جامعة الإسكندرية
ALEXANDRIA UNIVERSITY
كلية الهندسة

Computer & Systems Engineering Department

Data Structures and Algorithms

Implementing Graph Algorithms (Dijkstra – Bellman Ford - Floyd Warshall)

Contributors:

	Name	ID
1	Adel Mahmoud Mohamed Abdelrahman	20010769
2	Amr Ahmed Abdelazim	20011037
3	Marwan Essam Eldin	20011859
4	Mohamed Amr Abdelfattah	20011675

Time complexity of algorithms

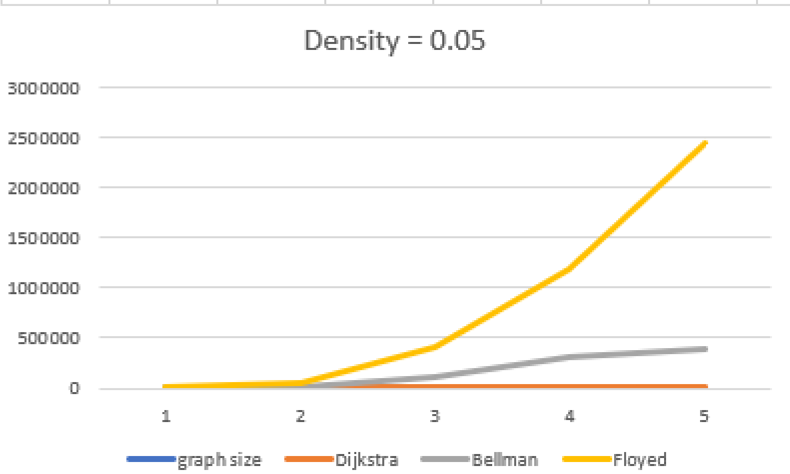
Algorithms	Time Analysis	Space Analysis
Dijkstra	O $((E+V)\log_2 V)$	O (V)
Bellman Ford	O $(V.E)$	O (V)
Floyd Warshall	O (V^3)	O (V^2)

Comparison between the 3 algorithms:

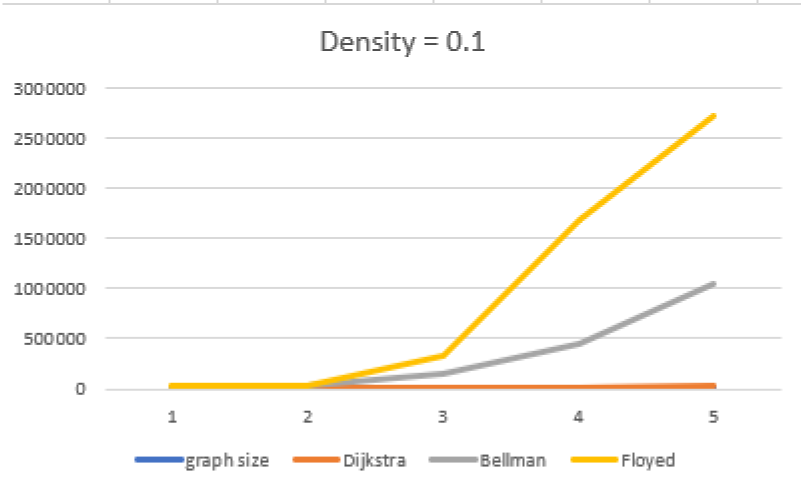
● Mean time to get the shortest path between 2 specific nodes under punch of different densities:

Density = 0.05

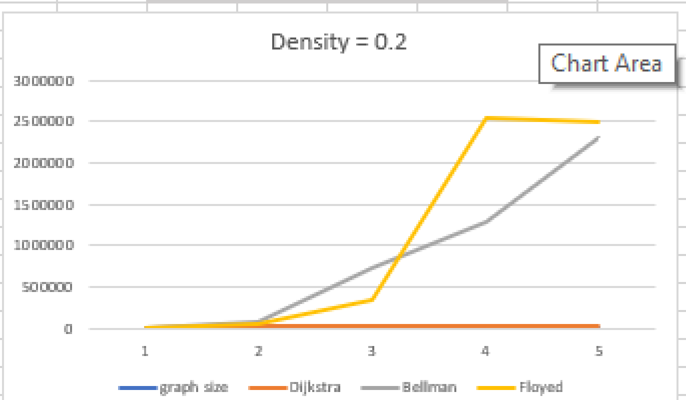
	Time(microseconds)		
graph size	Dijkstra	Bellman	Floyed
100	1099	7325.6	9519.8
200	4502	20783.3	43466.5
500	12064	119574	402850.3
750	18533	319897	1182756
1000	21099	394788	2443005
avg	**11459.4**	**172473.6**	**816319.5**



