任务描述

任务：对输入文件 file1.txt、file2.txt、file3.txt 中的数据进行排序。输入文件中的每行内容都是一个数字，即一个数据。要求输出中每行有两个间隔的数据，结果按照数据的字母顺序进行排序，每个数据占一行，其中第二个数字代表原始数据，第一个数字是原始数据在新数据集中的位次。

编程要求

根据提示，在“SortDrive.java”文件中的 Begin-End 间补充代码新建一个 JOB 和设置主类、MAPPER 类，同时设置输出的 key 和 value 的类； 在“Partition.java”文件中的 Begin-End 间补充代码实现对输入文件 file1.txt、file2.txt、file3.txt 中的数据进行排序。

> 注意：请先在右侧命令行中的任意目录使用命令启动 Hadoop。

file1.txt 文件内容如下：

```
1. 123
2. 5656
3. 45567
4. 54
5. 677
6. 43
7. 888
8. 12306
9. 6652
```

file2.txt 文件内容如下：

```
1. 994
2. 44455
3. 23432
4. 23309
5. 5
6. 866
7. 62
8. 9
9. 288
```

file3.txt 文件内容如下：

```
1. 66
2. 36
3. 237
4. 866
5. 45555
6. 23455
7. 12344
8. 22222
```

程序如下：

Partition.java

```
package com.ky;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapreduce.Partitioner;


    public  class Partition extends Partitioner<IntWritable, IntWritable>{

        @Override

        public int getPartition(IntWritable key,
                                IntWritable value,
                                int numpartition){
            int maxnum = 65223;
            int bond = maxnum /numpartition+1;
            int keyint = key.get();
            /**********Begin**********/
            return 1;

            /**********End**********/
            }
}
```

SortMapper.java

```
package com.ky;


import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;



public class SortMapper extends Mapper<Object, Text,IntWritable, IntWritable>{
      IntWritable result = new IntWritable();
    public void map(Object key,
                    Text value,
                    Context context) throws IOException, InterruptedException{
          String line =value.toString();
          int num =Integer.parseInt(line);
          result.set(num);
          context.write(result, new IntWritable(1));
      }

}
```

SortReduce.java

```
package com.ky;
```

```
import java.io.IOException;


import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapreduce.Reducer;

public class SortReduce extends Reducer<IntWritable,IntWritable, IntWritable,
IntWritable>{

    //定义一个全局变量，存储某个数的排位
    private static IntWritable num = new IntWritable(1);

    public void reduce(IntWritable key,
                       Iterable<IntWritable>  values,
                       Context context) throws IOException,
InterruptedException{

        for(IntWritable val:values){
            context.write(num, key);
            num = new IntWritable(num.get()+1);
        }




    }
}
```

SortDrive.java

```
package com.ky;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;


public class SortDrive {

    public static void main(String[] args) throws IOException,
ClassNotFoundException, InterruptedException {
        /**********Begin**********/
        // 新建一个JOB
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf);
        // 设置主类、MAPPER类
        job.setJarByClass(SortDrive.class);
        job.setMapperClass(SortMapper.class);
        /**********End**********/
```

```
            //Partition类
            job.setPartitionerClass(Partition.class);
            //设置Reduce的任务数
             job.setNumReduceTasks(5);
             job.setReducerClass(SortReduce.class);


            /**********Begin**********/
            //设置输出的key和value的类
            job.setOutputKeyClass(IntWritable.class);
            job.setOutputValueClass(IntWritable.class);
            /**********End**********/

            //设置输入输出的路径
            FileInputFormat.addInputPath(job, new Path("/input"));
            FileOutputFormat.setOutputPath(job, new Path("/output"));
            job.waitForCompletion(true);

    }

}
```