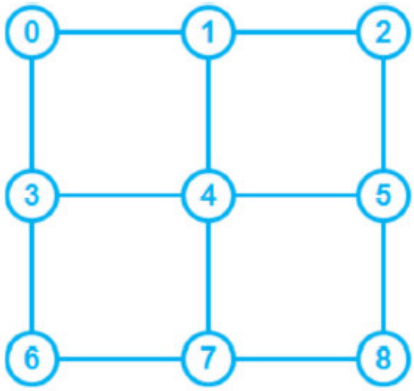
CSC 345 - Counting Snakes in a Box

In this project we will determine the number of simple, distinct paths in a graph from a specific node (called the source node) to a specific node (called the destination or target node).

Introduction - A graph is a collection of nodes connected by edges. Two nodes are adjacent if there is an edge connecting them. A path is a sequence of nodes connected by edges. A simple path is one that encounters each the nodes in a graph at most once. Two paths are considered to be distinct if there is any difference is the sequence of nodes encountered between the source and destination along the specified paths.

Consider the graph shown in Figure 1. This graph has nodes labeled 0 to 8, and 12 edges. For now assume that our source (starting) node will be 0 and the destination (target) node will be 8.



We want to count the number of distinct simple paths from node 0 to node 8. Example paths are provided here,

(0, 1, 2, 5, 8)
(0, 1, 4, 5, 8)
(0, 3, 4, 7, 8)
(0, 3, 6, 7, 5, 1, 2, 5, 8)

Note that two distinct paths can share some of the same intermediate nodes, so long as there is some difference in the sequence of nodes. Just for fun we will call these different paths *snakes in a box*. Now let's count the snakes.

Figure 1: *Snakes in a Box Graph*

Project - The goal of this project is to write a program that counts the number of distinct paths in a graph from a specified source node to a specified destination node. It is suggested that you start with the DFT project and modify it in the manner described to implement the path counter.

For the first step in our project, we will develop and implement a data structure to represent the graph. For this project we will use an adjacency matrix.

The user should be able to specify a source and destination node.

Next you will need to modify the DFT method to count the number of paths that reach the destination node. Since two different paths may share some of the same nodes, we will want to maintain a separate list of available nodes (avail[ ]) for each path rather than sharing one global avail list as we did in the depth-first traversal.

Finally the modified DFT method will need to check to see if the destination node has been reached. If so, it should add 1 to the total path count and terminate, rather than continuing to search for available nodes.

The same adjacency matrix can be used by all copies of DFT, so it can remain as global.

When all copies of DFT have completed your program should display the total path count.