

数据科学导论实验报告

实验三 MapReduce 编程初级实践



学 生：

年 级：2021 级

专 业：软件工程

重庆大学大数据与软件学院

2023 年 10 月 20 日

一、 实验目的：

1. 通过实验掌握基本的 MapReduce 编程方法。这将包括理解如何有效地使用 Map 和 Reduce 操作，以实现分布式计算任务。这些实验将为我们提供一个坚实的基础，使我们能够更好地理解和运用 MapReduce 框架；

2. 掌握用 MapReduce 解决一些常见的数据处理问题，包括数据去重（从数据集中识别和删除重复的条目）、数据排序（按特定标准对数据进行排序）和数据挖掘（运用 MapReduce 进行数据挖掘，以揭示数据中的模式、趋势和关联，从数据集中提取有价值的信息）等。

二、 实验要求

1. 编程实现文件合并和去重操作

对于两个输入文件，即文件 A 和文件 B，请编写 MapReduce 程序，对两个文件进行合并，并剔除其中重复的内容，得到一个新的输出文件 C。下面是参考示例：

输入文件 A 的样例如下：

20150101	x
20150102	y
20150103	x
20150104	y
20150105	z
20150106	x

输入文件 B 的样例如下：

20150101	y
20150102	y
20150103	x
20150104	z
20150105	y

根据输入文件 A 和 B 合并得到的输出文件 C 的样例如下：

20150101	x
20150101	y
20150102	y
20150103	x
20150104	y

20150104	z
20150105	y
20150105	z
20150106	x

2. 编写程序实现对输入文件的排序

现在有多个输入文件，每个文件中的每行内容均为一个整数。要求读取所有文件中的整数，进行升序排序后，输出到一个新的文件中，输出的数据格式为每行两个整数，第一个数字为第二个整数的排序位次，第二个整数为原待排列的整数。下面是参考示例：

输入文件 1 的样例如下：

```
33
37
12
40
```

输入文件 2 的样例如下：

```
4
16
39
5
```

输入文件 3 的样例如下：

```
1
45
25
```

根据输入文件 1、2 和 3 得到的输出文件如下：

```
1 1
2 4
3 5
4 12
5 16
6 25
7 33
```

8 37

9 39

10 40

11 45

3. 对给定的表格进行信息挖掘

对于给出的一个 **child-parent** 的表格，要求挖掘其中的父子辈关系，给出祖孙辈关系的表格。下面是参考示例：

输入文件内容如下：

child	parent
Steven	Lucy
Steven	Jack
Jone	Lucy
Jone	Jack
Lucy	Mary
Lucy	Frank
Jack	Alice
Jack	Jesse
David	Alice
David	Jesse
Philip	David
Philip	Alma
Mark	David
Mark	Alma

输出文件内容如下：

grandchild	grandparent
Steven	Alice
Steven	Jesse
Jone	Alice
Jone	Jesse
Steven	Mary

Steven	Frank
Jone	Mary
Jone	Frank
Philip	Alice
Philip	Jesse
Mark	Alice
Mark	Jesse

三、 开发环境：

- Lenovo Legion R9000P2021H
- VMware-workstation-full-17.0.0
- Bbuntu Kylin 16.04
- Eclipse-4.7.0-linux.gtk.x86_64
- 已配置完成的 Hadoop 伪分布式环境

四、 实验内容：

1. 编程实现文件合并和去重操作

对于两个输入文件，即文件A 和文件B，请编写MapReduce 程序，对两个文件进行合并，并剔除其中重复的内容，得到一个新的输出文件C。

（1）Java代码

①源代码（因篇幅较长，请见下一页）

```
MergeFiles5.java

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.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

public class MergeFiles5 {

    /**
     * @param args
     * 对A,B两个文件进行合并，并删除其中重复的内容，得到一个新的输出文件C
     */
    //在这重载map函数，直接将输入中的value复制到输出数据的key上 注意在map方法中要抛出异常：throws IOException
    public static class Map extends Mapper<Object, Text, Text, Text> {
        private static Text text = new Text();
        public void map(Object key, Text value, Context context) throws IOException, InterruptedException {

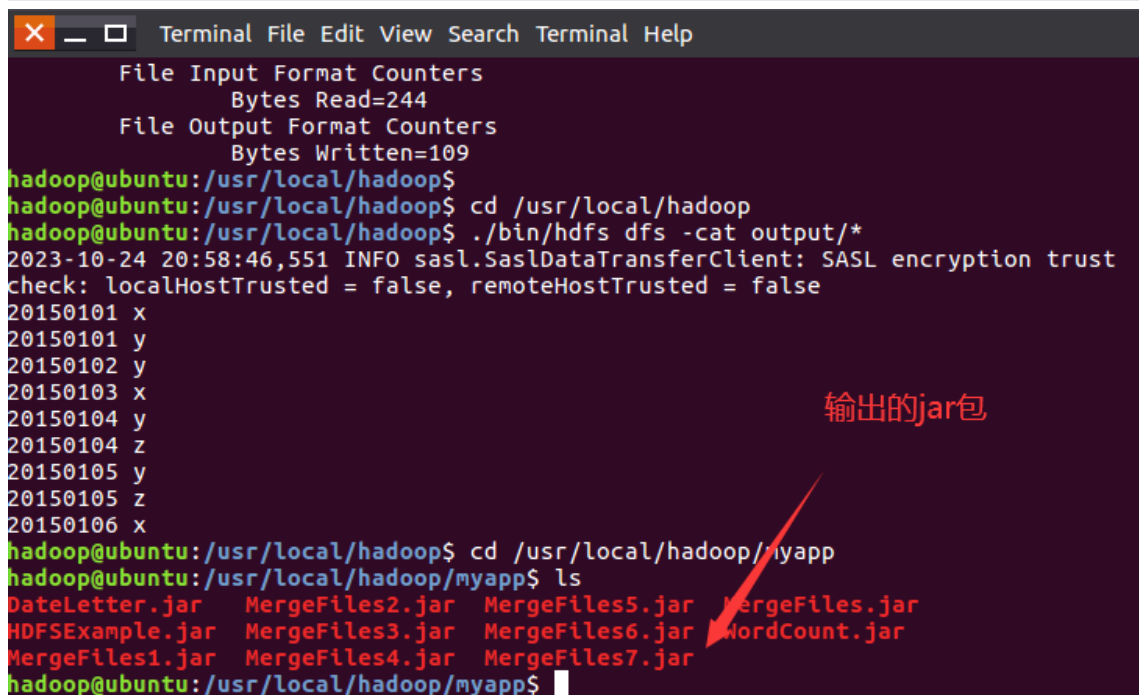
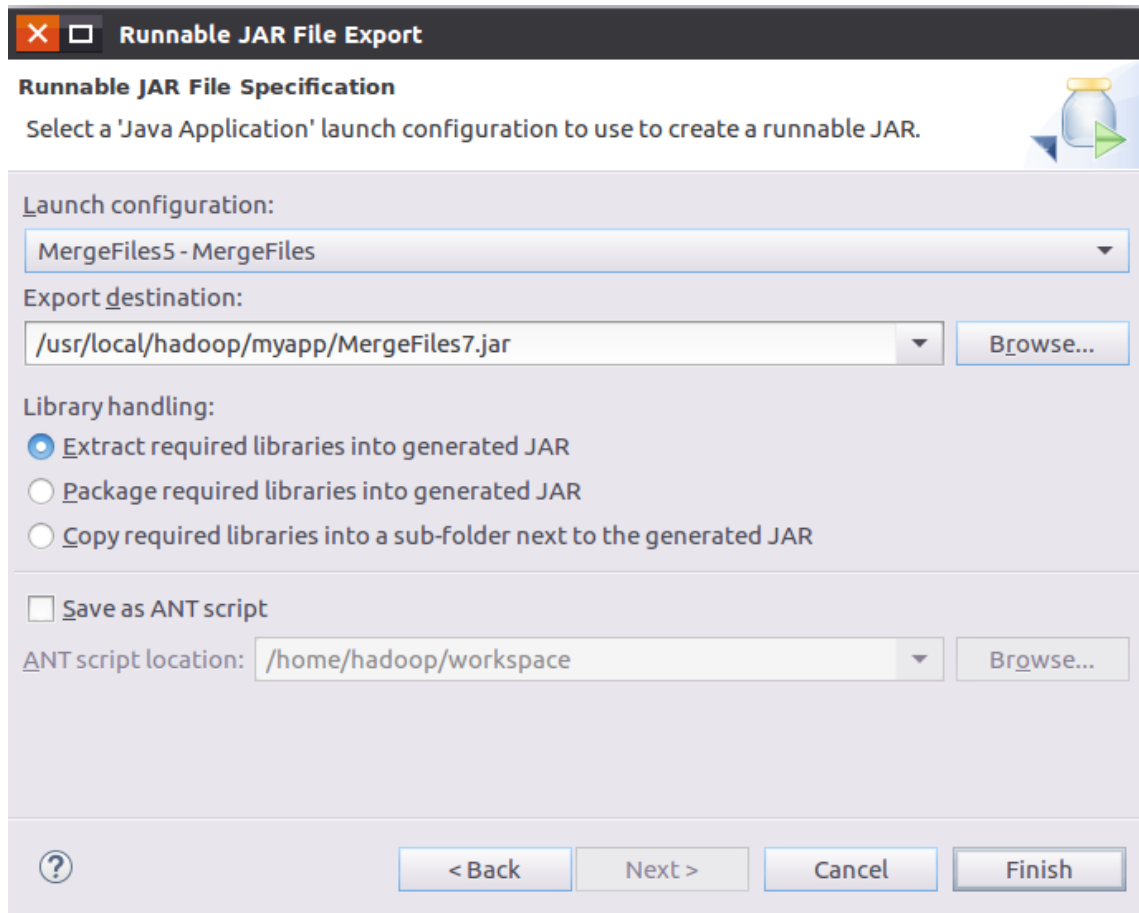
            text = value;
            context.write(text, new Text(""));
        }
    }

    //在这重载reduce函数，直接将输入中的key复制到输出数据的key上 注意在reduce方法上要抛出异常：throws IOException
    public static class Reduce extends Reducer<Text, Text, Text, Text> {
        public void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {
            context.write(key, new Text(""));
        }
    }

    public static void main(String[] args) throws Exception{

        // TODO Auto-generated method stub
        Configuration conf = new Configuration();
        conf.set("fs.default.name", "hdfs://localhost:9000");
        String[] otherArgs = new String[]{"user/hadoop/input", "user/hadoop/output"}; /* 这里 */
        if (otherArgs.length != 2) {
            System.err.println("Usage: wordcount <in> <out>");
            System.exit(2);
        }
        Job job = Job.getInstance(conf, "Merge and duplicate removal");
        job.setJarByClass(MergeFiles5.class);
        job.setMapperClass(Map.class);
        job.setCombinerClass(Reduce.class);
        job.setReducerClass(Reduce.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);
        FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
        FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

②打包成jar包



③传入数据并运行 jar 包

