# HW1

Network Analytics 2020-21, Solutions

*Note: You can (but not necessarily) use example/graph in your answer. But they will not be counted for marks*.

1. 1 A directed graph is strongly connected if there is a (directed) path from every node to every other node. Show that in a directed strongly connected graph containing more than one node, no node can have a zero indegree or a zero outdegree.

   *If a node has a 0 indegree, there cannot be a directed path from any other node to that node. Likewise if it has a 0 outdegree, there cannot be a directed path from that node to any other node.*

2. Show that every tree is a bipartite graph.

   *Start from any node (say a leaf node or a node with degree 1 ---a tree always have to have a leaf node), and put all nodes with an even distance on one side, and odd odes on the other side of the bipartition. There is a unique path from any node to any node, so this is well defined .*

   *Note: Some people used the fact that a graph is bipartite iff it has no odd cycles, but this has to be proved first (we did only one direction in class) so that is no easier.*

3. A directed acyclic graph (DAG) is a directed graph without cycles (the underlying graph may have cycles, so it is not a tree). Show that a DAG has a labeling of its nodes (that is a labeling *1, …, n* of its nodes) such that every arc goes from a lower-numbered node to a higher-numbered node. (You need to find a way to do it, and also show that is always possible)

   *Label according to the order you find in a Depth-first-search. This leads to a topological ordering where starting from a node (label 1) we go to a node it points to and label each node as we go along. You cannot have a higher numbered node pointing to a lower numbered node as that would mean a directed cycle.*

   *Note: You need to find a way to do it, and also show that is always possible. Some of you missed one of them. Also you need to give details about how to do it.*

   *The TA selected this from one of the submissions:*

We use the following labeling of nodes:
Let $G$ be a directed acyclic graph **,** this implies that there must be some vertex in $G$ that does not have any incoming edge. Let $v_1$ be a vertex in $G$ that does not have any incoming edges. Now we remove the vertex $v_1$ from graph $G$ , label it as 1 and the newly obtained graph is $G_1 = G - \{v_1\}$. The graph $G_1$ is also acyclic (because removal of edges does not add any cycles), therefore there must be some vertex again that does not have any incoming edges. Let $v_2$ be a vertex in $G_1$ that does not have any incoming edges. Remove the vertex $v_2$ from graph $G_1$ and

label it as 2. Therefore, the new obtained graph is: $G_2 = G_1 - \{v_2\}$ or $G_2 = G - \{v_1, v_2\}$. We can repeat this process over and over until every vertex in $G$ is removed.
The resulting numbering of nodes is a topological order because an edge( $v_i$, $v_j$) must be deleted before the deletion of vertex $v_j$. Therefore the vertex $v_i$ must be deleted before the vertex $v_j$ and $i < j$ holds for every edge( $v_i$, $v_j$).

3. *Programming*

   *Many solutions possible and accepted. But you need to run your codes and show your answers. Also PLEASE SUBMIT ALL CODE AS Jupyter html files ONLY*

   *Fastest would be to use pandas.corr function (say Pearson). Equivalents in scipy.*

   *Some of you use log function to transform the raw data, but we need to directly compute the correlations from raw data. Also, if your correlation matrix is wrong, grades will be taken from your second and third questions since they depend on your correlation matrix.*

   *Convert to a distance or pseudo-distance metric like absolute value or other possibilities (however, negative correlations might be as useful to know as positive depending on the context; say as a proximity measure and you are looking at how close the assets are to each other, )*

   *Use some threshold value to filter out low values as they are not important. Can try Max Spanning Tree to give a more cleaner visual, but not necessary.*

   *Try all the basic plotting functions in networkX*