

Tentamen – extra del

Förklaringar

Den här delen av tentamen gör möjligt att få betyg högre än E

Utöver den obligatoriska delen, kan studenten göra även den här delen av tentamen. Denna del har sin betydelse bara i fall att studenten har klarat den obligatoriska delen. I så fall kan studenten samla tillräckligt antal poäng på den här delen, och uppnå ett betyg som är högre än E.

Antalet poäng och betyg

Totalt: 20 poäng

För betyget D räcker med: 4 poäng

För betyget C räcker med: 8 poäng

För betyget B räcker med: 14 poäng

För betyget A räcker med: 17 poäng

Uppgifter

Uppgift 1 (4 poäng + 3 poäng)

En algoritm, som sorterar en sekvens med heltal, kan illustreras i samband med en konkret sekvens:

7 8 6 1 4 3 2 5

1 8 6 7 4 3 2 5

1 2 6 7 4 3 8 5

1 2 3 7 4 6 8 5

1 2 3 4 7 6 8 5

1 2 3 4 5 6 8 7

1 2 3 4 5 6 8 7

1 2 3 4 5 6 7 8

a) Skapa en metod `sort` som tar emot en heltalsvektor, och sorterar den enligt givna algoritmen.

b) Låt n beteckna antalet heltal som sorteras. Bestäm i så fall tidskomplexiteten för algoritmen i värsta fall när det gäller antalet elementutbyten. Kategorisera motsvarande komplexitetsfunktion: till vilken θ -mängd tillhör den?

Uppgift 2 (3 poäng)

Komplexitetsfunktioner för flera användbara algoritmer tillhör mängden $\theta(n^2)$. Flera sorteringsalgoritmer, till exempel, är $\theta(n^2)$ -algoritmer när det gäller tiden.

Men hur kan man avgöra om en komplexitetsfunktion tillhör denna mängd: definiera mängden $\theta(n^2)$ med matematisk precision.

Uppgift 3 (4 poäng + 3 poäng + 3 poäng)

Klassen `Vector` representerar en vektor, som kan anpassas till olika typer av element:

```
public class Vector<E>
// E - typ av element
{
    // vektorns element
    private Object[] elements;
```

```
// antalet element i vektorn
private int    countElements;

// Vector skapar en vektor med en given startkapacitet
public Vector (int initialCapacity)
{
    elements = new Object[initialCapacity];
    countElements = 0;
}

// toString returnerar vektorns strängrepresentation
public String toString ()
{
    StringBuilder    s = new StringBuilder ("[");
    for (int pos = 0; pos < countElements - 1; pos++)
        s.append (elements[pos] + ", ");
    if (countElements > 0)
        s.append (elements[countElements - 1]);
    s.append ("]");

    return s.toString ();
}

// add lägger till ett givet element i vektorn
// koden saknas här

// get returnerar det element som finns på en given position
// koden saknas här
}
```

En vektor skapas och används så här:

```
Vector<Integer>    v = new Vector<Integer> (2);
v.add (new Integer (10));
v.add (new Integer (20));
v.add (new Integer (30));
v.add (new Integer (40));
// v.add (new Double (50)); // (1)
System.out.println (v);

Integer    n = v.get (3);
System.out.println (n);
```

När detta kodavsnitt exekveras, skapas följande utskrift:

```
[10, 20, 30, 40]
40
```

Om satsen (1) inkluderas, uppstår ett kompileringsfel: bara element av typen `Integer` kan lagras i vektorn `v`.

- Implementera metoden `add`.
- Implementera metoden `get`.
- Rita det objekt som refereras med referensen `v`.