Tentamen i Algoritmer & Datastrukturer i Java

Hjälpmedel: Skrivhjälpmedel, miniräknare.
Ort / Datum: Halmstad / 2008-05-27
Skrivtid: 4 timmar
Kontakt person: Nicolina Månsson, tel. 035-167487
Poäng / Betyg: Max 49 poäng
_ >=20poäng ger betyg 3
_ >=30poäng ger betyg 4
_ >=40poäng ger betyg 5

Övrigt: Det finns två olika sorters frågekategorier, de som skall besvaras genom programmerings och de som skall besvaras genom förklaringar med text och illustrationer.
Programmeringslösningarna skall i största mån vara skrivna i korrekt Java-syntax.
Koden skall vara välstrukturerad och lättläst.
Koden skall innehålla lämpliga ledtexter (utskrifter) som instruerar användaren vad som krävs av honom / henne.
Om en lösning är uppenbart ”klumpig” och det anses att tentanden skall känna till en smidigare lösning kan den ”klumpiga” lösningen medföra poängavdrag.
Förklaringslösningar bör, om tillämpningsbart, innehålla illustrationer på relevanta datastrukturer och använda algoritmer.
Tänk på att vara noggrann och strukturerad. Det är Du som skall visa vad Du kan!

Lycka Till!
/ Nicolina Månsson

Uppgift 1 – Tidskomplexitet (4p+2p)

a) Givet är en array a med n heltal, sorterade i växande ordning. Man vill undersöka om det finns två tal i arrayen vars summa är ett visst tal **sum** med följande algoritm:

```
int i = 0;
while (i < a.length) {
boolean found = binarySearch(a, sum -a[i]);

if (found)
     return true;
else
     i++;
}
return false;
```

Metoden binarySearch(int[] a, int x) söker med binärsökning efter talet x i vektorn a.

1) Förklara hur algoritmen fungerar.
2) Uppskatta exekveringstiden för algoritmen i form av Big-Oh notation. Motivera svaret.
3) När inträffar värsta fallet för den givna algoritmen och vad blir tidskomplexiteten då om vektorn har längden n? Du får använda kända resultat avseende tidskomplexiteten för binärsökning.

b) Ange för nedanstående delar av olika algoritmer en uppskattning av exekveringstiden i form av Big-Oh notation. Motivera också din uppskattning.

1)
```
for( int i=1; i<n; i++)
     x=x+1;
 for(int j=1; j<n/2; j++)
     x=x+2;
```

2)
```
for(int i=n;i>=1;i=i/2){
  for( int j=1;j<n;j++)
        x=x+1;
 }
```

Uppgift 2 – Datastrukturer (9p+9p+2p)

a) Du skall tillämpa dina kunskaper om länkade listor i en registerapplikation
En register innehåller Person-objekt och skall implementeras som en länkad lista med metoderna add(Person p), remove(Person p), print().

Här är klassen Person som definierar en Person i en register applikation.

1) Komplettera klassen Person med metoden **equals()**.

```
class Person {

String namn;
String telNr;

public Person (String inamn, String itelNr)
{
   namn=inamn;
   telNr=itelNr;
}

public String toString()
{
  return namn+ ”  ” telNr;
}

// skriv din här kod

}
```

2) Du behöver också en klass som definierar en Node i en Register. Komplettera klassen med lämpliga konstruerare. Förutsätt att Node klassen är i samma fil med Register klassen, då kommer du åt Person-objeket med punkt notation.

```
class Node
{
    Person pers;
    Node next;

// skriv din här kod

}
```

3) Och till sist klassen som beskriver registret och som skall implementeras som en länkad lista.
Komplettera klassen med lämpliga konstruerare samt metoderna
-add(Person p), lägger till ny person i registret
-remove(Person p), ta bort en viss person x från registret
-print(), skriver ut registret

```
class Register
{
   Node header;

// skriv din här kod

}
```

b) Stacken tillhör bland de enkla datastrukturerna som kan implementeras med array.
I en applikation behöver jag en ”skräddarsydd” Stack av Object som inte tillåter dubbletter.

