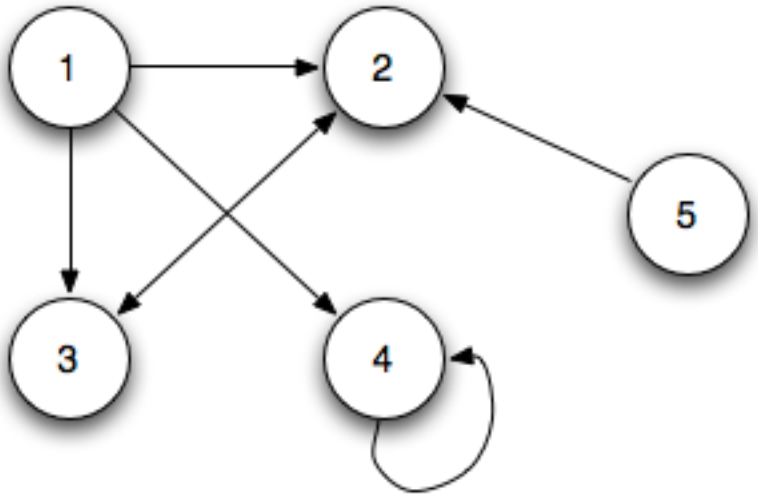
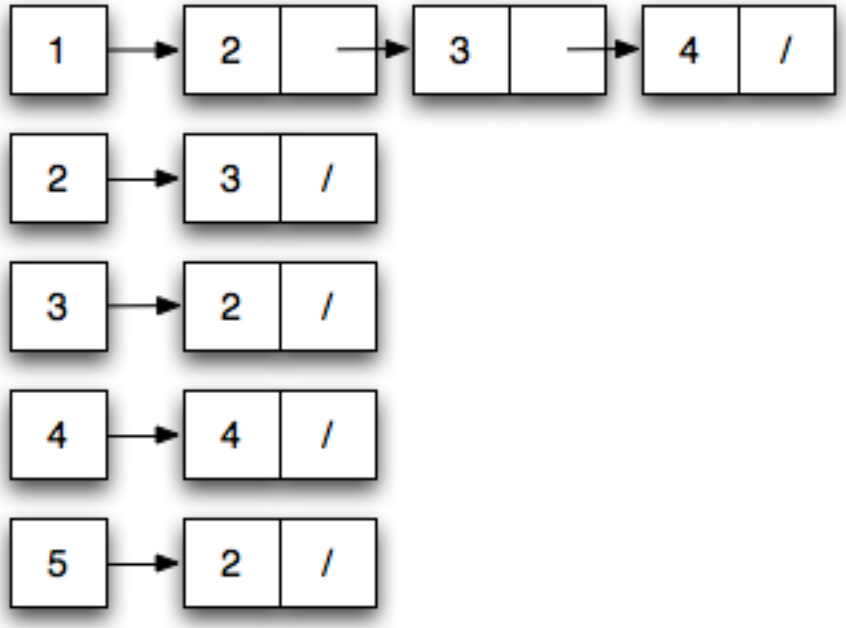
Graphs

We can represent a graph with an adjacency list

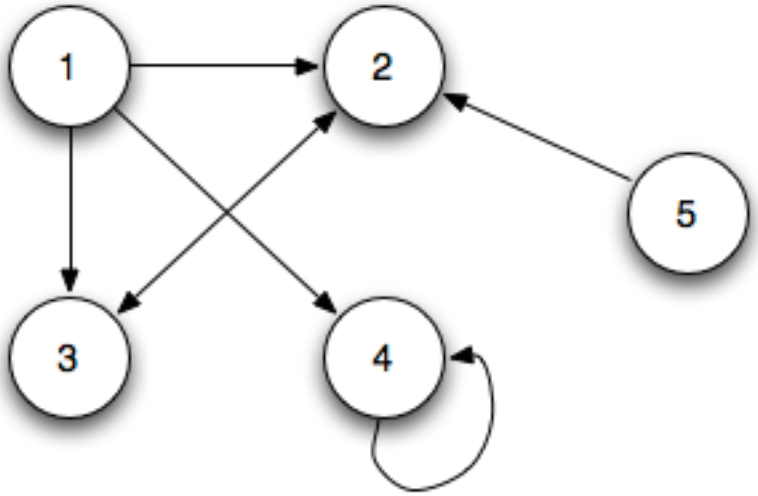




How much space does it take to represent a graph with V vertices and E edges using adjacency lists?

