Programming assignment 4 - Emil Collén

Problem 4

When it comes to finding the shortest path, there are some fundamental differences between Dijkstra´s algorithm and Bellman-Ford. For starters, Dijkstra´s uses a greedy approach that cannot handle negative weights. When we use Dijkstra´s on a graph with negative weights, we cannot rely on the result as it may be incorrect because it cannot re-evaluate. Bellman-Ford however does iterate V-1 times, making re-evaluations each step, making negative weights possible. It can also detect negative cycles as the path cost to the vertices in the cycle keeps reducing over and over until the iterations are done. It is printed out when there is a negative cycle in the graph. All cases below were ran with 20-100 iterations, and an average time was taken. When we allow negative cycles, we assume that Dijkstra´s is unreliable and is therefore not being run.

Regarding graphs with no cycles and only positive weights, Dijkstra´s and Bellman-Ford will produce the shortest path. The time to produce the result varies a lot, however. If we start by looking at figure 1, we can see that Bellman-Ford takes much longer the more edges we have. The Dijkstra´s representation almost looks infinitely small in comparison. We see the same thing when we look at figure 2, but with 5000 vertices instead of 1000. In figures 3 and 4, we also include negative weights for the Bellman-Ford algorithm. We can see it takes even more time when we allow negative weights. This result is expected, as we open for infinite loops when we allow negative cycles. When using Dijkstra´s however, we only run positive edge weights on, as described previously. In figure 5, only Dijkstra´s time growth is shown for the 5k-vertice graph, to we can see that the growth is not none or infinitely small, but actually almost linear.

We know that for Bellman-Ford, we have a time complexity of O(V*E) which aligns well with the results from the comparison. When increasing number of edges, we see that all cases of Bellman-Ford shows a linear growth corresponding to the known time complexity of the algorithm. Dijkstra´s algorithm we can get a time complexity of O(E*log(V)), since we use a min heap, which just above linear in relation to number of edges. The time for graphs to find the shortest path using Dijkstra's, with edges ranging from 10k to 80k is shown for 5k vertices in figure 5. All results are in presented as numbers in table 1.

As a conclusion I would say that which is best entirely depends on the graph. If we have a graph with only positive edge weights, Dijkstra´s is unbeatable as it outperforms Bellman-Fords by far in time consumption. If the graph has negative weights, Dijkstra´s cannot even be trusted to find the shortest path – therefore, Bellman-Ford is the recommended choice with negative edge weights.

