1 Topological sort

Do Exercise 22.4-2 (on p. 614).

2 Adjacency matrix of graph

Suppose you are given an undirected graph $G$, which is represented by adjacency matrix $A$. We assume there is no self-loops in $G$, and there is at most one edge between two nodes of $G$. In this problem, we want to find whether there exists three nodes $x, y$ and $z$ in $G$ such that $x, y$ and $z$ are pairwise connected by some edge in $G$ (i.e. there exists an edge between each pair out of the three nodes). A naive algorithm would try all possible $x, y$ and $z$ but this will take $O(n^3)$ (where $n$ is the number of nodes in $G$). Now give a faster algorithm for this problem. Hint: consider multiplying the adjacency matrix $A$ to itself; what will this product $A^2$ tell you?

3 Querying on a tree

You are given a rooted binary tree $T = (V, E)$, along with a designated root node $r \in V$. The size of $V$ is $n$. Recall that a node in a tree keeps tracks of its descendants and its parent node. Also recall that node $u$ is said to be an ancestor of node $v$ in the rooted tree, if the path from $r$ to $v$ in $T$ passes through $u$.

A commonly performed querying on the trees is: given two nodes $u$ and $v$, is $u$ an ancestor of $v$? You wish to preprocess the tree so that queries of this form for any two nodes $u$ and $v$ can be answered in constant time. The preprocessing itself should take linear time. That is, you can spend $O(n)$ time before any query arrives; then you must answer each query like “is node $u$ an ancestor of node $v$?” in constant time.

4 Semi-connected graphs

Do Exercise 22.5-7 (on p.621). To get full credit, your algorithm should run in time $O(|V| + |E|)$. 