YL5090

PageRank Algorithm:

For each node A, PR(A) = (1-d) + d \* (PR(B) / L(B) + PR(C) / L(C) + …)

While PR(B) is the PageRank of B, and L(B) is the number of out links from B

Data structure:

* Links : An RDD object, each element being (‘a’, <iterable object>). The iterable object is a list of all urls a outputs to. Example : <iterable object>.data returns (‘b’, ‘c’, ‘d’) which means a has outlink to b, c, and d
* Ranks: An RDD object, each element being (url, 1.0). This is the initial rank of each node. This also stores the final pagerank value, which is the probability of the random surfer being at this node after iteractions, multiplied by the number of pages.

The Map function:

* This is a flatMap function = a map + a flatten step
* Map step:
  + The Key is all the node with outlinks. E.g. ‘a’
  + The value is a list, element being (<nodes that this node links to>, 1/L). L is the total number of outlinks at this node. E.g. (‘b’, ½) if a outputs to ‘b’ and ‘c’
* Flatten step:
  + Take all the key-value pairs output in the map step. Remove the keys and merge all values. Eventually will get a list of elements like (‘b’, 1/2), which is the contribute of *some* node to node ‘b’

The Reduce function:

* Reduce by Key. Sum all the contributions made to a node. Multiplied by d. Then plus (1-d), as in the page rank formula.

Reference: Spark’s implementation of Pagerank. https://github.com/apache/spark/blob/master/examples/src/main/python/pagerank.py