Bellman Ford Algorithm :

It is a single source shortest path algorithm. This algorithm is used to find the shortest distance from the single vertex to all the other vertices of a weighted graph. There are various other algorithms used to find the shortest path like Dijkstra algorithm etc. If the weighted graph contains the negative weight ~~cycle~~ values, then the Dijkstra algorithm does not confirm whether it produces the correct answer or not. In contrast to Dijkstra algorithm, bellman ford algorithm guarantees the correct answer even if the weighted graph contains the negative weight values.

Rule of this algorithm :

we will go on relaxing all the edges (n-1) times, where 'n' is the number of vertices.

Relaxing means :

$$\text{if } (d(u) + c(u, v) < d(v))$$

then

$$d(v) = d(u) + c(u, v)$$

Steps for finding the shortest distance to all vertices from the source using Bellman-Ford algorithm.

1) This step initialises distances from the source to all vertices as infinite and distance to the source itself as 0. Create an array dist [ ] of size |V| with all values as infinite except dist [src] where src is source vertex.

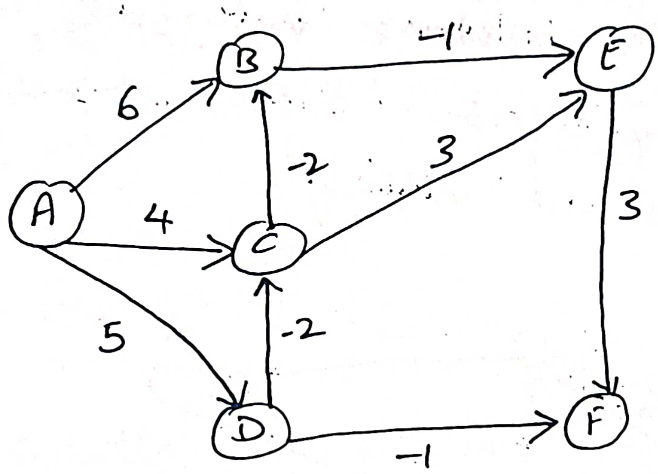
2) This step calculates shortest distances. Do the following for |V|-1 times where |V| is the number of vertices in given graph. Do this for each edge (u-v).

* If dist (v) > dist (u) + weight of edge (u,v) then update dist (v) to
  dist (v) = dist (u) + weight of edge (u,v)

3) This step reports if there is a negative weight cycle in the graph.

* If we iterate through all edges one more time i.e after (|V|-1) iterations and get a shortest path for any vertex, then it indicates there is a negative cycle.

Example :-

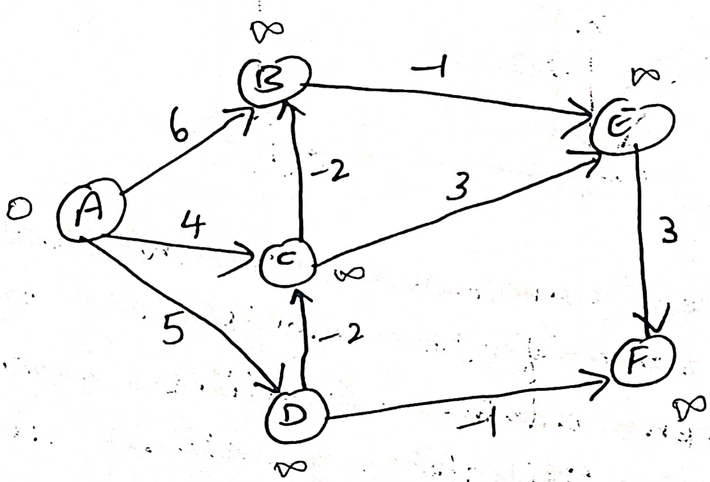


As we can observe in the above graph that some of the weights are negative. The above graph contains 6 vertices so we will go on relaxing till the 5 vertices. Here, we will relax all the edges 5 times. The loop will iterate 5 times to get the correct answer. If the loop is iterated more than 5 times then also the answer will be the same. i.e, there would be no change in the distance b/w the vertices.

