COMP9313 2018s1 Project 1 (10 marks)

Problem statement:

Given a directed graph, compute the average length of the in-coming edges for each node.

每个node：进来的edges的avg length

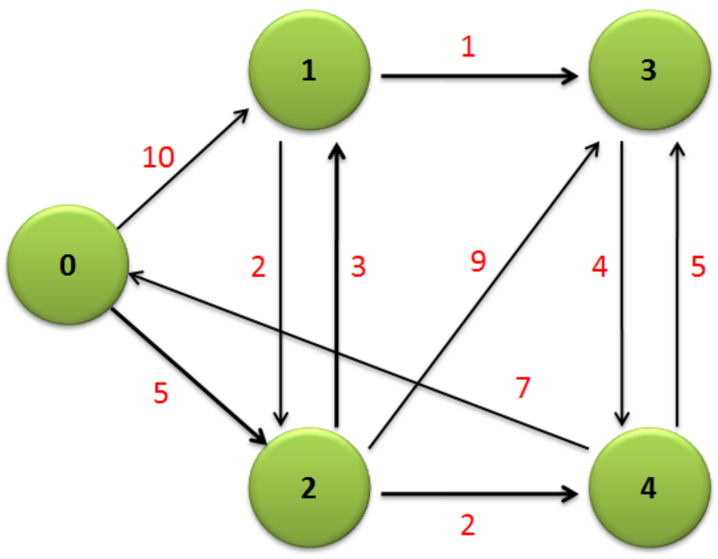
Input files:

In the input file, each line is in format of:

combiner 不能用结合律

"EdgeId FromNodeId ToNodeId Distance".

统计 txt中，第三列= toNode， 后边的 distance，前两列不用读?



In the above example, the input is like:

```
0 0 1 10.0
1 0 2 5.0
2 1 2 2.0
3 1 3 1.0
4 2 1 3.0
5 2 3 9.0
6 2 4 2.0
7 3 4 4.0
8 4 0 7.0
9 4 3 5.0
```

This sample file "tiny-graph.txt" can be downloaded at:

https://webcms3.cse.unsw.edu.au/COMP9313/18s1/resources/15769

Output: 用一个reducer

Set the number of reducers to 1. Each line in the single output file is in format of "NodeID\tAverage length of in-coming edges". The average length is of double precision (using DoubleWritable). Remove the nodes that have no in-coming edges and sort the output by NodeID in ascending order of its numeric value. Given the example graph, the output file is like:

用 Double

结果：
1. 没有coming的不带
2. sort id 0-->1
3.

nodeid avg len of incoming

```
0\t7.0
1\t6.5
2\t3.5
3\t5.0
4\t3.0
```

2个version:
1. combiner
2. In-mapper combining

Code format:

You are required to write TWO versions of MapReduce program to solve this problem. The first version utilizes a combiner, and the second version utilizes the "in-mapper combining" approach.

Write each version in a single java file (like WordCount.java used in Lab 2).

Name your first version as "EdgeAvgLen1.java", and the second version as "EdgeAvgLen2.java", and put them in the package "comp9313.ass1".

Compile:

Your java code will be compiled and packaged as a jar file, and we will use the following commands to check the correctness of your solution:

arg0 arg1

```
$ $HADOOP_HOME/bin/hadoop jar YOURJAR.jar YOURCLASS input output

$ $HADOOP_HOME/bin/hdfs dfs -cat output/*
```

Please ensure that the code you submit can be compiled and packaged. Any solution that has compilation errors will receive no more than 3 points for the entire assignment.

Documentation and code readability

Your source code will be inspected and marked based on readability and ease of understanding. The documentation (comments of the codes) in your source code is also important. Below is an indicative marking scheme:

Result correctness: 80%
Code structure, Readability, and Documentation: 20%

Submission:

Deadline: Sunday 1st Apr 09:59:59 PM

Log in any CSE server (e.g., williams or wagner), and use the give command below to submit your solutions:

```
$ give cs9313 assignment1 EdgeAvgLen1.java EdgeAvgLen2.java
```

Or you can submit through:
https://cgi.cse.unsw.edu.au/~give/Student/give.php

If you submit your assignment more than once, the last submission will replace the previous one. To prove successful submission, please take a screenshot as assignment submission instructions show and keep it by yourself. If you have any problems in submissions, please email to xuefeng.chen@student.unsw.edu.au.

Late submission penalty

You will receive zero marks for this assignment.

Plagiarism:

The work you submit must be your own work. Submission of work partially or completely derived from any other person or jointly written with any other person is not permitted. The penalties for such an offence may include negative marks, automatic failure of the course and possibly other academic discipline. Assignment submissions will be examined manually.

Relevant scholarship authorities will be informed if students holding scholarships are involved in an incident of plagiarism or other misconduct.

Do not provide or show your assignment work to any other person - apart from the teaching staff of this subject. If you knowingly provide or show your assignment work to another person for any reason, and work derived from it is submitted you may be penalized, even if the work was submitted without your knowledge or consent.