TENTAMEN: Algoritmer och datastrukturer

Läs detta!

- *Uppgifterna är inte avsiktligt ordnade efter svårighetsgrad.*
- Börja varje uppgift på ett nytt blad.
- Skriv ditt namn och personnummer på varje blad (så att vi inte slarvar bort dem).
- **Skriv rent dina svar. Oläsliga svar r ä t t a s e j!**
- Programmen skall skrivas i Java 5, vara indenterade och kommenterade, och i övrigt vara utformade enligt de principer som lärts ut i kursen.
- Onödigt komplicerade lösningar ger poängavdrag.
- Programkod som finns i tentamenstesen behöver ej upprepas.
- Givna deklarationer, parameterlistor, etc. får ej ändras, såvida inte annat sägs i uppgiften.
- Läs igenom tentamenstesen och förbered ev. frågor.

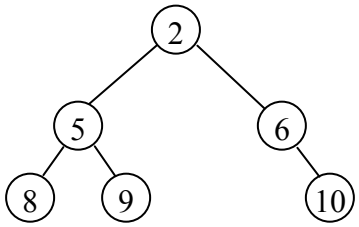
I en uppgift som består av flera delar får du använda dig av funktioner klasser etc. från tidigare deluppgifter, även om du inte löst dessa.

Lycka till!

Uppgift 1

Välj **ett** svarsalternativ på varje fråga. Motivering krävs ej. Varje korrekt svar ger två poäng. Garderingar ger noll poäng.

1. Para ihop rätt genomlöpningsordning med rätt utskrift för trädet



A = 2 5 8 9 6 10
B = 8 9 5 10 6 2
C = 8 5 9 2 6 10

   a. A inorder, B preoreder, C postorder
   b. B inorder, C preoreder, A postorder
   c. B inorder, A preoreder, C postorder
   d. C inorder, B preoreder, A postorder
   e. C inorder, A preoreder, B postorder
   f. A inorder, C preoreder, B postorder

2. Vilken datastruktur används i Dijkstras algoritm?
   a. Stack
   b. Prioritetskö
   c. Hashtabell
   d. Binärt sökträd
   e. Kö

3. Antag att tidskomplexiteten hos f är linjär

```
int p = 1;
for ( int i = 0; i < n; i++ )
    if ( i == p ) {
        p *= 2;
        f( i );
     }
```

   Vilket av a-d är en minsta övre begränsning för tidskomplexiteten hos kodavsnittet ovan?
   a. $O(\log n)$
   b. $O(n)$
   c. $O(n \cdot \log n)$
   d. $O(n^2)$

4. Tekniken med fältdubblering som används i en del datastrukturimplementeringar ger ett genomsnittligt minnesutnyttjande om
   a. 10%
   b. 25%
   c. 50%
   d. 75%
   e. 90%

forts.

5. Para ihop begreppen till vänster med motsvarande som passar bäst till höger

Ordnad mängd	Stack
LIFO	Fält
Associationslista	Hashtabell
Random access	Binärt sökträd
Oordnad mängd	Länkad lista
Flexibel insättning/uttag	Kö
FIFO	Map

(10 p)

Uppgift 2

a) Konstruera en rekursiv javametod som beräknar värdet av x∗y. Metoden får använda standardoperatorerna ==, + och –, men inte ∗ och den får inte innehålla några loopar. Talen kan vara negativa.

```
public static int mult( int x, int y )
```

(5 p)

b) Konstruera en javametod som sätter in ett heltal i en ordnad heltalslista så att ordningen bevaras. Listnoder definieras med klassen

```
public class ListNode {
      int data;
      ListNode next;

      public ListNode( int data ) {
            this.data = data;
            next = null;
      }

      public ListNode( int data, ListNode next ) {
            this.data = data;
            this.next = next;
      }
}
```

Metoden får bara skapa en ny nod för det insatta talet, inga noder i den befintliga listan får kopieras. Metoden skall returnera den nya listan som resultat.

```
public static ListNode insert( int x, ListNode l )
```

En rekursiv lösning ger max 10p, en iterativ max 5p.

(10 p)

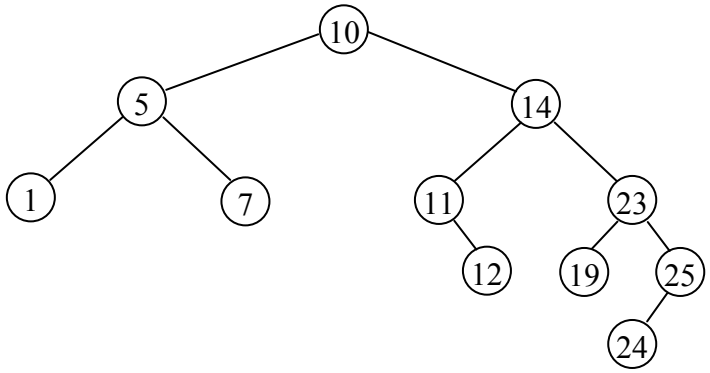
Uppgift 3

a) Vad är ett AVL-träd? Ge en definition i ord.