```

hadoop@ubuntu: /usr/local/hadoop$ ./bin/hdfs dfs -rm -r output
Deleted output

hadoop@ubuntu: /usr/local/hadoop$
hadoop@ubuntu: /usr/local/hadoop$ cd /usr/local/hadoop
hadoop@ubuntu: /usr/local/hadoop$ ./bin/hdfs dfs -mkdir input
hadoop@ubuntu: /usr/local/hadoop$ cd /usr/local/hadoop
hadoop@ubuntu: /usr/local/hadoop$ ./bin/hdfs dfs -put ./wordfile1.txt input
2023-10-24 22:59:26,116 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false
, remoteHostTrusted = false
hadoop@ubuntu: /usr/local/hadoop$ ./bin/hdfs dfs -put ./wordfile2.txt input
2023-10-24 22:59:28,269 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false
, remoteHostTrusted = false
hadoop@ubuntu: /usr/local/hadoop$ cd /usr/local/hadoop
hadoop@ubuntu: /usr/local/hadoop$ ./bin/hdfs dfs -rm -r /user/hadoop/output
rm: `user/hadoop/output': No such file or directory
hadoop@ubuntu: /usr/local/hadoop$ cd /usr/local/hadoop
hadoop@ubuntu: /usr/local/hadoop$ ./bin/hadoop jar ./myapp/MergeFiles3.jar MergeFiles input output
路径: hdfs://localhost:9000/user/hadoop/input/wordfile1.txt 文件大小: 139 权限: rw-r--r-- 内容: 2023-10-
24 22:59:55,280 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remote
HostTrusted = false
20150101 x
20150102 y
20150103 x
20150104 y
20150105 z
20150106 x
20214511 cq
20214511 quzz
20215353 cquse
20214511 cquse
cquse 20214511

```

传入数据

在input文件夹放入数据

运行jar包

```

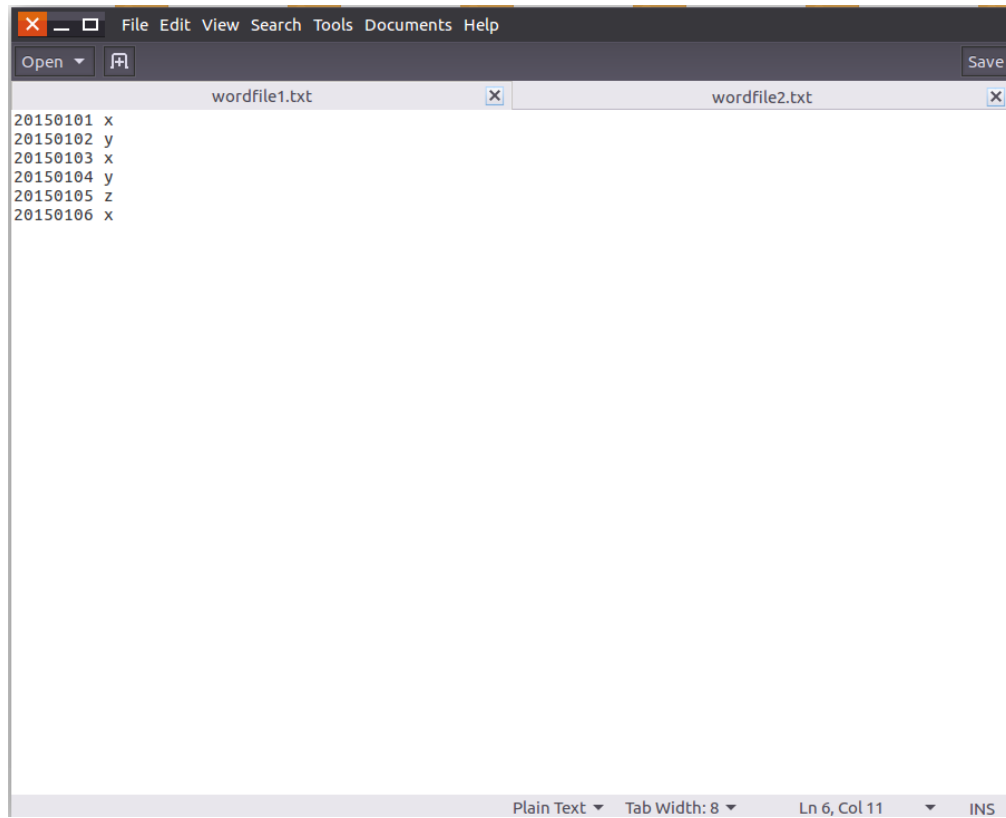
hadoop@ubuntu: /usr/local/hadoop$ ./bin/hdfs dfs -rm -r input
Deleted input
hadoop@ubuntu: /usr/local/hadoop$ ./bin/hdfs dfs -rm -r output
rm: `output': No such file or directory
hadoop@ubuntu: /usr/local/hadoop$ cd /usr/local/hadoop
hadoop@ubuntu: /usr/local/hadoop$ ./bin/hdfs dfs -put ./wordfile1.txt input
2023-10-24 22:05:55,800 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
hadoop@ubuntu: /usr/local/hadoop$ ./bin/hdfs dfs -put ./wordfile2.txt input
put: `input': File exists
hadoop@ubuntu: /usr/local/hadoop$ cd /usr/local/hadoop
hadoop@ubuntu: /usr/local/hadoop$ ./bin/hadoop jar ./myapp/MergeFiles7.jar MergeFiles5 input output
2023-10-24 22:06:09,199 INFO Configuration.deprecation: fs.default.name is deprecated. Instead, use fs.defaultFS
2023-10-24 22:06:09,746 INFO Impl.MetricsConfig: loaded properties from hadoop-metrics2.properties
2023-10-24 22:06:09,801 INFO Impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2023-10-24 22:06:09,801 INFO Impl.MetricsSystemImpl: JobTracker metrics system started
2023-10-24 22:06:10,056 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to
remedy this.
2023-10-24 22:06:10,259 INFO Input.FileInputFormat: Total input files to process : 1
2023-10-24 22:06:10,303 INFO mapreduce.JobSubmitter: number of splits:1
2023-10-24 22:06:10,330 INFO Configuration.deprecation: fs.default.name is deprecated. Instead, use fs.defaultFS
2023-10-24 22:06:10,384 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1765511474_0001
2023-10-24 22:06:10,385 INFO mapreduce.JobSubmitter: Executing with tokens: []
2023-10-24 22:06:10,447 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2023-10-24 22:06:10,448 INFO mapreduce.Job: Running job: job_local1765511474_0001
2023-10-24 22:06:10,450 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2023-10-24 22:06:10,453 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2023-10-24 22:06:10,453 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2023-10-24 22:06:10,454 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
2023-10-24 22:06:10,482 INFO mapred.LocalJobRunner: Waiting for map tasks
2023-10-24 22:06:10,495 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2023-10-24 22:06:10,495 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2023-10-24 22:06:10,506 INFO mapred.Task: Using ResourceCalculatorProcessFree: [ ]
2023-10-24 22:06:10,507 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/user/hadoop/input:0+139
2023-10-24 22:06:10,555 INFO mapred.MapTask: (EQUATOR) 0 kvl 26214396(104857584)
2023-10-24 22:06:10,555 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
2023-10-24 22:06:10,555 INFO mapred.MapTask: soft limit at 83886080
2023-10-24 22:06:10,555 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
2023-10-24 22:06:10,555 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
2023-10-24 22:06:10,559 INFO mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputBuffer
2023-10-24 22:06:10,572 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2023-10-24 22:06:10,610 INFO mapred.LocalJobRunner:
2023-10-24 22:06:10,617 INFO mapred.MapTask: Starting flush of map output
2023-10-24 22:06:10,617 INFO mapred.MapTask: Spilling map output
2023-10-24 22:06:10,617 INFO mapred.MapTask: bufstart = 0; bufend = 150; bufvoid = 104857600

```


(2) 实验结果

①使用原始数据

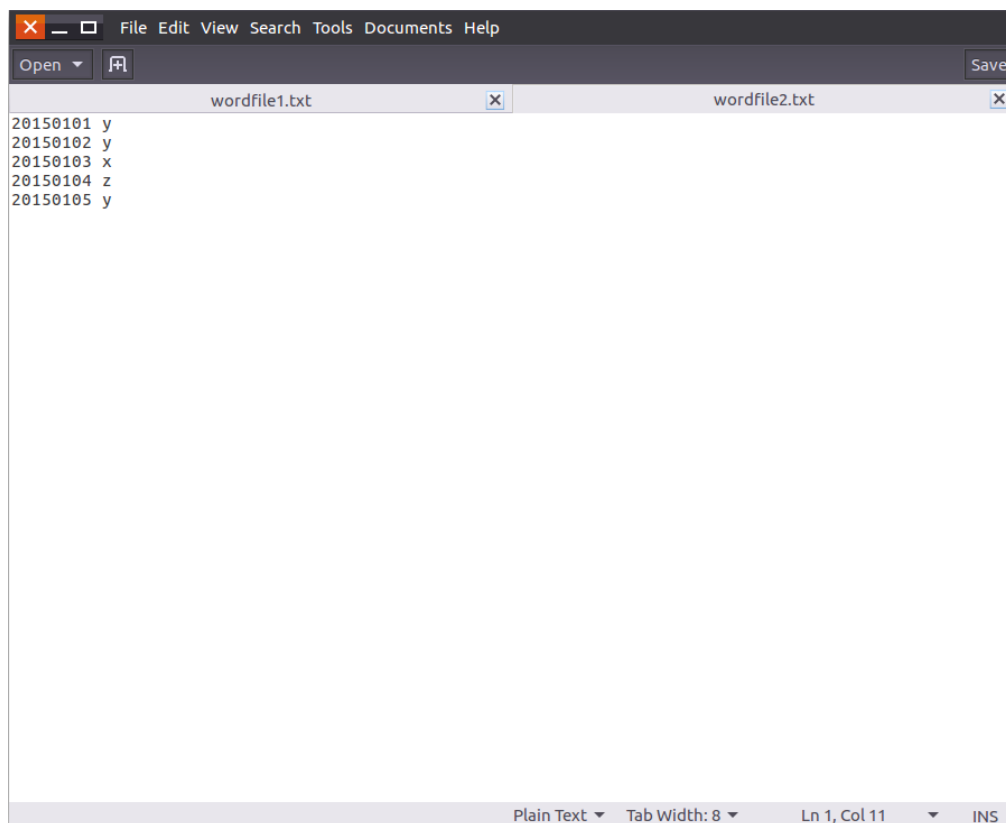
数据：



A screenshot of a text editor window with a dark theme. The menu bar includes File, Edit, View, Search, Tools, Documents, and Help. The toolbar has an 'Open' button and a 'Save' button. Two tabs are open: 'wordfile1.txt' and 'wordfile2.txt'. The 'wordfile1.txt' tab is active and contains the following text:

```
20150101 x
20150102 y
20150103 x
20150104 y
20150105 z
20150106 x
```

The status bar at the bottom indicates 'Plain Text', 'Tab Width: 8', 'Ln 6, Col 11', and 'INS'.

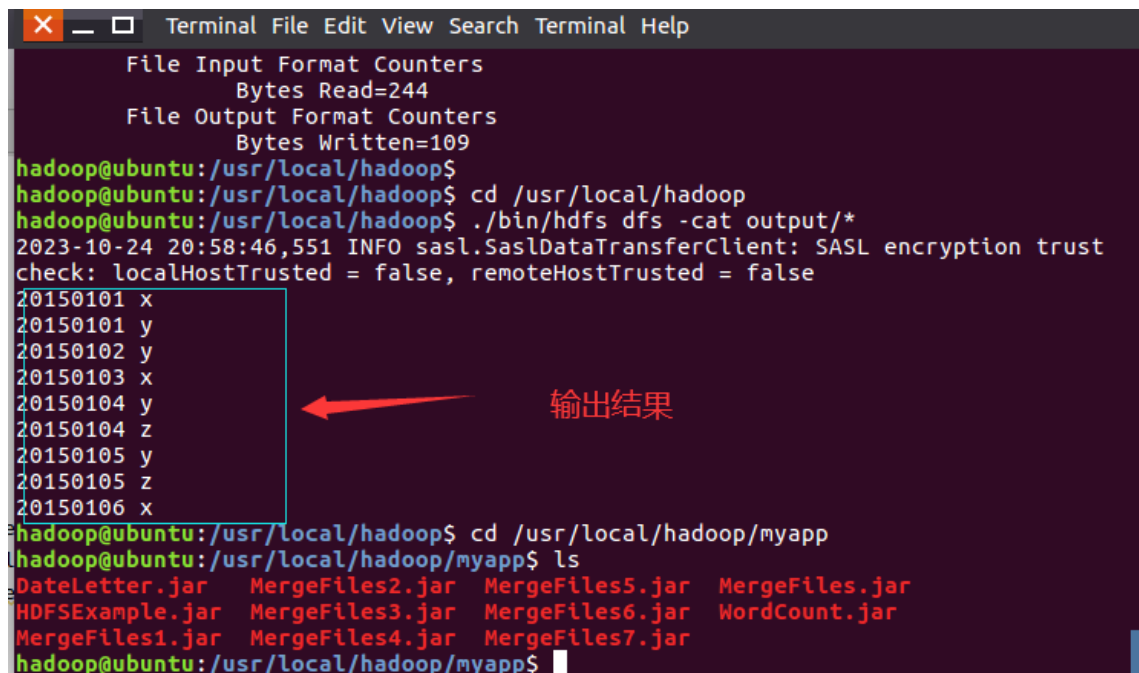


A screenshot of a text editor window with a dark theme, similar to the one above. The menu bar and toolbar are the same. Two tabs are open: 'wordfile1.txt' and 'wordfile2.txt'. The 'wordfile1.txt' tab is active and contains the following text:

```
20150101 y
20150102 y
20150103 x
20150104 z
20150105 y
```

The status bar at the bottom indicates 'Plain Text', 'Tab Width: 8', 'Ln 1, Col 11', and 'INS'.

结果:

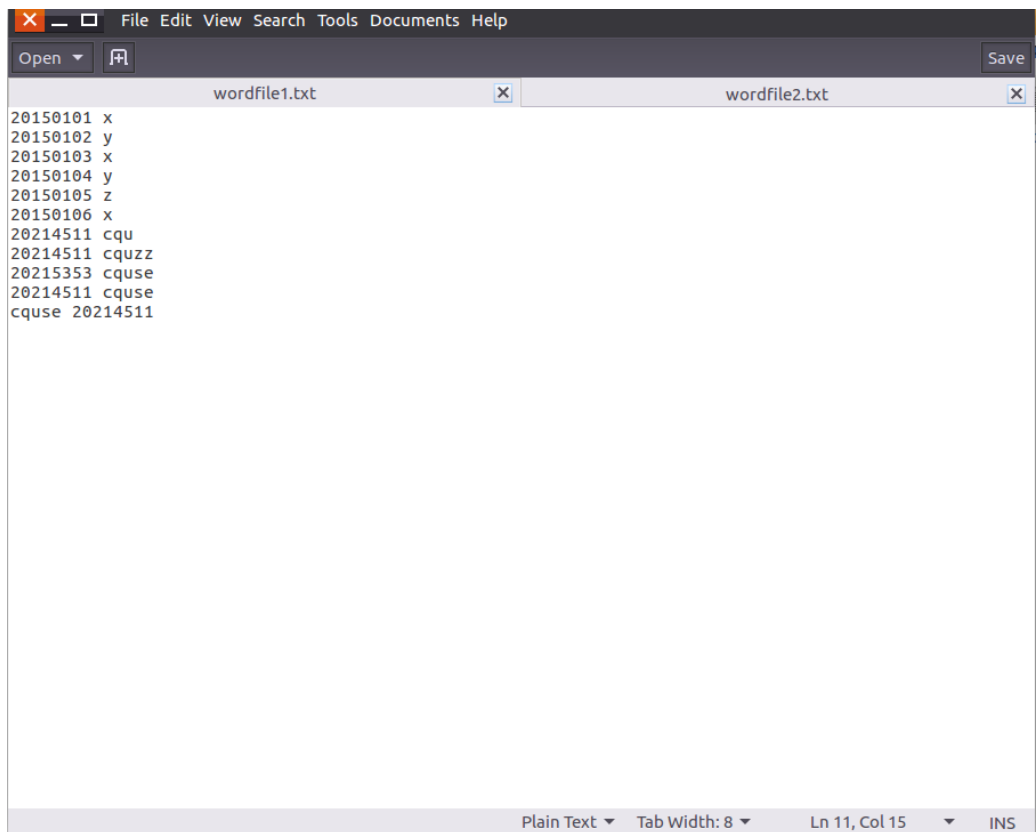