		Density =	0.1
	Time(microseconds)		
graph siz	Dijkstra	Bellman	Floyed
100	1657	13127	20860.8
200	4856	34351	39008.3
500	11740	141226	332666.5
750	18996	456918	1694214
1000	34253	1043356	2716835
avg	**14300.4**	**337795.6**	**960717**

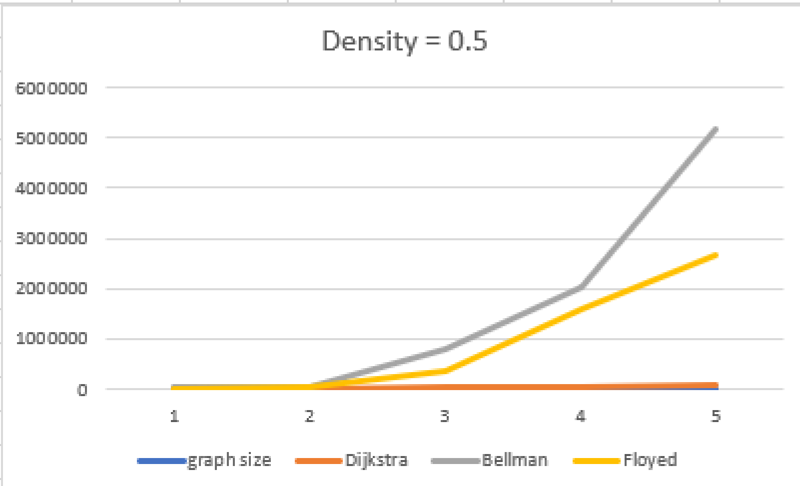


		Density =	0.2
	Time(microseconds)		
graph size	Dijkstra	Bellman	Floyed
100	4403	7692	9665.7
200	17394	70991	58450.2
500	21095	738825	335190
750	29954	1281353	2536527
1000	30286	2307399	2507222
avg	**20626.4**	**881252**	**1089411**



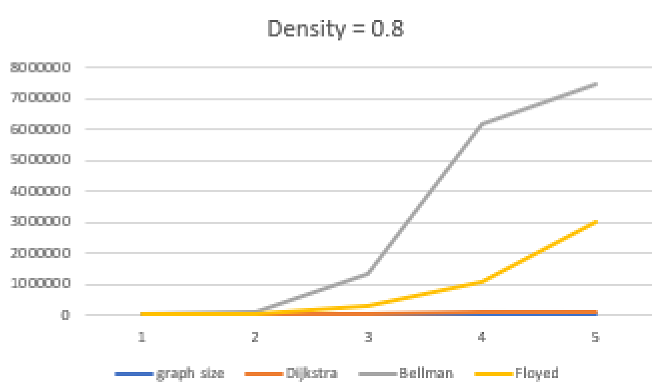
Density =	0.5	

	Time(microseconds)		
graph siz	Dijkstra	Bellman	Floyed
100	2923	18070	9444.5
200	8615	41951	32141.4
500	34441	797185	346461.4
750	33665	2037154	1587024
1000	52738	5177167	2650707
avg	**26476.4**	**1614305**	**925155.8**



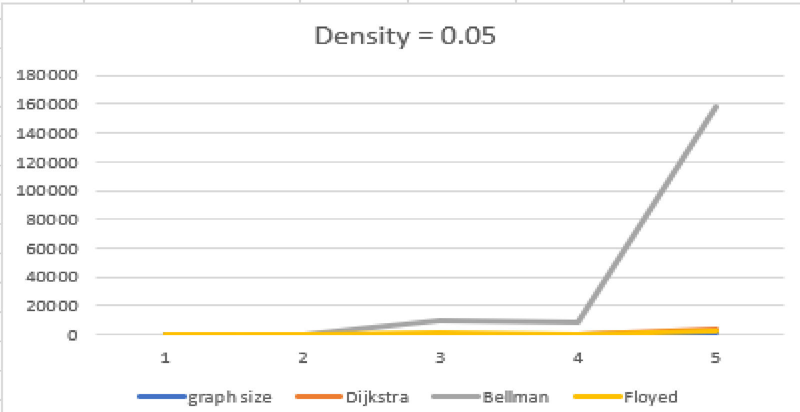
Density = (	0.8	

	Time(microseconds)		
graph size	Dijkstra	Bellman	Floyed
100	19458	63186	29769.5
200	20010	96330	43360.7
500	35524	1299350	311215.9
750	83848	6167584	1077212
1000	88829	7430757	3012181
avg	**49533.8**	**3011441**	**894747.8**

