CSE 5656 - Theory and Applications of Complex Networks

Fall 2023

Assignment 01
Graph Theory Review 1

Professor :
Dr. Marco Carvalho
mcarvalho@fit.edu

Student :
Venkata Sai Ram Polisetti
ID : 904037836
vpolisetti20232my.fit.edu

Objective

Exercise the fundamental concepts of Graph Theory presented in class.

Task Description

Using the GitHub repository assigned, I wrote a program using python language that creates and analyses a graph in the following way:

Program Specification:

Arguments:

- Graph Order (as an integer, between 10 and 200)
- Graph Density (as a positive double)

Execution:

- Given the arguments above, the program creates a directory for its output with a unique name.
- For each run, the program generates 16 directed simple random graphs (see details below) and saved each graph as a separate text file (see format below), in the output directory.
- For each randomly generated graph, the program performs a set of analysis tasks including the following:
    - Calculated and reported the size and order of the generated graph.
    - Calculated and identified the top 5 highest and 5 lowest node degrees, considering all nodes in the graph.
    - Calculated and reported the in-degree and out-degree of the selected nodes.
    - Calculated the average degree of all nodes in the graph.
    - Discretized the range of degrees, in-degrees, and out-degrees in 10 segments and indicate the number of nodes with degree counts mapped to each range (that is, build the histograms using fixed 10-bins for degrees, in-degrees, and out-degrees).
    - Wrote a simple function that tests if the graph is cyclic or not and report the result.
    - Calculated the sum of in-degrees and out-degrees of all nodes in the graph.
    - Compared the metrics calculated for each graph across all 16 graphs and reported the average and variation (or standard deviation) of metrics for all 16 graphs.

Deliverables:

All deliverables submitted in the GitHub repository created for the assignment. In addition to the code, and results for the cases defined below :

a) How to execute the Program :

Make sure we have Python installed on our computer. We can download Python from the official website: https://www.python.org/downloads/

Execution Steps:

- **Code Setup:**

To execute this program, we need to use any programming language. I have done in python and run it in Visual Studio Code.
(e.g., Visual Studio Code, PyCharm, Notepad++, etc.).

- **Save the Code:**

  Save the file with a .py extension. For example, you can save it as random_graph_generator.py.

- **Navigate to the Code's Directory:**

  Use the cd command to navigate to the directory where you saved your Python script. For example: CN/ASSIGNMENT

  Ex : cd /path/to/your/script/directory

- **Execute the Program:**

  Run the program in the main.py file (This program is based on the Principle of separation of concerns).

- **Separation of Concern:**

  Separation of concerns is a principle used in programming to separate an application into units, with minimal overlapping between the functions of the individual units. The separation of concerns is achieved using modularization, encapsulation and arrangement in software layers.

- **Program Execution:**

  The program will execute, generating random graphs, analysing them, and saving the results in unique output directories. We should see some output on the terminal as the program progresses.

- **Output Files:**

  The program will create multiple output text files containing generated graphs, analysis results, and summary metrics. We can find these text files in the same directory where our Python script is located.

- **View Results:**

  We can open the generated text files within the output directories to view the details of the generated graphs, analysis results, and summary metrics.

- **Repeat or Customize:**

  If needed, we can modify the parameters such as graph_order and graph_density in the main function to customize the behaviour of the program and I have also attached the code file to customize or enter the inputs(graph_order, graph_density) in the output terminal in this program.

- **Exit the Program:**
  The program will exit once it completes its execution and generates the output files in the directory.

b) Brief summary of the approach used for each item

1. Generating Random Graphs (generating_random_graphs function):

Approach: This function generates random directed graphs with a specified order and density.

Steps:

- Create an empty graph represented as an adjacency list.
- Calculate the desired number of edges based on the density.
- Use a loop to add edges randomly while avoiding self-loops and parallel edges.
- Return the generated graph.

2. Creating Unique Output Directory (unique_output_directory function):

Approach: This function creates a unique output directory based on a timestamp.

Steps:

- Get the current timestamp in the format YYYYMMDDHHMMSS.
- Create an output directory with a name like output_YYYYMMDDHHMMSS.
- If the directory already exists, it won't overwrite it.

3. Saving Generated Graphs to File (save_generated_graphs_as_file function):

Approach: This function saves the generated graph to a text file.

Steps:

- Open a file in write mode.
- Write the header indicating it's a random generated graph.
- Iterate through the graph's nodes and their adjacency lists, writing them to the file.

4. Saving Analysis Results to File (save_analyzed_results_as_file function):

Approach: This function saves analysis results to a text file.

