Problem 1 (Cycle Detection, 35 pts)

In this problem you will show how to detect whether an undirected graph $G = (V, E)$ has a cycle in time $O(|V|)$ (independent of the number of edges).

(a) Prove that an undirected graph with $|V|$ edges *must* contain a cycle.

(b) The above is not true for directed graphs. What is the maximum number of edges a DAG with n nodes can have. Briefly justify your answer. (It may help to remember our maxim: if you see a DAG think about a topological ordering of its vertices!)

(c) Explain how to modify an algorithm from class to detect if the undirected graph G contains a cycle in $O(|V|)$ time.

Problem 2 (Failed Magic, 20 pts)

In Lecture 17, we saw a magic trick that can turn a BFS algorithm into a DFS algorithm by simply replacing the queue with a stack. But I mentioned that this magic trick does not work for all BFS algorithms. In this problem, we will see why.

Below is the BFS algorithm from Lecture 16, but with the queue replaced by a stack attempting to make it into a DFS. Find a counterexmaple, i.e. construct a graph $G = (V, E)$ (can be either directed or undirected), such that for some vertex $s \in V$, running DFS-Visit(G, s) does not correctly give you a DFS tree. The issue is with **the sequence of vertices** that you visit, so you may safely ignore the teal-colored lines for the distances, as they are not used for DFS.

You should specify the graph G (drawing it out is fine) and the vertex s that you run DFS on, along with a quick explanation of what goes wrong with the (incorrect) DFS tree it produces. (*Hint: A simple example only needs 3 vertices.*)

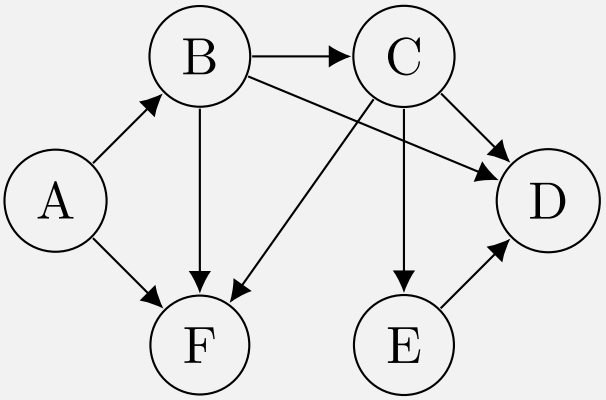
DFS-Visit(G, s)

```
1:  for all v ∈ V do
2:    dist(v) ← ∞
3:    parent(v) ← ⊥
4:    color(v) ← WHITE
5:  end for
6:  dist(s) ← 0
7:  color(s) ← YELLOW
8:  Stack.push(s)
9:  while Stack is not empty do
10:   u ← Stack.pop()
11:   for all v ∈ Adj[u] do
12:     if color(v) = WHITE then
13:       dist(v) ← dist(u) + 1
14:       parent(v) ← u
15:       color(v) ← YELLOW
16:       Stack.push(v)
17:     end if
18:   end for
19:   color(u) ← GREEN
20: end while
```

Problem 3 (Topological Sort, 20 pts)

How many valid topological sorts does the directed graph below have? List all the valid topological sorts in the following table. One of them has been listed as an example, where node A is output first and D is output last.

1.	A	B	C	F	E	D
2.						



Problem 4 (Number of Simple Paths, 25 pts)

Give an algorithm that given a Directed Acyclic Graph (DAG, defined as a directed graph without cycles) and two vertices s, t, returns the *number* of simple paths from s to t. Your algorithm should run in time $O(|V| + |E|)$. Justify the correctness and runtime of your proposed algorithm. (Your algorithm does not need to list the simple paths, only needs to give the count.)

(*Hint: What is the letter after C? What is the letter before Q?*)