Exercise 3:
Hadoop MapReduce (cont.)

TECHNISCHE UNIVERSITÄT DARMSTADT

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

Solution 2 Reverse Graph

Complete the following code for ReverseGraph, which should reverse the direction of the edges in a directed graph. The input format is given in Figure 1a, where each line represents 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.

The code snippet already contains a regular expression to retrieve the list of numbers for each input line.

```
1 (3, [1, 2])
2 (1, [2, 3])
```

(a) Input

```
1 (1, [3])
2 (2, [1, 3])
3 (3, [1])
```

(b) Expected output.

Figure 1: ReverseGraph

```
1  public static class TokenizerMapper extends Mapper<LongWritable, Text, Text, Text> {
2    private Text from = new Text();
3    private Text to = new Text();
4
5    private Pattern pattern = Pattern.compile("(\\d+)");
6
7    @Override
8    protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
9      Matcher m = pattern.matcher(value.toString());
10
11     m.find();
12     to.set(m.group());
13     while (m.find()) {
14         from.set(m.group());
15         context.write(from, to);
16     }
17
18   }
19 }
20
21 public static class InvertedReducer extends Reducer<Text, Text, Text, Iterable<Text>> {
22   @Override
23   protected void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {
24
25     context.write(key, values);
26
27   }
28 }
```

Solution 3 Relational Join

Complete the following code for RelationalJoin, which should join two tables based on the same numeric identifier. Figure 2a shows the input data for the tables `Department` and `Employee`. The output should indicate the assignments from employees to departments as shown in Figure 2b.

```
Department,1234,Sales
Employee,Susan,1234
Department,1233,Marketing
Employee,Joe,1233
Department,1233,Accounts
```

(a) Input

```
1233,Joe,Accounts
1233,Joe,Marketing
1234,Susan,Sales
```

(b) Expected output.

Figure 2: ReverseGraph

```
public static class JoinMapper extends Mapper<LongWritable, Text, Text, Text> {
  @Override
  protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

    String[] record = value.toString().split(",");
    if (record[0].equals("Department")) {
        context.write(new Text(record[1]), value);
    } else {
        context.write(new Text(record[2]), value);
    }

  }
}

public static class JoinReducer extends Reducer<Text, Text, Text, Text> {
  @Override
  protected void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {

    List<String[]> departments = new ArrayList<String[]>();
    List<String[]> employees = new ArrayList<String[]>();

    String[] record;
    for (Text val : values) {
        record = val.toString().split(",");
        if (record[0].equals("Department")) {
            departments.add(record);
        } else {
            employees.add(record);
        }
    }

    for (String[] department : departments) {
        for (String[] employee : employees) {
            context.write(null, new Text(key.toString() + "," + employee[1] + "," + department[2]));
        }
    }

  }
}
```