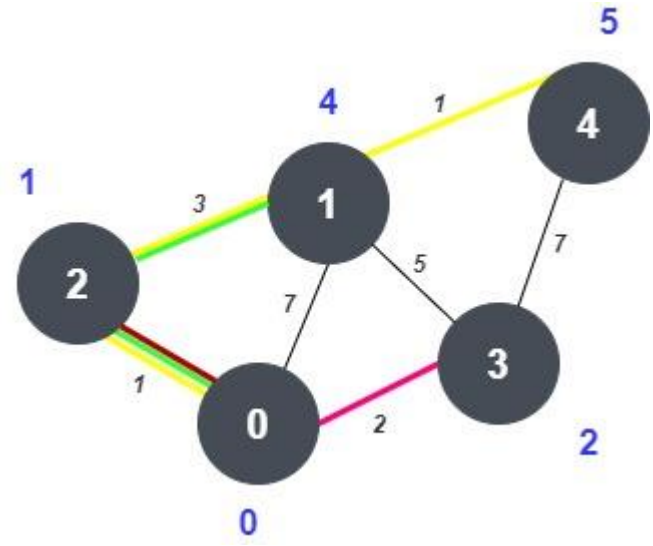
Find Shortest Path Weighted Graph using Dijakstra's Algorithm CodeStudio

You have been given an undirected graph of 'V' vertices (labeled 0,1,..., V-1) and 'E' edges. Each edge connecting two nodes ('X','Y') will have a weight denoting the distance between node 'X' and node 'Y'.

Your task is to find the shortest path distance from the source node, which is the node labeled as 0, to all vertices given in the graph.

Example:



Source: 0, Output: {0, 4, 1, 2, 5}

addEdge Function:

- **Purpose:**
  - Adds edges to the undirected 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 edges from both **u** to **v** and **v** to **u** 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 entries in the adjacency list for both vertices.**

Approach 1: Function to find the shortest path from the source to all vertices using Dijkstra's algorithm

- **Purpose:**
  - Finds the shortest path from a given source node to all other vertices in an undirected weighted graph.
- **Explanation:**
  - Initializes distances with infinity and sets the source distance to 0.
  - Maintains a priority queue (set) of vertices with their current distance from the source.
  - Repeatedly extracts the vertex with the smallest distance and relaxes its neighbors.
- **Time Complexity:**
  - **O((V + E) * log(V)), where V is the number of vertices and E is the number of edges.**
  - **Combined with addEdge Function:**
    - **Total Time Complexity: O((V + E) * log(V))**
- **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)**