Analysis of Algorithms: Assignment 7

Due date: November 11 (Thursday)

Problem 1 (3 points)
Using Figure 23.4 in the textbook as a model, illustrate the steps of depth-first search on the undirected graph of Figure 23.3. Assume that the main loop of the algorithm processes the vertices in the alphabetical order, from r to y.

Problem 2 (4 points)
The depth-first search algorithm may be used to identify the connected components of an *undirected* graph. Write a modified version of DFS for performing this task.

Your algorithm must determine the number k of connected components in an undirected graph and return this number. Furthermore, for every vertex u of the graph, the algorithm must assign an integer label $component[u]$, between 1 and k, that denotes the corresponding connected component. If two vertices are in the same component, they must get the same label. On the other hand, if vertices are in different components, their labels must be distinct.

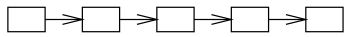
Problem 3 (3 points)
Suppose that you need to construct a minimum spanning tree for a graph represented by an *adjacency matrix,* rather than adjacency lists. Give a modified version of the MST-Prim algorithm for the adjacency-matrix representation; its running time must be $O(V^2)$.

Problem 4 (bonus)
This problem is optional, and it allows you to get 2 bonus points toward your final grade for the course. You cannot submit this bonus problem after the deadline.

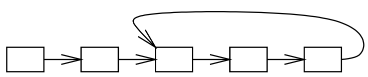
If we need to print all elements of a standard linked list (see Figure 1), then we may write a simple program that starts at the beginning of the list and follows pointers until reaching the end; however, novice programmers sometimes mistakingly create a list whose last element points into the middle of the list (see Figure 2), and then an attempt to print all elements leads to an infinite loop.

Write an algorithm Check-Loop(x) that determines whether a given linked list is "looped." The algorithm's argument x is the first element of the list. If a list is looped (Figure 2), Check-Loop returns true; if the list is not looped (Figure 1), it returns false.

Your algorithm has to run in *linear time.* Furthermore, it must run *in-place* (no extra memory) and preserve the initial contents of the list. These restrictions mean that you *cannot* mark the elements of the list that you have visited, because storing such marks would require a lot of additional memory.



1. Standard linked list.



2. Looped linked list.