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, eller senare, 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.

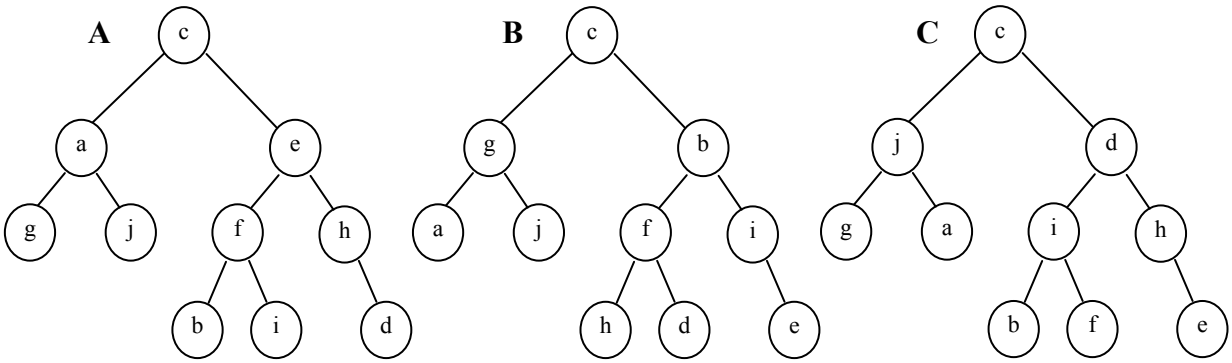
I en uppgift som består av flera delar får du använda dig av funktioner klasser etc. från tidigare deluppgifter, även om du inte löst dessa.

Lycka till!

Uppgift 1

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

1. Vilket av träden har postorder genomlöpning g, j, a, b, i, f, d, h, e, c och inorder genomlöpning g, a, j, c, b, f, i, e, h, d ?



2. För vilken av sorteringsmetoderna utgör en redan sorterad indataföljd värsta fallet?
   a. shellsort
   b. quicksort
   c. mergesort
   d. heapsort

3. Ett av sambanden 1-3 beskriver tidskomplexiteten hos f.

   1. $\begin{cases} T(1) = 1 \\ T(N) = T(N/2) + c \end{cases}$

   2. $\begin{cases} T(1) = 1 \\ T(N) = 2T(N/2) + c \end{cases}$

   3. $\begin{cases} T(1) = 1 \\ T(N) = T(N/2) + N \end{cases}$

```
int f( int n ) {
    if ( n == 0 )
        return 1;
    else {
        int s = f( n/2 );
        for ( int i = 0; i < n; i++ )
            s += 1;
        return s;
    }
}
```

Vilket av a-d är en minsta övre begränsning för lösningen till sambandet i fråga?

a. $O(\log N)$
b. $O(N)$
c. $O(N \cdot \log N)$
d. $O(N^2)$

v.g.v.

4. Antag att du skall konstruera en algoritm som med en textfil av ord som indata rapporterar radnumret i filen för den andra förekomsten av varje ord som påträffas minst två gånger. Vilken ADT är mest användbar för en effektiv lösning av problemet?
   a. lista
   b. kö
   c. stack
   d. mängd
   e. prioritetskö
   f. map

5. Följande funktion bryter mot vilken/vilka regler för rekursion?

```
int f( unsigned int n ) {
    if ( n == 0 )
        return 0;
    else
        return n + f( n/2 ) + f( n/2 + 1 );
}
```

   a. den saknar basfall
   b. den terminerar ej
   c. den utför redundant arbete
   d. två av ovanstående

(10 p)

Uppgift 2

Klassen `TreeNode` kan användas för att representera binära träd med element av godtycklig typ:

```
public class TreeNode<E> {
    public TreeNode<E> left, right;
    public E element;

    public TreeNode( E element, TreeNode<E> left, TreeNode<E> right ) {
        this.element = element;
        this.left = left;
        this.right = right;
    }
}
```

Antag att ett binärt sökträd representeras med noder av ovanstående typ. Definiera metoden

```
public static <E extends Comparable<? super E>>
TreeNode<E> insert( E x, TreeNode<E> t )
```

som sätter in `x` på rätt ställe i trädet med rot i `t`. Metoden skall returnera en referens till trädets rot efter insättningen. Duplikat får ej sättas in, i sådana fall kastas undantaget `IllegalArgumentException`. Lös problemet rekursivt. Balansering av trädet ingår ej i uppgiften.

