Exercise 8:
Apache Spark

TECHNISCHE UNIVERSITÄT DARMSTADT

Concepts and Technologies for Distributed Systems and Big Data Processing – SS 2016

Solution 2 Count Errors

You can download the code for the solution for this task from the course website.

Solution 3 Approximate Pi

You can download the code for the solution for this task from the course website.

Solution 4 Reverse Graph

Referring to exercise 3, where ReverseGraph was implemented in Hadoop, complete the following code for ReverseGraph using Spark. ReverseGraph should reverse the direction of the edges in a directed graph.

```
Seq(
  3 -> Seq(1, 2),
  1 -> Seq(2, 3))
```

(a) Input

```
Seq(
  1 -> Seq(3),
  2 -> Seq(1, 3),
  3 -> Seq(1))
```

(b) Expected output.

Figure 1: ReverseGraph

A possible input is given in Figure 1a, where each element in the sequence is a pair which assigns the list of outgoing edges to the nodes in the graph. The expected output is given in Figure 1b. As you can see, for each edge $a \rightarrow b$ in the input there is a corresponding edge $b \rightarrow a$ in the output.

```
val sc = new SparkContext
val graph = Seq( /* ... */ )

val reversedGraph = (sc parallelize graph flatMap {
  case (node, edges) => edges map { (_, node) }
}).groupByKey.collect
```

For this task, you should look at the following Spark operators available on `RDD[T]` objects (`RDD[T]` objects are created by methods such as `sc.textFile(`*path*`)` or `sc.parallelize(`*seq*`)`):

```
def map[U](f: T => U): RDD[U]
def flatMap[U](f: T => TraversableOnce[U]): RDD[U]
```