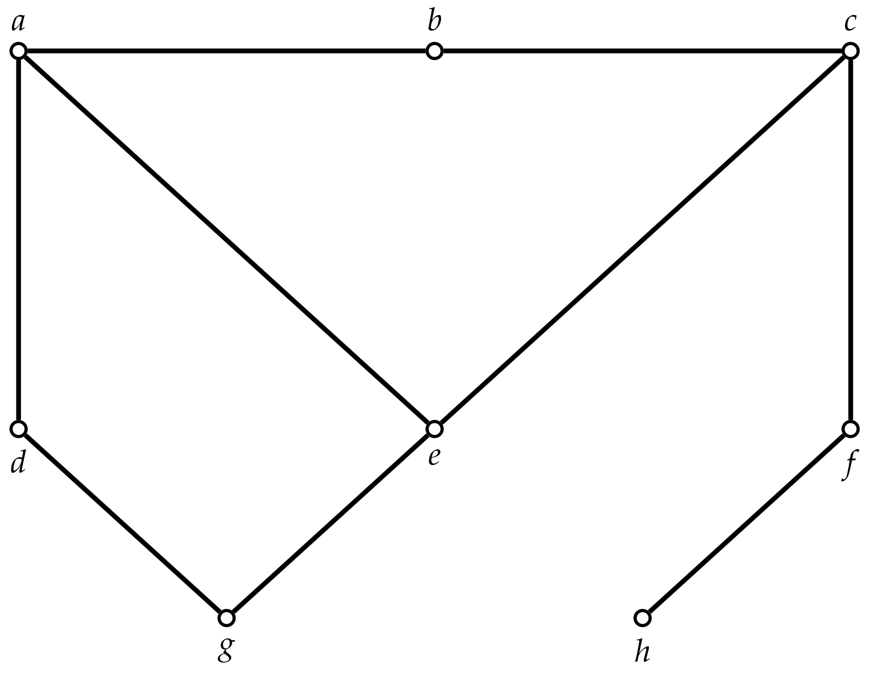
This assignment is **due on October 1** and should be submitted on Gradescope. All submitted work must be *done individually* without consulting someone else's solutions in accordance with the University's "Academic Dishonesty and Plagiarism" policies.

As a first step go to the last page and read the section: Advice on how to do the assignment.

Problem 1. (15 points) Consider the following undirected graph.



a) Is the graph bipartite? If so, give its red and blue sets of vertices. If not, briefly explain why.

b) List all cut edges of the graph.

(You do **not** have to write down the execution of any algorithm you use.)

Problem 2. (30 points) We want to construct a data structure supporting the following operations:

- INSERT(k): insert key k
- GET-MINIMUM(): return the minimum of all keys stored (do not modify the contents of the data structure)
- GET-MAXIMUM(): return the maximum of all keys stored (do not modify the contents of the data structure)
- GET-AVERAGE(): return the average of all keys stored (do not modify the contents of the data structure)

Your task is to come up with a data structure implementation that uses $O(n)$ space, where n is the size of the input. All operations should run in $O(1)$ time. Remember to:

a) Describe your data structure implementation in plain English.

b) Prove the correctness of your data structure.

c) Analyze the time and space complexity of your data structure.

Problem 3. (30 points) We are given a simple undirected graph $G = (V, E)$ and a set of vertices $T \subseteq V$. We want to compute the set of vertices that is reachable from a specific starting vertex s using a path that visits at most k vertices of T ($0 \leq k \leq n$).

Your task is to design an algorithm that computes all vertices G that are reachable from s using a path that contains at most k vertices of T. For full marks your algorithm needs to run in $O(n + m)$ time. Note that k is *not* a constant and your running time should *not* depend on it.

a) Design an algorithm that solves the problem.

b) Briefly argue the correctness of your algorithm.

c) Analyse the running time of your algorithm.

Advice on how to do the assignment

- Assignments should be typed and submitted as pdf (no handwriting).
- Start by typing your student ID at the top of the first page of your submission. Do **not** type your name.
- Submit only your answers to the questions. Do **not** copy the questions.
- When designing an algorithm or data structure, it might help you (and us) if you briefly describe your general idea, and after that you might want to develop and elaborate on details. If we don't see/understand your general idea, we cannot give you points for it.
- Be careful with giving multiple or alternative answers. If you give multiple answers, then we will give you marks only for "your worst answer", as this indicates how well you understood the question.
- Some of the questions are very easy (with the help of the lecture notes or book). You can use the material presented in the lecture or book without proving it. You do not need to write more than necessary (see comment above).
- When giving answers to questions, always prove/explain/motivate your answers.
- When giving an algorithm as an answer, the algorithm does not have to be given as (pseudo-)code.
- If you do give (pseudo-)code, then you still have to explain your code and your ideas in plain English.
- Unless otherwise stated, we always ask about worst-case analysis, worst case running times etc.
- As done in the lecture, and as it is typical for an algorithms course, we are interested in the most efficient algorithms and data structures.
- If you use further resources (books, scientific papers, the internet,...) to formulate your answers, then add references to your sources.
- If you refer to a result in a scientific paper or on the web you need to explain the results to show that you understand the results, and how it was proven.