Tentamen i Algoritmer & Datastrukturer i Java

Hjälpmedel: Skrivhjälpmedel, miniräknare.
Ort / Datum: Halmstad / 2007-03-13
Skrivtid: 4 timmar
Kontakt person: Nicolina Månsson, tel. 035-167487
Poäng / Betyg: Max poäng 43
_ >= 17 poäng ger betyg 3
_ >= 26 poäng ger betyg 4
_ >= 35 poäng 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 (1p+2p+2p+1p+2p)

Studera följande algoritm:

```
public static <T extends Comparable<T>> void förvirra(T[] a)
{
for( int p = 1; p < a.length; p++ ){
T tmp = a[ p ];
int j = p;

for( ; j > 0 && tmp.compareTo( a[ j - 1 ] )<0; j-- )
a[ j ] = a[ j - 1 ];
a[ j ] = tmp;
}
}
```

a) Vad gör algoritmen? Beskriv den genom att ge ett exempel.
b) Ange en uppskattning av exekveringstiden i form av Big-Oh notation. Motivera svaret.
c) Hur påverkas exekveringstiden för algoritmen om input storleken dubbleras?
d) Kan en preliminär ordning av elementerna i arrayen påverka exekveringstiden? Motivera.

e) Följande algoritm beräknar om arrayen **a** innehåller två nummer vars summa är **k**. Ange en uppskattning av exekverigstiden för algoritmen. Motivera dit svar.

```
static boolean sum2ToK (int [] a, int k){
for(int i = 0; i <= a.length/2+1 ;i++)
    if(search(k-a[i],a,i))return true;
return false;
}

static boolean search(int x, int[]a, int ix){
for(int i = 0; i<a.length;i++)
     if(x==a[i])
         return i!=ix;

return false;
}
```

Uppgift 2 – Datastrukturer (4p+ 6p+4p+4p)

a)I java standard finns det definierat klassen ArrayList med metoderna
insert(<AnyType> e), AnyType get(), size(), Iterator iterator () mm ....

För att skriva ut alla element i en lista av Integer element kan vi använda följande loop:

```
void printAll(ArrayList<Integer> li){
  for ( int i=0;i<size() i++)
    System.out.println(li.get(i));
}
```

men kan även göra den mer generellt genom att använda listans egen Iterator-objekt.

a1) Komplettera nedanstående metod printAll() så att den använder Iterator- objektet. Kom ihåg att ett Iterator-objekt har metoderna hasNext() och next().

```
void printAll(ArrayList<Integer> li){

   // din kod här
}
```

a2) Om man vill skriva ut innehållet av en lista i omvänd ordning kan man använda som hjälp en annan datastruktur. Vilken? Implementera sedan metoden med hjälp av din tänkta datastruktur.

```
void printReverse(ArrayList<Integer> li )
  {
      // din kod här
  }
```

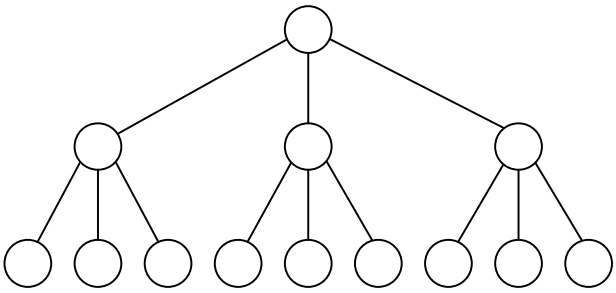
b) När man skriver program för enheter med begränsat minne är det viktigt att vi ”skräddarsyr ” egna datastrukturer som lämpas bäst för respektive uppgift. Implementera en enkel länkad lista av heltal där en Node definieras som:

```
class Node
 {
   int varde;
   Node next;

public Node(int ivarde, Node n)
 {
 varde=ivarde;
 next=n;
 }
}
```

och de enda metoderna är:
-insertFirst()
-removeFirst()
-insertLast()
-removeLast()

c) Vid ett tillfälle ville man implementera en *heap* där varje nod kan ha max tre barn istället för två (som i den binära heapen). Som ni vet lagras elementen för en binär heap i en array med start på position 1 och där barnen (för varje nod x) går att nå på position 2x resp. 2x+1. Er uppgift är att bestämma **arraypositionerna** för varje barn för noden **x** i heapen (motsvarande *2x* reps. *2x+1* i den binära heapen). En heap med tre barn kan illustreras som följande:



d) I vissa specialfall kan man utnyttja en enklare implementering av en prioritetskö än heap utan att förlora i effektivitet. Antag t ex att vi har en mängd element som tilldelas prioritet i form av ett heltal i ett intervall 0..k, där 0 anger högsta prioritet och k lägsta. Vi kan då bilda k+1 vanliga köer, en för varje prioritet. Nya element sätts in i den kö som deras prioritetstal anger. Vid borttagning tar man första element i den första icke-tomma kön.

