Lab 11 Report: Finding Cycles

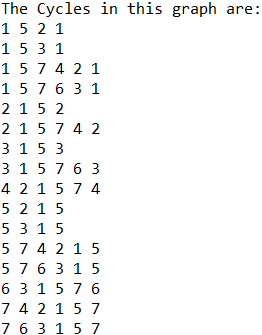
Problem

With the given graph below use DFS to detect all possible cycles. A cycle is a path (ordered set of vertices) that starts from an arbitrary vertex and ends at the same vertex. You may use either an ardency matrix or the linked form of a graph to achieve this.

Proposed Solution

* 1. Create a class that is an Adjacency Matrix Graph
     1. Instance Variables:
        1. adjMatrix: a double 2 dimensional array value that will store the weights of the matrix
        2. DEFAULT\_VERT\_AMT: final int that is size of the array, it will be square.
        3. Also implement a LinkedList such as by using arrayList, and a Queue of type Integer.
     2. Constructors: Default and Parameterized
     3. Methods:
        1. printDFS: takes in an int and goes through each value of the adjMatrix and finds each cycle, it does this by finding each edge from a vertex and going through all of them until it reaches itself again. Add and remove to the markedList accordingly.
        2. printDFS\_forAll: for loop that iterates through adjMatrix length, clearing and calling for each vertex in printDFS.

Tests and Results



Problems Encountered

The main problem when readapting the depth first search method was after figuring out that a vertex has multiple edges, you had to go to the smaller edge until you reached the original vertex first, but still have to keep track of the other edge when the vertex had multiple paths to take. This was done by adding and removing from a list.

Conclusions and Discussion

This lab was a great way to start an initial understanding of graphs and their functions, and how to keep track of traversing through one. Now it needs to be expanded on with the use of weights, distances, and finding the most efficient route with algorithms such as Dijkstra’s.

Additional Questions

1. Could BFS also be used to find cycles? Describe why or why not.

It is possible, but not as efficient and is harder to implement. Once DFS finds a cycle, it will have a stack that contains the nodes in the cycle, but not for BFS. BFS is based on indices and is harder to keep track of where the nodes are based on this.

1. How could your algorithm be modified to discover if there is a way back to the starting vertex?

DFS would have to be modified in order to discover if there is a way back to the starting vertex. When traversing from a vertex to its neighboring edges, you would have to keep going through the edges and their edges to find if they point back or equal the vertex, which requires a stack to store this traversal.