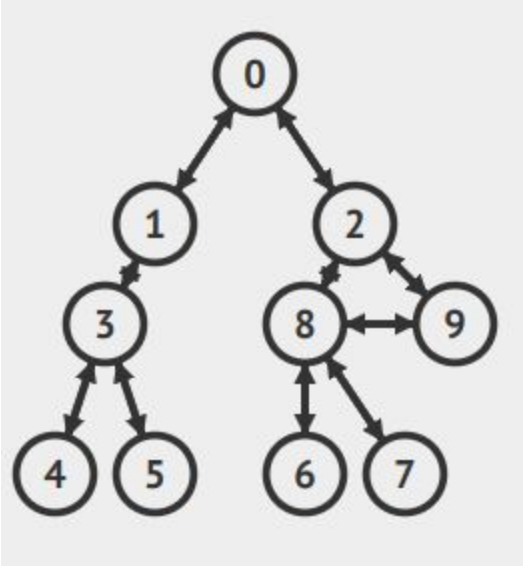
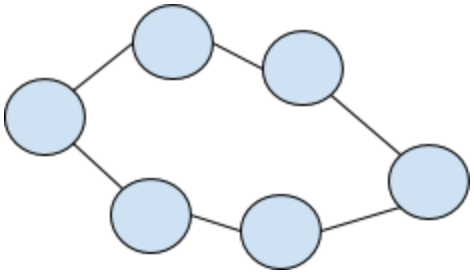
3.6:
Theorem 3.7 states that in a DFS search tree all non-tree edges connect ancestors to descendants. If the non-tree edge connects over more than one generation, the BFS search tree will include that edge, or else two connected nodes would differ by more than one against Theorem 3.4. The only case we have to left is where we have a triangle like in this picture:



(from https://visualgo.net/en/dfsbfs)
but even here, the trees will be different. BFS will still follow the tree cycle and not include the cross edge, since 8 and 9 are equidistant from 0. DFS will include the cross edge since it starts from either 8 or 9 and sees there is an unexplored node connected to it before it gets back to 2. Therefore the only way BFS and DFS could have the same search tree is if they both use all the edges in a graph.
If the only edges for a tree, both DFS and BFS search trees will have to follow this tree.
3.7.
True:
For a graph to not be connected there must be two nodes *x, w* which are in different connected components.
The size of *x*'s connected component must be n/2 +1 when we include *x* itself
The same is true for *w*.
The total number of nodes required for 2 such components is n + 2, which is clearly impossible since there are only n nodes in the graph.

3.9 A minimal network with two independent paths looks like this:



If L is the length of the path, for 2 paths that do not share any nodes you need 2 * (L -1) + 2 = 2L nodes, L - 1 nodes for each path and one nodes for the start and one for the end. Since we require L to be greater than n/2, we need more than n nodes to make the paths. Since the network has only n nodes this is impossible. To find the node we run BFS on the graph, this will produce a number of layers, to find the node we need to find the layer with only one node and remove it.