```
else {
    cout<< "NO\n";  }
}
```

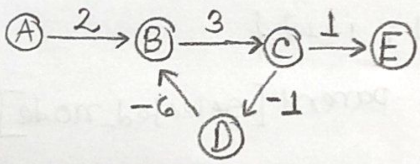
* Bellmann Ford Algorithm
→ Dynamic programming
* Basic principle is Relaxation
* more efficient is Dijkstra algorithm on time complexity.

—x—

—x— Practice Problem - 17.5 —x—

1. "It is impossible to get the shortest distance to a node in a graph with negative cycle.

→ Yes, it is impossible to get the shortest distance.



A → B　　C → E
B → C
C → D
D → B

Iteration no-01:

A → 0, B → 2, C → 5, D → 4 B → -2, E → 6

Iteration no-02:

A → 0, B → -2, C → -1, D → -2, B → -8, E → 0

Iteration no-03:

A → 0, B → -8 C → -5, D → -6, B → -12, E → -4

Iteration no-04:

A → 0, B → -12, C → -9 D → -10, B → -16, E → -8

Iteration no-05:

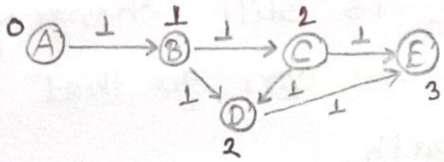
A → 0, B → -16, C → -13, D → -14, B → -20, E → -12

After maximum iteration, the value is changing and it will change forever and cannot get the shortest from it.

—x—

② Will Bellman-Ford work on unweighted graphs?

→ Yes, Bellman-Ford will work on unweighted graph. When it is unweighted it assumes that the weight is ① as edge count then we can see:



From A to E

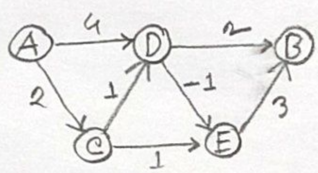
A → B → D → E

A → B → C → E

This time the algorithm counts the edges.

—x—

③



A → D   A → C   D → E   E → B
D → B   C → D   C → E

Iteration - 01:

A → 0   D → 4   B → 6   C → 2   D → 3   E → 2   B → 5

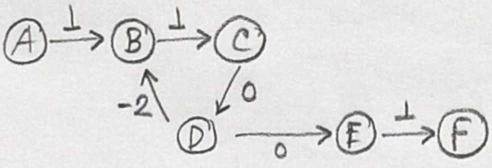
Iteration - 02:

A → 0   D → 3   B → 5   C → 2,   E → 2

So, in 2 iteration the values become fixed. 2 < 4. so it has no negative cycle.

—x—

④



A → B   D → B   E → F
B → C   C → D   D → E

Iteration - 01:

A → 0, B → 1, C → 2, B → 1, D → 2, F → ∞, E → 2

Iteration - 02:

A → 0, B → 1, C → 2, B → ~~1~~ 0, D → 2, F → 3, E → 2

Iteration - 03:

A → 0, B → 0, C → 1, D → 1, F → 3, E → 1

Iteration - 04:

A → 0, B → 0, C → 1, B → -1, D → 1, F → 2, E → 1

Iteration - 05:

A → 0, B → -1, C → 0, D → 0, F → 2, E → 0

Iteration - 06:

A → 0, B → -1, C → 0, B → -2, D → 0, F → 1, E → 0

We can see that the value is still changing at the maximum iteration, so, we can see that it has negative cycle → B C D B path.

— x —