Team Members

Andrea Armani
Dimitrios Tsesmelis

A. Exercise 1

In this chapter we are describing each superstep that will be performed for the given graph.

Superstep 0

Vertex ID	Initial value	Received message(s)	Calculation done	Result value	Sent message(s)	Receivers (Vertex ID)
1	9	2147483647	return vertexValue	9	9	1
2	1	2147483647	return vertexValue	1	-	-
3	6	2147483647	return vertexValue	6	-	-
4	8	2147483647	return vertexValue	8	-	-

In the first superstep, all nodes receive the Integer.MAX_VALUE and hence the result state is same as their initial (each node keeps each initial value). Regarding the communication, only the Vertex 1 needs to send a message to Vertex 2, as its value is bigger than the destination value (9 > 1).

Superstep 1

Vertex ID	Initial value	Received message(s)	Calculation done	Result value	Sent message(s)	Receivers (Vertex ID)
1	9	-	-	9	-	-
2	1	9	return Math.max(vertexValue,message)	9	9	3, 4
3	6	-	-	6	-	-
4	8	-	-	8	-	-

In the second superstep, the only node that gets active as it receives a message, is node 2. Regarding the calculations, it compares its old value (1) with the value received (9) and it returns the biggest one (9). The next action is to propagate the new value to the neighbor nodes that have smaller values than 9. In our case, nodes 3 and 4 will receive a message as their value is 6 and 8, respectively.

Superstep 2

Vertex ID	Initial value	Received message(s)	Calculation done	Result value	Sent message(s)	Receivers (Vertex ID)
1	9	-	-	9	-	-

2	9	-	-	9	-	-
3	6	9	return Math.max(vertexValue,message)	9	-	-
4	8	9	return Math.max(vertexValue,message)	9	-	-

In the third superstep, both nodes 3 and 4 receive a message from node 2 with the value 9. They compare this value with their initial value, and they return the maximum, which is 9 for both. In this superstep, no messages are sent because node 3 and 4 do not have outgoing edges that go to node with smaller value. For instance, node 3 has an outgoing edge to node 4, but both nodes have the value 9 and hence no message is sent.

The Pregel operators terminates iteration and returns the maximum value when there are no messages remaining.

B. Exercise 2

In this exercise we implement a distributed algorithm that given a graph, it computes the shortest path between node A and all the other nodes. The code for the functions VProg, sendMsg and merge is identical to exercise 1 with some modifications:

- Regarding the VProg function, each node returns its value during superstep 0 and the **minimum** value between its value and the value of the received message during the other supersteps.
- Regarding sendMsg, each node calculates **mysum**, which is the cost to get to this node + the weight of the edge that links the current node with its neighbor node. In case that mysum is less than the current cost to go to the neighbor node, a message is sent to the neighbor node with the mysum as message. Otherwise, no operation is performed.

C. Exercise 3

In exercise 3, we are requested to compute the cost from node A to every other node as well as the corresponding path. The code that we used is very similar to the previous exercise with the difference that we changed the attributes that are stored inside each node of the graph. More specifically, in each node we store an **id** (Object), the **cost** (Integer) and the **path** (List<Long>) to get to this node, while in exercise 2 we stored only the id and the cost.

In this case, we modified the messages that are sent from one node to the other, as we need to update both the cost and the path. As a result, every time that a node sends the new cost to another node, it also updates the path by adding the id of the destination node to the current path.

Finally, every node has the minimum cost required to visit this node from node A as well as a List of Long numbers (node ids). While outputting the results, we transform the node ids to labels (e.g. the list [1,2] is transformed to [A,B]).

D. Exercise 4

In this chapter we are presenting the strategy that we applied in order to identify the 10 pages in the Wikipedia graph that have the highest PageRank.

As we know, the PageRank algorithm takes as parameter the damping factor and the maximum number of iterations, which are both given by the user. To find the best parameter for our dataset we first tried to fix the number of iterations and then we experimented with different values for the damping factor. Below we can see the results.

