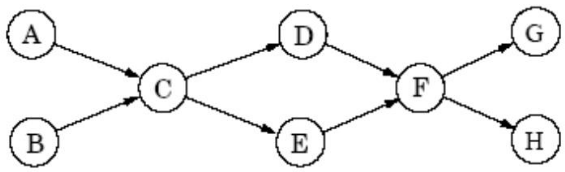
Graph Practice Problems

Q1

Run the DFS-based topological ordering algorithm on the following graph. Whenever you have a choice of vertices to explore, always pick the one that is alphabetically first.



Q2

For each node u in an undirected graph, let $\texttt{twodegree}[u]$ be the sum of the degrees of u's neighbors. Show how to compute the entire array of $\texttt{twodegree}[\cdot]$ values in linear time, given a graph in adjacency list format.

Q3

Design a linear-time algorithm which, given an undirected graph G and a particular edge e in it, determines whether G has a cycle containing e.

Q4

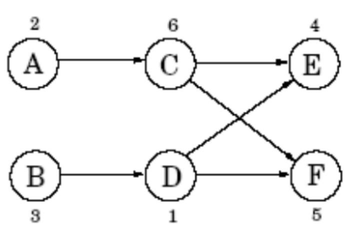
Give an efficient algorithm which takes as input a directed graph $G = (V, E)$, and determines whether or not there is a vertex $s \in V$ from which all other vertices are reachable.

Q5

You are given a directed graph in which each node $u \in V$ has an associated *price* p_u which is a positive integer. Define the array $\texttt{cost}$ as follows: for each $u \in V$,

$$\texttt{cost}[u] = \text{price of the cheapest node reachable from } u \text{ (including } u \text{ itself).}$$

For instance, in the graph below (with prices shown for each vertex), the $\texttt{cost}$ values of the nodes A, B, C, D, E, F are $2, 1, 4, 1, 4, 5$, respectively.



Your goal is to design an algorithm that fills in the *entire* $\texttt{cost}$ array (i.e., for all vertices).

Q6

Often there are multiple shortest paths between two nodes of a graph. Give a linear-time algorithm for the following task.

> *Input:* Undirected graph $G = (V, E)$ with unit edge lengths; nodes $u, v \in V$.
> *Output:* The number of distinct shortest paths from u to v.

Q7

A *bipartite graph* is a graph $G = (V, E)$ whose vertices can be partitioned into two sets ($V = V_1 \cup V_2$ and $V_1 \cap V_2 = \emptyset$) such that there are no edges between vertices in the same set (for instance, if $u, v \in V_1$, then there is no edge between u and v).

Give a linear-time algorithm to determine whether an undirected graph is bipartite.