Name : Ansari M.Saeem M.Saleem

Uid : 2019430001

Subject : AAC

Expt no : 5

Aim : Write a program to implement and Analysis Bellman-Ford & Johnson Algorithm

Aim:

Write a program to Implementation and Analysis of Bellman-Ford algorithm and Johnson’s algorithm

Objectives:

- Initialization and insertion of element in graph using C/Java/python 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.

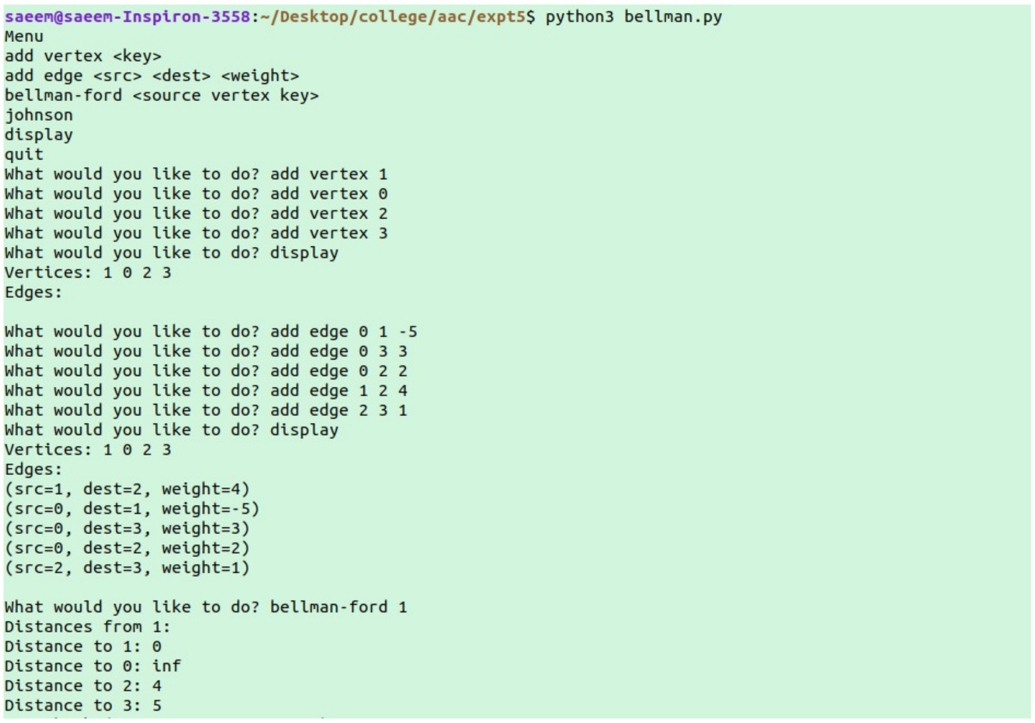
Time Complexity:

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 run time 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.

Johnson’s 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 Vvertices which is O(V2log(V)+VE).

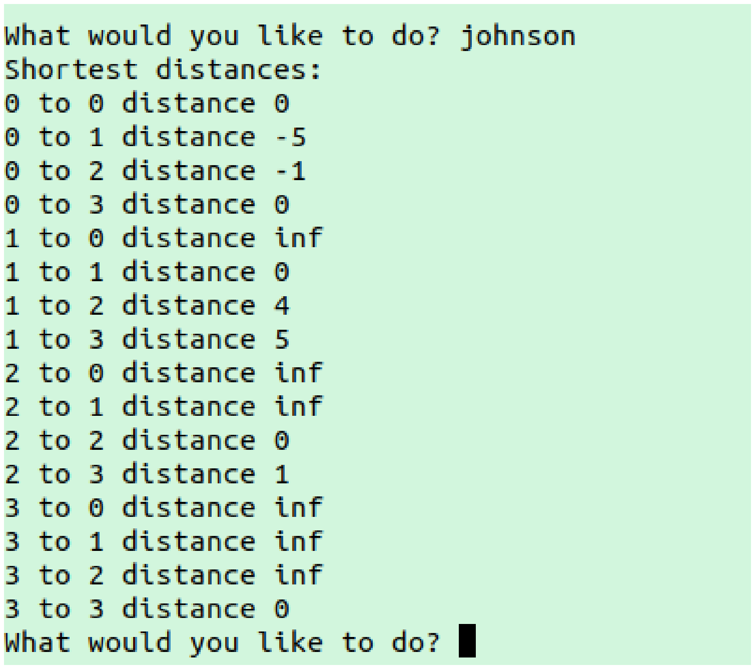
Results:

