1. T F: Consider a directed graph G = (V, E) and a vertex s ∈ V. Suppose that for all v ∈ V, there exists a directed path in G from s to v. Suppose that a DFS is run on G, starting from s. Then, true or false: the number of edges of G that will be labeled tree edges while running the DFS is exactly |V |−1.

2. True or false: even though BFS and DFS have the same space complexity, they do not always have the same worst case asymptotic time complexity.

3. A DFS forest is defined as the collection of all the trees generated when DFS is run on a graph. Then, true or false: Any DFS forest of an undirected graph contains the same number of trees.

4. Let G = (V, E) be a directed graph with arbitrary (possibly negative) edge weights. Suppose s, t ∈ V are two distinct vertices in G such that all directed paths from s to t in G contain no cycles, and at least one such path exists. Then, true or false: the Bellman-Ford algorithm, starting from a source vertex s, will correctly calculate the weight of a shortest path from s to t, even if G contains negative cycles.

5. Let G = (V, E) be a directed graph with arbitrary (possibly negative) edge weights, but no negative cycles. Let k be a positive integer. Suppose that, for all pairs of vertices u, v ∈ V, the graph G contains some directed path from u to v that uses ≤ k edges. Then, true or false: the Bellman-Ford algorithm correctly outputs shortest paths when modified to only perform k rounds of relaxations, instead of the usual
|V |−1 rounds.

6. Consider a weighted, directed graph G = (V,E) with a source vertex s. Suppose that:
   - No edges enter the source s.
   - All edges leaving the source s have negative weights.
   - All other edges in G have non-negative weights.

Then, true or false: Dijkstra's algorithm, will correctly find a shortest path from s to any other vertex t in G.

7. Which of the following is an advantage of adjacency matrices over adjacency lists when representing an undirected graph G?
   (a) It is often faster to add and remove edges from G when using an adjacency matrix.
   (b) With adjacency matrices, iterating over all neighbors incident to a vertex v requires only O (δ (v)) time, where δ (v) is the degree of v.

(c) Any graph G can be represented using fewer bits of space with an adjacency matrix than with adjacency lists.

8. Let G = (V,E) be a weighted, directed graph with arbitrary (possibly negative) edge weights, but no negative cycles. Suppose we wish to find the shortest path P that starts at a source vertex s, ends at a target vertex t, and also passes through two additional detour vertices u and v. P is permitted to contain cycles, which might be necessary when visiting u and v. Provide an algorithm that does this in O(V E) time.

9. T F The depth of a breadth-first search tree on an undirected graph G = (V, E) from an arbitrary vertex v ϵ V is the diameter of the graph G. (The diameter d of a graph is the smallest d such that every pair of vertices s and t have $\delta(s, t) \leq d$.)

10. T F Given a graph G = (V, E) with positive edge weights, the Bellman-Ford algorithm and Dijkstra’s algorithm can produce different shortest-path trees despite always producing the same shortest-path weights.

11. T F Dijkstra’s algorithm may not terminate if the graph contains negative-weight edges.

12. Depth-first search can be modified to check if there are cycles in an undirected graph.

13. Breadth-first search can be modified to check if there are cycles in an undirected graph.

14. If we represent a graph with |V| vertices and Θ (|V |) edges as an adjacency matrix, the worst-case running time of breadth-first search is Θ ($|V|^2$).

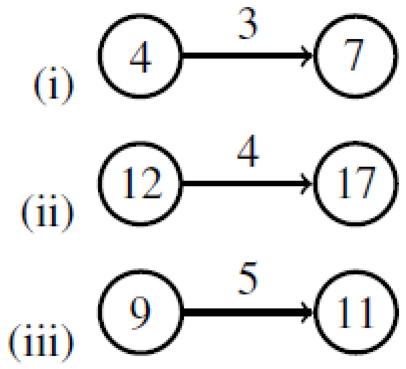
15. We can use Dijkstra’s algorithm to find the shortest path between two vertices in a graph with arbitrary edge weights.