Graphs

We can represent a graph with an adjacency matrix

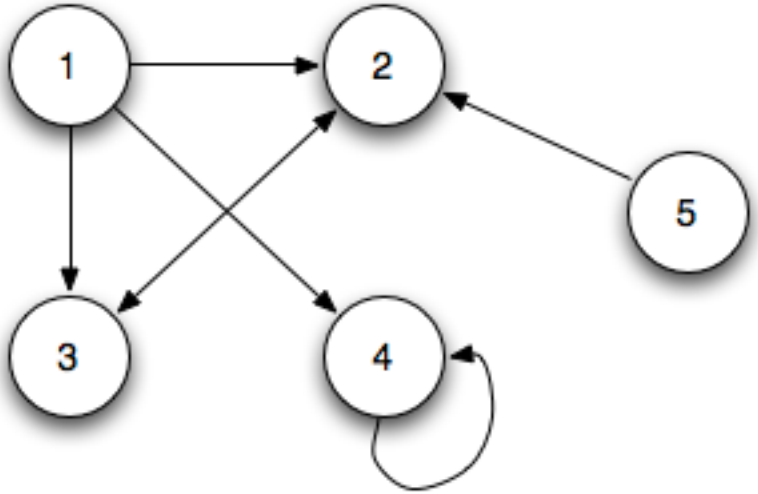


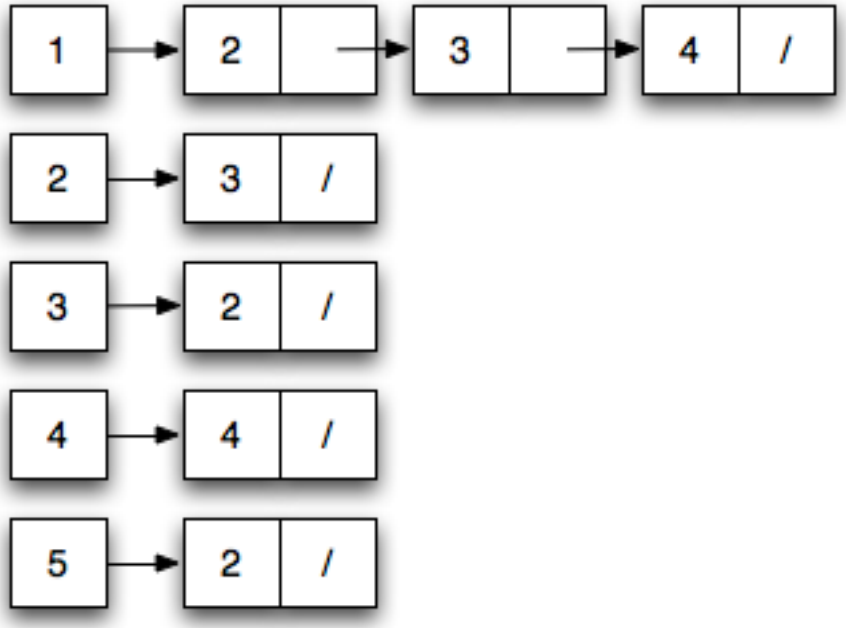
	1	2	3	4	5
1	0	1	1	1	0
2	0	0	1	0	0
3	0	1	0	0	0
4	0	0	0	1	0
5	0	1	0	0	0

How much space does it take to represent a graph with V vertices and E edges using adjacency matrix?