(8 p)

Uppgift 3

Höjden hos en nod i ett binärt träd definieras i kursboken som väglängden från roten till den djupast liggande noden.

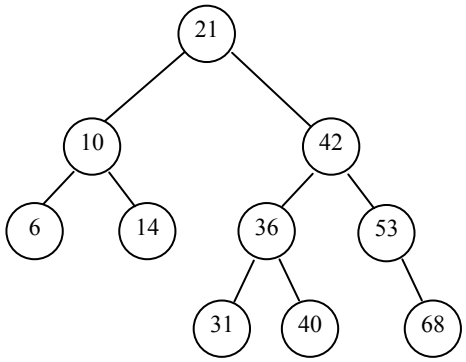
a) Rita det binära sökträdet som fås efter sekvensen

```
insert(20); insert(30); insert(25); insert(10);
    insert(15); insert(28); insert(26);
```

vi antar att trädet är tomt från början och att inga balanseringsoperationer utförs.

(2 p)

b) Visa med en bild hur AVL-trädet nedan ser ut efter insättning av 37 med nödvändig(a) rotation(er).



(4 p)

Uppgift 4

a) Vad innebär begreppet primärt kluster i en hashtabell? När uppkommer primära kluster? Vilken metod används för att undvika primär klusterbildning?

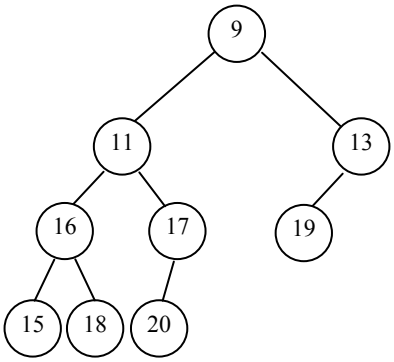
(3 p)

b) Ange två villkor som måste vara uppfyllda vid kvadratisk sondering (probing).

(2 p)

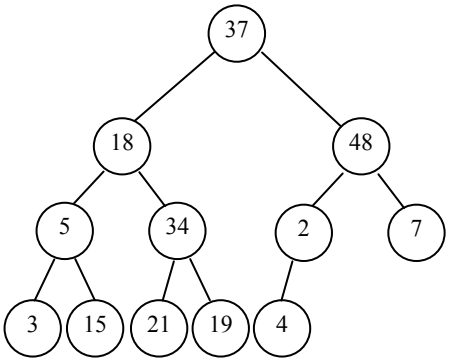
Uppgift 5

a) Är trädet nedan en binär hög? Motivera ditt svar!



(2 p)

