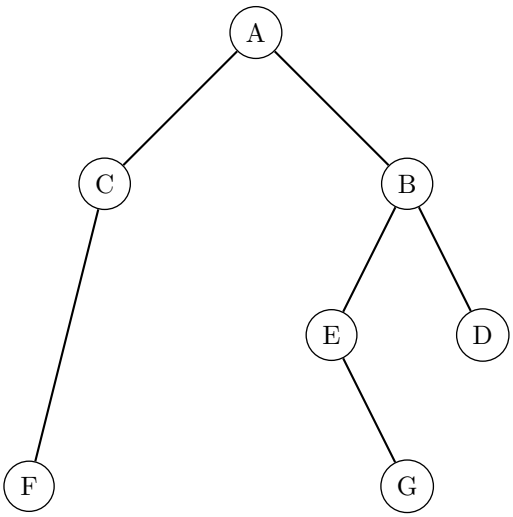
CAB203
Tutorial 8

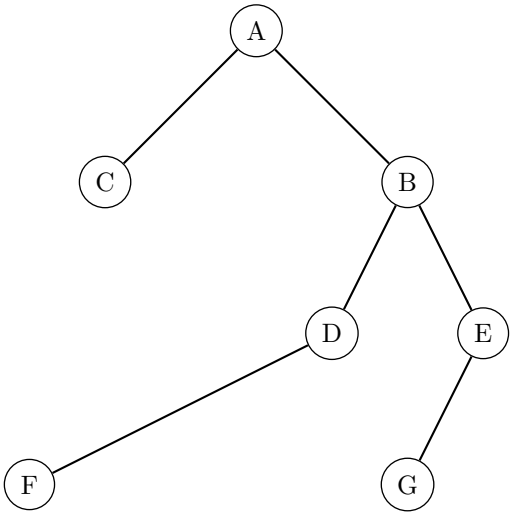
1 Trees

1. For each rooted tree with root A, give determine the ancestors, descendants, parent, and children of the vertices C and D. For each tree give the set of leaves.

   (a)

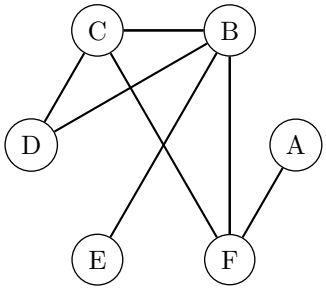


   (b)

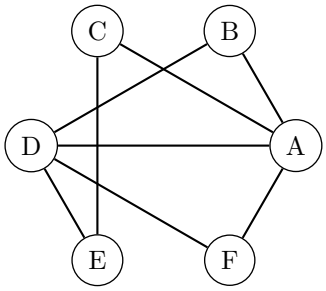


2. For the following graphs, give a spanning tree with root A such that all paths from A to another vertex u in the tree are the minimal length.

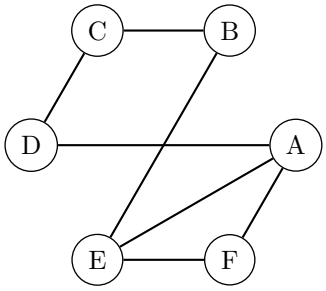
   (a)



(b)



(c)



3. A power company needs to connect up several towns with power. The company needs to apply for permission for every power line that it installs (one permission per line connecting two towns). It would be impractical to construct a direct line between some towns due to challenging geography. The company wants to design a network that requires the least number of permissions, connects all the towns together, and avoids the impractical lines.

2 Case study: OSPF

Open Shortest Path First is a protocol used in networking for dynamically generating routing tables on medium sized networks with many routers. (By contrast, BPG is the protocol used between routers at the ISP level.) We will explore a simplified version of the problem that OSPF solves, which we will call the *routing table problem* (RTP).

Routers are assumed to already know about other routers that they are directly connected to, i.e. their neighbours. Each router broadcasts its list of neighbours to all other routers. Then each router reconstructs the graph of the entire network and constructs a *routing table* for itself, which tells the router, for each possible destination router, the first step on a shortest path to that destination.

1. Routers receive information from each other as lists of neighbours. Such a collection of lists is known as an *adjacency list* and it is common for graphs to be given in such a format (which is more efficient for certain applications.) How should the adjacency list for all routers be modelled:

   (a) using mathematical notation?

   (b) in Python?

2. Given an adjacency list in the format given above, how can we find the vertex and edge sets of the graph?

   (a) using mathematical notation?

   (b) using Python?

3. Write a short Python function that takes a graph (V, E) and vertices u and v and returns the *first* vertex (after u) on a shortest path from u to v in the graph. We will call this vertex the *first hop*. You may make calls to previous functions in the unit. What should you do if $u = v$?

4. Write a short Python function that takes a graph (V, E) and starting vertex u returns a routing table which gives the first vertex for each destination vertex v. What form should the return take?

5. Combine the above to create a function which solves the RTP: given an adjacency list and a starting vertex, give a routing table that gives the first hop for any destination vertex.