Bellman Ford

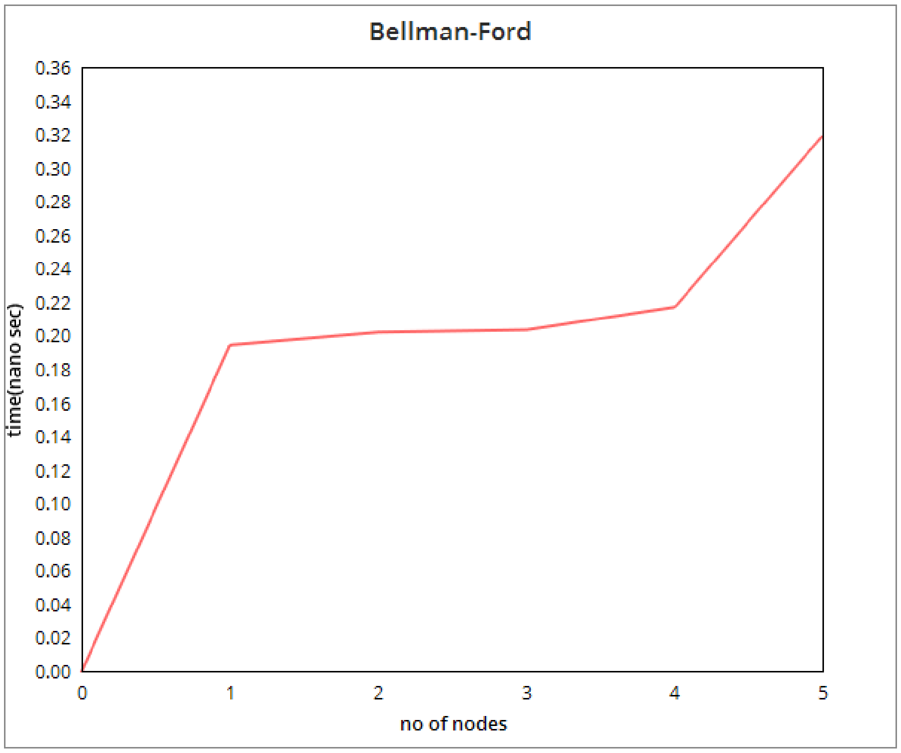


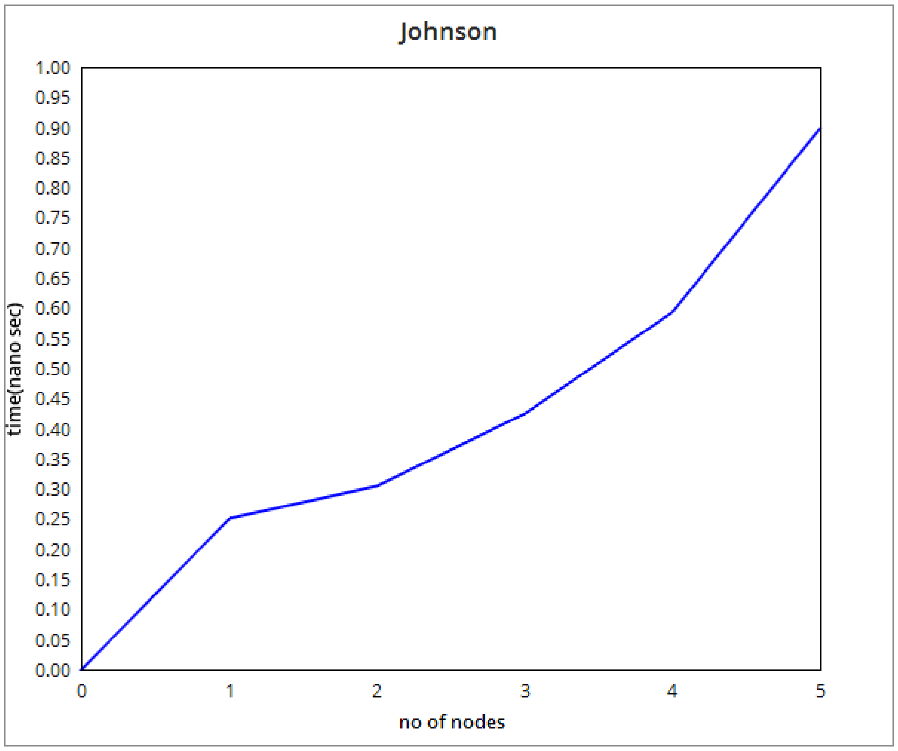
```
saeem@saeem-Inspiron-3558:~/Desktop/college/aac/expt5$ python3 bellman.py
Menu
add vertex <key>
add edge <src> <dest> <weight>
bellman-ford <source vertex key>
johnson
display
quit
What would you like to do? add vertex 1
What would you like to do? add vertex 0
What would you like to do? add vertex 2
What would you like to do? add vertex 3
What would you like to do? display
Vertices: 1 0 2 3
Edges:

What would you like to do? add edge 0 1 -5
What would you like to do? add edge 0 3 3
What would you like to do? add edge 0 2 2
What would you like to do? add edge 1 2 4
What would you like to do? add edge 2 3 1
What would you like to do? display
Vertices: 1 0 2 3
Edges:
(src=1, dest=2, weight=4)
(src=0, dest=1, weight=-5)
(src=0, dest=3, weight=3)
(src=0, dest=2, weight=2)
(src=2, dest=3, weight=1)

What would you like to do? bellman-ford 1
Distances from 1:
Distance to 1: 0
Distance to 0: inf
Distance to 2: 4
Distance to 3: 5
```