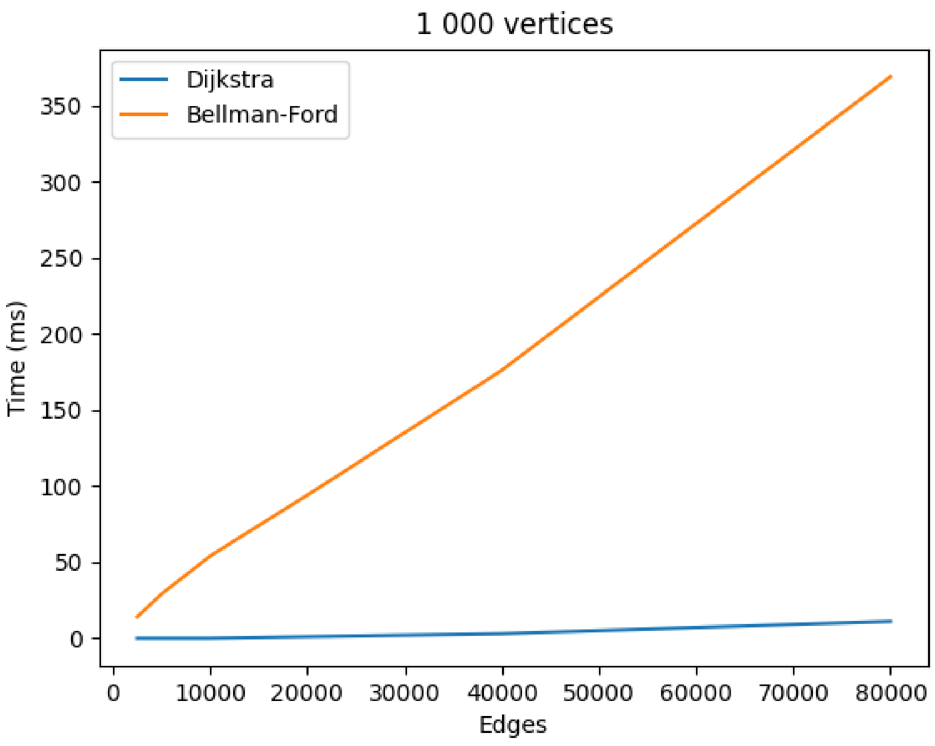

Figure 1 - 1k vertices with only positive edge weights

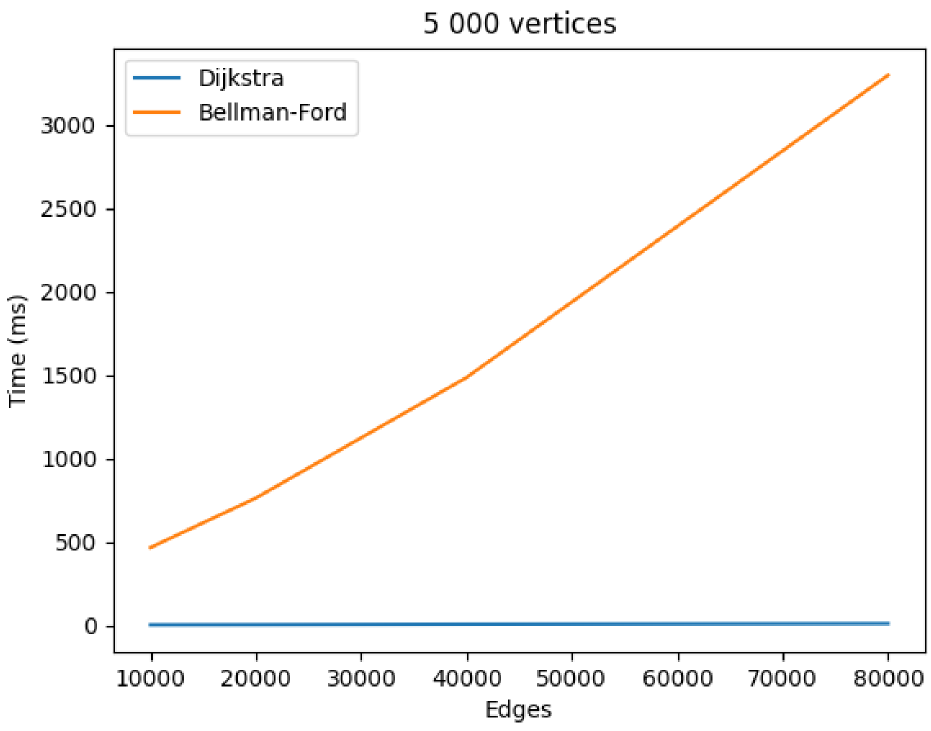

Figure 2 - 5k vertices with only positive edge weights

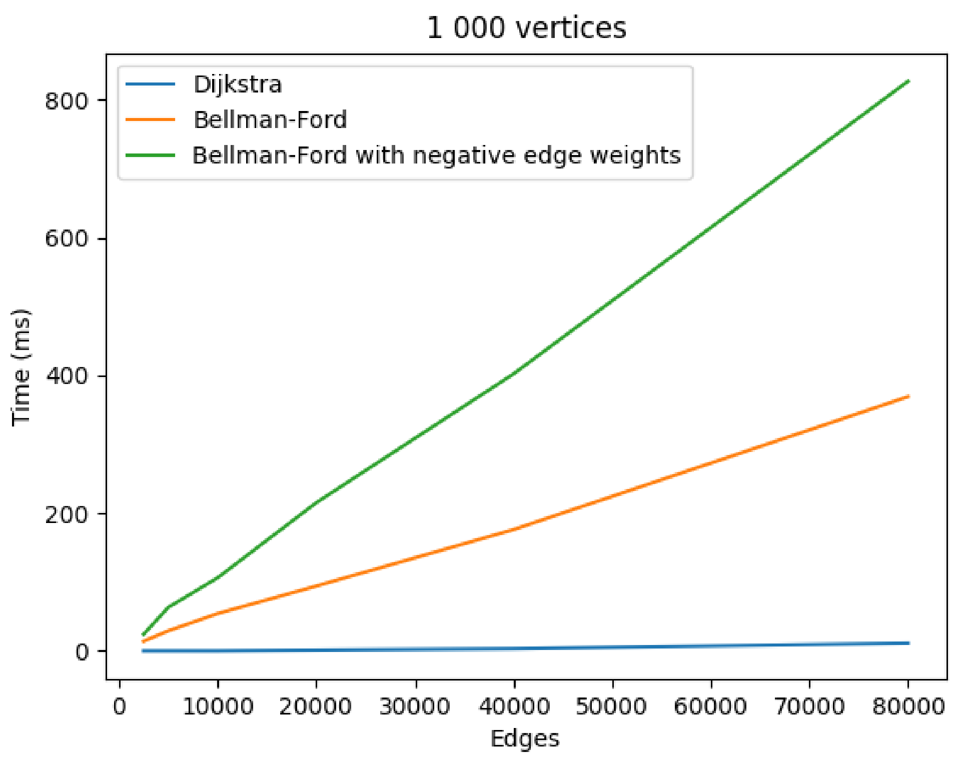

Figure 3 - 1k vertices with both positive and negative edge weights

