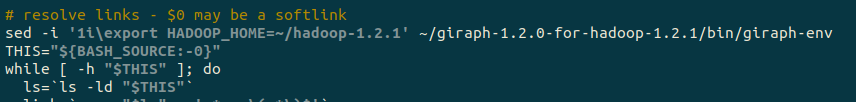
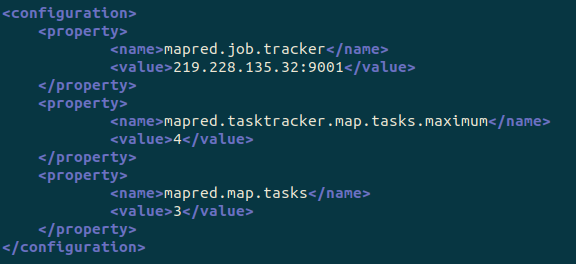
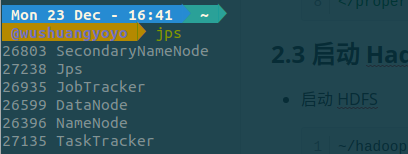
# Giraph 基于 MapReduce v1 部署

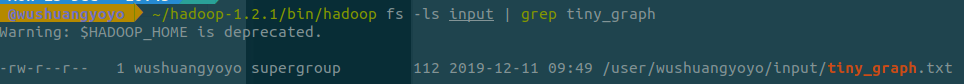
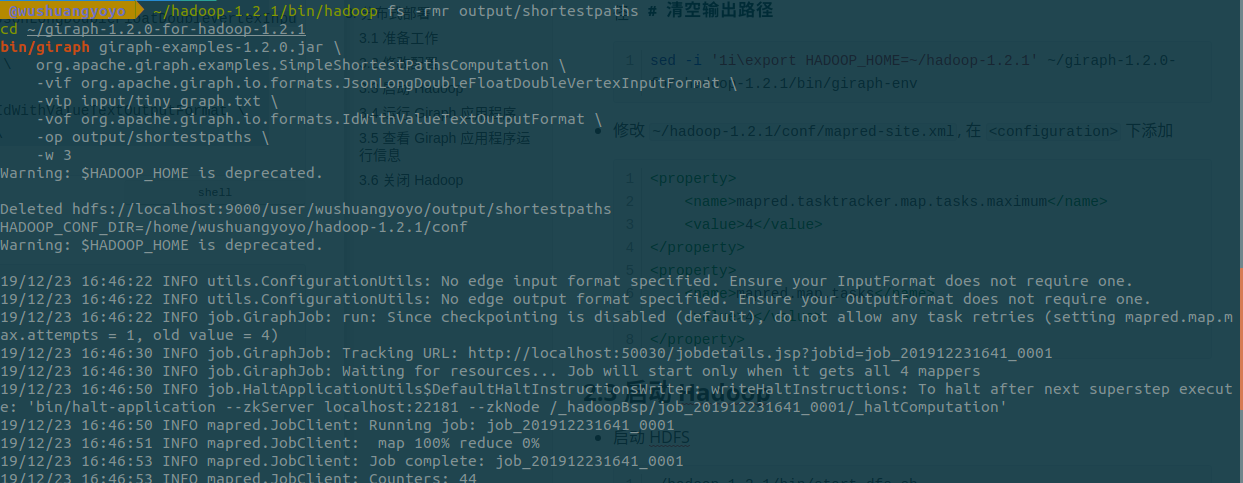
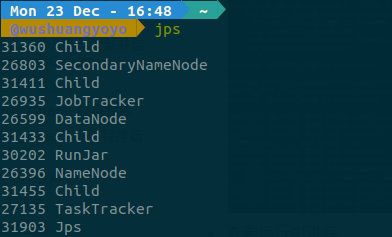
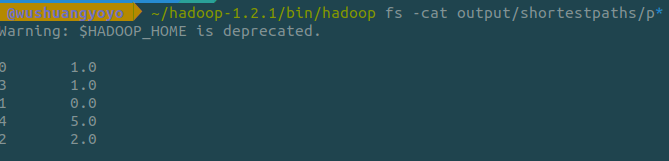
## 1 单机伪分布式部署

