**Part 1)** Hadoop Streaming

**Command Line:**

time hadoop jar hadoop-streaming-2.6.4.jar -input /data/lineorder.tbl,/data/dwdate.tbl -output /data/output1 -mapper HW4\_mapper.py -reducer HW4\_reducer.py -file HW4\_mapper.py -file HW4\_reducer.py

**Time: Approximately 24 seconds.**

A close up of a computer screen

Description automatically generated

**Mapper: Note - further comment on Assignment4-CSC555.py file.**

A screenshot of a computer code

Description automatically generated

**Reducer: Note - further comment on Assignment4-CSC555.py fiile.**

A screenshot of a computer

Description automatically generated

**Output:** Total number of rows: 49,698 exclude column name.



**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Part 2)** Multi-node cluster.

**Pre-requisite knowledge:**

* SHA-256 (Secure Hash Algorithm 256) is a cryptographic algorithm that produces a fixed-length, 256-bit (32-byte) hash value.
* Hexadecimal numbers are represented by only 16 symbols. These symbols or values are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F. Each digit represents a decimal value
* False positive: When Bloom filter indicates the presence of the value while it's actually not in the set.

**Coding logic:**

* **Note:** My depaulid **- ‘mngo’ -** hashlib.sha256(b"mngo 1").hexdigest()
* **“mngo X”** is the hash template where X is a number string.
* Two hash functions: 1st function - first hex digit, i.e., ‘0’ index | 2nd function – second hex digit, i.e., ‘1’ index. Given this ‘hashFunction = 2’, and ‘for i in range(self.hashFunctions)’will yield ‘0’ and ‘1’.

**Class BloomFilter code**

**A screenshot of a computer code

Description automatically generated**

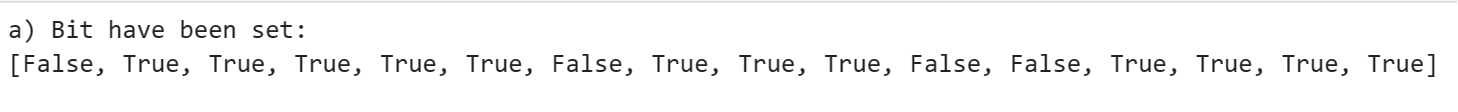
A screenshot of a computer program

Description automatically generated

**Adding and Checking Element:**

A screenshot of a computer program

Description automatically generated  
**a) Add the strings ’1’, ’3’, ’5’, . . ., ’19’ to this Bloom filter and write down which bits have  
been set.**

****

Bits 1, 2, 3, 4, 5, 7, 8, 9, 12, 13, 14, 15 have been set.

**b) Test the resulting Bloom filter for the values ’2’, ’4’, ’6’, . . ., ’20’. Do you get any false  
positives, and if so, what are they?**

A white text on a white background

Description automatically generated

My false positive are values ‘2’, ‘10’, ‘12’, ‘14’.

**Part 3)** Compute Page Rank.

**A diagram of a diagram

Description automatically generated**

**Note: I did both manually and in Python (manual matrix calculation and NetworkX library), please view Assignment4-CSC555.py. My manual calculation output matched with 6 iterations of Python calculation.**

Given network graph:

**Iteration 0:**

Follow class example, initial importance value for each node is 1/ total number of nodes = 1/4

**Iteration 1 - Page Rank Value – Used initial importance values:**

* +
* =
* =
* = +

**Iteration 2 - Page Rank Value – Used output from Iteration 1:**

* +
* =
* =
* = +

**Iteration 3 - Page Rank Value – Used output from Iteration 2:**

* +
* =
* =
* = +

**Iteration 4 - Page Rank Value – Used output from Iteration 3:**

* +
* =
* =
* = +

**Iteration 5 - Page Rank Value – Used output from Iteration 4:**

* +
* =
* =
* = +

**Iteration 6 - Page Rank Value – Used output from Iteration 5:**

* + = 0.40625
* = = 0.109375
* = = 0.1875
* = + = 0.296875

**Note: Rank order of 1 for highest Page Rank value.**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Node | Iteration 0 | Iteration 1 | Iteration 2 | Iteration 3 | Iteration 4 | Iteration 5 | Iteration 6 | Rank  Order |
| A |  |  |  |  |  |  |  | **1** |
| B |  |  |  |  |  |  |  | **4** |
| X |  |  |  |  |  |  |  | **3** |
| Y |  |  |  |  |  |  |  | **2** |

**Python Matrix Multiplication:**

**A screenshot of a computer code

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**Part 4)** Dead-end Nodes Page Rank.

**A diagram of a diagram

Description automatically generated**

**There are 2 ways solving this problem as discussed in class. I’m doing both.**

**Approach 1: Teleporting randomly – Random Walk.**

**Note: I did both manually and in Python (manual matrix calculation and NetworkX library), please view Assignment4-CSC555.py. My manual calculation output matched with 6 iterations of Python calculation.**

**What is the page rank of Q?** Since Q and A both have the same importance incoming edges, their importance values are always the same. Hence, it could be either 3rd or 4th rank. However, I am coding node A comes before Q, and ranked by indexing order. Thus, it Q would rank 4th in this case.