```
File Input Format Counters
  Bytes Read=244
File Output Format Counters
  Bytes Written=109
hadoop@ubuntu:/usr/local/hadoop$
hadoop@ubuntu:/usr/local/hadoop$ cd /usr/local/hadoop
hadoop@ubuntu:/usr/local/hadoop$ ./bin/hdfs dfs -cat output/*
2023-10-24 20:58:46,551 INFO sasl.SaslDataTransferClient: SASL encryption trust
check: localhostTrusted = false, remoteHostTrusted = false
20150101 x
20150101 y
20150102 y
20150103 x
20150104 y
20150104 z
20150105 y
20150105 z
20150106 x
hadoop@ubuntu:/usr/local/hadoop$ cd /usr/local/hadoop/myapp
hadoop@ubuntu:/usr/local/hadoop/myapp$ ls
DateLetter.jar  MergeFiles2.jar  MergeFiles5.jar  MergeFiles.jar
HDFSExample.jar  MergeFiles3.jar  MergeFiles6.jar  WordCount.jar
MergeFiles1.jar  MergeFiles4.jar  MergeFiles7.jar
hadoop@ubuntu:/usr/local/hadoop/myapp$
```

输出结果

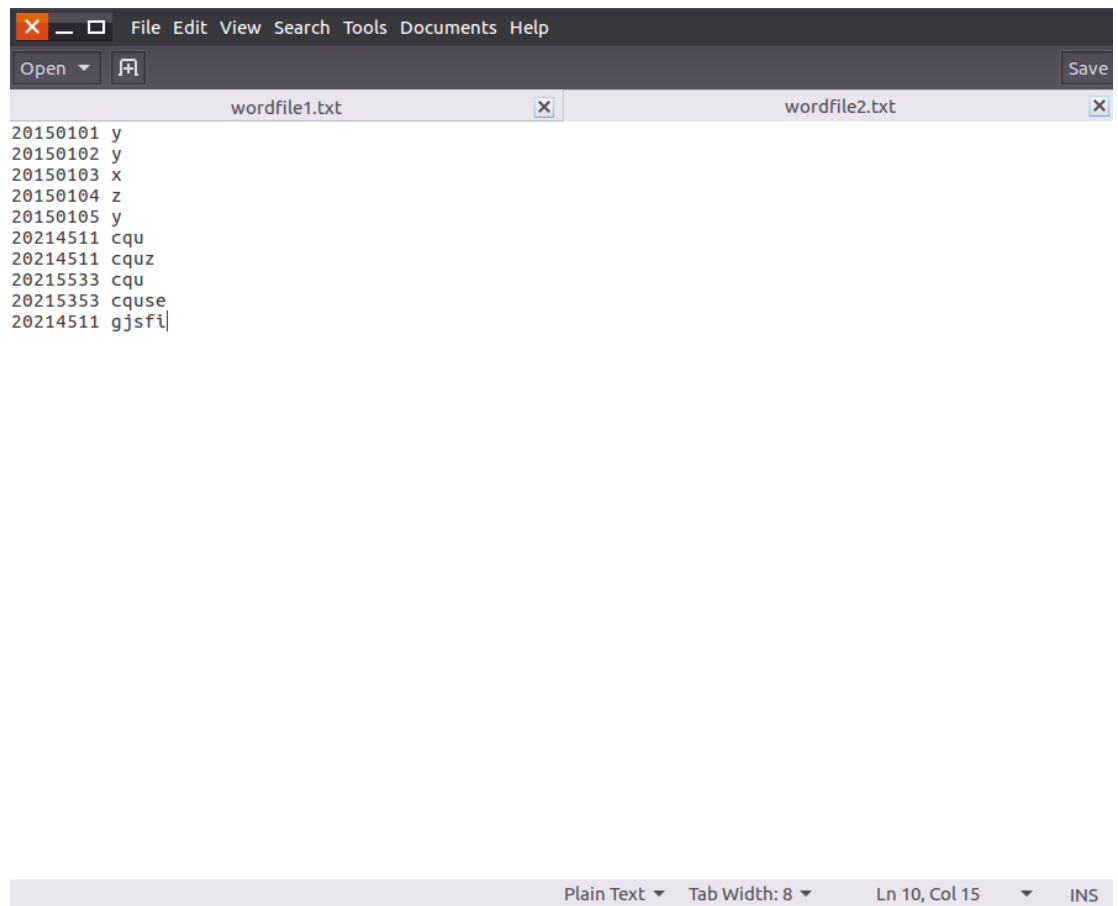
②对原始数据进行添加后

数据:

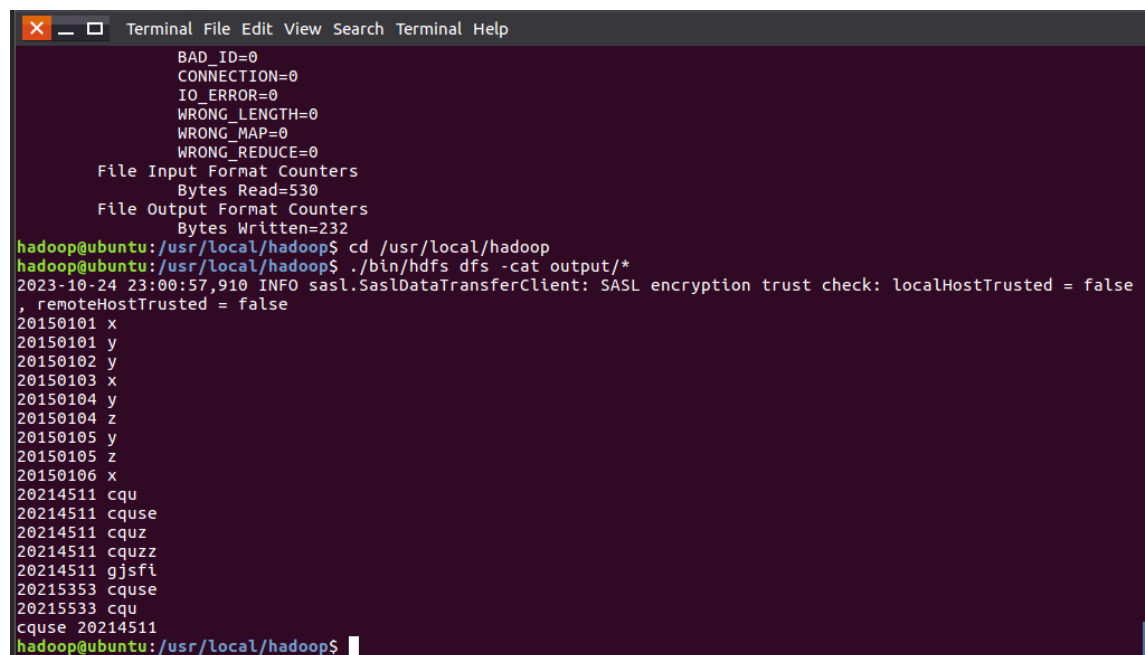


```
wordfile1.txt
20150101 x
20150102 y
20150103 x
20150104 y
20150105 z
20150106 x
20214511 cqu
20214511 cquzz
20215353 cquse
20214511 cquse
cquse 20214511

wordfile2.txt
```



结果:



2. 编写程序实现对输入文件的排序

现在有多个输入文件，每个文件中的每行内容均为一个整数。要求读取所有文件中的整数，进行升序排序后，输出到一个新的文件中，输出的数据格式为每行两个整数，第一个数字为第二个整数的排序位次，第二个整数为原待排列的整数。

(1) Java代码

①源代码（因篇幅较长，请见下一页）

