Bellman Ford Algorithm

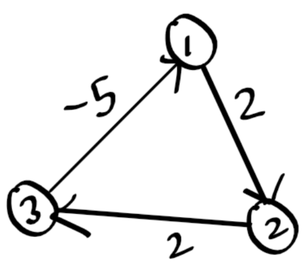
① Differences b/w Dijkstra's and Bellman Ford algorithms

Dijkstra's	Bellman Ford
→ Cannot be applied on graphs containing edges with negative weights	→ Can be applied on graphs containing edges with negative weights
→ Time complexity is: $O((V+E)\log V)$	→ Time complexity is: $O(VE)$
* Better than Bellman Ford Time complexity	* Worse than Dijkstras Time complexity

② Drawback of Bellman Ford algorithm

The algorithm can't be applied to graphs containing negative cycles.

Eg:



$\sum_{ij \in C} c_{ij} < 0$ (c_{ij}: cost)

↓ directed cycle

Here $-5+2+2 < 0$

$-1 < 0$

∴ Can't apply B.F.A.

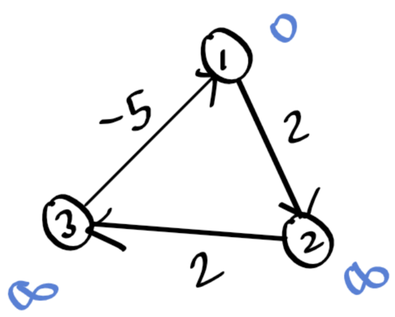
Reason:

Consider Source = 1

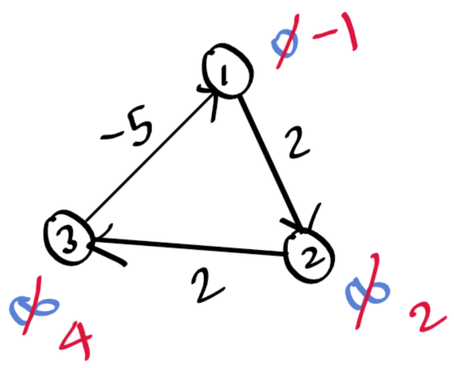
V = 3 E = 3

∴ Iterations should be $|V|-1 = 3-1 = 2$

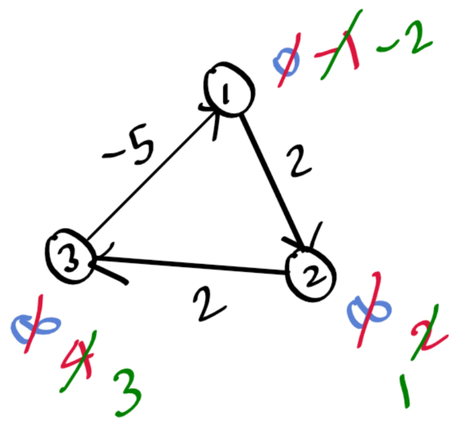
Initially:



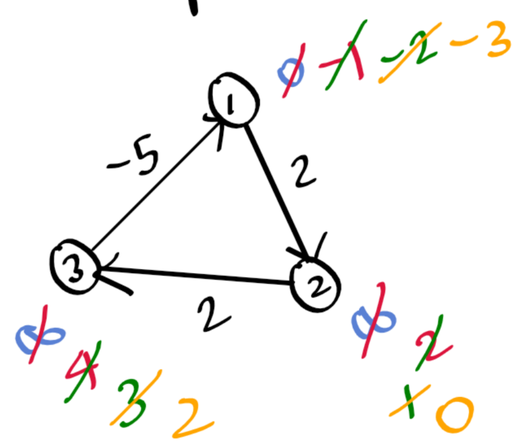
1st iteration:



2nd iteration



* If "3rd iteration" was there, we get a different shortest path



* we go into an ∞ loop of shortest paths.

