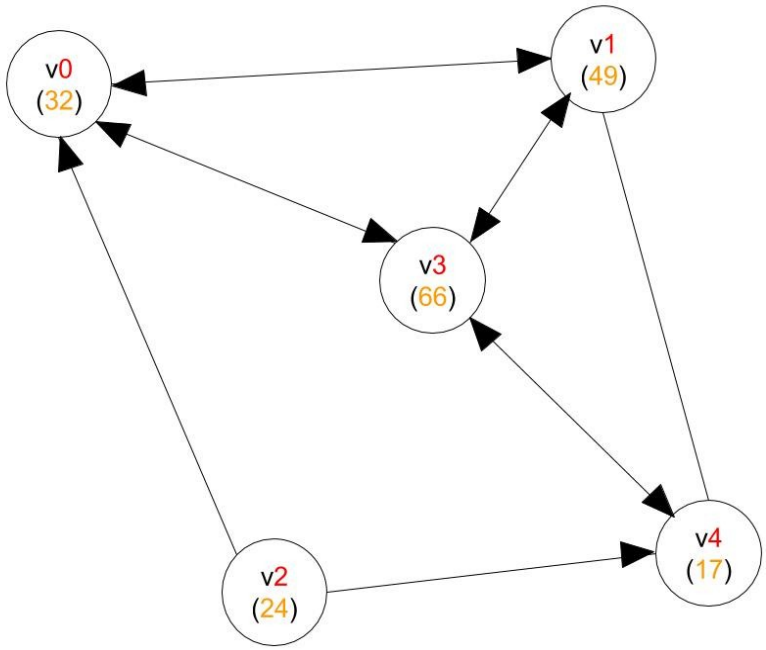
PolyU COMP4434 Assignment 4
Introduction

By doing this assignment, you will understand how to use Giraph to implement basic graph algorithms.

Your job

Write a Giraph program to find the **age of the oldest follower** of each user in a social network graph (e.g. Facebook).

Below is an example follower/followee graph of a social website. Each node represents a user annotated with her age (e.g., user v0 is 32 years old). An directed edge (vi, vj) means user vi is a follower of user vj (e.g., user v2 is a follower of user v0).



Input file format

We use only one plain text file to represent a graph, where each line is in the adjacency-list format [source_id,source_value,[[dest_id,edge_weight],[dest_id,edge_weight],...]]

The input file for the example above (i.e., **graph-data1.txt**) will be like this:

[0, 32, [[1, -1], [3, -1]]]
[1, 49, [[0, -1], [3, -1]]]
[2, 24, [[0, -1], [4, -1]]]
[3, 66, [[0, -1], [1, -1], [4, -1]]]
[4, 17, [[3, -1]]]

For example, [2, 24, [[0, -1], [4, -1]]] means that Node v2 connects to Node v0 and Node v4 **in a directed manner**. Some remarks in terms of the input format:

- The edge weights are not used, so we set them as -1;
- Initially, the node values are set as their age; after running the correct algorithm, we shall set them as the age of each node's **oldest follower**.

Output file format

The computed result should be like this:

0 66.0
2 -1.0
1 66.0
3 49.0
4 66.0

It represents the node value (i.e., **the oldest follower**) of each node. Some remarks in terms of the output format:

- For Node v2 with node value -1.0, it means that Node v2 does NOT have any follower.
- The order of lines in the output file can be different.

To begin with

You are given a zip file Assignment4.zip.
All actions are performed in the VirtualBox image we provided.
Create a new folder ~/Assignment4, put the zip file there and uncompressed it.

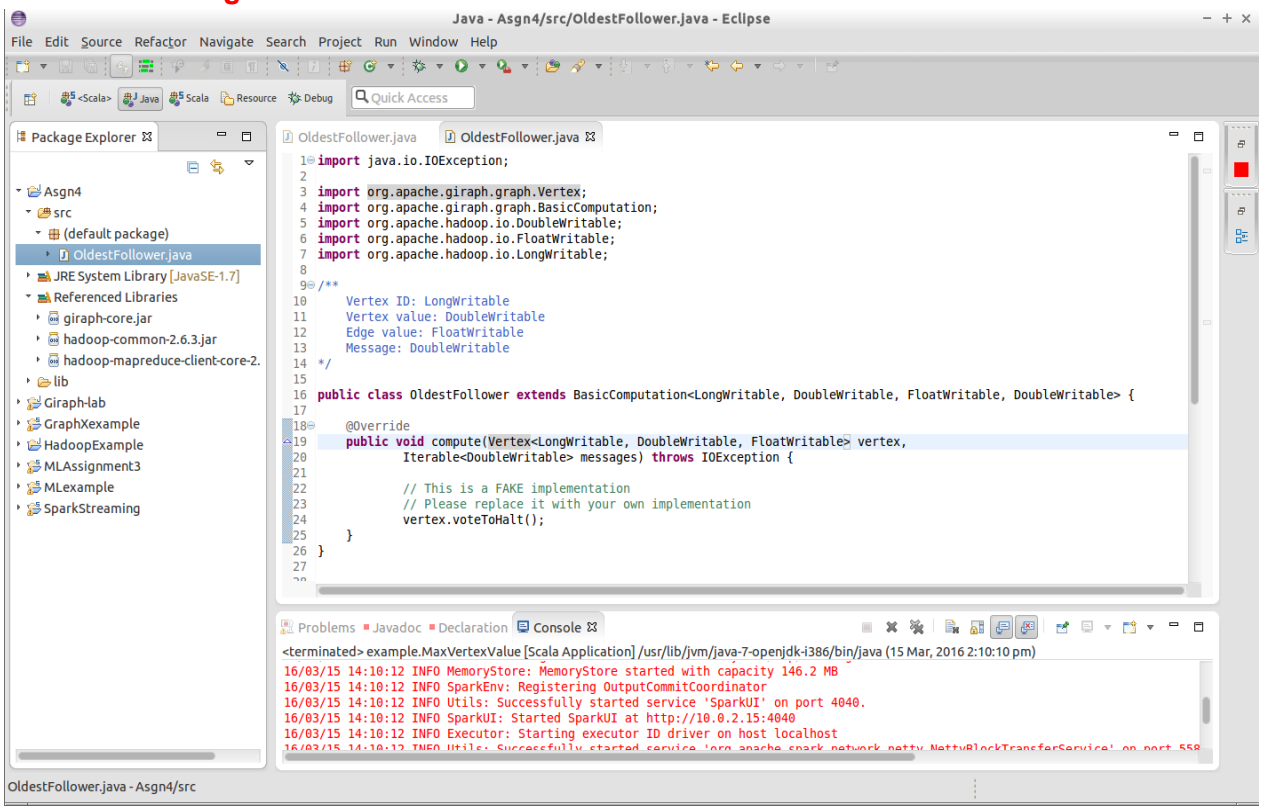
Then you will find:

- ReadMe.pdf (this file)
- **data** folder, it contains two test datasets:
  - graph-data1.txt is the first test dataset:
  - graph-data2.txt is the second test dataset.
  - **Please use these two datasets to test your program (see below), but we will use other two datasets to grade your program.**
- **grading** folder, containing the grading script for two test datasets we provided.
- The archive file Asgn4EclipseProject.zip. You need to work on it.

How to run the program

The main steps are as follows (for details of some steps, please refer to the lab material):

1) Import Asgn4EclipseProject.zip to Eclipse (version 4.3 or above).
2) Work on OldestFollower.java. We have constructed the skeleton of the program for you, and it is runnable; that is to say, even if you edit nothing in OldestFollower.java, you can still continue the remaining steps, only resulting in failed case testing finally. So, **don't modify any given parts of the file except the function body of compute() or otherwise your will get 0 marks. Your job is to complete the compute() function in order to get correct results.**



3) Export the project to a **JAR** file and name it as OldestFollower.jar. As described in lab, do not include the lib folder during the export.

4) Copy your OldestFollower.jar to ~/Programs/hadoop/share/hadoop/common/
5) Start Hadoop. Assume that your current directory is ~/Programs/hadoop/:

```
$ ./sbin/start-all.sh
```

6) Upload the input file (say, **graph-data1.txt**) to HDFS:

```
$ ./bin/hadoop fs -put ~/Assignment4/data/graph-data1.txt  /user/bigdata/giraph/input/graph-data1.txt
```

7) Execute giraph program:

```
$ ./bin/hadoop jar share/hadoop/common/giraph-core.jar
```

```
org.apache.giraph.GiraphRunner OldestFollower -vif
org.apache.giraph.io.formats.JsonLongDoubleFloatDoubleVertexInputFormat -vip
/user/bigdata/giraph/input/graph-data1.txt -vof
org.apache.giraph.io.formats.IdWithValueTextOutputFormat -op
/user/bigdata/giraph/output -w 1 -ca giraph.SplitMasterWorker=false
```

8) Then you can check the output file by:

```
$ ~/Programs/hadoop/bin/hadoop fs -cat /user/bigdata/giraph/output/part-m-00000
```

It may output:

```
0       66.0
2       -1.0
1       66.0
3       49.0
4       66.0
```

How to test your program

- Run testing script. Assume your current directory is ~/Assignment4/grading:

```
$ ~/Programs/hadoop/bin/hadoop fs -cat /user/bigdata/giraph/output/part-m-00000 | java Grading 1
```

Note: replace 1 with 2 if your are testing the second test case.

- If your implementation is incorrect, it may output:

```
Test result for case 1:
-----------------------------------
Node 0 failed, it should have oldest follower with age 66, but the program outputs 32
Node 1 failed, it should have oldest follower with age 66, but the program outputs 49
Node 2 failed, it should have oldest follower with age -1, but the program outputs 24
Node 3 failed, it should have oldest follower with age 49, but the program outputs 66
Node 4 failed, it should have oldest follower with age 66, but the program outputs 17
You scored 0/50 for test case 1
```

- If your implementation is correct, it shall output:

```
Test result for case 1:
-----------------------------------
Node 0 passed
Node 1 passed
Node 2 passed
Node 3 passed
Node 4 passed
You scored 50/50 for test case 1
```

- Note that by default, the output directory in HDFS cannot be written repeatedly. Therefore, if you want to execute giraph program again, **please remove the output directory first**:

```
$ ~/Programs/hadoop/bin/hadoop fs -rmr /user/bigdata/giraph/output
```

What to submit

Submit only **OldestFollower.java** (Don't change the file name or you will get 0 marks). Other files will be ignored.
Your submitted file shuld be free of compilation error and the problem should terminate within 10 minutes. Otherwise you will receive 0 marks.

Grading rubrics

- You can use the grading script to test whether your implementation is correct based on the given graphs.
- When grading, we will feed in 2 other graphs (with N1 nodes and N2 nodes, respectively). Assume your program outputs C1 and C2 nodes **with correct oldest follower**, you can get:
  final sore = 50 * C1/N1 + 50 * C2/N2

Deadline: 18 April 2016, 11:59AM

Late Penalty

late x day: your score = raw score * (100 – 20 * x)%

Plagiarism

It is easy to detect the similarity of source files, and cases will be strictly handled according to the University's regulation, so please don't risk doing that.

Questions?

Reach out your teaching assistant Mr. Wenjian Xu (cswxu@comp.polyu.edu.hk)