

Lab 9: Implement Tsort algorithm

In this lab, you will complete a provided partial implementation of a topological sort that mimics (but not exactly) the Unix command `tsort`. You should begin by reading about the [tsort](#) command and then using it until you have a good understanding of what it is and does. You can learn a bit more about `tsort` by entering `man tsort` at the command line. Press `q` to quit. You can learn even more by entering `info coreutils 'tsort invocation'` at the command line. Press `q` to quit.

A topological sort of a directed acyclic graph (DAG) is an ordering of its vertices such that, if there is a path from vertex v_1 to vertex v_2 , then v_1 must come before v_2 in the ordering. The graph must be acyclic (without cycles) because in a graph with a cycle there exist vertices with paths to themselves. This would imply that a vertex must come before itself (which it cannot). A simple algorithm for finding a topological sort is:

- Build an adjacency list for all of the vertices **and include** each vertex's *in degree* (number of incoming edges) as well as the specific vertices adjacent to it.
 - Store the adjacency list using a dictionary where the key is the string name of the vertex, and the value is some structure to hold the vertex's *in degree* and vertices that the vertex is adjacent to.
- Push all vertices with an *in degree* of zero on to a stack. Push the vertices in the order in which they were encountered while building the adjacency list.
 - For the implementation of a stack data structure, you must use your `Stack` class from `stack_array.py` from Lab 2 (and also used in Project 2). You must **add, commit, and push** a correct implementation of this file.
 - In order to keep track of the order in which the vertices were encountered, you should use a separate data structure. This will not necessarily be in alphabetical order, and this order cannot be determined by the adjacency list described above.
- While the stack is not empty:
 - Pop and output a vertex.
 - Reduce the *in degree* of the vertices that were adjacent to the just-popped vertex.
 - If reducing the *in degree* of a vertex results in a value of 0, push the vertex immediately.

The following starter files are available on GitHub. Complete the implementations and ensure that your implementations are committed and pushed to GitHub.

- `tsort.py`
- `tsort_tests.py`

4. Submission

Ensure that the following files are committed and pushed to your GitHub repository:

- `tsort.py`
- `tsort_tests.py`
- `stack_array.py`