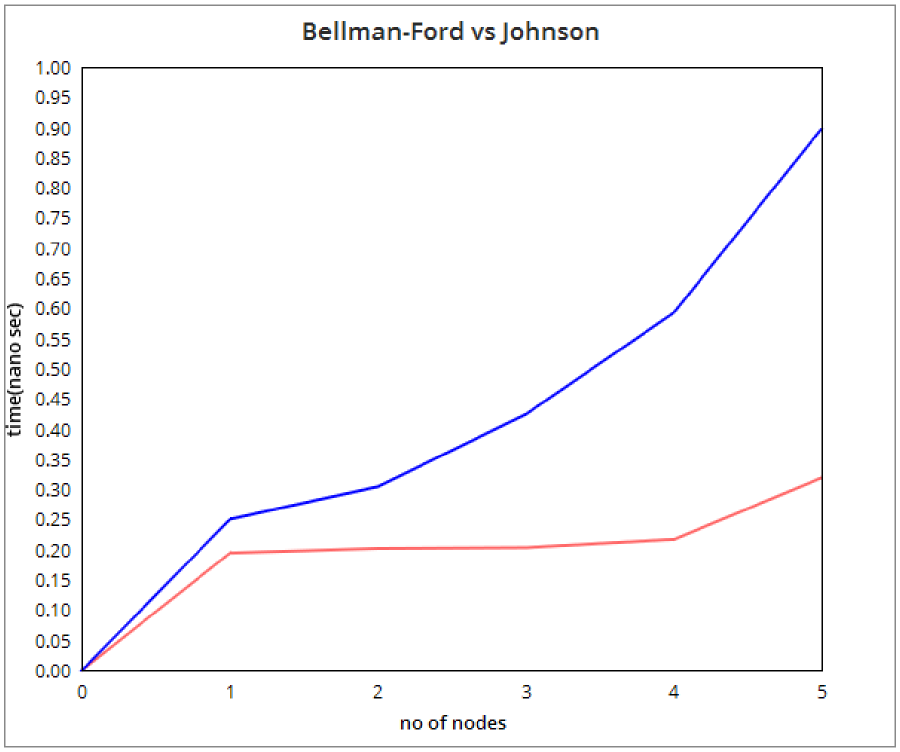
Johnsons Algorithm



```
What would you like to do? johnson
Shortest distances:
0 to 0 distance 0
0 to 1 distance -5
0 to 2 distance -1
0 to 3 distance 0
1 to 0 distance inf
1 to 1 distance 0
1 to 2 distance 4
1 to 3 distance 5
2 to 0 distance inf
2 to 1 distance inf
2 to 2 distance 0
2 to 3 distance 1
3 to 0 distance inf
3 to 1 distance inf
3 to 2 distance inf
3 to 3 distance 0
What would you like to do?
```


Bellman-Ford
time(nano sec)
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
0.18
0.20
0.22
0.24
0.26
0.28
0.30
0.32
0.34
0.36
0
1
2
3
4
5
no of nodes


Johnson
time(nano sec)
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.45
0.50
0.55
0.60
0.65
0.70
0.75
0.80
0.85
0.90
0.95
1.00
0
1
2
3
4
5
no of nodes



Conclusion:

Here we can conclude that Bellman-Ford works on negative edge weight which is the advantage of Dijkstra's algorithm. Johnson's Algorithm is based on Bellman-Ford algorithm and Dijkstra's algorithm. Bellman-Ford required less time to find shortest path but on the other hand Johnson's algorithm required more time as we increased the density of graph.