Bellman-Ford’s Algorithm:

Bellman Ford's algorithm is used to find the shortest paths from the source vertex to all other vertices in a weighted graph. It depends on the following concept: Shortest path contains at most (n-1) edges, because the shortest path couldn't have a cycle.

So why shortest path shouldn't have a cycle ?
There is no need to pass a vertex again, because the shortest path to all other vertices could be found without the need for a second visit for any vertices.

Algorithm Steps:

The outer loop traverses from 0:n-1

- Loop over all edges, check if the next node distance > current node distance + edge weight, in this case update the next node distance to "current node distance + edge weight".

This algorithm depends on the relaxation principle where the shortest distance for all vertices is gradually replaced by more accurate values until eventually reaching the optimum solution. In the beginning all vertices have a distance of "Infinity", but only the distance of the source vertex = 0, then update all the connected vertices with the new distances (source vertex distance + edge weights), then apply the same concept for the new vertices with new distances and so on.

A very important application of Bellman Ford is to check if there is a negative cycle in the graph,

Time Complexity of Bellman Ford algorithm

Time Complexity of Bellman Ford algorithm is relatively high O(V . E), in case $E = V^2$, it becomes $O(V^3)$.
Let's discuss an optimized algorithm which Dijkstra’s Algorithm

Dijkstra’s Algorithm:

Dijkstra's algorithm has many variants but the most common one is to find the shortest paths from the source vertex to all other vertices in the graph.

Algorithm Steps:

- Set all vertices distances = infinity except for the source vertex, set the source distance = 0
- Push the source vertex in a min-priority queue in the form (distance , vertex), as the comparison in the min-priority queue will be according to vertices distances.
- Pop the vertex with the minimum distance from the priority queue (at first the popped vertex = source).

- Update the distances of the connected vertices to the popped vertex in case of "current vertex distance + edge weight < next vertex distance", then push the vertex with the new distance to the priority queue.
- If the popped vertex is visited before, just continue without using it.
- Apply the same algorithm again until the priority queue is empty.

Time Complexity of Dijkstra's Algorithm

is $O(V^2)$ but with min-priority queue it drops down to $O(V + Log(E))$.

Floyd Warshall’s Algorithm

Floyd Warshall's Algorithm is used to find the shortest paths between between all pairs of vertices in a graph, where each edge in the graph has a weight which is positive or negative. The biggest advantage of using this algorithm is that all the shortest distances between any vertices could be calculated in $O(V^3)$, where V is the number of vertices in a graph.

Algorithm Steps

For a graph with N vertices:

- Initialize the shortest paths between any 2 vertices with Infinity.
- Find all pair shortest paths that use 0 intermediate vertices, then find the shortest paths that use 1 intermediate vertex and so on.. until using all N vertices as intermediate nodes
- Minimize the shortest paths between any 2 pairs in the previous operation.
- For any 2 vertices (i,j) , one should actually minimize the distances between this pair using the first K nodes, so the shortest path will be: min(dist[i][k]+dist[k][j] , dist[i][j]) .
- dist[i][k] represents the shortest path that only uses the first K vertices, dist[j][k] represents the shortest path between the pair (j,k) . As the shortest path will be a concatenation of the shortest path from i to k , then from k to j.

