Harp PageRank

**Goal**

For this project you will once again be utilizing PageRank, this time with Harp rather than Hadoop.

**Deliverables**

Zip your source code and output as username\_harp-pagerank.zip. Please submit this file to the Canvas Assignments page.

**Evaluation**

The point total for this project is 20, where the distribution is as follows:

1. Completeness of your code (16 points)
2. Correct output (4 points)

**Prerequisites**

1. Before working on Harp Page Rank, make sure you can successfully run Harp WordCount following the instructions of Lab10 (the presentation is in Canvas).
2. Download **simplepagerank** folder from Canvas assignments under **B649\_Project8** folder.
3. Copy that folder to **harp2-project-master/harp2-app/src/edu/iu**
4. Open **harp2-project-master/harp2-app/build.xml** and add the following line next to where you find other **include** tags. Note that this is within the **javac** tag of the build.xml

**<include name="edu/iu/simplepagerank/\*\*" />**

1. The **input** folder contains the adjacency matrix for 5000 URLs split into two files as **pr\_0** and **pr\_1**.

**Harp PageRank Architecture**

**0**

**1** 2

**2** 1

**3** 0 1

Partial Graph

Partial Initial PR Table

Allgather

Complete Current PR Table

Partial Final PR Table

Allreduce

Complete Final PR Table

**4**

**5** 2

**6** 1

**7** 0 1

Partial Graph

Partial Initial PR Table

Complete Current PR Table

Partial Final PR Table

Complete Final PR Table

With + operator

**Harp Implementation**

Most of the code is completed for you and your task will be to perform the **Compute PR** step in the above diagram. The code for this can be found in **simplepagerank/PageRankMapper.java**

**Compilation and Running**

**public void** computePartialPR(Map<Long, ArrayList<Long>> partialGraph, Long2DoubleKVTable localPRTable, Long2DoubleKVTable globalPRTable){  
   
 **for** (Entry<Long, ArrayList<Long>> entry : partialGraph.entrySet()) {  
 Long sourceUrl = entry.getKey();  
 ArrayList<Long> targetUrls = entry.getValue();  
 System.***out***.println(**"sourceURL: "**+sourceUrl);  
 **if**(targetUrls == **null** ){  
 *// simply assume that the IDs of pages are: 0,1,2,...,(numUrls-1)* System.***out***.println(**"targetUrls is null"**);  
 **double** pr = localPRTable.getVal(sourceUrl) / **numUrls**;  
 *//* ***TODO - Write Code*** *// Add pr to the page rank of all other URLs in globalPRTable  
 //* ***Note. The addKeyVal(key, val) method in globalPRTable will*** *// automatically sum values if the key exists. If the key does  
 // NOT exist then a new entry will be made.* }**else**{  
 **int** numOfOutLinks = targetUrls.size();  
 **double** pr = localPRTable.getVal(sourceUrl) / numOfOutLinks;  
 *//* ***TODO - Write Code*** *// Add pr to the page rank of all target URLs.* }  
 }  
}

1. To compile the code, simply go into **harp2-project-master/harp2-app** and type **ant**
2. Then copy the **harp2-project-master/harp2-app/build/harp2-app-hadoop-2.6.0.jar** to **$HADOOP\_HOME**
3. Before running the program, copy input files from the **input** folder to HDFS.

hdfs dfs -mkdir -p pr/input5k

hdfs dfs -put <path-to-input-dir>/pr\_\* pr/input5k

1. To run the program, use the following command within **$HADOOP\_HOME:**

hadoop jar harp2-app-hadoop-2.6.0.jar edu.iu.simplepagerank.HarpPageRank pr/input5k pr/output5k 5000 10

This will run PageRank against 500 URLs. The program will run 2 parallel map tasks for 100 iterations. Be aware you may want to give unique directory names for the last two parameters each time you run a test. Otherwise, you may run into issues because of existing directories. Alternatively, you can delete old directories and reuse the same names.

1. To get the output, perform the following commands and look at the part- files as usual.

hdfs dfs -get pr/output5k

vim output5k/part-m-00000