Marburg University

Department of Computer Science
Prof. Dr. Sebastian Wild

Date: 2025-11-14
Version: 2025-11-18 21:22

Exercise Sheet 5 for Effiziente Algorithmen (Winter 2025/26)

Hand In: *Until 2025-11-21 18:00, on ILIAS.*

Disclaimer: English translations of our exercise sheets are provided as a best-effort service; in case of doubt, the German versions take precedence.

Problem 1

20 + 20 + 20 points

Consider the following *partial sorting* problem:

> Given an array $A[0..n)$ with n (pairwise distinct) elements, and a number $k \in \{0, \dots, n-1\}$. Order the elements in the array, such that the first k positions in the array contain the smallest k elements in sorted order.

After solving the problem, the following must hold:

$$A[0] \leq A[1] \leq \cdots \leq A[k-2] \leq A[k-1] \quad \text{and} \quad \forall i \in \{k, \dots, n-1\} : A[k-1] \leq A[i].$$

Elements in sorting problems can be large or complex objects. In this exercise, we simply assume that a total order exists and can be found using the comparison operation $<$.

a) Develop an algorithm for this problem. Your solution should have an (expected) runtime of $O(n + k \log n)$.

b) Attempt to develop solutions for the following more advanced questions:

   Question 1: Can you create an algorithm with $O(n + k \log k)$ runtime?

   Question 2: Can you develop a solution that requires only $O(1)$ additional storage space?

   Question 3: Can you specify an algorithm that achieves the runtime in the worst case?

c) Prove that every comparison-based algorithm for the partial sorting problem has a runtime of $\Omega(n + k \log k)$.

   Note that this means that an algorithm for the extended question 1 has an optimal complexity, i.e., has an asymptotically optimal runtime up to constant factors.

Problem 2

50 points

If possible, use the Master Theorem to determine the solutions to the following recurrences. If the Master Theorem is not applicable, explain why.

1. $T(n) = 8T(\frac{n}{2}) + n^3$
2. $T(n) = 4T(\frac{n}{3}) + n \log n$
3. $T(n) = 4T(\frac{n}{2}) + n^2\sqrt{n}$
4. $T(n) = 27T(\frac{n}{3}) + n^3/\log n$
5. $T(n) = 3T(\frac{n}{2}) + n$

Problem 3

10 + 10 + 20 points

In k-way mergesort, we divide the input into k parts. In particular, for $k = 2$, k-way mergesort is identical to standard mergesort.

In the following question, you must analyse k-way mergesort from two points of view: first in terms of the number of comparisons, second in terms of the number of element visits. An element visit occurs both when reading an element or writing an element to the output.

Note: For simplicity, you can assume that there exists $i \in \mathbb{N}_0$ such that $n = k^i$.

a) Set up recurrence relations for the two quantities. For each, provide the most accurate function possible for the "conquer" step of the algorithm.

b) If possible, use the Master Theorem to determine the solutions to the two recurrences. If the Master Theorem is not applicable, explain why.

c) Treat k as variable, and specify the smallest possible upper bound for the recurrence as a function of n and k. Justify your solution (e.g., by iterative substitution). Discuss the solutions to parts b) and c). In particular, address the comparison to the standard mergesort algorithm ($k = 2$).

Problem 4

40 points

In a tournament, n teams compete against each other, with each team playing against every other team. Assume n is a power of two, i.e., $n = 2^k$.

Design an efficient Divide & Conquer algorithm that creates a corresponding match schedule in the form of a $n \times (n-1)$ table, where each row represents a team and each column represents a round. The entry at position (i, j) determines the opponent of team i in round j.

$n = 2$

$_i\backslash^j$	1
1	2
2	1

$n = 4$

$_i\backslash^j$	1	2	3
1	2	4	3
2	1	3	4
3	4	2	1
4	3	1	2

Table 1: Possible match plans for $n = 2$ and $n = 4$.

Examples can be seen in Table 1

Explain the correctness of your algorithm and determine its runtime (in terms of Θ class).