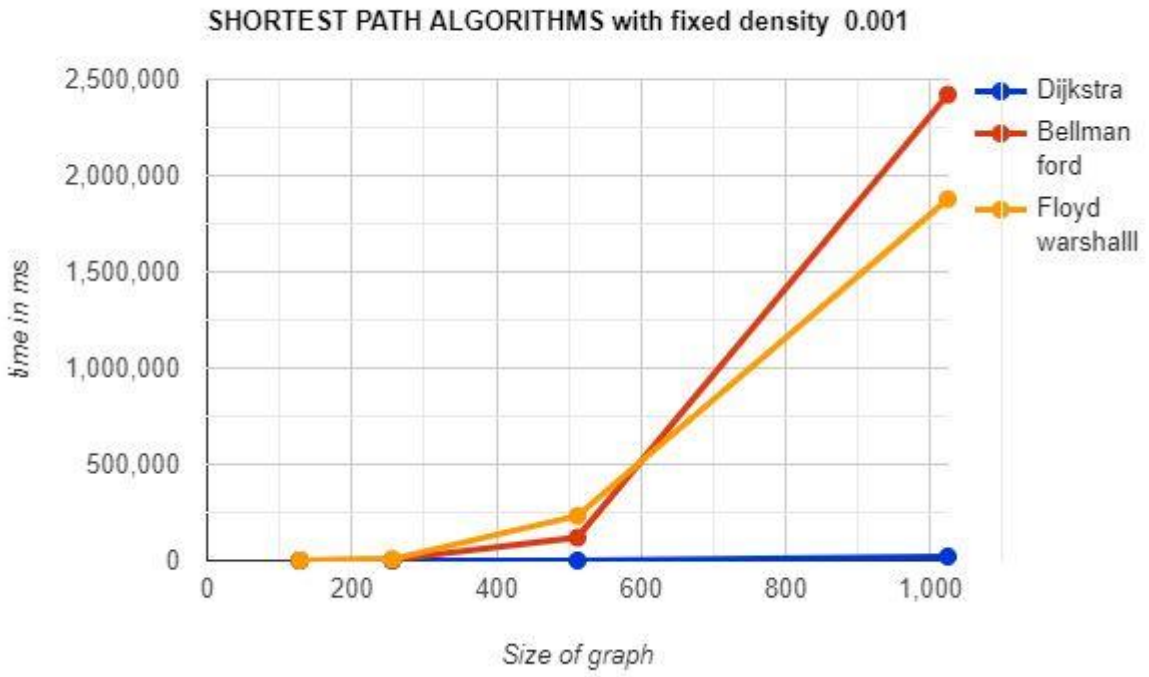
Time Complexity of Floyd Warshall’s Algorithm

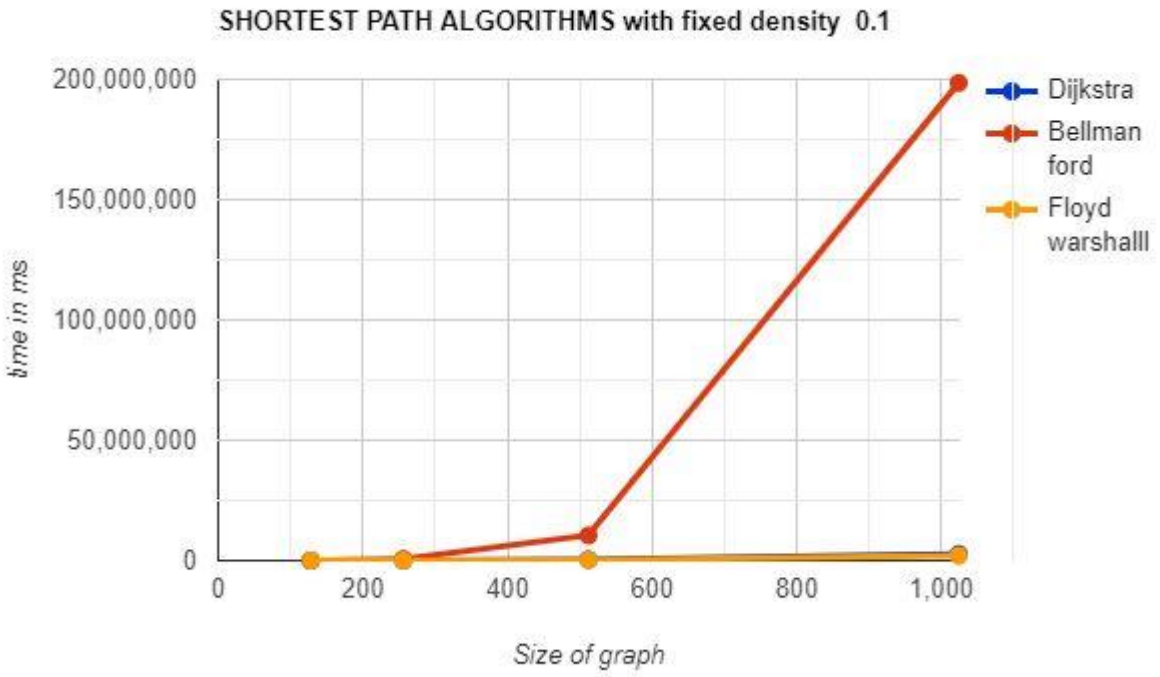
Time Complexity of Floyd Warshall's Algorithm is $O(V^3)$, where V is the number of vertices in a graph.

