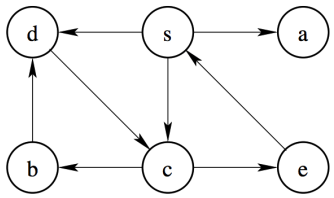
Homework 3

This homework is due by the beginning of class Tuesday October 17. It will be accepted for full credit if uploaded to blackboard by 5pm October 17. It will be accepted with a 20% deduction if uploaded by 9am Wednesday Oct. 18, after which it will not be accepted.

1. Cfonsider the following directed graph:



   (a) Suppose that you traverse this graph using a BFS, starting at node s, and every time that you have a collection of edges that you can visit in any order, you choose to visit them in alphabetical order. List the order in which you visit the nodes (for example, a wrong answer in the correct format might look like: s, d, b, c, e, f, a). No justification is necessary.

   (b) Suppose that you traverse this graph using a DFS, starting at node s, and every time that you have a collection of edges that you can visit in any order, you choose to visit them in alphabetical order. List the order in which you visit the nodes. No justification is necessary.

   (c) Give an example undirected graph with 6 nodes (labelled s,a,b,c,d,e) that visit nodes in the same order with a DFS and BFS when starting from node s (using the same alphabetical order rule), and state what that order is.

2. An undirected graph is bipartite if it is possible to partition the vertices into two subsets, V_1 and V_2, such that all the edges go between V_1 and V_2. If the graph has n total vertices and e total edges. Describe an $O(n+e)$ algorithm to determine whether a given connected un-directed graph is bipartite. Explain your algorithm and derive its running time. (Hint: Use BFS.)

3. Consider a graph with n nodes. Usually, when graphs are represented as adjacency matrices, algorithms require time $\Omega(n^2)$ (this means, "at least n^2 time") because it takes that long to look at every location in the matrix. This problem explores an exception. Show that deciding whether a directed graph G contains a universal sink a vertex with in-degree $n1$ and out-degree 0 is possible in time $O(n)$ when the graph is given as an adjacency matrix. (Hint, the best solution I know of checks about $3n$ of the adjacency matrix values).