07/12/2020 CNLAB KARTHIK VENUGOPAL

WRITEUP 1BM18CS043

Dijikstra's algorithm

Dijikstra's algorithm is used to find the shortest path from a starting node to a target node in a weighted graph.

Program :

```
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 \t Distance from Source")
        for node in range(self.V):
            print(node, "\t", dist[node])

    def min_distance(self, dist, sptSet)
        min = 9999
        for ~~[illegible]~~ v in range(self.V):
            if dist[v] < min and sptSet[v]
                                   == False.
                min = dist[v]
                min_index = v
```

07/12/2020 CN LAB KARTHIK VENUGOPAL

WRITE UP 1BM18CS043

```
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 cout in range(self.V):
        u= self.min_distance (dist, sptSet)
        sptSet[u] = True
        for v in range (self.V):
            if self.graph[u][v]> 0
        and sptSet[v] = False and
        dist[v] > dist[u] + self.graph[u][v]
            dist[v]= dist[u]+ self.graph[u]
                                    [v]
    self.print_solution (dist)

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

07/12/2020 CN LAB KARTHIK VENUGOPAL

WRITE UP 1BM18CS047

```
for i in range(e):
    src, dest, cost = [int(_) for _ in
        input(" Enter [SRC] [DEST] [WEIGHT]: ")
            .split(' ')]

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

g.dijstra(sr)
```