ASSIGNMENT

INSTRUCTIONS

Deadline: Original deadline is April 10, 2020, by 18:00. Due to Passover, this deadline is postponed up until April 20, 2020, by $18:00$.
Number of students per submission: At most 2 students.

- You are not to consult or discuss in any way shape or form anything related to this assignment with any person that is not your sole partner for this specific assignment.
- Any references you find and use in the literature or the web you are obligated to report in full following the style of the course notes.

PROBLEMS

The message of the first problem in this assignment is that matchings exhibit a behaviour that is reminiscent of that seen for independent sets in linear spaces; in that, any independent set can be extended into a base. The following problem does not assert an extension property as natural as that seen for independent sets, but pleasant nonetheless.

Problem 1. Let G be a graph and let M be a matching in G. Prove that there is a maximum matching M' in G satisfying $V(M) \subseteq V(M')$.
Suggestion. Consider a maximum matching in G that amongst all such matchings maximises $|M' \cap M|$ as well.

Whitney's theorem asserts that a graph is 2-connected if and only if any two of its vertices lie on a common cycle. The following is a generalisation of this result (which we in fact did discuss in class albeit under a different guise).

Problem 2. Prove that a graph is 2-connected if and only if any two of its edges lie on a common cycle (assuming naturally that the graph has at least two edges and no degree zero vertices).

Imagine, if you can, that you are a network designer. As cut-vertices are clear points of weakness of any network, you have an interest in adding cables/connections to the network in order to increase its (local) connectivity. Unfortunately, each such addition is expensive. The following problem is then of clear interest to you.

Problem 3. Devise an algorithm that given a graph returns the least number of (non-)edges that upon adding those to the graph a 2-connected graph is attained (perhaps this number is zero).
Instructions. Organise your answer as follows. First, specify the algorithm using pseudo-code following the style of the course notes; avoid specifying algorithms in a manner akin to programming rendering the algorithm unreadable. Second, prove that the algorithm you are proposing is correct. Third, state and prove the running time of your proposed algorithm.

The following problem concerns the so called *longest path problem* in which one seeks to find a longest (simple) path in a given graph. You will be prompted to design a *conceptual* dynamic-programming inspired algorithm for the longest path problem.

PROBLEM 4. Compile a list of problems all phrased for 2-connected graphs only, such that if magical black-boxes solving all said problems on your list within polynomial time were ever to be provided to you, then you would be able to devise a polynomial time algorithm solving the longest path problem on any graph, even if the latter is not 2-connected.

- Your answer should be comprised of two parts; the list of problems for 2-connected graphs that you would like a magical black-box for, and the algorithm that employs all of these boxes as to solve the problem in arbitrary graphs.
- As the going rate of magical black-boxes is essentially infinite, you are asked to minimise the number of magical black-boxes you seek to employ.
- The list of problems you are to compile for 2-connected graphs can have their input consist of more than a 2-connected graph. A fixed number of prescribed vertices and edges are allowed to be part of the input as well (note that 'fixed' means $O(1)$).

Suggestion. Recall that the blocks (and cut-vertices) of any connected graph can be arranged in a tree.

BONUS

`This part of the assignment is not obligatory`

For any $k \geq 2$, any k-vertices in a k-connected graph lie on a common cycle (see lecture notes). Above you proved that for 2-connected graphs one may replace the word 'vertices' with the word 'edges'. Unfortunately, this is where this line of results ends as far as edges are concerned. Indeed, it is not true that any k-edges in a k-connected graph lie on a common cycle. Trivial obstacles are odd edge cuts and having these edges form at least one star of degree at least 3. For $k = 3$, the answer is quite elegant.

THEOREM. *Any three edges of a* 3*-connected graph lie on a common cycle unless these form a* claw *(i.e.,* $K_{1,3}$*) or an edge-cut in the graph.*

Prove the above theorem.