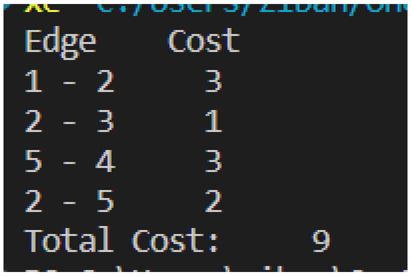
Justin Li
4/13/23

COEN 239 HW 2: Minimum Spanning Tree - Prim’s Algorithm

```
# Minimum Spanning Tree - Prim's Algorithm Program
# Source: https://www.geeksforgeeks.org/prims-minimum-spanning-tree-mst-greedy-algo-5/#

import sys

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 printG(self, parent):
        print("Edge \tCost")
        cost = 0
        for i in range(1, self.V):
            print(parent[i]+1, "-", i+1, "\t ", self.graph[i][parent[i]])
            cost += int(self.graph[i][parent[i]])
        print(f"Total Cost:\t{cost}")

    # 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 prim(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):
            u = self.minKey(key, mstSet)
            mstSet[u] = True

            for v in range(self.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.printG(parent)


# Driver code
g = Graph(5)
g.graph = [[1, 3, 4, 5, 6],
            [3, 1, 1, 4, 2],
            [4, 1, 1, 5, 7],
            [5, 4, 5, 1, 3],
            [6, 2, 7, 3, 1]]

g.prim()
```

Results:



```
Edge    Cost
1 - 2     3
2 - 3     1
5 - 4     3
2 - 5     2
Total Cost:     9
```