To find the shortest path of the above graph, the first step is note down all the edges which are given below:

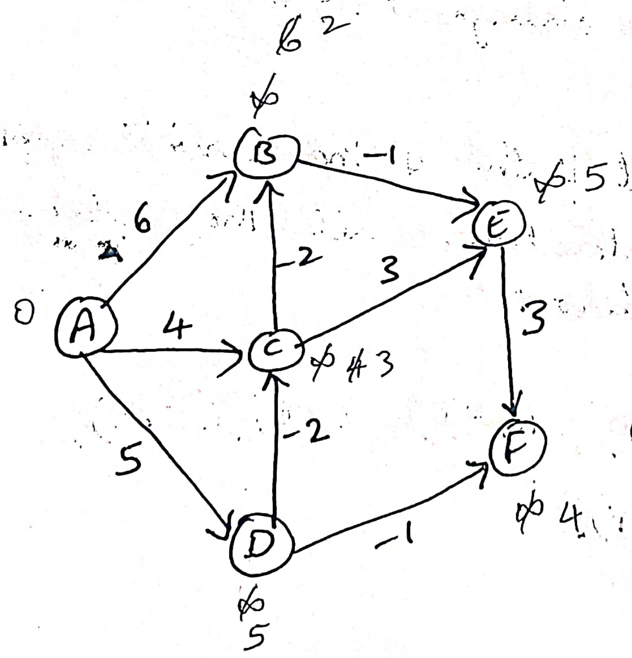
(A, B), (A, C), (A, D), (B, E), (C, B), (C, E), (D, C), (D, F), (E, F).

Let us consider the source vertex as 'A', therefore, the distance value at vertex A is 0 and the distance value at all the other vertices as infinity.



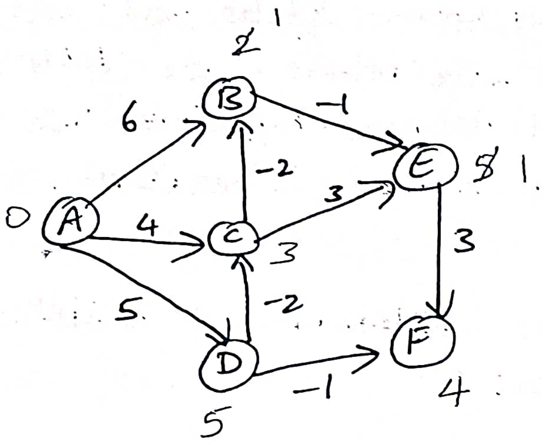
Since the graph has six vertices so it will have five iterations.

First iteration :



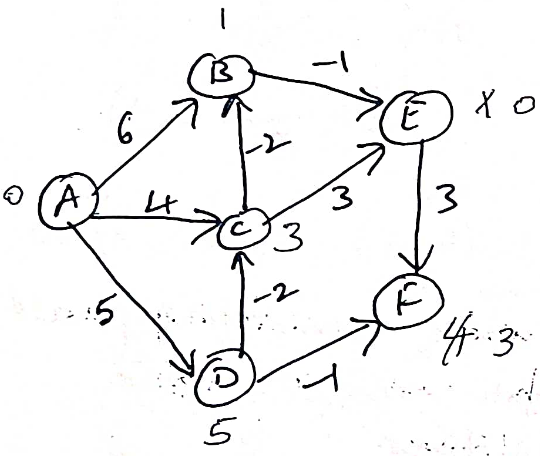
A	B	C	D	E	F
0	2	3	5	5	4

Second iteration :-



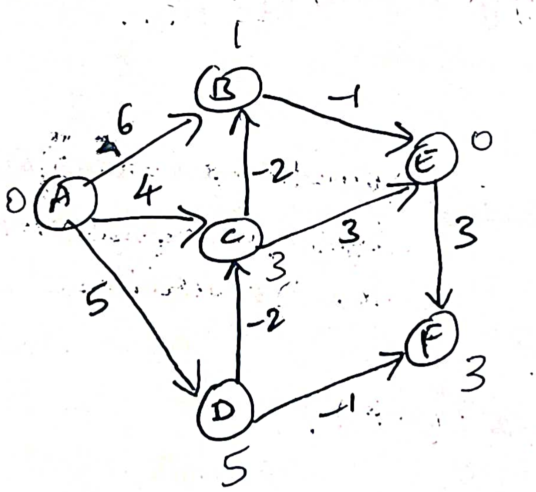
A	B	C	D	E	F
0	1	3	5	1	4

Third iteration :



A	B	C	D	E	F
0	1	3	5	0	3

Fourth iteration :-



A	B	C	D	E	F
0	1	3	5	0	3

we will perform the same steps as we did in the previous iterations. we will observe that there will be no updation in the distance of vertices. Hence iteration 5 is not performed.

