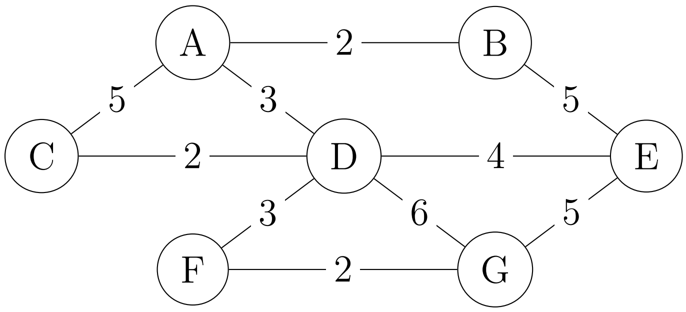
Shortest Path and Minimum Spanning Tree

Question Bank

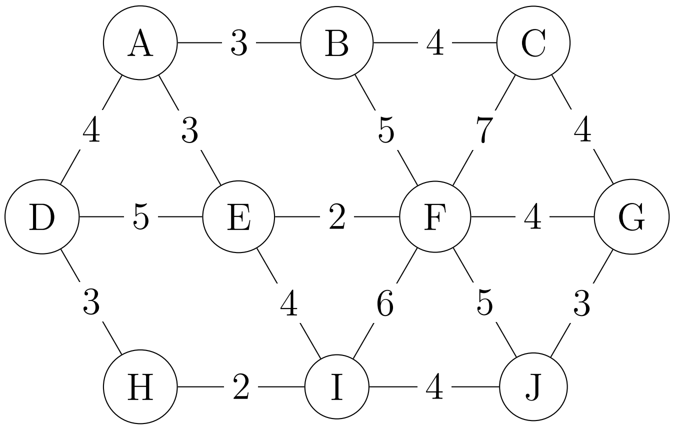
Note: Remember that some graphs have many minimum spanning trees. So if your tree looks different to a solution you can check that the total of all edge weights on your tree is as small as the example tree.

1. For the following graph:



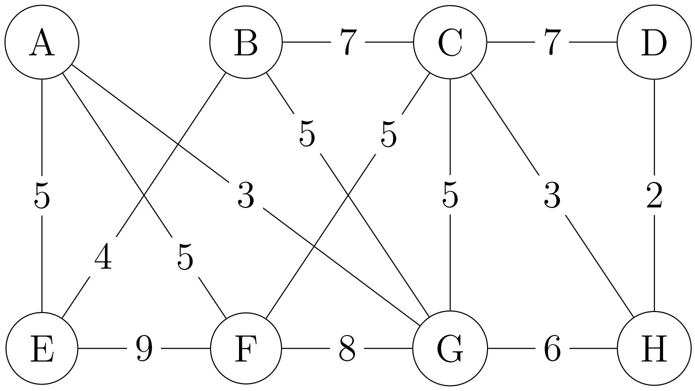
   a) Find the shortest path from C to G.
   b) Find the shortest path from F to B.
   c) Draw a minimum spanning tree.

2. For the following graph:



   a) Find the shortest path from C to D.
   b) Find the shortest path from H to B.
   c) Draw a minimum spanning tree.