Uppgiften går ut på att implementera en sådan prioritetskö enligt följande specifikation:

```
public class SimplePrioQueue<E> {

private Queue<E>[] qs;

/** Skapar prioritetskön bestående av k+1 tomma köer */
public SimplePrioQueue(int k) {...}

/** Sätter in elementet med prioriteten prio. Om prio har
orimligt värde genereras IllegalArgumentException. */
public void insert(int prio, E element) { ... }

/** Returnerar element med högsta prioritet; null om priokön
är tom */
public E deleteMin() {...} }
```

Implementera **konstruktorn** samt operationerna
insert() och **deleteMin()**

Du kan utgå ifrån att det finns en klass **Queue<E>**
med följande operationer:

```
public Queue(); // skapar en tom kö
public boolean isEmpty(); // undersöker om kön är tom
public void enqueue(E e); // lägger in e sist i kön
public E dequeue(); // tar ut första elementet i kön; null om tom
```

Uppgift 3 – Kända algoritmer (4p+4p+2p+2p)

a) När en dator beräknar ett aritmetiskt uttryck, omvandlas först uttrycket från infixnotation till postfixnotation. Visa tydligt steg för steg hur denna omvandling går till! Visa stacken hur den förändras för varje tecken i uttrycket. Använd följande uttryck när du visar:
$(2+5\cdot(8-9)+3\cdot(6-4))\cdot 4$
Ledning!
Operatorstacken används på följande sätt: När en operand(7..) syns ska den läggas direkt i en lista , när en operator(+..) syns, alla operatorer med högre prioritet ”popas” från stacken och läggs i listan, därefter ska den nya operatorn ”pushas” på stacken.

b) Vilken datastruktur är lämplig att använda i en algoritm som ska avgöra i fall parenteserna matchar varandra i en sträng med blandade parentestyper. Exempel: strängen "ab([c(d)])" har matchande parenteser, men "(ab(v[d])" har inte det. Implementera en enkel algoritm som använder den tänkta datastrukturen och testar om strängen är rätt ”paranteserad”.
Metoden har följande signatur: **public static boolean isPranteserad(String uttryck)**

c)Anta att du sorterar en stor array av heltal genom att använda mergesort. Därefter använder du binärt sökning för att fastställa om ett visst heltal finns eller inte i arrayen. Slutligen skriver du ut alla heltal i arrayen.
Rangordna de tre algoritmerna utifrån den uppskadade exekveringstiden (BigOh).

d) Både quickSort och mergeSort är så kallade ”underkvadratiska” sorteringsalgoritmer. Varför kallas de ”underkvadratiska”. Förklara kort sorteringsprincipen för dessa två algoritmer.

Uppgift 4 – Träd och graph(1p+2p+ 2p+2p+2p)

a) Förklara skillnaden mellan ett binärt träd och ett binärt sökträd. Ge exempel.
b)Vissa för både obalancerad och balancerad träd, hur det binära sökträdet (tom i början) utvecklas när du gör ”insert” av följande värde: 3, 1, 4, 6, 9, 2, 5 och 7.
c) I ett binärt sökträd hittas det minsta värdet i trädet allra längst till vänster i trädstrukturen. Implementera den metoden ***findMinInSearchTreeNode*** som returnerar det minsta värdet i trädet.

```
static BinaryNode findMinInSearchTreeNode( BinaryNode n){

  // här din kod

}
```

Obs! Se nedan definitionen av en BinaryNode.

```
class BinaryNode
{
   Obect element;       // The data in the node
   BinaryNode left; // Left child
   BinaryNode right; // Right child

BinaryNode( Object theElement, BinaryNode l, BinaryNode r){
   element = theElement;
   left = l;
   right = r;
}
}
```

d) Nedan finns en graf (se nedan) som representerar geografiskt skilt placerade platser (städer). Denna graf kan används för att räkna ut den kortaste vägen mellan två platser. Vilken eller vilka grafalgoritmer är lämpliga att använda i detta fallet? Motivera ditt/dina svar.

e) Det finns två mycket kända sätt att representera grafer. Vilka? Visa representationen av grafen ovan i båda fallen.