TDDI16 Datastrukturer och algoritmer

11 januari 2010, kl 14-18

1. [2p]

Analysera tidskomplexiteten för följande program. Ge en exakt lösning. Antag att n är antalet element i listan.

```
procedure Sort( A : list of sortable items );
var i,j, value : integer;
begin
    for i := 1 to length[A]-1 do begin
        value := A[i];
        j := i - 1;
        while j >= 0 and A[j] > value do begin
            A[j + 1] := A[j];
            j := j - 1;
        end
        A[j + 1] := value;
    end
end
```

2. [2 + 2 = 4p]

(a) Definiera ADT *Stack* (modell + operationerna - förklara vad operationerna gör).

(b) Beskriv en implementation av ADT *Stack* så att varje operation kan köras i konstant tid.

3. [3 + 2 = 5p]

(a) Skriv en algoritm i pseudokod som använder ADT *Tree* för att räkna antalet noder i ett träd. (Använd operationer så som de är definierade för ADT Tree.) Vad är tidskomplexitetet av din algoritm? Förklara.

(b) Förklara implementationen av ADT *Tree* som använder en 'parent'-array för att lagra information om vilken nod som är förälder till en nod. Ange tidskomplexitet för varje operation för ADT *Tree* vid användning av denna implementation.

4. [2 + 2 = 4p]

(a) Antag en hashtabell av storlek 7 och nedanstående operationer. Antag öppen hashing. Visa hur tabellen ser ut efter VARJE operation. (Operationerna körs i den givna ordningen.) Använd de givna hashvärdena h(x).

(i)	*Insert*(A,'LECTURES')	h('LECTURES') = 3
(ii)	*Insert*(A,'ABOUT')	h('ABOUT') = 2
(iii)	*Insert*(A,'DURING')	h('DURING') = 2
(iv)	*Delete*(A,'ABOUT')	h('ABOUT') = 2

(b) Förklara linjär sondering. Ge ett exempel.

5. [1 + 2 = 3p]

(a) AVL-träd

Visa hur AVL-trädet i figur 1 ser ut efter operationen *Insert*('b').

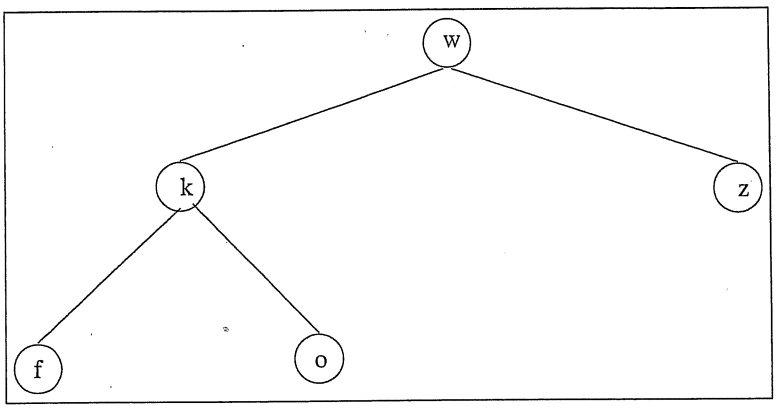

Figure 1: AVL-träd

(b) PriorityQueue

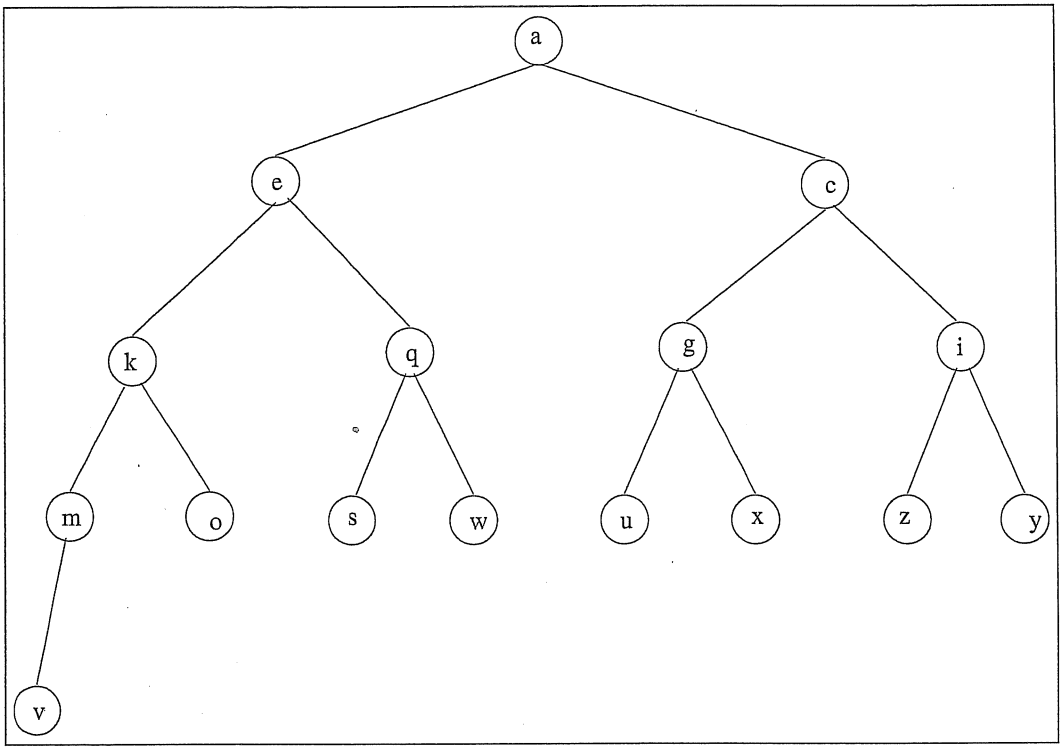

Figure 2: heap

(i) Visa hur heapen i figur 2 ser ut efter operationen *Insert*('b').

(ii) Visa hur heapen i figur 2 ser ut efter operationen *DeleteMin* (*utan* den tidigare *Insert*).

6. [2 + 2 + 1 + 1 = 6p]

(a) Visa hur *QuickSort* (med initial swap) sorterar sekvensen

$$< 23, 45, 57, 13, 11, 14, 20, 32 >$$

Låt det största av de första två skillda nycklar vara *pivot*.

(b) Visa hur sekvensen sorteras med hjälp av 2-positions *RadixSort* med bas 10.

(c) Sorteringsproblemet tar $\Omega(n \log n)$ tid, medans *RadixSort* har tidskomplexitet $O(n)$. Förklara hur detta är möjligt.

(d) Ange tidskomplexitet för *QuickSort* och *MergeSort* i värsta fall OCH medelfallet.

7. [2p]

Lös följande rekursiv ekvation.

$$\begin{array}{lll} T(n) = & d & \text{if } n = 1 \\ & T(n/2) + n & \text{if } n \geq 2 \end{array}$$