Name: Krishna Somani **2021058**

Objective

The objective of this project was to extract citation relationships between academic papers stored in a Neo4j database, construct a directed graph using NetworkX, and compute similarity measures (SimRank) between nodes (papers) for efficient analysis. The task included fetching the data from Neo4j, constructing the graph, and performing similarity analysis on specified query nodes.

Task Breakdown

1. Neo4j Data Fetching:
   - A query was run against the Neo4j database to fetch citation relationships. Specifically, the query matched paper nodes connected by the CITES relationship, fetching the IDs of source and target papers.
   - Data was retrieved in the form of (source, target) pairs representing the citation relationship between papers.

2. Adding Train Data to Neo4j:
   - The project also involved adding a train file to the Neo4j database. This file, named train.json, contains metadata for papers, including information such as paper IDs, author details, references, and text data.
   - The data from train.json was imported into the Neo4j database, with each paper represented as a node and citations as relationships between nodes.
   - This process ensured that the Neo4j database was populated with the required data for the subsequent citation and similarity analysis.

3. Data Transformation:
   - The fetched citation data was converted into a list of edges.
   - A Spark DataFrame was created to handle the edges efficiently and allow distributed processing.

4. Graph Construction:
   - A directed graph (DiGraph) was created using NetworkX from the fetched edges.
   - The graph was constructed by iterating over the edges and adding them to the NetworkX graph object.

5. SimRank Calculation:
   - A SimRank implementation was used to calculate the similarity between query nodes. SimRank is a measure of the similarity between two nodes in a graph based on their neighbors.
   - The calculation was optimized by caching in-neighbors to avoid repeated computation and making the function more memory efficient.

6. Data Export:
   - The final constructed graph was saved as an edge list (graph.edgelist), which could be later used for further analysis or visualization.

Process Overview

1. Graph Construction:
   - The data was retrieved from Neo4j using a Cypher query, and the resulting edges were converted into a Spark DataFrame. This data was then used to construct a directed graph using NetworkX.

2. SimRank Function:
   - The SimRank function was implemented recursively. To avoid recomputation, in-neighbors were precomputed for all nodes in the graph and cached in a dictionary.

3. Adding Train Data to Neo4j:
   - The train data, which contained metadata about academic papers, was added to the Neo4j graph database. This data included citations and references, which were later used to construct the graph and perform similarity analysis.

Results & Output
- The constructed graph was saved as an edge list (graph.edgelist).
- The similarity between specified query nodes was computed, and the results were sorted to return the top k similar nodes based on the SimRank score.

Conclusion
The task was successfully completed by fetching data from a Neo4j graph database, transforming it into a Spark DataFrame, constructing a directed graph using NetworkX, and implementing SimRank for node similarity analysis. Additionally, the train.json file was successfully added to Neo4j, enhancing the dataset used for the analysis. The process was optimized for memory efficiency, and the results were exported in a format suitable for further analysis or visualization.