Study Questions **Graph Algorithms**

In addition to the homeworks already assigned, the following are good study questions for the midterm exam.

1. Draw a simple undirected graph G that has 12 vertices, 18 edges, and 3 connected components. Why would it be impossible to draw G with 3 connected components if G had 66 edges?

2. Let G be a simple connected graph with n vertices and m edges. Explain why $O(\log m)$ is $O(\log n)$.

3. Draw a simple connected directed graph with 8 vertices and 16 edges, such that the in-degree and out-degree of each vertex is 2. Show that there is a single cycle (which may not necessarily be simple) that includes all the edges of your graph, that is, you can trace all the edges in their respective directions without ever lifting your pencil. (Recall that such a cycle is called an *Euler tour*.)

4. Bob loves foreign languages and wants to plan his course schedule to take the following nine language courses: LA15, LA16, LA22, LA31, LA32, LA126, LA127, LA141, and LA169. The course prerequisites are:
   - LA15: (none)
   - LA16: LA15
   - LA22: (none)
   - LA31: LA15
   - LA32: LA16, LA31
   - LA126: LA22, LA32
   - LA127: LA16
   - LA141: LA22, LA16
   - LA169: LA32.

   Find a sequence of courses that allows Bob to satisfy all the prerequisites.

5. Suppose we represent a graph G having n vertices with an adjacency matrix. Why, in this case, would inserting an undirected edge in G run in $O(1)$ time while inserting a new vertex would take $O(n^2)$ time?

6. Describe the details of an $O(n + m)$-time algorithm for computing all the connected components of an undirected graph G with n vertices and m edges.

7. Let T be the spanning tree rooted at the start vertex produced by the depth-first search of a connected, undirected graph, G. Argue why every edge of G, not in T, goes from a vertex in T to one of its ancestors, that is, it is a *back edge*.

8. Suppose G is a graph with n vertices and m edges. Describe a way to represent G using $O(n + m)$ space so as to support in $O(\log n)$ time an operation that can test, for any two vertices v and w, whether v and w are adjacent.

9. Draw a simple, connected, weighted, undirected graph with 8 vertices and 16 edges, and with distinct edge weights. Identify one vertex as a "start" vertex and illustrate a running of Dijkstra's algorithm on this graph.

10. Show how to modify Dijkstra's algorithm for the case when the graph is directed and we want to compute shortest *directed paths* from the source vertex to all the other vertices.

11. Draw a (simple) directed weighted graph G with 10 vertices and 18 edges, such that G contains a minimum-weight cycle with at least 4 edges. Show that the Bellman-Ford algorithm will find this cycle.

12. The dynamic programming algorithm of Floyd-Warshall for computing all-pairs shortest paths computes only shortest- path distances, not actual paths. Describe a version of this algorithm that outputs the set of all shortest paths between each pair of vertices in a directed graph. Your algorithm should still run in $O(n^3)$ time.

13. Give an example of a weighted directed graph, G, with negative-weight edges, but no negative-weight cycle, such that Dijkstra's algorithm incorrectly computes the shortest-path distances from some start vertex v.

14. Suppose that every shortest path from some vertex, v, in an n-vertex weighted graph, G, to any other vertex in G has at most $k < n - 1$ edges. Show that it is sufficient to run the Bellman-Ford algorithm for only k iterations, instead of $n - 1$, to solve the single-source shortest-paths problem for v in G.

15. Draw a simple, connected, undirected, weighted graph with 8 vertices and 16 edges, each with unique edge weights. Illustrate the execution of Kruskal's algorithm on this graph. (Note that there is only one minimum spanning tree for this graph.)

16. Repeat the previous problem for the Prim-Jarnik algorithm.

17. Repeat the previous problem for Baruvka's algorithm.

18. Give an example of weighted, connected, undirected graph, G, such that the minimum spanning tree for G is different from every shortest path tree rooted at a vertex of G.

19. Suppose G is a weighted, connected, undirected, simple graph and e is a largest-weight edge in G. Prove or disprove the claim that there is no minimum spanning tree of G that contains e.

20. Suppose G is an undirected, connected, weighted graph such that the edges in G have distinct edge weights. Show that the minimum spanning tree for G is unique.

21. Suppose Joseph Kruskal had an evil twin, named Peter, who designed an algorithm that takes the exact opposite approach from his brother's algorithm for finding an MST in an undirected, weighted, connected graph, G. Also, for the sake of simplicity, suppose the edges of G have distinct weights. Peter's algorithm is as follows: consider the edges of G by decreasing weights. For each edge, e, in this order, if the removal of e from G would not disconnect G, then remove e from G. Peter claims that the graph that remains at the end of his algorithm is a minimum spanning tree. Prove or disprove Peter's claim.

22. What is the definition of degree centrality, closeness centrality, and betweenness centrality?

23. Describe efficient algorithms for computing each of the centralities of the previous problem. What are the running times for each of these algorithms?