Due Nov. 19, 11.55pm

This project will use IMDB's movie/actor datasets to demonstrate various algorithms on graphs. In part 1, you will use the movie dataset to perform traversals on parts of the graph, while in part 2, you will use Dijkstra's shortest path algorithm to compute the Bacon number, i.e. the shortest path through the movie/actor graph that connects two actors – the so called 'Six Degrees of Separation' problem. You will use BRIDGES to display the marked up graphs.

BRIDGES Relevant Calls

The BRIDGES API is accessible from

`http://bridgesuncc.github.io/doc/api/0.99.993/`

Refer to the HelloWorld example that illustrates how to create vertices(nodes) and edges in BRIDGES, as well as modify their attributes:

`http://bridgesuncc.github.io/main/HelloWorld-Tutorial_part1/`

To access the children of a graph node (needed in the traversal algorithms), refer to the Vertex class, in particular the getNeighbors() method, that returns a collection of the children of the vertex.

Dataset:

We will use a small version of IMDB's actor/movie dataset for this project. The dataset will be provided as a text file. The format of the dataset will be two items per line: actor and a movie he/she has acted in.

Tasks.

1. You will use the actor/movie dataset to build a graph, where each node is a movie or an actor and edges are constructed from an actor to a movie he/she has acted in, and movies connect to all actors in that movie. Algorithmically, reading in an actor/movie pair will result in edges in both directions. However, if an actor or movie node has already been constructed, those should be used, so that the set of nodes contain a set of unique actors and movies. You can set a private data structure to keep track of the movies/actors that have been read in(destroyed once the graph is built) and check or use a hash map for faster performance.
2. Implement the BFS and DFS traversal algorithms on graphs(see text for details). You will construct a modified versions of the two algorithms so that a user can specify the number of levels to recurse (we will likely restrict the recursion to 2 or 3 levels at most). Note that the BFS traversal requires a queue (you can use the book's version of the queue for the implementation, or the BRIDGES implementation).
3. Graph Display: Color code the edges and nodes of your traversal graph by either varying the shades of the nodes as the algorithm progresses by levels or use a series of different colors as the levels increase. Visually, this will make it clear how the traversal progresses from the start node. Use a unique color for the start node(given by user input).

Evaluation:

Set up your driver so that a user can either (1) enter an actor name and number of levels, or (2) list the actor names and let the user choose among them, followed by the BFS and DFS traversal of the graph and the generation of the two visualizations. This project will be evaluated by interactive demonstrations. Use the update() call on BRIDGES to generate the two visualizations simultaneously.

Submission Requirements.

Turn in your source code to Moodle; ensure it is well documented.