Grundzüge von Algorithmen und Datenstrukturen, WS 15/16: Lösungshinweise zum 2. Übungsblatt

Alexey Pospelov

Aufgabe 2.1 (Lösungshinweise)

a) Es wird das Element $a[n-i]$ in $a[n-i+1..n]$ einsortiert. Dazu werden alle kleineren Elementen eine Stelle nach vorne kopiert und dann $a[i]$ an die richtige Stelle eingefügt:

Insert(i, $a[1..n]$)

```
1: j := n − i + 1
2: h := a[n − i]
3: while j ≤ n and a[j] < h do
4:     a[j − 1] := a[j]
5:     j + +
6: end while
7: a[j − 1] := h
```

b)

Insertion-sort($a[1..n]$)

```
1: for i = 1,...,n-1 do
2:     Insert(i,a)
3: end for
```

c) Invariante: Zu Beginn des i-ten Durchlaufes ist $a[n-i+1..n]$ sortiert.

Für $i = 1$ ist dies richtig, da ein Element immer sortiert ist. Stimmt die Invariante vor Durchlauf i, so auch vor Durchlauf $i+1$, da durch den Aufruf von Insert das Array $a[n-i..n]$ sortiert ist, da wir nach Induktionsvoraussetzung annehmen dürfen, dass $a[n-i+1..n]$ sortiert ist.

d) Insert$(i, a[i])$ macht maximal i Vergleiche. Damit ergibt sich die bekannte Summe $1+2+\cdots+n-1 = O(n^2)$.

Aufgabe 2.2 (Lösungshinweise) Sei

$$M_i = \begin{pmatrix} L_i & L_{i+1} \\ L_{i+1} & L_{i+2} \end{pmatrix}, \quad i \geq 0, \qquad \text{also } M = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}.$$

Damit gilt $v_{i+1} = Mv_i$ und per Induktion $v_{i+1} = M^i v_0$. Für die Multiplikation von zwei (2×2)-Matrizen reichen 4 Additionen und 8 Multiplikationen. Für die Berechnung von

M^{n-1} aus M sind $O(\log n)$ Multiplikationen von (2×2)-Matrizen genug. (Square and Multiply: $M^{2i} = (M^i)^2$ bzw. $M^{2i+1} = M(M^i)^2$.)

Aufgabe 2.3 a) Man sortiert die beiden Arrays in Zeit $O(n \log n)$. Dann muss man nur noch $a[i]$ mit $b[i]$ vergleichen für $1 \leq i \leq n$. Das geht in $O(n)$.

b) Wir modifizieren Mergesort so, dass zusätzlich eine Permutation mitberechnet wird, d.h. Mergesort$(a[1..n], p[1..n])$ sortiert a und zusätzlich steht in $p[i]$ die Stelle, an der das (neue) Element $a[i]$ vor dem Sortieren stand. Dazu ist nur die Prozedur Merge zu ändern, in der die beiden Permutation, die rekursiv ermittelt werden, zusammengefügt werden. Seien p_1 und p_2 diese beiden Permutationen. Wird in der Prozedur Merge $b[k] := a[i]$ ausgeführt (siehe Skript), so setzt man $p[k] := p_1[i]$. Wird stattdessen $b[k] := a[j]$ ausgefürt, so setzt man $p[k] := p_2[j] + t$. (Der Offset t rührt daher, dass das Element aus der zweiten Hälfte kommt, die Prozedur Mergesort in der Rekursion die Grenzen $t + 1..n$ durch $1..n - t$ ersetzt. Ganz klar ist das nicht, da die Semantik nicht ganz geklärt ist. Hier sollte man großzügig sein.)

Hat man nun zwei Permutation, die a und b sortieren, seien dies p und q, so erhält man eine Permutation r, die die beiden Arrays ineinander überführt, indem man $p[i]$ auf $q[i]$ abbildet für $1 \leq i \leq n$, d.h. $r[q[i]] := p[i]$.