```
import sys
```

```
# Function to find the vertex with minimum key value
```

```
def min_key(key, mst_set):
    min_val = sys.maxsize
    min_index = -1

for v in range(len(key)):
    if key[v] < min_val and not mst_set[v]:
        min_val = key[v]
        min_index = v

return min_index</pre>
```

Prim's MST function

```
def prim_mst(graph):
    num_vertices = len(graph)
    key = [sys.maxsize] * num_vertices
    parent = [None] * num_vertices
    mst_set = [False] * num_vertices

    key[0] = 0
    parent[0] = -1 # Start from first vertex

for _ in range(num_vertices):
    u = min_key(key, mst_set)
    mst_set[u] = True

for v in range(num_vertices):
```

```
if graph[u][v] != 0 and not mst_set[v] and graph[u][v] < key[v]:
    key[v] = graph[u][v]
    parent[v] = u

# Print the MST

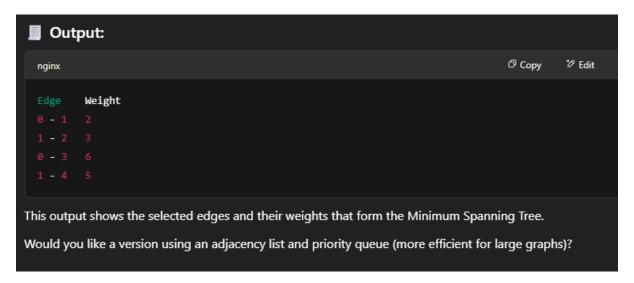
print("Edge \tWeight")

for i in range(1, num_vertices):
    print(f"{parent[i]} - {i} \t{graph[i][parent[i]]}")</pre>
```

Example: Undirected weighted graph (adjacency matrix)

```
graph = [
[0, 2, 0, 6, 0],
[2, 0, 3, 8, 5],
[0, 3, 0, 0, 7],
[6, 8, 0, 0, 9],
[0, 5, 7, 9, 0]
]
```

prim_mst(graph)



Viva Questions for This Practical

1. What is Prim's Algorithm?

Prim's Algorithm is a greedy algorithm used to find the Minimum Spanning
 Tree (MST) of a connected, weighted graph.

2. What are the key data structures used in Prim's Algorithm?

- The key data structures are:
 - key[]: Stores the minimum weight edge connecting each vertex to the MST.
 - mst_set[]: Tracks which vertices are included in the MST.
 - parent[]: Stores the parent of each vertex in the MST.

3. What is the time complexity of Prim's Algorithm?

 The time complexity is O(V^2) with an adjacency matrix. With a priority queue (min-heap), the complexity can be reduced to O(E log V).

4. What does the min_key function do?

 The min_key function finds the vertex with the smallest key value that is not yet included in the MST.

5. Why does the algorithm use a greedy approach?

 It uses a greedy approach because at each step, it selects the vertex with the minimum edge weight to include in the MST, making the locally optimal choice to build the MST.

6. How does Prim's Algorithm differ from Kruskal's Algorithm?

 Prim's Algorithm grows the MST from a single vertex by adding edges one by one, while Kruskal's Algorithm considers all edges and sorts them by weight, adding them to the MST while avoiding cycles.

7. What is a Minimum Spanning Tree (MST)?

 An MST is a tree that connects all the vertices of a graph with the minimum possible total edge weight, ensuring there are no cycles.

8. What is the role of the parent array in the algorithm?

The parent[] array keeps track of the parent vertex of each vertex in the MST,
 which helps in printing the MST edges at the end.

9. What is the significance of initializing key values to infinity?

 Initializing key values to infinity ensures that the first selected vertex will always be the one with the minimum edge weight.

10. How do we handle disconnected graphs in Prim's Algorithm?

 Prim's Algorithm requires the graph to be connected. If the graph is disconnected, it does not produce a valid MST.