Brandon Phillips

Calculate Graph Density of Network01 and Network02 for simple directed and simple undirected Graphs

After performing the density calculations for the graphs, we want to know the ratio between the number of edges and the maximum edges. I found that the density for a simple directed graph of the Network01 to be equal to 0.5 and Network02 to be equal to 0.0000159. When calculating the simple undirected graphs, we know that the maximum possible number of edges is twice of directed graphs, so we can find that Network01 is equal to 0.25 and Network02 is equal to 0.000005797.

Depth-First Search of Network01 and Network02 for simple directed and simple undirected Graphs

After finding the density, depth-first search can be used to find the number of trees. After performing calculations, I found that Network01 directed had 2 trees, and Network01 undirected, had 1 tree. Then we find that Network02 had 1 directed tree and 1 undirected tree. This number of trees represents a DFS forest. This is simply the collection of all the DFS trees. It shows some connections to the graphs can't be made so it will essentially be its own "tree." Both of our undirected graphs are equal to one because our nodes can traverse the graph in either direction. However, our undirected graphs show that the Network01 has two trees. This is because it cannot calculate the full tree path due to the direction and lack of predecessors to the unreachable node.

Adjacency list or Adjacency Matrix Efficiency

In these calculations we can know an adjacency list has a space complexity of O(n+m) with n being the number of nodes and m correlating to the number of edges. And adjacency list however has a space complexity of O(n^2). So, if we take an undirect adjacency list, the space complexity becomes O(n+n). This is more efficient than an adjacency matrix space complexity. When searching for edges in an undirected graph, it will take O(1) time. So, the adjacency list will be the more efficient choice for representing graphs.

Dijkstra's Algorithm Variant

If we were to modify Dijkstra's algorithm to use source vertex s, it would not take advantage of this change because s and t would not exist in the same tree, therefore no path exists, and this variant of Dijkstra's would not function properly.