Steps:

- Open a file in append mode.
- Write the header indicating it's analysis results.
- Iterate through the analysis results dictionary and write key-value pairs to the file.

5. Saving Metrics to File (save_metrics_as_file function):

Approach: This function saves summary metrics to a text file.

Steps:

- Open a file in write mode.
- Iterate through the summary metrics dictionary.
- For each key-value pair, write it to the file, handling lists/tuples differently.

6. Checking Graph Cyclicality (is_cyclic_util and is_cyclic functions):

Approach: These functions determine whether the generated graph is cyclic or acyclic.

Steps:

- is_cyclic_util uses a depth-first search (DFS) approach with recursive calls to detect cycles in the graph.
- is_cyclic iterates through all nodes, calling is_cyclic_util for each unvisited node to check for cycles.

7. Main Function (main):

Approach: The main function orchestrates the entire program's execution.

Steps:

- Define graph order and density parameters.
- Loop to generate and analyze random graphs multiple times.
- For each iteration:
  a. Create a unique output directory.
  b. Generate a random graph.
  c. Perform various analyses on the graph.
  d. Save the graph, analysis results, and summary metrics to files.
  e. Exit the program.

8. Input Section (if __name__ == "__main__":):

Approach: This part of the code ensures that the program runs when executed directly (not imported as a module).

Steps:

- Calls the main function to initiate the program's execution.

c) Comments or observations noted from the results obtained for the following arguments:

a) Order = 20, Density = 0.4:

Graph Size:

- The order of the graph is 20, which means it has 20 nodes.
- The density of 0.4 suggests that each node, on average, has 0.4 * (20 - 1) = 7.6 outgoing edges, which is approximately 8 edges per node.

Cyclic or Acyclic:

- The program would likely generate mostly acyclic graphs for this order and density combination.

Node Degrees:

- Node degrees would vary across the graph, with most nodes having around 8 outgoing edges on average (due to the density).
-

Top 5 Highest and Lowest Node Degrees:

- The top 5 highest node degrees would likely be close to 8 (or slightly lower due to randomness).
- The top 5 lowest node degrees may include nodes with 0 or 1 outgoing edge.

In-Degrees and Out-Degrees:

- The in-degrees and out-degrees for selected nodes (e.g., top 5) would follow similar patterns.
- In-degrees and out-degrees would be close to 8 for nodes with high out-degrees.

Graph Cyclicality:

- The program might rarely detect cycles in the generated graphs due to the low density.

Summary Metrics:

- Summary metrics, such as averages and standard deviations, for node degrees and other measures, would provide insights into the distribution and variability of graph properties.

b) Order = 60, Density = 0.4:

Graph Size:

- The order of the graph is 60, meaning it has 60 nodes.
- The density of 0.4 suggests that each node, on average, has 0.4 * (60 - 1) = 23.6 outgoing edges, which is approximately 24 edges per node.

Cyclic or Acyclic:

- The program may generate both cyclic and acyclic graphs for this order and density combination.

Node Degrees:

- Node degrees would vary, with most nodes having around 24 outgoing edges on average (due to the density).

Top 5 Highest and Lowest Node Degrees:

- The top 5 highest node degrees would likely be close to 24 (or slightly lower due to randomness).
- The top 5 lowest node degrees may include nodes with 0 or 1 outgoing edge.

In-Degrees and Out-Degrees:

- The in-degrees and out-degrees for selected nodes (e.g., top 5) would follow similar patterns.
- In-degrees and out-degrees would be close to 24 for nodes with high out-degrees.

Graph Cyclicality:

- The program might detect cycles in some of the generated graphs due to the higher density.
-

Summary Metrics:

- Summary metrics would provide insights into the distribution and variability of node degrees and other graph properties.

Output File Format:

A line for each node, followed the by the comma separated list of adjacent nodes. The implication of direction is from the source (line, or node number) to each of the destination nodes listed in the line.

- The code in this file (**“graph_theory_assignment”)** can be executed by giving the desired inputs of graph order and graph density .
- The code in this file (**“graph_theory_assignment_20”)** can be directly executed by the following inputs of Order = 20, density = 0.4; of graph order and graph density .
- The code in this file (**“graph_theory_assignment_60”)** can be directly executed by the following inputs of Order = 60, density = 0.4; of graph order and graph density .

Output:

- The program gives an output as 16 text files, one for each generated graph, the format ofthe file is in the text file format(graphs and analysis represented in the text format. The program also gives output one single file with the metric results for each of the generated graph, and the comparison for all graphs.