Bellmon Ford Alg:-
(Shortest Path finding)

① Only applicable for directed graph.
(If given undirected graph then change it by side edges).

② When dijistra's algo fails (-ve weight edges and -ve weighted cycle), it will work fine

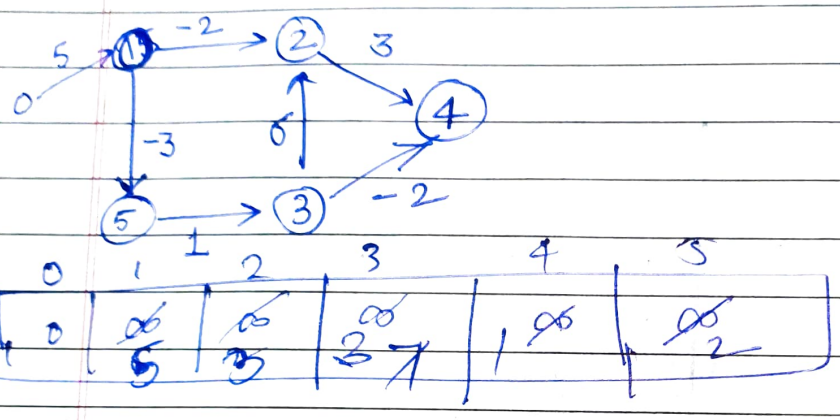
③ It helps to detect -ve cycle.

Note ⇒ order of edge given in Question doesn't matter.

Date / /
Page No.

Algo ⇒ ① Apply Relaxation on each edge
② now again apply Relaxation on each edge.
③ do this N − 1 time where N = |V| (no. of vertices)
④ Relax is same as in dijistra algo.

Ex ⇒



(u, v, wt)
(i) (3, 2, 6)
ii) (5, 3, 1)
iii) (0, 1, 5)
iv) (1, 5, -3)
v) (1, 2, -2)
vi) (3, 4, -2)
vii) (2, 4, 3)

1st Iteration.

(i) dist[3] + 6 < dist[2] × (no relaxation bcz condi fails)
(ii) d[5] + 1 < d[3] ×
(iii) d[0] + 5 < d[1] (5 < ∞) ✓
(iv) d[1] + (-3) < d[5] (2 < ∞) ✓
(v) d[1] + (-2) < d[2] (3 < ∞) ✓
(vi) d[3] + (-2) < d[4] ×
(vii) d[2] + 4 < d[3] (7 < ∞) ✓

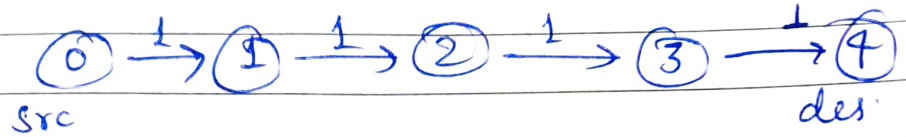
IInd iter :-

(i) (7 + 6) < 3 ×
(ii) (2 + 1) < 7 ✓
(iii) 0 + 5 < 5 ×
(iv) 5 − 3 < 2 ×
(v) 5 − 2 < 3 ×
(vi) (3 − 2) < (∞) ✓
(vii) (3 + 3) < 1 ×

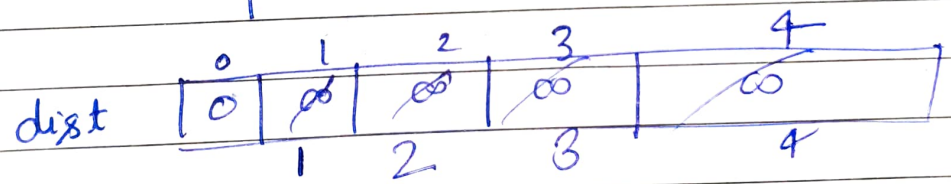
Similarly applying 3 more iteration and total 5 iterations will be completed.

Q. Why only (n-1) iteration required ?

Ans:- Intution :-



(u, v, wt)	1st iter	2nd	3rd	4th
(3,4,1)	$\infty+1<\infty$ (x)	x	x	$3+1<\infty$ (v)
(2,3,1)	$\infty+1<\infty$ (x)	x	$2+1<\infty$ (v)	x
(1,2,1)	$\infty+1<\infty$ (x)	$1+1<\infty$ (v)	x	x
(0,1,1)	$0+1<\infty$ (v)	x	x	x



We can clearly observe that at max upation of starting node can reach to last node in (n-1) no. of time.

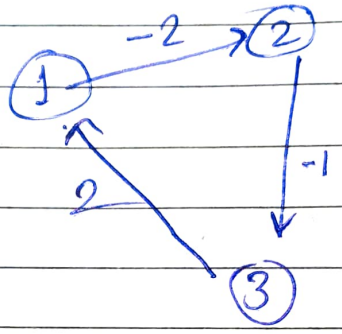
In other words ; → Since in a graph of N nodes, in worst case, you will take n-1 edges to reach from 1st node to last. Hence (n-1) at max iteration required.

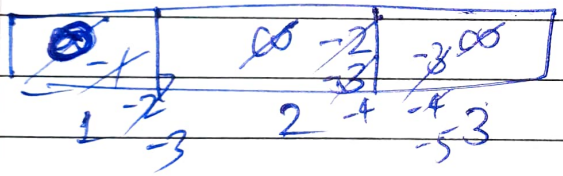
~~Most logical Answer~~

Q ⇒ How to detect cycle in graph ?

Ans ⇒ Apply Bellman Ford Algo; {(N-1) time iteration} Now apply 1 more iteration. If at this iteration some updation in distance array are observed then we can sure & say this will be a sign of -ve weight cycle.

Ex ⇒





{u, v, w}	1st Iter.	2nd Iter.	3rd Iter
{1, 2, -2}	(0-2) < ∞ (✓)	-1+(-2) < (-2) (✓)	-2-2 < -3 (✓)
{2, 3, -1}	(-2-1) < ∞ (✓)	-3+(-1) < -3 (✓)	-4-1 < -4 (✓)
{3, 1, 2}	(-3+2) < ∞ (✓)	-4+2 < -1 (✓)	-5+2 < -2 (✓)

Bellman Ford Algo

Cycle Detection at 3rd iteration (Nth Iteration)