IE 172 Laboratory 7: Getting Social With Networks

Dr. T.K. Ralphs

Due April 14, 2016

1 Laboratory Description and Procedures

1.1 Learning Objectives

You should be able to do the following after completing this laboratory.

1. Understand the concept and use of a graph.
2. Understand how to construct a graph and the two basic data structures for storing a graph.
3. Understand how to implement a basic graph search.
4. Understand how inheritance works in Python
5. Understand the importance of object-oriented design and code re-use.

1.2 Key Words

You should be able to define the following key words after completing this laboratory.

1. graph
2. vertex
3. edge
4. adjacency list
5. adjacency matrix
6. graph search
7. depth-first search

1.3 Scenario

Social networks have taken over. They're everywhere you look: Facebook, LinkedIn, Google+, Twitter, and more. Why is there so much fuss about these networks? Why is Facebook valued at close to $100 Billion? The answer is that there is a gold mine's worth of information in the data these social networks own. Underlying all of these social networks are graphs that tell us everything from what the real-world relationships between users are to what hobbies they share, what products they like, what Web sites they visit, where they went on vacation, and more. This

kind of information is worth millions to advertisers, who can use it to target individual ads to user according to their interests. All of this requires the ability to analyze truly massive graphs and has brought on the newly minted industry of "social network analysis." There are consultants specializing solely in how to analyze these graphs.

The "Facebook Graph" is the most comprehensive and well-known of the social graphs that exists today and its full description is a closely guarded secret. The graph includes nodes for everything from people to Web sites to companies and edges that represent the relationships of these entities in the real world. This graph is essentially what makes Facebook worth so much money. You can get access to parts of this graph if you have an account, develop an application, or are an advertiser. You can even explore parts of the graph by going to `http://graph.facebook.com`.

This lab will explore data structures for graphs in general and social networks in particular. A social network has a natural representation as a graph because the relationships are one-to-one. In Facebook, these relationships are defined by who you are "friends" with and also what things you have "liked." As an example of this kind of graph, we will construct a graph using data from the Internet Movie Database. Here, the relationships are between actors and actresses. Their connections are through having performed together in movies. Hence, this is the social network of Hollywood.

1.4 Design and Analysis

This lab will be focused primarily on implementation and use of a basic graph data structure. Performance analysis will be part of the series of laboratories that follow this one. One of the goals of this lab is to illustrate the efficiencies that can be gained through re-use of object-oriented code. Another concept we'll be illustrating here is that of *inheritance*. We'll be implementing a class for constructing and manipulating a *social graph*, which is a particular kind of graph, so the social graph class will *inherit* from the general graph class. In Python, this just means that all attributes of the graph class (called either the *parent*, *super*, or *base* class) are automatically also part of the social graph class (the *child*, *sub*, or *derived* class). In the social graph class, the methods will refer to attributes of a social network, such as whether two people are connected (know each other) rather than attributes of the underlying graph. Class inheritance allows us to specialize the API of a class. We will further specialize the social graph class into a class specifically for storing a network of actors and actresses.

Syntactically, inheritance is indicated in Python by putting the base class name in parentheses following the subclasses name in the definition:

```
class Graph:

    __init__(self):
    ...

class SocialNetwork(Graph):
    __init__(self):
        Graph.__init__(self)
    ...
```

Note that the constructor of the base class will be called automatically when an instance of a subclass is instantiated *unless the subclass has its own constructor*. In the latter case, it is necessary to explicitly call the constructor of the base class if necessary (as illustrated above).

1.5 Program Specifications

Your task is to use the graph class from the package called GiMPy (Graph Methods in Python) to do the exercises given. You will use GiMPy to create an application that can read in data from the Internet Movie Database, create the corresponding social network and answer basic queries, such as whether two actors/actresses are connected and what the components are. GiMPy can be installed with `easy_install coinor.gimpy`. The social graph is defined in the file `social_graph.py` and has methods related to constructing a social graph.

1.5.1 Program Function

The social network class has already been implemented to support the interface defined in `social_graph.py`, which is based on that of the graph class. Finally, the actor graph class is a specialized version of the social network class.

1.5.2 Algorithms

The basic algorithm exemplified in this lab is graph search, as described in Lecture 18.

1.5.3 Data Structures

The data structures used in this lab are dictionaries (for implementing the adjacency lists), a stack or a queue (for implementing the graph search), and an adjacency list representation of a graph.

2 Laboratory Assignments

2.1 Programming (30 points)

1. (10 points) GiMPy has a search method, but it is more general than what we need for the purposes of this lab and parts of it may be difficult to understand. Implement a simplified `search` method for the social network class that finds all people connected to a given person, as we discussed in class. Your method should be capable of printing the search tree as a list of edges (for this, you should store the precedence list).

2. (10 points) Add a `degrees_of_separation` method to the `SocialNetwork` class that finds the path between two actors/actresses that has the fewest number of links in the actor social network. We will show in class that such a path is obtained by finding the path connecting two nodes in the breadth-first search tree. If you do this with Kevin Bacon, you will find the Bacon Number of the given actor.

3. (10 points) Add a `popularity()` method to the `SocialNetwork` class for determining the "popularity" of a person based on the number of other people that person is connected to in the network.

2.2 Follow-up Questions (30 points)

1. (10 points) A "clique" in a graph $G = (N, E)$ is a set of nodes that are all mutually connected, i.e., $C \subseteq N$ such that $i, j \in C \Rightarrow \{i, j\} \in E$. A "clique" in a social network graph can be interpreted in the usual sense of the word when used in English—a group of people who are all mutual friends or associates. Describe a method for finding a "maximal" clique in a social

network, i.e., a clique that can't be enlarged by adding any single node. Note that solving the problem of finding a clique of *maximum* size is a difficult problem in general. You only have to propose a method for finding a clique of that is *maximal*.

2. (10 points) Explain how to modify your graph search method in order to determine whether or not a given undirected graph $G = (V, E)$ has a cycle. Prove formally that it is correct. Note that there are two directions to the proof. One direction should argue that if the algorithm detects a cycle, then there really must be one and we can produce it. The other direction should argue that if the graph *does* contain a cycle, then the algorithm will detect it. For the latter, you can use a contradiction argument.

3. Describe a graph with n vertices for which graph search takes $O(n^2)$ time using an adjacency matrix representation of the graph.