Python Code :-

| # A Python program for Prim's Minimum Spanning Tree (MST) algorithm. # The program is for adjacency matrix representation of the graph   import sys # Library for INT\_MAX   **class** **Graph**():    **def** **\_\_init\_\_**(**self**, vertices):  **self**.V = vertices  **self**.graph = [[0 **for** column **in** range(vertices)]  **for** row **in** range(vertices)]    # A utility function to print the constructed MST stored in parent[]  **def** **printMST**(**self**, parent):  print ("Edge \tWeight")  **for** i **in** range(1, **self**.V):  print (parent[i], "-", i, "\t", **self**.graph[i][parent[i]])    # A utility function to find the vertex with  # minimum distance value, from the set of vertices  # not yet included in shortest path tree  **def** **minKey**(**self**, key, mstSet):    # Initialize min value  min = sys.maxsize    **for** v **in** range(**self**.V):  **if** key[v] < min **and** mstSet[v] == False:  min = key[v]  min\_index = v    **return** min\_index    # Function to construct and print MST for a graph  # represented using adjacency matrix representation  **def** **primMST**(**self**):    # Key values used to pick minimum weight edge in cut  key = [sys.maxsize] \* **self**.V  parent = [None] \* **self**.V # Array to store constructed MST  # Make key 0 so that this vertex is picked as first vertex  key[0] = 0  mstSet = [False] \* **self**.V    parent[0] = -1 # First node is always the root of    **for** cout **in** range(**self**.V):    # Pick the minimum distance vertex from  # the set of vertices not yet processed.  # u is always equal to src in first iteration  u = **self**.minKey(key, mstSet)    # Put the minimum distance vertex in  # the shortest path tree  mstSet[u] = True    # Update dist value of the adjacent vertices  # of the picked vertex only if the current  # distance is greater than new distance and  # the vertex in not in the shortest path tree  **for** v **in** range(**self**.V):    # graph[u][v] is non zero only for adjacent vertices of m  # mstSet[v] is false for vertices not yet included in MST  # Update the key only if graph[u][v] is smaller than key[v]  **if** **self**.graph[u][v] > 0 **and** mstSet[v] == False **and** key[v] > **self**.graph[u][v]:  key[v] = **self**.graph[u][v]  parent[v] = u    **self**.printMST(parent)   g = Graph(5) g.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]]   g.primMST(); |
| --- |

Output:-

Edge Weight

0 - 1 2

1 - 2 3

0 - 3 6

1 - 4 5