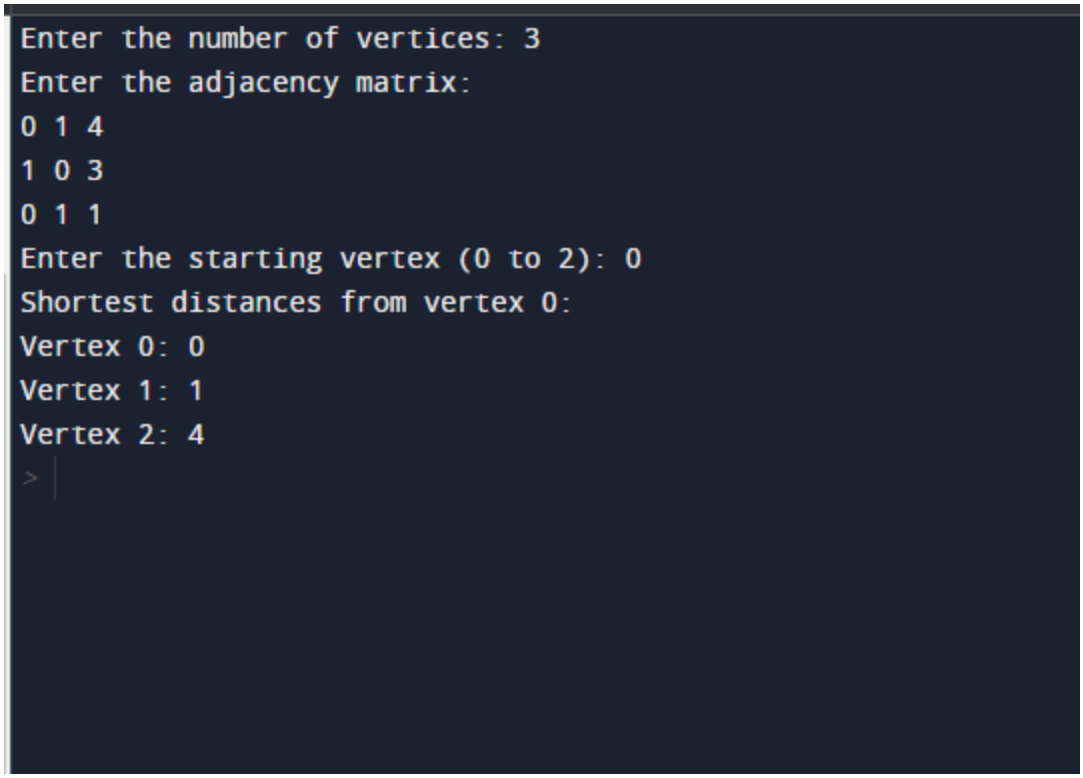
Assignment 5 : Dijikstra's and Prim's Algorithm

```
def dijkstra(graph, start):
    vertices = len(graph)
    visited = [False] * vertices
    dist = [float('inf')] * vertices
    dist[start] = 0

    for _ in range(vertices):
        min_dist = float('inf')
        for v in range(vertices):
            if not visited[v] and dist[v] < min_dist:
                min_dist = dist[v]
                u = v

        visited[u] = True

        for v in range(vertices):
            if not visited[v] and graph[u][v] > 0:
                if dist[u] + graph[u][v] < dist[v]:
                    dist[v] = dist[u] + graph[u][v]

    return dist

# Input for the graph
n = int(input("Enter the number of vertices: "))
graph = []

print("Enter the adjacency matrix:")
for _ in range(n):
    row = list(map(int, input().split()))
    graph.append(row)

start_vertex = int(input("Enter the starting vertex (0 to {}): ".format(n - 1)))

shortest_distances = dijkstra(graph, start_vertex)

print("Shortest distances from vertex {}:".format(start_vertex))
for i, distance in enumerate(shortest_distances):
    print("Vertex {}: {}".format(i, distance))
```



Prim’s Algorithm

```
def prim(graph):
    vertices = len(graph)
    parent = [-1] * vertices
    key = [float('inf')] * vertices
    key[0] = 0
    mst_set = [False] * vertices

    for _ in range(vertices):
        min_key = float('inf')
        for v in range(vertices):
            if not mst_set[v] and key[v] < min_key:
                min_key = key[v]
                u = v

        mst_set[u] = True
```

```
        for v in range(vertices):
            if graph[u][v] > 0 and not mst_set[v] and graph[u][v] < key[v]:
                parent[v] = u
                key[v] = graph[u][v]

    return parent

# Input for the graph
n = int(input("Enter the number of vertices: "))
graph = []

print("Enter the adjacency matrix:")
for _ in range(n):
    row = list(map(int, input().split()))
    graph.append(row)

minimum_spanning_tree = prim(graph)

print("Minimum Spanning Tree:")
for i in range(1, n):
    print("Edge: {} - {}".format(minimum_spanning_tree[i], i))
```

```
Enter the number of vertices:  4
4
Enter the adjacency matrix:
1 0 1 3
0 1 0 4
1 2 3 4
0 5 0 1
Minimum Spanning Tree:
Edge: 2 - 1
Edge: 0 - 2
Edge: 0 - 3
>
```