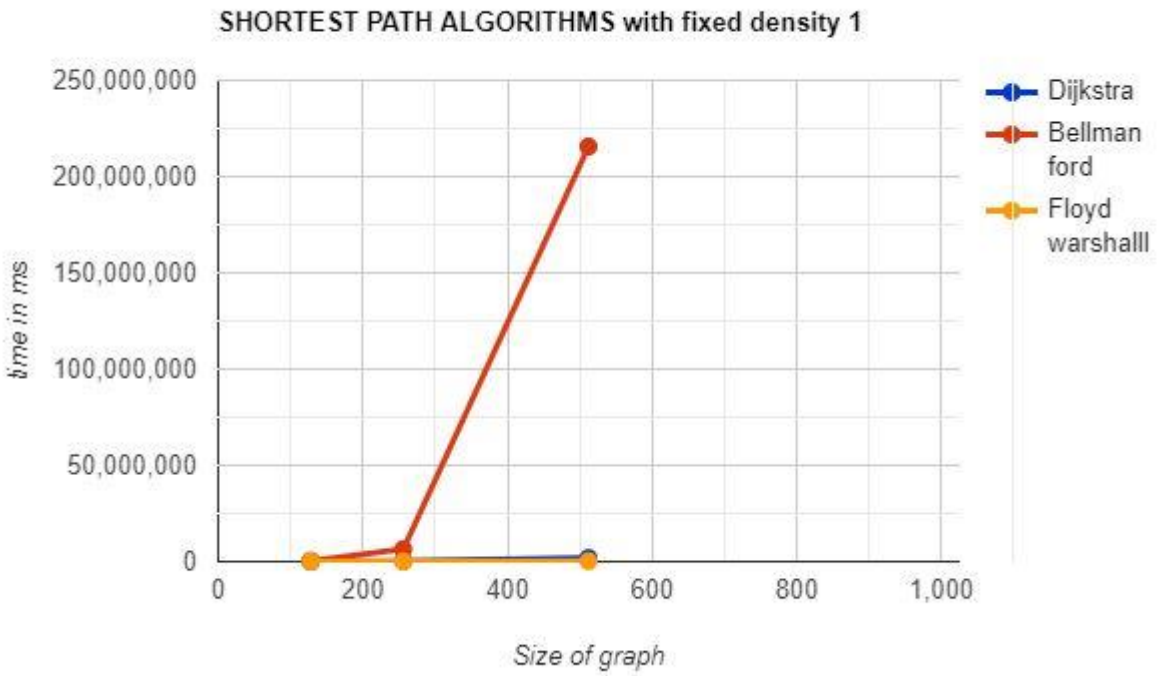
Algorithms Comparisons:

All pairs

Graph Size	Dijkstra	Bellman-Ford	Floyd Warshall
1024 graph with 0.001 density	21,046	2,420,562	1,877,675
512 graph with 0.001 density	439	118946	232361
1024 graph with 0.01 density	1,240,245	24,596,453	2,718,072
128 graph with 0.001 density	38	284	953
1024 graph with 0.1 density	2,699,925	198,515,994	2,054,257
128 graph with 0.01 density	429	4,554	1,539
128 graph with 1 density	27,647	362831	4127
256 graph with 0.01 density	7255	54529	14361
256 graph with 1 density	365511	6525943	32554
512 graph with 0.01 density	82636	945936	166793
512 graph with 1 density	1832662	215617193	237098
256 graph with 0.001 density	38	4,422	6,761
128 graph with 0.1 density	8375	56292	4297
256 graph with 0.1 density	59684	644613	28961
512 graph with 0.1 density	462940	10460830	238913

