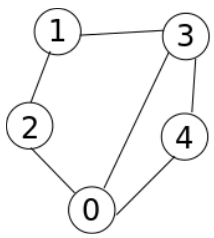
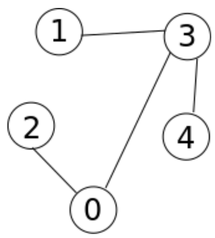
ECE 606, Fall 2021, Assignment 4
Due: Thursday, October 7, 11:59pm

Submission: submit three things: (i) your solutions for the written problems to crowdmark, (ii) the page with your name and student ID for the **[python3]** problem to crowdmark, and, (iii) your **[python3]** file to the appropriate Dropbox on Learn.

1. **[python3]** You are given as input (i) a connected undirected graph G, and, (ii) a spanning-tree T for it. (See Assignment 2 for what a spanning tree is.) Devise and implement a linear-time algorithm in subroutine `anotherst` that outputs another spanning tree $T' \neq T$ for G if one exists, and the Python 3 constant `None` otherwise.

   The vertices in $G = \langle V, E \rangle$ are always named $0, 1, 2, \ldots$, and both G and T are given to you as adjacency lists. For example, the following G to the left is represented as: [[2,3,4],[2,3],[0,1],[0,1,4],[0,3]] and the T to the right is represented as [[2,3],[3],[0],[0,1,4],[3]]. (Yes, we assume that each list is in sorted order.)

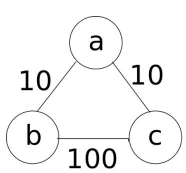
   

   You should return T' as an adjacency list similarly to how the argument T is encoded.

2. Alice makes the following claim. For every weighted undirected graph $G = \langle V, E, w \rangle$ with $w : E \to \mathbb{R}^+$, i.e., positive edge-weights, there exists a vertex $u \in V$ such that Breadth First Search (BFS) with u as source results in $\pi[\cdot]$ values that together comprise correct (weighted) shortest paths to all vertices from u. By BFS we of course mean that we ignore the weights on the edges and then run BFS from Lecture 4 of your textbook.

   For example, in the following graph, BFS with b as source incorrectly yields $\pi[c] = b$. However, BFS with a as the source yields $\pi[a] = \text{NIL}, \pi[b] = a, \pi[c] = a$, which indeed correspond to correct shortest paths from a.

   Prove via counterexample that Alice's claim is not true.

   Your counterexample needs to be valid no matter in what order neighbours are chosen in Line (12) of BFS. As a further clarification, your counterexample, which is a weighted graph $G = \langle V, E, w \rangle$, should be such that for every $u \in V$, there should exist $v \in V$ such that no unweighted shortest path from u to v is a shortest path when we incorporate the weights.

3. Recall that in an unweighted graph, we characterize the length of a path in terms of the number of edges in it. This is equivalent to adopting the weight function $w: E \to \{1\}$.

   In an unweighted undirected graph of $n \geq 2$ vertices, what is a tight upper-bound on the number of shortest paths that can exist from a vertex u to another vertex v? You are allowed to provide the solution in $\Theta(\cdot)$ if you feel it makes it easier for you. (But I am not sure it does.)

4. In a directed graph, a *sink* is a vertex which has no edges that leave it. Consider the following algorithm to topologically sort a directed acyclic graph (DAG).

   > Pick a sink, u
   > Append u to our topologically sorted list of vertices
   > Remove u and all edges incident on it from the graph
   > Repeat till we run out of vertices

   (a) Prove that every non-empty DAG has a sink.

   (b) Prove that the algorithm is correct.

   (c) Show, via presentation of pseudo-code and brief explanations, how the algorithm can be made to run in linear-time. Assume the input graph is encoded as an adjacency list, and that the vertices are named $1, 2, \ldots, |V|$.

5. Prove Corollary 24.3 in Lecture 4 of your textbook.

   > Let $G = \langle V, E \rangle$ be a weighted, directed graph with source vertex s and weight function $w: E \to \mathbb{R}$. Then for each vertex $v \in V$, there is a path from s to v if and only if BELLMAN-FORD terminates with $d[v] < \infty$ when it is run on G.