James Sager

Topological Sorting Algorithm

CSCE 350 Program 3

Summary

The objective of the programming assignment is to be able to do topological sorting on a digraph. The program has been tested with several graphs, some of which cannot be topologically sorted. When this happens, the program generates an appropriate message.

Inputs

While most input data was directly provided, an additional input was generated *(modified for the program)*, using the University of South Carolina Computer Science Prerequisite Guide. This was done for the most part using the vim substitute command, which can basically used like find and replace, using a SED format.

Possible Algorithms

The algorithm chosen for topological sorting is the Recursive DFS Algorithm. It was chosen because because it seemed like less code than attempting to keep up with incoming and outgoing rays. If I were to consider the source removal algorithm, I would consider a list of parent nodes and consider pointers over integers. Assuming that a similar algorithm would be used to check for a cycle, *one involving recursively removing leaf nodes until either the tree is empty or a cycle is found*, I felt like the Recursive DFS Algorithm fit this assignment best.

DFS Pseudo Code

```
function topologicalSort( vector nodeGraph, vector orphanNodeList )
    for orphan in orphanNodeList
        recurseDFS( orphan, nodeGraph)
     print nodeStack from top

function recursiveDFS( Node n, Graph g)
    n.SetAsVisted
    for c in n.getChildren
        if c.isNotVisited
            recursiveDFS( c, g )
    nodeStack.push(n)
```

Description

- Start at an orphan node and set it as visited
- Visit children to the depth of the graph until a leaf is reached
- Push the leaf onto the stack
- Continue pushing nodes to the stack in reverse order until the original orphan node is reached
- If there are other siblings, follow those branches until leaves are found and repeat