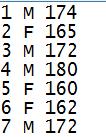
案例描述

本案例假设我们需要对某个省的人口 (1 亿) 性别还有身高进行统计，需要计算出男女人数，男性中的最高和最低身高，以及女性中的最高和最低身高。本案例中用到的源文件有以下格式, 三列分别是 ID，性别，身高 (cm)。



Scala生成数据：

**import** java.io.{File, FileWriter}  
  
**import** scala.util.Random  
  
*/\*\*  
 \* Created by 2281444815 on 2016/11/14.  
 \*/***object** HighCount {  
 **def** main(args: Array[String]): Unit = {  
 **val** writer = **new** FileWriter(**new** File("D:\\peopledata.txt"), **false**);  
 **val** rand = **new** Random()  
 **for** (i <- 1 to 1000) {  
 **var** height = rand.nextInt(220)  
 **if** (height < 50) {  
 height = height + 50;  
 }  
 **var** gender = *getRandomGender*;  
 **if** (height < 100 && gender == "M")  
 height = height + 100  
 **if** (height < 100 && gender == "F")  
 height = height + 50  
 writer.write(i + " " + *getRandomGender*() + " " + height)  
 writer.write(System.*getProperty*("line.separator"))  
 }  
 writer.flush()  
 writer.close()  
 *println*("People Information File generated successfully.")  
 }  
  
 **def** getRandomGender(): String = {  
 **val** rand = **new** Random()  
 **val** randNum = rand.nextInt(2) + 1  
 **if** (randNum % 2 == 0) {  
 "M"  
 } **else** {  
 "F"  
 }  
 }  
  
}

案例分析

对于这个案例，我们要分别统计男女的信息，那么很自然的想到首先需要对于男女信息从源文件的对应的 RDD 中进行分离，这样会产生两个新的 RDD，分别包含男女信息；其次是分别对男女信息对应的 RDD 的数据进行进一步映射，使其只包含身高数据，这样我们又得到两个 RDD，分别对应男性身高和女性身高；最后需要对这两个 RDD 进行排序，进而得到最高和最低的男性或女性身高。

对于第一步，也就是分离男女信息，我们需要使用 filter 算子，过滤条件就是包含”M” 的行是男性，包含”F”的行是女性；第二步我们需要使用 map 算子把男女各自的身高数据从 RDD 中分离出来；第三步我们需要使用 sortBy 算子对男女身高数据进行排序。

c. 编程实现

在实现上，有一个需要注意的点是在 RDD 转化的过程中需要把身高数据转换成整数，否则 sortBy 算子会把它视为字符串，那么排序结果就会受到影响，例如 身高数据如果是：123,110,84,72,100，那么升序排序结果将会是 100,110,123,72,84，显然这是不对的。

基于本地的运行：

**import** org.apache.spark.{SparkConf, SparkContext}  
  
*/\*\*  
 \* Created by 2281444815 on 2016/11/15.  
 \*/***object** PeopleInfoCacular {  
  
  
 **def** main(args: Array[String]): Unit = {  
 **val** logFile = "D:\\peopledata.txt";  
 **if** (logFile .length < 1) {  
 *println*("Usage:PeopleInfoCalculator datafile");  
 System.*exit*(1)  
 }  
 **val** conf = **new** SparkConf().setAppName("People Info Calculator").setMaster("local");  
 **val** sc = **new** SparkContext(conf);  
 **val** dataFile = sc.textFile(logFile , 1);  
 //过滤  
 **val** maleData = dataFile.filter(line => line.contains("M"))  
 .map(line => (line.split(" ")(1) + " " + line.split(" ")(2)));  
 **val** femaleData = dataFile.filter(line => line.contains("F"))  
 .map(line => (line.split(" ")(1) + " " + line.split(" ")(2)));  
 //取得身高，此时我们不需要保留(key,value)队，所以无需采用map(b =>(b.\_2,b.\_1)).sortByKey(false).map(b =>(b.\_2,b.\_1))这种方法。  
 **val** maleHeightData = maleData.map(line => line.split(" ")(1).toInt);  
 **val** femaleHeightData = femaleData.map(line => line.split(" ")(1).toInt);  
 //排序  
 **val** lowestMale = maleHeightData.sortBy(x => x, **true**).first();  
 **val** lowestFeMale = femaleHeightData.sortBy(x => x, **true**).first();  
 **val** highestMale = maleHeightData.sortBy(x => x, **false**).first();  
 **val** highestFeMale = femaleHeightData.sortBy(x => x, **false**).first();  
  
 *println*("Number of Male People:" + maleData.count());  
 *println*("Number of Female People:" + femaleData.count());  
 *println*("Lowest Male:" + lowestMale)  
 *println*("Lowest Female:" + lowestFeMale)  
 *println*("Highest Male:" + highestMale)  
 *println*("Highest Female:" + highestFeMale)  
 }  
}

结果如下：

Number of Male People:498

Number of Female People:502

Lowest Male:100

Lowest Female:100

Highest Male:219

Highest Female:219