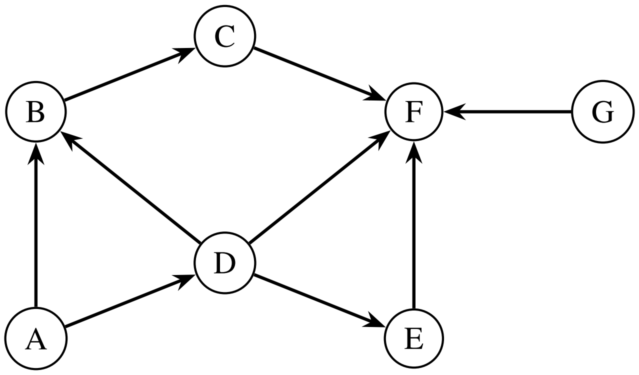
CS 61B Discussion 11 Spring 2017

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?

```
BFS explores/processes the closest vertices first and then moves outwards
    away from the source. Given this, you want to use a data structure that
    gives you the oldest element based on the order they were inserted. A
    queue is what you need in this case since it is first-in-first-out(FIFO).
Whereas DFS explores as far as possible along each branch first and then
    bracktracks. For this, a stack works better since it is
    LIFO(last-in-first-out). You will get depth as when you add a node last,
    you'll explore this node next.
```



2 DFS and BFS (again)

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

```
DFS preorder: ABCFDE
DFS postorder: FCBEDA
BFS: ABDCEF
```

3 Topological Sorting

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

```
One valid topological sort is GADEBCF. There are many others. In particular,
    G can go anywhere except after F, since it has no incoming edges and only
    one outgoing edge (to F).
```

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):

```
.*p.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

```
(on|un|my|Data|App)([0-24-9A-Za-hj-z]+)(ly|\.io|\.fm|\.tv|sy|ify)
```

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 $E > V$.

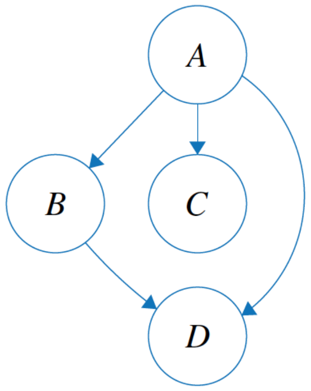
```
The key realization here is that the shortest directed cycle involving a
    particular source vertex is just some shortest path plus one edge back to
    s. Using this knowledge, we can create a shortestCycleFromSource(s)
    subroutine. This subroutine first runs BFS on s, then checks every edge
    in the graph to see if it points at s. For each such edge originating at
    vertex v, it computes the cycle length by adding one to distTo(x) (which
    was computed by BFS).

This subroutine takes $O(E+V)$ time because it is BFS. To find the shortest
    cycle in the entire graph, we simply call shortestCycleFromSource() for
    each vertex, resulting in an $V * O(E+V) = O(EV+V^2)$ runtime. Since $E > V$,
    this is just $O(EV)$.
```

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.


A
B
C
D