3. For the following graph:

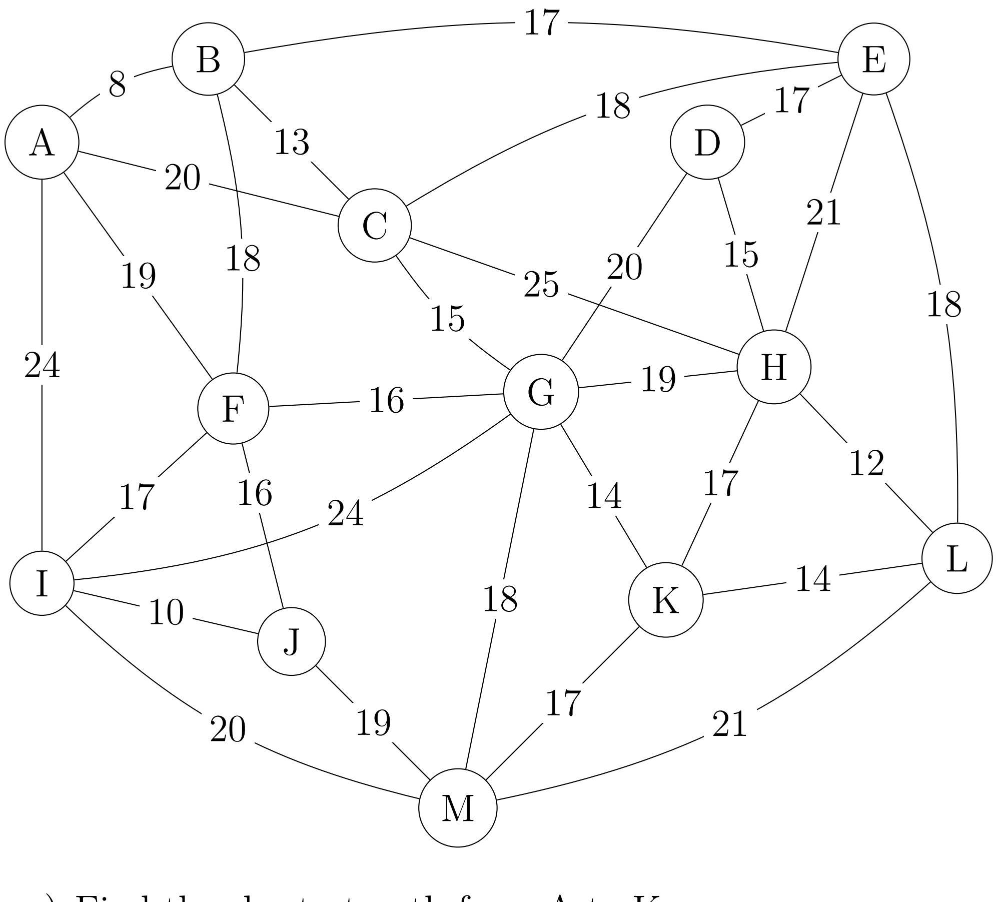


a) Find the shortest path from E to C.

b) Find the shortest path from D to E.

c) Draw a minimum spanning tree.

4. For the following graph:



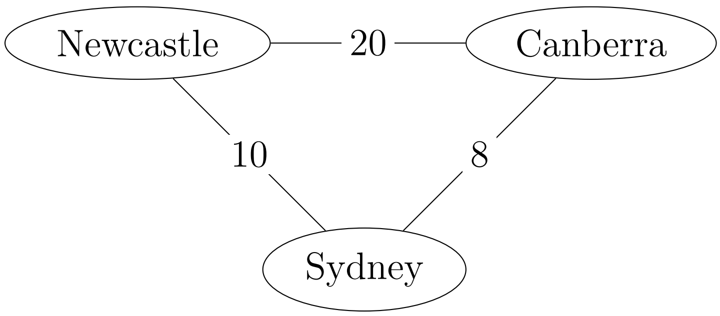
a) Find the shortest path from A to K.

b) Find the shortest path from I to E.

c) Draw a minimum spanning tree.

5. Consider that we want to try a simple greedy algorithm for finding the shortest path. In our new algorithm we will walk along the shortest edge leading out from the node we are currently at, that takes us to a node we have yet to visit. This algorithm will likely work for many graphs, but can you think of an example graph where this algorithm would work? and a graph where it would not?

6. The following graph is an abstract representation of the cost, due to toll roads, of driving from one city to another.

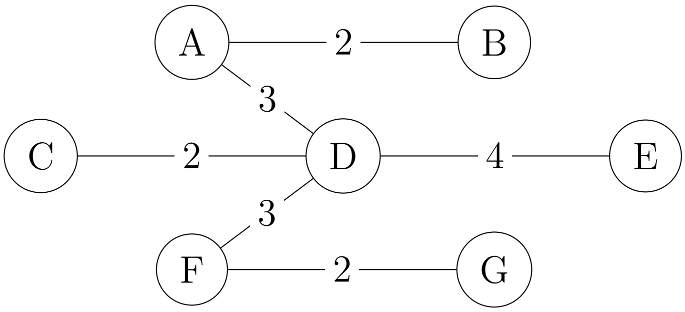


For example the cost of travelling from Newcastle to Canberra would be \$18 via Sydney or \$20 directly. In order to reduce car through traffic in Sydney, the city plans to charge cars \$6 when they enter the city. This would mean a trip from Newcastle to Canberra via Sydney would now be \$24, and similarly a trip from Newcastle to Sydney would be \$16. Conversely, a trip from Sydney to Canberra should still only cost \$8. Can you design a graph that represents this new change to the toll system? (You might need to use a directed graph.)

Answers

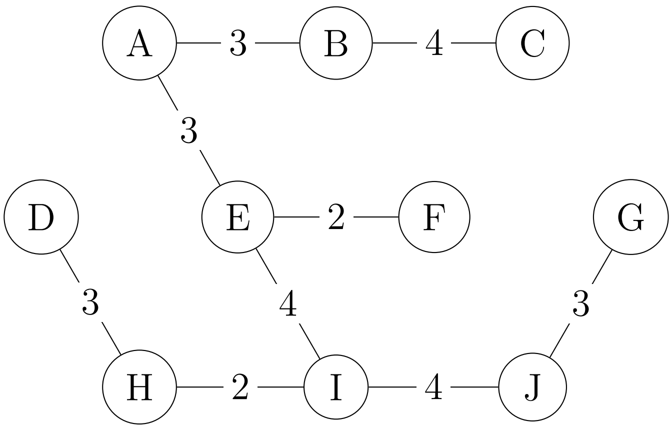
1. a) C → D → F → G, weight of 7

   b) F → D → A → B, weight of 8

   c) 

