**PROJECT 2: REPORT**

PageRank is an [algorithm](https://en.wikipedia.org/wiki/Algorithm) used by [Google Search](https://en.wikipedia.org/wiki/Google_Search) to rank websites in their search engine results. It is named after [Larry Page](https://en.wikipedia.org/wiki/Larry_Page). PageRank is a way of measuring the importance of website pages. PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.

PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR (p_j)}{L(p_j)}Below is the formula used to calculkate page rank.

…….. Equation (1)

where: PR(pi) - page under consideration

L(pi), the number of outbound links on page pj

d, **damping factor,** which can be set between 0 and 1 (usually set to 0.85)

N, total number of pages

Note: PageRank forms a probability distribution over web pages, so the sum of all web pages' PageRanks will be one.

**DESCRIPTION OF THE MAIN STEPS AND DATA FLOW IN THE PROGRAM:**

Initially the HadoopPageRank is run. It contains the main method. This is the driver class where all the jobs are invoked. The given input file is an adjacency matrix. The first column represents the source url. The other clumns represent their corresponding target urls.

Example:

0

1 2

2 1

3 0 1

In the second row, 1 is the source url, 2 is its target url. Below is the graphical representaion of the above matrix.

**CreateGraphMap:**

1. The input to the CreateGraphMap is in the form of <offset1, line1 from the file>, <offset2, line2 from the file>,…, <offsetN, lineN from the file>
2. The line value is split using the space delimiter. The first part of the string is the source url. The second part contains the target urls.
3. The probability (pageRank) of going to any page is given by: **1/N**, where N is the total number of nodes. [Since the sum of pageRanks of all the nodes is 1]
4. The mapper adds one column: 'initial pagerank value', here 1/N, which is appended with the target urls separated by hash symbols (**#**).
5. The CreateGraphMap output is of this form: <sourceUrl, (1/N)#targetUrl1#targetUrl2…>

Example:

1. 0.25
2. 0.25#2
3. 0.25#1
4. 0.25#0#1

**CreateGraphReduce:** The reducer just passes the <key, value> pairs (in the format explained above) to the PageRankMap job.

**PageRankMap:** It takes the output of the **CreateGraphReduce** and calculates the page rank values for all the nodes as explained below.

1. The Record reader gives the output of the CreateGraphReduce (line by line) as input to the **PageRankMap** job in the format <offset, sourceUrl pageRank#targetUrl1#targetUrl2…>

Example: <offset, 3 0.25#0#1>

1. The Rank record constructor is called which splits the value based on pound sign and tab.

Consider the example “3 0.25#0#1”

strArray[0] = 3 0.25

sourceUrl = 3

rankValue = 0.25

strArray[1] = 0 (targetUrl1)

strArray[2] = 1 (targetUrl2)

1. If there are no target urls, then it is a **dangling node**. Its rank is equally distributed to all the nodes by calculating: 1/N. In our implementation, we are not emitting any outbound link information if the node under consideration is an dangling node.

Example: Here the sourceurl 0 is a dangling node, thus we get,

0, 0.0625

1, 0.0625

2, 0.0625

3, 0.0625

1. If it is not a dangling node, then we calculate the rankPerTargetUrl value, which is the pageRank that needs to be equally distributed to each of the node’s targetUrls.

Then we emit <targetUrl1, rankPerTargetUrl>, <targetUrl2, rankPerTargetUrl>…… and <sourceurl, #targetUrl1#targetUrl2#targetUrl3…>

Consider the example “3 0.25#0#1”

<0, 0.125>

<1, 0.125>

<3, #0#1>

**PageRankReduce:** It sums up the pageRanks for each of the sourceUrls, concatenates the targetUrlsList outputs in the format <sourceUrl, sumOfPageRankValuesReceived#targetUrl1#targetUrl2#targetUrl3…>.

If the node is a dangling node then the output is in the format <sourceUrl, sumOfPageRankValuesReceived>

Few examples: <0, 0.1875>

<1, 0.4375#2>

<2, 0.3125#1>

<3, 0.0625#0#1>

**CleanUpResultsMap:** It removes the target urls column and outputs <sourceUrl, calculatedPagerank> as the final result. (Source url and rank is split using the rank record and sent on to the reducer.)

Example: <0, 0.1875>

<1, 0.4375>

<2, 0.3125>

<3, 0.0625>

**CleanUpResultsReduce:** The mapper outputs the source url and page rank as <key,value> pairs.

**OUTPUT RESULTS:**

The page rank output in descending order, for 10 iterations is:

4 0.12069854390945797

34 0.10776863798237149

0 0.0965106743001587

20 0.07730804787564148

2 0.036904835289051814

146 0.035173344193093356

3424 0.03098594895361128

14 0.01645932844842567

16 0.011379122999302937

12 0.010968028739305434