58131 Data Structures
III exercise, week 44/2003, English translation

Exercise III.1: Consider stable sorting, which retains the original order between equal input items.

(a) Any sorting algorithm can be made stable, if we can use

$$\mathcal{O}(\text{number of input items})$$

auxiliary storage locations. How?

(b) The lectures gave a version of merge sort on lists which is not stable. However, merge sort can be easily made stable. Give a stable merge sort version for lists.

Exercise III.2: *Denning's sequence* consists or all natural numbers of the for

$$2^i \cdot 3^j \cdot 5^k$$

in strictly ascending order. Hence the sequence begins as follows:

$$\begin{aligned}
1 &= 2^0 \cdot 3^0 \cdot 5^0 \\
2 &= 2^1 \cdot 3^0 \cdot 5^0 \\
3 &= 2^0 \cdot 3^1 \cdot 5^0 \\
4 &= 2^2 \cdot 3^0 \cdot 5^0 \\
5 &= 2^0 \cdot 3^0 \cdot 5^1 \\
6 &= 2^1 \cdot 3^1 \cdot 5^0 \\
8 &= 2^3 \cdot 3^0 \cdot 5^0 \\
9 &= 2^0 \cdot 3^2 \cdot 5^0 \\
10 &= 2^1 \cdot 3^0 \cdot 5^1 \\
12 &= 2^2 \cdot 3^1 \cdot 5^0. \\
&\vdots
\end{aligned}$$

Give an algorithm which gets the number n as input and prints out the first n numbers in this sequence.

Your algorithm must work in $O(n \cdot \log(n))$ steps. (Every multiplication counts as one step.) (*Hint:* Priority queue.)

Exercise III.3: The input consists of a number k and pairwise distinct numbers $a_1, a_2, a_3, \ldots, a_n$.

(a) You must print the k smallest numbers of these a_i. Of course, you could sort the numbers first, and find the answer easily afterwards. But how can you do it faster?

(b) What if you must only find the kth smallest number of these a_i?

(*Hint:* Some parts of an otherwise suitable sorting algorithm are now unnecessary.)

Exercise III.4: Can the worst-case time requirement of the partition subroutine in the quick sort algorithm be as low as

$$o(\text{number of items to partition})$$

steps, or *strictly* faster than linear? Justify your answer.

Exercise III.5: Apply the idea behind radix sorting to putting ASCII strings in alphabetic order.

If the strings are

```
Benny
Bengt
Benedict
```

then we can stop immediately after the 4th characters, because we have then read past their common part. This may be a significant improvement in practice.

Write therefore your string sorter so that it can take this possibility into account. How does your algorithm look like?

Exercise III.6: The Dutch flag consists of 3 horizontal lines:

blue on top

white in the middle

red at bottom.

The corresponding algorithmic problem involves the input items $a_1, a_2, a_3, \ldots, a_n$ and a function color(a_i) which returns whether the given item a_i is **blue**, **white** or **red**. The aim is to rearrange the items so that all the **blue** items come first, then all the **white** items and finally all the **red** items. The relative order of items within the same color does not matter.

(a) What is the connection between this algorithmic problem and making the quick sort algorithm faster?

(b) Solve this algorithmic problem in a way which might be useful as part of the quick sort algorithm.

(Total number of exercises: 6 pcs.)