The following are the distances of vertices from the source vertex :

A : 0

B : 1

C : 3

D : 5

E : 0

F : 3

<u>Time Complexity</u> : we are considering all the edges in the graph and we are relaxing it for $(n-1)$ times.

$O[(\text{Edges}) \cdot (\text{Relaxing the edges})]$

$\therefore O[E \cdot (n-1)]$

$O[n \cdot (n-1)]$ $\qquad$ $\because$ Sometimes no. of edges is equal to no. of vertices.

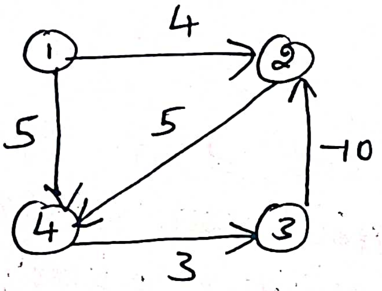
$O(n^2)$

For complete graph, it is $O(n^3)$.

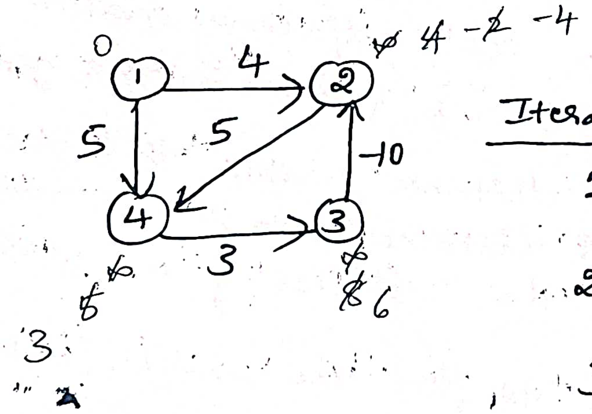
Drawback of Bellman Ford :

The bellmanford algorithm does not produce a correct answer if the sum of the edges of a cycle is negative.

Let us understand this property through an example.



Edge list : (1, 2), (1, 4), (2, 4), (3, 2), (4, 3)



Iterations	1	2	3	4
1	0	4	8	5
2	0	-2	8	5
3	0	-2	6	3
4	0	-4	6	3

Since the graph contains 4 vertices, according to the bellmanford algorithm, there should be only 3 iterations. If we try to perform 4^{th} iteration on the graph, the distance of the vertices from the given graph should not change. If the distance varies, it means that the bellman ford algorithm is not providing the correct answer.

In this case, when we perform 4^{th} iteration, the value of the vertex is updated. so, we conclude that the bellman ford algorithm does not work when the graph contains the negative weight cycle.

~~Algorithm Bellman-Ford (V, a[][], s)~~
~~// n - no. of nodes in the graph, a - two dimensional array,~~
~~// s - source node~~
~~Step 1:~~

Algorithm BellmanFord (v, cost, d, n)

// Purpose : To compute shortest distance from a given node to all other nodes of the graph.

// Input : v is the source node
cost is the cost adjacency matrix
n is the no. of nodes in the graph.

// Output : d is the distance matrix that gives the shortest distance from source node v to all other vertices of the graph

Step 1 : Initialization to find the shortest distance.

for i ← 1 to n do
    d[i] ← cost[v, i]
end for

Step 2 :- Compute the shortest distance
for k ← 2 to n-1 do

```
        for each u such that u ≠ v and u
has atleast one incoming edge
            for each (i, u) in the graph
                if (d[i] + cost[i, u] < d[u])
                    d[u] = d[i] + cost[i, u]
                end if
            end for
        end for
end for
```