16. T F If a depth-first search on a directed graph $G = (V, E)$ produces exactly one back edge, then it is possible to choose an edge $e \in E$ such that the graph $G_0 = (V, E - \{e\})$ is acyclic.

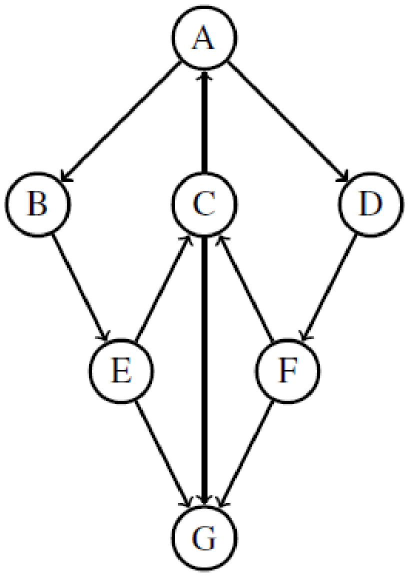
17. T F If a directed graph G is cyclic but can be made acyclic by removing one edge, then a depth-first search in G will encounter exactly one back edge.

18. What is the running time of depth-first search, as a function of $|V|$ and $|E|$, if the input graph is represented by an adjacency matrix instead of an adjacency list?

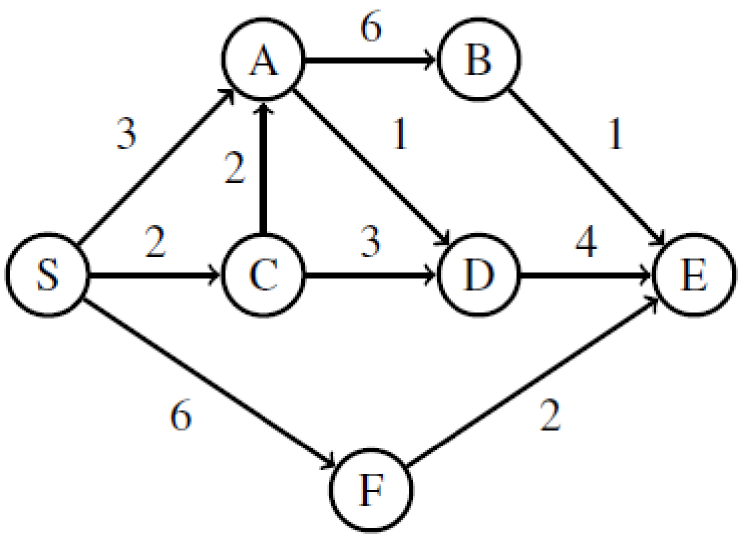
19. What is the result of relaxing the following edges?



20. Perform a depth-first search on the following graph starting at A. Label every edge in the graph with T if it's a tree edge, B if it's a back edge, F if it's a forward edge, and C if it's a cross edge. Assume that whenever faced with a decision of which node to pick from a set of nodes, pick the node whose label occurs earliest in the alphabet.



21. Run Dijkstra’s algorithm on the following directed graph, starting at vertex S. What is the order in which vertices get removed from the priority queue? What is the resulting shortest-path tree?



22. Suppose that you want to get from vertex s to vertex t in an un-weighted graph $G = (V,E)$, but you would like to stop by vertex u if it is possible to do so without increasing the length of your path by more than a factor of α.
 Describe an efficient algorithm that would determine an optimal s-t path given your preference for stopping at u along the way if doing so is not prohibitively costly. (It should either return the shortest path from s to t or the shortest path from s to t containing u, depending on the situation.)

23. In a BST, we can find the next smallest element to a given element in $O(1)$ time.

24. Give an $O(V + E)$-time algorithm to remove all the cycles in a directed graph $G = (V, E)$. Removing a cycle means removing an edge of the cycle. If there are k cycles in G, the algorithm should only remove $O(k)$ edges.

25. Let $G = (V;E)$ be a weighted, directed graph with exactly one negative weight edge and no negative-weight cycles. Give an algorithm to find the shortest distance from s to all other vertices in V that has the same running time as Dijkstra.