**Name: Saeem Ansari**

**Roll no: 2019430001**

**Subject: AAC**

**Experiment No 5**

**Aim:** Implementation of Bellman-Ford algorithm and Johnson’s algorithm

**Objectives:**

* Initialization of graph in Java programming language.
* Relaxation of distances between nodes in C/C++ programming language.
* Finding shortest path from source node to every other node.

**Methodology:**

* Bellman Ford Algorithm:

Like Dijkstra's shortest path algorithm, the Bellman-Ford algorithm is guaranteed to find the shortest path in a graph. Though it is slower than Dijkstra's algorithm, Bellman-Ford is capable of handling graphs that contain negative edge weights, so it is more versatile. It is worth noting that if there exists a negative cycle in the graph, then there is no shortest path. Going around the negative cycle an infinite number of times would continue to decrease the cost of the path. Because of this, Bellman-Ford can also detect negative cycles which is a useful feature.

* Johnson’s Algorithm

Johnson's algorithm is a shortest path algorithm that deals with the all pairs shortest path problem. The all pairs shortest path problem takes in a graph with vertices and edges, and it outputs the shortest path between every pair of vertices in that graph. Johnson's algorithm is very similar to the Floyd-Warshall algorithm; however, Floyd-Warshall is most effective for dense graphs, while Johnson's algorithm is most effective for sparse graphs.