Graphs

We can represent a graph with an adjacency list

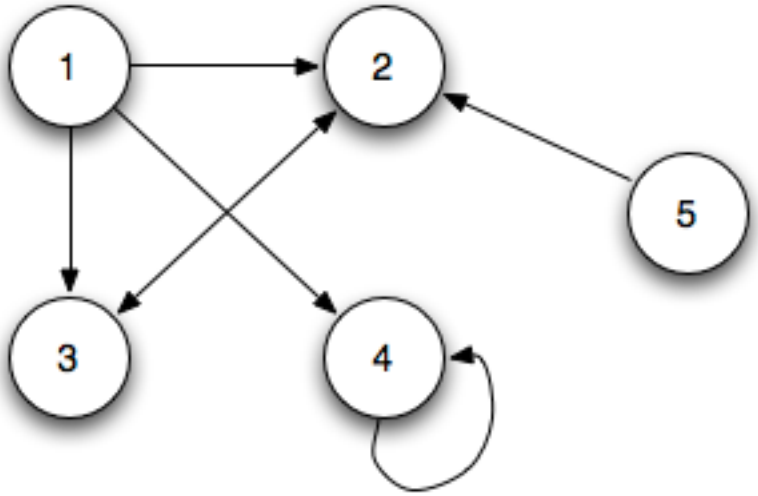




How much time does it take to add an edge (u,v) to a graph with V vertices and E edges using an adjacency list representation?

Graphs

We can represent a graph with an adjacency matrix

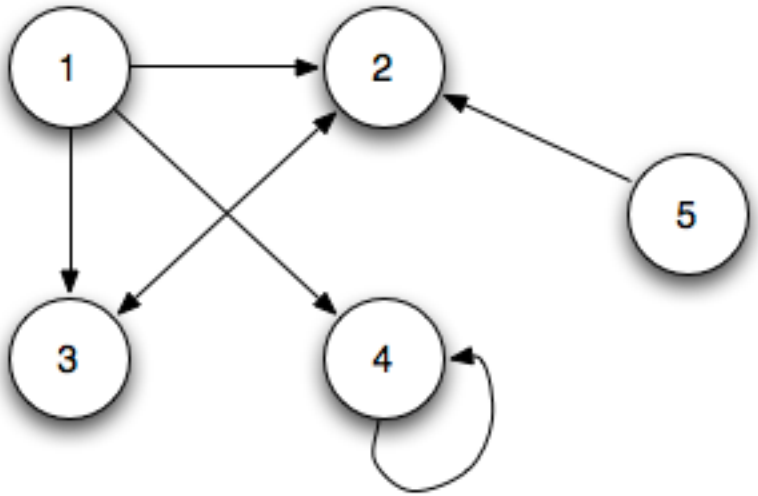


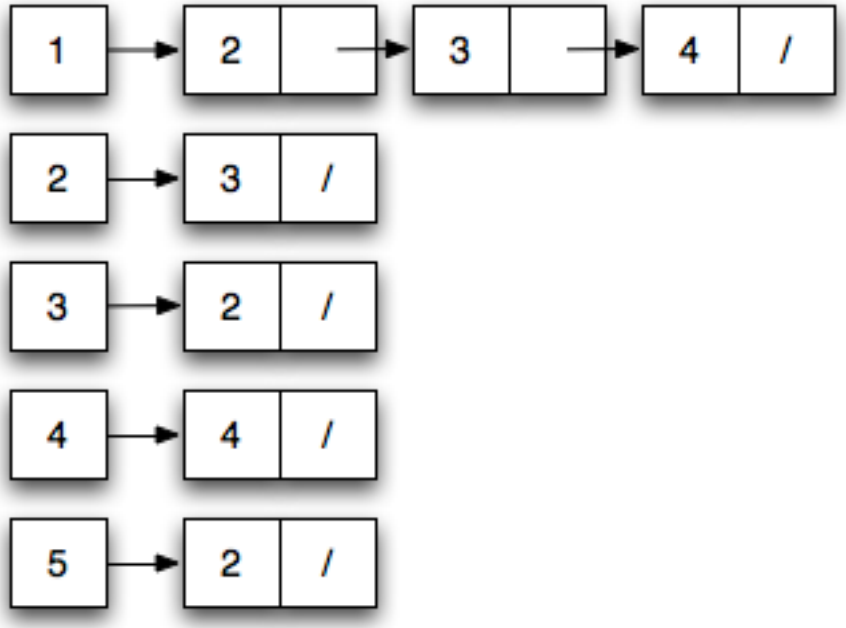
	1	2	3	4	5
1	0	1	1	1	0
2	0	0	1	0	0
3	0	1	0	0	0
4	0	0	0	1	0
5	0	1	0	0	0

How much time does it take to add an edge (u,v) to a graph with V vertices and E edges using an adjacency matrix representation?

