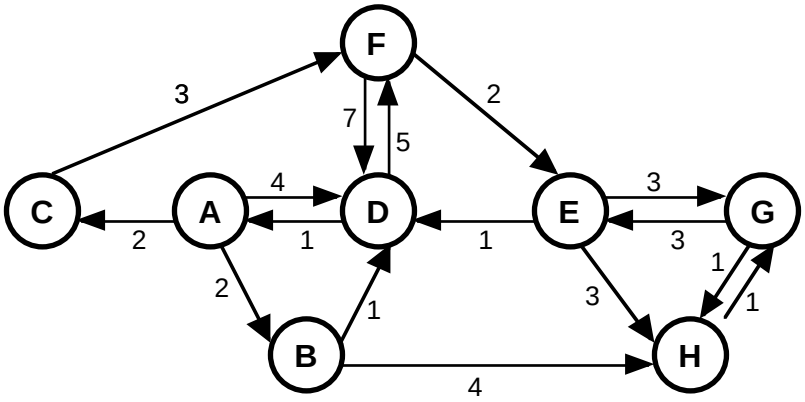
CSCI E-124 Homework 2 Out: Feb. 13, 2003
Due: Feb 28, 2003

For all homework problems where you are asked to give an algorithm, you must prove the correctness of your algorithm and establish the best upper bound that you can give for the running time. You should always write a clear informal description of your algorithm in English. You may also write pseudocode if you feel your informal explanation requires more precision and detail. As always, try to make your answers as clear and concise as possible.

1. List the order in which the nodes in the graph below are visited in a *depth-first search* starting from node *A*. In case of ties visit the alphabetically first node first. Also classify all graph edges as tree edges, back edges, forward edges, or cross edges. List the order of visits for a *breadth-first search* from *A*.



2. Find the shortest path, and its length, from *A* to *E* in the graph above using Dijkstra's algorithm. List all the steps in reasonable detail.

3. A *bipartite graph* is a graph whose vertices can be partitioned into two disjoint sets V_1 and V_2 such that there are no edges between vertices in V_1 nor any edges between vertices in V_2. (See Figure 1.) Give a linear time algorithm for determining whether an undirected graph is bipartite.

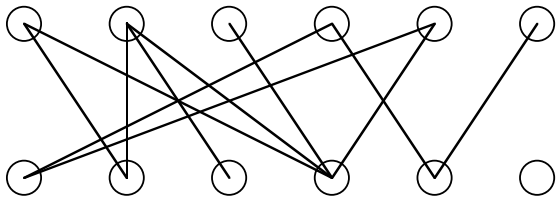


Figure 1: A bipartite graph.

4. Giles has asked Buffy to optimize her procedure for nighttime patrol of Sunnydale. (This takes place sometime in Season 2.) Giles points out that proper slaying technique would allow Buffy to traverse all of the streets of Sunnydale in such a way that she would walk on each side of each street, exactly once. Buffy now has slayer homework: how can it be done? (If you have to assume anything about the layout of the city of Sunnydale, make it clear!)

5. The *bottleneck* of a path is defined to be the length of the longest edge in the path. Design an efficient algorithm to solve the single source smallest bottleneck problem; i.e. find the paths from a source to every other node such that each path has the smallest possible bottleneck.

6. The *risk-free currency exchange problem* offers a risk-free way to make money. Suppose we have currencies $c_1, \ldots, c_n$. (For example, c_1 might be dollars, c_2 rubles, c_3 yen, etc.) For every two currencies c_i and c_j there is an exchange rate $r_{i,j}$ such that you can exchange one unit of c_i for $r_{i,j}$ units of c_j. Note that if $r_{i,j} \cdot r_{j,i} > 1$, then you can make money simply by trading units of currency i into units of currency j and back again. This almost never happens, but occasionally (because the updates for exchange rates do not happen quickly enough) for very short periods of time exchange traders can find a sequence of trades that can make risk-free money. That is, if there is a sequence of currencies $c_{i_1}, c_{i_2}, \ldots, c_{i_k}$ such that $r_{i_1,i_2} \cdot r_{i_2,i_3} \ldots \cdot r_{i_{k-1},i_k} \cdot r_{i_k,i_1} > 1$, then trading one unit of c_{i_1} into c_{i_2} and trading that into c_{i_3} and so on will yield a profit.

   Design an efficient algorithm to detect and output a risk-free currency exchange if any exist.

7. Give an efficient algorithm that takes as input a DAG G and two vertices s and t and outputs the number of paths from s to t in G.