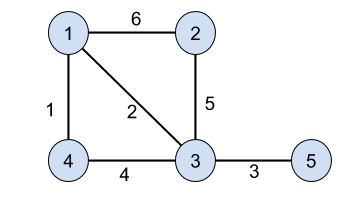
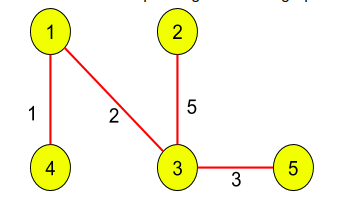
Minimum Spanning Tree in Undirected Weighted Graphs using Prim’s and Krukshal’s Algorithms [CodeStudio](https://www.codingninjas.com/studio/problems/kruskal-s-minimum-spanning-tree-algorithm_1082553?leftPanelTabValue=PROBLEM)

A minimum spanning tree is a subset of the edges of a connected, edge-weighted undirected graph that connects all the vertices without any cycles and with the minimum possible total edge weight.



Output: **11**



**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.**

**Approach 1: Prim's Algorithm to find Minimum Spanning Tree Weight**

* **Purpose:**
  + Finds the MST weight using Prim's algorithm.
* **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.
  + Calculates and returns the total weight of the MST.
* **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.**

**Approach 2: Kruskal's algorithm to find the minimum spanning tree weight**

* **Purpose:**
  + Finds the MST weight using Kruskal's algorithm.
* **Explanation:**
  + Sorts edges based on their weights.
  + Initializes parent array and rank array for union-find operations.
  + Iterates through sorted edges and calculates the weight of the MST if it doesn't form a cycle.
  + Returns the total weight of the MST.
* **Time Complexity:**
  + **O(E \* log(E)), where E is the number of edges.**
* **Space Complexity:**
  + **O(V), where V is the number of vertices. The parent array is used for union-find operations.**

**Conclusion:**

* The **addEdge** function has a time complexity of O(E) and space complexity of O(E), directly proportional to the number of edges.
* Prim's algorithm is efficient for dense graphs, while **Kruskal's algorithm may be preferable for sparse graphs or scenarios where edge sorting is inexpensive**.