ADS1 Exam / 2022

The exam will consist of:

1. Two questions about an algorithm or a data structure from the lecture – describing the algorithm or data structure, proving they are correct, giving the complexity analysis.
2. Two tasks similar to those from the tutorial – either apply an algorithm / data structure from the lecture to some problem (need to model the problem appropriately) OR adapt the algorithm / data structure to solve some problem (need to understand how it works internally to be able to adapt it appropriately)

The form of the exam is that you will come, get the question sheet, and work on the answers. Once you are finished with one of the answers, you hand it in, I (or one of my colleagues) will read it, point out anything which is missing / incorrect, give you hints if needed, and you can revise it. At some point either you will reach a correct solution, or you won't want to try to improve it again, or I will feel like I can't give you any more hints, and then we'll reach some grade.

Grading

To get a 3 it suffices to know definitions and algorithms / data structures and (with hints) finish at least one of the "type 2" tasks (perhaps suboptimally).

To get a 2 you need to be able to solve the tasks with some hints, or find a suboptimal (complexity-wise) algorithm. If we proved some intermediate claims during the lecture, and these are used in the proofs of correctness/complexity, you need to at least know the statements of these claims.

To get a 1 you need to analyze the algorithms (correctness and complexity) including the proofs, and you need to be able to solve the "type 2" tasks mostly independently. (Advice like "try dynamic programming" and similar is still fine though.)

Topics

Disclaimer: the topics below are NOT specific instances of "type 1" questions. It is clear that some topics are wider and some narrower. However, if you have a good understanding of all the topics below, you should not be surprised by anything during the exam.

- BFS, DFS and their edge classifications
- DFS applications: topological sorting, detecting strongly connected components
- Dijkstra's algorithm, d-ary heaps
- Bellman-Ford's algorithm
- Floyd-Warshall's algorithm (small topic)
- Jarník's algorithm; cut lemma
- Borůvka's algorithm

- Kruskal's algoritmus + Union-Find data structure using trees
- Binary Search Trees (BSTs) in general
- AVL trees
- (a, b)-trees
- Hashing with chaining
- Universal hashing
- Master theorem – analyzing the complexity of Divide & Conquer algorithms using the recursion tree
- Integer multiplication using Karatsuba's algorithm
- MomSelect (finding the k-th smallest element in an array in linear time)
- QuickSelect with random pivots works in expected $\mathcal{O}(n)$ time, QuickSort with random pivots in expected $\mathcal{O}(n \log n)$ time.
- Edit Distance dynamic programming algorithm
- Longest Increasing Subsequence dynamic programming algorithm
- Lower bounds: searching in a sorted array is $\Omega(\log n)$, sorting is $\Omega(n \log n)$ (only applies to deterministic and comparison-based algorithms)