

# Exercise 3: Hadoop MapReduce (cont.)



---

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

---

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

(a) Input                      (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.

1	Department,1234,Sales	
2	Employee,Susan,1234	
3	Department,1233,Marketing	1 1233,Joe,Accounts
4	Employee,Joe,1233	2 1233,Joe,Marketing
5	Department,1233,Accounts	3 1234,Susan,Sales

(a) Input                      (b) Expected output.

Figure 2: ReverseGraph

```
1 public static class JoinMapper extends Mapper<LongWritable, Text, Text, Text> {
2     @Override
3     protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
4
5         String[] record = value.toString().split(",");
6         if (record[0].equals("Department")) {
7             context.write(new Text(record[1]), value);
8         } else {
9             context.write(new Text(record[2]), value);
10        }
11    }
12 }
13 }
14
15 public static class JoinReducer extends Reducer<Text, Text, Text, Text> {
16     @Override
17     protected void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {
18
19         List<String[]> departments = new ArrayList<String[]>();
20         List<String[]> employees = new ArrayList<String[]>();
21
22         String[] record;
23         for (Text val : values) {
24             record = val.toString().split(",");
25             if (record[0].equals("Department")) {
26                 departments.add(record);
27             } else {
28                 employees.add(record);
29             }
30         }
31
32         for (String[] department : departments) {
33             for (String[] employee : employees) {
34                 context.write(null, new Text(key.toString() + "," + employee[1] + "," + department[2]));
35             }
36         }
37     }
38 }
39 }
```