The main program reads an input file which contains the topological information among 1000 random generated urls. The input file will be parsed and converted to an adjacency matrix stored in a hash map <Url, OutboundLinks>.

To store and update the parameters for the algorithm, we use a hash map <UrlId, NumberOfOutbound>. The out4bound degree is the number of urls that the current url is pointing to. Also we have another hash map <UrlId, PageRank> to store updated page rank value after each iteration. And the initial page rank for each url is 1/N, which N is the number of total urls.

Then we calculate the page rank value of each url by the following formula,

PR(C) = ((1-d) + d \* PR(T1) / D(Tn) + .... + d \* PR(Tn) / D(Tn))/N,

Where PR(C) is the page rank of current url, d is the damping factor, PR(Tn) is the page rank of inbound links and d is the damping factor of value 0.85.

One challenge is to handle the dangling notes which have no outbound links. There are two ways to solve it.

a) Before running the algorithm, find all dangling nodes and add outbound links to every other urls.

b) After running the algorithm, find the dangling nodes and add page rank value to themselves.

Both method works well and the second one is more time efficient. We use the command

java SequentialPageRank PageRankData/pagerank.input.1000.urls.6

qiuwshou\_HadoopPageRank\_output.txt 10 0.85

Also we verify that the page rank value of all urls sums up to 1.