Lab activity: analysis of transcriptional regulatory networks

1. Session 3: Initial Network analysis

In this third part of the lab activity, we will perform an analysis of the *E. coli* transcriptional regulatory network we generated using RegulonDB data. We will use this analysis to understand how this biological network is organized, what its main properties are, and how these map to the underlying biology.

2. Implementation

A key parameter that we will analyze is the degree distribution of the *E. coli* transcriptional regulatory network.

1. Implement a function that, given a network (directed graph), the type of degree to analyze (in-degree, out-degree or general), the type of node we want to analyze (TF, TG or all) and the number of *N* top (highest degree) nodes to be reported, will return a tuple containing: 1) a list with the degree count distribution (counts for every possible degree, from lowest to highest degree) and 2) the dictionary with node names (locus_tags in the *E. coli* network) as keys and their degree values as values, for the top *N* nodes.

```python
def degree_distribution(G : nx.DiGraph, degree_type : str, \
                        node_type : str, bestN : int) -> tuple:
    '''Compute the in-, out- or general degree distribution.

    Parameters
    ----------
    - param: G : Networkx graph
        Graph to analyze
    - param: degree_type : str
        Type of degree to compute ('in', 'out', 'general')
    - param: node_type : str
        Type of nodes on which we report degrees ('TG', 'TG', 'all')

    - return: list
        list with degree frequency distribution
    - return: dict
        dictionary with node names as keys and their degrees as values
        for the top N nodes in the degree distribution
    '''
    # ------- IMPLEMENT HERE THE BODY OF THE FUNCTION ------- #
    pass
    # ----------------- END OF FUNCTION --------------------- #
```

The inferred *E. coli* transcriptional regulatory network is a directed, disconnected graph. Some of the analyses we want to perform require that the graph be connected, so we will implement a function to obtain the largest connected component of the underlying graph.

2. Implement a function that, given a network (undirected graph), will return a new network corresponding to the largest connected component of the input graph.

```python
def largest_CC_graph(G : nx.Graph) -> nx.Graph:
    '''Generate the largest connected component graph.

    Parameters
    ----------
    - param: G : Networkx graph
        Graph to analyze
    - return: Networkx graph
        the graph corresponding to the largest connected component in G
    '''
    # ------- IMPLEMENT HERE THE BODY OF THE FUNCTION ------- #
    pass
    # ----------------- END OF FUNCTION --------------------- #
```

3. Using the previous functions:

    a. Compare the two E. Coli TRN networks (with and without operons). This comparison should include, at least, order, size degree distributions (in, out, general), and size of the largest component. Additionally, compute and report any supplementary topological metrics that you find useful for a structural comparison of the two networks.

    b. Further study the indegree distributions. Compute and plot the difference indegree distribution (counts in network with operons minus counts in networks without operons). Is there a discernible peak in the difference distribution?

    c. Determine which nodes exhibit the highest in and outgdegrees connectivity, then investigate the biological processes linked to those nodes.

    d. Compare the operon network with `intergenic_dist` 100 to other operon networks with varying `intergenic_dist`. How does the intergenic distance affect the resulting network?

3. Evaluation and submission

As explained in the introduction, the evaluation of the lab activities will consider in-class presentations, defense of your solutions, and engagement in discussions; code; and a lab exam.

The code part of this session needs to be delivered via the virtual campus the day before the next lab session (that is, at most on December, 2nd). The submission has to be a single zip file with:

- The `AGICI_lab_S3.py` file with your implementation of the degree distribution and largest connected component functions, and the graphs analysis and comparisons.