INF 553 – Homework #5

HITS and Graph Mining

Due: May/01/2019, 11:59pm, to Blackboard

1. [70 points, **HITS**] Implement the HITS algorithm in Python Spark. Name your code "hits.py". You can take the Spark code for computing pagerank in parallel as the starting point.

   Your program should compute authority and hub scores of nodes via mutual recursion: start with the **initial hub scores** of nodes, all being **1's**, compute authority scores using the initial hub scores; then use the authority scores to compute new hub scores; and so on. Typically, the process can be terminated when no significant changes on authority and hub scores can be observed from a new iteration. But in this assignment, your script will take the desired number of iterations as the input, to simplify the problem.

   Your program should perform normalization of scores, such that the largest component is 1 (see the lecture).

   The program takes three inputs (in the following order):

   - Graph: This is a text file with tab-delimited lines. Each line represents an edge going from the first node to the second node. For example, the line "1 2" indicates that there is a directed edge from node 1 to node 2. You may assume that all nodes are represented in integers, starting from 1.
     For example, "graph.txt" is an example graph file provided to you for testing.
   - No of nodes: This is an integer representing the number of nodes in the graph, e.g., 5.
   - No of iterations: This is also an integer for the number of iterations the algorithm will perform the re-computation. Note that the computation of new authority and new hub is counted as one iteration. Note that the first set of authority scores are computed from the initial hub scores (all 1's as indicated above).

   For example, here is how your script will be invoked:

   ```
   bin/spark-submit hits.py graph.txt 5 2
   ```

   Output format: Here are the expected output, where "1 0.50" indicates that node 1 has the authority score of .50 at the end of the first iteration.

```
Iteration: 1
        Authorities:
                1 0.50
                2 1.00
                3 1.00
                4 1.00
                5 0.50
        Hubs:
                1 1.00
                2 0.50
                3 0.17
                4 0.67
Iteration: 2
        Authorities:
                1 0.30
                2 1.00
                3 1.00
                4 0.90
                5 0.10
        Hubs:
                1 1.00
                2 0.41
                3 0.03
                4 0.69
```

2. [Graph mining, 30 points]
    a. [15 points] Implement a graph partitioning program in Python, partition.py, which uses the edge betweenness measure to remove the edges from a given graph. You may use the "networkx" library for this question. The program stops the partitioning once the graph has been decomposed into a specified number of components/clusters. Note: after each removal of edge, the betweenness scores of remaining graph(s) need to be recomputed.

       Execution format: python partition.py <graph-data> k

       Where <graph-data> is a text file describing the graph, in the same format as the "graph.txt" in Problem 1, k is the number of clusters.

       Output the clusters line by line, with one cluster per line. Each cluster is an ordered list of ids of nodes.

    b. [15 points] Implement a graph partitioning program in Python, fiedler.py, which uses the Fiedler's vector to partition the given graph. You may use the "linalg" library for this question. Input and output formats are the same as part a.