**What is the page rank of Z?** Since Z and Y have importance incoming edges from Q and A, respectively. Given Q and A importance values are always the same; thus, Z and Y importance values are always the same. With this, it could be either 1st or 2nd rank. However, I am coding node Y comes before Z, and ranked by indexing order. Thus, it Z would rank 2nd in this case.

**Note: Since this is a ‘dead end’ network, node Z will get importance values of 1/5 for each ‘potential connection’ random teleporting. This is to satisfy stochastic matrix column sum of 1, original it’s 0.**

Given network graph:



**Iteration 0:**

Follow class example, initial importance value for each node is 1/ total number of nodes = 1/5

**Iteration 1 – Used initial importance values:**

* =
* =
* =
* =
* =

**Iteration 2 – Used output from Iteration 1:**

* =
* =
* =
* =
* =

**Iteration 3 – Used output from Iteration 2:**

* =
* =
* =
* =

**Iteration 4 – Used output from Iteration 3:**

* =
* =
* =
* =
* =

**Iteration 5 – Used output from Iteration 4:**

* =
* =
* =
* =
* =

**Iteration 6 – Used output from Iteration 5:**

* = 4
* =
* =
* =
* =0.2415

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Node | Iteration 0 | Iteration 1 | Iteration 2 | Iteration 3 | Iteration 4 | Iteration 5 | Iteration 6 | Page Rank |
| A |  |  |  |  |  |  |  | **3** |
| Q |  |  |  |  |  |  |  | **4** |
| X |  |  |  |  |  |  |  | **5** |
| Y |  |  |  |  |  |  | 0.2415 | **1** |
| Z |  |  |  |  |  |  | 0.2415 | **2** |

**Python Matrix Multiplication:**

**A screenshot of a computer code

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a computer code

Description automatically generated**

**A screenshot of a computer code

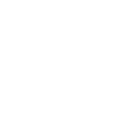
Description automatically generated**

**Approach 2: Removing dead ends.**

Given Q and Z are dead ends removing recursively. We will conduct Page Rank of the remaining 3 nodes: A, X and Y.

**A diagram of a diagram

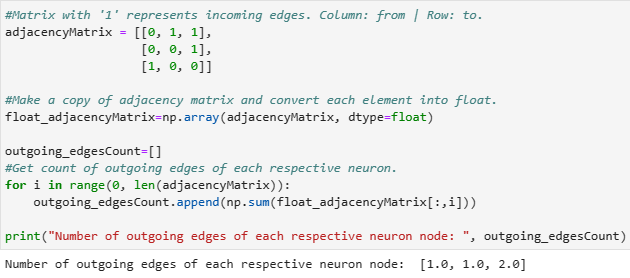
Description automatically generated**



Given network graph:



**Calculate Using Python till converged.**

****

**A screenshot of a computer code

Description automatically generated**

**A screenshot of a computer code

Description automatically generated**

**Now we have final ranking of A, X and Y. Using original network graph:**

**A diagram of a diagram

Description automatically generated**

* = = = 0.233
* = 0.233

**At iteration 85, page rank value converged.**

**Page Rank of Q equals to Page Rank of Z = 0.233**

**Part 5)** Exercise 5.1.6 from Mining of Massive Datasets.

A diagram of a graph

Description automatically generated

**Exercise 5.1.6: Suppose we recursively eliminate dead ends from the graph, solve the remaining graph, and estimate the PageRank for the dead-end pages as described in Section 5.1.4. Suppose the graph is a chain of dead ends, headed by a node with a self-loop, as suggested in Fig. 5.9. What would be the Page- Rank assigned to each of the nodes?**

In given network, assuming we have eliminated all dead-end nodes, there will only be the first origin node (furthest left) in the network. This will leave its Page-rank value of 1, given origin node has an outgoing edge to itself.

Now, adding the first dead-end node.

A diagram of a circle with a line

Description automatically generated

First dead-end node received importance Page-Rank value from the origin node (self-loop). Therefore, Page Rank value of first dead-end node will be of Origin node Page-rank = \* 1 =

Now, adding the second dead-end node.

A black line with a circle and a arrow

Description automatically generated

Second dead-end node received importance Page-Rank value from the first dead-end node fully. Which means Page Rank value of second dead-end node will be the same as first dead-end node of

Finally, since the graph **is a chain of dead ends** with current dead-end nodereceived full importance Page-Rank value from the previous dead-end node. **Page Rank for the Origin node (self-loop) will be 1, and for all dead-end nodes.**

**Part 6)** Page Rank Algorithm.

**1. Report how many nodes and edges the web-Stanford.txt contains.**

* **Command:** cat web-Stanford.txt | less
* **Nodes - i.e., webpage:** 281,903
* **Edges – i.e., hyperlinks:** 2,312,497

**A screenshot of a computer

Description automatically generated**

**2. Report the runtime (took me 4 mins 38 secs).**

**A close-up of numbers

Description automatically generated**

**3. Screenshot of the first page of nodes.**

**A screenshot of a computer

Description automatically generated**