

Lösningsförslag till tentamen

Kursnamn
Tentamensdatum

Algoritmer och datastrukturer
2015-04-18

Program
Läsår
Examinator

DAI2+I2
2013/2014, lp 4
Uno Holmer

Uppgift 1 (10 p)

Ingen lösning ges. Se kurslitteraturen.

Uppgift 2

a)

```
public static int mult( int x, int y ) {  
    if ( x == 0 || y == 0 )  
        return 0;  
    if ( x < 0 )  
        return -mult( -x, y );  
    else  
        return y + mult( x - 1, y );  
}
```

b) Följande implementeringar förutsätter representation utan listhuvud.

Rekursivt:

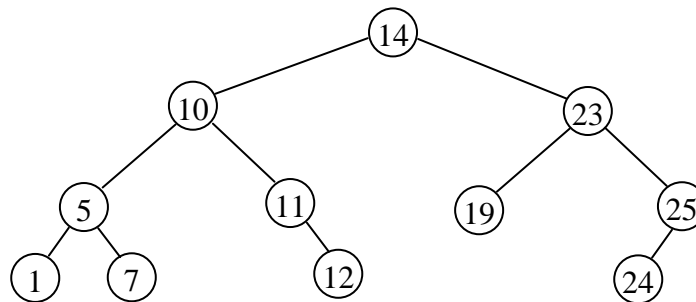
```
static ListNode insert( int x, ListNode l ) {  
    if ( l == null || x <= l.data )  
        return new ListNode( x, l );  
    else {  
        l.next = insert( x, l.next );  
        return l;  
    }  
}
```

Iterativt:

```
static ListNode insertIter( int x, ListNode l ) {  
    if ( l == null || x <= l.data )  
        return new ListNode( x, l );  
    else {  
        ListNode p = l;  
        while ( p.next != null && x > p.next.data )  
            p = p.next;  
        p.next = new ListNode( x, p.next );  
        return l;  
    }  
}
```

Uppgift 3

- a) Ett AVL-träd är ett binärt sökträd där höjdskillnaden mellan vänster och höger delträd inte för någon nod överstiger 1.
- b) Talet var 24. Den djupaste obalanserade noden är den som innehåller 10. En vänsterrotation runt denna, motsvarande fall nr 4 i Weiss löser problemet. v.g.v.



Uppgift 4

- a) Linjär sondering kan leda till primär klusterbildning. Talen 4, 25 och 34 bildar ett sådant där 34 har hamnat på avstånd 2 från sin hashposition. Varje ev. efterföljande inhashning av nya element på någon av positionerna 4-6 kommer att bidra tillklustrets tillväxt. Söktiderna blir längst för element med hashvärdet 4. Ett annat kluster bildas av elementen 48, 8 och 28.
- b) Vid kvadratisk sondering skall tabellstorleken vara ett primtal och belastningsfaktorn λ får ej överstiga 0.5. Den minsta storleken som uppfyller detta för 6 element är $M = 13$.
- c) Så här ser en tabell med 13 platser ut efter insättningssekvensen 25, 4, 34, 48, 8, 28. Talet 8 kolliderar med 34 som har samma hashvärde, därefter med 48, 25 och 4. Alternativpositionerna på kvadratisk avstånd från 8 (modulo 13) som provas är $8 + 1$, $8 + 4$, $8 + 9$ samt $8 + 16$.

0	
1	
2	28
3	
4	4
5	
6	
7	
8	34
9	48
10	
11	8
12	25

Uppgift 5

- a) Trädet t1 är en binär hög. t2 är ingen binär hög eftersom ordningsvillkoret inte är uppfyllt (däremot är det binärt sökträd). t3 är inte komplett så strukturvillkoret uppfylls ej.
b)

	1	2	5	6	10	9	8	11
0	1	2	3	4	5	6	7	8

Uppgift 6

- a) Bågen D-B ingår i båda cyklerna A-C-D-B-A resp. E-D-B-E. Om man tar bort bågen i fråga blir grafen en DAG.
b) De möjliga topologiska ordningarna är: BAFECD, BACFED, BAFCED, BFACED, BFAECD, BFEACD, FBACED, FBAECD, FBEACD.

Uppgift 7

a)

```
public class Graph {
    private HashMap<String,TreeSet<String>> graphTable =
        new HashMap<String,TreeSet<String>>();

    public void addEdge( String v, String w ) {
        TreeSet<String> neighbours = getNeighbours( v );
        if ( neighbours == null ) {
            neighbours = new TreeSet<String>();
            graphTable.put( v, neighbours );
        }
        neighbours.add( w );
    }

    public TreeSet<String> getNeighbours( String v ) {
        return graphTable.get( v );
    }
}
```

b)

```
public static void bfs( Graph g, String startNode ) {
    HashSet<String> visited = new HashSet<String>();
    Queue<String> q = new LinkedList<String>();
    q.add( startNode );
    while ( ! q.isEmpty() ) {
        String v = q.poll();
        if ( ! visited.contains( v ) ) {
            visited.add( v );
            System.out.println( v );
            TreeSet<String> neighBours = g.getNeighbours( v );
            if ( neighBours != null )
                for ( String w : neighBours )
                    q.add( w );
        }
    }
}
```