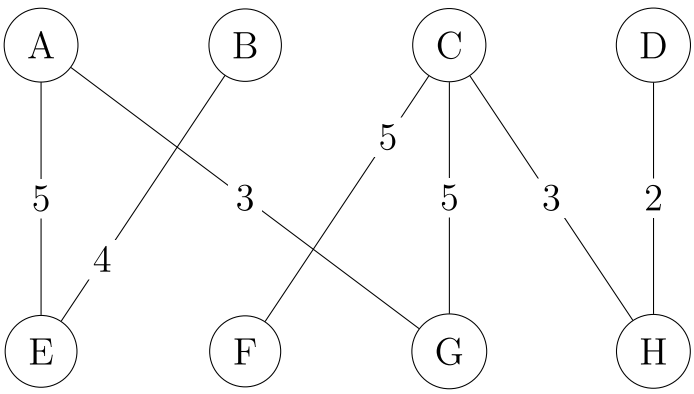
2. a) C → B → A → D, weight of 11

   b) H → D → A → B, weight of 10

   c) 

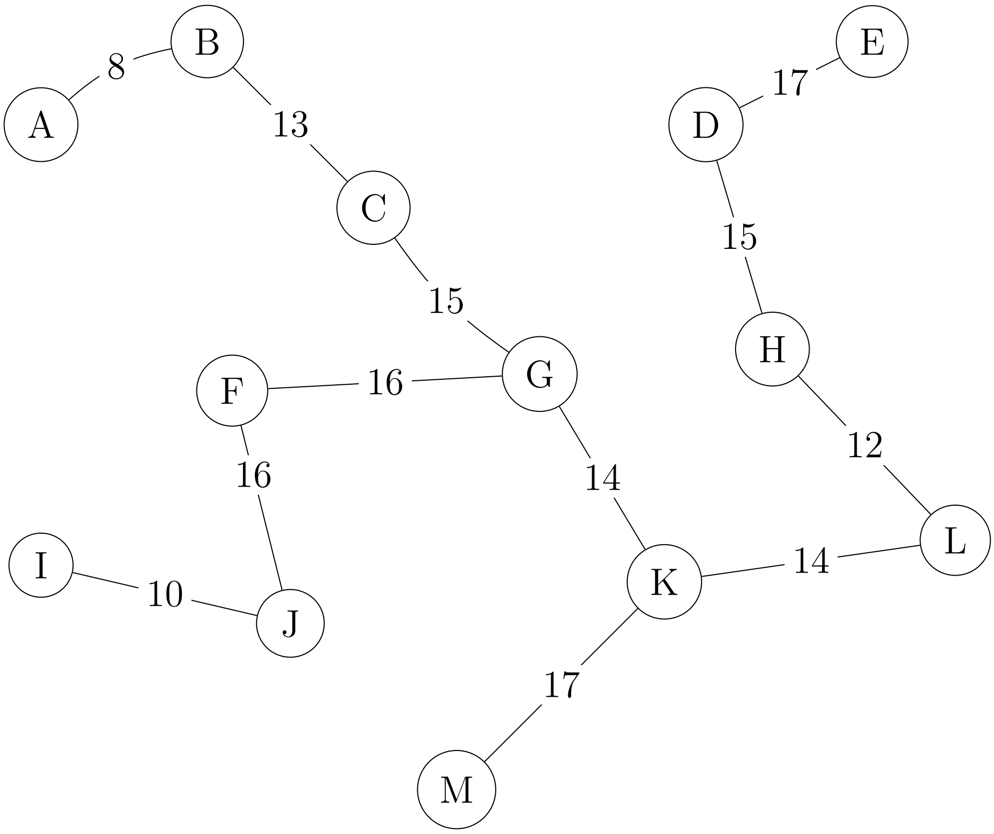
3. a) E → A → G → C, weight of 13

   b) D → H → C → G → A → E, weight of 18

   c) 

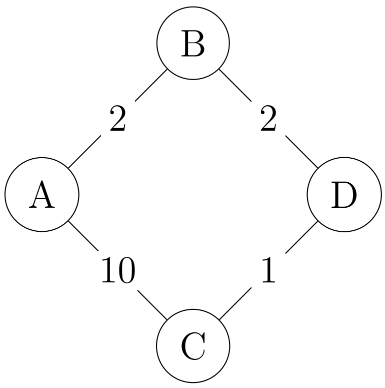
4. a) A → C → G → K or A → F → G → K, weight of 49

   b) I → G → C → E, weight of 57

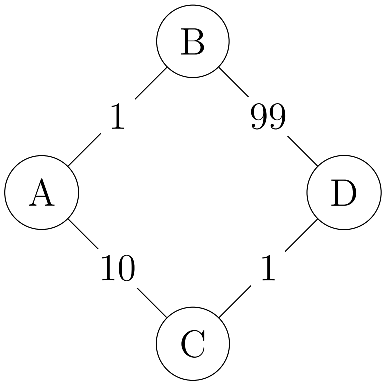


c)

5. A simple graph for which this algorithm does work is the following graph if we want to find the shortest path from A to D.



A graph for which this algorithm would not might look like:



Because first we would travel A $\rightarrow$ B, but now we are locked into travelling along the edge from B $\rightarrow$ D with a weight of 99.

6. An example of using a directed graph to model this