**DMFS 2022**

**– Ugeseddel 6 –**

# General Plan

This week we will continue talking about graphs and graph algorithms. We will finish our discussion of minimum spanning trees in CLRS Chapter 23, then make a detour to learn about the useful heap data structure in CLRS Chapter 6, and finally study how to compute shortest paths in graphs as in CLRS Chapter 24.

As already mentioned, graphs are a truly foundational topic in computer science. They can be used to model all kinds of real-world problems, and efficient graph algorithms therefore have applications that are too numerous to list. Since this is an introductory course in discrete math and algorithms, though, we do not really have time to get into any serious applications at this point (you will see them later during your studies), but will focus on covering some of the basic algorithms used to manipulate these objects.

# Reading Instructions

As usual, I will do my best to try to cover as much as possible of the most important material in class, but it is very important that you also read the textbook, which contains additional material.

- CLRS Chapter 23 about minimum spanning trees (all of it).
- CLRS introduction to Part II (pages 147–150) (especially if you want to get a bit of an overview why we are so interested in sorting data).
- CLRS Chapter 6 about heaps (all of it).
- Parts of CLRS Chapter 24 including the introduction, Section 24.3, and Section 24.5. We will not talk about Sections 24.1–2, but they are warmly recommended reading and there is a high risk you will get exposed to this in your very next algorithms course. We will also not cover Section 24.4 (although it is a cool application).

# Exercises

Some of the exercises below were listed also last week, but I am mentioning them here as well to give an overview of good exercises concerning graphs.

## CLRS Chapter 22: Elementary Graph Algorithms

1. Draw some moderately sized directed graphs and make sure that you can run DFS and BFS on them.

2. Run topological sort on the same algorithms. Check that the algorithm works precisely when your directed graphs are acyclic.

3. Use the algorithm in Sec 22.5 to compute strongly connected components of some of your directed graphs.

4. CLRS Section 22.1 exercises 22.1-3, 22.1-4, 22.1-5, and 22.1-6.

5. CLRS Section 22.2 exercises 22.2-2, 22.2-4, 22.2-7, and 22.2-9.

6. CLRS Section 22.3 exercises 22.3-2 and 22.3-7.

7. CLRS Section 22.4 exercise 22.4-3.

8. CLRS exercise 22-3.

## CLRS Chapter 23: Minimum spanning trees

1. Draw some moderately sized undirected graphs with edge weights (making sure to consider also graphs with several edges of the same weight) and run Prim's algorithm starting from different vertices. Can you find graphs where different starting points yield different MSTs?

2. Draw some moderately sized undirected graphs with edge weights (making sure to consider also graphs with several edges of the same weight) and run Kruskal's algorithm. Can you find graphs for which you get different MSTs depending on which order edges of the same weight happen to be sorted?

3. CLRS Section 23.1 exercises 23.1-1, 23.1-3–23.1-7, and 23.1-9

4. CLRS Section 23.2 exercises 23.2-1 and 23.2-2

5. [*] CLRS exercises 23.2-7 and 23-4

6. CLRS Section 23.2 exercises 23.2-2,

7. [*] In some modern programming languages there are efficient implementations of priority queues in the standard libraries, but without support for DECREASE-KEY (that is, once an element is inserted with a certain priority this cannot be changed). Can you modify Dijkstra's algorithm to run as efficiently as before (asymptotically) even with this kind of priority queue?

## CLRS Chapter 6: Heaps

1. Take some moderately sized arrays filled with numbers (or comparable keys of your choice) and build min-heaps and max-heaps from them.

2. CLRS Section 6.1 exercises 6.1-1–6.1-7

3. CLRS Section 6.2 exercises 6.2-1 and 6.2-3–6.2-6

4. CLRS Section 6.3 exercises 6.3-1–6.3-3

5. CLRS Section 6.4 exercises 6.4-1–6.4-4

6. CLRS Section 6.5 exercises 6.5-1, 6.5-2, 6.5-5, and 6.5-7–6.5-9

7. CLRS exercise 6-1

8. Suppose that a weighted undirected graph $G$ has unique heaviest edge $e$. Could this edge ever be included in an MST? Give an example of how this can happen or prove that it is impossible.

## CLRS Chapter 24: Single-source shortest paths

1. Draw some moderately sized directed graphs with weights on the edges and run Dijkstra's algorithm on these graphs. Work out the details of how the priority queue heap changes during execution.

2. CLRS Section 24.3 exercises 24.3-1, 24.3-2, 24.3-3, 24.3-4, and 24.3-6.

3. CLRS Section 24.5 exercises 24.5-1, 24.5-4, and 24.5-7.

4. Let $T$ be a shortest path tree from a vertex $s$ in a graph $G$. Suppose that we add some constant $c$ to all edge weights in $G$. Is $T$ still a shortest path tree? Prove that this is so or give a counter-example.

5. [*] Let $G$ be a directed graph where all *vertices* have weights, and where the weight of a path in $G$ is the sum of the weights of all vertices on the path. Give an algorithm for computing the shortest path between two vertices $s$ and $t$ in $G$.