Find Shortest Path
   // Find the shortest path from some chosen node of a symmetric
   // network graph to each other node of the graph

Input:
   Number of vertices, N
   Chosen start point, S
   Table of weights, Wht[N][N] from each vertex to each other vertex
   (for convenience assume that Wht is a 1-based array)

Output:
   Two-column table, including the shortest Distance, and Path required
   to meet that distance from the S to each vertex, V

Data Structure:
   Table of N rows, one per vertex, and 4 columns, vertex number (int),
   vertexHasBeenVisited (boolean), Distance (int), Preceding vertex (int)

Initialization:
   Table, T, includes one row per vertex with values, vertex #, False,
   infinite, 0

T[S][vertexHasBeenVisited] = True
T[S][D] = 0
T[S][P] = 0

do N-1 times
{
   // find as yet unvisited node Y, such that T[Y][D] is minimum

   pick nodes X,Y such that T[X][vertexHasBeenVisited] == True, T[Y][vertexHasBeenVisited] == False and T[X][D] + Wht[X][Y]
   is a minimum.

   T[Y][vertexHasBeenVisited] = True
   T[Y][D] = T[X][D] + Wht[X][Y]
   T[Y][P] = X
}

// Build output two-column table OT, with N rows, and columns for
// Distance (int), and Path (list of ints) from S

For i from 1 to N inclusive
{
   OT[i][D] = T[i][D]
   j = i
   while T[j][P] != 0 do
   {
      insert P at FRONT of OT[i][P]
      j = T[j][P]
   }
}