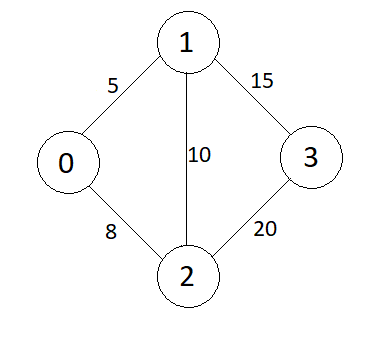
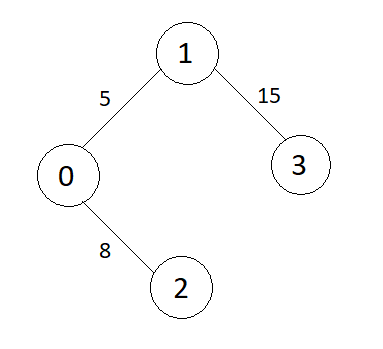
Minimum Spanning Tree in Undirected Graph using Prim’s Algorithm [CodeStudio](https://www.codingninjas.com/studio/problems/prim-s-mst_1095633)

You are given an undirected connected weighted graph having ‘N’ nodes numbered from 1 to 'N'. A matrix ‘E’ of size M x 2 is given which represents the ‘M’ edges such that there is an edge directed from node E[i][0] to node E[i][1]. You are supposed to return the minimum spanning tree where you need to return weight for each edge in the MST.



Output:



**addEdge Function:**

* **Purpose:**
  + Adds edges to the undirected graph, considering the weights.
* **Explanation:**
  + Iterates through each edge in the **edges** vector.
  + Extracts the source vertex **u**, destination vertex **v**, and weight **w**.
  + 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 Minimum Spanning Tree using Prim's Algorithm**

* **Purpose:**
  + Finds the MST using Prim's algorithm.
* **Explanation:**
  + Initializes vectors to store key values, MST set, and parent vertices.
  + Starts from the first vertex.
  + Iteratively selects the vertex with the minimum key value not yet in the MST.
  + Includes the selected vertex in the MST and updates key values and parent vertices for neighboring vertices.
  + Constructs the result in the form of edges and their weights.
* **Time Complexity:**
  + **O(V^2), where V is the number of vertices.**
* **Space Complexity:**
  + **O(V + E), where V is the number of vertices and E is the number of edges.**

**Approach 2: Function to find the Minimum Spanning Tree using Optimized Prim's Algorithm with a Priority Queue**

* **Purpose:**
  + Finds the MST using an optimized version of Prim's algorithm with a priority queue.
* **Explanation:**
  + Initializes vectors to store key values, MST set, and parent vertices.
  + Uses a priority queue to efficiently find the minimum key value.
  + Iteratively selects the vertex with the minimum key value not yet in the MST.
  + Includes the selected vertex in the MST and updates key values and parent vertices for neighboring vertices.
  + Constructs the result in the form of edges and their weights.
* **Time Complexity:**
  + **O((V + E) \* log(V)), where V is the number of vertices and E is the number of edges.**
* **Space Complexity:**
  + **O(V + E), where V is the number of vertices and E is the number of edges.**

**Conclusion:**

* Prim's algorithm efficiently finds the Minimum Spanning Tree in an undirected weighted graph.
* The optimized version using a priority queue further improves the algorithm's efficiency.