Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Departement of Computer Science
Johannes Lengler, Markus Püschel, David Steurer
Kasper Lindberg, Kostas Lakis, Lucas Pesenti, Manuel Wiedmer

20 October 2025

Algorithms & Data Structures — Exercise sheet 5 — HS 25

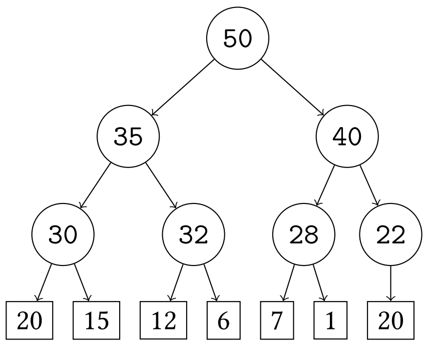
The solutions for this sheet are submitted on Moodle until 26 October 2025, 23:59.

Exercises that are marked by * are challenge exercises. They do not count towards bonus points.

You can use results from previous parts without solving those parts.

Exercise 5.1 *Max-Heap operations* **(1 point)**.

Consider the following max-heap:



Draw the max-heap after two ExtractMax operations.

Exercise 5.2 *Guessing an interval.*

Alice and Bob play the following game:

- Alice selects two integers $1 \leq a < b \leq 200$, which she keeps secret.
- Then, Alice and Bob repeat the following:
  - Bob chooses two integers $0 \leq a' < b' \leq 201$.
  - If $a = a'$ and $b = b'$, Bob wins.
  - If $a' < a$ and $b < b'$, Alice tells Bob 'my numbers are strictly between your numbers!'.
  - Otherwise, Alice does not give any clue to Bob.

Bob claims that he has a strategy to win this game in 12 attempts at most. Prove that such a strategy cannot exist.

Hint: *Represent Bob's strategy as a decision tree. Each edge of the decision tree corresponds to one of Alice's answers, while each leaf corresponds to a win for Bob.*

Hint: *After defining the decision tree, you can show that there is at most one leaf for every non-leaf node and the number of non-leaf nodes is at most $2^n - 1$ for a tree of depth n for $n \in \mathbb{N}_0 = \mathbb{N} \cup \{0\}$.*

Exercise 5.3 *Counting function calls in recursive functions* **(1 point)**.

For each of the following functions g, h, and k, provide an asymptotic bound in big-O notation on the number of calls to f as a function of n. You can assume that n is a power of two.

(a)

Algorithm 1

```
function $g(n)$
    $i \leftarrow 1$
    while $i < n$ do
        $f()$
        $i \leftarrow i + 2$
    $g(n/2)$
    $g(n/2)$
    $g(n/2)$
```

(b)

Algorithm 2

```
function $h(n)$
    $i \leftarrow 1$
    while $i < n$ do
        $f()$
        $i \leftarrow i + 1$
    $k(n)$
    $k(n)$
function $k(n)$
    $i \leftarrow 2$
    while $i < n$ do
        $f()$
        $i \leftarrow i^2$
    $h(n/2)$
```

Exercise 5.4 *Bubble sort invariant* **(1 point)**.

Consider the pseudocode of the bubble sort algorithm on an integer array $A[1, \ldots, n]$:

Algorithm 3 $\textsc{BubbleSort}(A)$

```
for $1 \leq j < n$ do
    for $1 \leq i < n$ do
        if $A[i] > A[i + 1]$ then
            $t \leftarrow A[i]$
            $A[i] \leftarrow A[i + 1]$
            $A[i + 1] \leftarrow t$
return $A$
```

(a) Formulate an invariant INV(j) that holds at the end of the j-th iteration of the outer for-loop.

(b) Using the invariant from part (a), prove the correctness of the algorithm. Specifically, prove the following three assertions:

(1) INV(1) holds.

(2) If INV(j) holds, then INV($j + 1$) holds (for all $1 \leq j < n$).

(3) INV(n) implies that BubbleSort(A) correctly sorts the array A.

Hint: *F*or the induction step in part (2), suppose INV(j) holds, then observe how the largest element in $A[1, \dots, n - j]$ (at the end of the jth iteration) moves throughout the $(j + 1)$th iteration of the outer loop. Where would this element need to be to located in the array in order to satisfy INV($j + 1$)?

Data structures.

Remark. We advise waiting until after the Tuesday lecture to solve this exercise

Exercise 5.5 *Implementing abstract data types.*

In the lecture, you saw how we can implement the abstract data type list with operations insert, get, delete and insertAfter. In this exercise, the goal is to see how we can implement two other abstract data types, namely the stack (german "Stapel") and the queue (german "Schlange" or "Warteschlange"). The abstract data type stack is, as the name suggests, a stack of elements. For a stack S, we want to implement the two following operations; see also Figure 1.

- push(x, S): Add x on top of the stack S.
- pop(S): Remove (and return) the top element of the stack S.

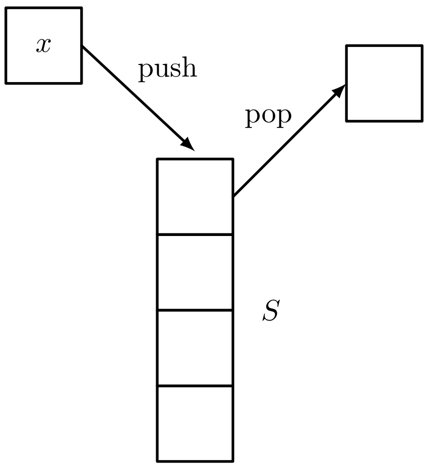


Figure 1: Abstract data type stack

The abstract data type queue is a queue of elements. For a queue Q, we want to implement the following two operations; see also Figure 2.

- enqueue(x, Q): Add x to the end of Q.
- dequeue(Q): Remove (and return) the first element of Q.

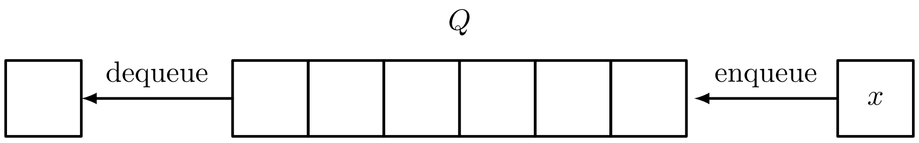


Figure 2: Abstract data type queue

(a) Which data structure from the lecture can be used to implement the abstract data type stack efficiently? Describe for the operations push and pop how they would be implemented with this data structure and what the run time would be.

(b) Which data structure from the lecture can be used to implement the abstract data type queue efficiently? Describe for the operations enqueue and dequeue how they would be implemented with this data structure and what the run time would be.