CN lab8

Dijkstra's algorithm to compute shortest path through a graph

```
class Network():

    def __init__(self, nodes):
        self.V = nodes
        self.graph = [[0 for column in range(nodes)]
                      for row in range(nodes)]

    def printTable(self, dist, src, path):
        print("Shortest Path Table of {}".format(chr(ord('A') + src)))
        for node in range(self.V):
            print("{0}\t{1}\t{2}".format(chr(ord('A') + node),
                                         dist[node], path[node]))

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

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

Kshitij R
1BM18CS048

```
for cout in range(self.V):
    u = self.minDistance(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]
            if u == src:
                path[v].append(chr(ord('A') + v))
            else:
                path[v].append(chr(ord('A') + u))
                path[v].append(chr(ord('A') + v))

self.printTable(dist, src, path)
```