Homework Assignment #9

Due Time/Date

2:20PM Tuesday, December 5, 2023. Late submission will be penalized by 20% for each working day overdue.

How to Submit

Please write or type your answers on A4 (or similar size) paper. Drop your homework by the due time in Yih-Kuen Tsay's mail box on the first floor of Management College Building 2. You may discuss the problems with others, but copying answers is strictly forbidden.

Problems

There are five problems in this assignment, each accounting for 20 points. (Note: problems marked with "(X.XX)" are taken from [Manber 1989] with probable adaptation.)

1. (7.16 modified)

   (a) Run the strongly connected components algorithm (the original version) on the directed graph shown in Figure 1. When traversing the graph, the algorithm should follow the given DFS numbers (from 9 down to 1). Show the *High* values as computed by the algorithm in each step.

   (b) Add the edge $(6, 5)$ to the graph and discuss the changes this makes to the execution of the algorithm.

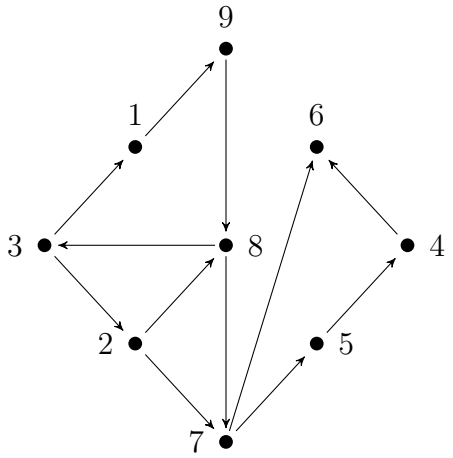

Figure 1: A directed graph with DFS numbers (from 9 down to 1)

2. (7.88) Let $G = (V, E)$ be a directed graph, and let T be a DFS tree of G. Prove that the intersection of the edges of T with the edges of any strongly connected component of G form a subtree of T.

3. We have discussed in class the idea of using DFS to find an augmenting path (if one exists) in a network with some given flow. Please present the algorithm in suitable pseudocode.

4. Consider designing an algorithm by dynamic programming to determine the length of a longest common subsequence of two strings (sequences of letters). For example, "abbcc" is a longest common subsequence of "abcabcabc" and "aaabbbccc", and so is "abccc".

   (a) Formulate the solution using recurrence relations.

   (b) Present the algorithm in suitable pseudocode, based on the previous recursive formulation. What is the time complexity of your algorithm?

5. The cost of finding a key value in a binary search tree is linearly proportional to the depth/level of the node where the key value is stored, with the root considered to be at level 0. Obviously, for a key value that is known to be looked up more frequently, it is better stored in a node at a smaller level.

   Consider designing by dynamic programming an algorithm that, given the look-up frequencies of n key values, constructs an optimal binary search tree that will incur the least total cost for performing all the look-ups.

   (a) Formulate the solution using recurrence relations; let $F[1..n]$ be the look-up frequencies of the n key values $K[1..n]$, which are in sorted order.

   (b) Present the algorithm in suitable pseudocode, based on the previous recursive formulation. What is the time complexity of your algorithm?