lolRajat Bhageria

Professor Sheth

NETS 150

22 January 2014

Homework 1

Part 1:



|  |  |  |
| --- | --- | --- |
| Node | In-Degree | Out-Degree |
| V | 1 | 1 |
| W | 2 | 1 |
| S | 2 | 1 |
| Q | 1 | 3 |
| T | 1 | 2 |
| X | 2 | 1 |
| Z | 1 | 1 |
| Y | 3 | 1 |
| U | 1 | 1 |
| R | 0 | 2 |

* 1. Q -> s -> w -> t -> v -> x -> y -> z -> r -> u
  2. Q -> s -> v -> w -> t -> x -> z -> y -> r -> u

1. Without loss of generality, say that there are more indie actors than mainstream actors (in other words, assume that there could be more mainstream actors than indie). We can represent this situation as a bipartite graph. Number each of the k actors from 1 to k. Designate the first actor as indie. Designate the second actor as mainstream. Now keep alternating designations until there are r indie actors and r mainstream actors. At this point, match consecutive actors with each other (so that actor 1 is matched with actor 2, actor 5 is matched with actor 6, and so on and so forth till actor r-1 is matched with actor r). Now, there will be exactly r pairs, and k-r indie actors remaining. Moreover, each of the r pairs will have one mainstream actor and one indie actor.

Programming Part

* 1. The distance between nodes 40 and 1050 is 143
  2. Yes. I created a second version of the BFS algorithm that doesn’t have a “target node.” Rather, the point of it is to access every connected node in the graph. This algorithm returns back the number of nodes that the algorithm processed. Additionally, in the AdjacencyMatrix class, I created a method called getNumberOfVertcies() that returns back the total number of vertices that the graph passed in (via edge list) has. To figure out the answer to this question, I see if the total number of vertices in the graph is equal to the number of nodes processed by the BFS algorithm. In this case, the answer is true.
  3. I’m getting different numbers. For example if I start at node 0, I get 3967; but if I start at node 21, I get 3956. This is expected because the more centralized the node is, the less steps it should take. And the less centralized the node is, the more steps it should take.
  4. 990. The way I computed this was the following: in my runBFS() method, I placed a Print statement that told me the size of the set (that tells me whether I have visited the node already) when my counter variable is equal to 4 (in other words, when the algorithm is searching a distance of four from the initial start node of 1344).

**Extra Credit:**

The Estimated Time Loading for matrix is: 10256000

The Estimated Time Loading for list is: 18524000

The Estimated Time for DFS for matrix is: 9354000

The Estimated Time for DFS for list is: 1647000

The Estimated Time for BFS for matrix is: 63119000

The Estimated Time for BFS for list is: 398000

This makes sense because the data passed in dense dataset. When there is a lot of nodes and a lot of edges, it is useful to use an Adjacency Matrix because it is O(V^2) not O(E).

However, it also makes sense that the BFS and DFS algorithms should take less time with the AdjacencyList than with the AdjacencyMatrix; this is because the algorithms can more easily find and access the data with an LinkedList than with a 2D array.