TDDI16 Datastrukturer och algoritmer

24 augusti 2011, kl 8-12

1. [**2p**]

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

```
procedure S( A[1..n] : list of integers );
var i,j,k: integer;
var B[1..n] : list of integers;
begin
    for k := 1 to n do B[k] := 0;
    for i := n downto 1 do
        for j := 1 to i do
                B[i] := B[i]+A[j];
end
```

2. [**4p**]

(a) Definiera ADT *Deque* (modell + operationerna - förklara vad operationerna gör), som är en lista där all insättning och borttagning sker vid både ändena (*front* och *rear*).

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

3. [$\mathbf{2 + 2 + 1 = 5p}$]

(a) Skriv en algoritm i pseudokod som använder ADT $Tree$ för att räkna antalet löv i ett träd. (Använd operationer så som de är definierade för ADT Tree.)

(b) Välj sedan en implementation av ADT $Tree$ och ange tidskomplexitet för varje operation för ADT $Tree$ vid användning av denna implementation.

(c) Vad är tidskomplexiteten av din algoritm i (a) med valet av implementationen för ADT $Tree$ i (b)? Förklara.

4. [**2p**]

Antag en hashtabell av storlek 5 och nedanstående operationer. Antag stängd hashing och *kvadratisk* sondering (quadratic probing) som kollisionshanteringsstrategi (med **mod 5**). Visa hur tabellen ser ut efter VARJE operation. (Operationerna körs i den givna ordningen.) Använd de givna hashvärdena h(x).

(i)	$Insert(A,$'a')	h('a') = 3
(ii)	$Insert(A,$'b')	h('b') = 2
(iii)	$Insert(A,$'c')	h('c') = 2
(iv)	$Delete(A,$'b')	h('b') = 2

5. [**1 + 1 = 2p**]

a) Varför tar insätting i ett binärt sökträd O(n) tid? Förklara. (Ange vilka operationer som görs och hur mycket tid dessa tar.)

b) Antag att vi har heltalen 1, 2, 3, 4, 5, 6, 7 och vi lagrar dessa i binära sökträd.
(i) Rita ett binärt sökträd med minst möjliga höjden som innehåller dessa element.
(ii) Rita ett binärt sökträd med störst möjliga höjden som innehåller dessa element.

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

Antag 2-3-trädet A i figur 1.

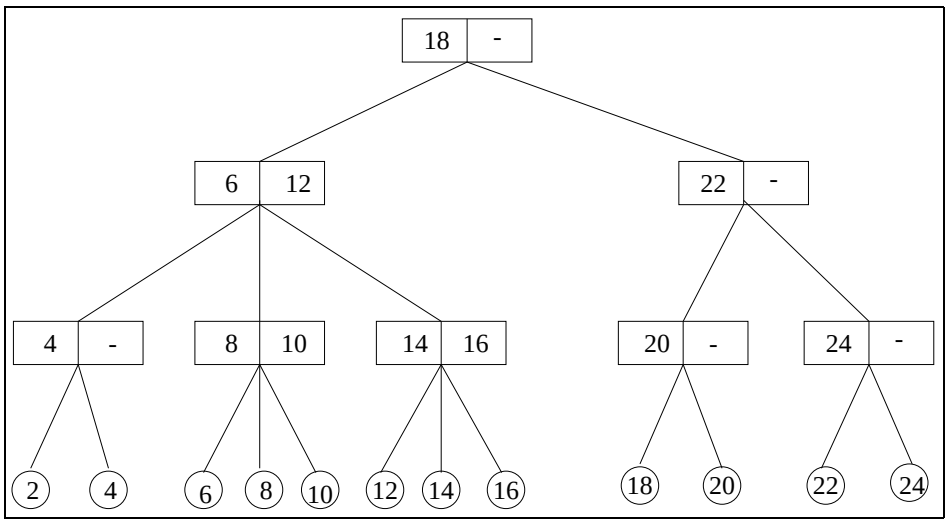

Figure 1: 2-3-tree

(a) Visa hur trädet ser ut efter operationen $Insert(17, A)$.

(b) Visa hur trädet ser ut efter operationen $Delete(24, A)$ (*utan* den tidigare $Insert$).

7. [$\mathbf{2+3+1+1=7p}$]

(a) Visa hur *MergeSort* sorterar

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

(b) Visa hur *HeapSort* sorterar

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

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

(d) Är en algoritm med optimal tidskomplexitet också alltid snabbast? Förklara ditt svar.

8. [**2p**]

Lös följande rekursiv ekvation.

$$\begin{array}{lll} T(n) = & d & \text{if } n = 1,2 \\ & T(n-2) + 1 & \text{if } n \geq 3 \end{array}$$