Algorithms: Assignment 9

Due date: November 16 (Thursday)

Problem 1 (3 points)
Give an example of a directed graph with negative-weight edges for which Dijkstra's algorithm produces incorrect answers. Your graph must *not* include negative-weight cycles, which would make the shortest-paths problem meaningless.

Problem 2 (3 points)
Suppose that G is a weighted directed acyclic graph, u and v are its vertices, and the graph has at least one path from u to v. Give an algorithm Longest-Path(G, u, v) that finds the *maximum-weight path* from u to v.

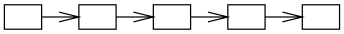
Problem 3 (4 points)
Suppose that G is a weighted directed graph, and you are interested in the *minimum-weight cycle* through a given vertex. Design an efficient algorithm Shortest-Cycle(G, v) that returns the weight of the minimum-weight cycle through v; your algorithm does *not* need to find the shortest cycle itself.

Problem 4 (bonus)
This problem is optional, and it allows you to get 2 bonus points toward your final grade.

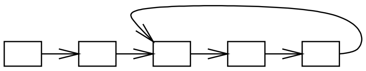
If we need to print all elements of a linked list (see Figure 1), we may write a program that starts at the beginning of the list and follows pointers until reaching its end; however, novice programmers sometimes mistakingly create a list whose last element points into the middle of the list (see Figure 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. Thus, you *cannot* mark the elements of the list that you have visited, because storing such marks would require a lot of memory.



1. Standard linked list.



2. Looped linked list.