Homework 6
Due at 11:59 p.m. on Wed. Nov. 26

By handing in the homework you are agreeing to the Homework Rules; see EdStem.

NOTE: In one or more of the following problems, you are asked to describe an algorithm to solve a problem. In your description, you can make use of algorithms we studied in class, without providing details about it. For example, you can just say "Use mergesort to sort the array."

1. Each of the following problems can be solved by (1) forming a graph, (2) running either BFS, DFS, topological sort, or the SCC algorithm, and (3) possibly doing some additional processing. Indicate how to do (1), (2), and (3) for each of the following problems. For (1), just describe how the vertices and edges of the graph correspond to the information in the problem, and state whether the graph is directed or undirected. For (2), if more than one algorithm would work, choose just one.

The first problem is given to you as an example.

(a) You are given a list of cities and a table T. The columns of the table are indexed by the cities, and the rows of the table are also indexed by the cities. Entry $T[i, j] = 1$ if there is a highway from City i to City j, and $T[i, j] = 0$ otherwise. All highways are two-way.

Given two cities A and B, determine whether it is possible to drive from city A to city B, taking some or all of the highways indicated in the table.

Answer: (1) Form an undirected *graph G where the cities are the nodes, and there is an edge from City u to City v iff there is a highway between these two cities. (2) Run BFS on G, using A as the source. (3) Check whether the shortest path distance from A to B, computed by the BFS, is infinity. If so, output "no" else output "yes".*

(b) An airline offers a benefit to its employees. It allows employees to fly for free on certain flights. It maintains a list of these free flights, showing the origin and destination for each flight on the list. Given the list of all cities served by this airline, and the free flight list, determine whether employees can travel whereever they want to go within the airline's flight network for free. That is, determine whether an employee wanting to fly from one city served by the airline to another will always be able to do so by taking one or more free flights, no matter which two cities are involved.

(c) You are given a list L_1 of n movies, $m_1, \ldots, m_n$. A group of n people were given access to these movies via a streaming service. Each person in the group watched some or all of the movies, and then filled out a survey. You are given a list L_2 of all ordered pairs of movies (m_i, m_j) such that at least one person in the group said that he or she preferred movie m_i to movie m_j. A "ranked listing" of the movies is just a way to number (rank) the movies from 1 to n, and the number assigned to movie m_i is called the "rank" of movie m_i.

A ranked listing is consistent with everyone's feedback if, for every pair of movies (m_i, m_j) in L_2, the number assigned to movie m_i is less than the number assigned to movie m_j.

Determine whether there is a ranked listing of the movies that is consistent with everyone's feedback, and if so, output such a listing.

(d) Each person in a company has an email account on an internal company network. Each person has contact list, consisting of selected people also on this email network. A person can send emails only to people on their contact list. Given the names of two employees in the company, who we'll call A and B, determine the minimum number of times an email from A would need to be forwarded in order to reach B.

2. Consider the directed graph $G = (V, E)$ where $V = \{v_1, v_2, v_3, v_4, v_5\}$ and $E = \{(v_1, v_2), (v_2, v_1), (v_1, v_3), (v_3, v_4)\}$

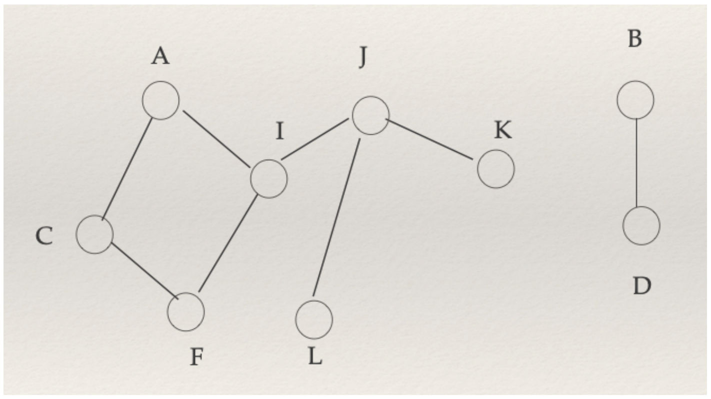
(a) Draw a picture of the graph.

(b) Draw the adjacency list representation of the graph.

(c) Draw the adjacency matrix representation of the graph.

