Self-Practice Week 7 - Graphs (part 1)

Undirected and Directed Graphs

The goal of this assignment is to develop a better understanding of data structures and algorithms to represent and manipulate undirected and directed graphs.

Exercise 1 – Union-Find Algorithm for Cycle Detection

Consider the Union-Find algorithm seen at the very beginning of the course. Use it to determine whether an un-directed connected graph G contains cycles or not.
Hint: start with each vertex in the graph belonging to its own (disjoint) set, then process one edge `(u, v)` at a time, to check whether a cycle forms.

Exercise 2 – Is G bipartite?

Given an undirected graph G with no cycles, design and implement an algorithm to check whether G is bipartite
Hint: use BFS

Exercise 3 – Transitive Closure of a Graph

Given a digraph G, the transitive closure is a digraph G' such that `(u, v)` is an edge in G' if there is a directed path from `u` to `v` in G. The resulting digraph G' representation as an adjacency matrix is called connectivity matrix. Design and implement an algorithm to compute the connectivity matrix of any given digraph G.
Hint: use DFS

Exercise 4 – IMDB Movie co-stars

The Internet Movie Database (see file `3-imdbtest.txt`) lists several thousands of movie titles; for each title, it also lists the names of the actors who have performed in them (file format: one movie title per line, followed by a / separated list of actor names). Design and implement an efficient data structure to represent actors' co-starring in movies. Efficiently support the following API:

- Has actor `a` ever performed in a movie with actor `b`?
- In how many movies have `a` and `b` performed together?
- How many degrees of separation are there between actors `a` and `b`?

Hint: represent actor co-starring as a weighted undirected graph (using adjacency lists). To compute degrees of separation, run BFS from `a` and see if (after how many "layers") you can reach `b`.