### 1.1 准备工作&修改配置并启动Hadoop和Giraph

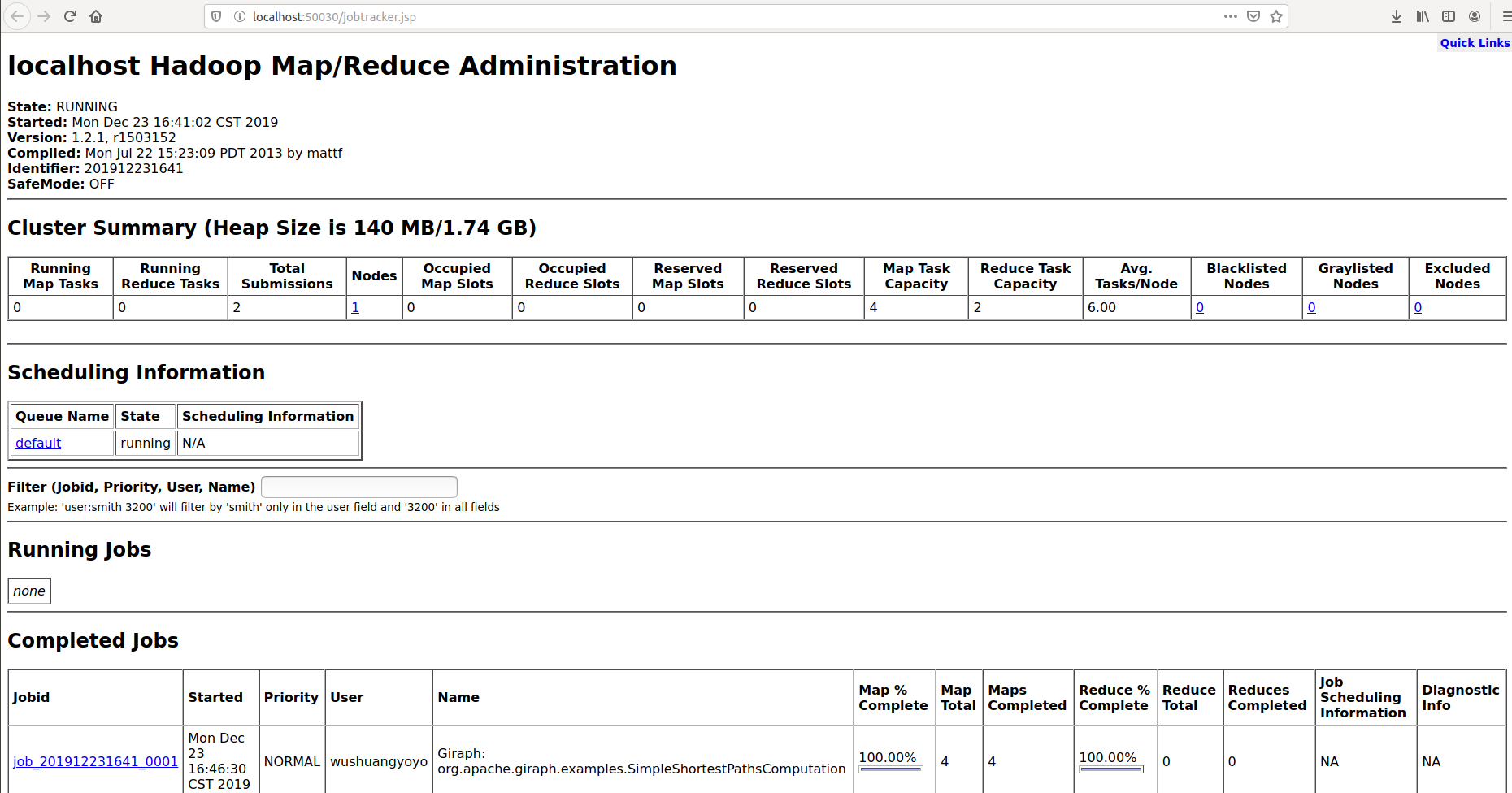
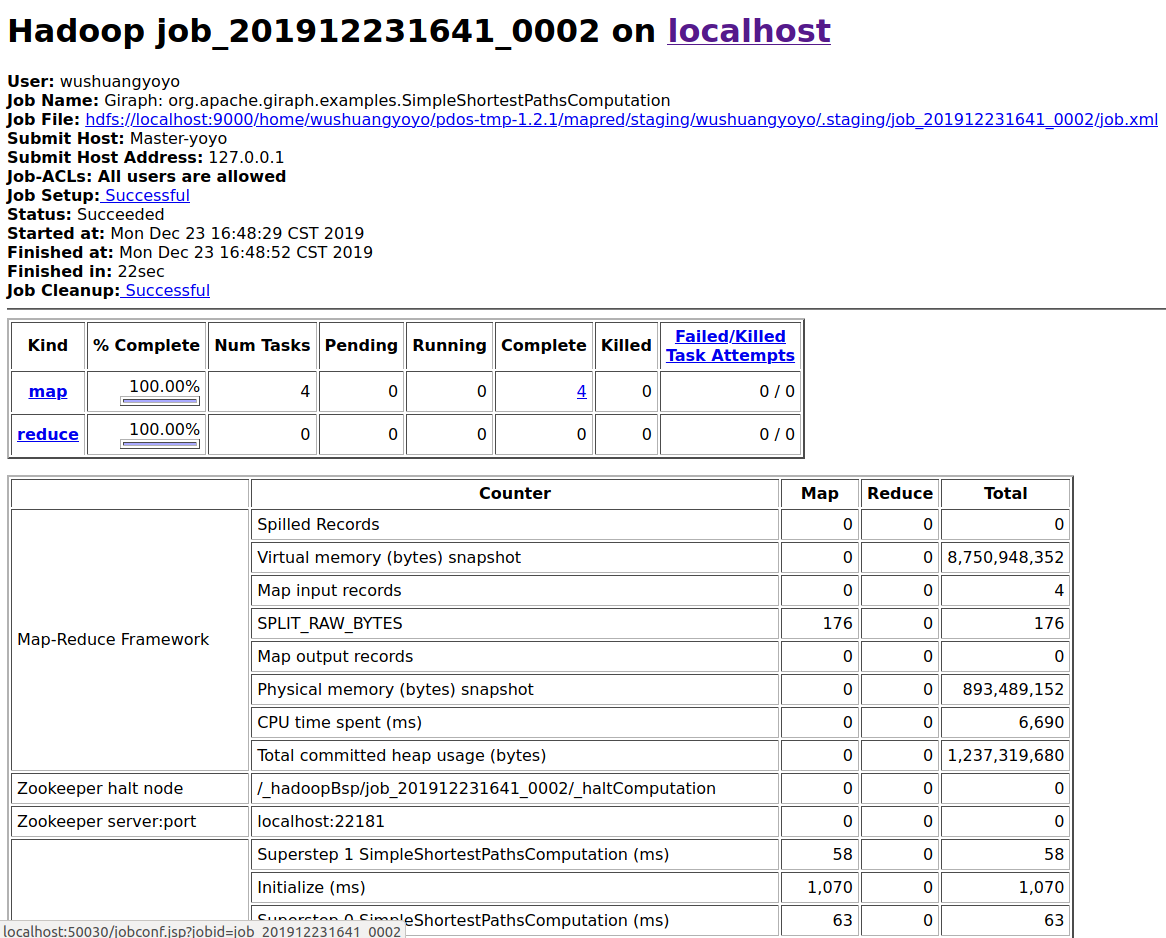
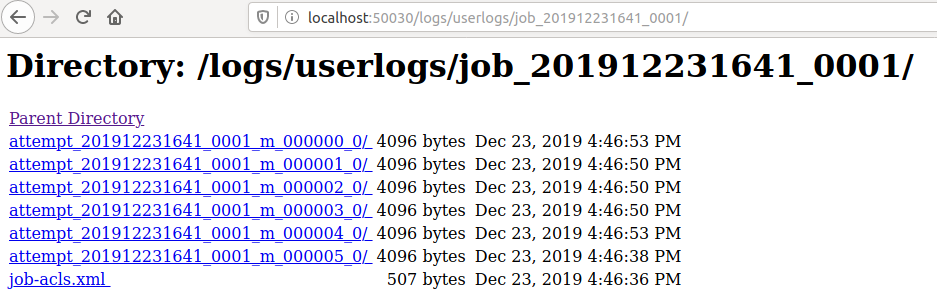
* 修改 ~/giraph-1.2.0-for-hadoop-1.2.1/bin/giraph-env, 指定 Hadoop 安装路径：
* 
* 修改 ~/hadoop-1.2.1/conf/mapred-site.xml，结果如下：
* 
* 启动 HDFS及 MapReduce
* ~/hadoop-1.2.1/bin/start-dfs.sh  
  ~/hadoop-1.2.1/bin/start-mapred.sh
* 通过运行jps来检验进程状态：
* 