Maximum number of iterations:

We found that the best value for this parameter is around **20**. The PageRank algorithm is an iterative algorithm and hence we need to find the value of iterations where the algorithms converges. In our case, this number is around 20, as for smaller values (e.g. 17), we see that the PageRank is much different than the PageRank with maximum_iterations = 20. On the other hand, when we use bigger values than 20 (e.g. 30), the PageRank is pretty much the same. Some of the results are depicted below.

Table 1: Max_iterations = 12, damping_factor = 0.9

ID	PageRank	Title
8830299306937918434	3177.666591	University of California
1746517089350976281	1625.834499	Berkeley, California
8262690695090170653	394.211693	Uc berkeley
7097126743572404313	225.4905796	Berkeley Software Distribution
8494280508059481751	203.909073	Lawrence Berkeley National Laboratory
1735121673437871410	199.8148512	George Berkeley

Table 2: Max_iterations = 20, damping_factor = 0.9

ID	PageRank	Title
8830299306937918434	3177.978101	University of California
1746517089350976281	1623.147402	Berkeley, California
8262690695090170653	393.8532174	Uc berkeley
7097126743572404313	225.0611791	Berkeley Software Distribution
8494280508059481751	203.3862454	Lawrence Berkeley National Laboratory
1735121673437871410	199.6022853	George Berkeley

Table 3: Max_iterations = 30, damping_factor = 0.9

ID	PageRank	Title
8830299306937918434	3178.142071	University of California
1746517089350976281	1622.919836	Berkeley, California
8262690695090170653	393.8220947	Uc berkeley
7097126743572404313	225.0313902	Berkeley Software Distribution
8494280508059481751	203.3302605	Lawrence Berkeley National Laboratory
1735121673437871410	199.6124193	George Berkeley

Damping factor:

It is stated by Google, which initially introduced the PageRank algorithm, that the "best" value for the damping factor is around 0.85. In our results, we confirm this statement and we conclude that a very good range for this parameter is from **0.8 to 0.9**, with values close to 0.85 to be the best.

Definitely, very low damping factor is NOT a good choice, as the relative PageRank of the pages will be too close to each other. That is because every page will have a very big weight times (*) a constant value and a smaller wight times (*) the PageRank received from external pages.

On the other hand, a very high damping factor (e.g. 0.99) makes the calculations more difficult, which means that the algorithm needs more iterations to converge. It is also known that numeric instability arises when damping factor is too close to 1.

Table 4: Max_iterations = 20

Title	PageRank (Df = 0.9)	PageRank (Df = 0.85)	PageRank (Df = 0.8)	PageRank (Df = 0.3)
University of California	3177.978101	3124.234891	3058.08916	1648.246695
Berkeley, California	1623.147402	1572.46549	1520.543032	787.140678
Uc berkeley	393.8532174	384.2585598	373.8586808	200.1300395
Berkeley Software Distribution	225.0611791	214.0642208	203.2334965	95.13724785
Lawrence Berkeley National Laboratory	203.3862454	193.7021786	187.3239037	91.56960394
George Berkeley	199.6022853	193.6699518	184.2896003	84.79804189
Busby Berkeley	119.9088869	113.2364035	108.9002161	55.24502862
Berkeley Hills	117.596993	105.9199828	93.27842872	40.07019796
Xander Berkeley	73.18999502	71.84579009	70.73743761	38.51917279
Berkeley, CA	72.53990452	68.48570379	64.39776961	37.55470586

From the above results we can observe that by keeping fixed the maximum number of iterations, the relative PageRank obtained using damping factor = 0.3 are much closer to each other compared to bigger damping factors. This confirms our initial hypothesis.

Moreover, the results using damping factor of 0.8, 0.85 and 0.9 are similar which confirms that this is a suitable range for the parameter.