```
MergeSort.java

import java.io.IOException;

import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Partitioner;

import java.util.Iterator;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

public class MergeSort {

    /**
     * @param args
     * 输入多个文件，每个文件中的每行内容均为一个整数
     * 输出到一个新的文件中，输出的数据格式为每行两个整数，第一个数字为第二个整数的排序位次，第二个整数为原待排
     */
    //map函数读取输入中的value，将其转化成IntWritable类型，最后作为输出key
    public static class Map extends Mapper<Object, Text, IntWritable, IntWritable>{

        private static IntWritable data = new IntWritable();
        public void map(Object key, Text value, Context context) throws IOException, Interruption {
            /******* Begin *****/
            String line = value.toString();
            data.set(Integer.parseInt(line));
            context.write(data, new IntWritable(1));
            /******* End *****/
        }
    }

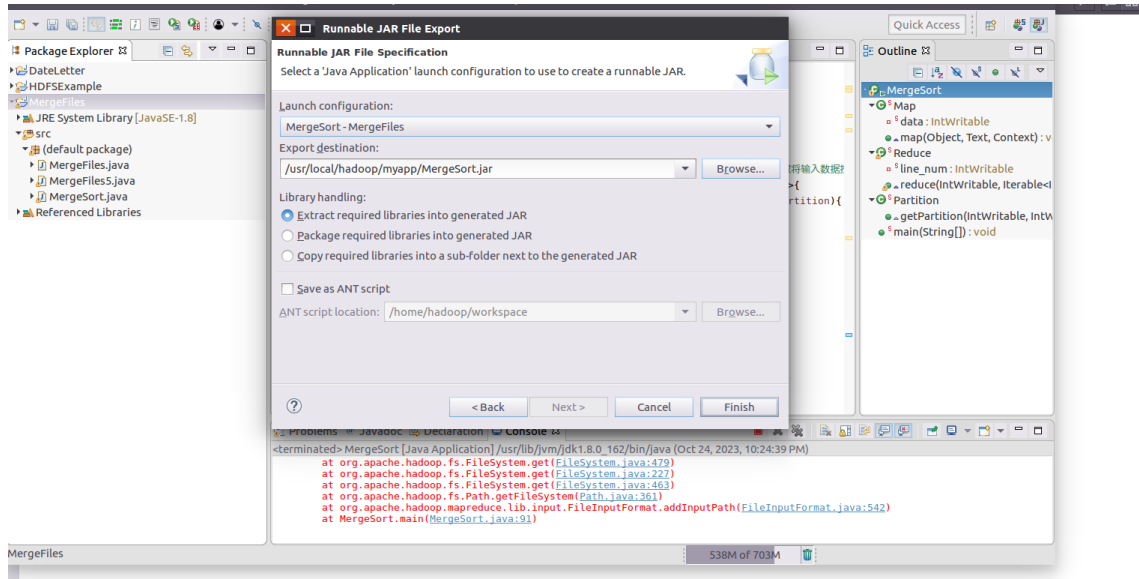
    //reduce函数将map输入的key复制到输出的value上，然后根据输入的value-list中元素的个数决定key的输出
    public static class Reduce extends Reducer<IntWritable, IntWritable, IntWritable, IntWritable> {
        private static IntWritable line_num = new IntWritable(1);

        public void reduce(IntWritable key, Iterable<IntWritable> values, Context context) throws IOException, Interruption {
            /******* Begin *****/
            for(IntWritable num : values) {
                context.write(line_num, key);
                line_num = new IntWritable(line_num.get() + 1);
            }
            /******* End *****/
        }
    }

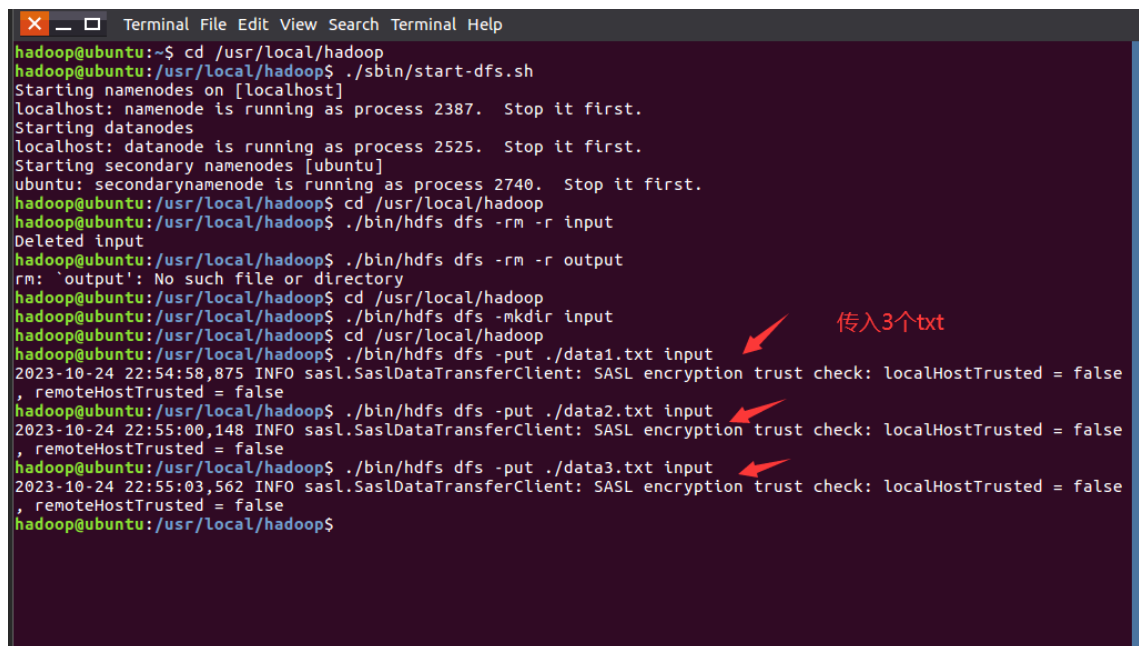
    //自定义Partition函数，此函数根据输入数据的最大值和MapReduce框架中Partition的数量获取待输入数据
    public static class Partition extends Partitioner<IntWritable, IntWritable>{
        public int getPartition(IntWritable key, IntWritable value, int num_Partition){
            /******* Begin *****/
            int Maxnumber = 65223; //int型的最大数值
            int bound = Maxnumber / num_Partition + 1;
            int Keynumber = key.get();
            for(int i = 0; i < num_Partition; i++){
                if(Keynumber < bound * i && Keynumber >= bound * (i - 1)) {
                    return i - 1;
                }
            }
            return -1;
            /******* End *****/
        }
    }

    public static void main(String[] args) throws Exception{
        // TODO Auto-generated method stub
        Configuration conf = new Configuration();
        conf.set("fs.default.name", "hdfs://localhost:9000");
        String[] otherArgs = new String[]{"user/hadoop/input", "user/hadoop/output"};
        if (otherArgs.length != 2) {
            System.err.println("Usage: wordcount <in> <out>");
            System.exit(2);
        }
        Job job = Job.getInstance(conf, "Merge and Sort");
        job.setJarByClass(MergeSort.class);
        job.setMapperClass(Map.class);
        job.setReducerClass(Reduce.class);
        job.setPartitionerClass(Partition.class);
        job.setOutputKeyClass(IntWritable.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
        FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

