University of Augsburg, Institute of Computer Science | SS 2019
Prof. Dr. P. Fischer | 27. May 2019
J. Kastner, L. Rudenko | Solution 4

Analyzing Massive Data Sets

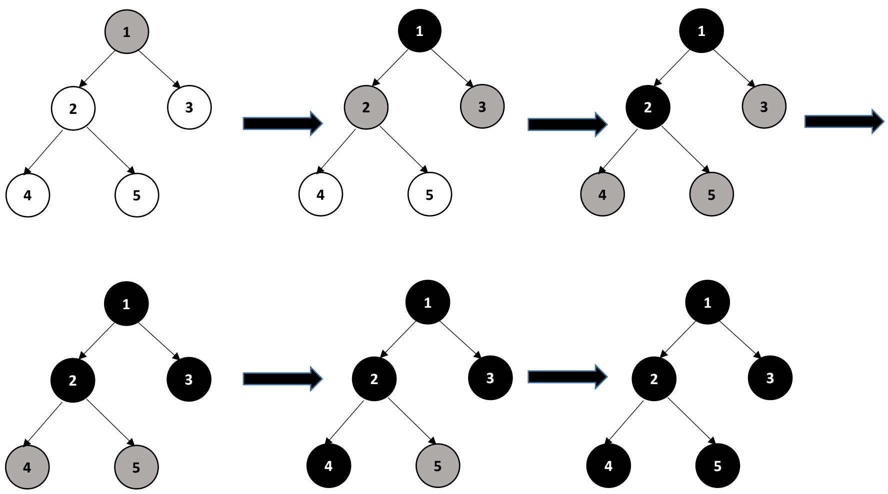
Exercise 1: MapReduce - Path Search in the Graph (homework)

We want to output a list of all nodes to be reached from a **certain start node**.

Input format: ($start\_node$, $distance\_from\_start\_node$, $neighboring\_nodes$, $color$):

- $start\_node$ - node for which we want to output a list of all reachable nodes.
- $distance\_from\_start\_node$ - in the beginning we do not know the distance and we mark it with **unknown** ($float('inf')$ or something like that). The distance for the $start\_node$ is set to 0.
- $neighboring\_nodes$ - list of the directly adjacent nodes.
- $color$ - the color tells us whether or not we have seen the node before, so this starts off as **white**. The $start\_node$ is marked as **grey**.

Easy example flow:



The *mapper* is responsible for "exploding" all gray nodes. For each directly adjacent nodes of each gray node the *mapper* emmits a new gray node with $distance = distance + 1$. It also then emit the input gray node, but colors it *black*. The *mapper* also emits all non-grey nodes with no change. The *mapper* doesn't know, what to write for the $neighboring\_nodes$, so it leaves it blank.

The *reducer* receive all data for a given key (in this case - all "copies" of each node). The *reducers* job is to construct a new node using the non-null list of $neighboring\_nodes$, the minimum distance and the darkest color.

You are ready when the are no output nodes that are colored *grey*. However, the *white* nodes can still exist, these are those that are not accessible from the given $start\_node$.

```
from mrsim import mr_simulator

# start node is 'C', there are the pathes C->D, C->DE, C->DF
prepared_nodes = [('A', (float('inf'), ('B','C'), 'white')),
                  ('B', (float('inf'), None, 'white')),
                  ('C', (0, ('D'), 'grey')),
                  ('D', (float('inf'), ('E','F'), 'white')),
                  ('E', (float('inf'), None, 'white')),
                  ('F', (float('inf'), None, 'white'))]

startNode = None
for node in prepared_nodes:
    if(node[1][2] == 'grey'):
        startNode = node[0]

def map(key, val):
    res = []
    if(val[2] == 'grey'):
        if val[1] != None:
            for node in val[1]:
                res.append((node, (val[0]+1, None, 'grey')))
        res.append((key, (val[0], val[1], 'black')))
    else:
        res.append((key, (val[0], val[1], val[2])))
    return res

def reduce (key, val):
    res = []

    if(len(val) > 1):

       min = float('inf')
       neighborsList = None
       color = 'white'

       for t in val:
           if t[0] < min:
               min = t[0]
           if t[1] != None:
               neighborsList = t[1]
           if t[2] == 'black':
               color = t[2]
           elif t[2] == 'grey' and color != 'black':
               color = t[2]
       node = key, (min, neighborsList, color)
       res.append(node)
    else:
       node = key, val[0]
       res.append(node)
    return res


def computationCont(res):
    cont = False
    for node in prepared_nodes:
        if(node[1][2] == 'grey'):
           cont = True
    return cont


while computationCont(prepared_nodes) == True:
      intermediateResult = mr_simulator(prepared_nodes, map, reduce)
      prepared_nodes = list(intermediateResult)

print(prepared_nodes)
```

```
reachableNodes = []
for node in prepared_nodes:
    if(node[1][2] == 'black' and node[0] != startNode):
        reachableNodes.append(node[0])

print('From start node ' + startNode + ' we can reach the following nodes: '
      + str(reachableNodes) + '.')
```

The number of steps needed fo the given graph depends on the **start node** For C, for example, we need 4 iterations, for D - 3 and so on.

For the existing solution we don't need to know the size of the graph.

Exercise 2: Spark - Duplicate Files (live)

The solution was discussed in the exercise.

Exercise 3: Expressing Similarity (live)

The solution was discussed in the exercise.