UNIVERSITY OF MINNESOTA
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
4041
ALGORITHMS AND DATA STRUCTURES
FALL 2017

ASSIGNMENT 2:
Assigned: 11/04/17 Due: 11/12/17 at 11:55pm (submit via moodle, in a zip if multiple files)

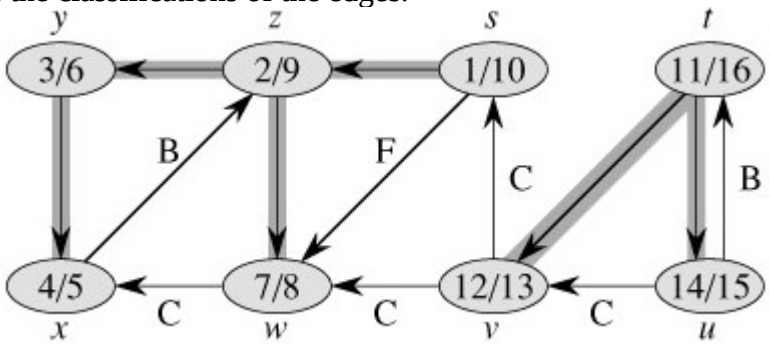
Problem 1. (15 points)
Show the full Rabin-Karp method of string matching (the one that converts letters to numbers) on the following pattern and search string. ("full" means you have to use an appropriately small base and modular arithmetic.) Find all true and false-positive string matches (along with which ones are actually true matches). Show work for each step.

Pattern = abacab
String = abacacabacabacababcabaca

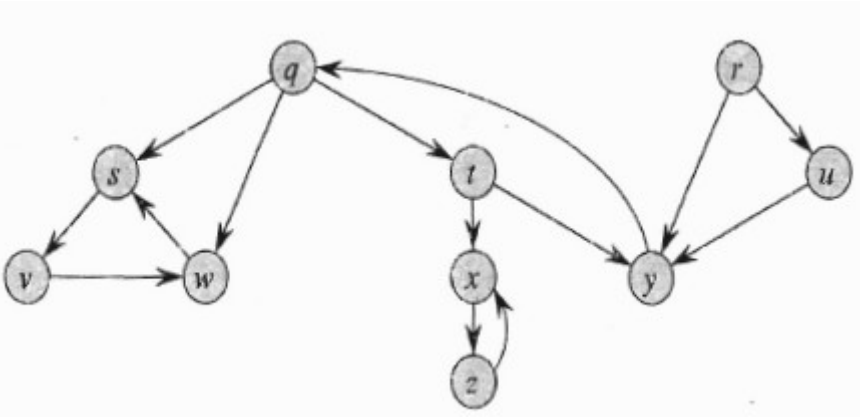
Problem 2. (20 points)
Show the finite state automata for the pattern below. Then use this finite automata to find matches in the String. Find all matches and show your work for each step. (Same pattern and string as Problem 1.)

Pattern = abacab
String = abacacabacabacababcabaca

Problem 3. (10 points)
Give an adjacency-list representation of the the following graph. Give an equivalent adjacency-matrix representation. Ignore the classifications of the edges.



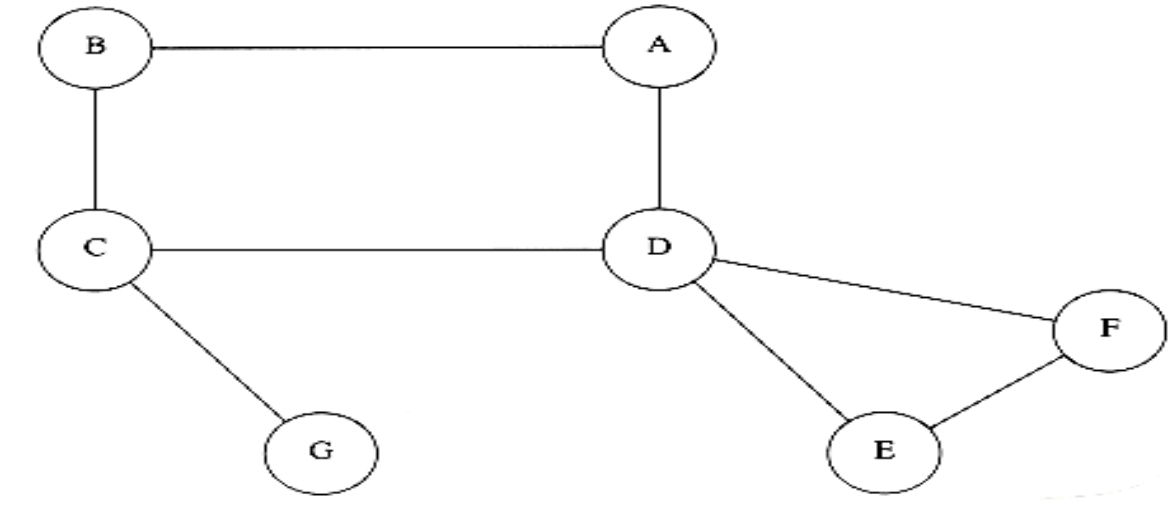
Problem 4. (15 points)
Show how depth-first search works on the following graph. Assume that the for loop of lines 5-7 of the DFS procedure considers the vertices in alphabetical order and assume that each adjacency list is ordered alphabetically. Show the discovery and finishing times for each vertex, and show the classification of each edge (tree, back, forward, cross).



Problem 5. (15 points)
Modify the pseudocode for depth-first search so that it prints out every edge in a directed graph G, as (source, destination) and it's type (tree, back, forward, cross).

Problem 6. (15 points)
Formally prove the following: if depth first search finds a "back" edge, there is a cycle in the graph.

Problem 7. (10 points)
Show the results of running breadth-first search on the following undirected graph. Use vertex C as the source. Show each step (the changes in the queue) and the final result (with red arrows/pi values).



Extra credit: Problem 5. (10 extra credit points)
When an adjacency-matrix representation is used, most graph algorithms require time $\Theta(|V|^2)$, but there are some exceptions.

Show that determining whether a directed graph contains a sink–a vertex with in-degree $|V| - 1$ and out-degree 0–can be determined in time $O(|V|)$, even if an adjacency-matrix representation is used.

Show the steps of the algorithm you devised on a graph with a sink.