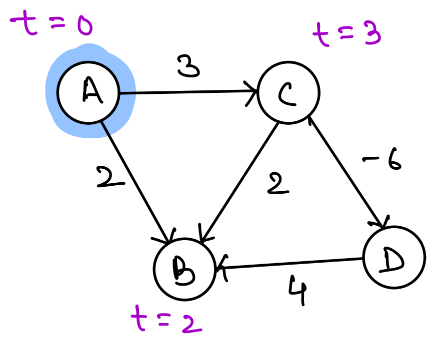
Dijkstra's Algorithm with -ve weights:-



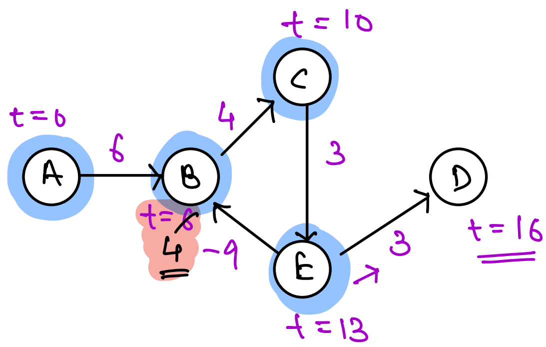
Src : A

Dest : B

$\{A \rightarrow B = 2\}$

Correct ans:-

$A \rightarrow C \rightarrow D \rightarrow B \Rightarrow 1$



$A \rightarrow D$

$A \rightarrow B \rightarrow C \rightarrow E \rightarrow D \Rightarrow 16$

$A \rightarrow \underbrace{B \rightarrow C \rightarrow E \rightarrow B}_{-2} \rightarrow C \rightarrow E \rightarrow D \Rightarrow 14$

-ve cycle

⇒ Keep going in the cycle & weight keeps on decreasing due to -ve cycle weight.

⇒ If -ve cycle is present, the shortest path is NOT defined.

Idea of Dijkstra's :-

1) Blast the node with min value.

2) Update all adjacent nodes.

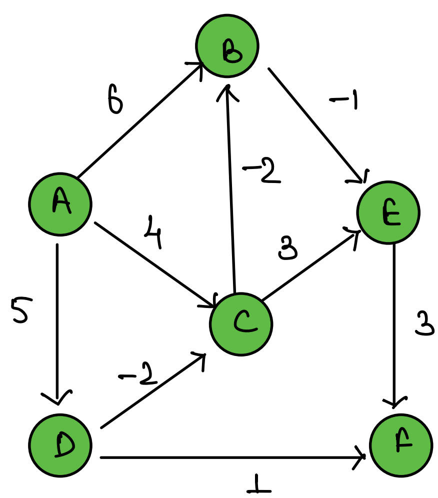
New Idea :- BELLMAN FORD ALGO

- +ve wts ✓
- −ve wts ✓
- −ve cycle. ✗

→ Iterate on every edge & update Nodes.

→ Repeat this for N−1 times.

N → No. of nodes.



dist :	A	B	C	D	E	F
	∞	∞	∞	∞	∞	∞
Iter1 :	0	~~6~~ 2	~~4~~ 3	5	5	6
Iter2 :	0	~~2~~ 1	3	5	1	~~6~~ 4
Iter3	0	1	3	5	~~1~~ 0	~~4~~ 3
Iter4	0	1	3	5	0	3

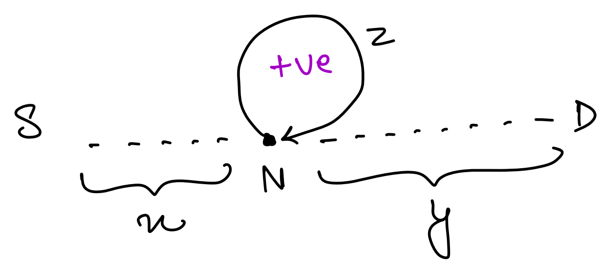
Iter5 : There was NO change from iter 3 to iter 4, we can break.

$A \xrightarrow{6} B$, $A \xrightarrow{4} C$, $A \xrightarrow{5} D$, $B \xrightarrow{-1} E$, $C \xrightarrow{3} E$,

$C \xrightarrow{-2} B$, $D \xrightarrow{-2} C$, $D \xrightarrow{1} F$, $E \xrightarrow{3} F$

* $\underline{N}$ nodes : At max the length (No. of edges) of any path can be $N-1$

$$S \underbrace{\text{— — — — —}}_{(N-1)\text{ edges}} D \Rightarrow \text{dist} = x+y$$



$\Rightarrow$ dist = $x+y+z$

dist = $\min(x+y,\ x+y+z)$

$= \underline{x+y}$

$$u \xrightarrow{w} v$$

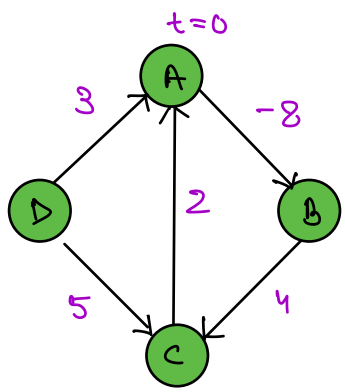
```
                                    u          v     w
Bellmanford ( list < pair < int, pair < int, int > > > Edges,
                                     int N, int src ) {
    int dist [N+1] = INT_MAX ;
    dist [src] = 0;
    int e = Edges.size();
    for ( K = 1 ; K < N; K++ ) {
          bool flag = false;
          for( i = 0; i < edges.size(); i++ ) {
               pair< int, pair< int, int > d = edges[i];
                u = d.first;
                v = d.second.first;
                w = d.second.second;
                if(dist[u] + w < dist[v] ) {
                       dist[v] = dist[u] + w ;
                       flag = true ;
                }
          }
          if ( flag == false ) {
                 break;
          }
    }
}
```

TC : O(NE)

SC : O(N)

-ve cycle detection using Bellman ford Algo :-

Given a directed graph, check if there's a -ve cycle in the graph.



dist :	A	B	C	D
	~~∞~~	~~∞~~	∞	∞
i)	~~0~~ -2	-8	-4	∞
ii)	~~-2~~ -4	~~-8~~ -10	~~-4~~ -6	∞
iii)	-6	-12	-8	∞
iv)	-8	-14	-10	∞

In 4^{th} iteration, dist array is still changing, -ve cycle is present in Graph.

$A \xrightarrow{-8} B$, $B \xrightarrow{4} C$, $C \xrightarrow{2} A$, $D \xrightarrow{3} A$, $D \xrightarrow{5} C$

Note :- If dist array is changing at N^{th} iteration then -ve cycle is present in the graph.

TC : O(NE)

SC : O(N)

——— * ———

⇒ Strongly Connected Components. **

⇒ Articulation points. [Optional]