CS340 – Advanced Data Structures and Algorithm Design – Fall 2020
Handout 7 – December 7, 2020

Dr. Sandra Zilles, Department of Computer Science, University of Regina

preparation for final exam

Note. The final exam will take place on Monday, **December 21**, in UR Courses, proctored via ProctorTrack, starting at **9.00am**. The duration of the exam is **120 minutes (2 hours)**. This exam is **closed-book** – no auxiliary material is allowed. Attempted cheating in the exam may lead to a fail grade for the whole course.

Material covered. The material about which exam questions may be asked includes all the material that was covered in lectures or assignments up to and including December 7. Handouts on external sorting and parallel algorithms will not be covered in the exam. For example, this contains material including (but not limited to) the following.

Chapter 1. Algorithm Analysis. Which model do we use to analyze algorithms? How is the running time of an algorithm defined? What is worst case running time, what is average case running time, what is the best case running time? What is amortized running time and how is it calculated? How can we compare two algorithms in terms of running time efficiency? How do different classes of growth rates of functions compare asymptotically? What do the asymptotic notations O, o, Ω, and Θ mean and how are they defined? What are their most important properties? How do we determine which O, o, Ω, Θ classes a given function belongs to? How do we analyze a simple program fragment and determine its running time complexity? How do we determine how often the body of a loop is executed? How does a proof by induction work?

Excursion. Splay Trees. What is a splay tree, how do we implement it? What are the most efficient algorithms for basic operations on splay trees and how did we analyze their running time cost (and with which result)? Illustrate all operations on example trees! How do Splay Trees relate to AVL Trees?

Chapter 2. Priority Queues. What is a priority queue and for what type of application is it needed? What is a binary heap and how do we implement it? What are the most efficient algorithms for basic operations on binary heaps and how did we analyze their running time cost (and with which result)? Illustrate all operations on example binary heaps! How would you answer the previous questions for d-heaps, leftist heaps, skew heaps, binomial queues (instead of binary heaps)?

Chapter 3. Sorting. How does insertion sort work? How did we analyze it (and with which result)? How to illustrate insertion sort for a small example? How does Shellsort work? How to illustrate Shellsort for a small example? What are typical increments used with Shellsort and how do they affect the performance of the algorithm (running time)? How does Heapsort work? How to illustrate it with a small example? How did we analyze Heapsort and with which result? How does Mergesort work? How to illustrate it with a small example? How did we analyze Mergesort and with which result? How does Quicksort work? How to illustrate it with a small example? What do you know about its running time? What typical methods for choosing the pivot are there? What is an in-place sorting algorithm? How do we solve recurrence relations? What is the best lower bound we proved for the running time of any comparison-based sorting algorithm? How did we prove that bound? What is an inversion? What is a stable sorting algorithm?

Chapter 4. Graph Algorithms. What is the basic graph terminology we agreed on? What options are there for representing graphs and when would you use which one? What is a topological sort/ordering, when does one exist, and how to find one? How to solve the SSSPP in the weighted and in the unweighted case? How to solve network flow problems? What is a minimum spanning tree, when does one exist, and which methods did we study for finding one? What in general, is the union-find / disjoint set structure? How does Depth-First Search work? Can you compare it to Breadth-First Search? Do you know applications of either method? What is an Euler circuit? For each of the graph algorithms we discussed, make sure you know its running time, and make sure you can illustrate it on a small example!

Chapter 5. Tractable, Intractable, and Unsolvable Problems. What is the principle of reduction – how does a reduction work and what does reducibility imply? Can you give examples of reductions from class? What

does it mean for a problem to be unsolvable, in P, or in NP? Can you name and define some unsolvable problems, some problems in P, and some problems in NP? What is NP-completeness and do you know any examples of it? How does a polynomial reduction work and what does polynomial reducibility imply?

Chapter 6. Algorithm Design Techniques. Which different algorithm design techniques did we study? Can you name algorithm examples for each technique? What are the dangers when applying parallelism carelessly? What is a greedy algorithm? Can you give examples of when greedy algorithms work and when they don't work? What is a Divide-and-Conquer algorithm and how do we analyze it usually? How does Dynamic Programming work? How to solve the All-Pairs Shortest Path Problem with Dynamic Programming? What is a backtracking method? Can you give an example problem that can be solved with backtracking?

Typical questions. Questions may, for example, be of the following types (not exclusively).

- questions similar to those in the theory parts of the assignments 1 through 8, or to the theory questions in the textbook chapters we covered in class,
- questions asking to explain concepts or results introduced in class,
- multiple-choice or true/false questions,
- questions asking for algorithms in pseudocode, related to tasks discussed in class or in the assignments,
- questions about which data structure or which algorithm to prefer in a particular situation,
- ...

The questions will be of varying difficulty. For most questions it will be possible to achieve partial marks. Even when you have only a partial idea, it is advisable to write it down.