Reminder: \( \log n \) denotes the binary logarithm, i.e., \( \log n = \log_2 n \). We also write \( \log^c n \) as an abbreviation for \( (\log n)^c \).

Make sure to put your name on every piece of paper that you hand in!

1 \( O \)-notation

1. Let \( f : \mathbb{N} \to \mathbb{N} \) be a function. Define the set \( O(f(n)) \).

\[
O(f(n)) = \{ g(n) : \text{There exist positive constants } c \text{ and } n_0 \text{ such that } 0 \leq g(n) \leq cf(n) \text{ for all } n \geq n_0 \}
\]

2. Give a formal proof of the statement:

\[
2n^2 + 5n \in O(n^2)
\]

We need to show that there are positive constants \( c, n_0 \) such that \( 2n^2 + 5n \leq c \cdot n^2 \) holds, for every \( n \geq n_0 \). This inequality is identical to \( 2n + 5 \leq cn \) or \( 5 \leq n(c - 2) \). Chosing \( c = 3 \), we obtain the condition \( 5 \leq n \), which is true for every \( n \geq 5 \). We thus select \( n_0 = 5 \).

3. For each of the following statements, indicate whether it is true or false: (no justification needed)

(a) \( 10n \in O(n \log n) \) \[ \text{true} \]
(b) \( \log^2 n \in O(n^3) \) \[ \text{true} \]
(c) \( \frac{1}{3} \log n \in O(\sqrt{\log n}) \) \[ \text{false} \]
(d) \( n! \in O(\sum_{i=0}^{n} 2^{i}) \) \[ \text{false} \]
(e) \( 2^{\log \log n} \in O(\log^2 n) \) \[ \text{true} \]
(f) \( f(n) \in \Theta(g(n)) \) implies \( g(n) \in \Omega(f(n)) \) \[ \text{true} \]
(g) \( f(n) \in O(g(n)) \) implies \( 2^{f(n)} \in O(2^{g(n)}) \) \[ \text{false} \]
2 Sorting

1. What is the runtime of Insertionsort in $\Theta$-notation (our aim is to sort the input in increasing order) on the following array of length $n$: (no justification needed)

\[ A[i] = 1, \text{ if } 0 \leq i \leq \lfloor \frac{n}{2} \rfloor, \text{ and } A[i] = 0 \text{ otherwise} \]

\[ \Theta(n^2) \]

2. Heapsort interprets an array of length $n$ as a binary tree. What is the runtime of the Build-Heap() operation that transforms the initial binary tree into a heap? (no justification needed) \( \Theta(n) \)

3. Suppose that Mergesort is executed on an array of length $2^k$, for some integer $k$. What is the height of the corresponding recursion tree of this execution in $\Theta$-notation? (no justification needed) \( \Theta(k) \)

3 Loop Invariant

Consider the following algorithm: (it operates on an array $A$ of length $n$ of positive integers)

\begin{verbatim}
Algorithm 1
Require: $A$ is an array of $n$ positive integers
1: $x \leftarrow A[0]$
2: for $i \leftarrow 1, \ldots, n - 2$ do
3: \hspace{1em} $x \leftarrow x \cdot A[i] \cdot A[i+1]$
4: end for
5: return $x$
\end{verbatim}

Consider the following loop invariant:

At the beginning of iteration $i$ (i.e., after $i$ is updated in Line 2 and before the code in Line 3 is executed) the following property holds:

\[ x = \frac{A[0]}{A[i]} \]

1. **Initialization:** Argue that at the beginning of the first iteration, i.e. when $i = 1$, the loop invariant holds.

At the beginning of the first iteration, i.e., when $i = 1$, the loop invariant states $x = \frac{A[0]}{A[1]}$. Observe that in Line 1 of the algorithm, $x$ is initialized to this value. The loop invariant thus holds for $i = 1$.

2. **Maintenance:** Suppose that the loop invariant holds at the beginning of iteration $i$. Argue that the loop invariant then also holds at the beginning of iteration $i + 1$. 

Let $x_i$ denote the value of $x$ at the beginning of iteration $i$. Since the loop invariant holds at the beginning of iteration $i$, we have $x_i = \frac{A[0]}{A[i]}$. Observe that in iteration $i$, the value of $x$ is updated in Line 3. We thus obtain:

$$x_{i+1} = x_i \cdot \frac{A[i]}{A[i+1]} = \frac{A[0]}{A[i]} \cdot \frac{A[i]}{A[i+1]} = \frac{A[0]}{A[i+1]}.$$ 

The loop invariant thus holds at the beginning of iteration $i+1$.

3. *Termination:* What does the algorithm compute? Argue that this follows from the loop invariant.

The algorithm computes the value $\frac{A[0]}{A[n-1]}$. Observe that the state at the end of iteration $n-2$ is identical to the state of a non-existing iteration $n-1$. The loop-invariant thus yields the value $\frac{A[0]}{A[n-1]}$. 