Exercise 4.1  Questions about sorting algorithms.

Answer the following questions and give a brief explanation of your answer.

a) Is the sorted sequence 1, 2, . . . , n a Min-Heap?

b) When all the elements in a Min-Heap are different, at which positions could the largest element be found?

c) A comparison-based algorithm is called stable if the relative order of identical elements is not changed. If the element '5', for example, occurs twice in an array, then the first 5 is never moved past the second 5. Which of the comparison-based sorting methods that you know are stable, or can easily be adapted accordingly?

d) A sorting algorithm is called in-situ if it works on the input sequence using only a constant amount of additional space for storing parts of the sequence. Which sorting algorithms that you know are in-situ, or can easily be adapted accordingly?

e) In some textbooks, Radixsort is called a “linear-time sorting algorithm”. Does this make sense when sorting sequences without any two identical numbers? Identify a condition as weak as possible such that any sequence of numbers that satisfies this condition is sorted in $O(n)$ steps by Radixsort.

Exercise 4.2  Various topics.

a) The integer multiplication method of Karatsuba/Ofman computes the product of two numbers recursively using a formula that, except for additions and multiplications with the base (here: 10), contains three products. Give two numbers $x$ and $y$ such that these products are $(15 \cdot 86)$, $(87 \cdot 72)$ and $(15 \pm 72) \cdot (87 \pm 86)$.

b) Give a sequence of 5 numbers such that bubble sort needs exactly 10 swaps to sort it.

c) Provide the key pairs that are considered in the first four comparisons when the following array is sorted with insertion sort.

<table>
<thead>
<tr>
<th>9</th>
<th>16</th>
<th>8</th>
<th>13</th>
<th>5</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>14</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
</tbody>
</table>

Please turn over.
Exercise 4.3  _Extended Heaps._

In this exercise we are considering an array \( A[1..|A|] \) representing a Min-Heap. We want to use \( A \) to maintain a set of \( n \) keys. Describe how the following operations can be implemented efficiently (i.e., with a running time in \( O(\log n) \)).

a) **MIN**: Computes the smallest key.

b) **REPLACE(\( i, k \))**: Removes the key \( A[i] \) and replaces it by \( k \).

c) **INSERT(\( k \))**: Inserts a new key with value \( k \) in the heap.

d) **DELETE(\( i \))**: Removes the key \( A[i] \) from the heap.

Note: Of course you have to make sure that the heap property is maintained after each operation. You can assume that \( A \) was chosen large enough to store all occurring keys.

Exercise 4.4  _Lower bounds / Algorithm design._

Consider a set of \( n \) coins that contains exactly one false one. This false coin is heavier than all other ones. To find the false coin you can only use a balance scale. Using this you can only determine whether the coins on the left are lighter, heavier or have exactly the same weight as the coins on the right.

a) Construct a strategy for \( n = 9 \) coins that uses as few weighings as possible.

b) For a general \( n \), provide an algorithm that uses in the worst case exactly \( \log_3(n) \) many weighings.

c) Show that even the best algorithm will use at least \( \log_3(n) - 1 \) many weighings in the worst case.

Note: You can assume that \( n \) is a power of 3.