Assignment #1 part 2, Total points: 50

(Course: CS 401)

These are the last two problems for Assignment 1.

For regular students, the deadline is **March 11, Monday, in class**.

For special needs students, the deadline is **March 18, Monday, in class**.

No late assignments will be accepted.

Special note: Any answer that is not sufficiently clear even after a reasonably careful reading will not be considered a correct answer, and only what is written in the answer will be used to verify accuracy. No vague descriptions or sufficiently ambiguous statements that can be interpreted in multiple ways will be considered as a correct answer, nor will the student be allowed to add any explanations to his/her answer after it has been submitted.

Problem 1 (35 points): Let $G = (V, E)$ be a **directed** graph and let s and t be two nodes of G. Let n and m be the number of nodes and edges of G, respectively. In the class we say how to decide if there is a path **from** s **to** t, namely, we start a (directed) BFS starting from s and check if t appears among the list of nodes that are visited during BFS. The purpose of the assignment is to decide if such a path exists under some **additional constraints**. Let u and v be two other nodes of G that are **not** s or t.

(*i*) [15 points] Decide if G has a path from s and t that **avoids** using **both** the nodes u and v.

(*ii*) [20 points] Decide if G has a path from s and t that **uses both** the nodes u and v.

Both of your algorithms should run in $O(m + n)$ time. You may assume that the graph is given in its adjacency list representation. If you are using BFS, there is no need to give codes for it; simply saying "do a BFS starting at such-and-such node" will suffice.

Problem 2 (15 points): Give an algorithm to detect whether a given undirected graph is a tree or not. The graph is given to you in its adjacency list representation. The running time of your algorithm should be $O(m + n)$ for a graph with n nodes and m edges.