(1 p)

b) Ett tal har just satts in så att AVL-vilkoret bröts. Vilket var talet? Genomför lämplig rotation så att AVL-villkoret uppfylls.

(3 p)



Uppgift 4

I en hashtabell med tio platser har sex tal satts in i ordningen 25, 4, 34, 48, 8, 28 med metoden linjär sondering (linear probing). Hashfunktionen definieras $hash(x) = x \bmod M$, där M är antalet platser i tabellen.

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

a) Vad kallas prestandaproblemet som linjär sondering kan leda till? Hur yttrar det sig i tabellen ovan

(2 p)

b) Vilken är den minsta lämpliga tabellstorleken för sex element om istället kvadratisk sondering används? Motivera!

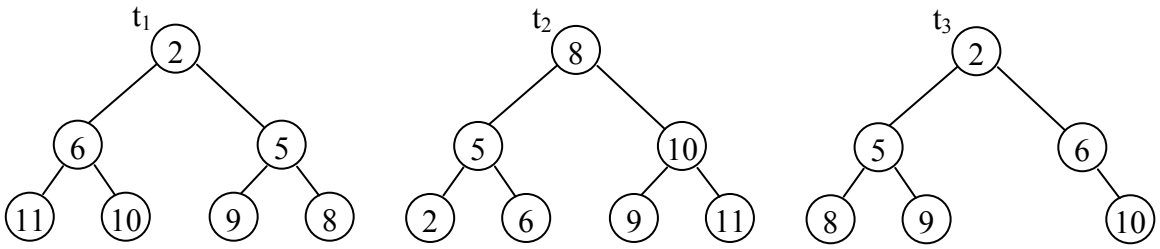
(1 p)

c) Visa hur en tabell av storlek enligt b) ser ut efter samma insättningssekvens som ovan om samma hashfunktion används. En förutsättning för poäng på denna deluppgift är att du svarat rätt på deluppgift b).

(4 p)

Uppgift 5

a) Ett av träden är en binär hög, vilket?



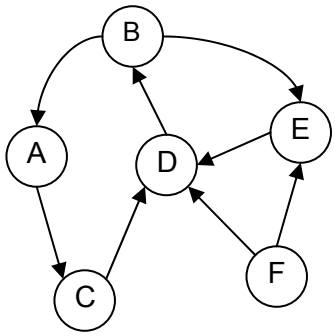
(1 p)

b) Hur ser fältrepresentationen av den binära högen i a ut efter insättning av 1?

(2 p)

Uppgift 6

a) Vilken båge i grafen måste tas bort för att noderna skall kunna ordnas topologiskt? Varför?



(1 p)

b) Ange alla möjliga topologiska ordningar av noderna när bågen enligt a) tagits bort.

(9 p)

Uppgift 7

En enkel klass för riktade grafer utan kostnader på bågarna kan definieras

```
public class Graph {
    // datarepresentation

    public void addEdge( String v, String w ) {
      // adds an edge from v to w in the graph
    }

    public TreeSet<String> getNeighbours( String v ) {
        // returns the neighbours of v or null if
        // none exist.
    }
}
```

a) Skriv färdigt klassen `Graph`. Använd lämplig datastruktur.

(4 p)

b) Implementera metoden

```
public static void bfs( Graph g, String startNode )
```

som skriver ut alla noder i en graf som kan nås från angiven startnod enligt besöksordningen **b**redden-**f**örst. Metoden måste kunna hantera grafer som innehåller cykler. Använd klassen i a) och andra lämpliga datastrukturer. Ex på graf och resultat från `bfs`:

```
Graph g = new Graph();
g.addEdge( "A", "B" );
g.addEdge( "A", "D" );
g.addEdge( "B", "C" );
g.addEdge( "B", "G" );
g.addEdge( "C", "E" );
g.addEdge( "C", "F" );
g.addEdge( "D", "C" );
g.addEdge( "D", "I" );
g.addEdge( "E", "H" );
g.addEdge( "F", "D" );
g.addEdge( "F", "E" );
g.addEdge( "G", "E" );
g.addEdge( "H", "F" );
g.addEdge( "I", "G" );
```

```
bfs( g, "E" );

utskrift
E
H
F
D
C
I
G
```

(7 p)