## 实验五：MapReduce初级编程实践

### 一、实验目的

* 通过实验掌握基本的MapReduce编程方法；
* 掌握用MapReduce解决一些常见的数据处理问题，包括数据去重、数据排序和数据挖掘等。

### 二、实验平台

* 操作系统：Linux（建议Ubuntu16.04）
* Hadoop版本：2.7.1

### 三、实验步骤

**（一）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。下面是输入文件和输出文件的一个样例供参考。

输入文件A的样例如下：

|  |
| --- |
| 20170101 x  20170102 y  20170103 x  20170104 y  20170105 z  20170106 x |

输入文件B的样例如下：

|  |
| --- |
| 20170101 y  20170102 y  20170103 x  20170104 z  20170105 y |

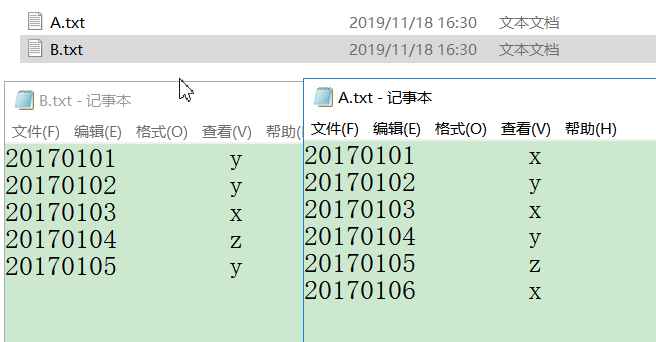
根据输入文件A和B合并得到的输出文件C的样例如下：

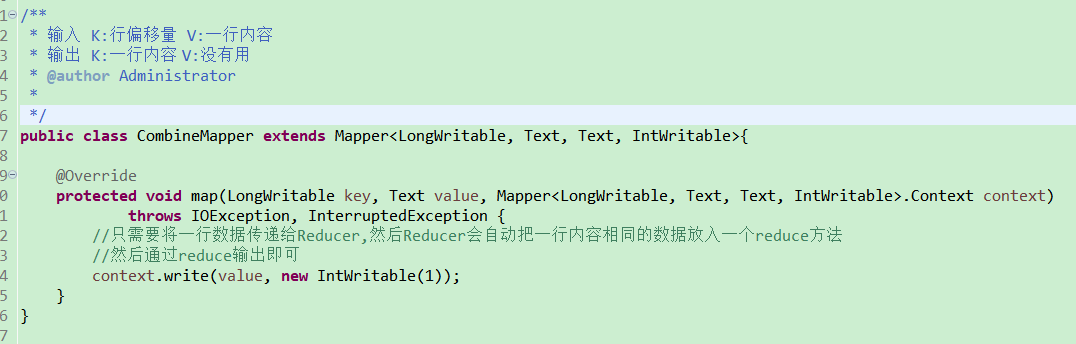
|  |
| --- |
| 20170101 x  20170101 y  20170102 y  20170103 x  20170104 y  20170104 z  20170105 y  20170105 z  20170106 x |

实验思路:

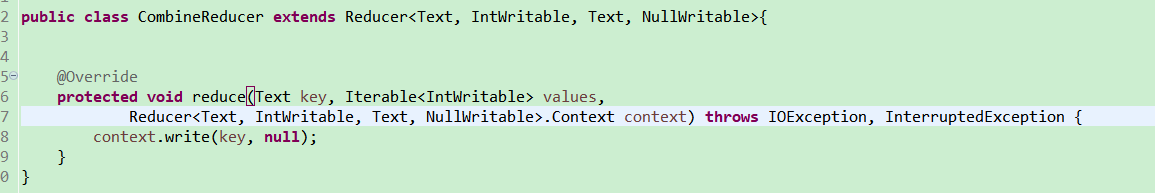
题目要求可以总结成两个字---去重,首先确定去重的目标是一行的内容,然后联系到MapReduce一般的工作流程,数据会流入map方法,输出后到达Reduce方法,最后输出到文件中,在map方法中并没与什么特殊的地方在该题目中有所适用,在Reduce方法中有一个特点,他会将map端输出的相同k放入到一个reduce方法中去处理,因此,这里在map方法中只需要将数据传递给reduce,不进行任何处理,这样每一行数据都会经过过map方法输出到reduce方法中,需要注意的是map输出的k必须是一行的内容,v可以随便,这样当数据到达reduce端的时候,相同行的内容会到达同一个reduce方法,由于可能某一行可能会有多个重复行,因此reduce方法只需要进行输出一次即可,也就是将相同的多行只输出了一次,这样就实现了”去重”的效果

