assignment5

November 19, 2024

1 Assignment 5 - Ishaan Sathaye

Write Spark programs in Scala to solve the following problems. Do not create any data structures, such as hash maps or tree. Use functional, rather than empirical, programming style. That is, do not use traditional **for**, **while**, or **do-while** statements. Using **foreach** to print the result is fine.

1. The program reads a file full of integers and computes the number of times each integer that is divisible by 3 occurs.

Example input file:

```
1 3 10 3
6 6 9 6
```

Example output:

```
3 appears 2 times, 6 appears 3 times, 9 appears 1 time
```

Do not worry about sorting the result.

Solution:

```scala
import scala.io._

object q1 {
    def main(args: Array[String]): Unit = {
        Logger.getLogger("org").setLevel(Level.OFF)
        Logger.getLogger("akka").setLevel(Level.OFF)

        val conf = new SparkConf().setAppName("assignment5")
        val sc = new SparkContext(conf)

        val lines = sc.textFile("input.txt").persist()
        val numbers = lines.flatMap(_.split(" ")).map(_.toInt)
        val divisibleBy3 = numbers.filter(_ % 3 == 0)
        val counts = divisibleBy3.map((_, 1)).reduceByKey(_ + _)
        val output = counts.collect()
        val res = output.map { case (number, count) =>
            s"$number appears $count times" }.mkString(", ")
        println(res)

        sc.stop()
    }
}
```

2. The program reads a file with employees and a file with departments. The program should print the employee name and department name for each employee. Use a cartesian product and filter to solve the problem. Note that your filter can have the syntax: `.filter{ case (emp, dep) => ... }`. This means that if the input is a tuple of two values, you can process each value individually. Similarly, you can use `.map{ case (emp, dep) => ... }` to format the output.

Example Employee file:

```
John, 23 // John works in department 23
Bob, 55
Steward, 20
Elizabeth, 44
```

Example Department file:

```
23, Computer Science
20, Mechanical Engineering
44, Industrial Engineering
55, Electrical Engineering
```

Example Output:

```
John, Computer Science
Steward, Mechanical Engineering
Bob, Electrical Engineering
Elizabeth, Industrial Engineering
```

Solution:

```scala
import scala.io._

object q2 {
    def main(args: Array[String]): Unit = {
        Logger.getLogger("org").setLevel(Level.OFF)
        Logger.getLogger("akka").setLevel(Level.OFF)

        val conf = new SparkConf().setAppName("assignment5")
        val sc = new SparkContext(conf)

        val employees = sc.textFile("employees.txt")
            .map(_.split(", "))
            .map { case Array(name, dep) =>
                (name, dep.toInt) }.persist()

        val departments = sc.textFile("departments.txt")
            .map(_.split(", "))
            .map { case Array(dep, name) =>
                (dep.toInt, name) }.persist()
        val res = employees.cartesian(departments)
            .filter { case (emp, dep) => emp._2 == dep._1 }
            .map { case (emp, dep) =>
                s"${emp._1}, ${dep._2}" }.collect()

        res.foreach(println)

        sc.stop()
    }
}
```

3. Consider the following input file:

```
Bob Wilson, 11, B CS201
John Back, 23, A CSC369, B CSC366
```

It contains the student name, student ID, and a list of courses the student has taken (grade and course). Write a program with the following example output:

```
Bob Wilson, 11, 3.0
John Back, 23, 3.5
```

The program prints the student name, student ID, and their GPA. You can assume there is a single entry per student in the input file. Not that you can use `line.split(", ", 3)(2).trim()` to only consider the first two commas of a string and get everything after the second comma. Feel free to create a separate method that computes the GPA and a map that converts a letter grade to a numeric value.

Use the **aggregate** operator to solve the problem.

Solution:

```
import scala.io._

object q3 {
    def main(args: Array[String]): Unit = {
        Logger.getLogger("org").setLevel(Level.OFF)
        Logger.getLogger("akka").setLevel(Level.OFF)

        val conf = new SparkConf().setAppName("assignment5")
        val sc = new SparkContext(conf)

        val lines = sc.textFile("input.txt").persist()
        val gpaMap = Map("A" -> 4.0, "B" -> 3.0, "C" -> 2.0, "D" -> 1.0, "F" -> 0.0)

        val res = lines.map { line =>
            val parts = line.split(", ", 3)
            val name = parts(0)
            val id = parts(1)
            val courses = parts(2).split(", ").map(_.trim)
            val (total, numCourses) = courses.aggregate(0.0, 0)(
                { case ((sum, count), course) =>
                    (sum + gpaMap(course.split(" ")(0)), count + 1)},
                { case ((sum1, count1), (sum2, count2)) =>
                    (sum1 + sum2, count1 + count2)})

            val gpa = total / numCourses

            s"$name, $id, $gpa"
        }

        res.collect().foreach(println)

        sc.stop()
    }
}
```