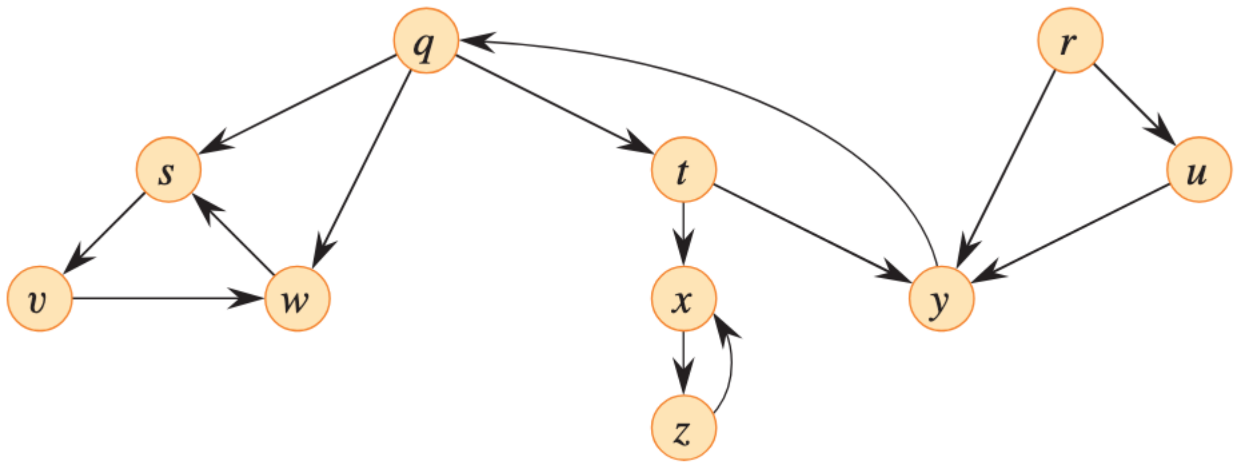
3. The pseudocode for BFS is shown in the textbook on p. 556 (and was also presented in Lecture 19). Show the execution of BFS on the following graph, with source vertex C, in the following way. (Note that the source is NOT vertex A.)

For the start of each iteration of the while loop of lines 10-18, give the values of v.d (the distance of v from the source) and the contents of the queue Q. Include the start of the last "iteration," where Q is empty. You can either do this using pictures, as on p. 557 but with less detail, or you can just give Q and the values v.d for each iteration without pictures.



4. Design an algorithm that takes as input a DAG G whose vertices are $v_1, \ldots, v_n$. For each vertex v_i in G, the algorithm computes the minimum index j such that there is a directed path from v_i to v_j (that is, the minimum index j of any vertex v_j that is reachable from v_i). Your algorithm should run in time $O(V + E)$. You do not need to give pseudocode as long as your description of the algorithm is clear. If you do give pseudocode, make sure to include comments.

5. Run the Strongly Connected Components algorithm on the following graph. In the first run of DFS, process vertices and adjacency lists in alphabetical order.



For your answer to this problem, do the following on the drawing of the graph:

(a) Label the vertices with their discover and finish times in the first DFS.

(b) Circle the strongly connected components produced after the running of DFS on the transpose of the input graph. Number these components in the order they are discovered.

6. For each of the following statements, indicate whether it is True or False. If false, show a small graph G (with no more than 5 nodes) which demonstrates that it is false. If it is true, briefly explain why.

Note: In class, for DFS on a directed graph, we defined the finish time of a strongly connected component to be the maximum finish time of any vertex in that strongly connected component. Similarly, define the discovery time of a strongly connected component to be the *minimum* start time of any vertex in that component.

Consider running DFS on a directed graph G.

(a) Let u and v be vertices in two different connected components of G, which we'll call C_u and C_v respectively. If the discovery time for C_u is earlier than the discovery time for C_v, and (u, v) is an edge in G, then $u.d < v.d$.

(b) Let u and v be vertices in two different connected components of G, which we'll call C_u and C_v respectively. If the discovery time for C_v is earlier than the discovery time for C_u, and (u, v) is an edge in G, then $v.f < u.f$.

(c) Let (u, v) be an edge in G. If v is gray when edge (u, v) is explored (when v is found in u's neighbor list), then the DFS causes (u, v) to become a back edge.