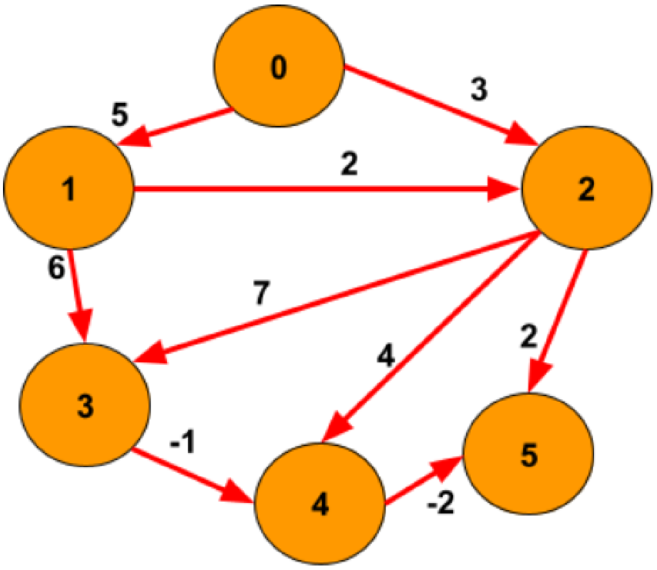
Shortest Path in a Directed Acyclic Graph

List of shortest distances denoting the shortest path from ‘Src’ to all other nodes in the DAG.

So the problem is to find the shortest path from a given source to All other nodes in the weighted DAG.

Example:



Source: 1, Output: {2147483647, 0, 2, 6, 5, 3}

addEdge Function:

- **Purpose:**
    - Adds edges to the graph, considering the weights if provided.
- **Explanation:**
    - Iterates through each edge in the **edges** vector.
    - Extracts the source vertex **u**, destination vertex **v**, and weight **w**.
    - If the weight is not provided, defaults to 1.
    - Adds an edge from **u** to **v** with weight **w** in the adjacency list.
- **Time Complexity:**
    - **O(E), where E is the number of edges in the input vector.**
- **Space Complexity:**
    - **O(E), where E is the number of edges. Each edge results in the creation of an entry in the adjacency list.**

Approach 1: Function to find the shortest path using topological sorting and relaxation

- **Purpose:**
  - Finds the shortest path from a given source node to all other vertices using topological sorting.
- **Explanation:**
  - Performs DFS to obtain the topological ordering of nodes.
  - Initializes distances with infinity and sets the source distance to 0.
  - Processes nodes in topological order, updating distances through relaxation.
  - Utilizes a stack for topological sorting and a vector to store distances.
- **Time Complexity:**
  - **O(V + E), where V is the number of vertices and E is the number of edges.**
  - **Combined with addEdge Function:**
    - **Total Time Complexity: O(V + E)**
- **Space Complexity:**
  - **O(V + E), where V is the number of vertices and E is the number of edges.**
  - **Combined with addEdge Function:**
    - **Total Space Complexity: O(V + E)**