Name : Ansari M.Saeem M.Saleem

Uid : 2019430001

Subject : NAD

Expt no : 3

Aim : Write a program to implement dynamic routing strategy in finding optimal path for data transmission (Bellman ford algorithm).

Aim:

Write a program to implement dynamic routing strategy in finding optimal path for data transmission (Bellman ford algorithm).

Objectives:

- To apply dynamic routing strategy to find optimal path for data transmission in a network.
- To find the shortest path from source node to every other node even with negative edge using Bellman-Ford algorithm.

Theory:

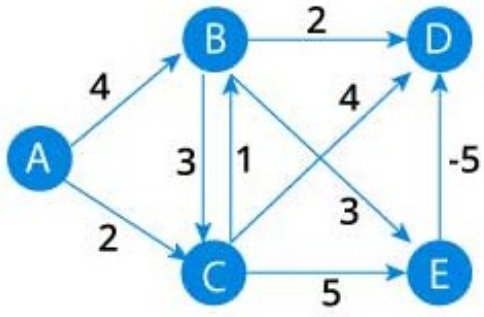
Dynamic Routing is also called adaptive routing, is a process where a router can forward data via a different route or given destination based on the current conditions of the communication circuits within a system. Routing algorithms view the network as a graph.

Bellman-Ford algorithm is guaranteed to find the shortest path in a graph. 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.

Methodology :

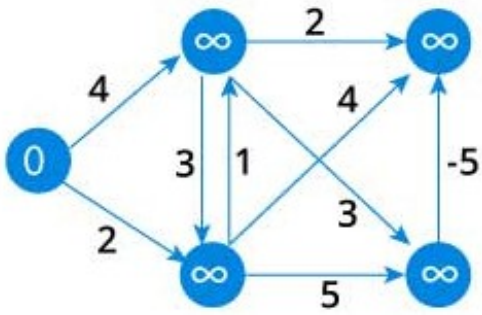
1

Start with a weighted graph



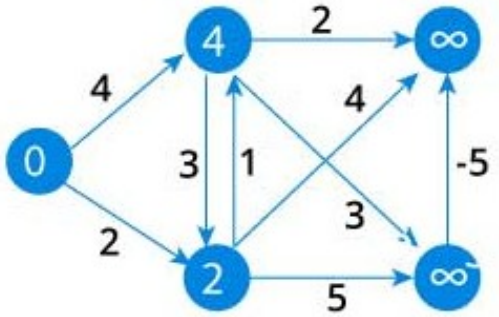
2

Choose a starting vertex and assign infinity path values to all other vertices



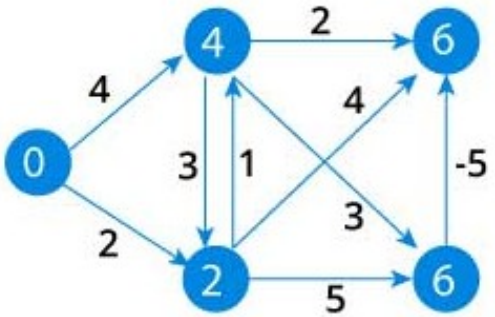
3

Visit each edge and relax the path distances if they are inaccurate



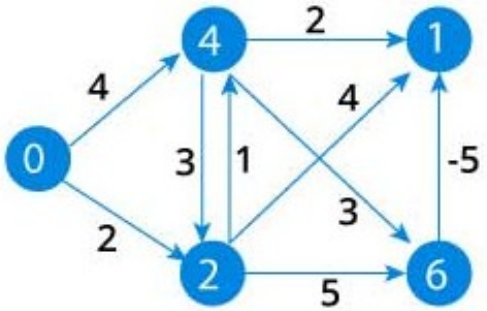
4

We need to do this V times because in the worst case, a vertex's path length might need to be readjusted V times



5

Notice how the vertex at the top right corner had its path length adjusted



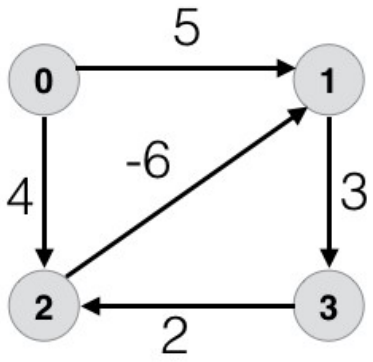
6

After all the vertices have their path lengths, we check if a negative cycle is present.

