TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Dr. Thomas Neumann

Exercises for *Foundations in Data Engineering*, WiSe 23/24

Alexander Beischl, Maximilian Reif (i3fde@in.tum.de)
http://db.in.tum.de/teaching/ws2324/foundationsde

Sheet Nr. 11

Exercise 1

1. Implement iterative PageRank computation in pseudo code on a directed graph. Instead of an informed termination criterion, just do 20 iterations.

$$PR_0(n) = \frac{1}{graph.nrNodes} \qquad \text{(init)}$$

$$PR_{i+1}(n) = \frac{1-d}{graph.nrNodes} + d \sum_{o \in in(n)} \frac{PR_i(o)}{|out(o)|} \qquad \text{(step)}$$

Perform one initialization step. Then in one iteration, update the PageRank PR for all nodes in the graph. Repeat 20 times. To work on the graph, these fragements may be helpful:

```
# Retreives number of nodes in the graph
nrNodes = graph.nrNodes()
# iterates over all nodes in graph
for(n : graph.nodes()) /* do something with n */
for(neigh : graph.out(n)) # iterates over the neighbors
for(neigh : graph.in(n)) # iterates over the
                          # reverse neighbors (incoming edges)
# Creates a property on the graph.
# A property stores information for every node.
pr = graph.makeProperty()
pr[n] = 5 # sets property pr for node n to 5
```

2. Assume this computation is performed in main memory. Which parts of the implementation exhibit good behavior for modern computers, which parts are problematic and why? Do caches help to alleviate this?
3. What happens when the memory requirements of your implementation exceed the available main memory?
4. Can you think of an optimization that helps when the working set is only slightly larger than main memory?
5. Create a simple adaption of your PageRank implementation to distribute the code on multiple machines. Assume that you have at least these two functions for networking available: `MachineId getMachineIdForGraphNode(Node n)` and `send(MachineId m, Message m)`.
6. At which point may this implementation make inefficient use of available resources?
7. Implement the same computation with MapReduce.
8. How is message passing between machines handled now? How is random access addressed? How is the out-of-memory case handled?