**TASK2: SPARK Data Types**

1- Read the matrix in coordinate format. Hand in the output that shows you correctly created the RDD in coordinate format.

2- Convert the matrix from coordinate format to row format and show the output that demonstrates that you have correctly done the conversion.

**Task3: Jupyter and Python or Mappers-Reducers adjusted for Spark**

**Problem:** Relative frequencies of word-pairs problem: Solve the following problem with PySpark code, using the basic principles of Hadoop MapReduce. You must write Hadoop MapReduce pseudocode first and clearly identify and distinguish the phases you produce through MapReduce. Then, implement and execute your program using the proper graph in PySpark.

Input: text documents

key: document id

value: text document

Output: key/value pairs where

key: pair(word1, word2)

value: #co-occurrences(word1, word2)/#co-occurrences(word1, any word)

**Solution:**

**Task 4: Hadoop-Wordcount Example in Java and Python:**

a) Compile and execute the wordcount code in the wordcount/java folder using the Make file. Show the screenshot of the output.

b) Go to the wordcount/streaming directory. Write the wordcount map and reduce codes in python (may.py and reduce.py). Also edit the Make file in the same directory to compile your code. Hand-in the code, make file, and a screenshot of the output.

**Task 5: Hadoop-Sparse Matrix-Matrix Multiply in Python:**

a) Write the hadoop map and reduce files in python to read the input files for matrices A and B from the input folder and print their multiplication. The rank of each matrix is 100 and they are both sparse square matrices.

b) Create a make file similar to the one in the wordcount example to run your code.

c) Hand-in the code, makefile, and a screenshot of the output.

**Task6: Write the corresponding Mappers and Reducers in Hadoop to find the 100 words that get most frequently used in a document:**

1. Write the corresponding Mappers and Reducers in Hadoop for Task 7. For your ease you may modify accordingly the ones you used for the Word Count and you may also modify the make file to include the text you have used.
2. Please submit the code, the solution, and your output.
3. To test for correctness , use the following document: <http://www.gutenberg.org/files/1342/1342-0.txt>
4. **What if we play with the combiner as well? The results you have obtained so far (if you reached this far) is without the combiner.** But remember that in the shuffle and sort phase, the partitioner will send every single word (the key) with the value “1” to the reducers. Will we get any benefits if we attempt to reduce locally the data emitted by the mapper? What is the role of the combiner here? Think about how to change the combiner and recompute your Mapper-Reducer jobs. Compare against the original version and have some print screens on the map and reduce input-output records, bytes, combine input-output records, shuffle input-output and compare. **Hint:** under which circumstances (mathematical properties of the reduce function) can we use the reducer as a combiner?
5. How can we change the mapper to include the combiner in its own function?