Problem: Find a build order given a list of projects and dependencies.

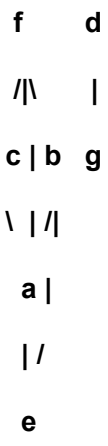
Constraints

- **Is it possible to have a cyclic graph as the input?**
  - **Yes**
- **Can we assume we already have Graph and Node classes?**
  - **Yes**
- **Can we assume this is a connected graph?**
  - **Yes**
- **Can we assume the inputs are valid?**
  - **Yes**
- **Can we assume this fits memory?**
  - **Yes**

Test Cases

- **projects: a, b, c, d, e, f, g**
- **dependencies: (d, g), (f, c), (f, b), (f, a), (c, a), (b, a), (a, e), (b, e)**
- **output: d, f, c, b, g, a, e**

Note: Edge direction is down



Test a graph with a cycle, output should be None

Algorithm

We can determine the build order using a topological sort.

- **Build the graph with projects (nodes) and dependencies (directed edges)**

- **Add initially non-dependent nodes to processed_nodes**
    - **If none exist, we have a circular dependency, return None**
- **While the length processed_nodes < the length of graph nodes**
    - **Remove outgoing edges from newly added items in processed_nodes**
    - **Add non-dependent nodes to processed_nodes**
        - **If we didn't add any nodes, we have a circular dependency, return None**
- **Return processed_nodes**

Complexity:

- **Time: O(V + E)**
- **Space: O(V + E)**