A	B	C	D	E
0	∞	∞	∞	∞
0	4	2	∞	∞
0	3	2	6	6
0	3	2	1	6
0	3	2	1	6

Results :

Input Graph :



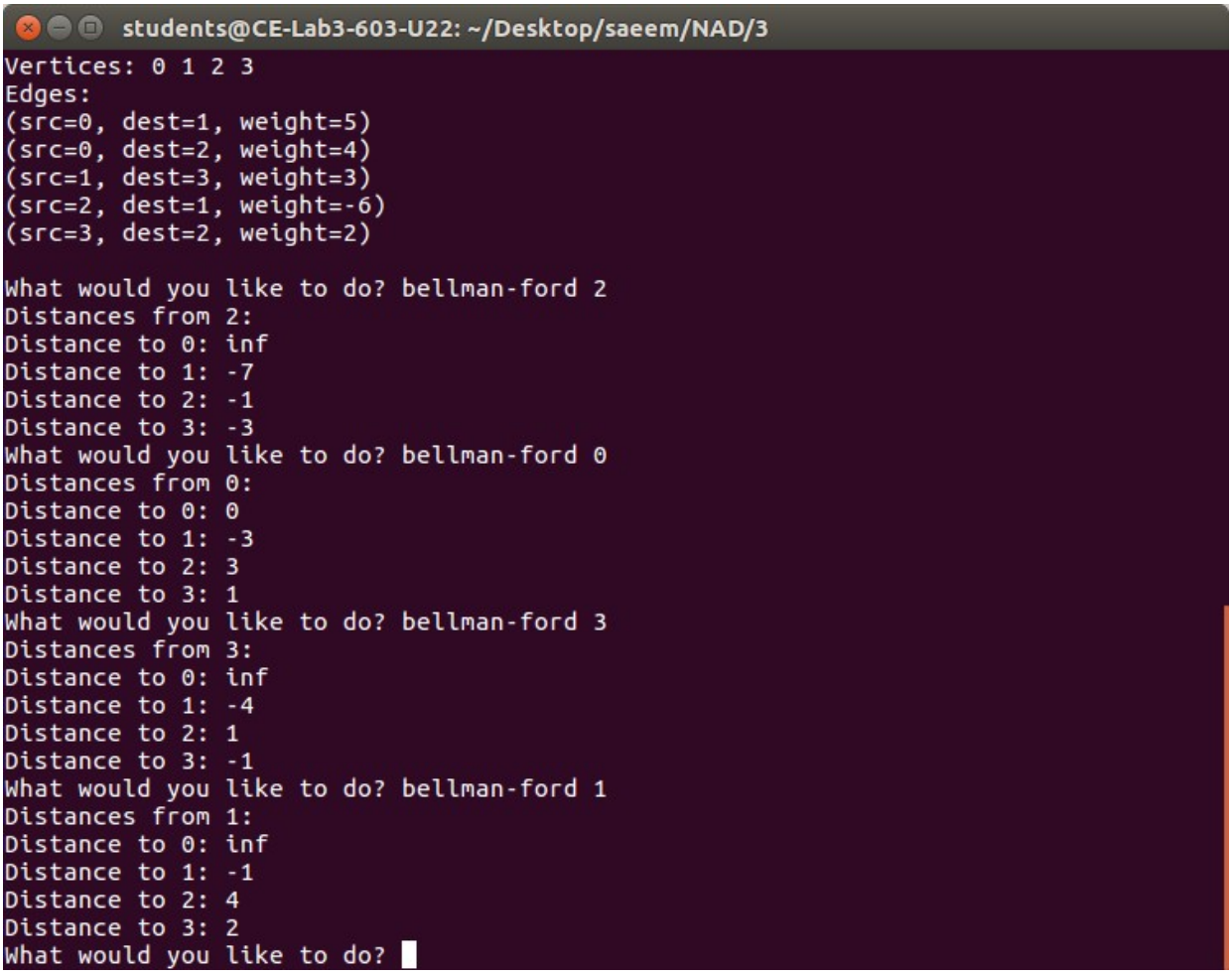
Output :

```
students@CE-Lab3-603-U22: ~/Desktop/saeem/NAD/3

students@CE-Lab3-603-U22:~/Desktop/saeem/NAD/3$ python bellman.py
Menu
add vertex <key>
add edge <src> <dest> <weight>
bellman-ford <source vertex key>
display
quit
What would you like to do? add vertex 0
What would you like to do? add vertex 1
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: 0 1 2 3
Edges:

What would you like to do? add edge 0 1 5
What would you like to do? add edge 0 4 4
Vertex 4 does not exist.
What would you like to do? add edge 0 2 4
What would you like to do? add edge 1 3 3
What would you like to do? add edge 2 1 -6
What would you like to do? add edge 3 2 2
What would you like to do? display
Vertices: 0 1 2 3
Edges:
(src=0, dest=1, weight=5)
(src=0, dest=2, weight=4)
(src=1, dest=3, weight=3)
(src=2, dest=1, weight=-6)
(src=3, dest=2, weight=2)

What would you like to do? bellman-ford 2
```



Conclusion:

Here we can conclude that Bellman-Ford works on negative edge weight and can be used to find the optimal path for data transmission in a network.