School of Computing and Information Systems
COMP90038 Algorithms and Complexity Tutorial Week 5

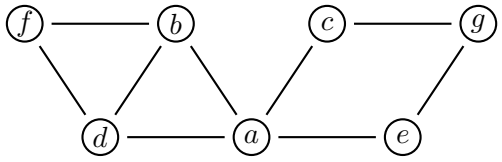
1. *Sparse* graphs $\langle V, E\rangle$ are ones for which $|E| \in O(|V|)$, that is, the number of edges stay, asymptotically, in the order of the number of nodes. Which representation is better for sparse graphs, adjacency matrices or adjacency lists?

2. Draw the undirected graph whose adjacency matrix is

	a	b	c	d	e
a	0	1	1	0	0
b	1	0	1	0	0
c	1	1	0	1	1
d	0	0	1	0	1
e	0	0	1	1	0

   Starting at node a, traverse the graph by depth-first search, resolving ties by taking nodes in alphabetical order.

3. Consider the following graph:



   (a) Write down the adjacency matrix representation for this graph, as well as the adjacency list representation (assume nodes are kept in alphabetical order in the lists).

   (b) Starting at node a, traverse the graph by depth-first search, resolving ties by taking nodes in alphabetical order. Along the way, construct the depth-first search tree. Give the order in which nodes are pushed onto to traversal stack, and the order in which they are popped off.

   (c) Traverse the graph by breadth-first search instead. Along the way, construct the breadth-first search tree.

4. In the lectures we discussed how to adapt depth-first traversal to decide whether an undirected graph is cyclic. We identified this complication: Suppose we are at node v and the call DfsExplore(v) takes us to the previously un-visited node w. As part of the cycle test we wanted to check whether w has some neighbour that has already been visited (and if so, classify the graphs as cyclic). However, v is a neighbour of w and v has been visited. So using this approach, every graph that has an edge will be deemed cyclic!

   Write an algorithm to classify all edges of an undirected graph, so that depth-first tree edges can be distinguished from back edges. Then show how this algorithm can be adapted to decide whether an undirected graph is cyclic.

5. Explain how one can use breadth-first search to see if a graph has cycles. Which of the two traversals, depth-first and breadth-first, will be able to find cycles faster? (If there is no clear winner, give an example where one is better, and another example where the other is better.)

6. Explain how one can use depth-first search to identify the connected components of a graph.

7. Design an algorithm to check whether an undirected graph is 2-colourable, that is, whether its nodes can be coloured with just colours in such a way that no edge connects two nodes of the same colour. Hint: Adapt one of the graph traversal algorithms.

8. **Just for fun** ...

   Given an 8-pint jug full of water, and two empty jugs of 5- and 3-pint capacity, get exactly 4 pints of water in one of the jugs by completely filling up and/or emptying jugs into others. Solve this problem using breadth-first search.