Week 11: Lab

Module 5: Graphs

COLLABORATION LEVEL 0 (NO RESTRICTIONS). OPEN NOTES.

1. Briefly describe (i.e. give high-level pseudocode) and analyze algorithms for the following problems. In each problem you are given an undirected graph $G = (V, E)$ as an adjacency list.

   (a) Is G connected?

   (b) Given two vertices, are they in the same CC?

   (c) Given two vertices u, v, return a path between them, or `null` if a path does not exist.

   (d) Given two vertices u, v, describe how you can find the shortest path between them.

   (e) How many CCs are in G?

   (f) Does G contain a cycle?

   (g) Is G a *tree* (a tree is an undirected graph that's connected and has no cycles)?

2. Briefly describe (i.e. give high-level pseudocode) and analyze algorithms for the following problems. In each problem you are given a **directed** graph $G = (V, E)$ as an adjacency list.

   (a) Find all vertices reachable from a given vertex u.

   (b) Given a vertex u, compute all vertices v such that u is reachable from v.

   (c) Given two vertices u, v, is there a (directed) path from u to v? If yes, how do you find such a path?

   (d) Does G have a cycle, or is it acyclic? (a graph is called acyclic if it does not have a cycle).

3. All-pair connectivity: Given an undirected graph, you want to be able to answer connectivity queries: given two arbitrary vertices u, v, are they connected?

   (a) Describe (high-level psedocode) and analyze an efficient algorithm to answer such a query.

   (b) Assume the graph is large and you have a lot of such queries to answer. In this case it is worth it to *pre-process* the graph so that you can speed up query time. Specifically, you will compute a V-by-V matrix TC, so that $\text{TC}[i][j] = 1$ if and only if vertices i and j are connected, and 0 otherwise (TC is the adjacency matrix of a graph that's called the *transitive closure* of G).

   Describe (i.e. high-level pseudocode) and analyze an algorithm to compute this matrix. Note that once TC is computed, we can use it to answer connectivity queries between arbirtary vertices in $O(1)$ time. Also note that TC uses $\Theta(V^2)$ space, so we are trading space and pre-processing time for improvement in query time.

   Note: The same strategy can be used to speed up reachability and shortest paths queries.

4. The *square* of a digraph $G = (V, E)$ is the graph $G^2 = (V, E')$ such that $(u, w) \in E'$ if and only if for some vertex $v \in V$, both $(u, v) \in E$ and $(v, w) \in E$. That is, G^2 contains an edge from u to w whenever G contains a path with exactly two edges from u to w.

   (a) Draw a small directed graph (4-5 vertitces) and then show its square.

   (b) Draw a small graph G such that G^2 has more edges than G.

   (c) Draw a small graph G such that G^2 has fewer edges than G.

   (d) Draw a graph G with $|E| = \Theta(|V|)$ egdes, such that G^2 has $|E'| = \Theta(|V|^2)$ edges.

   (e) Describe (pseudocode) an efficient algorithm for computing G^2 from G. Your algorithm should take as input the adjacency list of G and create an adjacency list for G^2. Note that some of the edges you generate will be duplicates, and you will need to do a pass and remove the duplicates.

   (f) Analyze your algorithm in terms of $|V|$ and $|E'|$.

Additional Problems

1. **The Kevin Bacon game:** One of the classic application of graphs is to find the degree of separation between two individuals in a social network. We'll discuss this in terms of the Kevin Bacon game. Most of you have heard about Kevin Bacon. He is a known actor who appeared in a lot of movies. We assign every Hollywood actor a Kevin Bacon number (BN) as follows: Bacon himself has BN=0; any actor (except Bacon himself) who has been in the same movie as Kevin Bacon has a BN= 1; every actor who does not have a BN of 0 or 1, and has been in a movie with an actor who has a BN of 1, gets a BN=2; and so forth.

   Example: Meryl Streep has BN=1, because she appeared in *The river Wild* with Kevin Bacon. Nicole Kidman's number is 2 because she did not appear in any movie with Kevin Bacon, she was in *Days of Thunder* with Tom Cruise, and Cruise appeared in *A few good men* with Kevin Bacon.

   Given an actor/actress name, the simplest version of the game is to find a sequence of movies alternating with actors connecting that actor to Kevin Bacon. For example: a movie buff might know that Tom Hanks was in *Joe versus the volcano* with Lloyd Bridges, who was in *High noon* with Grace Kelly, who was in *Dial M for murder* with Patrick Allen, who was in *The eagle has landed* with Donald Sutherland, who was in *Animal house* with Kevin Bacon. Based on this, Tom Hanks is at distance 5 from Kevin Bacon. But this is *not* Hanks' BN: Hanks has BN=1, because he was in *Apollo 13* with Kevin Bacon.

   Model this problem as a graph problem and describe algorithmically how you would solve the Kevin Bacon Game, by answering the questions below:

   (a) What are the vertices and edges?

   (b) Assume we get as input a file (e.g. `movies.txt`) from the Internet Movie Database. This file consists of lines, each line contains a movie title, followed by all actors who played in that movie. Think how you would go about representing and building the graph corresponding to this file. What would be the vertices and edges in this graph?

   Note that we are dealing with `String`s, either movie names or actor names, whereas in theory we think of the vertices of the graph as being sitting in an array indexed from 1 to n. For vertices that are strings we would use HashMaps to store the edges.

   (c) Given an actor, you want to find the sequence of movies /actors that connect him/her to Kevin Bacon. Describe how you would do this.