Marburg University

Department of Computer Science
Prof. Dr. Sebastian Wild

Date: 2026-06-13
Version: 2026-06-14 13:17

Exercise Sheet 9 for Algorithmen und Datenstrukturen (Sommer 2026)

Hand In: *Until 2026-06-20 18:00, on ILIAS.*

Problem 1

20 points

Give an implementation of merge sort on linked lists. Your description should contain an explanation of the idea of the algorithm, pseudocode of the implementation, and arguments of correctness and efficiency.

For full credit, your implementation should only use constant extra space on top of the original input (given as a singly linked list) and it should only use a constant number of memory allocations.

Hint: Use bottom-up mergesort.

Problem 2

30 + 30 points

In the following task, you will be asked to implement a data structure *using stacks*. What this means is the following: your data structure is permitted to store only $O(1)$ extra objects (e.g. integers, etc.), all other memory usage must be within a certain prescribed number of stacks that you are given access to. These stacks have the standard stack operations e.g. push and pop.

a) Implement a standard queue using 2 stacks. This queue must support a push operation and a pop operations. The push operation pushes to the back of the queue, the pop operation pops from the front of the queue. These operations may be interspersed in any way. Each operation must use amortised $O(1)$ operations.

b) Implement a *double ended* queue using 3 stacks. This data structure represents a sequence of values, and supports 4 operations: pushing a new element to the left and to the right of the double ended queue, as well as popping the leftmost or the rightmost element of the double ended queue. These operations may be interspersed in any way. Each operation must use amortised $O(1)$ operations.

Problem 3

10 + 10 + 20 + 20 points

In the magical kingdom of eternal complexity, there is an immense stone statue of the current year in decimal. Every year, you are tasked with updating this statue. A wizard of the kingdom is capable of instantaneously sculpting the replacement digits — but they are too lazy to actually place them appropriately, so it is your task to do this.

Suppose you started doing this in the year 0 (when there is no statue at all), and continued doing this until the year n (inclusive). We are interested in the total number of digit changes you had to do in this time.

Example: For $n = 12$, the following happens. You start at year 0, when there is no statue at all. In each of the years 1, 2, 3, 4, 5, 6, 7, 8, 9, exactly 1 digit change occurs, for a cost of 9 digit changes. In the year 10, 2 digit changes occur, then for years 11 and 12 each, one digit change occurs. Overall, there are $9 + 2 + 1 + 1 = 13$ digit changes.

Hint: It may be useful to consider the following fact: if I have numbers $a_1, \ldots, a_k$ each in range $1, \ldots, v$, then $a_1 + \cdots + a_k$ is equal to the sum of $s_1 + \cdots + s_v$, where s_i is the number of elements a_j with $a_j \geq i$.

a) What is this number for $n = 100$?

b) What is this number for $n = 123134$?

c) Give a asymptotically tight bound for this number in terms of n.

d) What is this number for

$$n = 1529096378904370940?$$

(If you use a program to compute this, please include it in your submission together with an explanation for its operations.)

Problem 4

30 points

Create a data structure which represents a sequence of integers, which supports the following operations, each in $O(1)$ amortised time.

- Add a value v to the beginning of the sequence.
- Add a value v to the end of the sequence.
- Remove the first value from the sequence and return it
- Remove the last value from the sequence and return it
- Combine two sequences, in the following way: If we combine a sequence $x_1, \ldots, x_n$ with a sequence $y_1, \ldots, y_m$, where $n \leq m$, we want to generate the sequence

$$x_1 + y_1, \ldots, x_n + y_m, y_{n+1}, \ldots, y_m.$$

If $n > m$, we proceed symmetrically. This operation *destroys* the original 2 sequences.