Algorithmen und Datenstrukturen
Übungszettel 3

Martin Avanzini <martin.avanzini@uibk.ac.at>
Thomas Bauereiß <thomas.bauereiss@uibk.ac.at>
Herbert Jordan <herbert@dps.uibk.ac.at>
René Thiemann <rene.thiemann@uibk.ac.at>

29. März, zur Besprechung am 5. April

Aufgabe 1) Quicksort In der Vorlesung wurde der Sortieralgorithmus Quicksort vorgestellt.

1. Erläutern Sie folgende Eigenschaften von Sortieralgorithmen:
   (a) worst-case Komplexität
   (b) average-case Komplexität
   (c) insitu
   (d) stabil
   (e) sequentiell

   Welche Eigenschaften weist Quicksort auf?
2. Finde Permutationen der Liste $[0, \ldots, 9]$ so dass Quicksort eine minimale bzw. eine maximale Laufzeit aufweist. Zu betrachten sind folgende Pivot-Funktionen:
   (a) wähle den Index $\ell + \lfloor \frac{1}{3} \cdot (r - \ell) \rfloor$
   (b) wähle den Index des Medians des zu sortierenden Bereiches

Aufgabe 2) Bucket-Sort und Radix-Sort

1. In der Vorlesung wurde Bucket-Sort mit Eingabe- und Ausgabe-Array vorgestellt. Programmieren Sie eine insitu-Version dieses Algorithmus mit gleichbleibender asymptotischer worst-case Komplexität.

2. Welche Eigenschaften aus Aufgabe 1 gehen in Ihrer Implementierung verloren?

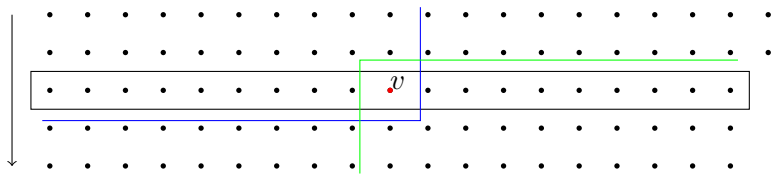
3. Betrachten Sie folgende Implementierung von Radix-Sort.

```
1  import java.util.Arrays;
2
3  public class RadixInplace {
4
5      static int radixBits = 8;
6
7      static void radixSortInplace(int[] a) {
8      int mask = 0;
9      for (int i=0; i<radixBits; i++) {
10         mask = (mask << 1) | 1;
11     }
12     for (int shift=0; shift < 32; shift += radixBits) {
13         bucketSortInplace(a, mask, shift);
14     }
15     }
16
17     static void bucketSortInplace(int[] a,int mask,int shift) {
18     throw new RuntimeException("Methode nicht implementiert");
19     }
20 }
```

   Ist diese Implementierung anhand ihrer Version von Bucket-Sort korrekt? Erläutern Sie Ihre Antwort.

Aufgabe 3) Quick-Select In der Vorlesung wurde eine Version von Quick-Select mit linearer Komplexität vorgestellt, wobei die Pivot-Funktion pivotSelect das Quick-Select Verfahren rekursiv aufruft.

1. Sei (links,rechts) die Partitionierung des Eingabearrays in Elemente kleiner bzw. größer dem mittels pivotSelect bestimmten Pivot-Elementes. Dann gilt $|\text{links}| \geqslant \frac{3}{10} \cdot n - 3$ und $|\text{rechts}| \geqslant \frac{3}{10} \cdot n - 3$. Erläutern Sie dies anhand folgender Abbildung:



2. Vervollständigen Sie die Implementierung von quickSelect.

```
1  public class QuickSelect {
2
3     static void swap(int[] a, int i, int j) {
4        int tmp = a[i];
5        a[i] = a[j];
6        a[j] = tmp;
7     }
8
9     // finded Position des minimalen Elementes
10    static int minimum(int[] a) {
11       int n = a.length;
12       if (n > 0) {
13          int min = a[0];
14          int index = 0;
15          for (int j=1; j<n; j++) {
16             if (a[j] < min) {
17                index = j;
18                min = a[j];
19             }
20          }
21          return index;
22       } else {
23          return -1;
24       }
25    }
26
27    static void quickSelect(int[] a, int x) {
28       if (x < 0 || x >= a.length) {
29          throw new RuntimeException("no x-th element");
30       }
31       swap(a, 0, minimum(a));
32       if (x > 0) {
33          quickSelect(a, 1, a.length-1, x);
34       }
35    }
36
37    static void quickSelect(int[] a, int l, int r, int x) {
38       while (l < r) {
39          int p = pivotSelect(a,l,r);
40          int v = a[p]; // Partition: von hier bis ...
41          int i = l - 1;
42          int j = r;
43          swap(a,p,r);
```

```
44            while (true) {
45               do i++; while (a[i] < v);
46               do j--; while (a[j] > v);
47               if (i >= j) {
48                  break;
49               }
50               swap(a,i,j);
51            }
52            swap(a,r,i); // ... hier gleich zu quickSort
53            if (i > x) {
54               r = i-1;
55            } else if (i == x) {
56               return;
57            } else {
58               l = i+1;
59            }
60         }
61      }
62
63      static int pivotSelect(int[] a, int l, int r) {
64          throw new RuntimeException("Methode nicht implementiert");
65      }
66  }
```