Distributed Graph Lab

Ferran Torres Morales, Goktug Cengiz

March 2019

Exercise 1

Given a graph, in this first exercise it will be explained the Supersteps performed for it so that every node has the maximum state of the hole.

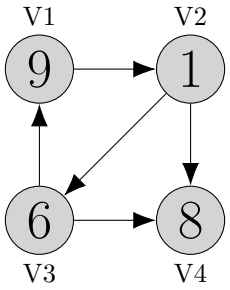

Figure 1: Initial state of the graph

In Superstep 0, all nodes sent their (`Integer.MAX_VALUE`) to the nodes connected and all of them start as active nodes. After that, only the vertices with a higher value than the neighbours and an outgoing connection to them, would send their value, so only V1 will send their initial state of 9 to V2 because V2 has a lesser state value, as shown in Figure 2.

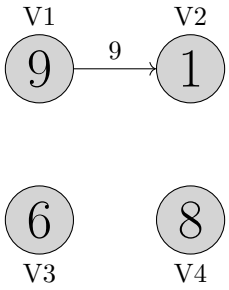

Figure 2: Message send in Superstep 0

In Superstep 1, V2 would be the only vertex to receive a message, and therefore the only one to execute the algorithm. This vertex would replace their current value (1) with the value received by V1 (9) because it is higher. After changing his current state to 9 the vertex will send the value to all his neighbours who have a state with a lesser value. That would mean that V2 would send their current state (9) to vertexes V3 and V4 as can be seen in Figure 3.

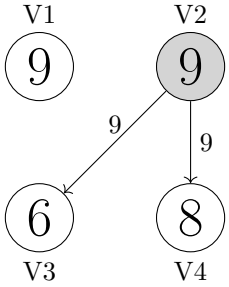

Figure 3: State changes and messages in Superstep 1

Finally, in Superstep 2 V3 and V4 will receive a message by V2 that will cause an update in their states, both will set it to 9 because it is a higher value. After that, they will not send any message: all neighbours of vertex V3 have equal or higher state values and V4 does not have any outgoing edge so no message can be send by him. If no more messages are sent, convergence is reached and the algorithm finishes. The final state is the one shown in figure 3 and therefore the maximum value will be 9.

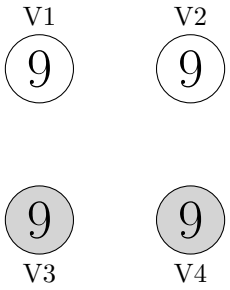

Figure 4: Final state after Superstep 2

Exercise 2

For the second exercise it needs to compute the shortest path from one vertex to all the other vertexes in the graph. The exercise would be computed for node 1 to the rest of the network.

As we want the shortest path, for the function merge we would return the minimum value of both passed. Therefore, we will only store the minimum value to an specific node even though it could be reached by different vertex although this did not happened for our graph. The same thing happens for the VProg function, as it will send just the minimum value and the message received.

Finally, the sendMsg function will send the message received from the previous vertex. If it is not the maximum value, it will recompute the total cost to reach the node and send it to the next one with the cost of the path.

Exercise 3

In this exercise it is almost the same as exercise 2 but not only the cost is stored but also the path. If the cost is updated, so is the path to reach an specific node. For that reason, an auxiliary class has been created to store the path with the IDs of vertices navigated and the cost to reach that vertex.

In the merge function of the algorithm only the cost is taken into account. In case of having the same cost to reach a vertex but by two different paths, it could be improved by choosing the one path with less vertexes, but it is not implemented because for this graph it is not used since any vertex receives two

messages at the same time.

Exercise 4

For this exercise the edges and vertexes have been loaded and the pageRank algorithm has been executed and the files were passed to a csv format. Therefore, the MAX_ITERATIONS of the algorithm needs to be defined and also the DAMPING_FACTOR. For the maximum iterations when it is higher than 400 the algorithm does not reach convergence, it does not finish, and it runs out of memory. If using a maximum iterations value between 100 and 200 it takes between 30 and 40 seconds to finish. As for the damping factor, values too big or too small do not work well, so values between 0.15 and 0.85 have been used with minimum differences. Also varying the damping factor from 0.8 up to 0.9 the top 10 most cited papers are the same but with different values and in some cases some order is different.

The final results using maximum iterations at 100 and damping factor at 0.85 are:

```
+------------------+--------------------+------------------+
|                id|               title|          pagerank|
+------------------+--------------------+------------------+
|8830299306937918434|University of Cal...| 3124.264171486379|
|1746517089350976281|            Berkeley|1572.4151005334245|
|8262690695090170653|         Uc berkeley|384.25144673972784|
|7097126743572404313|Berkeley Software...| 214.0572551757324|
|8494280508059481751|Lawrence Berkeley...|193.68997507337514|
|1735121673437871410|     George Berkeley| 193.6716555575404|
|6990487747244935452|      Busby Berkeley|113.25606403048357|
|1164897641584173425|      Berkeley Hills|105.89779003610528|
|5820259228361337957|     Xander Berkeley| 71.85250422941961|
|6033170360494767837|     Berkeley County| 68.47601926652182|
+------------------+--------------------+------------------+
```

The damping factor as mentioned in the article [1], should be around 85%, so the one used of 80% it is accurate enough for the exercise. Also convergence could be ensured by adding an additional node that is pointed by every node and that points to every one of the nodes.

References

[1] Serge Abiteboul, Ioana Manolescu, Philippe Rigaux *http://webdam.inria.fr/Jorge/files/wdm.pdf*. 2011