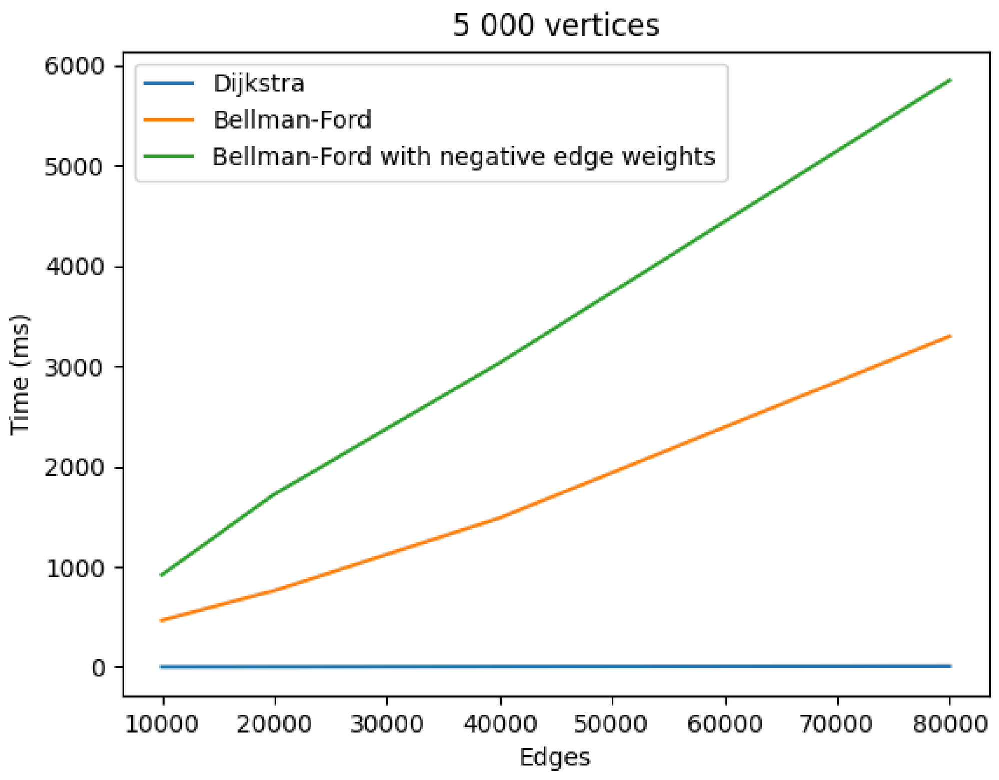

Figure 4 - 5k vertices with both positive and negative edge weights

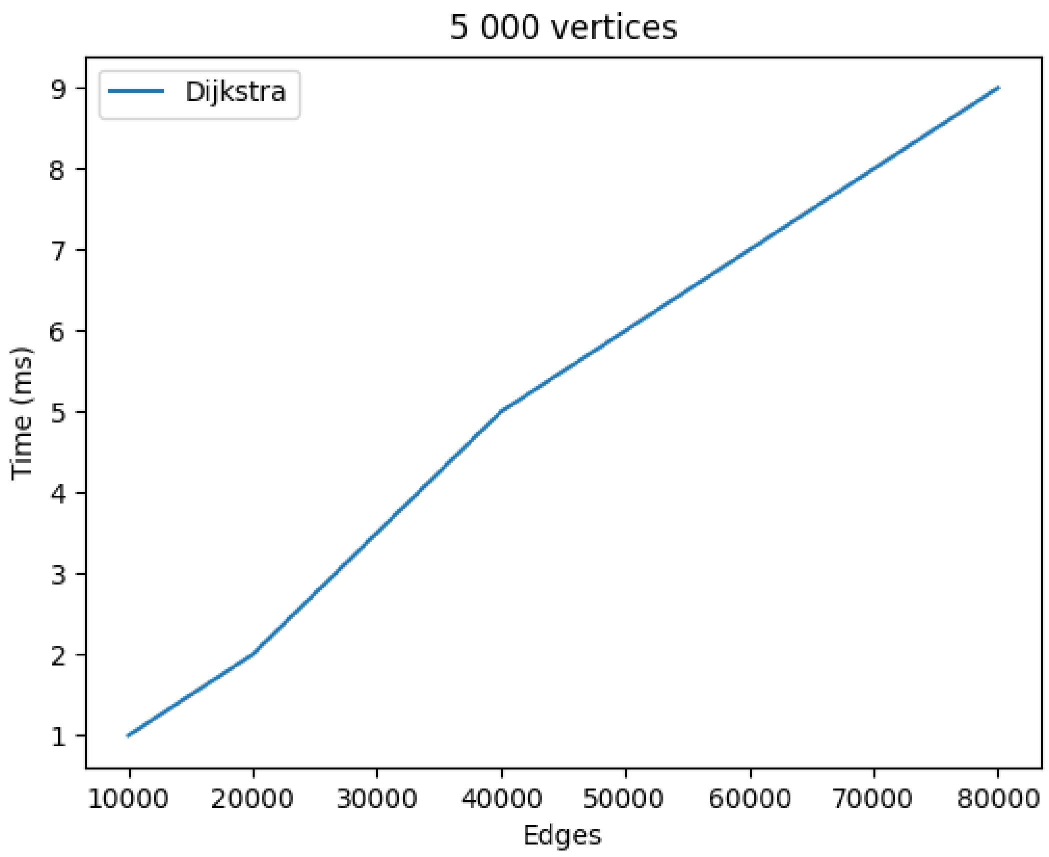

Figure 5 - 5k vertices showing the time growth of Dijkstra

Table 1 - Table showing recorded times

Vertices	Edges	Dijkstra (ms)	Bellman-Ford (ms)	Bellman-Ford (neg) (ms)
1 000	2 500	0	14	24
1 000	5 000	0	29	63
1 000	10 000	0	54	106
1 000	20 000	1	94	215
1 000	40 000	3	176	402
1 000	80 000	11	369	827
5 000	10 000	1	464	919
5 000	20 000	2	761	1 722
5 000	40 000	5	1 484	3 031
5 000	80 000	9	3 297	5 850