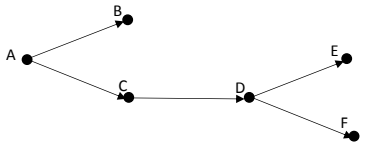
University of Augsburg, Institute of Computer Science
Prof. Dr. P. Fischer
J. Kastner, L. Rudenko

SS 2019
17. May 2019
Sheet 4

Analyzing Massive Data Sets

Exercise 1: MapReduce - Path Search in the Graph (homework)

Given is a directed acyclic graph that should be analyzed using the MapReduce framework. Write Python code for *map* and *reduce* functions, which allow to output a list of all nodes to be reached from a certain start node. The actual path steps do not not need to be provided, just the reached nodes. How do you express the iteration? How many steps are needed for the given graph? What would you do if the size of the graph is not known beforehand?



Example: For the graph in Figure and the start node C the result set of reachable nodes is $[D, E, F]$, for the start node D - $[E, F]$, for B as start - en empty set and so on.

Exercise 2: MapReduce to Spark (live)

a) In this exercise you should write the Python code that can execute any MapReduce job step (specified by *input list*, *map* and *reduce* functions written for our *mr_simulator*) on Spark using Spark's RDD API provided by Pysparkling. The original *input list*, *map* and *reduce* functions must be reused in your solution.

You can test your solution on a concrete example, such as the well known Duplicate Files exercise (see Sheet 2, Exercise 1).

As reminder the *map* and *reduce* functions, which output the duplicate files:

```
def mapper(md5hash, filename):
    return [(md5hash, filename)]

def reducer(md5hash, filenames):
    res = []
    if (len(filenames) > 1):
        filenames = list(set(filenames))
        res.append((md5hash, filenames))
    return res
```

As input use the following list:

```
inputs = [(123, 'Name1'), (123, 'Name2'), (456, 'Name1'), (345, 'Name3'), (456, 'Name2'),
(123, 'Name1'), (123, 'Name3'), (789, 'Name4'), (789, 'Name2'), (136, 'Name5')]
```

Name the advantages and disadvantages of this solution for this specific task (finding duplicate files) in comparison to the solution for Exercise 3 a) from Sheet 3.

b) Now make your solution reusable for other MapReduce exercises by creating a Python module *mrsim_spark*, that defines the function *mr_simulator(inputseq, mapfunc, reducefunc, debug)* (same name and arguments as im *mrsim.py* provided on Digicampus), implemented using Pysparkling. Make sure to add a possibility for printing intermediate results to *stdout* (output of *map* and input to *reduce* functions). Use this new module to test your implementation on another exercise, that was solved with *mr_simulator* (such as calculating of vector length).

c) Consider whether it makes sense to use approach from subtask a) or b) each time you want to port a task that was solved with MapReduce to Spark.

Exercise 3: Expressing Similarity (live)

Following documents are given:

- D_1 : "the red cat sits on the red chair"
- D_2 : "the red chair is next to the blue chair"
- D_3 : "the black cat sits next to the blue chair"

a) First of all determine the **word frequencies**. After that calculate the **similarities** of **each document to each other document** by using

   i) the **Jaccard** similarity.
   ii) the **Euclidean** distance.
   iii) the **Cosine** similarity.

b) Now ignore **stopwords** like **the**, **is**, **on** and **to** and recalculate the **similarities** given in subtask a). Which change can you recognize?