CSCI-4974/6971: Homework 1
<v1.0> updated February 1, 2026

Graph Connectivity

Due Date: Friday 13 February 2026, 11:59pm via Submitty

For this assignment, we're going to analyze the connectivity and properties of several graphs. For background material and reference, use:

- https://en.wikipedia.org/wiki/Strongly_connected_component
- https://sci-hub.ru/10.1007/3-540-45591-4_68
- https://en.wikipedia.org/wiki/Biconnected_component
- https://en.wikipedia.org/wiki/Power_law#Maximum_likelihood
- https://en.wikipedia.org/wiki/Gini_coefficient#Definition
- https://www.cs.cornell.edu/home/kleinber/networks-book/networks-book.pdf – Ch. 13 - discussing the web structure

We will be using Submitty to collect assignments. Upload a single *.py file that outputs responses for all of the below. **You can use any NetworkX and builtin Python functionality you wish, as well as any libraries we've used in class, but do not use any other external libraries**.

First, if you haven't done so already, download and install Python, NetworkX, and any other dependencies you might need. Work through the examples we've done in class to get a feel for working in the Python+NetworkX environment. Next, we'll be using a couple datasets for this analysis, so download the following:

- Simple Wiki: http://cs.rpi.edu/~slotag/classes/SP26m/hw/out.link-dynamic-simplewiki.data
  - Set of hyperlinks from the *Simple English Wikipedia*
  - Vertices: pages, Edges: hyperlinks
- p2p-Gnutella: http://cs.rpi.edu/~slotag/classes/SP26m/hw/p2p-Gnutella31.data
  - Crawler snapshot of the Gnutella social network

– Vertices: peers, Edges: connections between peers

1. **Forward-Backward Algorithm:** The forward-backward algorithm (as we'll be terming it) is a straightforward algorithm for identifying the strongly connected component containing some root vertex. The algorithm is relatively simple: starting from vertex v on directed graph G, perform two BFS/DFS traversals, following out edges and then in edges. Vertices discovered in both traversals are in the strong component containing the root. See the included paper of Fleischer et al. for a description of the DCSC algorithm (Fig. 1), as they call it. Note that we won't be doing any parallelization. Implement the algorithm as defined in the homework template.

2. **Bow-tie Structure:** For the first graph (Simple Wiki), let's first see how its structure compares to the "bow-tie" structure observed on the Internet. You should ignore the timestamp data on the edges. Compute the following quantities for each graph (see pg. 389 from EK) and output as is given in the template code file:

   (a) Number of vertices in the largest weak component.

   (b) Number of vertices in each of SCC, IN, and OUT.

   (c) Number of vertices in each of Tendrils and Tubes.

   You should use only your FWBW algorithm implementation to determine all of the above except for the vertices in the weak component. We'll use the following definitions for each structural aspect:

   - WCC – Vertices in the largest weak component.
   - SCC – Vertices in the largest strong component.
   - IN – Vertices that can reach SCC.
   - OUT – Vertices that can be reached from SCC.
   - Tubes – Vertices that can be reached from IN and can reach OUT, but are not in SCC.
   - Tendrils – Can reach or can be reached by vertices in the above sets, as well as other vertices that can reach or be reached by vertice in Tendrils. Basically, everything in WCC that is not in one of the above sets.

3. **Biconnectivity:** See the "block-cut tree" defined in the Wikipedia page for Biconnected Components. We will be doing something similar, but without the cut vertices. First determine the maximal biconnected components of the p2p-Gnutella input. Then, create a new graph where each vertex represents a biconnected component of the original input and an edge between these vertices represent the fact that they share a cut vertex. See the code template for more specific details.

4. **Measurements:** Let's look at calculating various graph measurements on both p2p-Gnutella and our coarsening biconnectivity graph.

(a) **Degree Skew:** Determine the skew of the degree distribution for each graph by

   i. Plotting them on a log-log scale and visually examing the result.

   ii. Estimating the power-law coefficient using the *maximum likelihood* method.

   iii. Computing the Gini coefficient.

(b) **Diameter:** Recall that calculating a graph's diameter is also deceptively difficult. Implement the approximation scheme we discussed in class, where you'll iterate $\log n$ times at a maximum distance before terminating.