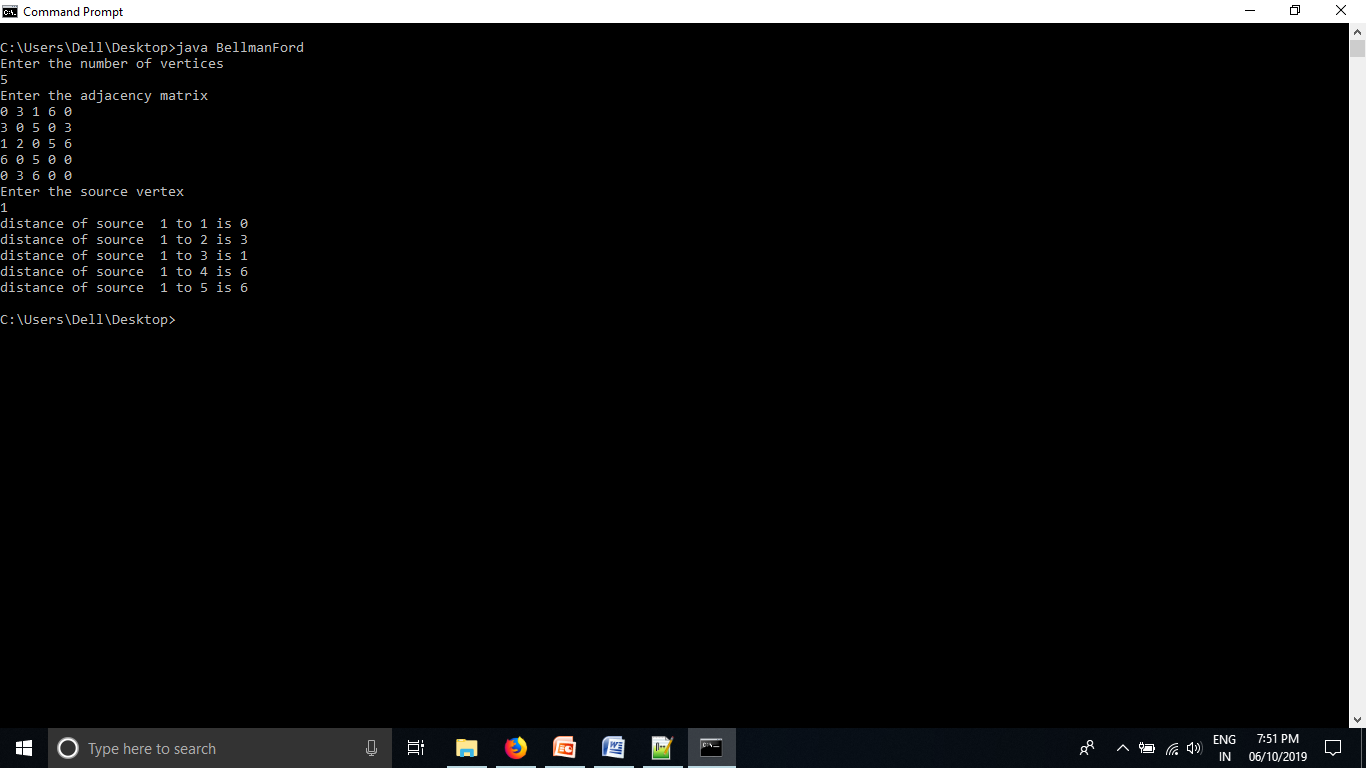
**Time Complexity:**

Bellman Ford: Initialization takes O(V)time, relaxation takes O(E(V−1))=O(VE)time, and detecting negative cycles takes O(E)time. Overall, the runtime of Bellman-Ford is O(VE). For certain graphs, only one iteration is needed, and hence in the best case scenario, only O(∣E∣)is needed

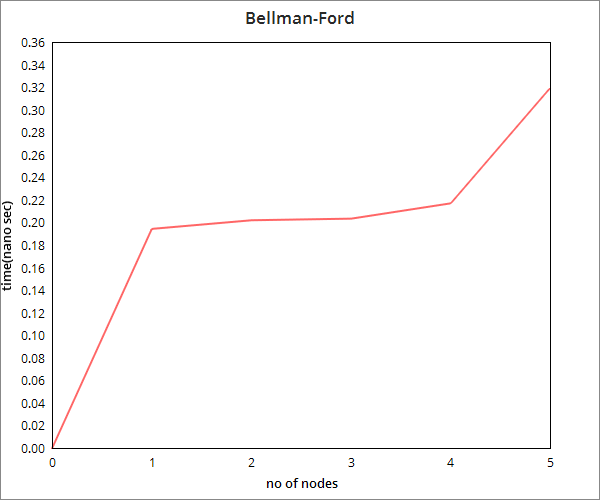
Johnsons Algorithm: The first step in the algorithm is simple and runs in O(V) time because it needs to make a new edge to all vertices in the graph. The second step of the algorithm, Bellman-Ford, runs in O(VE) time, as does Bellman-Ford. The final step of the algorithm is Dijkstra's algorithm run on all VVV vertices which is O(V2log(V)+VE).

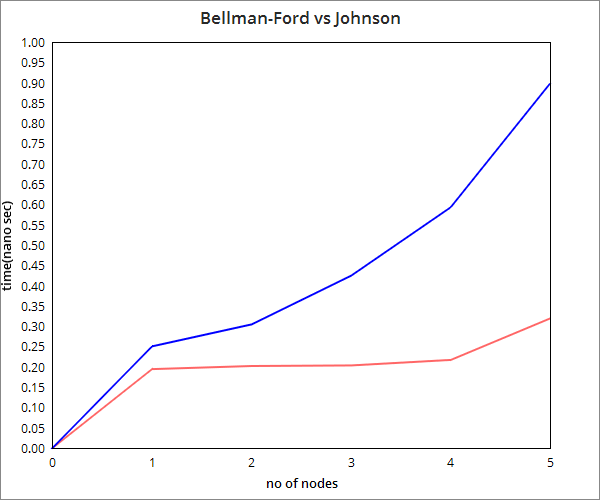
**Results:**

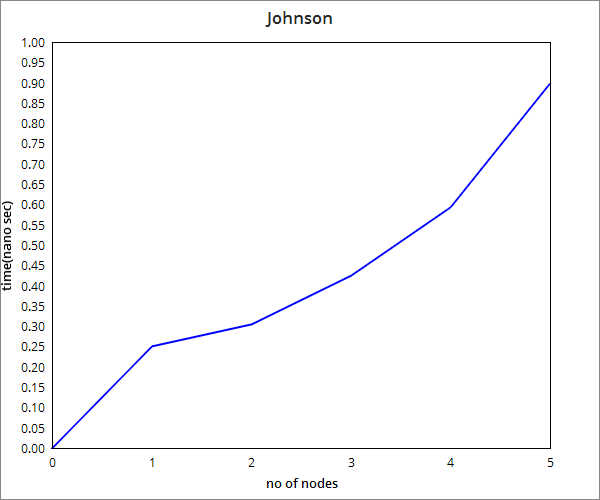
Bellman Ford

****

Johnsons Algorithm







**Conclusion:**

It can be observed that Bellman Ford doesn’t follow a greedy approach like Dijkstra’s and it works on negative edge weights which is not the case with the latter. Johnson’s Algorithm works on integrating the Bellman Ford as well as the Dijkstra’s shortest path algorithm.