1. **Dijkstra's Limitations :**
   SSSP in a weighted (Positive) graph problem solve.
   For negative weighted value, we cann't use Dijkstra.

   Negative Cycle value tense to become : – infinity.

   To detect negative Cycle, we cann't use Dijkstra.

2. **Bellman-Ford Algo's Intuition & Simulation :**

   By doing normal brute-force iteration, we can see, its guaranteed that after V-1 {Number of Nodes - 1}, times iteration, all node relaxation of a graph will complete, after that, there will be no change.
   So, (v-1) iteration needed to get the final distances of a graph of v nodes.

3. **Bellman-Ford Pseudocode & Complexity :**

   Input : A weighted Graph with no negative Cycle
   Output : Shorted Distance from source node to all other nodes.

   - Create a distance array "d" with all value to infinity.
   - d[source] = 0
   - for i=1 to v -1 : **O(n)**
     - For all edge "e(u, v, w)": **O(e)**
       - if d[u] + w < d[v] **O(1)**
         d [v] = d[u] + w // Relaxation **O(1)**

   - print the distance array "d". **O(n)**

   Time Complexity : O (|V| * |E|) :: O (|V| * n^2) :: O(n^3)
   Space Complexity : O (n)

4. **Bellman-Ford Code : Implementation**

5. **Negative Cycle Detection :**
   No change in d[Nodes] in N-th iteration Time. if any change detected that means, there is a cycle.

   So if, in N'th iteration any relaxation happens, it will have a negative cycle.

6. **Negative Cycle Detection Pseudocode & Code** :

   - Create a distance array "d" with all value to infinity.
   - d[source] = 0

   relaxed = false.

   - for i=1 to **v** :
       - For all edge "e(u, v, w)":
           - if d[u] + w < d[v]
               d [v] = d[u] + w
               if i == n :
                   relaxed = true.

   If relaxed == true:
       Negative Cycle Detected.