### 1.2 运行 Giraph 应用程序

Simple shortest paths computation 示例程序

* 将 tiny\_graph.txt 上传至 hdfs:///user/you/input 下
* ~/hadoop-1.2.1/bin/hadoop fs -mkdir input  
  ~/hadoop-1.2.1/bin/hadoop fs -put ~/tiny\_graph.txt input/
* 查看 HDFS 的文件信息：
* 
* 执行程序
* 按照如下代码执行：
* 
* 查看运行中进程
* 
* 运行完成后查看输出
* 

### 1.3 查看 Giraph 应用程序运行信息

* 访问 JobTracker 网页 (http://localhost:50030)，
* 
* 点击正在运行或已完成的 Giraph 应用程序, 可看到 Giraph 应用程序的统计信息  
  
* 查看程序日志
  + JobHistory 日志默认位置: ~/hadoop-1.2.1/logs
  + 
  + Task 日志默认位置: ~/hadoop-1.2.1/logs/userlogs/<jobid>/<attempt-id>
  + 

### 3.6 关闭 Hadoop

* 关闭 HDFS
* ~/hadoop-1.2.1/bin/stop-dfs.sh
* 关闭 MapReduce
* ~/hadoop-1.2.1/bin/stop-mapred.sh
* 分布式在构建过程中有很多错误，实在无法进行，故放弃。

# Giraph 应用编程实践

## 1. 编写Giraph程序

##### - 创建maven项目

参考文档：[create maven.md](/Create Maven Project/create maven.md)

##### - 添加pom依赖

在pom.xml文件中添加以下依赖：giraph-core、giraph-examples、hadoop-common和hadoop-client。

<dependencies>  
 <!-- https://mvnrepository.com/artifact/org.apache.giraph/giraph-core -->  
 <dependency>  
 <groupId>org.apache.giraph</groupId>  
 <artifactId>giraph-core</artifactId>  
 <version>1.2.0</version>  
 </dependency>  
 <!-- https://mvnrepository.com/artifact/org.apache.giraph/giraph-examples -->  
 <dependency>  
 <groupId>org.apache.giraph</groupId>  
 <artifactId>giraph-examples</artifactId>  
 <version>1.2.0</version>  
 </dependency>  
 <!-- https://mvnrepository.com/artifact/org.apache.hadoop/hadoop-core -->  
 <dependency>  
 <groupId>org.apache.hadoop</groupId>  
 <artifactId>hadoop-core</artifactId>  
 <version>1.2.1</version>  
 </dependency>  
 <!-- https://mvnrepository.com/artifact/org.apache.hadoop/hadoop-client -->  
 <dependency>  
 <groupId>org.apache.hadoop</groupId>  
 <artifactId>hadoop-client</artifactId>  
 <version>1.2.1</version>  
 </dependency>  
</dependencies>

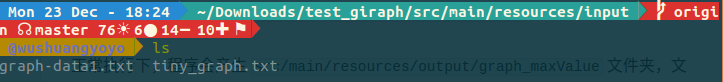
##### - 编写Giraph应用程序代码

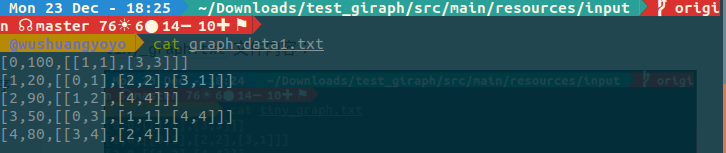
* 新建 src/main/java/example/MaxVertexValue.java 类
* package example;  
    
  import org.apache.giraph.graph.BasicComputation;  
  import org.apache.giraph.graph.Vertex;  
  import org.apache.hadoop.io.DoubleWritable;  
  import org.apache.hadoop.io.FloatWritable;  
  import org.apache.hadoop.io.LongWritable;  
    
  import java.io.IOException;  
    