②打包成jar包



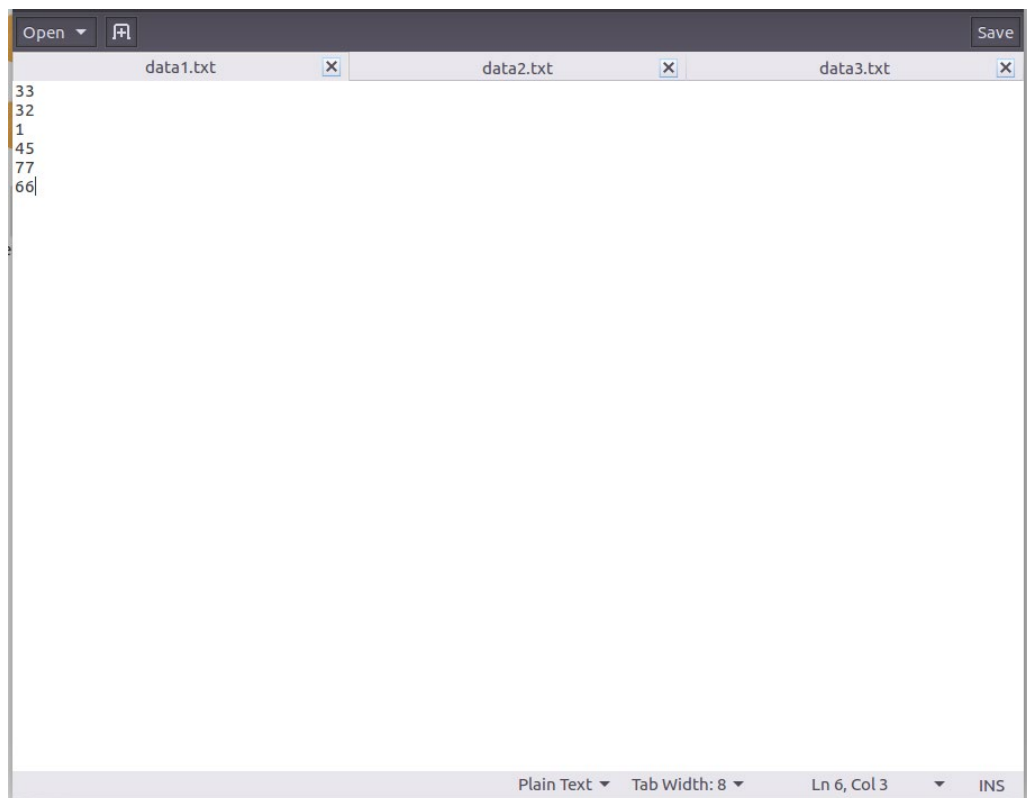
③传入数据并运行jar包



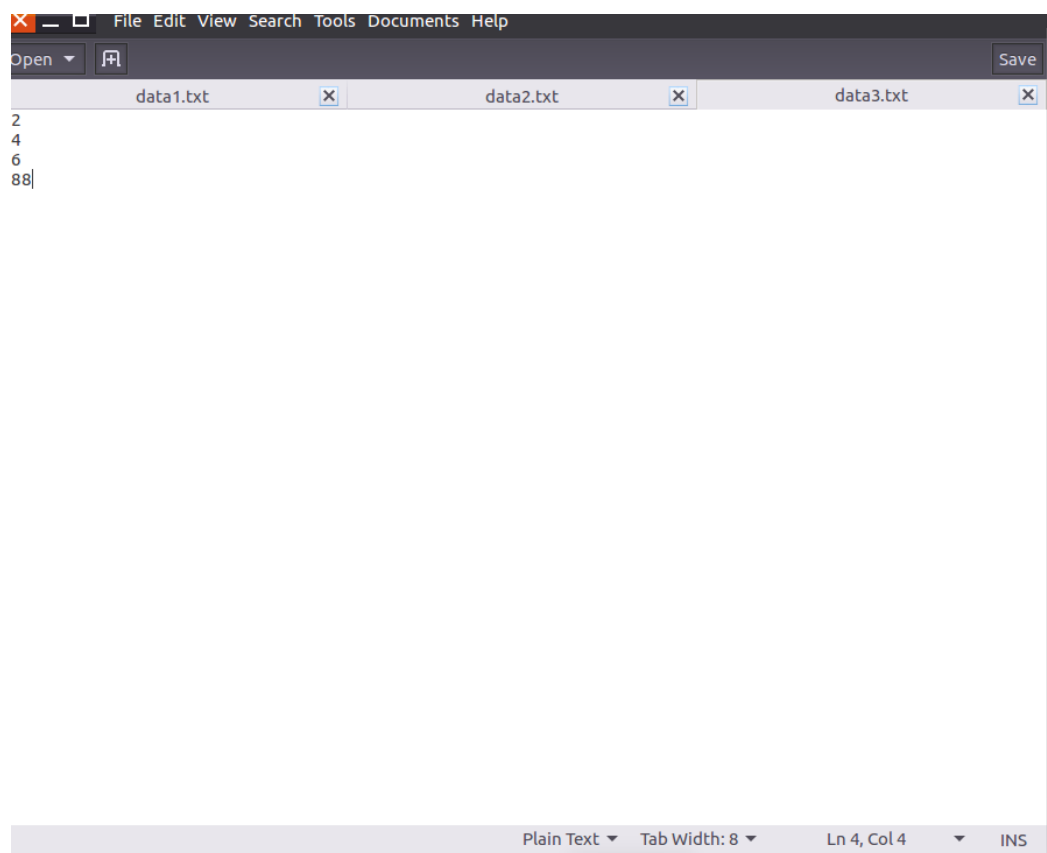
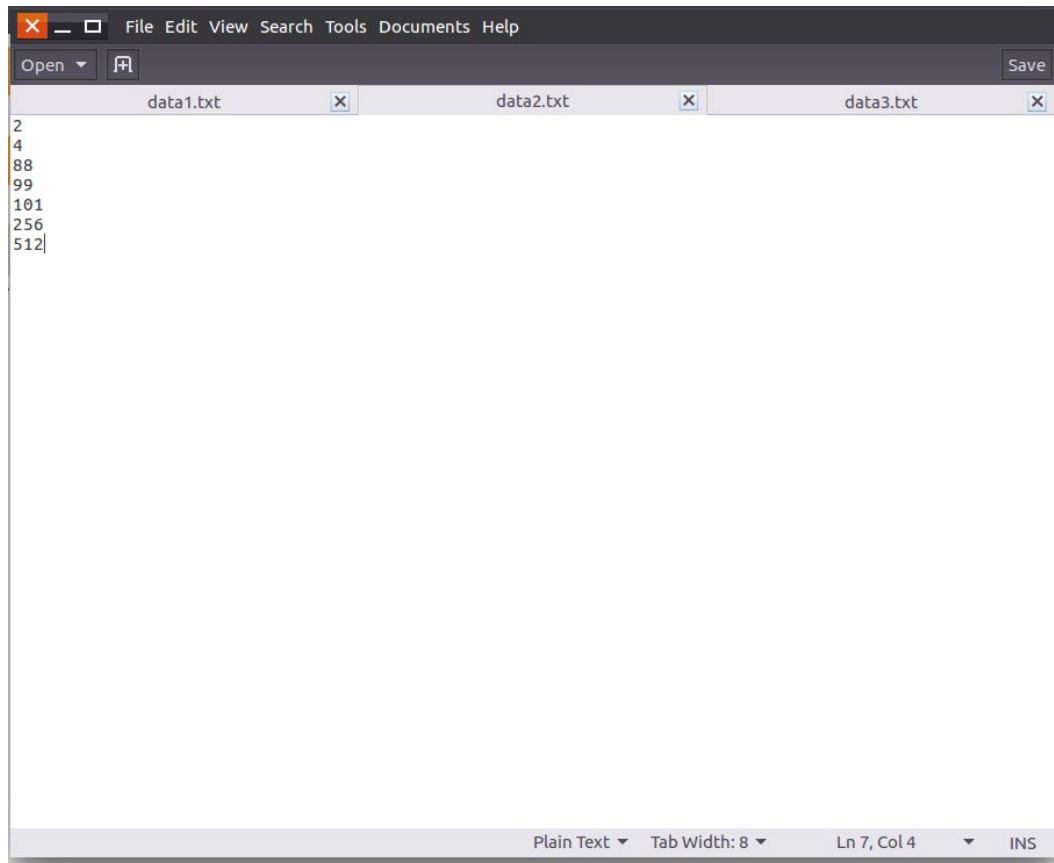
```
Terminal File Edit View Search Terminal Help
hadoop@ubuntu:/usr/local/hadoop$ ./bin/hdfs dfs -mkdir input
hadoop@ubuntu:/usr/local/hadoop$ cd /usr/local/hadoop
hadoop@ubuntu:/usr/local/hadoop$ ./bin/hdfs dfs -put ./data1.txt input
2023-10-24 22:54:58,875 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false
, remoteHostTrusted = false
hadoop@ubuntu:/usr/local/hadoop$ ./bin/hdfs dfs -put ./data2.txt input
2023-10-24 22:55:00,148 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false
, remoteHostTrusted = false
hadoop@ubuntu:/usr/local/hadoop$ ./bin/hdfs dfs -put ./data3.txt input
2023-10-24 22:55:03,562 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false
, remoteHostTrusted = false
hadoop@ubuntu:/usr/local/hadoop$ cd /usr/local/hadoop
hadoop@ubuntu:/usr/local/hadoop$ ./bin/hadoop jar ./myapp/MergeSort.jar MergeSort input output
2023-10-24 22:56:02,549 INFO Configuration.deprecation: fs.default.name is deprecated. Instead, use fs.defaultF
S
2023-10-24 22:56:03,015 INFO impl.MetricsConfig: loaded properties from hadoop-metrics2.properties
2023-10-24 22:56:03,049 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2023-10-24 22:56:03,049 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2023-10-24 22:56:03,292 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. I
mplement the Tool interface and execute your application with ToolRunner to remedy this.
2023-10-24 22:56:03,534 INFO input.FileInputFormat: Total input files to process : 3
2023-10-24 22:56:03,557 INFO mapreduce.JobSubmitter: number of splits:3
2023-10-24 22:56:03,584 INFO Configuration.deprecation: fs.default.name is deprecated. Instead, use fs.defaultF
S
2023-10-24 22:56:03,651 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1948511003_0001
2023-10-24 22:56:03,652 INFO mapreduce.JobSubmitter: Executing with tokens: []
2023-10-24 22:56:03,724 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2023-10-24 22:56:03,724 INFO mapreduce.Job: Running job: job_local1948511003_0001
2023-10-24 22:56:03,727 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2023-10-24 22:56:03,732 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2023-10-24 22:56:03,733 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders un
der output directory:false ignore cleanup failures: false
```