Graphs

We can represent a graph with an adjacency list

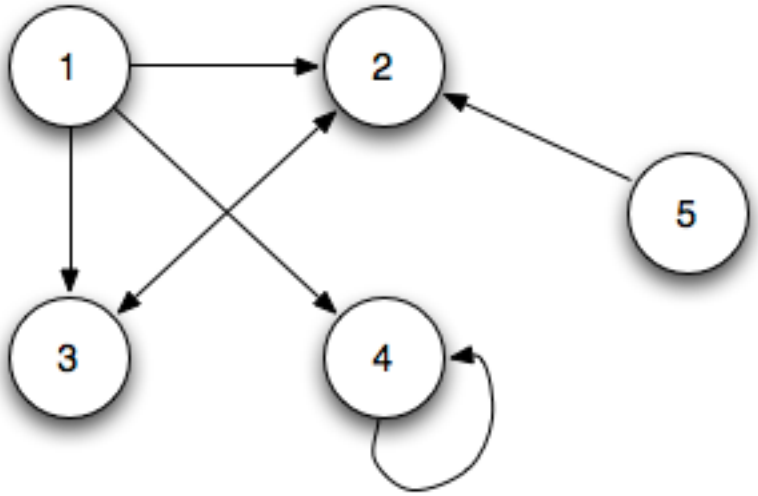




How much time to determine if there is an edge (u,v) in a graph with V vertices and E edges using an adjacency list, assuming that the degree of u is k?

Graphs

We can represent a graph with an adjacency matrix



	1	2	3	4	5
1	0	1	1	1	0
2	0	0	1	0	0
3	0	1	0	0	0
4	0	0	0	1	0
5	0	1	0	0	0

How much time to determine if there is an edge (u,v) in a graph with V vertices and E edges using an adjacency matrix, assuming that the degree of u is k?

Graph Traversals

Tricolor algorithm:

White nodes are undiscovered nodes that have not been seen yet in the current traversal and may even be unreachable.

Black nodes are nodes that are reachable and that the algorithm is done with.

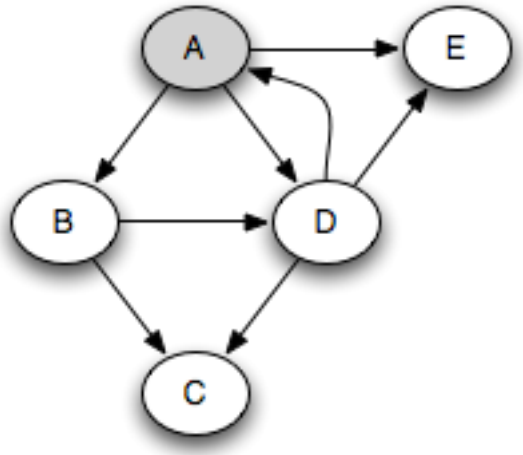
Gray nodes are nodes that have been discovered but that the algorithm is not done with yet. These nodes are on a frontier between white and black.

The algorithm pseudo-code is as follows:
Color all nodes white, except for the root nodes, which are colored gray.
While some gray node *n exists:*
color some white successors of n gray.
if n has no white successors, optionally color n black.

What different ways can we choose gray node n?

