CSCI 2300
Introduction to Algorithms: HW3

Sean Hyde
RIN: 662096071

Due: September 19th, 2024

1 Problem 3.8

(A)

This problem can be modeled with a directed graph, where each node is a possible "state" or next step to the solution. The nodes in the graph can be represented as (X, Y, Z) where $0 \leq X \leq 4$, $0 \leq Y \leq 7$, and $0 \leq Z \leq 10$.

We know from the pouring behavior that only one container is being poured into another during one operation, so, one container will stay the same. Then, either one container must be empty or the other container must be full in the next node. This is because pouring only ends based on two conditions:

1. One container is full.
2. The other container is emptied.

For example, if the starting node is $(4, 7, 0)$, the next nodes would be $(4, 0, 7)$ and $(0, 7, 4)$. Each of these nodes is producible from the previous node based on the constraints of the pouring behavior. Therefore, an edge can exist between these nodes as it shows that a legal move can be made.

An edge can exist between two nodes if when the operation to the next node is completed, the next node satisfies the following:

1. One container has stayed the same from the previous node as only one pouring/operation happens at one time.
2. Either, one container is empty EX: in node (X, Y, Z), $X = 0$. OR, one container is full EX: in node (X, Y, Z), $X = 4$.

If the node satisfies these conditions then an edge can be made to the node.

The question that must be answered relating to the graph is: Is there a path from the source node u that goes to a node v such that, in the format of (X, Y, Z), either $X = 2$ or $Y = 2$?

(B)

The algorithm that can be applied to this problem is Breadth-First Search (BFS). We would start from the source node u, and BFS will find all nodes within the graph. For each node, we will check if either $X = 2$ or $Y = 2$.

1.1 Runtime

BFS is a linear time algorithm which equates to O(V+E), as it is checking each node within the graph and all edges, where V is equal to all vertices of the graph, and E is all edges within the graph.

2 Problem 3.24

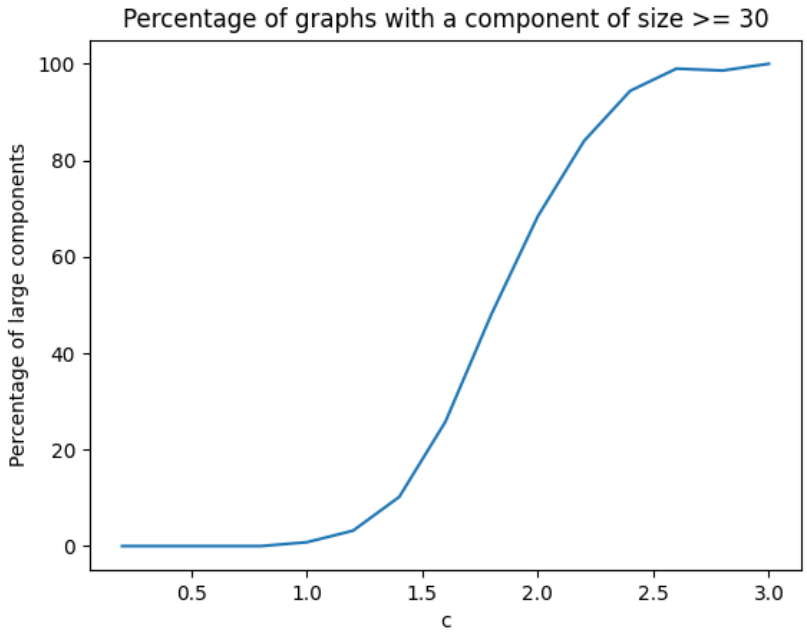
To determine if a directed acyclic graph G has a cycle, we can use the algorithm, topological sort. Since G is a DAG, a topological sort is the best choice. Topological sort will ensure that the graph for every edge from u to v, node u will appear before node v.

We can check whether, for every consecutive pair of vertices V_i and V_{i+1} in the topologically sorted order, there exists a directed edge between them. If this is true for all consecutive pairs, we know that there is a directed path that touches every vertex exactly once, as there is no way to travel backwards in a DAG.

2.1 Runtime

Topological sort is a linear time algortithm, as it can be computed by $O(V+E)$, where V equals the number of vertices in the DAG, and E equals the number of edges.

3 Problem 3


Percentage of graphs with a component of size >= 30
Percentage of large components
100
80
60
40
20
0
0.5
1.0
1.5
2.0
2.5
3.0
c