Matthias Jauernig
Kurs: 03In1

27.04.04
linux-related.de

5. Übungsaufgaben zur LV
Algorithmen & Datenstrukturen

Abgabetermin: Mi, 28.04.04

Laufzeitvergleich der verschiedenen Sortierverfahren

1.) Ausgabe des Programms:

- Windows XP:

	Auswahl	Bubble	Einfuege	Merge	Quick	Heap	Radix	Bit	Stream HD	Stream FD
n=1000	0.48 Sek.	1.633 Sek.	0.21 Sek.	0.04 Sek.	0.02 Sek.	0.03 Sek.	0.02 Sek.	0.01 Sek.	0.03 Sek.	40.789 Sek.
n=2000	1.722 Sek.	6.76 Sek.	0.831 Sek.	0.04 Sek.	0.03 Sek.	0.071 Sek.	0.02 Sek.	0.01 Sek.	0.05 Sek.	49.751 Sek.
n=3000	3.665 Sek.	14.851 Sek.	1.973 Sek.	0.06 Sek.	0.04 Sek.	0.1 Sek.	0.02 Sek.	0.02 Sek.	0.07 Sek.	53.628 Sek.
n=4000	6.85 Sek.	26.078 Sek.	3.735 Sek.	0.08 Sek.	0.07 Sek.	0.161 Sek.	0.04 Sek.	0.03 Sek.	0.11 Sek.	54.348 Sek.
n=5000	10.886 Sek.	42.381 Sek.	5.858 Sek.	0.121 Sek.	0.08 Sek.	0.17 Sek.	0.05 Sek.	0.03 Sek.	0.13 Sek.	63.121 Sek.

- ROCKLinux (KDE3.1):

	Auswahl	Bubble	Einfuege	Merge	Quick	Heap	Radix	Bit	Stream HD	Stream FD
n=1000	0.615 Sek.	2.195 Sek.	0.294 Sek.	0.065 Sek.	0.026 Sek.	0.045 Sek.	0.041 Sek.	0.014 Sek.	0.037 Sek.	0.053 Sek.
n=2000	2.164 Sek.	8.455 Sek.	1.13 Sek.	0.056 Sek.	0.043 Sek.	0.086 Sek.	0.03 Sek.	0.013 Sek.	0.065 Sek.	0.073 Sek.
n=3000	4.832 Sek.	19.282 S...	2.514 Sek.	0.069 Sek.	0.059 Sek.	0.129 Sek.	0.043 Sek.	0.021 Sek.	0.088 Sek.	0.961 Sek.
n=4000	8.644 Sek.	34.881 S...	4.645 Sek.	0.103 Sek.	0.073 Sek.	0.187 Sek.	0.054 Sek.	0.028 Sek.	0.128 Sek.	0.836 Sek.
n=5000	17.835 S...	65.618 S...	8.267 Sek.	0.15 Sek.	0.11 Sek.	0.257 Sek.	0.098 Sek.	0.037 Sek.	0.167 Sek.	0.908 Sek.

Die Laufzeit von StreamSort auf der Diskette ist sehr gering im Gegensatz zu Windows. Dies ist allerdings mitnichten ein Fehler im Algorithmus, sondern rührt einfach von der Eigenschaft her, dass unter Linux alle blockorientierten Devices gemountet/-unmountet werden müssen. Windows muss die Daten sofort auf Diskette schreiben, bei Linux werden sie hingegen zunächst in den Hauptspeicher geschrieben, bevor sie zu günstigen Zeitpunkten bzw. spätestens beim Unmounten auf Diskette „geflushed“ werden. Für solche Zwecke wie StreamSort ist dieses Verhalten natürlich optimal…

2.) Quelltext von Dialog_Sort.java (gekürzt):

