från det binära sökträdet i uppgift 2 och ta bort noden med nyckel 5. Ta sedan bort noden med nyckel 3 och sist noden med nyckel 8. Rita upp hur hela trädet ser ut efter varje borttagning.

(Fortsätt gärna att sätta in och ta bort element om du vill öva mer på binära sökträd)

4.
Ett program används för att lagra positionen och radien hos ett antal cirklar på ett rektangulärt område. Genom att rekursivt dela upp området i kvadranter säger man att en cirkel tillhör en kvadrant om den skär någon av kvadrantens mittlinjer:



Internt använder programmet ett quadträd där varje nod är kopplat till en kvadrant. Rotnoden är kopplat till hela planet, och barnen till en nod är kopplade till varsin av förälderns fyra kvadranter. Om en cirkel tillhör flera kvadranter lagras den bara i den nod som är högst upp i trädet

Rita trädet som representerar ovanstående figur, alltså ett träd där varje nod har högst fyra undernoder och innehåller de cirklar som tillhör motsvarande kvadrant. Om det inte finns några cirklar i en kvadrant behöver noden inte ritas.

(Rotnoden kommer endast att innehålla cirkeln B och ha tre undernoder)

5.
Skriv en funktion `treeHeight(t)` som returnerar höjden av ett binärt träd `t` (se definitionen nedan), dvs den längsta vägen från roten till ett löv. Bestäm sedan tidskomplexiteten i bästa och värsta fall, t.ex. genom att hitta en rekursiv formel och bestämma dess slutna form.

```
datatype 'a tree = Void | Node of 'a * 'a tree * 'a tree
```

6.
Gör en preorder-, inorder- och postorder-traversering av sökträdet från uppgift 2 (innan elementen togs bort i uppgift 3). Skriv sedan en ML-funktion `inOrder(t)` som returnerar en lista med noderna i ett binärt träd `t` (se ovan) i den ordning som de besöks vid en inorder-traversering. Vad är tidskomplexiteten för `inOrder`? Hur kan man ändra i koden för att få en pre- eller postorder-traversering? Påverkas komplexiteten?

***7.**
Betrakta nedanstående funktion (där `split` delar en lista i två lika stora delar):

```
fun foo [x] = [x]
  | foo l = let
                val (a,b) = split(l)
            in
                foo(a) @ foo(b) @ foo(a) @ foo(b)
            end;
```

Om `l` har $n = 2^k$ element ($k \in \mathbf{N}$), hur lång blir den resulterande listan av anropet `foo(l)`?

Lycka till!