实验过程:

1. 原数据
   1. 文件内容分析
      * + 1. 文件中的内容可以看成两列,在使用MapReduce进行对原数据进行处理的时候,如果实验中有涉及到对每行内容的切分的时候,可以方便后续对数据的处理
   2. 
2. 实验代码
   1. Mapper



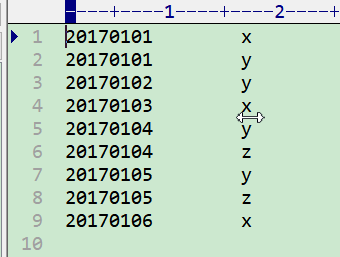
* 1. Reducer



* 1. Driver



1. 操作结果



**（二）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。下面是输入文件和输出文件的一个样例供参考。

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：

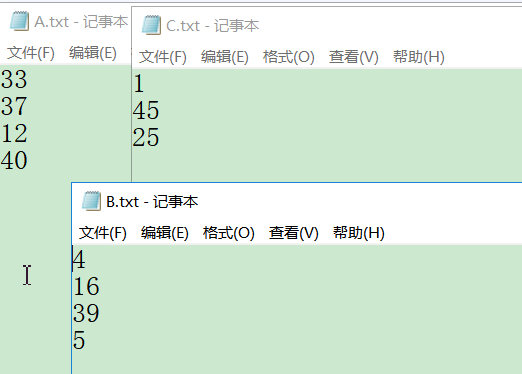
|  |
| --- |
| 1 1  2 4  3 5  4 12  5 16  6 25  7 33  8 37  9 39  10 40  11 45 |

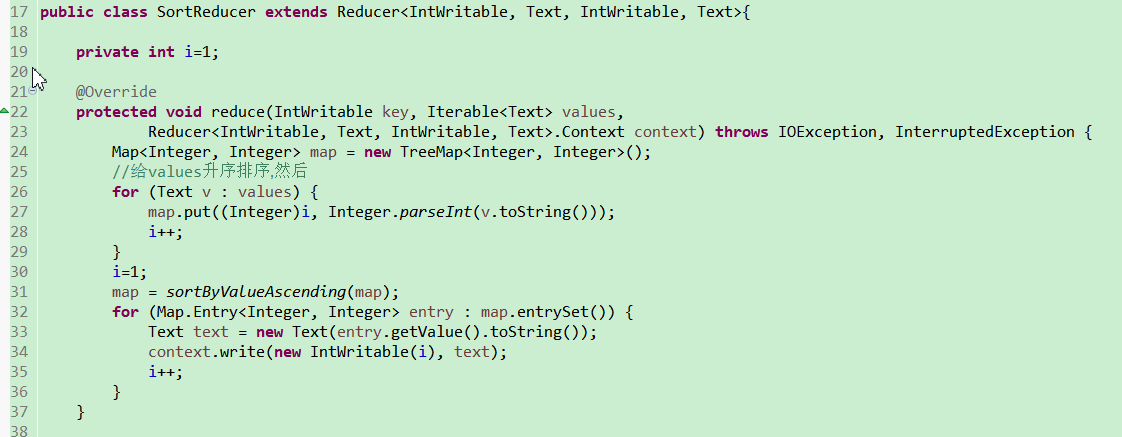
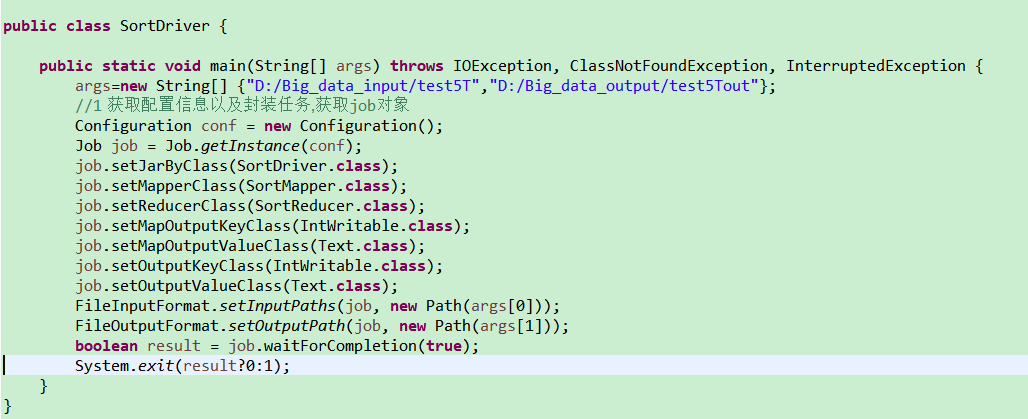
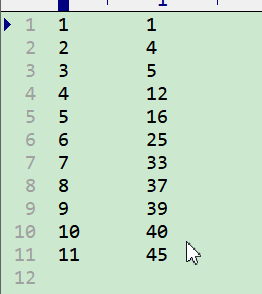
实验思路:

题目要求可以总结成两个字---”排序”,本题的排序有一个非常关键的特点---多文件数据的整体排序,注意到这一点之后,写代码就很简单了,关于map方法默认接收的数据来源是怎么分配的我记不清了,所以还是拿到reduce方法中去处理,仍然是依据上题中的同一点,让map方法把数据送到同一个reduce方法中,这样不同文件的全部数据都到一起了,进行排序就是其他问题了,那么如何将不同文件中的数据输出到同一个reduce方法中呢?首先需要明确的一点是reduce输入来源是map方法输出的,因此可以在map端做一些调整,这里采用的方案是在map的输出中指定相同的K,比如都是1,这样就可以将所有的数据做一个汇总,最后在reduce中进行排序就行了

实验过程:

1. 原数据



1. 实验代码
   1. Mapper
      1. 
   2. Reducer
      1. 
      2. 
   3. Driver
      1. 
2. 结果
   1. 

**（三）对给定的表格进行信息挖掘**

下面给出一个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。

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 |

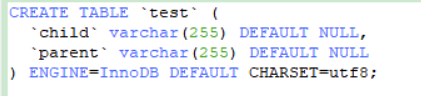
输出文件内容如下：

|  |
| --- |
| 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 |

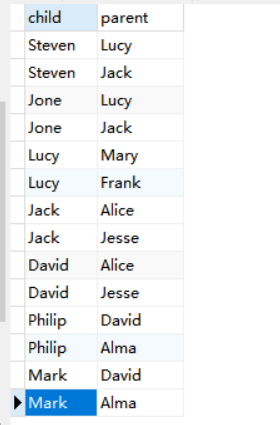
实验思路:

题目可以总结为三个字----”自关联”,从一个子-父关系中,找出子-祖的关系,让我想到了SQL语句中的自关联查询,这里我使用了一个SQL的案例演示一下:

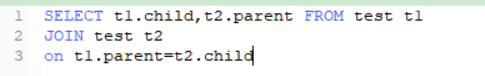
创建一个表test



插入题目中的数据



运行SQL进行查询

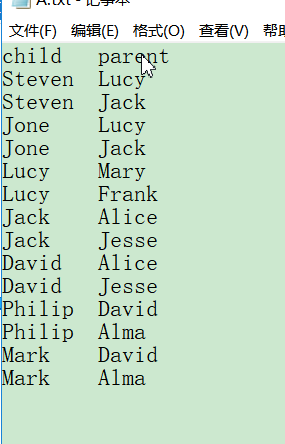


显示结果(符合题目的输出内容)



所以,现在的问题就是如何使用MR任务才实现这种自关联查询的效果,首先,题目中的数据可以看成是一张表,做自关联的话,需要额外”产生”其他数据,这里采用了将每行数据切分之后正反各输出一次的方式,这种输出方式的话,必然产生的是子-父,父-子的两种关系,只有当一个人既是子又是父的情况下,才能在该人的关系中找到子-祖的关系并得到输出,当输出到达reduce的时候,具备上述关系的人会存在于同一个reduce方法中,然后在reduce方法中,将子--祖关系通过代码提取出来,并进行输出即可

实验过程

1. 原数据
   1. 
2. 实验代码
   1. Mapper



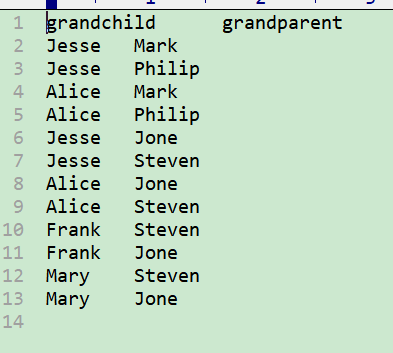
* 1. Reducer



* 1. Driver



1. 操作结果



### 四丶总结

在本次实验中,感觉比较关键的有两点,一个是对题目的理解,这个可以帮我去思考如果去用代码实现具体的逻辑,第二点是关于MR任务运行的工作流程的了解,这三题中并没有设置更复杂的逻辑,我觉得还是存在更加简便的方法对数据进行处理,并能实现预期的效果,这个估计就得结合MR当中涉及到的其他组件了;在编程中,发现MR任务的代码逻辑比较规范,可以一步步的去写,从map--reduce--driver,写完之后便可以输出结果,很方便.