**Submission Items:**

1. README.pdf - contains the relevant information and output results of this project
2. SimplePageRank.java - implements section 4 of the spec sheet (more details below)

**SimplePageRank**

**Overall Structure**

The map function processes the edges.txt file and group the entries based on both their source and destination node ID. In other words, every entry will be put into 2 buckets one for the source node ID bucket and the other one for the destination node ID bucket. Note that after the map function, we will emit 2 times the number of intermediate entries compared to the original edges.txt.

In the reduce function, we group the entries within each bucket into 2 arraylists based on whether the bucket key appears in the source ID or destination ID of the entry. This helps us differentiate the entries between incoming vs. outgoing links. In the 0th MapReduce pass, we initialize all nodes to have a pagerank of 1/(number of nodes) as well as count the degree of each node based on the number of outgoing links. In the ensuing step, we will output entries with the following format: srcID destID srcPR srcDegree. The 0th pass can be considered as an initialization to step to transform edges.txt into a format we can use more conveniently. On subsequent MapReduce pass, we will calculate the new pagerank of a node based on the constants and the pageranks of the incoming links (using one of the arraylists). We will also get the residual error compare to the previous result and store it in a counter.

After ever MapReduce pass, the main function will get the total value of residual error for all nodes and divide that by the number of nodes to get the average residual error for the pass.

**Compile and Run**

We built, ran and exported the project via eclipse but we did not use any additional jar file other than the ones in Hadoop 2.7.2. Therefore, as long as the classpath is configured correctly, there should not be an error to compile a jar file.

Please run the jar file with the follow two arguments:

1. input file: s3n://input\_folder/file\_name (e.g. s3n://edu-cornell-cs-cs5300s16-ah935-mapreduce/input/edges.txt)
2. output location: s3n://output\_folder/file\_name (e.g. s3n://edu-cornell-cs-cs5300s16-ah935-mapreduce/output/pass)

**Results**

Result from pass 1   
Average Residual Error: 2.337874

Result from pass 2   
Average Residual Error: 0.32303602

Result from pass 3   
Average Residual Error: 0.19205526

Result from pass 4   
Average Residual Error: 0.093996204

Result from pass 5   
Average Residual Error: 0.06276055

Here we can see that after 5 passes, the residual error is still around 6.27% compared to the previous iteration so we have not converged. Note that we have try and deploy our program on AWS EMR but for some weird reason the custom counter do not show up/get registered on EMR whereas it works perfectly on the local machine. However, the final output pagerank values on EMR is the same as the ones on local machine so we have decided to report residual errors based on our local test run (which should be the same on EMR because every MapReduce pass produces the same pagerank values)