VP WAP: Aufgabe 3
Mathematica

Siegmar Alber
0720046

28.01.2009

Inhaltsverzeichnis

1 Aufgabenstellung **1**

2 Mathematica-Quelltext **2**
2.1 Quicksort 2
2.2 Heapsort 3
2.3 Bubblesort 4
2.4 Main 4
2.4.1 Zufällige Liste 4
2.4.2 Aufsteigend sortierte Liste 5
2.4.3 Absteigend sortierte Liste 5

3 Grafiken **6**
3.1 Zufällige Liste 6
3.2 Aufsteigend sortierte Liste 7
3.3 Absteigend sortierte Liste 7

1 Aufgabenstellung

1. Programmieraufgabe Mathematica:
   (a) Eine Liste von Zufallszahlen erzeugen.
   (b) Sortierfunktionen implementieren:
      i. Quicksort
      ii. Heapsort
      iii. Bubblesort
   (c) Geschwindigkeiten der einzelnen Sortierverfahren mit verschiedenen Listenlängen messen.
   (d) Gemessene Geschwindigkeiten exportieren.
2. Ergebnisse mit gnuplot darstellen.
3. Grafiken in LaTeXeinbinden.
4. LaTeX-Dokument und Mathematica-Quelltext ausdrucken und abgeben.

2 Mathematica-Quelltext

2.1 Quicksort

```
quicksort[liste_] := Module[{list = liste, swap, teile, qsort},

  swap[p1_, p2_] := Module[{temp},
    temp = list[[p1]];
    list[[p1]] = list[[p2]];
    list[[p2]] = temp;
    ];

  teile[links_, rechts_] := Module[{aaa, i, j, pivot},
    i = links;
    j = rechts - 1;
    pivot = list[[rechts]];

    aaa := (
      While[list[[i]] <= pivot && i < rechts,
       i++
       ];
      While[list[[j]] >= pivot && j > links,
       j--
       ];

      If[i < j,
       swap[i, j];
       ];
      );

    aaa[];
    While[i < j, aaa[];];

    If[list[[i]] > pivot,
     swap[i, rechts];
     ];

    i
    ];


  qsort[links_, rechts_] := Module[{teiler},
    If[links < rechts,
      teiler = teile[links, rechts];
      qsort[links, teiler - 1];
      qsort[teiler + 1, rechts]];
    ];

  qsort[1, Length[list]];
  list
```

```
]
```

2.2 Heapsort

```
heapsort[liste_] := Module[{list = liste, swap, blub, macheHeap, sortiere},

  swap[p1_, p2_] := Module[{temp},
    temp = list[[p1]];
    list[[p1]] = list[[p2]];
    list[[p2]] = temp;
    ];

  blub[position_, länge_] :=
   Module[{kindLinks, kindRechts, kind, pos = position, laenge = länge},
    While[pos <= Floor[laenge/2], (
      kindLinks = pos*2;
      kindRechts = kindLinks + 1;

      kind = kindLinks;
      If[kindRechts <= laenge && list[[kindRechts]] > list[[kindLinks]],
       kind = kindRechts];

      If[list[[pos]] < list[[kind]], (
        swap[pos, kind];
        pos = kind;
        ),
       Goto[fertig];
       ];
      )];
    Label[fertig];
    ];

  macheHeap[] := Module[{i},
    For[i = Floor[Length[list]/2], i >= 1, i--,
     blub[i, Length[list]];
     ]
    ];

  sortiere[] := Module[{i},
    For[i = Length[list], i > 1, i--,
      swap[i, 1];
      blub[1, i - 1];
      ];
    ];

  macheHeap[];
  sortiere[];
  list
  ]
```

2.3 Bubblesort

```
bubblesort[liste_] := Module[{list = liste, body, vertauscht, n, swap},

   swap[p1_, p2_] := Module[{temp},
     temp = list[[p1]];
     list[[p1]] = list[[p2]];
     list[[p2]] = temp;
     ];

   body[] := Module[{i},
     vertauscht = False;
     For[i = 1, i <= n - 1, i++,
      If[list[[i]] > list[[i + 1]],
        swap[i, i + 1];
        vertauscht = True;
        ];
      ];
     n--;
     ];

   n = Length[list];
   body[];
   While[vertauscht && n >= 1, body[]];
   list
   ];
```