③ Working procedure:

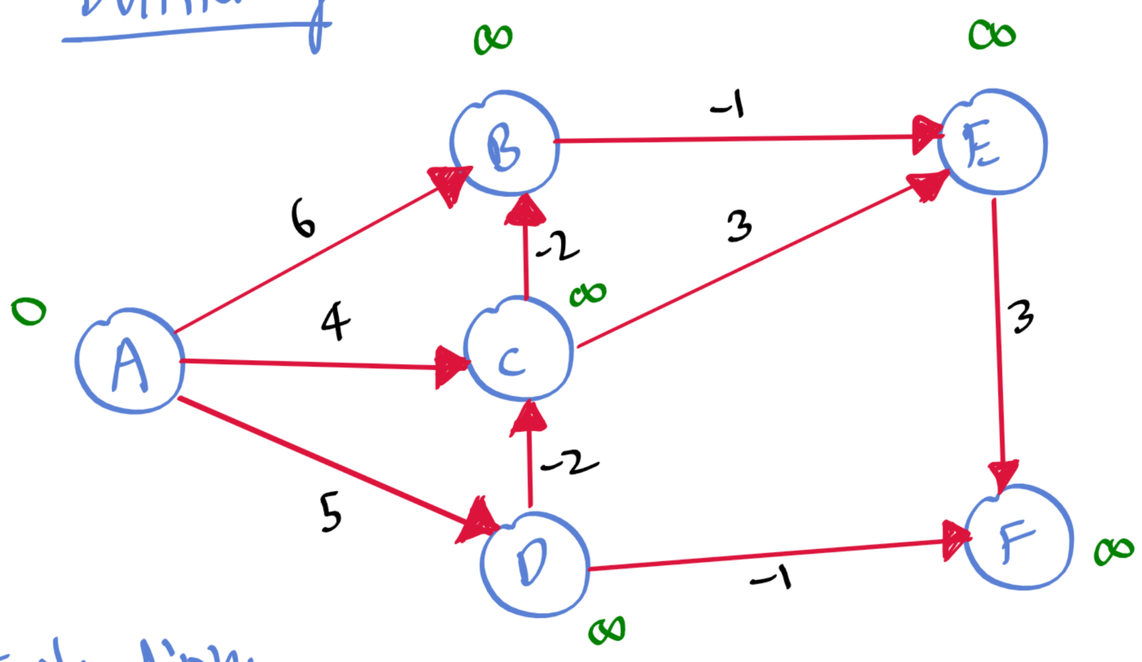
* Given a graph with no -ve cycles
* Initially soure distance =0 and distance of every other node is ∞
* use the formula $M(i,v) = \min\{M(i-1,v), (\min[i-1,w] + c_{vw})\}$

  and update

M → Array (2D) i → Edge v → Vertex c_{vw} → cost

* No. of iterations is |V|-1, consider each edge at every iteration.

Eg: Initially:

