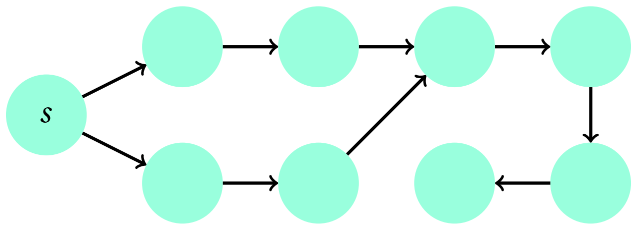
# Week 5 Preparation

(Solutions)

**Useful advice:** The following solutions pertain to the preparation problems. You are strongly advised to attempt the problems thoroughly before looking at these solutions. Simply reading the solutions without thinking about the problems will rob you of the practice required to be able to solve complicated problems on your own. You will perform poorly on the exam if you simply attempt to memorise solutions to these problems. Thinking about a problem, even if you do not solve it will greatly increase your understanding of the underlying concepts.

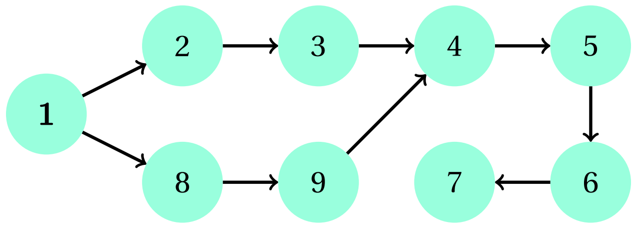
## Problems

**Problem 1.** Label the vertices of the following graph in the order that they might be visited by a depth-first search, and by a breadth-first search, from the source $s$.

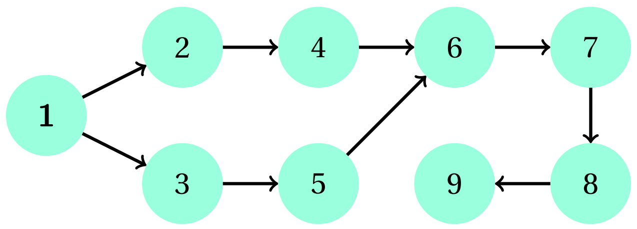


**Solution**

There are two possible solutions for depth-first search depending on which order you traverse the edges. One valid order is the following.



The other valid order is to visit 8 & 9 before 2 & 3. A valid order for breadth-first search is the following.



The other valid order swaps nodes 2 and 3 with each other, and nodes 4 and 5 with each other.

**Problem 2.** Write pseudocode for an algorithm that given a directed graph and a source vertex, returns a list of all of the vertices reachable in the graph from that source vertex. Your algorithm should run in $O(V + E)$ time.

**Solution**

This problem can be solved with a depth-first search or breadth-first search. The vertices reachable from a given vertex are simply those that are visited by a search when that vertex is the starting node. So we simply perform a DFS from $s$ and then return a list of the nodes that were visited.

```
function REACHABLE(G = (V, E), s)
    Set visited[1..$n$] = False
    DFS(S)
    Set reachable = empty array
    for each vertex u = 1 to $n$ do
        if visited[u] then
            reachable.append(u)
    return reachable

function DFS(u)
    visited[u] = True
    for each vertex v adjacent to u do
        if not visited[v] then
            DFS(V)
```