Lab 11
Graphs

Collaboration level 0 (no restrictions). Open notes.

1. You are given an **undirected** graph $G = (V, E)$ as an adjacency list. Describe and analyze algorithms for the following problems.

   (a) Determine if G is connected.

   (b) Given two arbitrary vertices, determine if they are in the same CC.

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

   (d) Given two vertices u, v, return the shortest path between them.

2. You are given a **directed** graph $G = (V, E)$ as an adjacency list. Describe and analyze algorithms for the following problems.

   (a) Given an arbitrary vertex u, find all vertices reachable from u.

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

   (c) Given two vertices u, v, determine if u reaches v, and find a path from u to v.

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 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 answer queries faster. Specifically, you will compute a V-by-V matrix TC, so that 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 and analyze an algorithm to compute this matrix. Note that once TC is computed, we can use it to answer connectivity queries between arbitrary 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.

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) Describe 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.

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. Many of you may have heard of Kevin Bacon, prolific actor who appeared in a lot of movies. We assign every actor a Kevin Bacon number (BN) as follows:

   - Kevin Bacon has BN=0
   - Any actor except Bacon who has been in a movie with Kevin Bacon has a BN= 1
   - Any actor (except Bacon and those with a BN of 1) who has been in a movie with an actor who has a BN of 1, has BN=2
   - and so forth.

   Example: Meryl Streep has BN=1, because she appeared in *The river Wild* with Kevin Bacon. Nicole Kidman's BN 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. Tom Hanks has BN=1, because he was in *Apollo 13* with Kevin Bacon.

   Given an actor/actress name, we want to find their BN and the sequence of movies/actors connecting them to Kevin Bacon. Model this problem as a graph problem and describe how you would solve it by answering the questions below:

   (a) What are the vertices and edges?

   (b) Assume we get as input a file (e.g. `movies.txt`) which 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 a Map 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.