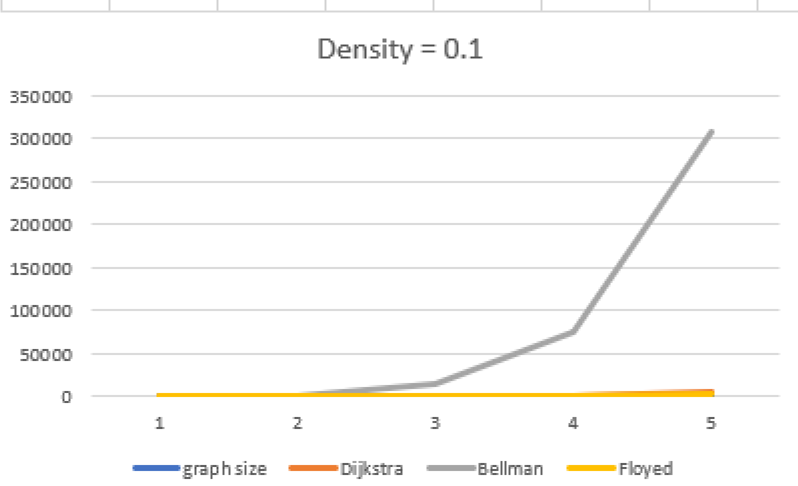


● **The time to get the shortest paths between all pairs of nodes:**

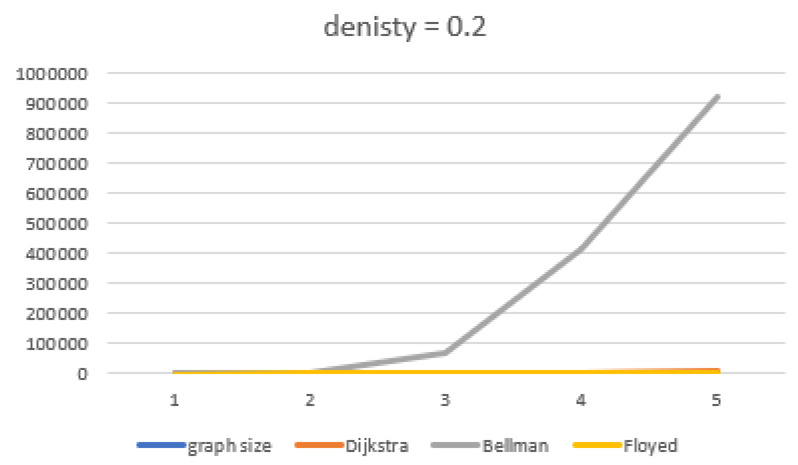
	Density =	0.05	
	Time(milliseconds)		
graph size	Dijkstra	Bellman	Floyed
100	14	52	11
200	86	432	37
500	680	9431	1130
750	754	8939	337
1000	3395	157662	2956
avg	**985.8**	**35303.2**	**894.2**



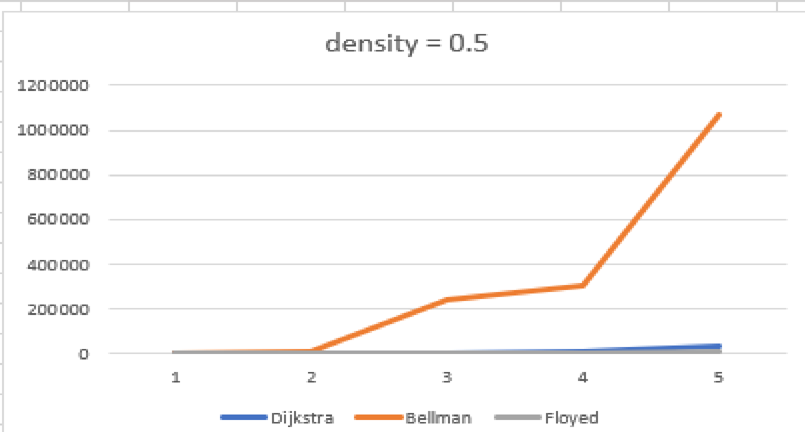
	Density =	0.1	
	Time(milliseconds)		
graph size	Dijkstra	Bellman	Floyed
100	29	66	8
200	172	1087	43
500	1210	14138	346
750	1946	74842	1176
1000	4441	308560	2987
avg	**1559.6**	**79738.6**	**912**