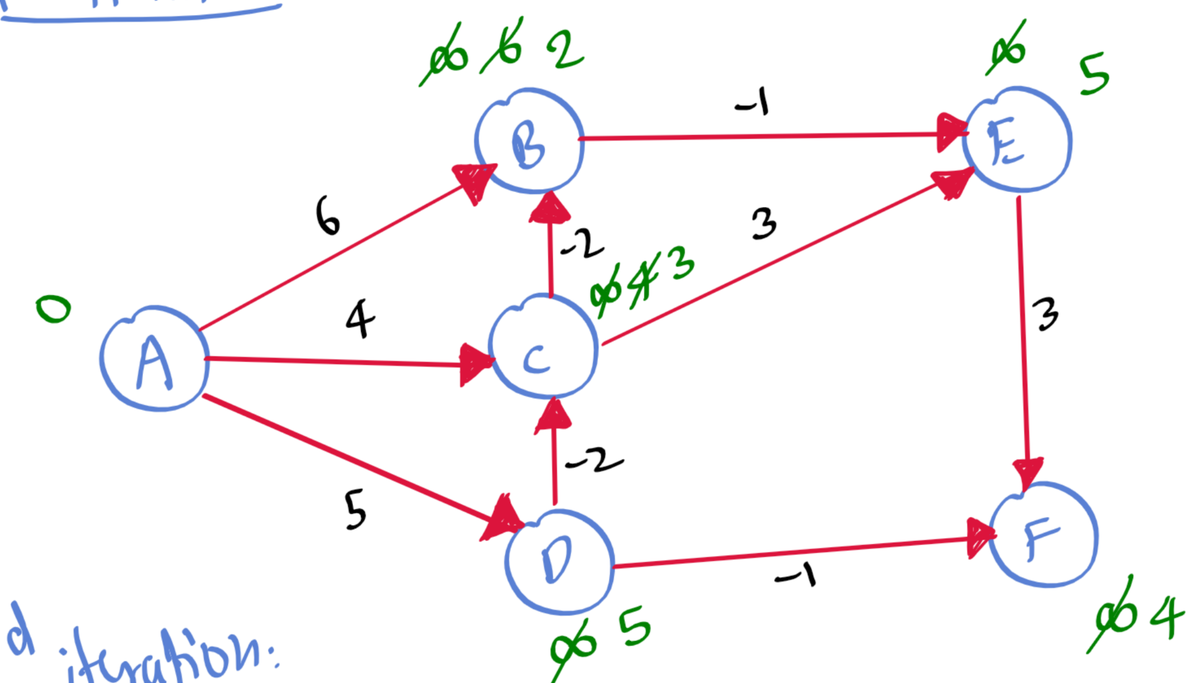


	Initial	1st	2nd	3rd	4th	5th
A	0	0	0			
B	∞	2	1			
C	∞	3	3			
D	∞	5	5			
E	∞	5	1			
F	∞	4	4			