Implementera klassen Stack med metoderna **push(Object x)** och **pop()**.
Använd metoden ***equals()***från klassen Object för att testa om objektet **x** finns redan på Stacken eller inte.

Tänk på ett sätt att hantera ”stack duplicate”, ”stack overflow” och stack ”underflow”.

c)I kursboken ges det programkod för en sk. *”wrap-around”* implementation av en kö. Förutsätt att det finns en konstruktor till klassen ArrayQueue som tillåter användaren att sätta storleken på datastrukturen.
Givet nedanstående testkod, illustrera steg för steg hur den arraybaserade kön används och markera tydligt front och back indexen!

```
public static void main(String [] args)
{
ArrayQueue queue = new ArrayQueue(4);
queue.enqueue(new Integer(5));
queue.enqueue(new Integer(2));
queue.enqueue(new Integer(6));
queue.dequeue();
queue.dequeue();
queue.enqueue(new Integer(1));
queue.enqueue(new Integer(8));
queue.dequeue();
queue.enqueue(new Integer(3));
queue.dequeue();
queue.enqueue(new Integer(0));
}
```

Uppgift 3 – Kända algoritmer (3p+3p)

a)Beskriv kortfattat sorteringsalgoritmerna mergesort, quicksort och heapsort med ord eller pseudokod. (Skriv bara några få meningar eller ”pseudokodsrader” per algoritm).

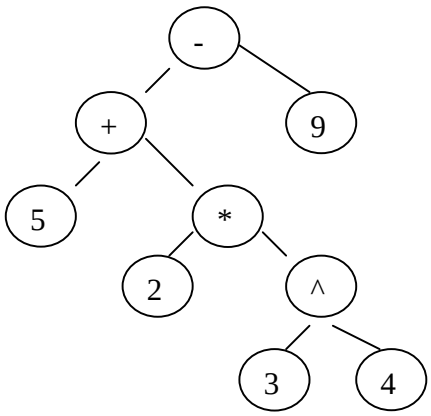
b) Påverkas tidskomplexiteten för någon/några av algoritmerna om elementen redan är sorterade. I så fall hur? Motivera ditt svar.

Uppgift 4 – Träd och tillämpningar(4p+2p+2p+4p)

a) Beskriv kortfattat principen med Huffman komprimering och visa tydligt steg för steg hur Huffmanträdet genereras för följande indata. Visa också vilka koder respektive tecken får.

a a b c d a a a b b b c c d

b) Vi kan använda binära träd för att representera algebriska uttryck. Dessa träd kallas ”expressions tree”. För följande träd:



Om man skriver ut innehållet i trädet genom att använda metoden printPostOrder() kommer du att uppnå aritmetiska uttryck i *postfix notation*. Vilket blir uttrycken för ovanstående träd.

c) För att beräkna aritmetiska uttryck i postfix notation används datastrukturen **operator stack**. Visa hur stacken förändras när du beräknar *postfix uttrycket* som du fått fram i uppgift a.

d) Vi har sett två implementationer av balanserade binära sökträd. En så kallad AVL träd och den andra AA träd.

Förklara balanseringsprincipen för dessa två typer av träd. Använd lämpliga exempel för att göra det tydligt.

Uppgift 5 – Binär Heap och Graph(3p+2p)

a)Vilka av följande påståenden om minheapar är sanna. Om du tycker ett påstående är sant så motivera det. Om du tycker det är falskt förklara även då varför t ex genom ett motexempel.

1) En heap som innehåller n element kan i värsta fall ha höjden n.

2) Den array som representerar en minheap utgör en växande följd av element.

3) Om man går på en gren från roten och ner mot ett löv i en minheap så bildar de element man passerar en icke-avtagande följd.

b)Visa genom exempel och förklara hur ”breadh first” algoritmen (kortaste vägen i en oviktad graf) fungerar.