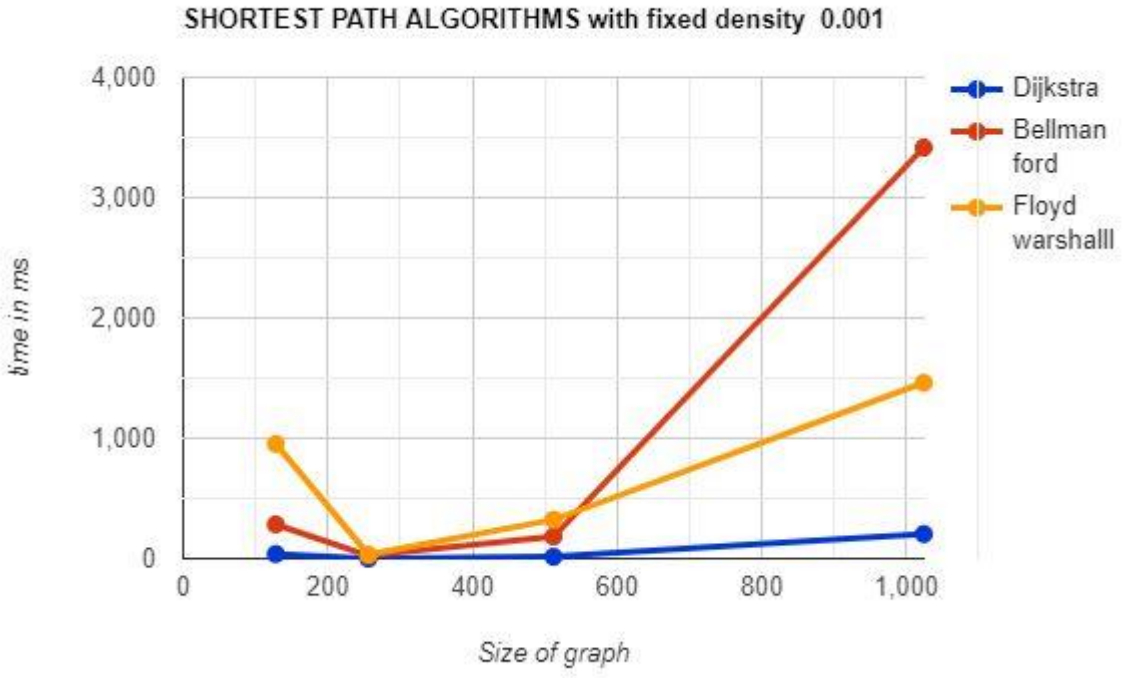


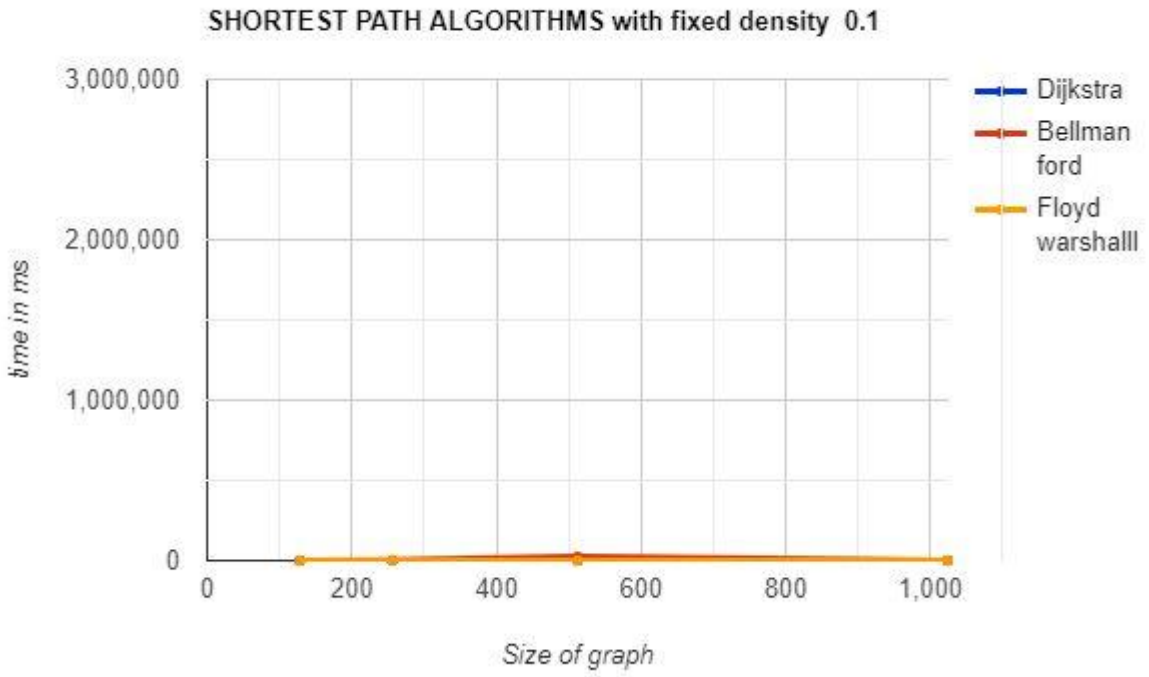

SHORTEST PATH ALGORITHMS with fixed density 0.1
200,000,000
150,000,000
100,000,000
50,000,000
0
time in ms
0
200
400
600
800
1,000
Size of graph
Dijkstra
Bellman ford
Floyd warshalll

