Computing the transpose of a directed graph (30 points)

A directed graph is a graph in which every edge has a direction, so it is a pair (u,v) of vertices, pointing from u to v, rather than a set {u,v}, which is what an edge in an undirected graph is. (Section 2.5 has more information.) Given a directed graph G = (V,E), we can define its tranpose, tr(G), as the graph in which the direction of every edge of G has been reversed.

For the following problems, let n = |V| (the number of vertices), and m = |E| (the number of edges).

A. What is the asymptotic running time of an algorithm calculating tr(G) from G in the adjacency matrix model? Express your answer in n and m as appropriate.

Answer: Asymptotic running time of an algorithm calculating tr(G) from G in the adjacency matrix model is O(V^2).

B. What is the asymptotic running time of an algorithm calculating tr(G) from G in the adjacency list model? Express your answer in n and m as appropriate.

Answer: the asymptotic running time of an algorithm calculating tr(G) from G in the adjacency list model is O(V+E).

C. Implement tr(G) for the adjacency list model and test-run it on some small sample graphs that I will provide.

Answer: Please look at the ArpanPatel_tr(G).py file for implementation.

Depth-first search (20 points)

Implement the basic Depth-First Search algorithm for traversing a graph given by an adjacency list. Print the vertices in the order that you encounter them. For this problem you can assume that the graph is connected. Hint: this is very close to the algorithm on page 175 but do not use any global variables.

Answer: Please look at the ArpanPatel_dfs_final.py

Topological sort (25 points)

Implement topological sort for sorting an input consisting of edges in an acyclic directed graph. I will provide test data. The output should be a list of the vertices in the graph in topological order.

Please look at the ArpanPatel_dfs_topologicalsort.py