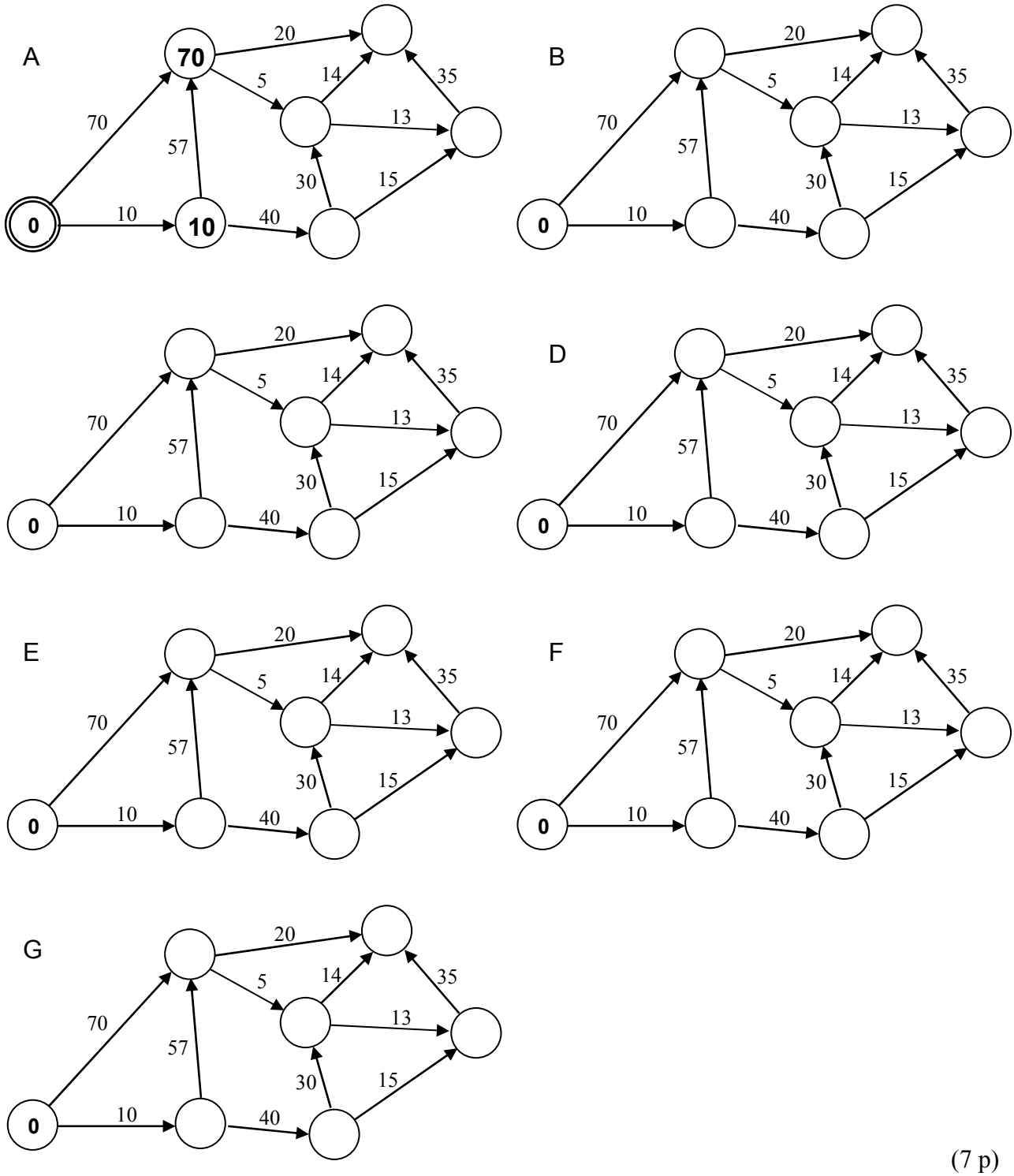
b) Visa hur trädet nedan ser ut efter att operationen buildHeap applicerats på det.



(4 p)

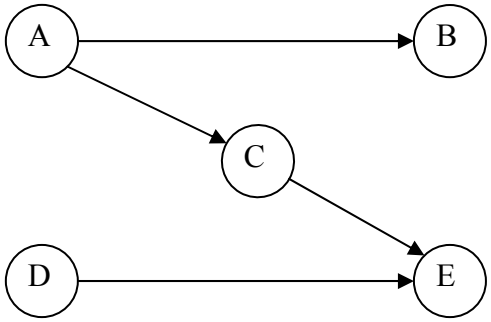
Uppgift 6

a) Visa steg för steg hur Dijkstras algoritm beräknar kortaste viktade väglängderna från noden märkt 0 till samtliga övriga noder. Fig. A visar hur det ser ut då första noden besökts. Visa de övriga stegen direkt i tesen, ett per figur i ordning B-G. Skriv in beräknade väglängder i noderna som exemplet A visar. Markera vilken nod som besöks i resp. steg med en extra ring runt noden. ***Använd blyertspenna!***



(7 p)
v.g.v.

b) Ange alla möjliga topologiska ordningar av noderna i grafen



(6 p)

Uppgift 7

Följande enkla klass kan representera riktade grafer utan kostnader på bågarna. Metoden `isDAG` returnerar `true` om grafen är en DAG, `false` annars.

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

    public void addEdge( String from, String to ) {
        if ( ! graphTable.containsKey( from ) )
            graphTable.put( from, new ArrayList<String>() );
        if ( ! graphTable.containsKey( to ) )
            graphTable.put( to, new ArrayList<String>() );

        graphTable.get( from ).add( to );
    }

    public boolean isDAG() { ... }

    ...
}
```

Implementera `isDAG`! *Ledning*: Lägg till en privat rekursiv hjälpmetod som undersöker om ett visst villkor som krävs för att grafen skall vara en DAG är uppfyllt. En mängd kan vara användbar i algoritmen.

(12 p)