2.4 Main

2.4.1 Zufällige Liste

```
Module[{laenge, quicksorttime, bubblesorttime, mathematicatime, heapsorttime,
  x, file},
 file = OpenWrite["random.dat"];
 For[laenge = 100, laenge <= 1000, laenge += 100, (
   x = RandomInteger[1000000, laenge];
   heapsorttime = Timing[heapsort[x]][[1]];
   quicksorttime = Timing[quicksort[x]][[1]];
   bubblesorttime = Timing[bubblesort[x]][[1]];
   mathematicatime = Timing[Sort[x]][[1]];
   Print[laenge, "\t", quicksorttime, "\t", heapsorttime, "\t",
    bubblesorttime, "\t", mathematicatime];
   Write[file,
    OutputForm[{laenge/100, laenge, quicksorttime, heapsorttime,
      bubblesorttime, mathematicatime}]];
   )];
 Close[file];
 ]
```

```
100 0.010368 0.025227 0.068933 0.000036
200 0.022529 0.054173 0.27489 0.000039
```

```
300 0.036467 0.117312 0.664839 0.000058
400 0.052316 0.125926 1.13873 0.000073
500 0.062243 0.150897 1.75322 0.000089
600 0.076757 0.19177 2.65083 0.000116
700 0.092454 0.22541 3.53807 0.000126
800 0.107352 0.26148 4.73844 0.000174
900 0.122115 0.296943 5.64427 0.000196
1000 0.133494 0.331511 7.07948 0.000213
```

2.4.2 Aufsteigend sortierte Liste

```
$RecursionLimit = Infinity; $IterationLimit = Infinity;

Module[{laenge, quicksorttime, bubblesorttime, mathematicatime, heapsorttime,
  x, file},
 file = OpenWrite["asc.dat"];
 For[laenge = 100, laenge <= 1000, laenge += 100, (
   x = Sort[RandomInteger[1000000, laenge]];
   heapsorttime = Timing[heapsort[x]][[1]];
   quicksorttime = Timing[quicksort[x]][[1]];
   bubblesorttime = Timing[bubblesort[x]][[1]];
   mathematicatime = Timing[Sort[x]][[1]];
   Print[laenge, "\t", quicksorttime, "\t", heapsorttime, "\t",
    bubblesorttime, "\t", mathematicatime];
   Write[file,
    OutputForm[{laenge/100, laenge, quicksorttime, heapsorttime,
      bubblesorttime, mathematicatime}]];
   )];
 Close[file];
 ]
```

```
100 0.025928 0.028127 0.000644 0.000017
200 0.082041 0.057791 0.001141 0.000027
300 0.171199 0.09024 0.001644 0.000036
400 0.292233 0.125433 0.002263 0.000048
500 0.448394 0.161051 0.002778 0.000056
600 0.630507 0.200557 0.003259 0.000099
700 0.871571 0.241028 0.003861 0.000105
800 1.11449 0.284192 0.004394 0.000124
900 1.38033 0.314423 0.00486 0.000128
1000 1.70624 0.356286 0.005542 0.000127
```

