### Topological Sort ( Due 30 Nov 2020 )

In this assignment you will be creating a graph from an input gif file called [dag.gif](https://www.cs.utexas.edu/users/mitra/csFall2020/cs313/assgn/dag.gif). You will **complete** the [topo.txt](https://www.cs.utexas.edu/users/mitra/csFall2020/cs313/assgn/topo.txt) file.

The first line in that file will be a single integer *v*. This number will denote the number of vertices to follow. The next *v* lines will be the labels for the vertices in alphabetical order. There will be one label to a line. The labels are unique. The next line after the labels for vertices will be a single number *e*. This number will denote the number of edges to follow. There will be one edge per line. Each edge will be of the form - fromVertex and toVertex. Assign a default weight of 1 to each edge.

Here is the outline of the [code](https://www.cs.utexas.edu/users/mitra/csFall2020/cs313/assgn/Graph.py) that we developed in class that you will be modifying. You will use *topo.txt* instead of *graph.txt* as your input file. You can add an Edge class if you want to. You may use any of the functions that you wrote for graph traversal in your last assignment. You can add more functions as needed. You will first test if the given Graph does not contain a cycle and then do a topological sort on that Graph. For your output, the vertices on a given level must be printed in alphabetical order.

class Graph (object):

# determine if a directed graph has a cycle

# this function should return a boolean and not print the result

def has\_cycle (self):

# return a list of vertices after a topological sort

# this function should not print the list

def toposort (self):

def main():

# create a Graph object

theGraph = Graph()

# test if a directed graph has a cycle

if (theGraph.has\_cycle()):

print ("The Graph has a cycle.")

else:

print ("The Graph does not have a cycle.")

# test topological sort

if (not theGraph.has\_cycle()):

vertex\_list = theGraph.toposort()

print ("\nList of vertices after toposort")

print (vertex\_list)

main()

For the data file given, your output will look as follows:

The Graph does not have a cycle.

List of vertices after toposort

['m', 'n', 'p', 'o', 'q', 's', 'r', 'u', 'y', 't', 'v', 'w', 'x', 'z']

For this assignment you may work with a partner. Both of you must read the paper on [Pair Programming.](https://www.cs.utexas.edu/users/mitra/csFall2020/cs313/assgn/PairProg-CACM-1999.pdf).

The file that you will be uploading will be called **TopoSort.py**. We are looking for clean and structured design. The file will have a header of the following form:

# File: TopoSort.py

# Description:

# Student Name:

# Student UT EID:

# Partner Name:

# Partner UT EID:

# Course Name: CS 313E

# Unique Number:

# Date Created:

# Date Last Modified:

If you are working with a partner only **one** of you will be submitting your program but make sure that you have your partner's name and eid in your program. If you are working alone, then remove the two lines that has the partner's name and eid in the header.

Use the [Canvas](http://canvas.utexas.edu/)system to submit your **TopoSort.py** file. We should receive your work by 11 PM on Monday, 30 Nov 2020. There will be substantial penalties if you do not adhere to the guidelines. Remember Python is case sensitive. The name of your file must match exactly what we have specified.

* Your Python program should have the proper header.
* Your code must run on the command line before submission.
* You should be submitting your file through the web based *Canvas* program. We will not accept files e-mailed to us.