In Lab 3, you generated the a directed graph of the NetPath EGFR signaling pathway and colored the nodes by the distance to the EGF ligand. A solution to Lab3 will be posted shortly to Moodle. You may use this solution if you cite it properly in your code.

1 Preliminaries

1. Open a new `HW3.py` file.
2. Download the `EGFR1-reachable.txt` file from Lab3. Anna may provide a second file with a more complete network over the weekend.
3. Write functions to read the edge file, and compute the shortest paths from EGF to each node. You may want to copy parts of `Lab3.py`, including posting the shortest-paths graph with a coloring scheme of your choosing.
4. For this homework, you may use either the original network or the graph with expanded complexes, as described in Lab3 and lecture.
5. Familiarize yourself with the `random` and the `math` modules (for computing the log of values).

2 Color the Nodes using a Random Walk

Implement a Random Walk with Restarts. Write a function `RWR()` to simulate a random walk with restarts. In this implementation, you will have a "walker" move about the network. Implement the "personalized PageRank" version, where nodes teleport *back* to the source node. You are given the following inputs:

- A *directed* graph $G = (V, E)$
- A source node $s \in V$
- A fixed probability q
- A number of time steps t

Pseudocode for the random walk with restarts is below. Implement the function with $s =$`EGF`, $t = 1,000,000$ and $q = 0.9$. In other words, 90% of the time move to a neighbor, and 10% of the time teleport.

Algorithm 1 RWR(G,s,q,t)

Initialize a *counts* dictionary
current $= s$
for each time step t **do**
  $counts[current] = counts[current] + 1$
  With probability q, set *current* to a neighbor node uniformly at random
  Otherwise, *current* $= s$
end for
Log-transform counts. For example, $counts[n] = math.log(counts[n])$
return *counts*

Note: *If a node has no outgoing neighbors then the walker must teleport to s.*
Note: *Take care when taking the log of 0*

Normalize the counts. Next, write a function to *normalize* the log counts. This function should have the same color scheme as the shortest paths graph: (it should have smaller values near EGF). It should be normalized so the smallest value is 0 and the largest value is 1. First, we calculate the minimum value min and the maximum value max of the log counts. For a node v with log count c_v, the normalized value is

$$1 - \frac{c_v - \min}{\max - \min}.$$

Color the nodes and post this graph to GraphSpace. You can vary the time step t and probability q and observe how the network changes.

3 Compare Shortest Paths and Random Walks

Now, we wish to compare the shortest path values and the random walk values. To do this, post a new graph with the absolute value of the *difference* between the normalized shortest path distance and the normalized random walk counts with $t = 1,000,000$ and $q = 0.9$.

- The values will be correlated, but they are not identical.
- You may change the color scheme here to highlight the differences.
- Try normalizing the differences; this will highlight the nodes that are the "most different".

In the Comments: *Write a few sentences about how changing t and q affects the comparison of Shortest Paths and Random Walks.*

4 Share Graphs and Submit Code

Share the following three graphs with the `HW3` group:

1. The shortest paths graph (from Lab3)
2. The random walk with restarts graph ($t = 1,000,000$ and $q = 0.9$)
3. The comparison graph

Submit your code to Moodle. Take care to comment the program, and be sure to note which networks you have shared with the HW3 group in the comments.

5 Extra Exercise

Write an additional function that calculates the probability distribution that the walker is at each node $v \in V$ at time step t. Given G,$s \in V$, q, and t, The probability $p(v)^t$ of a node $v \in V$ at time t is

$$p(v)^t = (1 - q) \times p(v)^0 + q \times \sum_{u:(u,v)\in E} \frac{p(u)^{t-1}}{d_{out}(u)}$$

where $p(v)^0$ is the initial probability distribution and $d_{out}(u)$ is the out-degree of node u. Post this graph to GraphSpace with a reasonably-sized t and $q = 0.90$.