2.4.3 Absteigend sortierte Liste

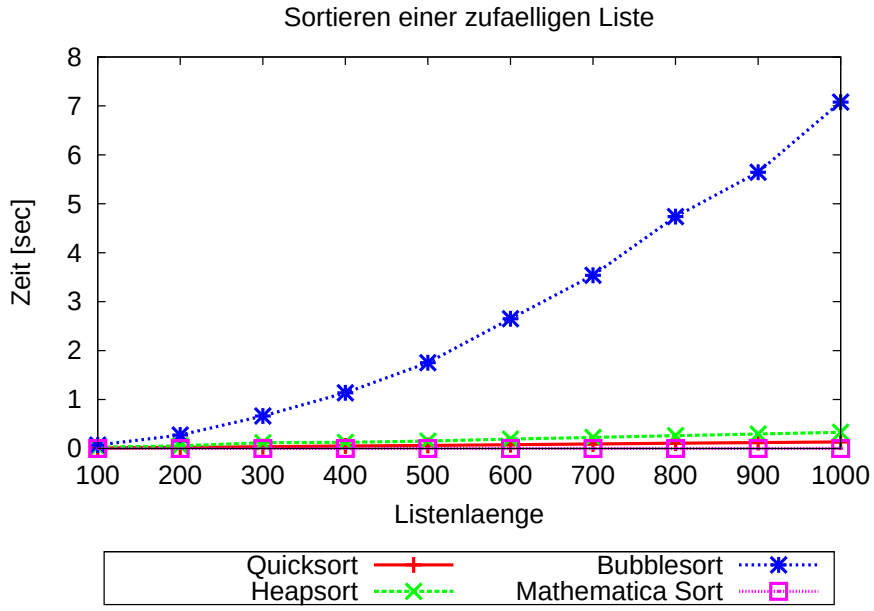
```
Module[{laenge, quicksorttime, bubblesorttime, mathematicatime, heapsorttime,
  x, file},
 file = OpenWrite["dsc.dat"];
 For[laenge = 100, laenge <= 1000, laenge += 100, (
   x = Sort[RandomInteger[1000000, laenge], (#1 > #2) &];
   heapsorttime = Timing[heapsort[x]][[1]];
```

```
  quicksorttime = Timing[quicksort[x]][[1]];
  bubblesorttime = Timing[bubblesort[x]][[1]];
  mathematicatime = Timing[Sort[x]][[1]];
  Print[laenge, "\t", quicksorttime, "\t", heapsorttime, "\t",
   bubblesorttime, "\t", mathematicatime];
  Write[file,
   OutputForm[{laenge/100, laenge, quicksorttime, heapsorttime,
     bubblesorttime, mathematicatime}]];
  )];
 Close[file];
 ]
```

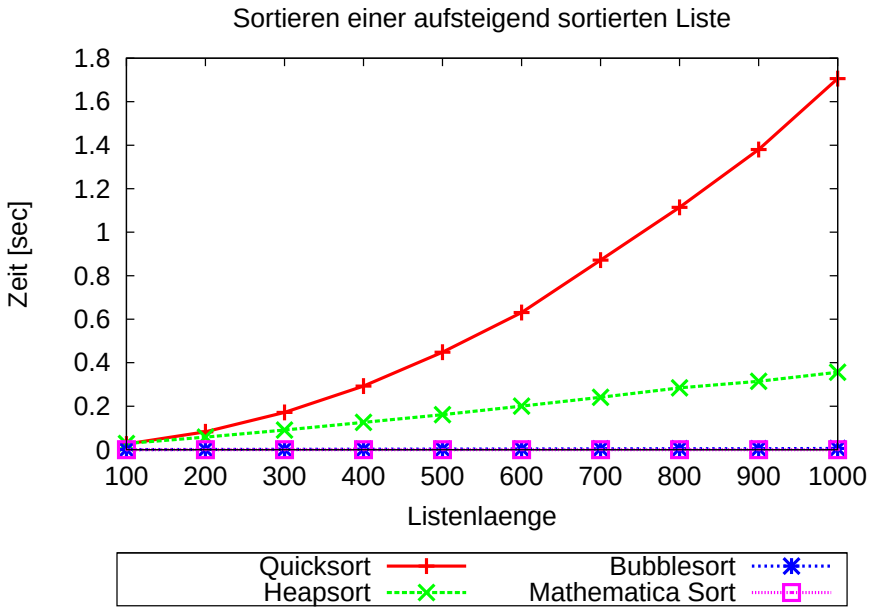
```
100 0.022669 0.020347 0.103164 0.000031
200 0.075599 0.04615 0.409731 0.000033
300 0.158146 0.074343 0.920926 0.000047
400 0.268661 0.101783 1.65807 0.000059
500 0.411633 0.13413 2.56831 0.000066
600 0.576772 0.161089 3.67872 0.000091
700 0.783641 0.193751 5.04629 0.000129
800 1.03289 0.23001 6.6132 0.000112
900 1.29986 0.260036 8.38182 0.000122
1000 1.55788 0.289618 10.3344 0.000144
```

3 Grafiken

3.1 Zufällige Liste



3.2 Aufsteigend sortierte Liste



3.3 Absteigend sortierte Liste