PAGE RANK WRITE UP

The Class Design and Methods Inside:

I have an outer class called the Directed graph. Instantiate Class Directed Graph to create a Directed Graph object. This has an Inner Static class Edge which is used to define an edge between two vertices. Each edge also has a cost associated with it. This can be expanded to accommodate different types of cost while implementing this on a civilization level for the Human Dx Project. The reason behind creating an inner static class is because an instance of Inner Class-Edge can exist only within an instance of Outer Class-Directed Graph.
The Node Class creates a node with attributes such as Indegree, outdegree and a list of neighboring vertices. Each of these attributes can be retrieved in constant time. A node or a collection of nodes constitute a graph.

Methods Inside Class and associated time and space complexities:

- ***addVertex (vertex or list of vertices):***
  - Adds a vertex or a list of vertices to the graph. Nothing happens if vertex exists.
  - Vertex is added into a hash map. The vertex is they key and the value is the list of directed edges connected from the vertex.

- ***addEdge (from vertex, to vertex):***
  - This method Adds an edge to the graph; if either vertex does not exist, it's added.
  - This implementation allows creation of multiple edges and self-loops.
  - Consequently, the Indegree, Outdegree and the list neighboring vertices of a vertex is incremented.

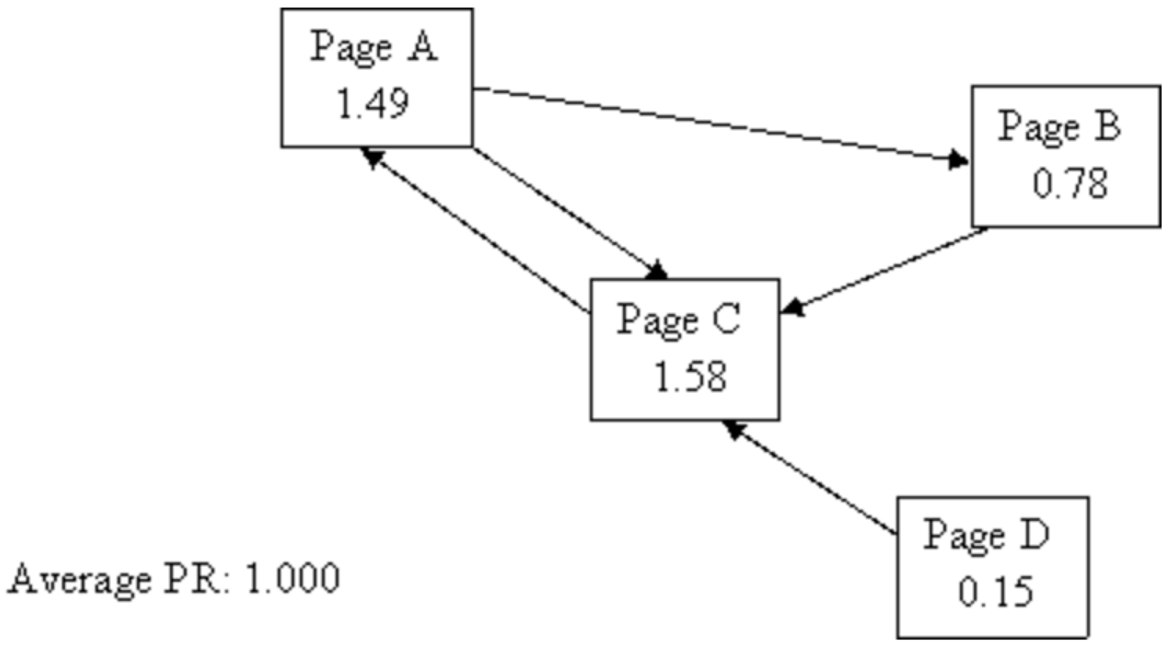
- ***containsVertex:***
  - This method returns true if a vertex is found in the graph.

- ***instantiateCostEdge:***
  - Method instantiateCostEdge Initializes cost of each out going directed edge of the vertex to 1/outdegree of the edge.

- ***getCost:***
    - Method returns the cost of an Edge.
    - Time complexity is the total number of edges for that vertex which is O(m).

- ***calculatePageRank:***
    - Method calculatePageRank calculates PageRank of each vertex.
    - We assume page A has pages T1...Tn which point to it.
    - The parameter d is a damping factor which can be set between 0 and 1.
    - I have set DAMPING_FACTOR to 0.85 which is accepted globally.
    - PR(A) = (1-d) + d (PR(T1)/C(T1) + ... + PR(Tn)/C(Tn))
    - This equation is iterated and run in a loop until the equilibrium is reached.
    - This also prints the PageRank value of the corresponding vertices or nodes from the updated pageMap Treemap.

