19. Identify an algorithm for determining if an undirected graph is connected.

Either a Breadth-First or Depth-First Traversal will do the job. As the algorithm runs, keep track of the visited nodes. If some are missed when the algorithm exits then it is disconnected, otherwise connected.

20. Identify an algorithm for determining if a digraph is strongly connected.

Warshall's Algorithm produces the "Transitive Closure" matrix for the graph. If the only zeros in the Transitive Closure matrix are on the main diagonal then a digraph is strongly connected. Otherwise it is not.

21. Given a digraph's transitive closure matrix, how can we determine if there is a path from vertex vi to vj, and a path from vertex vj to vi?

Both of the entries (vi, vj) and (vj, vi) are non-zero.

22. State the difference between a spanning tree and a minimum spanning tree.

A minimum spanning tree is only defined for weighted graphs. It is the spanning tree which has the smallest total edge-weight.

23. It is possible to have more than one minimum spanning tree for a graph, true or false?

I could easily draw a weighted graph that has two minimum spanning trees. This is True. (Not sure if it's possible if no two edges have the same weight. My guess is that it would be True as well. This is not a trivial issue.)

24. What does Dijkstra's Algorithm do that Floyd's Algorithm cannot do?

Floyd's algorithm is a lot slower than Dijkstra's. If we had a million nodes the difference could be excruciating. They can both be easily modified to also return the shortest path as well as the shortest distance.

25. What does Floyd's Algorithm do that Dijkstra's Algorithm cannot do?

Floyd's algorithm will work on directed graphs. Dijkstra's will not. The key is lines 14,15 of Dijkstra's algorithm where the output matrix is written to. The value is put in symmetrically. In line 5 of Floyd's algorithm, there is no such symmetry. Symmetry is a characteristic of undirected graphs. This is not inherent in the algorithm though.

Floyd's algorithm does it for all possible starting points. To do the same thing by running Dijkstra's algorithm n times is actually slower.

26. Dijkstra's Algorithm is to operate on a graph containing n vertices. Using Big-O notation, give the speed of the algorithm.

O(n2) + O(6n) -> O(n2)

This does agree with information I found online. I think it may actually be a lot worse than that. The findMinPath method that is called on line 9 has to loop over the nodes that are out and find the shortest distance to any of them from an already included node. This by itself could be O(n2) which makes the overall algorithm have O(n3)

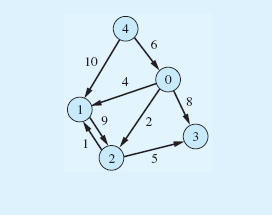
27. Floyd's Algorithm is to operate on a graph containing n vertices. Using Big-O notation, give the speed of the algorithm.

O(4n3)

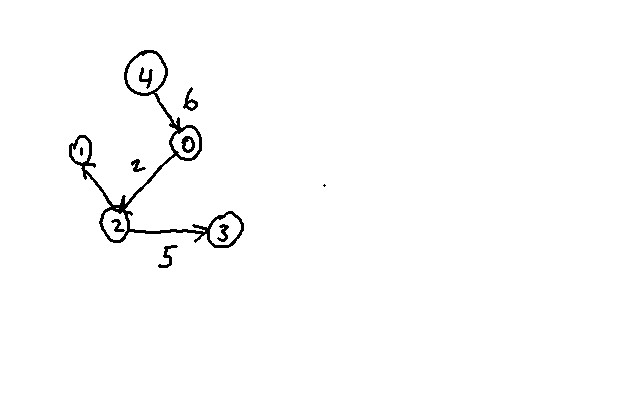
28. Draw the graph obtained when Dijkstra's Algorithm operates on the following graph, assuming vertex 4 is the starting vertex.

See above. The version of Dijkstra's algorithm in the book does not work on directed graphs.

Removing line 15 and using care with line 9 should work for directed graphs.



**Note:** This will be a graph similar to the one above with all of the vertices, but only the edges used by Dijksta’s Algorithm.



29. Give the contents of the returned array when Floyd's Algorithm operates on the graph shown in Exercise 28. **Note:** This will be an All-Points Shortest Path Matrix similar to that depicted in the text Figure 9.40. Use an asterisk for “impossibly high” path lengths.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 |
| 0 | \* | 3 | 2 | 7 | \* |
| 1 | \* | \* | 9 | 14 | \* |
| 2 | \* | \* | \* | 5 | \* |
| 3 | \* | \* | \* | \* | \* |
| 4 | 6 | 10 | 8 | 13 | \* |

30. True or false, Floyd's Algorithm can only operate on directed graphs?

False.

31. Of the algorithms we studied, which would be used to determine the two-way roads to close a connecting group of towns, and still allow access to all towns?

The Minimum Spanning Tree algorithm would work for this.

32. Of the algorithms we studied, which would be used to determine the toll roads to travel to minimize the tolls when traveling from a given town to all other towns?

Dijkstra's algorithm could be used unaltered for this.

33. Of the algorithms we studied, which would be used to determine if there is a way to pass through all towns connected by one-way streets.

I don't really think any of the algorithms in this chapter of the book are well suited to this problem. The easiest one to work with would be Floyd's because it allows for arbitrary starting points but it would have to be modified so that we are always looking for the next node from the current node rather than any from any node that is in play (it does no good to end up with a "asterisk" shape). (Starting with the graph in problem 28 Floyd's algorithm, as written, won't find the desired path.)

It would not be hard to write a recursive algorithm which would do the job. Start with an overall loop of all possible starting points and then from each node try to draw edges onward until either a path is found or it isn't. This could be done easily by *backtracking*. Such a path may not even exist, even in a connected graph, and this method would tell us that result.

34. Of the algorithms we studied, which would be used to determine the cheapest fares between all the cities that an airline flies to?

Floyd's algorithm is well suited to this task.