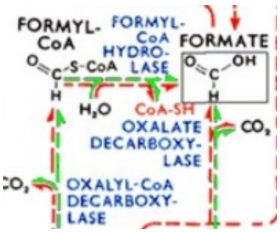
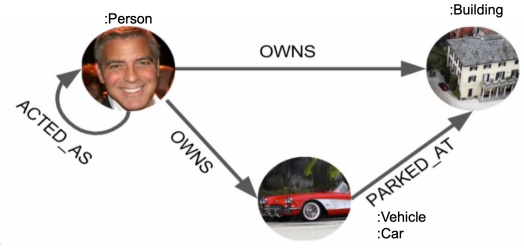
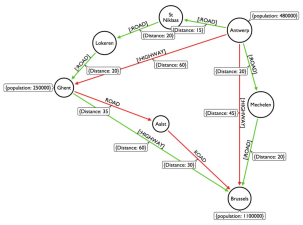
DS 3500: Advanced Programming with Data
Prof. Rachlin
Northeastern University

Property Graphs and Network Analysis







Introduction

In this assignment we will use object-oriented methods to implement a property graph model which will serve as the basis for a framework for doing network-centric data analysis. Many big-data problems are more naturally modeled as graphs, and graph databases are an important category of NoSQL (non-relational) databases, providing an alternative to traditional databases such as MySQL, SQL Server, and Oracle where all data is stored in flat two-dimensional tables. Applications of graph-based models include social networks, biological pathway modeling, routing problems, and supply-chain optimization.

Property Graphs are a kind of Graph

Property graphs like all graphs, are sets of vertices and edges. Graph databases like Neo4J instead use the terms Node and Relationship and we'll do the same in this assignment. Represent your graph using an adjacency list representation. Internally, the PropertyGraph is a dictionary where the key is a Node object and the value is a list of (Node, Relationship) tuples that define not only what nodes are adjacent, but the relationship that connects these two nodes. Note that there could be multiple relationships between the same two nodes.

Formally, nodes (vertices) have a name (string), a category (string), and an optional collection of key/value properties. Relationships (edges) are directed. Relationships have no name, but they do have a category (string), and their own optional collection of key/value properties.

Classes and Methods

A starting file, propertygraph_START.py has been provided for you. It contains specifications for three classes: Node, Relationship, and PropertyGraph.

1. Rename propertygraph_START.py to propertygraph.py

2. Implement the methods defined in propertygraph.py.

3. Test that your methods are correct using pytest. Write unit tests for all methods in all classes and place these unit tests in **test_propertygraph.py**. Save the output of pytest to a plain text file called **propertygraph_test_results.txt**. Be sure to include this file in your homework submission. Forgetting to include your test results will be a 20-point deduction.

Book Recommendations

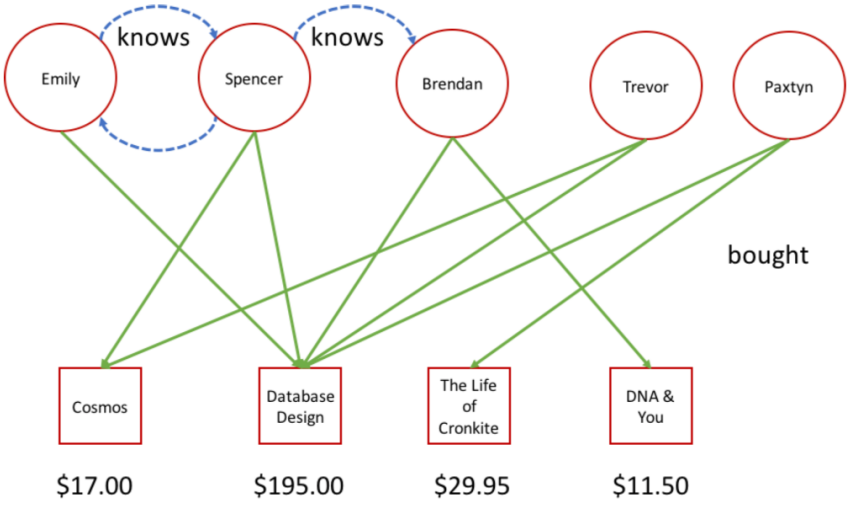
Companies use graph databases to build recommendation systems. Let's use your PropertyGraph object to recommend books.

1. Write a program called recommendation.py.

2. Build the property graph diagrammed below. This can be hard-coded. In this model, we have two categories of nodes: Person and Book. The books have an additional price property. There are two categories of relationships: Knows and Bought. Only book nodes have a property (the price).

3. Leverage your PropertyGraph methods to implement a simple recommendation engine. Recommend books for Spencer (or any Person) by finding all books bought by people that Spencer knows. Books already bought by Spencer should be discarded. Output the list of book recommendations as a new PropertyGraph with the Spencer Node and all recommended book nodes. The new PropertyGraph should be initialized as a subgraph containing Spencer and the recommended book nodes. Because Spencer did not buy any of these recommended books, the subgraph will consist of just two nodes and no relationships! Now add Recommend (category) relationships to your new PropertyGraph and print the result.

4. Print the original property graph and the property graph representing your recommendations for spencer. Include this output in the header of your program. Forgetting to include this output will be a 20-point deduction.



What to submit:

Submit each of the following files and ONLY these files. It is your responsibility to VERIFY that your submission is complete. Forgetting any portion of your submission will result in significant point deductions.

1. propertygraph.py: renamed from propertygraph_START.py with all methods implemented

2. test_propertygraph.py: Your unit tests, one for each method. It is up to you to define fixtures that generate a simple property graph, and that can be used to validate the various methods. Your unit test assertions should be comprehensive and cover all edge cases.

3. recommend.py: Your book recommendation engine demonstrating the utility of your PropertyGraph class. Remember to include the required output in the header of your program. The expected output is the original property graph, the subgraph containing Spencer and the books to be recommended, and finally the new property graph with Spencer linked to the recommended books.

4. propertgraph_test_results.txt: A human readable plain text file containing pytest output from the command: **python -m pytest -v --cov --cov-report term-missing**