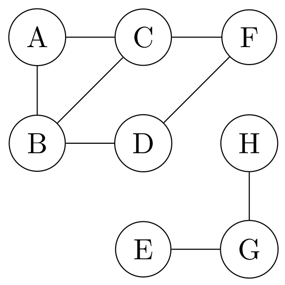
2019 Semester 2
COMP20003 Algorithms & Data Structures
School of Computing and Information Systems
The University of Melbourne

Week 10 Exam Style Practice Problems

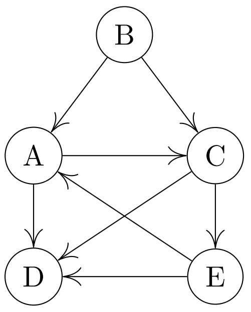
Tobias Edwards
Solutions by Akira Wang

Question 1. Consider the following graphs:

(i)



(ii)



Answer the following questions for graph (i) and graph (ii):

(a) Which graph is **directed**, and which is **undirected**?

Graph (ii): directed.
Graph (i): undirected.

(b) Give a **path** from A to D in the graph of length 2.

Graph (i): $A \to B \to D$
Graph (ii): $A \to C \to D$

(c) Which node in the graph has the highest **degree**? In the case of the directed graph, give the node with the highest **in-degree** (*i.e.,* the node with the most incoming edges).

Graph (i): Nodes C, B
Graph (ii): Node D

(d) Identify the **connected components**[1] in the graph.

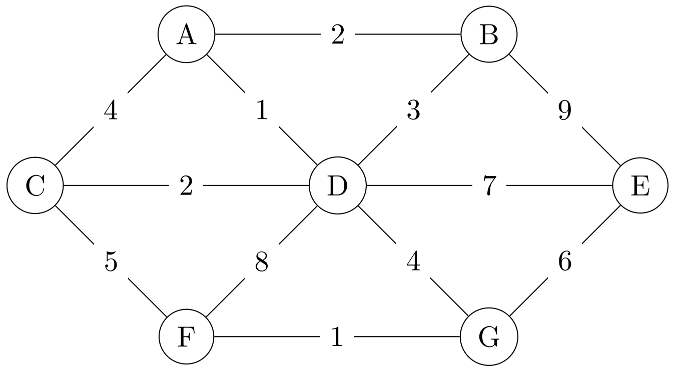
Graph (i): Nodes A, C, F, D, B and E, G, H
Graph (ii): Nodes A, C, E and B and D

(e) Give an example of a **cycle** in the graph.

Graph (i): Nodes A, C, B or A, C, F, D, B or B, C, F, D (and so on...)
Graph (ii): Nodes A, C, E

[1]Note that the corresponding concept in a directed graph is called a *strongly connected component*, which is a set of nodes such that there is a path between every pair of nodes (or, alternatively every node can reach every other node).

Question 2. Consider the following *weighted undirected graph*:



Answer the following questions by running the specified graph algorithm. For all parts break ties by visiting nodes in alphabetic order.

(a) Use the **depth first search** graph traversal algorithm to find a path from A to F.

> Recall that a DFS uses a stack and expands the deepest node first (ignoring all weights). We will also keep in mind a Boolean "visited" array so we don't visit already expanded nodes.
>
> Traversal: $A \to B \to D \to C \to F$

(b) Use the **breadth first search** graph traversal algorithm to find a path from A to F.

> Recall that a BFS uses a queue and expands the adjacent nodes first (ignoring all weights). We will also keep in mind a Boolean "visited" array so we don't visit already expanded nodes.
>
> Traversal: $A \to C \to F$

(c) Use **Dijkstra's** algorithm to find the *shortest path* from A to F.

> Recall that Dijkstra's uses a priority queue (heap) using the edge weight as the priority.
>
> Traversal: $A \to D \to G \to F$

(d) Which algorithm gave the **shortest path**?

> Dijkstra's.

(e) Dijkstra's algorithm is guaranteed to give the shortest path in graphs without negative edge weights. In which situations would **breadth first search** always give the **shortest path**?

> BFS will always give the shortest path if there are no edge weights or they are all uniform (i.e all of weight 1). In addition, there must be no loops in the graph.