**COS 598: Introduction to Data Science**

**Fall 2021**

**Homework Assignment 3**

Assigned: November 4, 2021

Due: November 18, 2021

*Submission instructions*:

You must submit your Python code in a single .py file, and include a brief write-up about what you did in your code to complete the tasks in this assignment.

The file(s) must be uploaded to the Brightspace submission site for Homework 3.

Total: 100 points (+50 bonus points).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Task 1. Page rank (90 points).

Page rank is a graph algorithm that is used by Google Search to rank websites in their search engine results. This algorithm iteratively updates a rank for each vertex by adding up contributions from all vertices that link to it. The algorithm can be summarized in the following steps:

1. Initialize the page rank of every vertex as 1.
2. During each iteration, let vertex *v* contribute rank(*v*)/|neighbors(*v*)| to its neighboring vertices, where rank(*v*) is the current page rank of vertex *v* and |neighbors(*v*)| denotes the number of vertices to which vertex *v* links (i.e., the number of outgoing edges of vertex *v*).
3. Set each vertex’s rank to 0.15 + 0.85 (contributions from all vertices that have edges pointing to the current vertex), where the contributions are computed in step 2.
4. Repeat steps 2 and 3 *k* times (you may set *k* = 10).
5. In the end, divide the page rank of every vertex by the total number of vertices we have in the input graph.

The page rank of a vertex measures its importance in the input graph. In this task, you will use Spark’s RDD API to implement the page rank algorithm. You may assume that the input graph to your program is given as a text file, where each line of the text file represents a directed edge from a vertex to another vertex. For example, an input text file may look like:

0 2

2 0

1 2

1 3

3 2

The above input file tells us that there are four vertices in the graph 0, 1, 2, and 3, and there is a directed edge from vertex 0 to 2, one from 2 to 0, one from 1 to 2, one from 1 to 3, and one from 3 to 2. You may assume that every vertex has at least one outgoing edge in your implementation.

Test your implementation on the input text file whose content is shown above.

Your implementation should output a text file, where each line contains a vertex and its page rank. With the test input text file shown above, your output text file should look like:

0 <page rank of 0>

1 <page rank of 1>

2 <page rank of 2>

3 <page rank of 3>

*Bonus* (+30 points): What if a vertex *v* has no outgoing edges? Such a vertex is known as a “dangling vertex.” In this case, one may create a dummy vertex *d* whose initial page rank is set to 0 (in step 1), and for every dangling vertex *v* with no outgoing edges, create a directed edge from *v* to *d* and let *v* contribute rank(*v*) to the dummy vertex (in step 2). Then, in step 3, sum up all the contributions to the dummy vertex *d* (say, the sum is equal to ), and set each vertex’s rank to 0.15 + 0.85 (/*N* + contributions from all real vertices that have edges pointing to the current vertex), where the contribution from a real (non-dummy) vertex is the same as before (i.e., the real vertex’s rank divided by the number of its outgoing edges), and *N* is the total number of vertices in the input graph. You may understand the term /*N* as the contribution to a vertex’s rank from the dummy vertex. Steps 4 and 5 are unchanged. Implement the modified version of page rank that can handle a graph with dangling vertices using the Spark’s RDD API.

Task 2. Spark SQL (10 points).

In Task 1, you have already computed the page ranks of the vertices 0, 1, 2, and 3 from the input graph defined by the small input text file used for testing your code.

1. Create a Spark data frame from the RDD containing the page ranks of the vertices 0, 1, 2, and 3 computed in Task 1.
2. Write a Spark SQL query to find the page rank of vertex 2. Print it to the screen.
3. Write a Spark SQL query to find the vertex with the largest page rank. Print both the vertex ID and its page rank.
4. Suppose that there is another input text file, where each line contains the meaning of a vertex. For example, suppose that we have another input text file which looks like:

0 Adam

1 Lisa

2 Bert

3 Ralph

where each vertex corresponds to a person. (You may think of the example graph in Task 1 as the “follows” graph in a social network, e.g., the edge from 0 to 2 means that Adam follows Bert – in this case, the page rank of a vertex measures the social influence of a person.) Create a Spark data frame from the input text file shown above.

1. Write a Spark SQL query to join the data frame containing the page rank information and the data frame containing the meaning of each vertex (in this case, the name of the person each vertex corresponds to). Save the query result in CSV format.

*Bonus Task 3. Real dataset* (+20 points).

Download the English Wikipedia hyperlink network data from:

<https://snap.stanford.edu/data/enwiki-2013.html>

This is a network of hyperlinks from a snapshot of English Wikipedia in 2013. An edge from *i* to *j* indicates a hyperlink on page *i* to page *j*.

The file enwiki-2013.txt.gz is a compressed text file, where each line corresponds to an edge (a directed link) from a Wikipedia page to another Wikipedia page.

The file enwiki-2013-names.csv.gz is a compressed CSV file, where each line contains a page index and the corresponding name of the Wikipedia page.

Run the page rank algorithm (which you implemented in Task 1) on this dataset. Find the names of the top 10 Wikipedia pages with the largest page ranks.

You may need to set up a Google Colab to complete this task.