Feb 17

Warning: some of the problems require thought – do not wait until the last day to start working on them! Please also try working on problems from the book, and discuss them with your partners.

If you cannot come up with algorithms that run in the required time, then provide (correct) slower algorithms for partial credit. Write your answers using pseudo-code in the same style as the textbook. These make the algorithm description precise, and easy to read (as opposed to code in C or some other language).

(1) Write out a pseudo-code description of an algorithm that will write down a list of the vertices that are cut vertices (or articulation vertices) in a given graph \( G = (V,E) \). You may assume that \( G \) is connected. Each cut vertex should be listed exactly once. The algorithm should run in time \( O(|E| + |V|) \).

(2) Given a rooted tree \( T \) with \( n \) nodes, we wish to pre-process the tree in \( O(n) \) time, so that we can answer the following queries in \( O(1) \) time: given a pair of nodes \( u \) and \( v \), is \( u \) an ancestor of \( v \) in the tree? How can we do this?

(3) Design an algorithm that takes as input a directed graph \( G = (V,E) \) and outputs the set of edges that do not have any directed cycle containing them. The algorithm should run in time \( O(|V| + |E|) \).

[Hint: Strong Connectivity!]

(4) Problem 2 (page 107 from book).