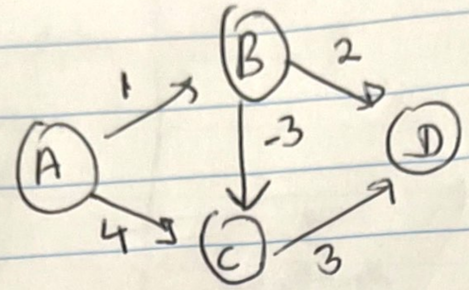
Bellman Ford Algorithms works with acyclic graph

I/P -



Output Distance = {0, 1, -2, 1}
A B C D

Idea - to find shortest path of edge length 1 first, shortest path of edge length 2 second and so on

Note: Edge length is defined as no. of nodes pointing towards it

Algorithm: we relax all edges V-1 time

Why?: $V_1 \to V_2$ --- $V_{n-1} \to V_n$

$V_{n-1} - V_n$

$V_{n-2} - V_{n-1}$

$V_{n-3} - V_{n-2}$

after V-1 iterations V_n will be updated

If d[V] > d[U] + weight(U, V)
d[V] = d[U] + weight[U,V)

Algorithm -

d[V] = {∞, ∞, ∞, ..., ∞]

d[src] = 0

for (count = 0 ; count < V-1 ; count++)
 for every edge (U, V)
  if (d[V] > d[U] + w(U, V]
   d[V] = d[U] + w(U, V)

Time Complexity → O(V·E)

Order of edges

B - C
B - D
C - D
A - B
A - C

Order doesn't matter

1st shortest path 2 edges
3rd

A	B	C	D
0	∞	∞	∞
0	1	∞	∞
0	1	-2	∞
0	1	-2	1

Note → If we do one more iteration, we can find negative cycle

```
for every edge (u,v)
    if (d[u] > d[v] > weight(u,v)
        print ("Negative cycle)
```