

# Exercise 6: Apache Spark



---

## Concepts and Technologies for Distributed Systems and Big Data Processing – SS 2017

---

---

### 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.

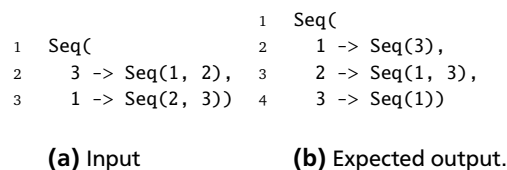


Figure 1: ReverseGraph

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.

```
1 val sc = new SparkContext
2 val graph = Seq( /* ... */ )
3
4 val reversedGraph = (sc.parallelize graph flatMap {
5   case (node, edges) => edges map { (_, node) }
6 }).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]
```