1. **Path Printing :**
   No Cycle :
   Store parent of v / parent of relaxed node = u/by which its relaxed.
   Then store parents from destination to source in a vector.
   The vector's reverse version will be the path for the graph.

2. **Cycle Printing:** Print the cycle.
   With Cycle :
   Detect the Negative cycle. And print the Negative cycle.
   In n'th iteration, we can see changes of nodes whose are the part of the negative cycle and those nodes that can be reached from the negative cycle.
   If we start tracing from a node that can be reached from the cycle. Then, we will be on the cycle in a certain period of time.
   To enter in the cycle from outer part but reachable node from the cycle, we need to trace parent n-1 iteration times.

3. **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.
   - Create a parent array
   - d[source] = 0
   - negative_cycle = false.
   - for i=1 to v -1 :
       - For all edge "e(u, v, w)":
           - if d[u] + w < d[v]:
               - d [v] = d[u] + w
               - parent[v] = u
               if i == n:
                   - negative_cycle = true.

   - IF negative_cycle == false : O(N)
       - selected_node = destination_node
       - declare a vector as named "path"
       - while True :

Path.push_back(selected_node)
- selected_node = parent[selected_node]
- If selected_node == src :
break

- reverse the vector and print it.

- IF negative_cycle == True : O(N) + O(N)
- create a "last_updated_node" variable. = -1
- if i == n:
- negative_cycle = true.
- last_updated_node = v [latest updated node]

- selected_node = last_updated_node
- for(i=1 to n-1):
- selected_node = parent[selected_node]
- int first_updatedNode = selected_node
- declare a vector as named "cycle"
- cycle.push_back(selected_node)
- while True:
- selected_node = parent[selected_node]
- cycle.push_back(selected_node)
- if (selected_node == first_updatedNode) :
- break.
- reverse the cycle vector and print it.

Time Complexity : O (V * E) [SAME]
Space Complexity : O (V) [SAME]

4. **Problem Solving : Implementation in Code :**
Cycle Finding : *https://cses.fi/problemset/task/1197*