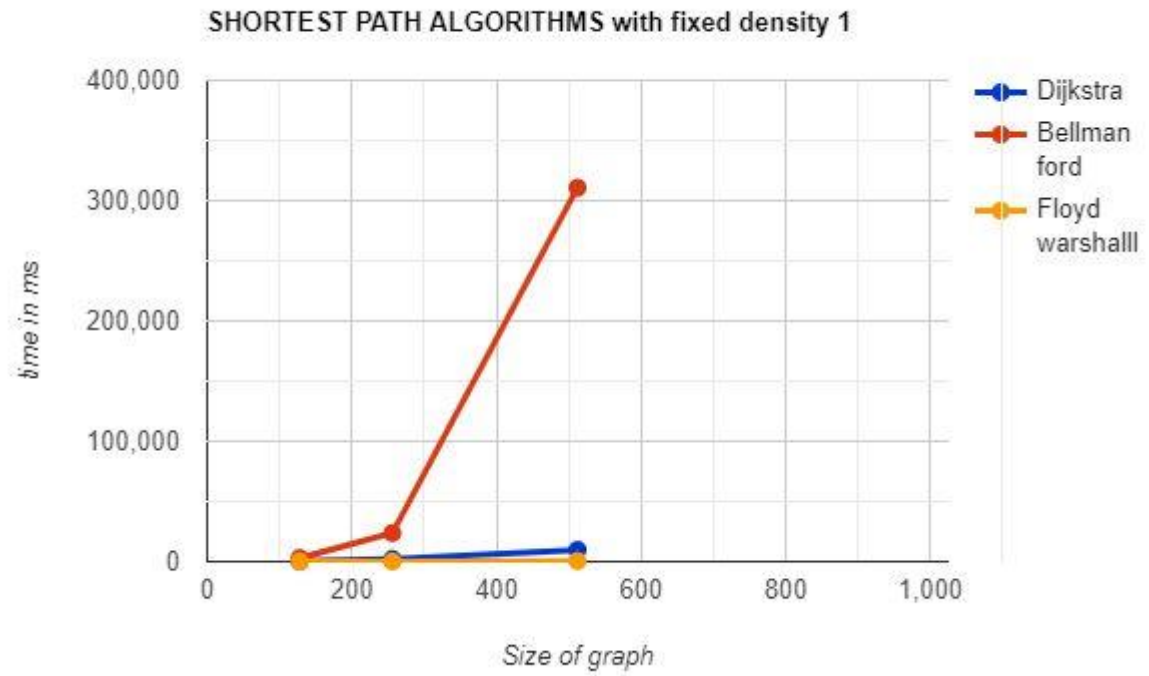

SHORTEST PATH ALGORITHMS with fixed density 1
250,000,000
200,000,000
150,000,000
100,000,000
50,000,000
0
time in ms
0
200
400
600
800
1,000
Size of graph
Dijkstra
Bellman ford
Floyd warshalll

Two Specific nodes

Graph Size	Dijkstra	Bellman-Ford	Floyd Warshall
1024 graph with 0.001 density	204	3417	1462
512 graph with 0.001 density	15	184	323
1024 graph with 0.01 density	2454	20034	1845
128 graph with 0.001 density	38	284	953
1024 graph with 0.1 density	12	1052	399
128 graph with 0.01 density	6	49	12
128 graph with 1 density	250	3024	34
256 graph with 0.01 density	51	304	44
256 graph with 1 density	2221	23735	153
512 graph with 0.01 density	287	2007	395
512 graph with 1 density	9782	310759	519
256 graph with 0.001 density	0	24	32
128 graph with 0.1 density	61	363	43
256 graph with 0.1 density	227	2519	156
512 graph with 0.1 density	846	24239	535




SHORTEST PATH ALGORITHMS with fixed density 0.1
3,000,000
2,000,000
1,000,000
0
time in ms
0
200
400
600
800
1,000
Size of graph
Dijkstra
Bellman ford
Floyd warshalll


SHORTEST PATH ALGORITHMS with fixed density 1
400,000
300,000
200,000
100,000
0
time in ms
0
200
400
600
800
1,000
Size of graph
Dijkstra
Bellman ford
Floyd warshalll