Cloud Computing – Project 2 (Map Reduce)

1. Submission -

* The code used to solve the task with map-reduce is in file **AdjList.java**
* The result file with output of steps 4 and 5 from the task section comprise of the following files –
  + directedlargestadjlist-m-00000 – This file contains the largest adjacency list in the **directed** graph. Format is - the node, followed by a space, and the node’s adjacency list (comma separated).
  + directedmaxconn-m-00000 – This file contains the node with the maximum connectivity in the **directed** graph. Format is - the node, followed by a space, and the count of connectivity.
  + directedminconn-m-00000 - This file contains the node with the minimum connectivity in the **directed** graph. Format is - the node, followed by a space, and the count of connectivity.
  + text-m-00000 – This file contains the largest adjacency list in the **undirected** graph. Format is - the node, followed by a space, and the node’s adjacency list (comma separated).
  + undirectedmaxconn-m-00000 - This file contains the node with the maximum connectivity in the **undirected** graph. Format is - the node, followed by a space, and the count of connectivity.
  + undirectedminconn-m-00000 - This file contains the node with the minimum connectivity in the **undirected** graph. Format is - the node, followed by a space, and the count of connectivity.

1. Explanation –
   1. The code uses **MultipleOutputs** class in the Hadoop API to be able to write different outputs to multiple files, each with a key piece of information as required in the submission.
   2. File names generated by Hadoop (ex. directedlargestadjlist-m-00000, text-m-00000, etc) are due to the usage of the **MultipleOutputs** class.
   3. The code uses a **single pass** to convert the graph into directed and undirected. In the mapper, I append a “d” to every node, and write it to the context. This creates the directed graph. Then, I again write the node and value as (key, value) and value and node as (key, value) to the context. This creates the undirected graph. In the reducer, I read the current node and check to see if it ends with “d” – if it does, I process it as a directed graph, else as an undirected graph. Having processed it, I write the output to different files, that would be generated by **MultipleOutputs**, based on what type of graph it is.
   4. Lastly, the code also contains logic to prevent duplicates (using HashSet) that would arise when converting the directed graph to undirected.