Bitte beachten: so wie die Methode Btn_actionPerformed() hier aufgeführt ist, ist sie nicht kompilierbar. JBuilder9 verlangt nach Exception-Behandlungen z.B. bei der Initialisierung von outFD, outHD oder auch bei outFD.close(), outHD.close(). Aus Platzgründen habe ich diese hier nicht mit aufgenommen, ich hoffe dies tut der Richtigkeit des Algorithmus' erst einmal keinen Abbruch…

```
//...
public class Dialog_Sort extends JDialog {
  /*...*/  int sanz=10;  int dLg=1000;
  String[][] Tab = new String[5][sanz+1];
  String[] TabKopf = new String[] {
      " ", "Auswahl", "Bubble", "Einfuege", "Merge", "Quick", "Heap", "Radix", "Bit",
      "Stream HD", "Stream FD"};
  JTable jT = new JTable(Tab,TabKopf);

  private void jbInit() throws Exception
  { /*...*/  for(int i=1;i<=5;i++)  jT.setValueAt("n="+dLg*i,i-1,0); }

  void LinBtn_actionPerformed(ActionEvent e){/*...*/}
  void WinBtn_actionPerformed(ActionEvent e){/*...*/}

  void Btn_actionPerformed(ActionEvent e, String verzHD, String verzFD) {
    int anz=dLg, i, j, k;  double start, ende;  DataOutputStream outHD, outFD;
    //########## Elementanzahl im Array festlegen ############################
    for(i=1; i<=5; i++,anz=dLg*i) { //von anz=dLg bis anz=5*dLg
      //benötigte Instanzen erzeugen
      File fHD=new File(verzHD+"Mist00.dat");  File fFD=new File(verzFD+"Mist00.dat");
      outHD = new DataOutputStream(new BufferedOutputStream(new FileOutputStream(fHD)));
      outFD = new DataOutputStream(new BufferedOutputStream(new FileOutputStream(fFD)));
      SortArray L[]=new SortArray[sanz-3];
      HeapSortverf L7=new HeapSortverf(anz);
      for(j=0;j<sanz-3;j++)  L[j]=new SortArray(anz);

      for(j=0; j<anz; j++){ //SortArrays mit zufälligen int-Werten füllen
        int zzahl=(int)Math.floor(Math.random()*(anz*2)+1);
        for(k=0;k<sanz-3;k++)
          L[k].add(new ElementtypI(zzahl));          //in SortArrays speichern
        L7.add(new ElementtypI(zzahl));              //in HeapSort
        (new ElementtypI(zzahl)).speichern(outHD);   //in StreamSort HD
        (new ElementtypI(zzahl)).speichern(outFD);   //in StreamSort FD
      }
      outHD.close();  outFD.close();
      StreamSort L8=new StreamSort(fHD,verzHD);  StreamSort L9=new StreamSort(fFD,verzFD);
      //########## Sortieren ####################################################
      for(j=0;j<sanz;j++){
        start=Calendar.getInstance().getTime().getTime();
        switch(j){
          case 0: L[0].AuswahlSort();                break;
          case 1: L[1].BubbleSort();                 break;
          case 2: L[2].EinfuegeSort();               break;
          case 3: L[3].MergeSort();                  break;
          case 4: L[4].QuickSort(0,L[4].count()-1);  break;
          case 5: L7.sortieren();                    break;
          case 6: L[5].RadixSort();                  break;
          case 7: L[6].BitSort(0,L[6].count()-1,16); break;
          case 8: L8.sortieren();                    break;
          case 9: L9.sortieren();                    break;
        }
        ende=Calendar.getInstance().getTime().getTime();
        jT.setValueAt((ende-start)/1000+" Sek.",i-1,j+1);
      }
      //########## prüfen, ob die Sortierungen äquivalent sind #################
      DataInputStream inHD, inFD;
      inHD = new DataInputStream(new BufferedInputStream(new FileInputStream(fHD)));
      inFD = new DataInputStream(new BufferedInputStream(new FileInputStream(fFD)));
      for(j=0;j<anz;j++){
        String S[]=new String[sanz];
        for(k=0;k<sanz-3;k++)
          S[k]=((ElementtypI)L[k].getElement(j)).getSchluessel();//SortArrays
        S[sanz-3]=((ElementtypI)L7.getElement(j)).getSchluessel();//HeapSort
        Elementtyp e1=ElementtypI.laden(inHD);//StreamSort HD
        if(e1!=null) S[sanz-2]=e1.getSchluessel(); else S[sanz-2]="null";
        e1=ElementtypI.laden(inFD);//StreamSort FD
        if(e1!=null) S[sanz-1]=e1.getSchluessel(); else S[sanz-1]="null";

        for(k=0;k<sanz-1;k++)
          if(!S[k].equals(S[k+1]))
          { jOptPan.showMessageDialog(this, "Sortierte Arrays unterscheiden sich!"); return;
      }
      inHD.close(); inFD.close();
    }
```

```
}//...
```