Topological Ordering
Topological Ordering Algorithm

Calculate in degrees // This might be done as the graph is built
Add vertices with in degrees of 0 to the queue
while the queue is not empty
    let i be the vertex removed from the queue
    add i to the topological ordering
    decrement the in degree of vertices adjacent to i by 1
    add new vertices with in degree of 0 to the queue
    increment the number of items in the topological order by 1

If the number of vertices in the topological order equals the number of vertices in the graph return the topological order
otherwise return the graph contains a cycle
Adjacency Matrix Implementation
public String topoSort() {

    indegree = new int[numVertices];
    queue = new LinkedList<>();

    calcIndegrees();
    initQueue();
    String topo = "";
    int topoCount = 0;
    while (queue.size() != 0) {
        int i = queue.remove();
        topo = topo+ ' '+ i;
        updateIndegrees(i);
        topoCount++;
    }
    if (topoCount == numVertices) return topo;
    return null;
}
private void calcIndegrees() {
    for (int i = 0; i < numVertices; i++) indegree[i] = 0;

    for (int j = 0; j < numVertices; j++)
        for (int i = 0; i < numVertices; i++)
            indegree[j] = indegree[j]+g[i][j];

}
private void initQueue() {
    for (int i = 0; i < numVertices; i++) {
        if (indegree[i] == 0) {
            queue.add(i);
        }
    }
}
public void updateIndegrees(int vertex) {
    for (int j = 0; j < numVertices; j++) {
        if (g[vertex][j] == 1) {
            indegree[j]--;
            if (indegree[j] == 0)
                queue.add(j);
        }
    }
}