Depth First Search

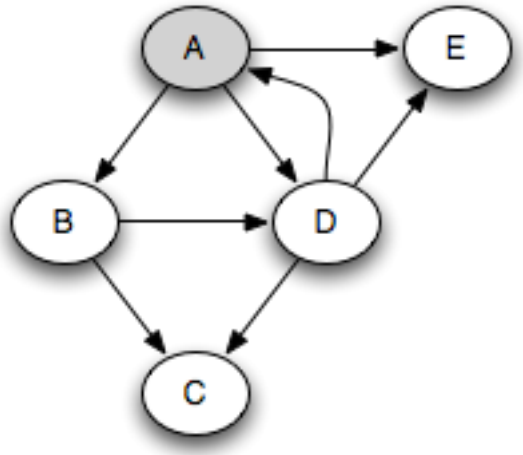
```
DFS(Vertex v) {
        set color of v to gray
        for each successor v' of v {
                if v' not yet visited {
                        DFS(v') }
        }
        set color of v to black
}
```



Trace through the algorithm with this graph marking vertices gray and black.
What order do they become gray?
What order do they become black?

DFS

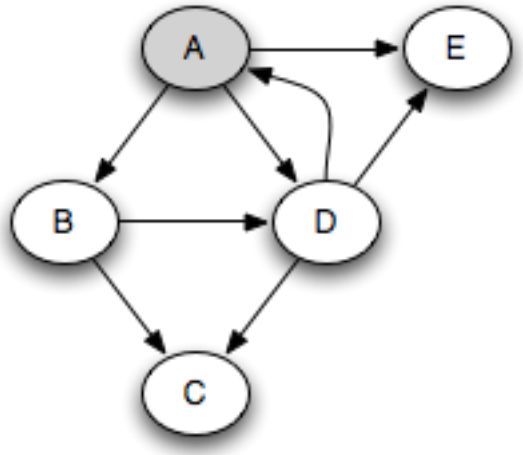
```
DFS(Vertex v) {
        set color of v to gray
        for each successor v' of v {
                if v' not yet visited {
                        DFS(v') }
        }
        set color of v to black
}
```



How many calls to DFS are made by the algorithm if we have V vertices and E edges?

DFS

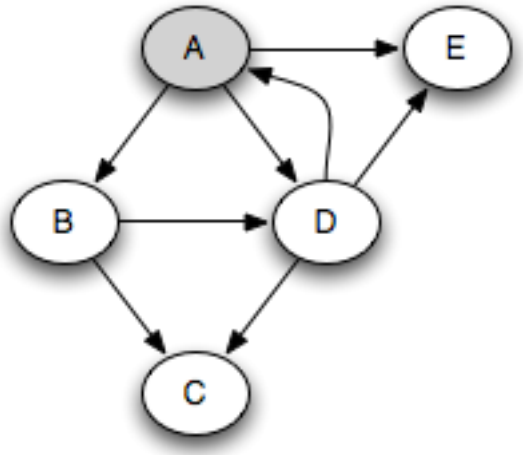
```
DFS(Vertex v) {
        set color of v to gray
        for each successor v' of v {
                if v' not yet visited {
                        DFS(v') }
        }
        set color of v to black
}
```



What is the total number of times the for loop will be iterated in the algorithm if we have V vertices and E edges?

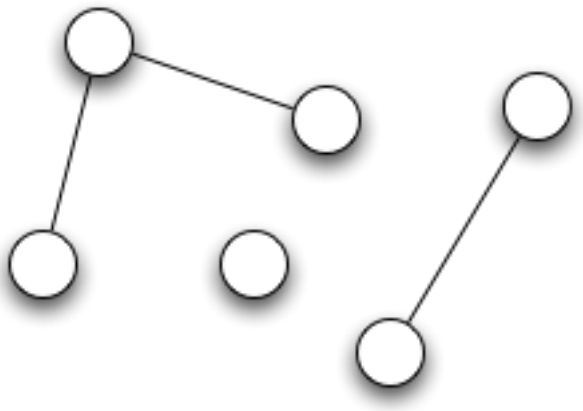
DFS

```
DFS(Vertex v) {
        set color of v to gray
        for each successor v' of v {
                if v' not yet visited {
                        DFS(v') }
        }
        set color of v to black
}
```