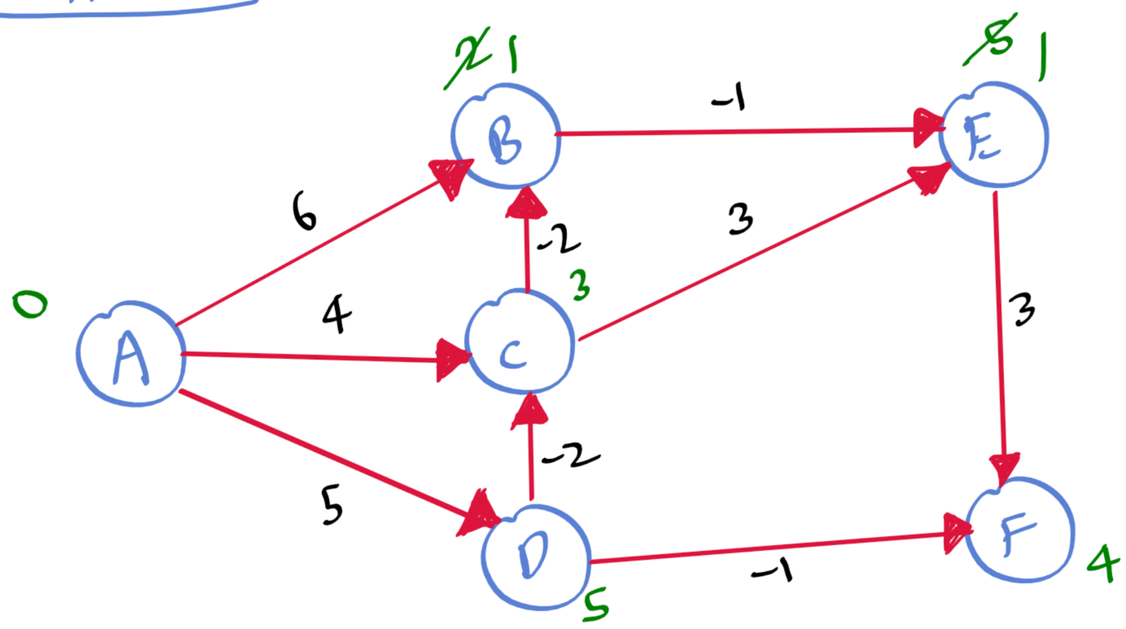
" Order of edges I have chosen,

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

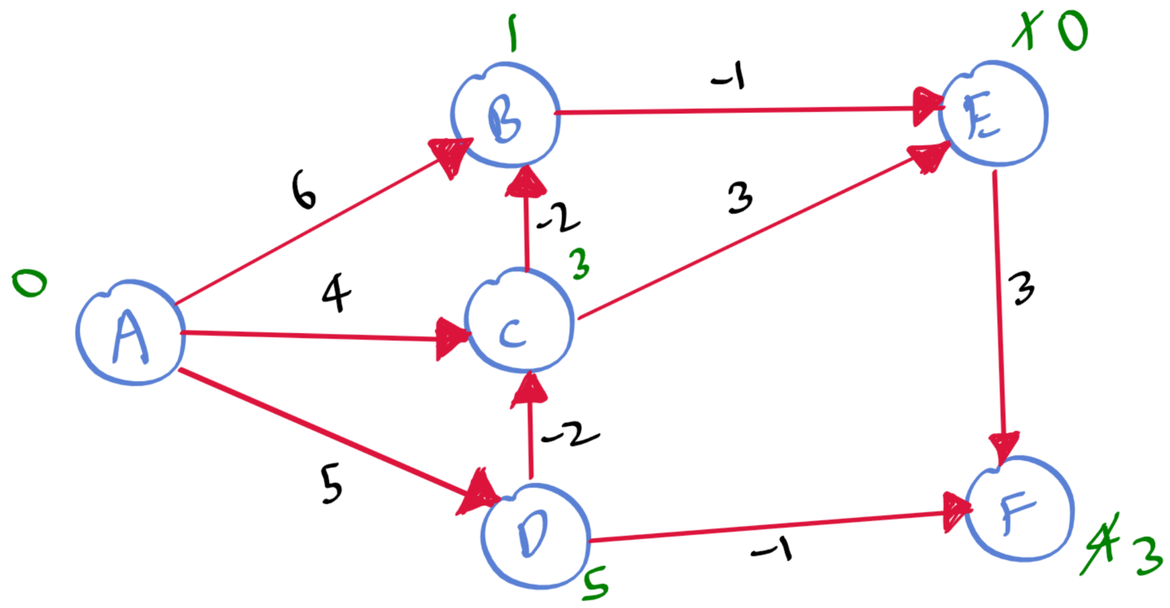
1st iteration:



2nd iteration:

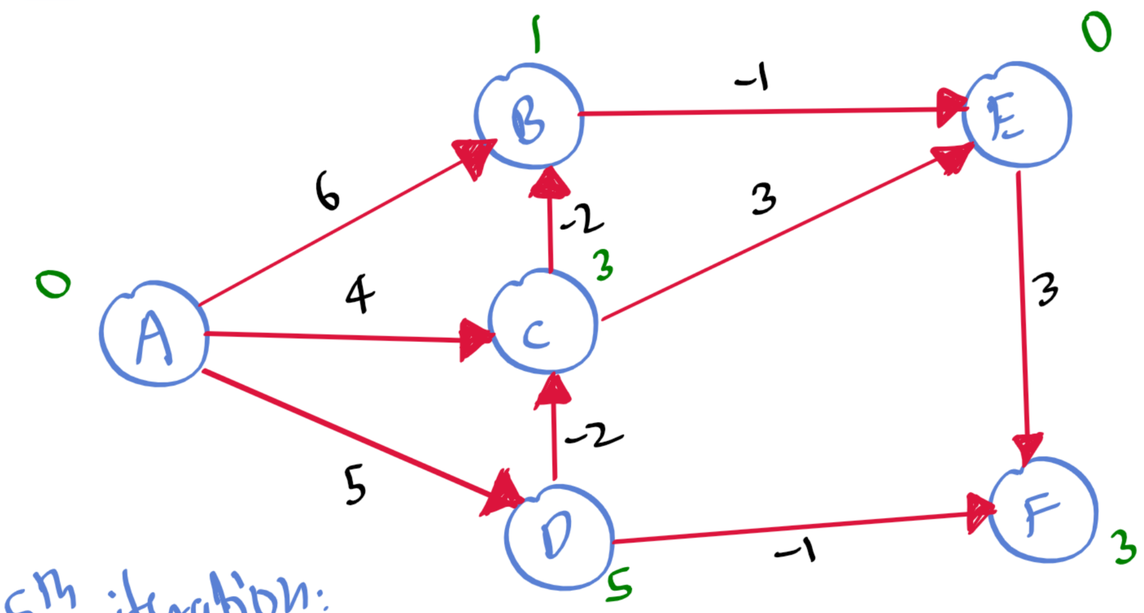


3rd iteration:



	Initial	1st	2nd	3rd	4th	5th
A	0	0	0	0	0	0
B	∞	2	1	1	1	1
C	∞	3	3	3	3	3
D	∞	5	5	5	5	5
E	∞	5	1	0	0	0
F	∞	4	4	3	3	3

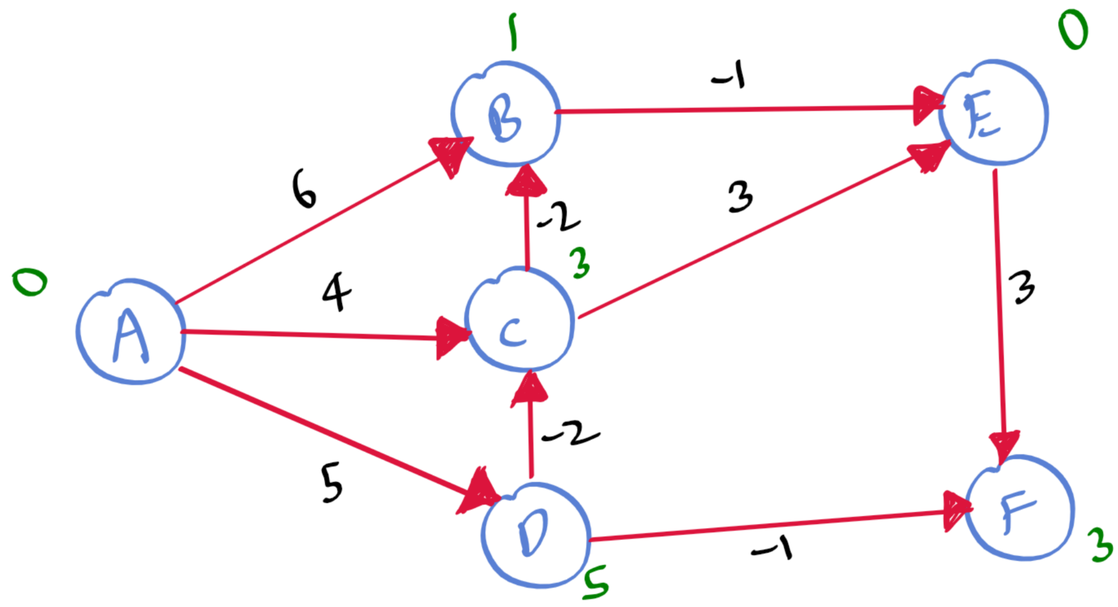
4th iteration:



" Order of edges I have chosen,

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

5th iteration:



Solution:

The shortest path from source A to other nodes is as follows:

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

④ Algorithm:

```
Shortest-Path(G, s, t)
  n = number of nodes in G
  Array M[0 ... n − 1, V]
  Define M[0, t] = 0 and M[0, v] = ∞ for all other v ∈ V
  For i = 1, ..., n − 1                          → O(V-1)
    For v ∈ V in any order                       → O(E)
      Compute M[i, v] using the recurrence (6.23)
    Endfor
  Endfor
  Return M[n − 1, s]
```

(6.23) *If $i > 0$ then*

$$\text{OPT}(i, v) = \min(\text{OPT}(i-1, v), \min_{w \in V}(\text{OPT}(i-1, w) + c_{vw})).$$

→ O(1)

Total time complexity: $O((V-1)E)$ or $O(VE)$