Computer Networks

Dijkstra's Shortest path Algorithm

Given a graph and a source vertex in a graph, find shortest paths from source to all ~~ver~~ other points in the given graph

We maintain two sets, one set contains all the points included in shortest path tree and other set includes points not yet included in shortest path tree
At every step of algorithm, we find a vertex which is in the other set (not yet included) and has a minimum distance from the source

```
import sys        #Library for INT_MAX

class Graph():
        def __init__(self, points):
                self.V= points
                self.graph= [ [0 for column in range(points)]
                                for row in range(points)]
```

```
def printSolution (self, dist):
    for node in range (self.V):
        print node, "\t", dist[node]


    # A utility function to find vertex with
    # minimum distance value, from set of points
    # not yet included in shortest path tree
    def minDistance (self, dist, sptSet):
        min = sys.maxint
        for v in range (self.V):
            if dist[v] < min and sptSet[v] == false:
                min = dist [v]
                min index = v.
        return min_index


    # Function that implements Dijkstra's algorithm
    #   for adjacency matrix
    def dijkstra (self, src):
        dist = [sys.maxint] * self.V
        dist[src] = 0
        sptSet = [False] * self.V
        for count 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]
    self.printSolution( dist)
```