SI-möte #14, Programkonstruktion och datastrukturer

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

28 februari 2012

Komplexitet

Ange vilka av följande implikationer som alltid stämmer

$i)$ $f(n) = \Theta(n) \Rightarrow f(n) = O(n^2)$	$ii)$ $f(x) = \Omega(x) \Rightarrow f(x) = O(x^2)$
$iii)$ $T(n) = O(\log n) \Rightarrow T(n) = O(n \log n)$	$iv)$ $p(x) = \Theta(q(x)) \Rightarrow q(x) = \Theta(p(x))$
$v)$ $f(t) = O(g(t)) \Rightarrow g(t) = \Omega(f(t))$	$vi)$ $A(u) = \Theta(B(u)) \Rightarrow A(u) = O(B(u))$
$vii)$ $k(t) = \Theta(y(t)) \Rightarrow y(t) = \Omega(k(t))$	$viii)$ $g(n) = O(f(n)) \Rightarrow f(n) = \Theta(g(n))$

Ange tidskomplexiteten för följande rekursiva funktioner. Bevisa att ditt svar stämmer.

```
fun foo([]) = 0
  | foo(f::r) = 1 + foo(r);

fun bar([]) = 2
  | bar(f::r) = foo(r) + bar(r);

fun baz(0) = 42
  | baz(n) = baz(n div 2) + baz(n div 2);
```

Imperativ programmering

$i)$ Skriv en funktion som skriver ut alla element i en sträng-lista i omvänd ordning (alltså sista elementet först) till terminalen.

$ii)$ Skriv en funktion `removeTags(is)` som givet en inström `is` returnerar en sträng med alla tecken från `is` (till filslut) men utan de tecken som står mellan "<" och ">". Om `is` pekar på en fil med innehållet "hej <hopp>på dig!", ska alltså strängen "hej på dig!" returneras.

`input1(is):instream -> char option` returnerar nästa tecken från `is` taggat med `SOME`, och `NONE` vid filslut.

Datastrukturer

Fyll i tabellen nedan med tidskomplexiteten i genomsnittliga och värsta fall för given operation och datastruktur (se till att den aktuella strukturens invariant bevaras). *Motivera*, åtminstone informellt, varje val av komplexitet.

Med *åtkomst* menas att komma åt ett element på en viss plats eller med en viss nyckel i strukturen, med *insättning* att sätta in ett element (på rätt plats) i strukturen och med *borttagning* att hitta och ta bort ett element (och vid behov modifiera datastrukturen). *Extrahera min* innebär att hitta och ta bort elementet med den minsta nyckeln. För de två sista operationerna kan data och nyckel ses som samma sak i listor och vektorer.

	Åtkomst		Insättning		Borttagning		Extrahera min	
(enkellänkad) Lista								
Sorterad lista								
Vektor*								
Sorterad vektor								
Binärt sökträd								
Red-black tree/AVL-träd								
Binomial min heap								
Hashtabell (chaining)								
Hashtabell (öppen adr.)								

**Antag att vektorn hålls helt fylld från vänster och att man håller reda på index för elementet längst till höger*

Sätt in följande element (i given ordning) i ett red-black tree

$$4, 10, 1, 8, 9, 13, 7, 6$$

Sätt in följande element (i given ordning) i en binomial max heap

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

och ta sedan bort det största elementet fyra gånger.

Sätt in följande element (i given ordning och utan att rehasha) i en hashtabell med 7 celler, hashfunktionen $h(k) = k \ mod \ 7$ och probfunktionen $p(i) = i^2$.

$$37, 13, 31, 48, 23, 69, 22$$

Leta upp och ta bort elementen $37, 22, 31$ och 69.

Lycka till!