(2) 实验结果

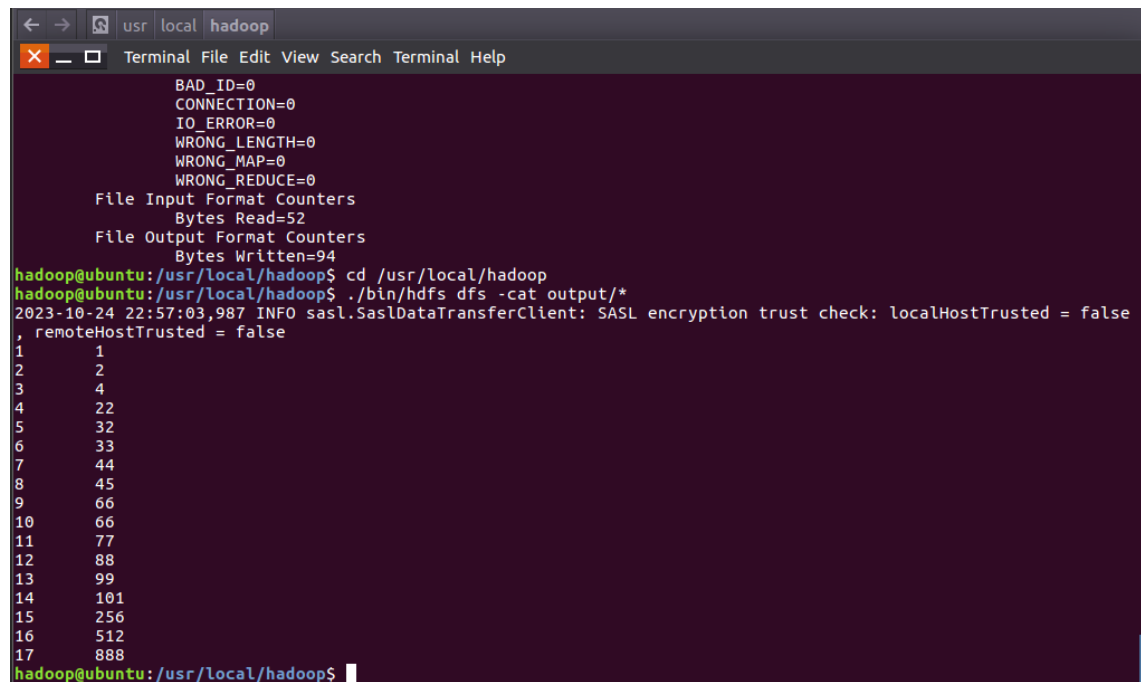
①数据（使用自己扩展之后的数据）



```
Open  data1.txt  data2.txt  data3.txt  Save
33
32
1
45
77
66
Plain Text  Tab Width: 8  Ln 6, Col 3  INS
```



②结果



```
usr local hadoop
Terminal File Edit View Search Terminal Help
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=52
File Output Format Counters
  Bytes Written=94
hadoop@ubuntu:/usr/local/hadoop$ cd /usr/local/hadoop
hadoop@ubuntu:/usr/local/hadoop$ ./bin/hdfs dfs -cat output/*
2023-10-24 22:57:03,987 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false
, remoteHostTrusted = false
1      1
2      2
3      4
4     22
5     32
6     33
7     44
8     45
9     66
10    66
11    77
12    88
13    99
14   101
15   256
16   512
17   888
hadoop@ubuntu:/usr/local/hadoop$
```

3. 对给定的表格进行信息挖掘

对于给出的一个child-parent 的表格，要求挖掘其中的父子辈关系，给出祖孙辈关系的表格。

(1) Java代码

①源代码（因篇幅较长，请见下一页）

```
findgrand.java
import java.io.IOException;
import java.util.*;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

public class findgrand{
    public static int time = 0;

    /**
     * @param args
     * 输入一个child-parent的表格
     * 输出一个体现grandchild-grandparent关系的表格
     */
    //Map将输入文件按照空格分割成child和parent, 然后正序输出一次作为右表, 反序输出一次作为左表, 需要注意!
    public static class Map extends Mapper<Object, Text, Text, Text>{
        public void map(Object key, Text value, Context context) throws IOException, InterruptedException {
            /***** Begin *****/
            String line = value.toString();
            String[] childAndParent = line.split(" ");
            List<String> list = new ArrayList<>();
            for (String childOrParent : childAndParent) {
                if (!"".equals(childOrParent)) {
                    list.add(childOrParent);
                }
            }
            if (!"child".equals(list.get(0))) {
                String childName = list.get(0);
                String parentName = list.get(1);
                String relationType = "1";
                context.write(new Text(parentName), new Text(relationType + "+"
                    + childName + "+" + parentName));
                relationType = "2";
                context.write(new Text(childName), new Text(relationType + "+"
                    + childName + "+" + parentName));
            }
            /***** End *****/
        }
    }

    public static class Reduce extends Reducer<Text, Text, Text, Text>{
        public void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {
            /***** Begin *****/

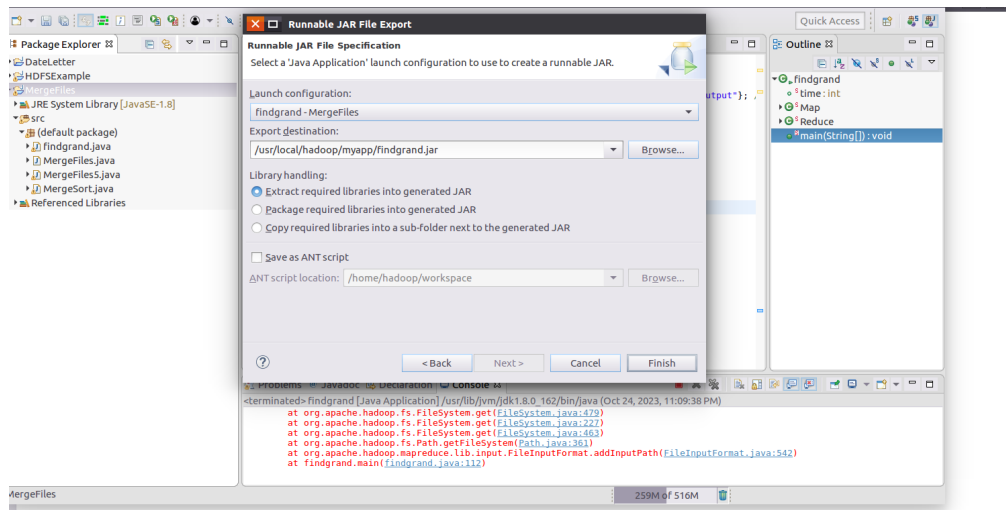
            //输出表头
            if (time == 0) {
                context.write(new Text("grand_child"), new Text("grand_parent"));
                time++;
            }

            //获取value-list中value的child
            List<String> grandChild = new ArrayList<>();
            //获取value-list中value的parent
            List<String> grandParent = new ArrayList<>();
            //左表, 取出child放入grand_child
            for (Text text : values) {
                String s = text.toString();
                String[] relation = s.split("\\+");
                String relationType = relation[0];
                String childName = relation[1];
                String parentName = relation[2];
                if (!"1".equals(relationType)) {
                    grandChild.add(childName);
                } else {
                    grandParent.add(parentName);
                }
            }

            //右表, 取出parent放入grand_parent
            int grandParentNum = grandParent.size();
            int grandChildNum = grandChild.size();
            if (grandParentNum != 0 && grandChildNum != 0) {
                for (int m = 0; m < grandChildNum; m++) {
                    for (int n = 0; n < grandParentNum; n++) {
                        //输出结果
                        context.write(new Text(grandChild.get(m)), new Text(
                            grandParent.get(n)));
                    }
                }
            }
            /***** End *****/
        }
    }

    public static void main(String[] args) throws Exception{
        // TODO Auto-generated method stub
        Configuration conf = new Configuration();
        conf.set("fs.default.name", "hdfs://localhost:9000");
        String[] otherArgs = new String[]{"user/hadoop/input", "user/hadoop/output"};
        if (otherArgs.length != 2) {
            System.err.println("Usage: wordcount <in> <out>");
            System.exit(2);
        }
        Job job = Job.getInstance(conf, "Single table join");
        job.setJarByClass(findgrand.class);
        job.setMapperClass(Map.class);
        job.setReducerClass(Reduce.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);
        FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
        FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

②打包成jar包



③传入数据并运行jar包