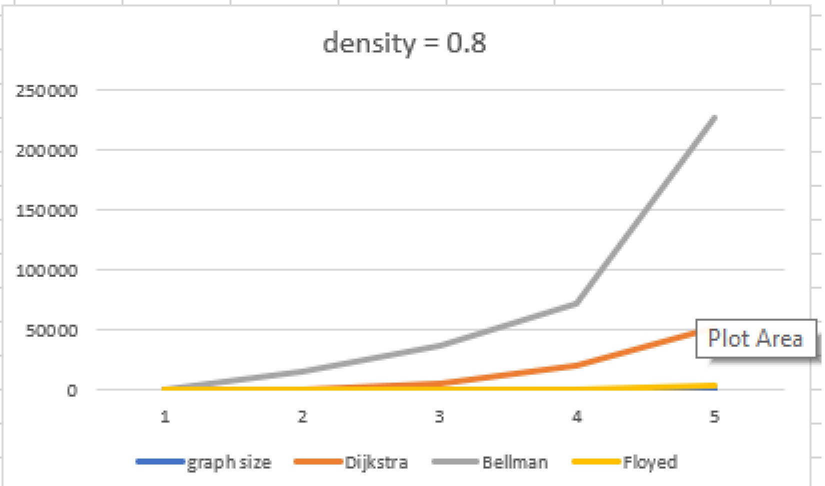
		Density =	0.2
		Time(milliseconds)	
graph size	Dijkstra	Bellman	Floyed
100	35	242	13
200	370	2620	52
500	1371	66760	351
750	3762	414520	1219
1000	11504	920115	2941
avg	3408.4	280851.4	915.2



		Density =	0.5
		Time(milliseconds)	
graph size	Dijkstra	Bellman	Floyed
100	47	440	15
200	323	11860	38
500	3232	242649	334
750	10501	306834	1143
1000	35576	1065731	6658
avg	9935.8	325502.9	1637.6



		Density =	0.8	
		Time(milliseconds)		
graph siz	Dijkstra	Bellman	Floyed	
100	246	301	19	
200	1258	15573	41	
500	5754	36835.6	385	
750	20856	71640.9	1360	
1000	52423	227296.9	3443	
avg	**16107.4**	**70329.48**	**1049.6**	



Conclusion:

- **<u>Time to get the shortest path between 2 specific nodes (us):</u>**
    1. As the graph size increases then the time to get the shortest path increases.
    2. The Dijkstra algorithm is the fastest of the three to get the shortest path between 2 specific nodes whatever the graph density is but its only downside is that it cannot handle negative weights.
    3. Bellman and Floyed can handle negative weights and furthermore they can detect negative cycles as well but to get the shortest path between 2 nodes we can tell that Bellman takes less time than Floyed in low and moderate graph dense.
    4. In high dense graphs we can find that Floyed is faster than Bellman.
    5. So, to sum up, in the application of finding the shortest path between 2 nodes then Dijkstra is the best of the three, then in low and moderate dense graphs Bellman is faster than Floyed but in high dense ones Floyed is faster.

- **<u>Time to get the shortest path between all pairs of nodes (ms):</u>**
    1. By increasing graph size, time to get all pairs shortest path increases.
    2. Floyd-Warshall algorithm is more efficient than the other two algorithms for getting all pairs shortest path.
    3. Bellman-Ford algorithm gives the worst performance especially for dense graphs.
    4. Dijkstra algorithm takes a slightly longer time than Floyd's at low density graphs, but for dense graphs the difference is significant, where Floyd's takes shorter time to complete.
    5. Floyd's and Bellman-Ford algorithms can handle negative edges, while Dijkstra's cannot.