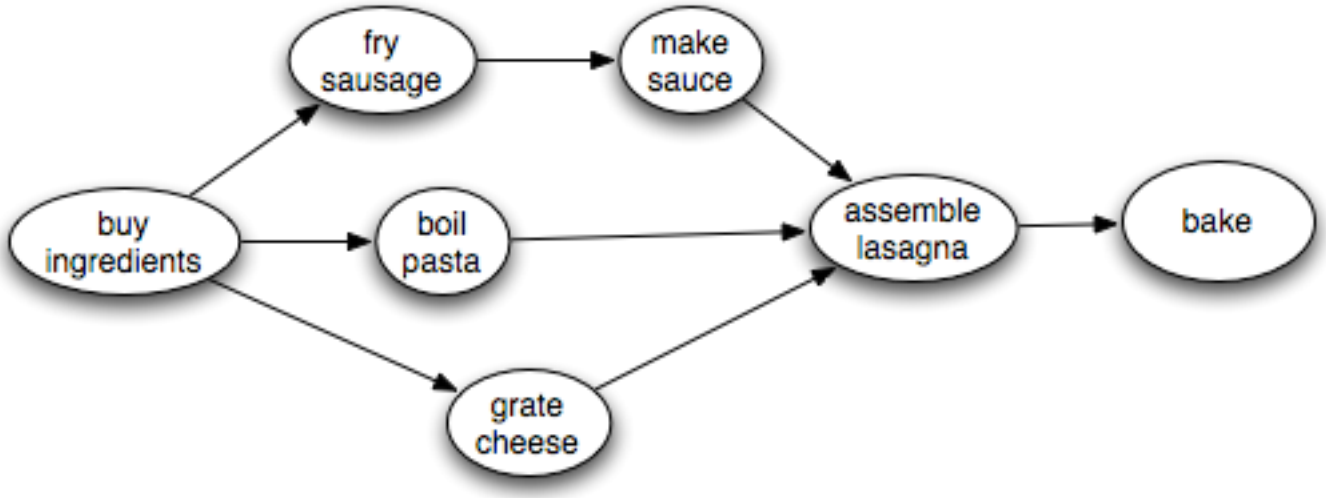
What is the order of growth for the time if we have V vertices and E edges?

Connected Components



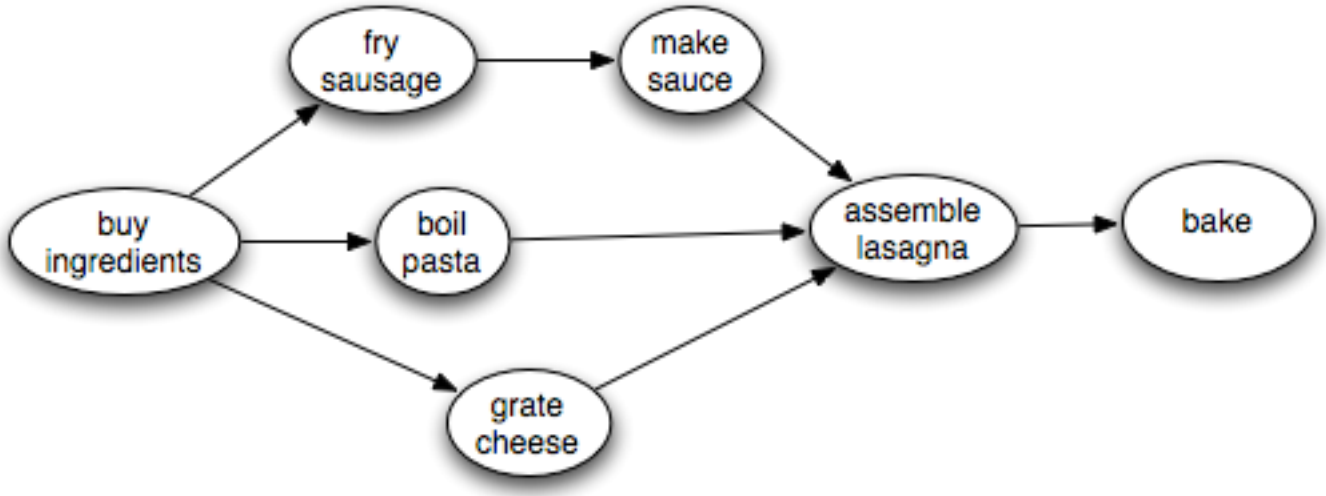
How can we use DFS to determine the number of connected components in a graph? Write the algorithm