```
hadoop@ubuntu: /usr/local/hadoop$ ./bin/hdfs dfs -cat output/*
2023-10-24 23:00:57,910 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false
, remoteHostTrusted = false
20150101 x
20150101 y
20150102 y
20150103 x
20150104 y
20150104 z
20150105 y
20150105 z
20150106 x
20214511 cqu
20214511 cquse
20214511 cquz
20214511 cquzz
20214511 gjsfi
20215353 cquse
20215533 cqu
cquse 20214511
hadoop@ubuntu: /usr/local/hadoop$ cd /usr/local/hadoop
hadoop@ubuntu: /usr/local/hadoop$ ./bin/hdfs dfs -rm -r input
Deleted input

hadoop@ubuntu: /usr/local/hadoop$ ./bin/hdfs dfs -rm -r output
Deleted output
hadoop@ubuntu: /usr/local/hadoop$ cd /usr/local/hadoop
hadoop@ubuntu: /usr/local/hadoop$ ./bin/hdfs dfs -mkdir input
hadoop@ubuntu: /usr/local/hadoop$ ./bin/hdfs dfs -put ./parent-son.txt input
2023-10-24 23:11:53,139 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false
, remoteHostTrusted = false
hadoop@ubuntu: /usr/local/hadoop$

hadoop@ubuntu: /usr/local/hadoop$ ./bin/hdfs dfs -rm -r input
Deleted input

hadoop@ubuntu: /usr/local/hadoop$ ./bin/hdfs dfs -rm -r output
Deleted output
hadoop@ubuntu: /usr/local/hadoop$ cd /usr/local/hadoop
hadoop@ubuntu: /usr/local/hadoop$ ./bin/hdfs dfs -mkdir input
hadoop@ubuntu: /usr/local/hadoop$ ./bin/hdfs dfs -put ./parent-son.txt input
2023-10-24 23:11:53,139 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false
, remoteHostTrusted = false
hadoop@ubuntu: /usr/local/hadoop$ cd /usr/local/hadoop
hadoop@ubuntu: /usr/local/hadoop$ ./bin/hadoop jar ./myapp/findgrand.jar findgrand input output
2023-10-24 23:16:30,421 INFO Configuration.deprecation: fs.default.name is deprecated. Instead, use fs.defaultF
S
2023-10-24 23:16:31,047 INFO impl.MetricsConfig: loaded properties from hadoop-metrics2.properties
2023-10-24 23:16:31,117 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2023-10-24 23:16:31,117 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2023-10-24 23:16:31,537 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. I
mplement the Tool interface and execute your application with ToolRunner to remedy this.
2023-10-24 23:16:31,812 INFO input.FileInputFormat: Total input files to process : 1
2023-10-24 23:16:31,829 INFO mapreduce.JobSubmitter: number of splits:1
2023-10-24 23:16:31,854 INFO Configuration.deprecation: fs.default.name is deprecated. Instead, use fs.defaultF
S
2023-10-24 23:16:31,910 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1365231761_0001
2023-10-24 23:16:31,911 INFO mapreduce.JobSubmitter: Executing with tokens: []
2023-10-24 23:16:31,975 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2023-10-24 23:16:31,975 INFO mapreduce.Job: Running job: job_local1365231761_0001
2023-10-24 23:16:31,978 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2023-10-24 23:16:31,982 INFO output.FileOutputCommitter: File OutputCommitter Algorithm version is 2
2023-10-24 23:16:31,982 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders un
der output directory:false, ignore cleanup failures: false
2023-10-24 23:16:31,983 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.F
```

(2) 实验结果

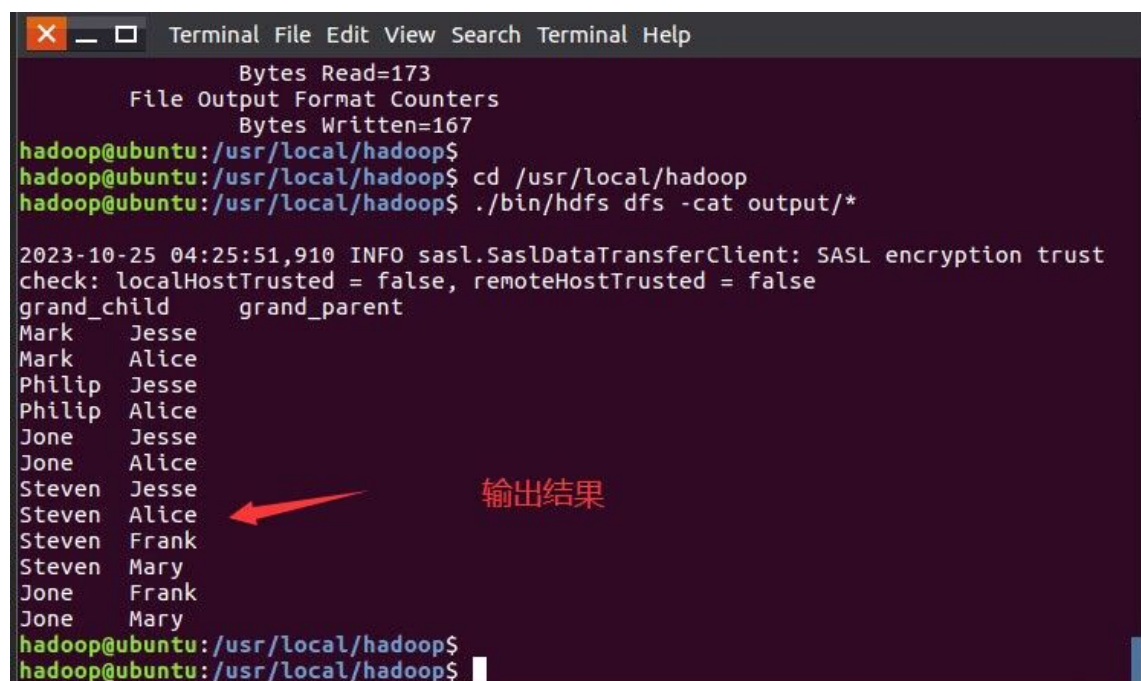
①数据



```
Open [icon] Save
child parent
Steven Lucy
Steven Jack
Jone Lucy
Jone Jack
Lucy Mary
Lucy Frank
Jack Alice
Jack Jesse
David Alice
David Jesse
Philip David
Philip Alma
Mark David
Mark Alma

Plain Text Tab Width: 8 Ln 11, Col 1 INS
```

②结果



```
Terminal File Edit View Search Terminal Help
Bytes Read=173
File Output Format Counters
Bytes Written=167
hadoop@ubuntu:/usr/local/hadoop$
hadoop@ubuntu:/usr/local/hadoop$ cd /usr/local/hadoop
hadoop@ubuntu:/usr/local/hadoop$ ./bin/hdfs dfs -cat output/*

2023-10-25 04:25:51,910 INFO sasl.SaslDataTransferClient: SASL encryption trust
check: localhostTrusted = false, remoteHostTrusted = false
grand_child    grand_parent
Mark    Jesse
Mark    Alice
Philip   Jesse
Philip   Alice
Jone     Jesse
Jone     Alice
Steven   Jesse
Steven   Alice
Steven   Frank
Steven   Mary
Jone     Frank
Jone     Mary
hadoop@ubuntu:/usr/local/hadoop$
hadoop@ubuntu:/usr/local/hadoop$
```

输出结果

五、 实验结果

1. 编程实现文件合并和去重操作

①通过编写 MapReduce 程序，成功合并两个输入文件，文件 A 和文件 B，创建一个新的输出文件 C；

②重复的内容已被有效剔除，确保文件 C 中的内容不包含任何重复数据；

③输出文件 C 中包含两文件 A 和 B 的所有唯一内容，实现了文件合并和去重操作。

2. 编写程序实现对输入文件的排序

①实验成功读取多个输入文件，每个文件中包含整数数据；

②所有整数已成功进行升序排序，结果被输出到一个新文件中；

③输出文件的格式满足要求，包含排序位次和原待排列整数的组合，提供了有序的数据。

3. 对给定的表格进行信息挖掘

①通过对给定的 child-parent 表格进行信息挖掘，成功挖掘父子辈关系；

②创建了一个新的表格，其中包含了祖孙辈关系的信息；

③挖掘过程确保了正确的父子关系，并生成了祖孙辈的关系表格，用于进一步分析和研究。

六、 问题和解决

问题 1:

Mapper 类中的键值对格式问题，Mapper 生成的键值对格式是 "K:Lucy Value: : -Steven,+Mary"，但该格式可能不够清晰和易于处理，键值对应该被明确定义，以便 Reducer 类可以正确处理。

解决方案:

重新定义 Mapper 输出的键值对格式。更好的方法是将 Mapper 的键值对格式定义为键为 "son"或"parent"，值为相应的名字。这可以通过以下方式实现：

- 键: "son", 值: "Lucy"
- 键: "parent", 值: "Steven"
- 键: "parent", 值: "Mary"

这样，Reducer 类可以更容易地识别和处理父子关系。

问题 2:

Mapper 和 Reducer 之间的协调问题，特别是在找到 grandchild 和 grandparent 关系时，需要清晰的逻辑来协调两个阶段。

解决方案:

需要在 Mapper 和 Reducer 之间建立协调逻辑，以确保正确的-grandchild 和-grandparent 关系。这可以通过在 Mapper 中将-grandchild 与-parent 建立关联，并将-grandparent 与-son 建立关联来实现。在 Reducer 中，根据这些关联建立-grandchild 和-grandparent 关系。确保这一过程的协调和顺序是正确的，以便正确识别和记录祖孙关系。