Graphs: (No deadline, Just Practice)

Notes: ***1 - All graphs are simple and represented as Adjacency List if not specified:***

2 - Linear time algorithm means O(V+E)

Q -1

The *reverse* of a directed graph $G = (V, E)$ is another directed graph $G^R = (V, E^R)$ on the same vertex set, but with all edges reversed; that is, $E^R = \{(v, u) : (u, v) \in E\}$.

Give a linear-time algorithm for computing the reverse of a graph in adjacency list format.

Q-2

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).

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

Q-3

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.

Q-4

In an undirected graph with 2 connected components it is always possible to make the graph connected by adding only one edge. Give an example of a directed graph with two strongly connected components such that no addition of one edge can make the graph strongly connected.

Q-5

Give a linear-time algorithm to find an odd-length cycle in a *directed* graph.

Q-6

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.

Q-7

Give an efficient algorithm that takes as input a directed acyclic graph $G = (V, E)$, and two vertices $s, t \in V$, and outputs the number of different directed paths from s to t in G.

Q-8

Give a linear-time algorithm for the following task.

Input: A directed acyclic graph G

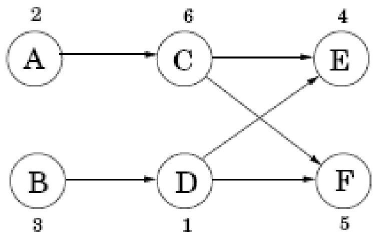
Question: Does G contain a directed path that touches every vertex exactly once?

Q-9

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 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 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* cost array (i.e., for all vertices).

(a) Give a linear-time algorithm that works for directed *acyclic* graphs. (*Hint:* Handle the vertices in a particular *order*.)

Q-10

Two paths in a graph are called *edge-disjoint* if they have no edges in common. Show that in any undirected graph, it is possible to pair up the vertices of odd degree and find paths between each such pair so that all these paths are edge-disjoint.

Q-11

Professor F. Lake suggests the following algorithm for finding the shortest path from node s to node t in a directed graph with some negative edges: add a large constant to each edge weight so that all the weights become positive, then run Dijkstra's algorithm starting at node s, and return the shortest path found to node t.

Is this a valid method? Either prove that it works correctly, or give a counterexample.

Q-12

Consider a directed graph in which the only negative edges are those that leave s; all other edges are positive. Can Dijkstra's algorithm, started at s, fail on such a graph? Prove your answer.

Q-13

You are given a directed graph with (possibly negative) weighted edges, in which the shortest path between any two vertices is guaranteed to have at most k edges. Give an algorithm that finds the shortest path between two vertices u and v in $O(k|E|)$ time.

Q-14

You are given a strongly connected directed graph $G = (V, E)$ with positive edge weights along with a particular node $v_0 \in V$. Give an efficient algorithm for finding shortest paths between *all pairs of nodes*, with the one restriction that these paths must all pass through v_0.

Q-15

Shortest paths are not always unique: sometimes there are two or more different paths with the minimum possible length. Show how to solve the following problem in $O((|V| + |E|) \log |V|)$ time.

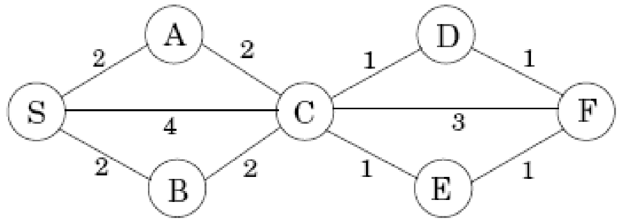
Input: An undirected graph $G = (V, E)$; edge lengths $l_e > 0$; starting vertex $s \in V$.
Output: A Boolean array $\texttt{usp}[\cdot]$: for each node u, the entry $\texttt{usp}[u]$ should be `true` if and only if there is a *unique* shortest path from s to u. (Note: $\texttt{usp}[s] = \texttt{true}$.)

Q-16

In cases where there are several different shortest paths between two nodes (and edges have varying lengths), the most convenient of these paths is often *the one with fewest edges*. For instance, if nodes represent cities and edge lengths represent costs of flying between cities, there might be many ways to get from city s to city t which all have the same cost. The most convenient of these alternatives is the one which involves the fewest stopovers. Accordingly, for a specific starting node s, define

$$\texttt{best}[u] = \text{minimum number of edges in a shortest path from } s \text{ to } u.$$

In the example below, the `best` values for nodes S, A, B, C, D, E, F are $0, 1, 1, 1, 2, 2, 3$, respectively.



Give an efficient algorithm for the following problem.

Input: Graph $G = (V, E)$; positive edge lengths l_e; starting node $s \in V$.
Output: The values of $\texttt{best}[u]$ should be set for *all* nodes $u \in V$.

Q-17

Consider an undirected graph $G = (V, E)$ with nonnegative edge weights $w_e \geq 0$. Suppose that you have computed a minimum spanning tree of G, and that you have also computed shortest paths to all nodes from a particular node $s \in V$.

Now suppose each edge weight is increased by 1: the new weights are $w'_e = w_e + 1$.

(a) Does the minimum spanning tree change? Give an example where it changes or prove it cannot change.

(b) Do the shortest paths change? Give an example where they change or prove they cannot change.

Q-18

The following statements may or may not be correct. In each case, either prove it (if it is correct) or give a counterexample (if it isn't correct). Always assume that the graph $G = (V, E)$ is undirected. Do not assume that edge weights are distinct unless this is specifically stated.

(a) If graph G has more than $|V| - 1$ edges, and there is a unique heaviest edge, then this edge cannot be part of a minimum spanning tree.

(b) If G has a cycle with a unique heaviest edge e, then e cannot be part of any MST.

(c) Let e be any edge of minimum weight in G. Then e must be part of some MST.

(d) If the lightest edge in a graph is unique, then it must be part of every MST.

(e) If e is part of some MST of G, then it must be a lightest edge across some cut of G.

(f) If G has a cycle with a unique lightest edge e, then e must be part of every MST.

(g) The shortest-path tree computed by Dijkstra's algorithm is necessarily an MST.

(h) The shortest path between two nodes is necessarily part of some MST.

(i) Prim's algorithm works correctly when there are negative edges.

(j) (For any $r > 0$, define an r-*path* to be a path whose edges all have weight $< r$.) If G contains an r-path from node s to t, then every MST of G must also contain an r-path from node s to node t.