

Dijkstra's Algorithm / or Parting

Krishna prasad V

18M11CS047

SA

class Graph():

def __init__(self, vertices):

self.V = vertices

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

def print_solution(self, dist):

print("Vertex & Distance from source")

for node in range(self.V):
 print(node, ":", dist[node])

def min_distance(self, dist, sptSet):

min = 9999

for v in range(self.V):

if dist[v] < min and sptSet[v] == False:

min = dist[v]

min_index = v

return min_index

def add_edge(self, src, dest, weight):

self.graph[src][dest] = self.graph[dest][src] = weight

def dijkstra(self, src):

dist = [9999] * self.V

dist[src] = 0

sptSet = [False] * self.V

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

```
    v = self.min_distance(dist, sptSet)
```

```
    sptSet[v] = True
```

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

```
    if self.graph[v][v] > 0 and sptSet[v] == False and  
        dist[v] > dist[v] + self.graph[v][v]:  
        dist[v] = dist[v] + self.graph[v][v]
```

```
self.print_solution(dist)
```

```
g = Graph(int(input("Enter number of nodes in the topology")))
```

```
c = int(input("Enter no of edges: "))
```

```
for i in range(c):
```

```
    src, dest, cost = [int(-) for - in input("Enter [SRC] [DST] [weight)"]  
                        .split(' ')]
```

```
g.add_edge(src, dest, cost)
```

```
src = int(input("Enter [SRC] to find cost:"))
```

```
g.dijkstra(src)
```