Practice 1

Exercise levels:
(L1) Reproduce: Reproduce basic facts or check basic understanding.
(L2) Apply: Follow step-by-step instructions.
(L3) Reason: Show insight using a combination of different concepts.
(L4) Create: Prove a non-trivial statement or create an algorithm or data structure of which the objective is formally stated.

▶ Lecture 1 Introduction

Exercise 1 *(L1)* Prof. C. Monster has an array A which he needs to sort. He knows that A contains elements with the keys $\{1, 1, 1, 2, 2, 4\}$, but he does not know in which order. Prof. C. Monster reckons that, because he knows exactly which values are stored in the array, he can simply overwrite them with the values in the right order. Explain why satellite data are sometimes important, and why his approach would lead to problems.

Exercise 2 *(L2)* (This is Exercise 2.1-4 from the book.) Consider the searching problem:

Input: A sequence of n numbers $\langle a_1, a_2, \dots, a_n\rangle$ stored in array $A[1:n]$ and a value x.
Output: An index i such that x equals $A[i]$ or the special value NIL if x does not appear in A.

Write pseudocode for linear search, which scans through the array from the beginning to the end, looking for x. Using a loop invariant, prove that your algorithm is correct. Make sure that the loop invariant fulfills the three necessary properties.

Exercise 3 *(L3)* (This is Exercise 2.3-3 from the book.) State a loop invariant for the while loop of lines 12–18 of the MERGE procedure. Show how to use it, along with the while loops of lines 20–23 and 24–27, to prove that the MERGE procedure is correct.

Exercise 4 *(L3)* (This is part of Problem 2-2 from the book.) Bubblesort is a popular, but inefficient, sorting algorithm. It works by repeatedly swapping adjacent elements that are out of order. The procedure BUBBLESORT sorts array $A[1:n]$.

```
BUBBLESORT(A, n)
for i = 1 to n − 1
    for j = n downto i + 1
        if A[j] < A[j − 1]
            exchange A[j] with A[j − 1]
```

(a) Let A' denote the array A after BUBBLESORT(A, n) is executed. To prove that BUBBLESORT is correct, you need to prove that it terminates and that

$$A'[1] \le A'[2] \le \dots \le A'[n].$$

In order to show that BUBBLESORT actually sorts, what else do you need to prove?

(b) State precisely a loop invariant for the **for** loop in lines 2–4, and prove that this loop invariant holds. Your proof should use the structure of the loop invariant proof in the lecture.

(c) Using the termination condition of the loop invariant proved in (b), state a loop invariant for the **for** loop in lines 1–4 that allows you to prove the inequality above. Your proof should use the structure of the loop invariant proof in the lecture.

▶ Lecture 2 Analysis of algorithms

Asymptotic Notation

Exercise 5

(a) *(L1)* Give definitions of the big-O, big-Ω, and big-Θ notations.

(b) *(L3)* Explain using the definitions why $f(n) = \Theta(n)$ when $f(n) = \Omega(n)$ and $f(n) = O(n)$.

Exercise 6 *(L2)* For each function on the left, $p(n)$, write the letter of a function on the right, $q(n)$, such that $p(n) \in \Theta(q(n))$. If no such function $q(n)$ is listed, then choose (l).

$$f(n) = \sum_{i=1}^{n}(4i-4) \quad \_\_\_\_\_\_\_\_$$

$$g(n) = \sum_{i=1}^{n}\sum_{j=1}^{i} i \quad \_\_\_\_\_\_\_\_$$

$$h(n) = \sum_{i=1}^{\log n} n \quad \_\_\_\_\_\_\_\_$$

$$k(n) = \sum_{i=0}^{n} \frac{4}{2^i} \quad \_\_\_\_\_\_\_\_$$

(a) 1	(g) $\log n$
(b) n	(h) $n \log n$
(c) $n(\log n)^2$	(i) n^2
(d) $n^2 \log n$	(j) n^3
(e) 2^n	(k) 2^{2n}
(f) n^n	(l) no match

Exercise 7 *(L2)* Rank the following functions of n by order of growth (starting with the slowest growing). Functions with the same order of growth should be ranked equal.

$n\sqrt[4]{n}$	$n^{-\log 4}$	$n^{0.00001}$	$\sum_{i=0}^{n} 2^i$	$4^{\log 8}$	$n!$
1	$\frac{\log(2n)}{2}$	$n \log n$	$\log\sqrt{n}$	$\sum_{i=1}^{\log n} i$	n^n

Exercise 8 *(L3)* Formally prove or disprove the following statements using the definitions of the respective asymptotic notations:

(a) $n^2 - 3n + 4 = \Theta(n^2)$

(b) $n^2 \log n = \Omega(n^3)$

(c) $\sqrt{n} = O(\log n)$

(d) $\Omega(n \log n) \cap O(n^2) \neq \emptyset$

(e) If $f(n) = \Omega(h(n))$ and $g(n) = \Omega(h(n))$, then $f(n)g(n) = \Omega(h^2(n))$.

Recurrences

Exercise 9 *(L2)* Using the Master theorem, prove asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for $n \leq 2$.

(a) $T(n) = 16\,T(n/4) + n^2$

(b) $T(n) = 7\,T(n/2) + n^2$

(c) $T(n) = 4\,T(n/2) + n^2\sqrt{n}$

(d) *(L4)* For the solutions of the recurrences (a)–(c), give another recurrence that solves to an exactly the same asymptotic bound, but using a different case of the Master theorem.

Exercise 10

(a) *(L3)* Prove using the substitution method that the solution of $T(n) = T(\lceil n/3 \rceil) + T(\lfloor 2n/3 \rfloor) + n$ with $T(1) = 1$ is $T(n) = O(n \log n)$.

(b) *(L2)* Can the master method be applied to $T(n) = 4\,T(n/2) + n^2 \log\log n$? Why or why not?

(c) *(L3)* Use a recursion tree to get an intuition of what the right upper asymptotic bound for $T(n) = 4\,T(n/2) + n^2 \log\log n$ might be.

(d) *(L3)* Prove the upper asymptotic bound from (c) using the substitution method.

Analysis of Algorithms

Exercise 11 For each of the algorithms below answer the following questions.

```
YUKKURI(m, n)
    Input: m, n ∈ ℕ
    r = 1
    k = n
    while k > 0
        k = k − 1
        r = r · m
    return r
```

```
HAYAKU(m, n)
    Input: m, n ∈ ℕ
    k = n
    r = 1
    h = m
    while k > 0
        if k even
            k = k/2
            h = h · h
        else k = k − 1
            r = r · h
    return r
```

(a) *(L3)* What does the algorithm compute?

(b) *(L2/3)* Prove your claim using a proof by loop invariant.

(c) *(L2)* Analyze the asymptotic running time of the algorithm.

▶ Lecture 3 Heaps

Exercise 12

(a) *(L1)* Give the definition of a max-heap.

(b) *(L2)* Is the sequence $\langle 24, 16, 21, 6, 8, 19, 20, 5, 7, 4\rangle$ a max-heap?

(c) *(L3)* We can store a max-heap in an array. Argue that the k-th node on level j of a max-heap is stored at position $2^j + k - 1$.

Exercise 13 *(L2)* Illustrate the execution of HeapSort on the following input: $\langle 7, 14, 9, 12, 25, 41, 2\rangle$. Show every step that changes the array. (*One call of Max-Heapify counts as one step.*)

Exercise 14 *(L4)* Describe a linear time algorithm MaxHeapVerify(A) that checks whether a given array A is a max-heap. Prove its correctness using the loop invariant, and analyze the running time.