IDMA 2024

– Ugeseddel 8 –

General Plan

As the final topic in the course curriculum, we will study how to compute shortest paths in graphs as in CLRS Chapter 22 on Monday.

The last items in the schedule are a question-and-answer session on Wednesday March 26 at 13:15-15:00 followed by a final exercise session with your TAs.

Please think ahead of time if there is anything in particular that you would like to hear about during the Q&A session, and send a message on Absalon if that is the case, so that I can plan accordingly. In the same way, do let your TAs know in advance if you have any specific wishes for the final exercise session.

Otherwise, all that remains is to prepare for the exam (and, of course, to hand in the final problem set in time, so that you are sure to be eligible for the exam).

I hope you have enjoyed this course. **Good luck on the exam!**

Jakob

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.

- Parts of CLRS Chapter 22 including the introduction, Section 22.3, and Section 22.5. We will not talk about Sections 22.1–2, 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 22.4 (although it is a cool application).

Exercises

CLRS Chapter 22: 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 22.3 exercises 22.3-1, 22.3-2, and 22.3-6.

3. CLRS Section 22.5 exercises 22.5-1, 22.5-4, and 22.5-7.

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.

6. [*] 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?