  /\*\*  
   \* Vertex ID: LongWritable  
   \* Vertex value: DoubleWritable  
   \* Edge value: FloatWritable  
   \* Message: DoubleWritable  
   \* <p>  
   \* Assumption:  
   \* 1. The graph is strongly connected  
   \*/  
  public class MaxVertexValue extends BasicComputation<  
   LongWritable, DoubleWritable, FloatWritable, DoubleWritable> {  
   public void compute(Vertex<LongWritable, DoubleWritable, FloatWritable> vertex, Iterable<DoubleWritable> messages) throws IOException {  
   boolean changed = false;  
    
   for (DoubleWritable msg : messages) {  
   /\* Collect messages from in-neighbours and update if necessary \*/  
   if (vertex.getValue().get() < msg.get()) {  
   vertex.setValue(new DoubleWritable(msg.get()));  
   changed = true;  
   }  
   }  
   /\* Send the message to out-neighbours at Superstep 0 or Vertex value is changed \*/  
   if (getSuperstep() == 0 || changed) {  
   sendMessageToAllEdges(vertex, vertex.getValue());  
   }  
   vertex.voteToHalt();  
   }  
  }
* 新建 src/main/java/GiraphDemoRunner.java 类
* import example.MaxVertexValue;  
  import org.apache.giraph.conf.GiraphConfiguration;  
  //import org.apache.giraph.examples.SimpleShortestPathsComputation;  
  import org.apache.giraph.io.formats.\*;  
  import org.apache.giraph.job.GiraphJob;  
  import org.apache.hadoop.conf.Configuration;  
  import org.apache.hadoop.fs.Path;  
  import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  
  import org.apache.hadoop.util.Tool;  
  import org.apache.hadoop.util.ToolRunner;  
    
  public class GiraphDemoRunner implements Tool{  
    
   private Configuration conf;  
   public Configuration getConf() {  
   return conf;  
   }  
   public void setConf(Configuration conf) {  
   this.conf = conf;  
   }  
    
   public int run(String[] arg0) throws Exception {  
   /\*\*  
   \* 设置输入输出路径  
   \* \*/  
  // String inputPath="src/main/resources/input/tiny\_graph.txt";  
  // String outputPath="src/main/resources/output/graph\_shortestPaths";  
   String inputPath="src/main/resources/input/graph-data1.txt";  
   String outputPath="src/main/resources/output/graph\_maxValue";  
    
   GiraphConfiguration giraphConf = new GiraphConfiguration(getConf());  
    
   /\*\*  
   \* 配置具体用户自定义应用计算类  
   \* \*/  
  // giraphConf.setComputationClass(SimpleShortestPathsComputation.class);  
   giraphConf.setComputationClass(MaxVertexValue.class);  
    
   giraphConf.setVertexInputFormatClass(JsonLongDoubleFloatDoubleVertexInputFormat.class);  
   GiraphFileInputFormat.addVertexInputPath(giraphConf, new Path(inputPath));  
   giraphConf.setVertexOutputFormatClass(IdWithValueTextOutputFormat.class);  
   giraphConf.setLocalTestMode(true);  
   giraphConf.setWorkerConfiguration(1, 1, 100);  
   giraphConf.SPLIT\_MASTER\_WORKER.set(giraphConf, false);  
   InMemoryVertexOutputFormat.initializeOutputGraph(giraphConf);  
   GiraphJob giraphJob = new GiraphJob(giraphConf,"GiraphDemo");  
   FileOutputFormat.setOutputPath(giraphJob.getInternalJob(), new Path(outputPath));  
   giraphJob.run(true);  
   return 0;  
   }  
    
   public static void main(String[] args) throws Exception{  
   ToolRunner.run(new GiraphDemoRunner(), args);  
   }  
  }

