EXERCISE 1 – BFS

Implement an efficient BFS algorithm.

The algorithm has been implemented and it is included in "Practical 2/Exercise1 – BFS" directory. It can be found on this GitHub site.

Use it to make an algorithm that outputs all connected components and their sizes (number of nodes). Test your algorithm on some real-world graphs, what can you say?

The BFS algorithm deployed is able to obtain both the diameter and the maximum size in all the test cases. As we have tested it with 10 different sizes of graphs, we can say that the algorithm deployed is scalable.

Use your BFS algorithm to make an algorithm that computes an approximation of the diameter of a graph. Test it on real-world graphs with a known diameter.

These are the results for the tests carried out over real-world graphs:

NAME	DIAMETER		MAXIMUM SIZE	
-	GIVEN	MEAS.	GIVEN	MEAS.
Email-Eu-core	7	7	986	986
Slashdot 2009	11	13	82168	82168
Slashdot 2008	10	12	77360	77360
Epinions	14	15	75877	75877
Wikipedia Vote	7	6	7066	7066
General Relativity	17	17	4158	4158
Amazon	44	47	334863	334863
DBLP	21	22	317080	317080
YouTube	20	24	1134890	1134890
Facebook	8	8	4039	4039

Comments on the results obtained:

Under this experiment, we found that Stanford University got different results in several cases for the computed diameter of the graphs under test. On the other hand, the maximum size of the graphs (number of nodes) is always the same in both cases.
Therefore, we can observe that, either their algorithm or ours (or both), is over/underestimating the diameter value.

EXERCISE 2 – TRIANGLES

Implement an efficient algorithm for counting triangles.

The algorithm has been implemented and it is included in "Practical 2/Exercise2 – Triangles" directory. It can be found on this GitHub site.

Generalize it to compute the transitivity ratio of the graph.
Generalize it to count the number of triangles each node belongs to and to compute the clustering coefficient of each node in the graph and the clustering coefficient of the graph.

These are the results for the tests carried out over real-world graphs:

NAME OF THE GRAPH	NUMBER OF TRIANGLES		CLUSTERING COEFFICIENT		TRASITIVITY RATIO	
-	GIVEN	MEAS.	GIVEN	MEAS.	GIVEN	MEAS.
Email-Eu-core	105461	105461	0.3994	0.45045	0.1085	0.26739
Slashdot 2009	602592	602592	0.0603	0.09238	0.00816	0.02410
Slashdot 2008	551724	551724	0.0555	0.08725	0.00818	0.024157
Epinions	1624481	1624481	0.1378	0.26051	0.0229	0.06567
Wikipedia Vote	608389	608389	0.1409	0.20885	0.04564	0.125479
General Relativity	48260	48260	0.5296	0.68653	0.3619	0.62984
Amazon	667129	667129	0.3967	0.42974	0.07925	0.20522
DBLP	2224385	2224385	0.6324	0.732135	0.1283	0.30637
YouTube	3056386	3056386	0.0808	0.172258	0.002081	0.00621
Facebook	1612010	1612010	0.6055	0.61700	0.2647	0.519174

Comments on the results obtained:

In all test cases, the number of triangles in the graph obtained is the same as the one published on the source website. On the other hand, the clustering coefficient computed by our algorithm is always bigger than the value proposed by Stanford University. Finally, the transitivity ratio obtained is as well bigger than the one published in the website.

As a conclusion, the difference in the values obtained for the clustering coefficient and the transitivity ratio might be caused by using different calculations to compute those parameters,

EXERCISE 3 – DIJKSTRA

Implement the Dijkstra algorithm.

The algorithm has been implemented and it is included in "Practical 2/Exercise3 – Dijkstra" directory. It can be found on this GitHub site.