Topological Sort



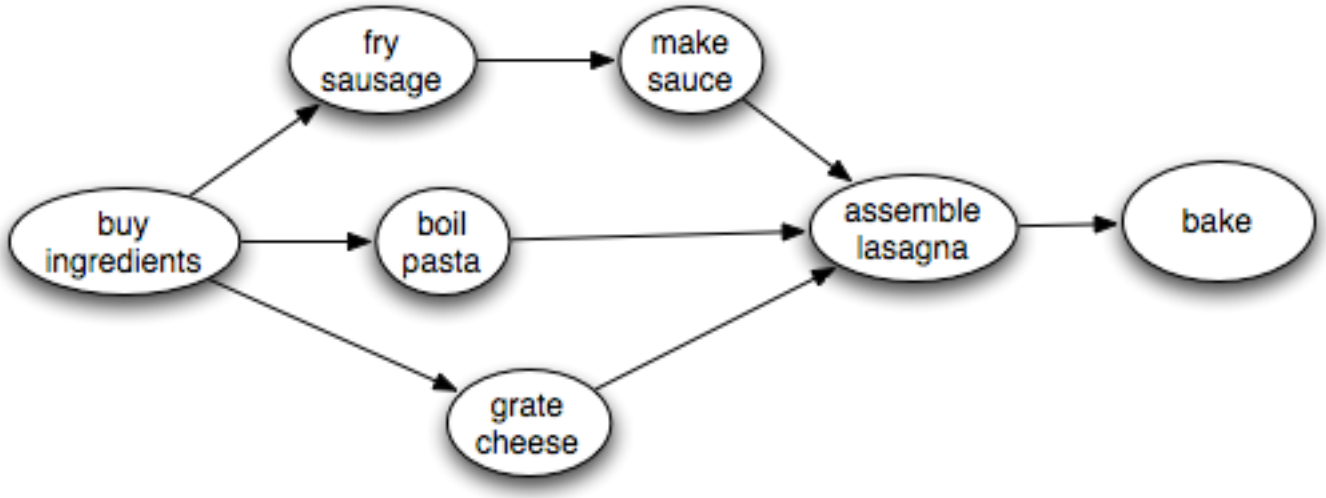
Give some ordering of these 7 steps that make sense.

Topological Sort



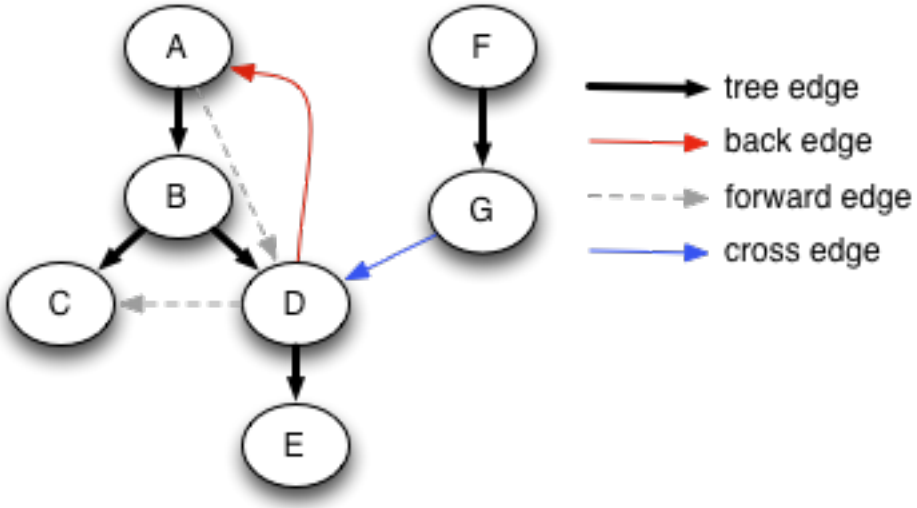
Perform a DFS on this graph and note the order in which nodes get marked black. What is your order?

