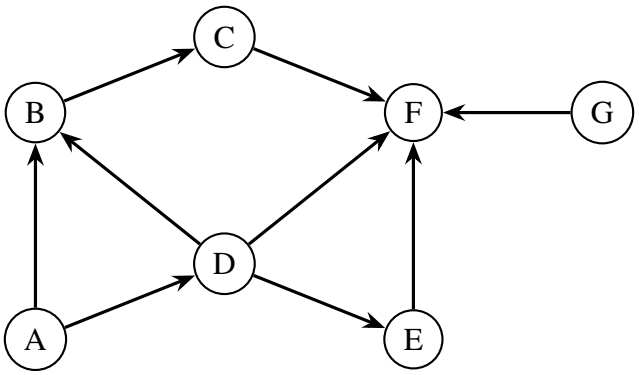
1 Datastructures and Graph Traversals

In order to implement DFS and BFS, we must add neighbors to a data structure (usually called the fringe) each time we visit a node, but the choice of data structure is very important and can help differentiate between DFS and BFS. What data structure should we use for DFS and why? What data structure should we use for BFS and why?



2 DFS and BFS (again)

Give the DFS preorder, DFS postorder, and BFS order of the graph starting from vertex *A*. Break ties alphabetically.

3 Topological Sorting

Give a valid topological sort of the graph above. (Hint: Use the reverse postorder.)

4 Regex Practice

Write a valid Regular Expression for the following scenarios:

1. The words spot, sp?t, spitter, and respite are matched, yet the words pt, spt, and part are not. (Hint: Look at the relationship between the p and the t):

2. The expression correctly matches any new dank start-up memes. A dank start-up meme can be identified as it starts with one of the words 'Data', 'App', 'my', 'on', 'un', contains 1 or more numbers or letters not including i or 3 in the middle, and ends with one of the following suffixes: ly, sy, ify, .io, .fm, .tv

5 Extra for Experts: Shortest Directed Cycles

Provide an algorithm that finds the shortest directed cycle in a graph in $O(EV)$ time and $O(E)$ space, assuming $E > V$.

6 Extra for Experts: DFS Gone Wrong

Consider the following implementation of DFS, which contains a crucial error:

```
create the fringe, which is an empty Stack
        push the start vertex onto the fringe and mark it
        while the fringe is not empty:
                pop a vertex off the fringe and visit it
                for each neighbor of the vertex:
                        if neighbor not marked:
                                push neighbor onto the fringe
                                mark neighbor
```

Give an example of a graph where this algorithm may not traverse in DFS order.