## 2. 调试Giraph程序

##### - 配置程序输入

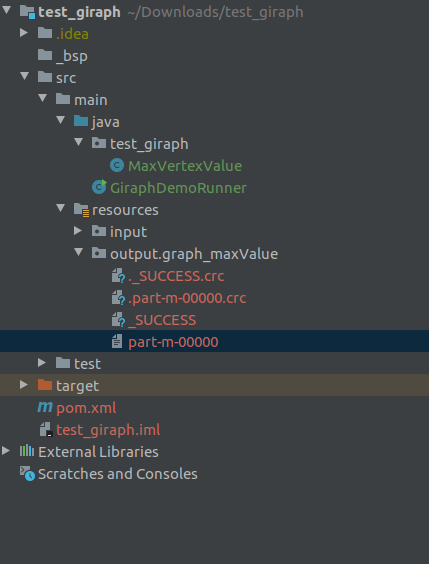
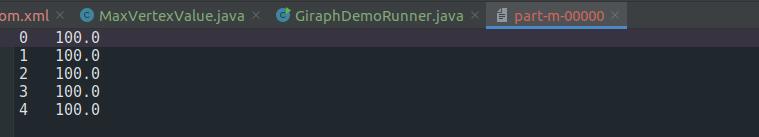
在 src/main/resources/input/ 路径下添加输入文件 graph-data1.txt 和 tiny\_graph.txt：



* graph-data1.txt 文件内容：
* 
* tiny\_graph.txt 文件内容：
* 

##### - IDE中直接运行Giraph MaxVertexValue 应用程序

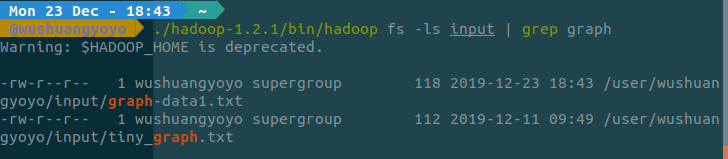
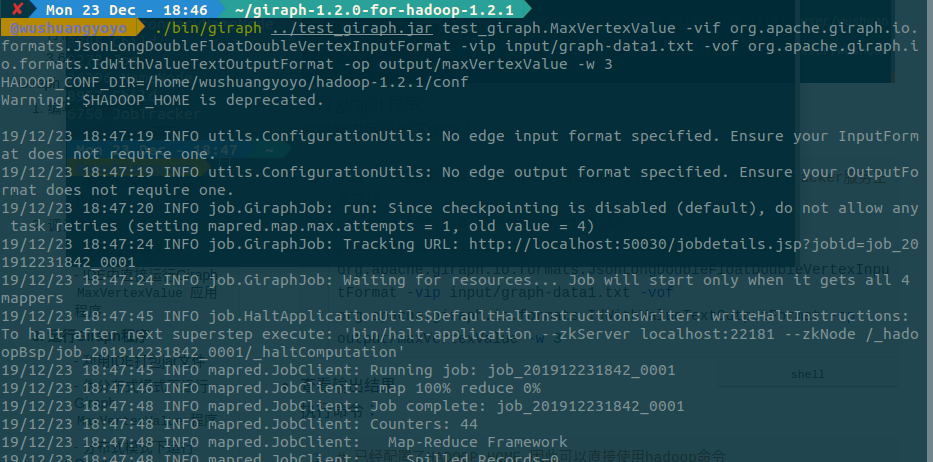
直接在 IDEA 中运行 src/main/java/GiraphDemoRunner.java 类，并查看输出结果。 正常执行下，程序会产生 src/main/resources/output/graph\_maxValue 文件夹，文件夹内包含程序输出内容。

* 正常执行情况下，项目结构：
* 
* 程序输出内容：
* 

## 3. 运行Giraph程序

##### - 利用IDE打包jar文件

##### - 伪分布式模式下运行Giraph MaxVertexValue 程序

* 上传输入文件至 HDFS
* 
* 运行 giraph 程序  
  在终端中运行如下命令：
* jps #查看和确保 NameNode,DataNode,JobTracker以及TaskTracker服务正常启动  
  cd giraph-1.2.0-for-hadoop-1.2.1/   
  ./bin/giraph giraph.jar example.MaxVertexValue -vif org.apache.giraph.io.formats.JsonLongDoubleFloatDoubleVertexInputFormat -vip input/graph-data1.txt -vof org.apache.giraph.io.formats.IdWithValueTextOutputFormat -op output/maxVertexValue -w 3
* 结果如下：
* 
* 查看输出结果
* 执行命令：
* 

##### - 分布式模式下运行Giraph MaxVertexValue 程序

和上一部分一致，无法完成。