Topological Sort



How can we use DFS to give a topological ordering? Write the algorithm

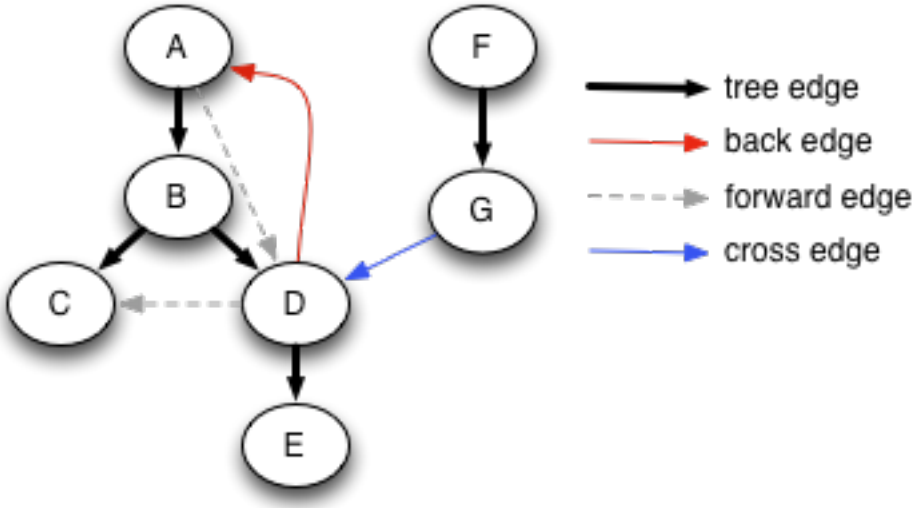
DFS – Edge Classification



Tree Edge: Destination node is white
Back Edge: Destination node is gray
Forward/Cross: Destination is black
Cross if goes from one tree to another

What do we know about a directed graph with no back edges?

Detect Cycles



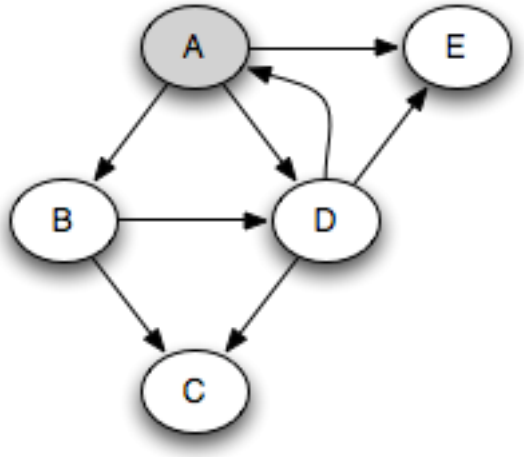
Tree Edge: Destination node is white
Back Edge: Destination node is gray
Forward/Cross: Destination is black
Cross if goes from one tree to another

How can we use DFS to determine if there is a cycle? Write the algorithm

Breadth First Search

```
BFS(Vertex root) {
        frontier = new Queue()
        mark root as gray
        frontier.enqueue(root)
        while frontier not empty {
            Vertex v = frontier.dequeue()
            for each successor v' of v {
                if v' white {
                    frontier.enqueue(v')
                    mark v' gray
                }
            }
            mark v as black
        }
}
```

Trace through the algorithm with this graph marking vertices gray and black.
What order do they become gray? What order do they become black?



Breadth First Search

```
BFS(Vertex root) {
        frontier = new Queue()
        mark root as gray
        frontier.enqueue(root)
        while frontier not empty {
            Vertex v = frontier.dequeue()
            for each successor v' of v {
                if v' white {
                    frontier.enqueue(v')
                    mark v' gray
                }
            }
            mark v as black
        }
}
```

How can we use BFS to find the shortest path (least number of edges) from a given vertex to a second vertex?