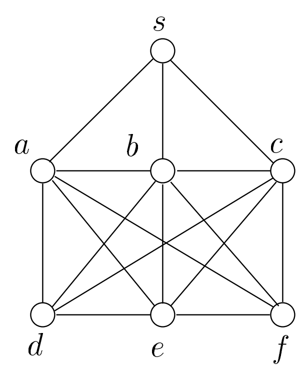
Computer Science B63 — March 22, 2007
Scarborough Campus — University of Toronto

Homework Assignment #5
Due: April 4, 2007, by 1:10 pm

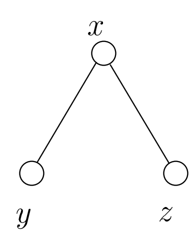
1. *Please complete and attach (with a staple) an assignment cover page to the front of your assignment. You may work alone or with one other student. If you work in a group, write both your names on the cover sheet and submit only one copy of your homework.*
2. *If you do not know the answer to a question, and you write "I (We) do not know the answer to this question", you will receive 20% of the marks of that question. If you just leave a question blank with no such statement, you get 0 marks for that question.*
3. *Unless we explicitly state otherwise, you should justify your answers. Your paper will be marked based on the correctness and completeness of your answers, and the clarity, precision and conciseness of your presentation.*

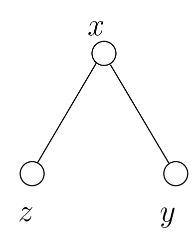
Question 1. (10 marks)

a. (5 marks) Consider the undirected graph shown below. How many different BFS trees can result from the execution of the Breadth-First Search algorithm starting at node s, over all possible orderings of the edges in the adjacency lists? Justify your answer.



Note: The order of a node's children in a BFS tree is *not* significant. For example, the two diagrams below represent *the same* tree.

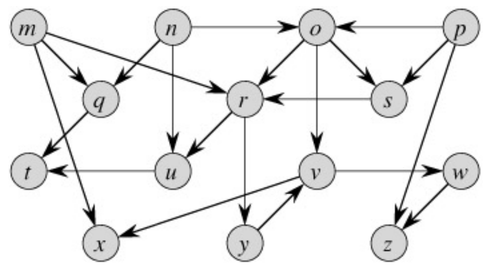




b. (5 marks) Prove or find a counterexample to the following claim:

> Let G be any *directed* graph (*without* self-loops) and u, v be any two nodes such that there is a directed path from u to v in G. For every Depth-First Search of G, if $d[u] < d[v]$ then v is a descendant of u in the corresponding depth-first search forest.

Question 2. (10 marks) Show the ordering of vertices produced by the TOPOLOGICAL-SORT procedure (on page 550 of the course textbook) when it is run on the directed acyclic graph shown below, under the assumption that the **for** loop of the DFS procedure considers the vertices in alphabetical order, and that each adjacency list is ordered alphabetically. There is no need to show any intermediate work.



Question 3. (30 marks)

Let G be an arbitrary undirected connected graph where each edge has a weight, and T be a minimum spanning tree of G.

a. (8 marks) Suppose we augment G by adding a new node v and new weighted edges connecting v to some nodes of G. Let G' be the resulting (weighted) graph. Prove or find a counterexample to the following statement: We can always obtain a minimum spanning tree T' of G' by adding to T an edge of minimum weight among all the new edges (those connecting v and the nodes of G).

b. (12 marks) Suppose we augment G by adding a new edge e of weight w between two nodes of G. Let G' be the resulting (weighted) graph. Describe an algorithm that builds a minimum spanning tree of G' in $O(n)$-time, where n is the number of nodes in G. Assume that each of G and its minimum spanning tree T, is given in the adjacency-list representation. Your algorithm's description should be clear and brief (no more than four English sentences). Explain why your algorithm's worst-case running time is $O(n)$.

c. (10 marks) Prove that your algorithm for part (b) works.

Question 4. (40 marks) This question is a programming assignment. To see its description follow the link given in the "Assignments" section of the course web page.