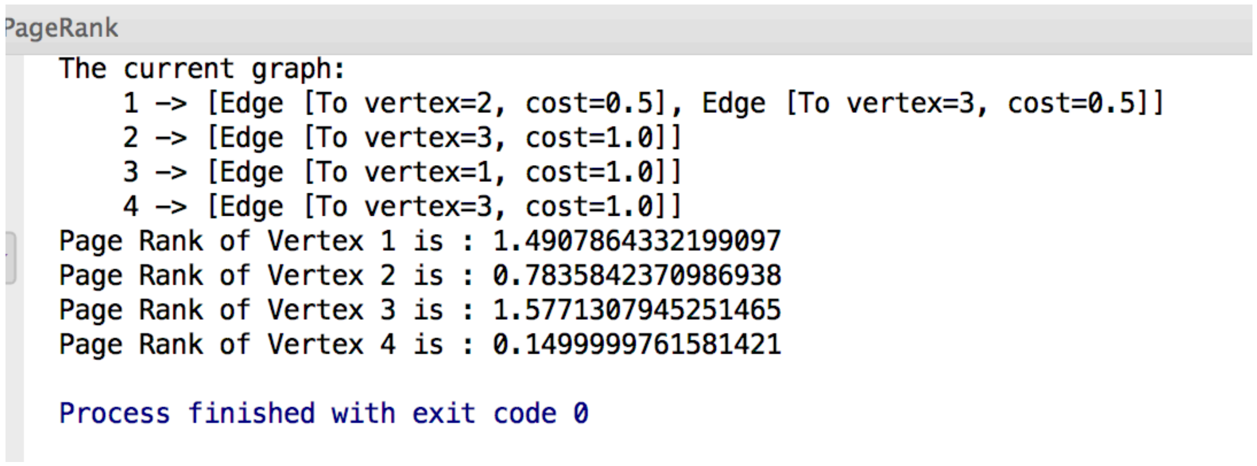
Example Snapshot of the Graph run in the program:



The figure:
 (source: http://www.cs.princeton.edu/~chazelle/courses/BIB/pagerank.htm) above shows a general graph with calculated Page rank values. With Average PR set to

1.0 initially. Page D has no backlinks; hence the equation looks like this: PR(D) = (1-d) + d*0.

Below is a screen shot of the calculated PR


PageRank

```
The current graph:
    1 -> [Edge [To vertex=2, cost=0.5], Edge [To vertex=3, cost=0.5]]
    2 -> [Edge [To vertex=3, cost=1.0]]
    3 -> [Edge [To vertex=1, cost=1.0]]
    4 -> [Edge [To vertex=3, cost=1.0]]
Page Rank of Vertex 1 is : 1.4907864332199097
Page Rank of Vertex 2 is : 0.7835842370986938
Page Rank of Vertex 3 is : 1.5771307945251465
Page Rank of Vertex 4 is : 0.1499999761581421

Process finished with exit code 0
```

The purpose behind the Human Dx project is, we are trying to build a bridge to connect different entities. This Project is building a common frame of reference for all stakeholders in the health system and It gives patients, family members, doctors, hospitals, and others a shared path to helping any person. There are different entities such as patients, family members, doctors, hospitals etc. All these entities form the nodes or the objects of the graph API. These objects will have ID's, different set of attributes and edges representing a relationship between them. There are different classes, sub classes and hierarchies to establish relationships.

For example, a doctor 'can belong' to a hospital or multiple hospitals. A Doctor has various attributes such as his/her specialty, a unique ID, number of patients he has treated, number of contributions towards a specific research etc. Based on this, we can come up with an importance/impact score for a given doctor when the need arises for a patient to get his symptoms checked or to receive a treatment. This is where the concept of PageRank comes into play. PageRank score may also determine the best possible course of action for a patient. It is based on many factors such as availability of the doctor, distance from the clinic, relationship to a family, etc. The in degree to a doctor can represent many relationships, such as 'he has cured' a patient. By weighing these different factors, we can come up with a customized PageRank calculation method.

An example for using machine learning and datamining is, finding similarity between each patient in a family or similarity between different families and learn from the illness history, the treatment they received. Hence, recommending patient x similar to patient y or family z towards the best possible treatment. Another example would be recording a patient's daily habit and provide important health tips by predicting what could go wrong if the patient continued with the same lifestyle.

The space required is HUGE and this needs a massive, fast-accessible distributed system. The concept of mapping entities and figuring out the right steps for a diagnosis or a treatment needs fast computation. For Example, a computation might be calculating the importance score of a doctor for a particular treatment. One of the many limitations in my implementation is that, it is not designed for distributed computing, such as the map reduce. Although I have the right approach of using hash maps and adjacency lists to store the graph. I can add more features to the graph API and extend it to undirected and Hyper-Graphs. The definition of cost of edge can vary and be handled differently. A future iteration would be implementing the map reduce framework on this.