Prims

import sys

class Graph():

def \_\_init\_\_(self, vertices):

self.V = vertices

self.graph = [[0 for column in range(vertices)]

for row in range(vertices)]

def printMST(self, parent):

print("Edge \tWeight")

total\_weight = 0

for i in range(1, self.V):

weight = self.graph[parent[i]][i]

print(parent[i] + 1, "-", i + 1, "\t", weight)

total\_weight += weight

print("Total weight of MST:", total\_weight)

def minKey(self, key, mstSet):

min = sys.maxsize

min\_index = -1

for v in range(self.V):

if key[v] < min and not mstSet[v]:

min = key[v]

min\_index = v

return min\_index

def primMST(self):

key = [sys.maxsize] \* self.V

parent = [None] \* self.V

key[0] = 0

mstSet = [False] \* self.V

parent[0] = -1

for \_ in range(self.V):

u = self.minKey(key, mstSet)

mstSet[u] = True

for v in range(self.V):

if self.graph[u][v] > 0 and not mstSet[v] and key[v] > self.graph[u][v]:

key[v] = self.graph[u][v]

parent[v] = u

self.printMST(parent)

if \_\_name\_\_ == '\_\_main\_\_':

g = Graph(6)

g.graph = [

[0, 2, 0, 1, 4, 0], # Node 1

[2, 0, 3, 0, 0, 7], # Node 2

[0, 3, 0, 5, 0, 8], # Node 3

[1, 0, 5, 0, 9, 0], # Node 4

[4, 0, 0, 9, 0, 0], # Node 5

[0, 7, 8, 0, 0, 0] # Node 6

]

g.primMST()

theory

Prim's Algorithm – Theory

Prim's Algorithm is a Greedy algorithm used to find the Minimum Spanning Tree (MST) of a connected, undirected, weighted graph. The MST connects all vertices with the minimum total edge weight, ensuring no cycles are formed.

🔹 Key Concepts:

Minimum Spanning Tree (MST): A subset of edges that connects all vertices in the graph with the minimum possible total edge weight and no cycles.

Prim’s Approach:

Start with any node (usually node 0).

At each step, add the minimum weight edge that connects a visited node to an unvisited node.

Repeat until all nodes are included in the tree.

Steps of Prim’s Algorithm:

Initialize:

A list key[] to store minimum weights (initially set to ∞).

A list parent[] to store the MST structure.

A boolean list mstSet[] to track included nodes.

Set key[0] = 0 (start from node 0) and parent[0] = -1.

Repeat V times (where V = number of vertices):

Pick the node with the minimum key value that is not yet included in MST.

Mark it as included.

Update the key values of its adjacent unvisited nodes if the connecting edge is smaller than the current key value.

After all nodes are added, the parent[] array holds the MST.