Praktische Informatik II SS 2007

Übungsblatt 2

Abgabe: 7.5.2007

Aufgabe 1: Sparen, aber ordentlich (70%)

Nachdem der Akademische Senat weitere Einsparungen nicht mittragen will, laufen nun also beim Senator für Bildung und Wissenschaft die Vorbereitungen von neuen Finanzplänen. Im dortigen Hochschuldezernat wurde ein Array mit Studiengängen erstellt, in denen gekürzt werden soll. Dabei hat der zuständige Sachbearbeiter die Studiengänge in der Reihenfolge in das Array geschrieben, wie sie ihm in den Kopf gekommen sind. Eigentlich sollte aber eine alphabetisch sortierte Liste entstehen.

Aufgabe 1:.1 Die richtige Ablagestrategie ist alles!

Der Sachbearbeiter schlägt folgenden Algorithmus zum Sortieren vor:

```
static void sachbearbeiter(String studiengang[])
{
   boolean tauschen;
   String merker;
   do
   {
      tauschen = false;
      for(int i = 1; i < studiengang.length; i++)
      {
         if(studiengang[i].compareTo(studiengang[i-1]) < 0)
         {
            merker = studiengang[i];
            studiengang[i] = studiengang[i-1];
            studiengang[i-1] = merker;
            tauschen = true;
         }
      }
    }while(tauschen);
}
```

Sein Chef bevorzugt hingegen diesen Algorithmus:

```
static void chef(String studiengang[])
{
   String merkerFeld;
   int merkerIndex;
   for(int i = 0; i < studiengang.length; i++)
   {
      merkerFeld = studiengang[i];
      merkerIndex = i;
      for(int j = i + 1; j < studiengang.length; j++)
      {
         if(studiengang[j].compareTo(merkerFeld) < 0)
         {
            merkerIndex = j;
            merkerFeld = studiengang[j];
         }
      }
      studiengang[merkerIndex] = studiengang[i];
```

```
      studiengang[i] = merkerFeld;
   }
}
```

Natürlich wird in diesem Fall der zweite Algorithmus verwendet. `:-)`
Ist der zweite Algorithmus wirklich besser? Bestimmt die Komplexität der beiden Algorithmen für den schlechtesten Fall, um die Lösung mit der geringeren Komplexität zu empfehlen.

Hinweis: Wir nehmen an, dass $O(compareTo()) = O(1)$ ist.

Aufgabe 1:.2 Sparen, koste es was es wolle

Um das Sortieren noch schneller und damit günstiger zu vollziehen, wird zusätzlich eine Unternehmensberatung beauftragt, ein Verfahren vorzuschlagen (Das ist modern und gibt ausserdem Gelegenheit, sich in einer Presseerklärung zu feiern). Als Ergebnis einer gründlichen Untersuchung wird folgender Algorithmus empfohlen:

```
static void consulting(String studiengang[], int anfang, int ende)
{
   int mitte;
   if(anfang < ende)
   {
      mitte = (anfang + ende) / 2;
      consulting(studiengang, anfang, mitte);
      consulting(studiengang, mitte + 1, ende);
      externalConsulting(studiengang, anfang, mitte, ende);
   }
}

static void externalConsulting(String studiengang[], int anfang, int mitte, int ende)
{
   int anfang1 = anfang;
   int ende1 = mitte;
   int anfang2 = mitte + 1;
   int ende2 = ende;

   String[] hilf = new String[studiengang.length];
   int i = anfang1;

   while((anfang1 <= ende1) && (anfang2 <= ende2))
   {
      if(studiengang[anfang1].compareTo(studiengang[anfang2]) < 0)
         hilf[i++] = studiengang[anfang1++];
      else
         hilf[i++] = studiengang[anfang2++];
   }
   while(anfang1 <= ende1)
      hilf[i++] = studiengang[anfang1++];
   while(anfang2 <= ende2)
      hilf[i++] = studiengang[anfang2++];

   for(i = anfang; i <= ende; i++)
      studiengang[i] = hilf[i];
}
```

Da dieser Algorithmus rekursiv funktioniert, kann die Komplexität nicht so einfach bestimmt werden. Geht folgendermaßen vor: Bestimmt zuerst die Komplexität der Methode
`externalConsulting()`. Überlegt euch dann, wie häufig der Grundalgorithmus ausgeführt wird (denkt dabei an die binäre Suche!) und bestimmt auf dieser Basis die

Komplexität des Algorithmus. Haben die Experten der Unternehmensberatung wirklich recht?

Hinweis: Wir nehmen an, dass $O(String[]\ hilf = new\ String[studiengang.length]) = O(1)$ ist.

Aufgabe 2: Gut, schlecht, mittel? (30%)

Überlegt euch für jeden der drei Algorithmen zusätzlich, wie die Komplexität im besten und im durchschnittlichen Fall aussieht. Vergleicht die Algorithmen auf dieser Basis nochmal.