Sortieralgorithmen in C#

Sortieralgorithmen gehören zu den wichtigsten und am häufigsten verwendeten Verfahren in der Informatik.

Ausgangspunkt ist ein unsortiertes Feld, z. B.:

41	5	12	3	10	8

Nachdem der Sortieralgorithmus angewendet wurde, ist das Feld nach einem Kriterium geordnet, z. B. nach ihrer Größe. Als Ergebnis erhält man für das oben dargestellte Feld:

3	5	8	10	12	41

Im Folgenden wird auf einfache Sortierverfahren eingegangen, die insbesondere für kleinere Datenmengen gut geeignet sind. Für größere Datenmengen sind andere Verfahren besser geeignet, diese sind aber wesentlich komplizierter.

a) Selection Sort (Sortieren durch Auswahl)

Dieses Sortierverfahren basiert auf der Maximum- bzw. Minimumsbestimmung vom Blatt Felder.

Idee:

Eingabe: unsortiertes Feld genannt feld mit n Elementen (erstes Element `feld[0]`)

Verfahren:
Es wird die Position (minpos) des kleinsten Elementes aus `feld[0],…, feld[n-1]` bestimmt. Dann vertauscht man `feld[0]` und `feld[minpos]`. Dann wird aus den übrig bleibenden Elementen `feld[1], … , feld[n-1]` wieder die Position des kleinsten bestimmt und dieses Element mit den an Position `feld[1]` vertauscht. Dies wiederholt für alle weiteren Elemente des Feldes.

Ausgabe: sortiertes Feld `feld[0] … feld[n-1]`

Umsetzung in C#:

```
void SortButtonClick(object sender, System.EventArgs e)
{
      // Listboxen löschen
      listBox1.Items.Clear();
      listBox2.Items.Clear();

      // Anzahl aus TextBox einlesen
      int anzahl=int.Parse(textBox1.Text);

      // Feld erzeugen
      int[] feld=new int[anzahl];

      // Feld mit Zufallszahlen (mit Hilfe der InfoKursTools) füllen
      // und in 1. Listbox ausgeben
      for(int i=0;i<feld.Length;i++)
      {
            feld[i]=randomNumbers1.Next();
            listBox1.Items.Add(feld[i]);
      }

      // eigentliches Sortierverfahren: Selection Sort
      for(int i=0;i<=feld.Length-2;i++)
      {
            int minpos=i;
            for(int j=i+1;j<=feld.Length-1;j++)
            {
                  // Wert an der aktuellen Feldposition j ist kleiner als
                  // Wert an Position minpos => neues Minimum
                  if(feld[j]<feld[minpos])
                  {
                        minpos=j;
                  }
            }

            // vertausche Wert an Position i mit Wert an Position minpos
            int hilf=feld[i];
            feld[i]=feld[minpos];
            feld[minpos]=hilf;
      }

      // Ausgabe des sortieren Feldes in 2. Listbox
      for(int i=0;i<feld.Length;i++)
      {
            listBox2.Items.Add(feld[i]);
      }
}
```

Den Aufwand, den man für ein Sortierverfahren braucht, schätzt man häufig mit der Anzahl an Vergleichen ab. Bei Selection Sort führt man im ersten Durchlauf n Vergleiche durch, im 2. Durchlauf n-1, insgesamt also: $n+(n-1)+\ldots+2+1 = \frac{1}{2} \cdot n \cdot (n+1) = \frac{1}{2} n^2 + \frac{1}{2} n$ Vergleiche.
Auch wenn das Feld sortiert ist, müssen alle Vergleiche durchgeführt werden. Eine Verbesserung in diesem Bereich bietet das folgende Sortierverfahren:

b) Bubble Sort (Sortieren durch lokales Vertauschen)

Idee:

Es werden ab dem ersten Element ständig zwei benachbarte Elemente des Feldes verglichen und, wenn sie in der falschen Reihenfolge sind, vertauscht. Die Vergleiche und das Vertauschen werden solange wiederholt, bis innerhalb eines Durchlaufs vom ersten bis zum letzten Element keine Vertauschungen mehr durchgeführt wurden.

Umsetzung in C#:

```
// eigentliches Sortierverfahren: Bubble Sort; Rest vom Programm, s. oben

// vertauscht gibt an, ob innerhalb der do-while-Schleife vertauscht wurde
bool vertauscht;
do
{
      // bisher wurde noch nichts vertauscht => deshalb: false
  vertauscht=false;
      // Schleife vom ersten bis zum vorletzten Element
      for(int i=0;i<=feld.Length-2;i++)  // alternativ: i<feld.Length-1
      {
            // Wert an Position i ist größer als der an Position i+1
            // => vertauschen!
            if(feld[i]>feld[i+1])
            {
              int hilf=feld[i];
              feld[i]=feld[i+1];
              feld[i+1]=hilf;
              // jetzt wurde vertauscht => deshalb: vertauscht=true
              vertauscht=true;
            }
      }
}
while(vertauscht==true);  // oder kurz: while(vertauscht)
```

Bubble Sort berücksichtigt eine Vorsortierung des Feldes und in einem sortierten Feld hat man nur n Vergleiche. In einem total unsortierten Feld hat man jedoch noch mehr Vergleiche als bei Selection Sort.

Aufgabe: Vollziehe beide Verfahren mit dem obigen Zahlenbeispiel nach und erstelle passende Struktogramme.

Sortieralgorithmen in C# © Wittye 2017