

## Assignment NO .: 3

**Name:** Vaidehi Santosh Pawar

**Roll NO:** TECOB - 15

**PracticalBatch .:** B1

**Title of the Assignment :** Implement Greedy search algorithm.

**Problem Statement :** Implement Dijkstra's Minimal Spanning Tree Algorithm

```
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 printSolution(self, dist):
```

```
        print("Vertex \t Distance from Source")
```

```
        for node in range(self.V):
```

```
            print(node, "\t", dist[node])
```

```
    def minDistance(self,dist,sptSet):
```

```
        min = sys.maxsize
```

```
        for u in range(self.V):
```

```
            if dist[u] < min and sptSet[u] == False:
```

```
                min = dist[u]
```

```
                min_index = u
```

```
    return min_index
```

```

def dijkstra(self,src):
    dist = [sys.maxsize] * self.V
    dist[src] = 0
    sptSet = [False] * self.V

    for cout in range(self.V):
        x = self.minDistance(dist,sptSet)
        sptSet[x] = True
        for y in range(self.V):
            if self.graph[x][y] > 0 and sptSet[y] == False and \
                dist[y] > dist[x] + self.graph[x][y]:
                dist [y] = dist[x] + self.graph[x][y]

    self.printSolution(dist)

```

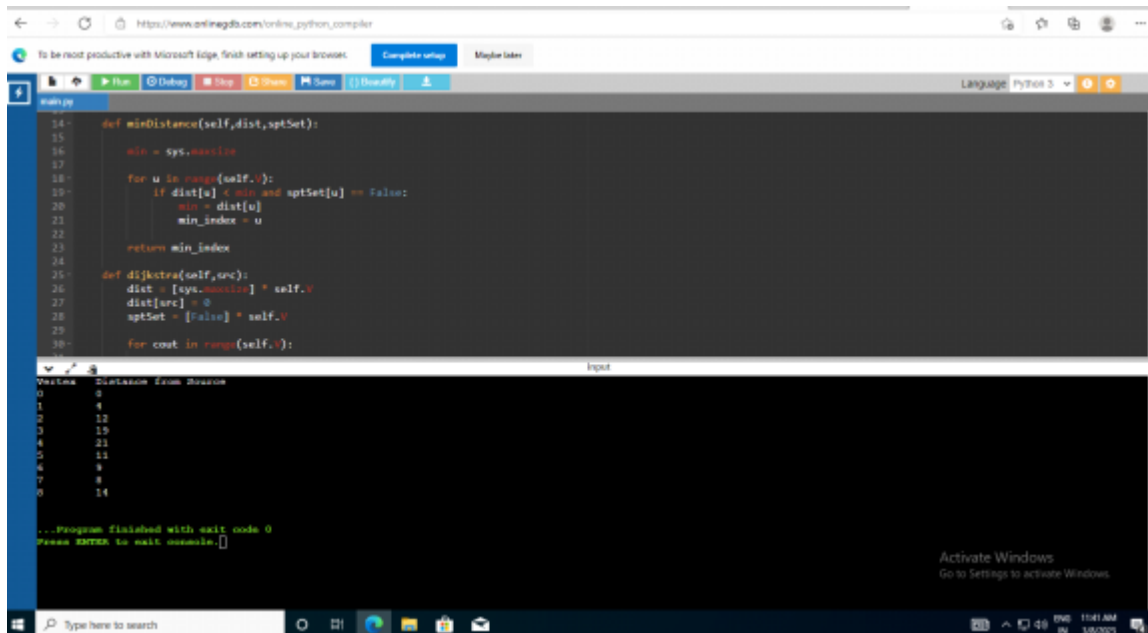
```

if __name__ == "__main__":
    g = Graph(9)
    g.graph = [[0, 4, 0, 0, 0, 0, 0, 8, 0],
                [4, 0, 8, 0, 0, 0, 0, 11, 0],
                [0, 8, 0, 7, 0, 4, 0, 0, 2],
                [0, 0, 7, 0, 9, 14, 0, 0, 0],
                [0, 0, 0, 9, 0, 10, 0, 0, 0],
                [0, 0, 4, 14, 10, 0, 2, 0, 0],
                [0, 0, 0, 0, 0, 2, 0, 1, 6],
                [8, 11, 0, 0, 0, 0, 1, 0, 7],
                [0, 0, 2, 0, 0, 0, 6, 7, 0]
    ]

```

g.dijkstra(0)

output:



The screenshot shows an online Python compiler interface. The code defines a class with two methods: `minDistance` and `dijkstra`. The `dijkstra` method calls `minDistance` for each vertex to find the shortest path from the source vertex (0). The output displays the shortest distance from the source vertex (0) to each of the 10 vertices.

```
def minDistance(self, dist, sptSet):
    min = sys.maxsize
    for u in range(self.V):
        if dist[u] < min and sptSet[u] == False:
            min = dist[u]
            min_index = u
    return min_index

def dijktra(self, src):
    dist = [sys.maxsize] * self.V
    dist[src] = 0
    sptSet = [False] * self.V
    for cout in range(self.V):
```

Vertex	Distance from Source
0	0
1	4
2	12
3	19
4	21
5	11
6	9
7	8
8	14

...Program finished with exit code 0  
Press ENTER to exit console.